

Linguistics

Part 1 - Theoretical linguistics

Contents

Articles

Introduction	1
Linguistics	1
Language	10
Natural language	39
Theoretical linguistics	43
Theoretical linguistics	43
Psycholinguistics	46
Sociolinguistics	51
Linguistic universal	57
Grammar	60
Grammar	60
English grammar	64
Clause	79
Phrase	87
Word	90
Morphology (linguistics)	94
Syntax	100
Phonology	104
Morphophonology	110
Phonetics	113
Semantics	116
Pragmatics	121
Formal grammar	128
Formal grammar	128
Formal language	133
Formation rule	139
String (computer science)	140
Formal semantics (logic)	146
Finite-state machine	147
Automata theory	159
Principle of compositionality	164

Parse tree	165
Deep structure	167
Ambiguous grammar	168
Phrase structure grammar	170
Chomsky hierarchy	173
Chomsky hierarchy	173
Unrestricted grammar	176
Turing machine	177
Recursively enumerable language	193
Recursive language	195
Machine that always halts	196
Context-sensitive grammar	198
Context-sensitive language	200
Context-free grammar	201
Context-free language	213
Pushdown automaton	215
Regular grammar	220
Regular language	222
Regular expression	226
Generative grammar	241
Generative grammar	241
Transformational grammar	244
Principles and parameters	249
Government and binding theory	253
Minimalist program	256
Relational grammar	261
Lexical functional grammar	262
Generalized phrase structure grammar	264
Head-driven phrase structure grammar	265
Categorial grammar	269
Tree-adjoining grammar	274
Nanosyntax	276
Arc pair grammar	276
Generative semantics	277
Dependency grammar	280

Dependency grammar	280
Recursive categorical syntax	291
Operator grammar	291
Functional generative description	293
Meaning–text theory	294
Word grammar	298
Link grammar	299
Quranic Arabic Corpus	302
Functional theories of grammar	304
Functional discourse grammar	304
Prague school	306
Systemic functional grammar	309
Cognitive grammar	313
Construction grammar	314
Role and reference grammar	319
Emergent grammar	320
References	
Article Sources and Contributors	321
Image Sources, Licenses and Contributors	329
Article Licenses	
License	331

Introduction

Linguistics

Linguistics is the scientific study of human language.^{[1][2][3][4][5]} Linguistics can be broadly broken into three categories or subfields of study: language form, language meaning, and language in context. The earliest known activities in descriptive linguistics have been attributed to Panini around 500 BCE, with his analysis of Sanskrit in *Ashtadhyayi*.^[6]

The first subfield of linguistics is the study of language structure, or grammar. This focuses on the system of rules followed by the users of a language. It includes the study of morphology (the formation and composition of words), syntax (the formation and composition of phrases and sentences from these words), and phonology (sound systems). Phonetics is a related branch of linguistics concerned with the actual properties of speech sounds and nonspeech sounds, and how they are produced and perceived.

The study of language meaning is concerned with how languages employ logical structures and real-world references to convey, process, and assign meaning, as well as to manage and resolve ambiguity. This category includes the study of semantics (how meaning is inferred from words and concepts) and pragmatics (how meaning is inferred from context).

Linguistics also looks at the broader context in which language is influenced by social, cultural, historical and political factors. This includes the study of evolutionary linguistics, which investigates into questions related to the origins and growth of languages; historical linguistics, which explores language change; sociolinguistics, which looks at the relation between linguistic variation and social structures; psycholinguistics, which explores the representation and function of language in the mind; neurolinguistics, which looks at language processing in the brain; language acquisition, on how children or adults acquire language; and discourse analysis, which involves the structure of texts and conversations.

Although linguistics is the scientific study of language, a number of other intellectual disciplines are relevant to language and intersect with it. Semiotics, for example, is the general study of signs and symbols both within language and without. Literary theorists study the use of language in literature. Linguistics additionally draws on and informs work from such diverse fields as acoustics, anthropology, biology, computer science, human anatomy, informatics, neuroscience, philosophy, psychology, sociology, and speech-language pathology.

Terminology

Before the 20th century, the term *philology*, first attested in 1716,^[7] was commonly used to refer to the science of language, which was then predominantly historical in focus.^[8] Since Ferdinand de Saussure's insistence on the importance of synchronic analysis, however, this focus has shifted^[9] and the term "philology" is now generally used for the "study of a language's grammar, history, and literary tradition", especially in the United States,^[10] where it was never as popular as it was elsewhere (in the sense of the "science of language").^[7]

Although the term "linguist" in the sense of "a student of language" dates from 1641,^[11] the term "linguistics" is first attested in 1847.^[11] It is now the usual academic term in English for the scientific study of language.

The term *linguist* applies within the field to someone who studies language, or specific languages. Outside the field, this term is commonly used to refer to people who speak many languages fluently.^[12]

Fundamental questions

Linguistics concerns itself with describing and explaining the nature of human language. Fundamental questions include what is universal to language, how language can vary, and how human beings come to know languages. Linguistic research can broadly be divided into the descriptive analysis of structure and grammar on the one hand and the study of non-linguistic influences on language on the other.

Formal and functional approaches

One major debate in linguistics concerns how language should be defined and understood. One prominent group of linguists use the term "language" primarily to refer to a hypothesised, innate module in the human brain that allows people to undertake linguistic behaviour. This "Universal grammar" is considered to guide children when they learn languages and to constrain what sentences are considered grammatical in any language. Proponents of this view, which is predominant in those schools of linguistics that are based on the generative theory of Noam Chomsky, do not necessarily consider that language evolved for communication in particular. They consider instead that it has more to do with the process of structuring human thought (see also formal grammar).

Another group of linguists, by contrast, use the term "language" to refer to a communication system that developed to support cooperative activity and extend cooperative networks. Such functional theories of grammar view language as a tool that is adapted to the communicative needs of its users, and the role of cultural evolutionary processes are often emphasised over that of biological evolution.

Variation and universality

While some theories on linguistics focus on the different varieties that language produces, among different sections of society, others focus on the universal properties that are common to all given languages at one given time on the planet. The theory of variation therefore would elaborate on the different usages of popular languages like French and English across the globe, as well as its smaller dialects and regional permutations within their national boundaries. The theory of variation looks at the cultural stages that a particular language undergoes, and these include the following. The first stage is pidgin, or that phase in the creation of a language's variation when new, non-native speakers undertake a mainstream language and use its phrases and words in a broken manner that often attempts to be overly literal in meaning. At this junction, many of the linguistic characteristics of the native speakers' own language or mother tongue influence their use of the mainstream language, and that is when it arrives at the stage of being called a creole. Hence, this process in the creation of dialects and varieties of languages as globally popular as English and French, as well as others like Spanish, for instance, is one that is rooted in the changing evolution and growth of each language. These variating factors are studied in order to understand the different usages and dialects that a language develops over time.

Universality, on the other hand, looks at formal structures and features that are common to all languages, and the template of which pre-exists in the mind of an infant child. This idea is based on the theory of generative grammar and the formal school of linguistics, whose proponents include Noam Chomsky and those who follow his theory and work.

Schools of thought

Early grammarians

The formal study of language began in India with Pāṇini, the 5th century BC grammarian who formulated 3,959 rules of Sanskrit morphology. Pāṇini's systematic classification of the sounds of Sanskrit into consonants and vowels, and word classes, such as nouns and verbs, was the first known instance of its kind. In the Middle East Sibawayh (سيبوه) made a detailed description of Arabic in 760 AD in his monumental work, *Al-kitab fi al-nahw* (الكتاب في النحو, *The Book on Grammar*), the first known author to distinguish between sounds and phonemes (sounds as units of a linguistic system).



Ancient Tamil inscription at Thanjavur

Western interest in the study of languages began as early as in the East,^[13] but the grammarians of the classical languages did not use the same methods or reach the same conclusions as their contemporaries in the Indic world. Early interest in language in the West was a part of philosophy, not of grammatical description. The first insights into semantic theory were made by Plato in his *Cratylus* dialogue, where he argues that words denote concepts that are eternal and exist in the world of ideas. This work is the first to use the word etymology to describe the history of a word's meaning. Around 280 BC one of Alexander the Great's successors founded a university (see Musaeum) in Alexandria, where a school of philologists studied the ancient texts in and taught Greek to speakers of other languages. This school was the first to use the word "grammar" in its modern sense, Plato had used the word in its original meaning as "téchnē grammatiskē" (Τέχνη Γραμματική), the "art of writing," which is also the title of one of the most important works of the Alexandrine school by Dionysius Thrax.^[14] Throughout the Middle Ages the study of language was subsumed under the topic of philology, the study of ancient languages and texts, practiced by such educators as Roger Ascham, Wolfgang Ratke and John Amos Comenius.^[15]

Historicism

In the 18th century, the first use of the comparative method by William Jones sparked the rise of comparative linguistics.^[16] Bloomfield attributes "the first great scientific linguistic work of the world" to Jacob Grimm, who wrote *Deutsche Grammatik*.^[17] It was soon followed by other authors writing similar comparative studies on other language groups of Europe. The scientific study of language was broadened from Indo-European to language in general by Wilhelm von Humboldt, of whom Bloomfield asserts:^[17]

"This study received its foundation at the hands of the Prussian statesman and scholar Wilhelm von Humboldt (1767—1835), especially in the first volume of his work on Kavi, the literary language of Java, entitled *Über die Verschiedenheit des menschlichen Sprachbaues und ihren Einfluß auf die geistige Entwicklung des Menschengeschlechts* ('On the Variety of the Structure of Human Language and its Influence upon the Mental Development of the Human Race')."

Structuralism

Early in the 20th century, Saussure introduced the idea of language as a static system of interconnected units, defined through the oppositions between them. By introducing a distinction between diachronic to synchronic analyses of language, he laid the foundation of the modern discipline of linguistics. Saussure also introduced several basic dimensions of linguistic analysis that are still foundational in many contemporary linguistic theories, such as the distinctions between syntagm and paradigm, and the langue- parole distinction, distinguishing language as an abstract system (*langue*) from language as a concrete manifestation of this system (*parole*).^[18] Substantial additional

contributions following Saussure's definition of a structural approach to language came from The Prague school, Leonard Bloomfield, Charles F. Hockett, Louis Hjelmslev, Émile Benveniste and Roman Jakobson.^[19]

Generativism

During the last half of the 20th century, following the work of Noam Chomsky, linguistics was dominated by the generativist school. While formulated by Chomsky in part as a way to explain how human beings acquire language and the biological constraints on this acquisition, in practice it has largely been concerned with giving formal accounts of specific phenomena in natural languages. Generative theory is modularist and formalist in character. Chomsky built on earlier work of Zellig Harris to formulate the generative theory of language. According to this theory the most basic form of language is a set of syntactic rules universal for all humans and underlying the grammars of all human languages. This set of rules is called Universal Grammar, and for Chomsky describing it is the primary objective of the discipline of linguistics. For this reason the grammars of individual languages are of importance to linguistics only in so far as they allow us to discern the universal underlying rules from which the observable linguistic variability is generated.

In the classic formalization of generative grammars first proposed by Noam Chomsky in the 1950s,^{[20][21]} a grammar G consists of the following components:

- A finite set N of *nonterminal symbols*, none of which appear in strings formed from G .
- A finite set Σ of *terminal symbols* that is disjoint from N .
- A finite set P of *production rules*, that map from one string of symbols to another.

A formal description of language attempts to replicate a speaker's knowledge of the rules of their language, and the aim is to produce a set of rules that is minimally sufficient to successfully model valid linguistic forms.

Functionalism

Functional theories of language propose that since language is fundamentally a tool, it is reasonable to assume that its structures are best analyzed and understood with reference to the functions they carry out. Functional theories of grammar differs from formal theories of grammar, in that the latter seeks to define the different elements of language and describe the way they relate to each other as systems of formal rules or operations, whereas the former defines the functions performed by language and then relates these functions to the linguistic elements that carry them out. This means that functional theories of grammar tend to pay attention to the way language is actually used, and not just to the formal relations between linguistic elements.^[22]

Functional theories then describe language in term of functions existing on all levels of language.

- Phonological function: the function of the phoneme is to distinguish between different lexical material.
- Semantic function: (Agent, Patient, Recipient, etc.), describing the role of participants in states of affairs or actions expressed.
- Syntactic functions: (e.g. subject and Object), defining different perspectives in the presentation of a linguistic expression
- Pragmatic functions: (Theme and Rheme, Topic and Focus, Predicate), defining the informational status of constituents, determined by the pragmatic context of the verbal interaction. Functional descriptions of grammar strive to explain how linguistic functions are performed in communication through the use of linguistic forms.

Cognitive linguistics

In the 1970s and 1980s, a new school of thought known as cognitive linguistics emerged as a reaction to generativist theory. Led by theorists such as Ronald Langacker and George Lakoff, linguists working within the realm of cognitive linguistics propose that language is an emergent property of basic, general-purpose cognitive processes. In contrast to the generativist school of linguistics, cognitive linguistics is non-modularist and functionalist in character. Important developments in cognitive linguistics include cognitive grammar, frame semantics, and conceptual metaphor, all of which are based on the idea that form-function correspondences based on representations derived from embodied experience constitute the basic units of language.

Cognitive linguistics interprets language in terms of the concepts, sometimes universal, sometimes specific to a particular tongue, which underlie its forms. It is thus closely associated with semantics but is distinct from psycholinguistics, which draws upon empirical findings from cognitive psychology in order to explain the mental processes that underlie the acquisition, storage, production and understanding of speech and writing. Cognitive linguistics denies that there is an *autonomous linguistic faculty* in the mind; it understands grammar in terms of *conceptualization*; and it claims that knowledge of language arises out of *language use*.^[23] Because of its conviction that knowledge of language is learned through use, cognitive linguistics is sometimes considered to be a functional approach, but it differs from other functional approaches in that it is primarily concerned with how the mind creates meaning through language, and not with the use of language as a tool of communication.

Sub-disciplines

Linguistic structures

Linguistic structures are pairings of meaning and form. Any particular pairing of meaning and form is a Saussurean sign. For instance, the meaning "cat" is represented worldwide with a wide variety of different sound patterns (in oral languages), movements of the hands and face (in sign languages), and written symbols (in written languages).

Linguists focusing on structure attempt to understand the rules regarding language use that native speakers know (not always consciously). All linguistic structures can be broken down into component parts that are combined according to (sub)conscious rules, over multiple levels of analysis. For instance, consider the structure of the word "tenth" on two different levels of analysis. On the level of internal word structure (known as morphology), the word "tenth" is made up of one linguistic form indicating a number and another form indicating ordinality. The rule governing the combination of these forms ensures that the ordinality marker "th" follows the number "ten." On the level of sound structure (known as phonology), structural analysis shows that the "n" sound in "tenth" is made differently from the "n" sound in "ten" spoken alone. Although most speakers of English are consciously aware of the rules governing internal structure of the word pieces of "tenth", they are less often aware of the rule governing its sound structure. Linguists focused on structure find and analyze rules such as these, which govern how native speakers use language.

Linguistics has many sub-fields concerned with particular aspects of linguistic structure. These sub-fields range from those focused primarily on form to those focused primarily on meaning. They also run the gamut of level of analysis of language, from individual sounds, to words, to phrases, up to discourse.

Sub-fields that focus on a structure-focused study of language:

- **Phonetics**, the study of the physical properties of speech (or signed) production and perception.
- **Phonology**, the study of sounds (or signs) as discrete, abstract elements in the speaker's mind that distinguish meaning (phonemes).
- **Morphology**, the study of morphemes, or the internal structures of words and how they can be modified
- **Syntax**, the study of how words combine to form grammatical sentences
- **Semantics**, the study of the meaning of words (lexical semantics) and fixed word combinations (phraseology), and how these combine to form the meanings of sentences

- **Pragmatics**, the study of how utterances are used in communicative acts, and the role played by context and non-linguistic knowledge in the transmission of meaning
- **Discourse analysis**, the analysis of language use in texts (spoken, written, or signed)
- **Stylistics**, the study of linguistic factors (rhetoric, diction, stress) that place a discourse in context.
- **Semiotics**, the study of signs and sign processes (semiosis), indication, designation, likeness, analogy, metaphor, symbolism, signification, and communication.

Many linguists would agree that these divisions overlap considerably, and the independent significance of each of these areas is not universally acknowledged. Regardless of any particular linguist's position, each area has core concepts that foster significant scholarly inquiry and research.

Inter-disciplinary factors

Alongside the structurally motivated domains of study, are other fields within the domain of linguistics. These fields are often distinguished by external factors that influence the study of language.

- Applied linguistics, the study of language-related issues applied in everyday life, notably language policies, planning, and education. (Constructed language fits under Applied linguistics.)
- Biolinguistics, the study of natural as well as human-taught communication systems in animals, compared to human language.
- Clinical linguistics, the application of linguistic theory to the field of Speech-Language Pathology.
- Computational linguistics, the study of linguistic issues in a way that is 'computationally responsible', i.e., taking careful note of computational consideration of algorithmic specification and computational complexity, so that the linguistic theories devised can be shown to exhibit certain desirable computational properties implementations.
- Developmental linguistics, the study of the development of linguistic ability in individuals, particularly the acquisition of language in childhood.
- Evolutionary linguistics, the study of the origin and subsequent development of language by the human species.
- Historical linguistics or diachronic linguistics, the study of language change over time.
- Language geography, the study of the geographical distribution of languages and linguistic features.
- Linguistic typology, the study of the common properties of diverse unrelated languages, properties that may, given sufficient attestation, be assumed to be innate to human language capacity.
- Neurolinguistics, the study of the structures in the human brain that underlie grammar and communication.
- Psycholinguistics, the study of the cognitive processes and representations underlying language use.
- Sociolinguistics, the study of variation in language and its relationship with social factors.

Semiotics is a larger discipline that investigates the relationship between signs and what they signify more broadly. From the perspective of semiotics, language can be seen as a sign or symbol, with the world as its representation.

Sub-fields

Historical linguistics

Historical linguists study the history of specific languages as well as general characteristics of language change. One aim of historical linguistics is to classify languages in language families descending from a common ancestor, an enterprise that relies primarily on the comparative method. This involves comparison of elements in different languages to detect possible cognates in order to be able to reconstruct how different languages have changed over time. Some historical linguists, along with non-linguists interested in language change, have also employed such tools as computational phylogenetics. The study of language change is also referred to as "diachronic linguistics", which can be distinguished from "synchronic linguistics", the study of a given language at a given moment in time without regard to its previous stages. Historical linguistics was among the first linguistic disciplines to emerge and was the most widely practised form of linguistics in the late 19th century. However, a shift in focus to the synchronic

perspective began in the early twentieth century with Saussure and became predominant in western linguistics through the work of Noam Chomsky.

Semiotics

Semiotics is the study of sign processes (semiosis), or signification and communication, signs, and symbols, both individually and grouped into sign systems, including the study of how meaning is constructed and understood. Semioticians often do not restrict themselves to linguistic communication when studying the use of signs but extend the meaning of "sign" to cover all kinds of cultural symbols. Nonetheless, semiotic disciplines closely related to linguistics are literary studies, discourse analysis, text linguistics, and philosophy of language. Semiotics, within the linguistics paradigm, is the study of the relationship between language and culture. Historically, Edward Sapir and Ferdinand De Saussure's structuralist theories influenced the study of signs extensively until the late part of the 20th century, but later, post-modern and post-structural thought, through language philosophers including Jacques Derrida, Mikhail Bakhtin, Michel Foucault, and others, have also been a considerable influence on the discipline in the late part of the 20th century and early 21st century. These theories emphasise the role of language variation, and the idea of subjective usage, depending on external elements like social and cultural factors, rather than merely on the interplay of formal elements.

Language documentation

Since the inception of the discipline of linguistics, linguists have been concerned with describing and analysing previously undocumented languages. Starting with Franz Boas in the early 1900s, this became the main focus of American linguistics until the rise of formal structural linguistics in the mid-20th century. This focus on language documentation was partly motivated by a concern to document the rapidly disappearing languages of indigenous peoples. The ethnographic dimension of the Boasian approach to language description played a role in the development of disciplines such as sociolinguistics, anthropological linguistics, and linguistic anthropology, which investigate the relations between language, culture, and society.

The emphasis on linguistic description and documentation has also gained prominence outside North America, with the documentation of rapidly dying indigenous languages becoming a primary focus in many university programs in linguistics. Language description is a work-intensive endeavour, usually requiring years of field work in the language concerned, so as to equip the linguist to write a sufficiently accurate reference grammar. Further, the task of documentation requires the linguist to collect a substantial corpus in the language in question, consisting of texts and recordings, both sound and video, which can be stored in an accessible format within open repositories, and used for further research.^[24]

Applied linguistics

Linguists are largely concerned with finding and describing the generalities and varieties both within particular languages and among all languages. Applied linguistics takes the results of those findings and "applies" them to other areas. Linguistic research is commonly applied to areas such as language education, lexicography, and translation. "Applied linguistics" has been argued to be something of a misnomer, since applied linguists focus on making sense of and engineering solutions for real-world linguistic problems, not simply "applying" existing technical knowledge from linguistics; moreover, they commonly apply technical knowledge from multiple sources, such as sociology (e.g., conversation analysis) and anthropology.

Today, computers are widely used in many areas of applied linguistics. Speech synthesis and speech recognition use phonetic and phonemic knowledge to provide voice interfaces to computers. Applications of computational linguistics in machine translation, computer-assisted translation, and natural language processing are areas of applied linguistics that have come to the forefront. Their influence has had an effect on theories of syntax and semantics, as modeling syntactic and semantic theories on computers constraints.

Linguistic analysis is a sub-discipline of applied linguistics used by many governments to verify the claimed nationality of people seeking asylum who do not hold the necessary documentation to prove their claim.^[25] This often takes the form of an interview by personnel in an immigration department. Depending on the country, this interview is conducted either in the asylum seeker's native language through an interpreter or in an international lingua franca like English.^[25] Australia uses the former method, while Germany employs the latter; the Netherlands uses either method depending on the languages involved.^[25] Tape recordings of the interview then undergo language analysis, which can be done either by private contractors or within a department of the government. In this analysis, linguistic features of the asylum seeker are used by analysts to make a determination about the speaker's nationality. The reported findings of the linguistic analysis can play a critical role in the government's decision on the refugee status of the asylum seeker.^[25]

Translation

The sub-field of translation includes the translation of written and spoken texts across mediums, from digital to print and spoken. To translate literally means to transmute the meaning from one language into another. Translators are often employed by organisations, such as travel agencies as well as governmental embassies to facilitate communication between two speakers who do not know each other's language. Translators are also employed to work within computational linguistics setups like Google Translate for example, which is an automated, programmed facility to translate words and phrases between any two or more given languages. Translation is also conducted by publishing houses, who convert works of writing from one language to another in order to reach varied audiences.

Description and prescription

Linguistics is descriptive; linguists describe and explain features of language without making subjective judgments on whether a particular feature is "right" or "wrong". This is analogous to practice in other sciences: A zoologist studies the animal kingdom without making subjective judgments on whether a particular animal is better or worse than another.

Prescription, on the other hand, is an attempt to promote particular linguistic usages over others, often favouring a particular dialect or "acrolect". This may have the aim of establishing a linguistic standard, which can aid communication over large geographical areas. It may also, however, be an attempt by speakers of one language or dialect to exert influence over speakers of other languages or dialects (see Linguistic imperialism). An extreme version of prescriptivism can be found among censors, who attempt to eradicate words and structures that they consider to be destructive to society.

Speech and writing

Most contemporary linguists work under the assumption that spoken language is more fundamental than written language. This is because:

- Speech appears to be universal to all human beings capable of producing and hearing it, while there have been many cultures and speech communities that lack written communication
- Speech evolved before human beings invented writing
- People learn to speak and process spoken language more easily and much earlier than writing.

Nonetheless, linguists agree that the study of written language can be worthwhile and valuable. For research that relies on corpus linguistics and computational linguistics, written language is often much more convenient for processing large amounts of linguistic data. Large corpora of spoken language are difficult to create and hard to find, and are typically transcribed and written. In addition, linguists have turned to text-based discourse occurring in various formats of computer-mediated communication as a viable site for linguistic inquiry.

The study of writing systems themselves is, in any case, considered a branch of linguistics.

References

- [1] *Linguistics* (<http://mitpress.mit.edu/catalog/item/examrequest.asp?ttype=2&tid=12240>) (6th ed.). The MIT Press. 2010. ISBN 0-262-51370-6. . Retrieved 25 July 2012.
- [2] . ISBN 0-262-51370-6.
- [3] Martinet, André (1960). *Elements of General Linguistics*. Tr. Elisabeth Palmer (Studies in General Linguistics, vol. i.). London: Faber. p. 15.
- [4] Halliday, Michael A. K.; Jonathan Webster (2006). *On Language and Linguistics*. Continuum International Publishing Group. p. vii. ISBN 0-8264-8824-2.
- [5] Greenberg, Joseph (1948). "Linguistics and ethnology". *Southwestern Journal of Anthropology* **4**: 140–47.
- [6] Vasu, S.C. (1996). *The Ashtadhyayi of Panini* (2 Vols.) (<http://www.vedicbooks.net/ashtadhyayi-panini-vols-p-2313.html>). Vedic Books. ISBN 9788120804098. .
- [7] Online Etymological Dictionary: philology (<http://www.etymonline.com/index.php?term=philology>)
- [8] McMahon, A. M. S. (1994). *Understanding Language Change*. Cambridge University Press. p. 19. ISBN 0-521-44665-1
- [9] McMahon, A. M. S. (1994). *Understanding Language Change*. Cambridge University Press. p. 9. ISBN 0-521-44665-1
- [10] A. Morpurgo Davies Hist. Linguistics (1998) 4 I. 22.
- [11] Online Etymological Dictionary: linguist (<http://www.etymonline.com/index.php?term=linguist>)
- [12] "Linguist". *The American Heritage Dictionary of the English Language*. Houghton Mifflin Harcourt. 2000. ISBN 978-0-395-82517-4.
- [13] Bloomfield 1914, p. 307.
- [14] Seuren, Pieter A. M. (1998). *Western linguistics: An historical introduction*. Wiley-blackwell. pp. 2–24. ISBN 0-631-20891-7.
- [15] Bloomfield 1914, p. 308.
- [16] Bloomfield 1914, p. 310.
- [17] Bloomfield 1914, p. 311.
- [18] Clarke, David S. (1990). *Sources of semiotic: readings with commentary from antiquity to the present*. Carbondale: Southern Illinois University Press. pp. 143–144.
- [19] Holquist 1981, pp. xvii-xviii.
- [20] Chomsky, Noam (1956). "Three Models for the Description of Language". *IRE Transactions on Information Theory* **2** (2): 113–123. doi:10.1109/TIT.1956.1056813.
- [21] Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton.
- [22] Nichols, Johanna (1984). "Functional Theories of Grammar". *Annual Review of Anthropology* **13**: 97. doi:10.1146/annurev.an.13.100184.000525. "[Functional grammar] analyzes grammatical structure, as do formal and structural grammar; but it also analyzes the entire communicative situation: the purpose of the speech event, its participants, its discourse context. Functionalists maintain that the communicative situation motivates, constrains, explains, or otherwise determines grammatical structure, and that a structural or formal approaches not merely limited to an artificially restricted data base, but is inadequate even as a structural account. Functional grammar, then, differs from formal and structural grammar in that it purports not to model but to explain; and the explanation is grounded in the communicative situation."
- [23] Croft, William and D. Alan Cruse (2004). *Cognitive Linguistics*. Cambridge: Cambridge University Press. p. 1.
- [24] Himmelmann, Nikolaus Language documentation: What is it and what is it good for? in P. Gippert, Jost, Nikolaus P Himmelmann & Ulrike Mosel. (2006) Essentials of Language documentation. Mouton de Gruyter, Berlin & New York.
- [25] Eades, Diana (2005). "Applied Linguistics and Language Analysis in Asylum Seeker Cases" (<http://songchau.googlepages.com/503.pdf>). *Applied Linguistics* **26** (4): 503–526. doi:10.1093/applin/ami021. .

Bibliography

- Akmajian, Adrian; Demers, Richard; Farmer, Ann; Harnish, Robert (2010). *Linguistics: An Introduction to Language and Communication*. Cambridge, MA: The MIT Press. ISBN 0-262-51370-6.

External links

- The Linguist List (<http://linguistlist.org/>), a global online linguistics community with news and information updated daily
- Glossary of linguistic terms (<http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/index.htm>)
- Language Log (<http://languagelog.ldc.upenn.edu/nll/>), a linguistics blog maintained by prominent linguists
- Glottopedia (<http://www.glottopedia.org>), MediaWiki-based encyclopedia of linguistics, under construction

- Linguistic sub-fields (<http://www.lsadc.org/info/ling-fields.cfm>) – according to the Linguistic Society of America
- Linguistics and language-related wiki articles on Scholarpedia (<http://www.scholarpedia.org/article/Language>) and Citizendum (<http://en.citizendum.org/wiki/Linguistics>)
- "Linguistics" section (http://www.unizar.es/departamentos/filologia_inglesa/garciala/bibliography.html) – A Bibliography of Literary Theory, Criticism and Philology, ed. J. A. García Landa (University of Zaragoza, Spain)
- An Academic Linguistics (<http://www.lingforum.com/forum>) Forum
- Linguistics (http://www.dmoz.org/Science/Social_Sciences/Linguistics/) at the Open Directory Project

Language



A mural in Teotihuacan, Mexico (ca. 200 AD) depicting a person emitting a speech scroll from his mouth, symbolizing speech.



Cuneiform is the first known form of written language, but spoken language predates writing by at least tens of thousands of years.



Two girls learning American Sign Language.



Braille writing represents language in a tactile form.

Language is the human capacity for acquiring and using complex systems of communication, and a **language** is any specific example of such a system. The scientific study of language is called linguistics. It is impossible to know precisely how many languages there are in the world, and the number depends on a partly arbitrary distinction between languages and dialects. However, estimates vary between around 6,000 and 7,000 languages in number. Natural languages are spoken or signed, but any language can be encoded into secondary media using auditory, visual or tactile stimuli, for example in graphic writing, braille, or whistling. This is because human language is modality-independent. When used as a general concept, "language" may refer to the cognitive ability to learn and use systems of complex communication, or to describe the set of rules that makes up these systems, or the set of utterances that can be produced from those rules.

Human language is unique among the lifeforms of Earth because its complex structure affords a much wider range of possible expressions and uses than any known system of animal communication, all of which are generally closed systems, with limited functions and mostly genetically rather than socially transmitted. In contrast to non-human communication forms, human language has the properties of productivity, recursivity, and displacement. Human language is also the only system to rely mostly on social convention and learning. Language is thought to have originated when early hominins first started cooperating, gradually changing their primate communication systems as they acquired the ability to form a theory of other minds and shared intentionality.

This development is thought to have coincided with an increase in brain volume, and many linguists see the structures of language as having evolved to serve specific communicative functions. Language is processed in many different locations in the human brain, but especially in Broca's and Wernicke's areas. Humans acquire language through social interaction in early childhood, and children generally speak fluently when they are around three years old. The use of language is deeply entrenched in human culture. Therefore, in addition to its strictly communicative uses, language also has many social and cultural uses, such as signifying group identity, social stratification, as well as for social grooming and entertainment.

All languages rely on the process of semiosis to relate signs with particular meanings. Oral and sign languages contain a phonological system that governs how symbols are used to form sequences known as words or morphemes, and a syntactic system that governs how words and morphemes are combined to form phrases and utterances. Languages evolve and diversify over time, and the history of their evolution can be reconstructed by comparing modern languages to determine which traits their ancestral languages must have had for the later stages to have occurred. A group of languages that descend from a common ancestor is known as a language family. The languages that are most spoken in the world today belong to the Indo-European family, which includes languages such as English, Spanish, Portuguese, Russian and Hindi; the Sino-Tibetan languages, which include Mandarin Chinese, Cantonese and many others; Semitic languages, which include Arabic, Amharic and Hebrew; and the Bantu languages, which include Swahili, Zulu, Shona and hundreds of other languages spoken throughout Africa. The general consensus is that between 50 to 90% of languages spoken today will probably have become extinct by the year 2100.^{[1][2]}

Definitions

The English word "language" derives ultimately from Indo-European *d̥nǵʰwéh₂s* "tongue, speech, language" through Latin *lingua*, "language, tongue", and Old French *langage* "language".^[3] The word is sometimes used to refer to codes, ciphers and other kinds of artificially constructed communication systems such as those used for computer programming. A language in this sense is a system of signs for encoding and decoding information. This article is specifically about the properties of natural human language as it is studied in the discipline of linguistics.

As an object of linguistic study "language" has two primary meanings: language as an abstract concept, and "a language" (a specific linguistic system, e.g. "French"). The Swiss linguist Ferdinand de Saussure, who defined the modern discipline of linguistics, first explicitly formulated the distinction, using the French word *langage* for language as a concept, and *langue* as a specific instance of a language system, and *parole* for concrete usage of

speech in a particular language.^[4]

When speaking of language as a general concept, some different definitions can be used that stress different aspects of the phenomenon.^[5] These definitions also entail different approaches and understandings of language, and they inform different and often incompatible schools of linguistic theory.^[6]

Mental faculty, organ or instinct

One definition sees language primarily as the mental faculty that allows humans to undertake linguistic behaviour: to learn languages and produce and understand utterances. This definition stresses the universality of language to all humans and the biological basis of the human capacity for language as a unique development of the human brain. The view that the drive to language acquisition is innate in humans is supported by the fact that all cognitively normal children raised in an environment where language is accessible will acquire language without formal instruction. Languages may even spontaneously develop in environments where people live or grow up together without a common language, for example in the case of creole languages, and the case of spontaneously developed sign languages such as Nicaraguan Sign Language. This view which can be seen as a view of language going back to Kant and Descartes often understands language to be largely innate, for example as in Chomsky's theory of Universal Grammar or American philosopher Jerry Fodor's extreme innatist theory. These kinds of definitions are often applied by studies of language within a cognitive science framework and in neurolinguistics.^{[7][8]}

Formal symbolic system

Another definition sees language as a formal system of signs governed by grammatical rules of combination to communicate meaning. This definition stresses the fact that human languages can be described as closed structural systems consisting of rules that relate particular signs to particular meanings. This structuralist view of language was first introduced by Ferdinand de Saussure,^[9] and his structuralism remains foundational for most approaches to language today.^[10]

Some proponents of this view of language have advocated a formal approach which studies language structure by identifying its basic elements and then formulating a formal account of the rules according to which the elements combine to form words and sentences. The main proponent of such a theory is Noam Chomsky, the originator of the generative theory of grammar, who has defined language as a particular set of sentences that can be generated from a particular set of rules. Chomsky considers these rules to be an innate feature of the human mind, and to constitute the essence of what language is.^[11] Formal definitions of language is commonly used in formal logic, and in formal theories of grammar and in applied computational linguistics.^{[12][13]}

Tool for communication

Yet another definition sees language as a system of communication that enables humans to cooperate. This definition stresses the social functions of language and the fact that humans use it to express themselves and to manipulate objects in their environment. Functional theories of grammar explain grammatical structures by their communicative functions, and understands the grammatical structures of language to be the result of an adaptive process by which grammar was "tailored" to serve communicative needs of its users.^{[14][15]}

This view of language is associated with the study of language in pragmatic, cognitive and interactional frameworks, as well as in socio-linguistics and linguistic anthropology. Functionalist theories tend to study grammar as a dynamic phenomenon, as structures that are always in the process of changing as they are employed by their speakers. This view places importance on the study of linguistic typology, classification of languages according to structural features, as it can be shown that processes of grammaticalization tend to follow trajectories that are partly dependent on typology. In the philosophy of language these views are often associated with Wittgenstein's later works and with ordinary language philosophers such as Paul Grice, John Searle and J. L. Austin.^[13]



Two men and a woman having a conversation in American Sign Language.

What makes human language unique

Human language is unique in comparison to other forms of communication, such as those used by non-human animals. Communication systems used by other animals such as bees or non-human apes are closed systems that consist of a closed number of possible things that can be expressed.^[16]

In contrast human language is open-ended and productive, meaning that it allows humans to produce an infinite set of utterances from a finite set of elements, and to create new words and sentences. This we can do because human language is based on a dual code, where a finite number of meaningless elements (e.g. sounds, letters or gestures) can be combined to form units of meaning (words and sentences).^[17] Furthermore the symbols and grammatical rules of any particular language are largely arbitrary, meaning that the system can only be acquired through social interaction.^[18] The known systems of communication used by animals, on the other hand, can only express a finite number of utterances that are mostly genetically transmitted.^[19]

Several species of animals have proven able to acquire forms of communication through social learning, such as the Bonobo Kanzi who learned to express himself using a set of symbolic lexigrams. Similarly many species of birds and whales learn their songs by imitating other members of their species. However while some animals may acquire large numbers of words and symbols,^[20] none have been able to learn as many different signs as is generally known by an average 4 year old human, nor have any acquired anything resembling the complex grammar of human language.^[21]

Human languages also differ from animal communication systems in that they employ grammatical and semantic categories such as noun and verb, or present and past, to express exceedingly complex meanings.^[21] Human language is also unique in having the property of recursivity; this is the way in which, for example, a noun phrase is able to contain another noun phrase (as in "[[the chimpanzee's lips]]") or a clause to contain a clause (as in "[I see [the dog is running]]").^[22] Human language is also the only known natural communication system that is *modality independent*, meaning that it can be used not only for communication through one channel or medium, but through

several - for example spoken language uses the auditive modality, whereas sign languages and writing use the visual modality and braille writing uses the tactile modality.^[23]

With regards to the meaning that it may convey and the cognitive operations that it builds on, human language is also unique in being able to refer to abstract concepts and to imagined or hypothetical events, as well as events that took place in the past or may happen in the future. This ability to refer to events that are not at the same time or place as the speech event is called *displacement*, and while some animal communication systems can use displacement (such as the communication of bees that can communicate the location of sources of nectar that are out of sight), the degree to which it is used in human language is also considered unique.^[17]

Origin



75-80,000 year old artefacts from Blombos cave, South Africa including a piece of ochre engraved with diagonal cross-hatch patterns, perhaps the oldest known example of symbols.



"The Tower of Babel" by Pieter Bruegel the Elder. Oil on board, 1563.

Humans have speculated about the origins of language throughout history. The Biblical myth of the Tower of Babel is one such account, other cultures have other stories of how language arose.^[24]

Theories about the origin of language can be divided according to their basic assumptions. Some theories are based on the idea that language is so complex that one can not imagine it simply appearing from nothing in its final form, but that it must have evolved from earlier pre-linguistic systems among our pre-human ancestors. These theories can be called continuity-based theories. The opposite viewpoint is that language is such a unique human trait that it cannot be compared to anything found among non-humans and that it must therefore have appeared fairly suddenly in the transition from pre-hominids to early man. These theories can be defined as discontinuity-based. Similarly, some theories see language mostly as an innate faculty that is largely genetically encoded, while others see it as a system that is largely cultural, that is learned through social interaction.^[25]

Currently the only prominent proponent of a discontinuity-based theory of human language origins is linguist and philosopher Noam Chomsky. Chomsky proposes that 'some random mutation took place, maybe after some strange cosmic ray shower, and it reorganized the brain, implanting a language organ in an otherwise primate brain'. While cautioning against taking this story too literally, Chomsky insists that 'it may be closer to reality than many other fairy tales that are told about evolutionary processes, including language'.^[26]

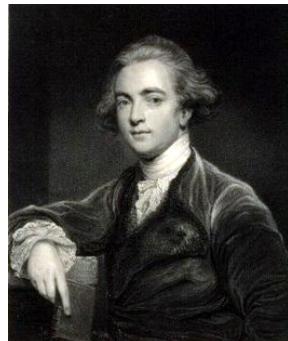
Continuity-based theories are currently held by a majority of scholars, but they vary in how they envision this development. Those who see language as being mostly innate, for example psychologist Steven Pinker, hold the precedents to be animal cognition,^[8] whereas those who see language as a socially learned tool of communication, such as psychologist Michael Tomasello, see it as having developed from animal communication, either primate

gestural or vocal communication to assist in cooperation.^[19] Other continuity-based models see language as having developed from music, a view already espoused by Rousseau, Herder, Humboldt and Charles Darwin. A prominent proponent of this view today is archaeologist Steven Mithen.^[27]

Because the emergence of language is located in the early prehistory of man, the relevant developments have left no direct historical traces and no comparable processes can be observed today. Theories that stress continuity often look at animals to see if, for example, primates display any traits that can be seen as analogous to what pre-human language must have been like. Alternatively, early human fossils can be inspected to look for traces of physical adaptation to language use or for traces of pre-linguistic forms of symbolic behaviour.^[28]

It is mostly undisputed that pre-human australopithecines did not have communication systems significantly different from those found in great apes in general, but scholarly opinions vary as to the developments since the appearance of the genus *Homo* some 2.5 million years ago. Some scholars assume the development of primitive language-like systems (proto-language) as early as *Homo habilis* (2.3 million years ago), while others place the development of primitive symbolic communication only with *Homo erectus* (1.8 million years ago) or *Homo heidelbergensis* (0.6 million years ago) and the development of language proper with *Anatomically Modern Homo sapiens* with the Upper Paleolithic revolution less than 100,000 years ago.^{[29][30]}

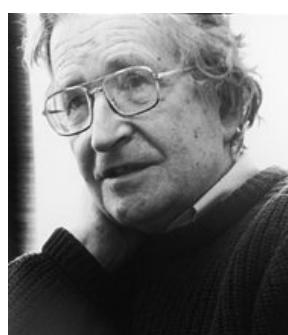
The study of language



William Jones discovered the family relation between Latin and Sanskrit, laying the ground for the discipline of Historical linguistics.



Ferdinand de Saussure developed the structuralist approach to studying language.



Noam Chomsky is one of the most important linguistic theorists of the 20th century.

The study of language, linguistics, has been developing into a science since the first grammatical descriptions of particular languages in India more than 2000 years ago. Today linguistics is a science that concerns itself with all aspects relating to language, examining it from all of the theoretical viewpoints described above.^[31]

Sub-disciplines

The academic study of language is conducted within many different disciplinary areas and from different theoretical angles, all of which inform modern approaches to linguistics. For example, descriptive linguistics examines the grammar of single languages; theoretical linguistics develops theories on how best to conceptualize and define the nature of language, based on data from the various extant human languages; sociolinguistics studies how languages are used for social purposes informing in turn the study of the social functions of language and grammatical description; neurolinguistics studies how language is processed in the human brain, and allows the experimental testing of theories; computational linguistics builds on theoretical and descriptive linguistics to construct computational models of language often aimed at processing natural language, or at testing linguistic hypotheses; and historical linguistics relies on grammatical and lexical descriptions of languages to trace their individual histories and reconstruct trees of language families by using the comparative method.^[32]

Early history

The formal study of language is often considered to have started in India with Pāṇini, the 5th century BC grammarian who formulated 3,959 rules of Sanskrit morphology. However Sumerian scribes already studied the differences between Sumerian and Akkadian grammar around 1900 BC. Subsequent grammatical traditions developed in all of the ancient cultures that adopted writing.^[33]

In the 17th century AD the French Port-Royal Grammarians developed the idea that the grammars of all languages were a reflection of the universal basics of thought, and therefore that grammar was universal. In the 18th century, the first use of the comparative method by British philologist and expert on ancient India William Jones sparked the rise of comparative linguistics.^[34] The scientific study of language was broadened from Indo-European to language in general by Wilhelm von Humboldt. Early in the 20th century, Ferdinand de Saussure introduced the idea of language as a static system of interconnected units, defined through the oppositions between them.^[9]

By introducing a distinction between diachronic and synchronic analyses of language, he laid the foundation of the modern discipline of linguistics. Saussure also introduced several basic dimensions of linguistic analysis that are still fundamental in many contemporary linguistic theories, such as the distinctions between syntagm and paradigm, and the Langue-parole distinction, distinguishing language as an abstract system (*langue*), from language as a concrete manifestation of this system (*parole*).^[35]

Contemporary linguistics

In the 1960s Noam Chomsky formulated the generative theory of language. According to this theory the most basic form of language is a set of syntactic rules that are universal for all humans and which underlies the grammars of all human languages. This set of rules is called Universal Grammar; for Chomsky, describing it is the primary objective of the discipline of linguistics. Thus he considered that the grammars of individual languages are only of importance to linguistics insofar as they allow us to deduce the universal underlying rules from which the observable linguistic variability is generated.^[36]

In opposition to the formal theories of the generative school, functional theories of language propose that since language is fundamentally a tool, its structures are best analyzed and understood by reference to their functions. Formal theories of grammar seek to define the different elements of language and describe the way they relate to each other as systems of formal rules or operations, while functional theories seek to define the functions performed

by language and then relate them to the linguistic elements that carry them out.^{[13][37]} The framework of cognitive linguistics interprets language in terms of the concepts, sometimes universal, sometimes specific to a particular language, which underlie its forms.^[38] Cognitive linguistics is primarily concerned with how the mind creates meaning through language.

Physiological and neural architecture of language and speech

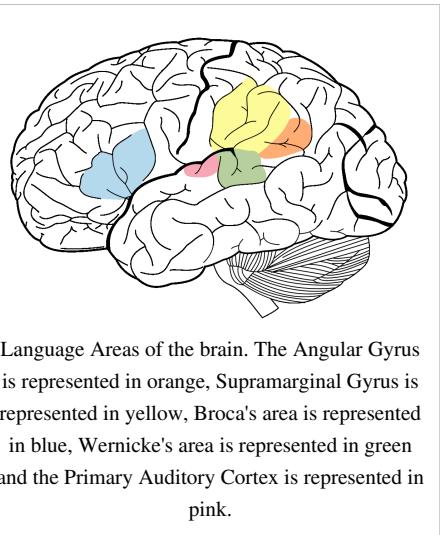
Speaking is the default modality for language in all cultures. The production of spoken language depends on sophisticated capacities for controlling the lips, tongue and other components of the vocal apparatus, the ability to acoustically decode speech sounds, and the neurological apparatus required for acquiring and producing language.^[39] The study of the genetic bases for human language is still on a fairly basic level, and the only gene that has been positively implied in language production is FOXP2, which may cause a kind of congenital language disorder if affected by mutations.^[40]

The brain and language

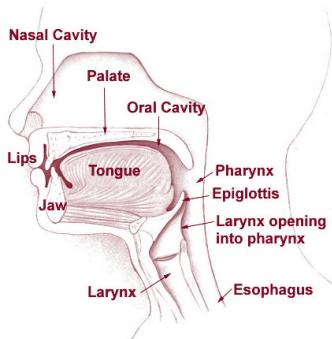
The brain is the coordinating center of all linguistic activity: It controls both the production of linguistic cognition and of meaning and the mechanics of speech production. Nonetheless our knowledge of the neurological bases for language is quite limited, though it has advanced considerably with the use of modern imaging techniques. The discipline of linguistics dedicated to studying the neurological aspects of language is called neurolinguistics.^[41]

Early work in neurolinguistics involved the study of language in people with brain lesions, to see how lesions in specific areas affect language and speech. In this way it was neuroscientists in the 19th century discovered that two areas in the brain are crucially implicated in language processing: Wernicke's area located in the posterior section of the superior temporal gyrus in the dominant cerebral hemisphere. People with a lesion in this area of the brain develop Receptive aphasia, a condition in which there is a major impairment of language comprehension, while speech retains a natural-sounding rhythm and a relatively normal sentence structure. The other area is Broca's area located in the posterior inferior frontal gyrus of the dominant hemisphere. People with a lesion to this area develop expressive aphasia, meaning that they know "what they want to say, they just cannot get it out."^[42] They are typically able to understand what is being said to them, but unable to speak fluently. Other symptoms that may be present in Broca's aphasia include problems with fluency, articulation, word-finding, word repetition, and producing and comprehending complex grammatical sentences, both orally and in writing. They also exhibit ungrammatical speech and show inability to use syntactic information to determine the meaning of sentences. Both Broca's and Wernicke's aphasia also affect the use of sign language, in analogous ways to how they affect speech, with Broca's aphasia causing signers to sign slowly and with incorrect grammar, whereas a signer with Wernicke's aphasia will sign fluently, but make little sense to others and have difficulties comprehending others' signs. This shows that the impairment is specific to the ability to use language, and not to the physiology used for speech production.^{[43][44]}

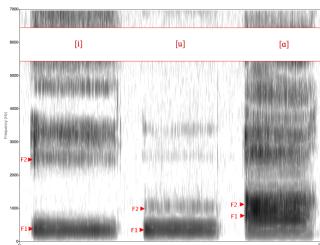
With technological advances in the late 20th century, neurolinguists have also adopted non-invasive techniques such as functional magnetic resonance imaging (fMRI) and electrophysiology to study language processing in individuals without impairments.^[41]



Anatomy of speech



The human vocal tract.



Spectrogram of American English vowels [i, u, ɑ] showing the formants f_1 and f_2



Real time MRI scan of a person speaking in Mandarin Chinese.

Spoken language relies on our physical ability to produce sound, which is a longitudinal wave propagated through the air at a frequency capable of vibrating the human ear drum. This ability depends on the physiology of the human speech organs. These organs consist of the lungs, the voice box (larynx) and the upper vocal tract - the throat, the mouth and the nose. By controlling the different parts of the speech apparatus the airstream can be manipulated to produce different speech sounds.^[45]

The sound of speech can be analyzed into a combination of segmental and suprasegmental elements. The segmental elements are those that follow each other in sequences, and which are usually represented by distinct letters in alphabetic scripts such as the Roman script. In free flowing speech, there are no clear boundaries between one segment and the next, nor usually are there any audible pauses between words. Segments therefore are distinguished by their distinct sounds which are a result of their different articulations, and they can be either vowels or consonants. Suprasegmental phenomena encompass such elements as stress, phonation type, voice timbre and prosody or intonation all of which may have effects across multiple segments.^[46]

Consonants and vowel segments combine to form syllables, which in turn combine to form utterances; these can be distinguished phonetically as the space between two inhalations. Acoustically, these different segments are characterized by different formant structures, that are visible in a spectrogram of the recorded sound wave (See illustration of Spectrogram of the formant structures of three English vowels). Formants are the amplitude peaks in the frequency spectrum of a specific sound.^{[46][47]}

Vowels are those sounds that have no audible friction caused by the narrowing or obstruction of some part of the upper vocal tract. They vary in quality according to the degree of lip aperture and the placement of the tongue within the oral cavity.^[46] Vowels are called *close* when the lips are relatively closed, as in the pronunciation of the vowel [i] (English "ee"), or *open* when the lips are relatively open, as in the vowel [a] (English "ah"). If the tongue is located towards the back of the mouth the quality changes, creating vowels such as [u] (English "oo"). And the quality also changes depending on whether the lips are rounded as opposed to unrounded, creating distinctions such as that between [i] (unrounded front vowel such as English "ee") and [y] (rounded front vowel such as German "ü").^[48]

Consonants are those sounds that have audible friction or closure at some point within the upper vocal tract. Consonants sounds vary by place of articulation, i.e. the place in the vocal tract where the airflow is obstructed - commonly at the lips, teeth, alveolar ridge, *palate*, *velum*, *uvula* or *glottis*. Each place of articulation produces a different set of consonant sounds, which are further distinguished by manner of articulation - the kind of friction - whether full closure, in which case the consonant is called *occlusive* or *stop*, or different degrees of aperture creating *fricatives* and *approximants*. Consonants can also be either *voiced* or *unvoiced*, depending on whether the vocal cords are set in vibration by the airflow during the production of the sound. Voicing is what separates English [s] in *bus* (unvoiced sibilant) from [z] in *buzz* (voiced sibilant).^[49]

Some speech sounds, both vowels and consonants, involve release of air flow through the nasal cavity, and these are called *nasals* or *nasalized* sounds. Other sounds are defined by the way the tongue moves within the mouth: such as the l-sounds (called *laterals*, because the air flows along both sides of the tongue), and the r-sounds (called *rhotics*) that are characterized by how the tongue is positioned relative to the air stream.^[47]

By using these speech organs, humans can produce hundreds of distinct sounds: some appear very often in the world's languages whereas others are much more common in certain language families, or language areas, or even specific to a single language.^[50]

Structure

When described as a system of symbolic communication, language is traditionally seen as consisting of three parts: signs, meanings and a code connecting signs with their meanings. The study of the process of semiosis, how signs and meanings are combined, used and interpreted is called semiotics. Signs can be composed of sounds, gestures, letters or symbols, depending on whether the language is spoken, signed or written, and they can be combined into complex signs such as words and phrases. When used in communication a sign is encoded and transmitted by a sender through a channel to a receiver who decodes it.^[51]

Some of the properties that define human language as opposed to other communication systems are: the arbitrariness of the linguistic sign, meaning that there is no predictable connection between a linguistic sign and its meaning; the duality of the linguistic system, meaning that linguistic structures are built by combining elements into larger structures that can be seen as layered, e.g. how sounds build words and words build phrases; the discreteness of the elements of language, meaning that the elements out of which linguistic signs are constructed are discrete units, e.g. sounds and words, that can be distinguished from each other and rearranged in different patterns; and the productivity of the linguistic system, meaning that the finite number of linguistic elements can be combined into a theoretically infinite number of combinations.^[51]



Ancient Tamil inscription at Thanjavur

The rules under which signs can be combined to form words and phrases are called syntax or grammar. The meaning that is connected to individual signs, morphemes, words, phrases and texts is called semantics.^[52] The division of language into separate but connected systems of sign and meaning goes back to the first linguistic studies of de

Saussure and is now used in almost all branches of linguistics.^[53]

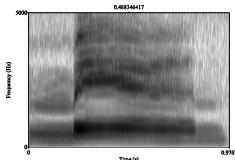
Semantics

Languages express meaning by relating a sign form to a meaning, its content. Sign forms must be something that can be perceived, for example in sounds, images or gestures, and they come to be related to a specific meaning by social convention. Because the basic relation of meaning for most linguistic signs is based on social convention, linguistic signs can be considered arbitrary, in the sense that the convention is established socially and historically, rather than by means of a natural relation between a specific sign form and its meaning.

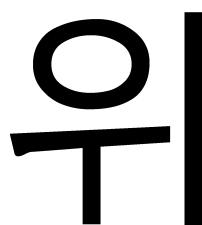
Thus languages must have a vocabulary of signs related to specific meaning—the English sign "dog" denotes, for example, a member of the species *Canis familiaris*. In a language, the array of arbitrary signs connected to specific meanings is called the lexicon, and a single sign connected to a meaning is called a lexeme. Not all meanings in a language are represented by single words - often semantic concepts are embedded in the morphology or syntax of the language in the form of grammatical categories.^[54]

All languages contain the semantic structure of predication—a structure that predicates a property, state or action. Traditionally semantics has been understood as the study of how speakers and interpreters assign truth values to statements, so that meaning is understood as the process by which a predicate can be said to be true or false about an entity, e.g. "[x [is y]]" or "[x [does y]]." Recently, this model of semantics has been complemented with more dynamic models of meaning that incorporate shared knowledge about the context in which a sign is interpreted into the production of meaning. Such models of meaning are explored in the field of pragmatics.^[55]

Sounds and Symbols



A spectrogram showing the sound of the spoken English word "man" which is written phonetically as [mæn]. note that in flowing speech there is no clear division between segments, only a smooth transition as the vocal apparatus moves.



The letter "wi" in the Hangul script.



The sign for "wi" in Korean Sign Language

Depending on modality language structure can be based on systems of sounds (speech), gestures (sign languages) or graphic or tactile symbols (writing). The ways in which languages use sounds or signs to construct meaning are studied in phonology.^[56] The study of how humans produce and perceive vocal sounds is called phonetics.^[57] In spoken language, meaning is produced when sounds become part of a system in which some sounds can contribute to expressing meaning and others do not. In any given language only a limited number of the many distinct sounds that can be created by the human vocal apparatus contribute to constructing meaning.^[58]

Sounds as part of a linguistic system are called phonemes.^[59] Phonemes are abstract units of sound, defined as the smallest units in a language that can serve to distinguish between the meaning of a pair of minimally different words, a so-called minimal pair. In English for example the words /bat/ [bat] and /pat/ [p^hat] form a minimal pair in which the distinction between /b/ and /p/ differentiates the two words as having different meanings. But each language contrasts sounds in different ways: for example in a language that does not distinguish between voiced and unvoiced consonants the sounds [p] and [b] would be considered a single phoneme and consequently the two pronunciations would have the same meaning. Similarly, the English language does not distinguish phonemically between aspirated and non-aspirated pronunciations of consonants as many other languages do: the unaspirated /p/ in /spin/ [spin] and the aspirated /p/ in /pin/ [p^hin] are considered as merely different ways of pronouncing the same phoneme (such variants of a single phoneme are called allophones), whereas in Mandarin Chinese the same difference in pronunciation distinguishes between the words [pá] "crouch" and [pá] "eight" (the accent above the á means that the vowel is pronounced with a high tone).^[60]

All spoken languages have phonemes of at least two different categories, vowels and consonants, that can be combined to form syllables.^[46] As well as segments such as consonants and vowels, some languages also use sound in other ways to convey meaning. Many languages, for example, use stress, pitch, duration and tone to distinguish meaning. Because these phenomena operate outside of the level of single segments they are called suprasegmental.^[61] Some languages have only a few phonemes, for example Rotokas and Pirahā language with 11 and 10 phonemes respectively, whereas languages like Taa may have as many as 141 phonemes.^[60] In sign languages the equivalent to phonemes (formerly called cheremes) are defined by the basic elements of gestures such as hand shape, orientation, location, and motion, which correspond to manners of articulation in spoken language.^[62]

Writing systems represent language using visual symbols, which may or may not correspond to the sounds of spoken language. The Latin alphabet (and those on which it is based or that have been derived from it) was originally based on the representation of single sounds, so that words were constructed from letters that generally denote a single consonant or vowel in the structure of the word. In syllabic scripts, such as the Inuktitut syllabary, each sign represents a whole syllable. In logographic scripts each sign represents an entire word,^[63] and will generally bear no relation to the sound of that word in spoken language.

Because all languages have a very large number of words, no purely logographic scripts are known to exist. Written language represents the way spoken sounds and words follow one after another by arranging symbols according to a pattern that follows a certain direction. The direction used in a writing system is entirely arbitrary and established by convention. Some writing systems use the horizontal axis (left to right as the Latin script or right to left as the Arabic script), while others such as traditional Chinese writing use the vertical dimension (from top to bottom). A few writing systems use opposite directions for alternating lines, and others such as the ancient Maya script can be written in either direction and relies on graphic cues to show the reader the direction of reading.^[64]

In order to represent the sounds of the world's languages in writing, linguists have developed the International Phonetic Alphabet, designed to represent all of the discrete sounds that are known to contribute to meaning in human languages.^[65]

Grammar

Grammar is the study of how meaningful elements called *morphemes*, within a language can be combined into utterances. Morphemes can either be *free* or *bound*. If they are free to be moved around within an utterance, they are usually called *words*, and if they are bound to other words or morphemes, they are called affixes. The way in which meaningful elements can be combined within a language is governed by rules. The rules obtaining for the internal structure of words are called morphology. The rules of the internal structure of phrases and sentences are called *syntax*.^[66]

Grammatical categories

Grammar can be described as a system of categories, and a set of rules that determine how categories combine to form different aspects of meaning.^[67] Languages differ widely in whether categories are encoded through the use of categories or lexical units. However, several categories are so common as to be nearly universal. Such universal categories include the encoding of the grammatical relations of participants and predicates by grammatically distinguishing between their relations to a predicate, the encoding of temporal and spatial relations on predicates, and a system of grammatical person governing reference to and distinction between speakers and addressees and those about whom they are speaking.^[68]

Word classes

Languages organize their parts of speech into classes according to their functions and positions relative to other parts. All languages, for instance, make a basic distinction between a group of words that prototypically denote things and concepts and a group of words that prototypically denote actions and events. The first group, which includes English words such as "dog" and "song," are usually called nouns. The second, which includes "run" and "sing," are called verbs. An other common category is the adjective, words that describe properties or qualities of nouns such as "red" or "big". Word classes can be "open", if new words can continuously be added to the class, or relatively "closed", if there is a fixed number of words in a class. In English the class of pronouns is closed, whereas the class of adjectives is open, since infinite numbers of adjectives can be constructed from verbs (e.g. "saddened") or nouns (e.g. with the -like suffix "noun-like"). In other languages such as Korean the situation is the opposite and new pronouns can be constructed, whereas the number of adjectives is fixed.^[69]

The word classes also carry out differing functions in grammar. Prototypically verbs are used to construct predicates, while nouns are used as arguments of predicates. In a sentence such as "Sally runs," the predicate is "runs," because it is the word that predicates a specific state about its argument "Sally." Some verbs such as "curse" can take two arguments, e.g. "Sally cursed John." A predicate that can only take a single argument is called *intransitive*, while a predicate that can take two arguments is called *transitive*.^[70]

Many other word classes exist in different languages, such as conjunctions that serve to join two sentences and articles that introduces a noun, interjections such as "agh!" or "wow!", or ideophones that mimic the sound of some event. Some languages have positionals, that describe the spatial position of an event or entity. Many languages have classifiers that identify countable nouns as belonging to a particular type or having a particular shape. For instance, in Japanese, the general noun classifier for humans is *nin* (人), and it is used for counting humans, whatever they are called:

san-nin no gakusei (三人の学生) lit. "3 human-classifier of student" — three students

While for trees, it would be:

san-bon no ki (三本の木) lit. "3 classifier-for-long-objects of tree" — three trees

Morphology

In linguistics, the study of the internal structure of complex words, and the processes by which words are formed is called morphology. In most languages, it is possible to construct complex words that are built of several morphemes. For instance the English word "unexpected" can be analyzed as being composed of the three morphemes "un-", "expect" and "-ed".^[71]

Morphemes can be classified according to whether they are independent morphemes, so-called *roots*, or whether they can only co-occur attached to other morphemes. These bound morphemes or *affixes* can be classified according to their position in relation to the root: *prefixes* precede the root, *suffixes* follow the root and *infixes* are inserted in the middle of a root. Affixes serve to modify or elaborate the meaning of the root. Some languages change the meaning of words by changing the phonological structure of a word, for example the English word "run" which in the past tense is "ran". This process is called *ablaut*. Furthermore, morphology distinguishes between the process of *inflection* which modifies or elaborates on a word, and the process of *derivation* which creates a new word from an existing one. In English the verb "sing" has the inflectional forms "singing" and "sung" which are both verbs, and the derivational form "singer" which is a noun derived from the verb with the agentive suffix "-er".^{[72][73]}

Languages differ widely in how much they rely on morphological processes of word formation. In some languages, for example Chinese, there are no morphological processes and all grammatical information is encoded syntactically by forming strings of single words. This type of morpho-syntax is often called isolating, or analytic, because there is almost a full correspondence between a single word and a single aspect of meaning. Most languages have words consisting of several morphemes, but they vary in the degree to which morphemes are discrete units. In many languages, notably in most Indo-European languages, single morphemes may have several distinct meanings that cannot be analyzed into smaller segments. For example in Latin the word *bonus* "good" consists of the root *bon-* meaning "good" and the suffix *-us* which means masculine gender, singular number and nominative case. These languages are called *fusional languages*, because several meanings may be fused into a single morpheme. The opposite type of fusional languages are agglutinative languages which construct words by stringing morphemes together in chains, but with each morpheme as a discrete semantic unit. An example of such a language is Turkish, where for example the word *evlerinizden* "from your houses" consists of the morphemes, *ev-ler-iniz-den* with the meanings *house-plural-your-from*. The languages that rely on morphology to the greatest extent are traditionally called polysynthetic languages. They may express the equivalent of an entire English sentence in a single word. For example the Yupik word *tuntussuqatarniksaitengqiggtuq* which means "He had not yet said again that he was going to hunt reindeer." The word consists of the morphemes *tuntu-ssur-qatar-ni-ksaite-ngqiggte-uq* with the meanings, reindeer-hunt-future-say-negation-again-third.person.singular.indicative, and except for the morpheme *tuntu* "reindeer", none of the other morphemes can appear in isolation.^[74]

Many languages use morphology to cross-reference words within a sentence. This is sometimes called *agreement*. For example, in many Indo-European languages adjectives must cross-reference the noun they modify in terms of number, case and gender, so that the Latin adjective *bonus* "good" is inflected to agree with a noun that is masculine gender and singular. In many polysynthetic languages verbs cross-reference their subjects and objects. In these types of languages, a single verb may include information that would require an entire sentence in English. For example in the Basque phrase *ikusi nauzu* "you saw me", the past tense auxiliary verb *n-au-zu* (similar to English "do") agrees with both the subject (you) expressed by the *n-* prefix, and with the object (me) expressed by the *-zu* suffix. The sentence could be directly transliterated as "see you-did-me"^[75]

Syntax

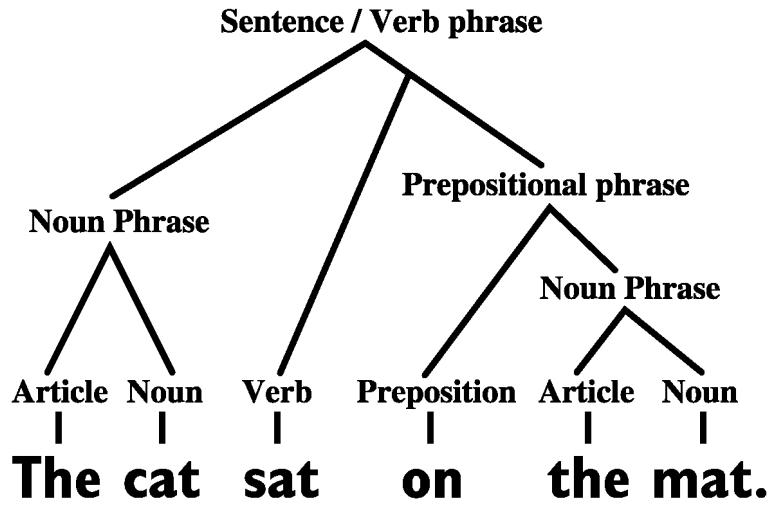
Another way in which languages convey meaning is through the order of words within a sentence. The grammatical rules for how to produce new sentences from words that are already known is called syntax. It is the syntactical rules of a language that determine why a sentence in English such as "*I love you*" is meaningful but "**love you I*" is not^[76]: syntactical rules determine how word order and sentence structure is constrained, and how those constraints contribute to meaning.^[77] For example, in English the two sentences "the slaves were cursing the master" and "the master was cursing the slaves" mean different things because the role of the grammatical subject is encoded by the noun being in front of the verb, and the role of object is encoded by the noun appearing after the verb.

But in Latin both *Dominus servos vituperabat* and *Servos vituperabat dominus* mean "the master was reprimanding the slaves", because *servos* "slaves" is in the accusative case showing that they are the grammatical object of the sentence and *dominus* "master" is in the nominative case showing that he is the subject.^[78]

Latin uses morphology to express the distinction between subject and object, whereas English uses word order. Another example of how syntactic rules contribute to meaning is the rule of inverse word order in questions which exists in many languages. This rule is the reason that in English, when the phrase "John is talking to Lucy" is turned into a question it becomes "Who is John talking to?" and not "John is talking to who?". The latter example may be used as a way of placing special emphasis on "who", thereby slightly altering the meaning of the question. Syntax also includes the rules for how complex sentences are structured by grouping words together in units, called phrases, that can occupy different places in a larger syntactic structure. Sentences can be described as consisting of phrases connected in a tree structure, connecting the phrases to each other at different levels.^[79] To the right is a graphic representation of the syntactic analysis of the English sentence "the cat sat on the mat". The sentence is analysed as being constituted by a noun phrase, a verb and a prepositional phrase; the prepositional phrase is further divided into a preposition and a noun phrase; and the noun phrases consist of an article and a noun.^[80]

The reason sentences can be seen as composed of phrases is because each phrase would be moved around as a single element if syntactic operations are carried out. For example "the cat" is one phrase and "on the mat" is another because they would be treated as single units if we decided to emphasize the location by moving forward the prepositional phrase: "[And] on the mat, the cat sat".^[81] There are many different formalist and functionalist frameworks that propose theories for describing syntactic structures, based on different assumptions about what language is and how it should be described. Each of them would analyze a sentence such as this in a different manner.^[13]

Basic constituent structure analysis of a sentence:



In addition to word classes, a sentence can be analyzed in terms of grammatical functions:
 "The cat" is the subject of the phrase, "on the mat" is a locative phrase, and "sat" is the core of the predicate.

Typology: universals and diversity

Languages can be classified in relation to their grammatical types. Languages that belong to different families nonetheless often have features in common, and these shared features tend to correlate.^[82] For example languages can be classified on the basis of their basic word order, the relative order of the verb, and its constituents in a normal indicative sentence. In English the basic order is SVO "The snake(S) bit(V) the man(O)", whereas for example the corresponding sentence in the Australian language Gamilaraay would be "*duyugu nama dayn yi:y*" (Snake Man Bit), SOV.^[83] Word order type is relevant as a typological parameters because basic word order type corresponds with other syntactic parameters, such as the relative order of nouns and adjectives, or of the use of prepositions of postpositions. Such correlations are called implicational universals. For example most (but not all) languages that are of the SOV type have postpositions rather than prepositions, and have adjectives before nouns.^[84]

Through the study of various types of word order it has been discovered that not all languages group the relations between actors and actions as English do into Subject, Object and Verb - this type is called the nominative-accusative type. Some languages called ergative, Gamilaraay among them, distinguish between Agents and Patients. In English transitive clauses, both the subject of intransitive sentences ("I run") and transitive sentences ("I love you") are treated in the same way, shown here by the nominative pronoun *I*. In ergative languages the single participant in an intransitive sentence such as I run is treated the same as the patient in a transitive sentence - giving the equivalent of "me run" and "you love me", only in transitive sentences would the equivalent of the pronoun *I* be used.^[83] In this way the semantic roles can map onto the grammatical relations in different ways, grouping an intransitive subject either with Agents (accusative type) or Patients (ergative type) or even making each of the three roles differently, which is called the tripartite type.^[85]

The shared features of languages which belong to the same typological class type may have arisen completely independently. Their co-occurrence might be due to the universal laws governing the structure of natural languages—language universals—or they might be the result of languages evolving convergent solutions to the recurring communicative problems that humans use language to solve.^[14]

Social contexts of use and transmission

While all humans have the ability to learn any language, they only do so if they grow up in an environment in which language exists and is used by others. Language is therefore dependent on communities of speakers in which children learn language from their elders and peers, and themselves transmit language to their own children. Languages are used by those who speak them to communicate, and to solve a plethora of social tasks. Many aspects of language use can be seen to be adapted specifically to these purposes.^[14] Due to the way in which language is transmitted between generations and within communities, language perpetually changes, diversifying into new languages or converging due to language contact. The process is similar to the process of evolution, where the process of descent with modification leads to the formation of a phylogenetic tree.^[86]

However, languages differ from biological organisms in that they readily incorporate elements from other languages through the process of diffusion, as speakers of different languages come into contact. Humans also frequently speak more than one language, acquiring their first language or languages as children, or learning new languages as they grow up. Because of the increased language contact in the globalizing world many small languages are becoming endangered as their speakers shift to other languages that afford the possibility to participate in larger and more influential speech communities.^[87]

Usage and meaning

The semantic study of meaning assumes that meaning is located in a relation between signs and meanings that are firmly established through social convention. But semantics does not study the way in which social conventions are made and affect language. However, when studying the way in which words and signs are used, it is often the case that words have different meanings depending on the social context of use. And signs also change their meaning over time, as the conventions governing their usage gradually change. The study of how the meaning of linguistic expressions change depending on context is called pragmatics. Pragmatics is concerned with the ways in which language use is patterned and how these patterns contribute to meaning. For example in all languages linguistic expressions can be used not just to transmit information, but to perform actions. Certain actions are made only through language, but nonetheless have tangible effects. For example the act of 'naming', which creates a new name for some entity, or the act of 'pronouncing someone man and wife' which creates a social contract of marriage. These types of acts are called speech acts, although they can of course also be carried out through writing or hand signing.^[88]

The form of linguistic expression often does not correspond to the meaning that it actually has in a social context. For example, if at a dinner table a person asks "can you reach the salt?", that is in fact not a question about the length of the arms of the one being addressed, but a request to pass the salt across the table. This meaning is implied by the context in which it is spoken, these kinds of effects of meaning are called conversational implicatures. These social rules for the ways in which certain ways of using language are considered appropriate in certain situations, and how to understand utterances in relation to their context, vary between communities, and learning them is a large part of acquiring communicative competence in a language.^[89]

Language acquisition

All healthy, normally-developing human beings learn to use language. Children acquire the language or languages used around them – whichever languages they receive sufficient exposure to during childhood. The development is essentially the same for children acquiring sign or oral languages.^[90] This learning process is referred to as first-language acquisition, since unlike many other kinds of learning it requires no direct teaching or specialized study. In *The Descent of Man*, naturalist Charles Darwin called this process "an instinctive tendency to acquire an art."^[8]

First language acquisition proceeds in a fairly regular sequence, though there is a wide degree of variation in the timing of particular stages among normally-developing infants. From birth, newborns respond more readily to human speech than to other sounds. Around one month of age, babies appear to be able to distinguish between different speech sounds. Around six months of age, a child will begin babbling, producing the speech sounds or handshapes of the languages used around them. Words appear around the age of 12 to 18 months; the average vocabulary of an eighteen-month old child is around 50 words. A child's first utterances are holophrases (literally "whole-sentences"), utterances that use just one word to communicate some idea. Several months after a child begins producing words, she or he will produce two-word utterances, and within a few more months begin to produce telegraphic speech, short sentences



All normal children acquire language if they are exposed to it in their first years of life, even in cultures where adults rarely address infants and toddlers directly.

that are less grammatically complex than adult speech, but that do show regular syntactic structure. From roughly the age of three to five years, a child's ability to speak or sign is refined to the point that it resembles adult language.^[91]

Acquisition of second and additional languages can come at any age, through exposure in daily life or courses. Children learning a second language are more likely to achieve native-like fluency than adults, but in general it is very rare for someone speaking a second language to pass completely for a native speaker. An important difference between first language acquisition and additional language acquisition is that the process of additional language acquisition is influenced by languages that the learner already knows.

Language and culture

Languages, understood as the particular set of speech norms of a particular community, are also a part of the larger culture of the community that speak them. Humans use language as a way of signalling identity with one cultural group and difference from others. Even among speakers of one language several different ways of using the language exist, and each is used to signal affiliation with particular subgroups within a larger culture. Linguists and anthropologists, particularly sociolinguists, ethnolinguists and linguistic anthropologists have specialized in studying how ways of speaking vary between speech communities.^[92]

A community's way of using language is a part of the community's culture, just as other shared practices are; it is a way of displaying group identity. Ways of speaking function not only to facilitate communication, but also to identify the social position of the speaker. In many languages there are stylistic or even grammatical differences between the ways men and women speak. Just as some languages employ different words depending on who is listening. For example in the Australian language Dyirbal a married man must use a special set of words to refer to everyday items when speaking in the presence of his mother in-law.^[93]

Linguists use the term "varieties" to refer to the different ways of speaking a language. This term includes geographically or socioculturally defined dialects as well as the jargons or styles of subcultures. Linguistic anthropologists and sociologists of language define communicative style as the ways that language is used and understood within a particular culture.^[93]

Languages do not differ only in pronunciation, vocabulary or grammar, but also through having different "cultures of speaking". Some cultures for example have elaborate systems of "social deixis", systems of signalling social distance through linguistic means.^[94] In English, social deixis is shown mostly through distinguishing between addressing some people by first name and others by surname, but also in titles such as "Mrs.", "boy", "Doctor" or "Your Honor", but in other languages such systems may be highly complex and codified in the entire grammar and vocabulary of the language. For instance, in several languages of east Asia, such as Thai, Burmese and Javanese, different words are used according to whether a speaker is addressing someone of higher or lower rank than oneself in a ranking system with animals and children ranking the lowest and gods and members of royalty as the highest.^[94]



Arnold Lakhovsky, *The Conversation* (circa 1935)

Writing, literacy and technology

Throughout history a number of different ways of representing language in graphic media have been invented. These are called writing systems.

The use of writing has made language even more useful to humans. It makes it possible to store large amounts of information outside of the human body and retrieve it again, and it allows communication across distances that would otherwise be impossible. Many languages conventionally employ different genres, styles and register in written and spoken language, and in some communities writing traditionally takes place in an entirely different language than the one spoken. There is some evidence that the use of writing also have effects on the cognitive development of humans, perhaps because acquiring literacy generally requires explicit and formal education.^[95]

The invention of the first writing systems is roughly contemporary with the beginning of the Bronze Age in the late Neolithic of the late 4th millennium BC. The Sumerian archaic cuneiform script and the Egyptian hieroglyphs are generally considered the earliest writing systems, both emerging out of their ancestral proto-literate symbol systems from 3400–3200 BC with the earliest coherent texts from about 2600 BC. It is generally agreed that Sumerian writing was an independent invention; however, it is debated whether Egyptian writing was developed completely independently of Sumerian, or was a case of cultural diffusion. A similar debate exists for the Chinese script, which developed around 1200 BC. The pre-Columbian Mesoamerican writing systems (including among others Olmec and Maya scripts) are generally believed to have had independent origins.^[64]



An inscription of Swampy Cree using Canadian Aboriginal syllabics, an abugida developed by Christian missionaries for Indigenous Canadian languages

Language change

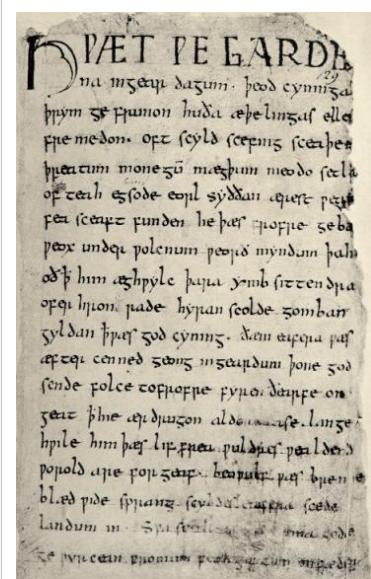
All languages change as speakers adopt or invent new ways of speaking and pass them on to other members of their speech community. Language change happens at all levels from the phonological level to the levels of vocabulary, morphology, syntax and discourse. Even though language change is often initially evaluated negatively by speakers of the language who often consider changes to be "decay" or a sign of slipping norms of language usage, it is natural and inevitable.^{[96][97]}

Changes may affect specific sounds or the entire phonological system. Sound change can consist of the replacement of one speech sound or phonetic feature by another, or of the complete loss of the affected sound, or even the introduction of a new sound in a place where there previously was none. Sound changes can be *conditioned* in which case a sound is changed only if it occurs in the vicinity of certain other sounds. Sound change is usually assumed to be *regular*, which means that it is expected to apply mechanically whenever its structural conditions are met, irrespective of any non-phonological factors. On the other hand, sound changes can sometimes be *sporadic*, affecting only one particular word or a few words, without any seeming regularity. Sometimes a simple change triggers a chain shift in which the entire phonological system is affected. This happened in the Germanic languages when the sound change known as Grimm's law affected all the stop consonants in the system. The original consonant *b^h became /b/ in the Germanic languages, and the previous *b in turn became /p/ and the previous *p became /f/. The same process applied to all stop consonants and explains why Italic languages such as Latin have *p* in words like *pater* and *pisces* whereas Germanic languages like English have *father* and *fish*.^[98]

Another example is the Great Vowel Shift in English, which is the reason that the spelling of English vowels do not correspond well to their current pronunciation, this is because the vowel shift brought the already established orthography out of synchronization with pronunciation. Another source of sound change is the erosion of words, as pronunciation gradually becomes increasingly indistinct and shortens words leaving out syllables or sounds. This kind of change caused Latin *mea domina* to eventually become the French *madame* and American English *ma'am*.^[99]

Change also happens in the grammar of languages as discourse patterns such as idioms or particular constructions become grammaticalized. This frequently happen when words or morphemes erode and the grammatical system is unconsciously rearranged to compensate for the lost element. For example in some varieties of Caribbean Spanish where word final /s/ has eroded away. Since Standard Spanish uses final /s/ is the morpheme marking the second person subject "you" on verbs, the Caribbean varieties now have to express the second person using the pronoun *tú*. This means that the sentence "what's your name" is *¿como te llamas?* ['komo te 'jamas] in Standard Spanish, but ['komo 'tu te 'jama] in Caribbean Spanish. The simple sound change has affected both morphology and syntax.^[100] Another common cause of grammatical change is the gradual petrification of idioms into new grammatical forms, for example the way the English "going to" construction lost its aspect of movement and in some varieties of English has almost become a full fledged future tense (e.g. *I'm gonna*).

Language change may be motivated by "language internal" factors, such as changes in pronunciation motivated by certain sounds being difficult to distinguish auditively or to produce, or because of certain patterns of change that cause certain rare types of constructions to drift towards more common types.^[101] Other causes of language change are social, such as when certain certain pronunciations become emblematic of membership of certain groups, such as



The first page of the Beowulf poem written in Old English in the early medieval period (800 - 1100 AD). Although old English language is the direct ancestor of modern English language change has rendered it unintelligible to contemporary English speakers.

social classes, or with ideologies, and therefore are adopted by those who wish to identify with those groups or ideas. In this way issues of identity and politics can have profound effects on language structure.^[102]

Language contact

One important source of language change is contact between different languages and resulting diffusion of linguistic traits between languages. Language contact occurs when speakers of two or more languages or varieties interact on a regular basis.^[103] Multilingualism is likely to have been the norm throughout human history, and today most people in the world are multilingual. Before the rise of the concept of the ethno-national state, monolingualism was characteristic mainly of populations inhabiting small islands. But with the ideology that made one people, one, state and one language the most desirable political arrangement monolingualism started to spread throughout the world. Nonetheless there are only 250 countries in the world corresponding to some 6000 languages, so most countries are multilingual and most languages therefore exist in close contact with other languages.^[104]

When speakers of different languages interact closely, it is typical for their languages to influence each other. Through sustained language contact over long periods linguistic traits diffuse between languages, and languages belonging to different families may converge to become more similar. In areas where many languages are in close contact this may lead to the formation of language areas in which unrelated languages share a number of linguistic features. A number of such language areas have been documented, among them: the Balkan language area, the Mesoamerican language area, and the Ethiopian language area. Also larger areas such as South Asia, Europe and South East Asia have sometimes been considered language areas, because of widespread diffusion of specific areal features.^{[105][106]}

Language contact may also lead to a variety of other linguistic phenomena, including language convergence, borrowing, and relexification (replacement of much of the native vocabulary with that of another language). In situations of extreme and sustained language contact it may lead to the formation of new mixed languages that cannot be considered to belong to a single language family. One type of mixed language called pidgins occurs when adult speakers of two different languages interact on a regular basis, but in a situation where neither group learns to learn to speak the language of the other group fluently. In such a case they will often construct a communication form that has traits of both languages, but which has a simplified grammatical and phonological structure, the language comes to contain mostly the grammatical and phonological categories that exist in both languages. Pidgin languages are defined by not having any native speakers, but only being spoken by people who have another language as their first language. But if a Pidgin language becomes the main language of a speech community, then eventually children will grow up learning the pidgin as their first language. As the generation of child learners grow up the pidgin will often be seen to change its structure and acquire a greater degree of complexity. This type of language is generally called a creole language. An example of such mixed languages are Tok Pisin the official language of Papua New-Guinea which originally arose as a Pidgin based on English and Austronesian languages; others are Kreyòl ayisyen the French based creole language spoken in Haiti, and Michif, a mixed language of Canada, based on the Native American language Cree and French.^{[107][108]}

Linguistic diversity

Language	Native speakers (in mil.) ^[109]
Mandarin	845
Spanish	329 ^[110]
English	328
Arabic languages	221
Hindi	182
Bengali	181
Portuguese	178
Russian	144
Japanese	122
German	90,3

A "living language" is simply one which is in wide use as a primary form of communication by a specific group of living people. The exact number of known living languages varies from 6,000 to 7,000, depending on the precision of one's definition of "language", and in particular on how one defines the distinction between languages and dialects. As of 2009, SIL Ethnologue catalogued 6909 living human languages. The Ethnologue establishes linguistic groups based on studies of mutual intelligibility, and therefore often include more categories than more conservative classifications. For example the Danish language that most scholars consider a single language with several dialects, is classified as three distinct languages by the Ethnologue.^[109]

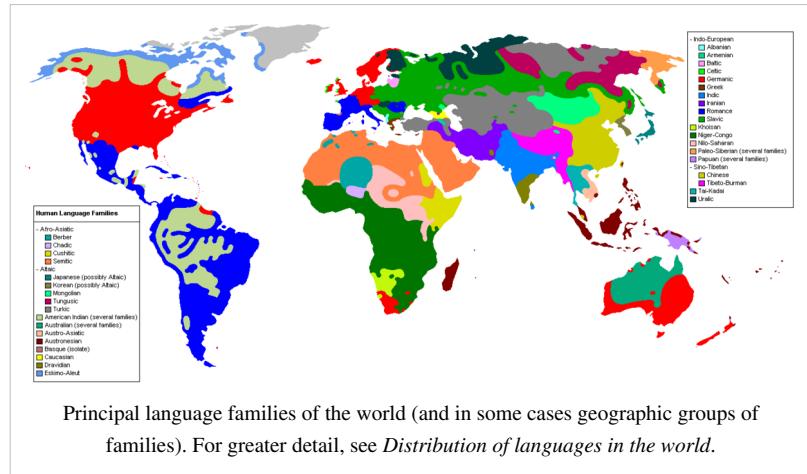
The Ethnologue is also sometimes criticized for using cumulative data gathered over many decades, meaning that exact speaker numbers are frequently out of date, and some languages classified as living may have already become extinct. According to the Ethnologue 389 (or nearly 6%) languages have more than a million speakers. These languages together account for 94% of the world's population, whereas 94% of the world's languages account for the remaining 6% of the global population. To the right is a table of the world's 10 most spoken languages with population estimates from the Ethnologue (2009 figures).^[109]

Languages and dialects

There is no clear distinction between a language and a dialect, notwithstanding a famous aphorism attributed to linguist Max Weinreich that "a language is a dialect with an army and navy".^[111] For example, national boundaries frequently override linguistic difference in determining whether two linguistic varieties are languages or dialects. Cantonese and Mandarin are for example often classified as "dialects" of Chinese, even though they are more different from each other than Swedish is from Norwegian. Before the Yugoslav civil war, Serbo-Croatian was considered a single language with two dialects, but now Croatian and Serbian are considered different languages, and employ different writing systems. In other words, the distinction may hinge on political considerations as much as on cultural differences, distinctive writing systems, or degree of mutual intelligibility.^[112]

Language families of the World

The world's languages can be grouped into language families consisting of languages that can be shown to have common ancestry. Linguists currently recognize many hundred language families, although some of them can possibly be grouped into larger units as more evidence becomes available and in-depth studies are carried out. At present there are also dozens of language isolates - languages that cannot be shown to be related to any other languages in the world. Among them is Basque, spoken in Europe, Zuni of New Mexico, P'urhépecha of Mexico, Ainu of Japan, Burushaski of Pakistan and many others.



Principal language families of the world (and in some cases geographic groups of families). For greater detail, see *Distribution of languages in the world*.

The language families of the world that have most speakers are the Indo-European languages, spoken by 46% of the world's population. This family includes major world languages like English, Spanish, Russian and Hindi/Urdu. The Indo-European family achieved prevalence first during the Eurasian Migration Period (c. 400-800 AD), and subsequently through the European colonial expansion which brought the Indo-European languages to a politically and often numerically dominant position in the Americas and much of Africa. The Sino-Tibetan languages are spoken by 21% of the world's population and includes many of the languages of East Asia including Mandarin Chinese, Cantonese and hundreds of smaller languages.

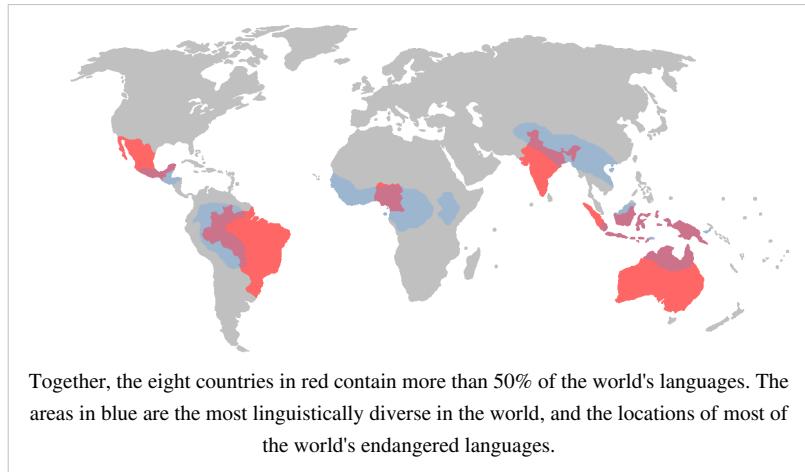
Africa is home to a large number of language families, the largest of which is the Niger–Congo languages which includes such languages as Kiswahili, Shona and Yoruba. Speakers of the Niger-Congo languages account for 6.4% of the world's population. A similar number of people speak the Afroasiatic languages, which include the populous Semitic languages such as Arabic, Hebrew language and the languages of the Sahara region such as the Berber languages and Hausa.

The Austronesian languages are spoken by 5.9% of the world's population and stretches from Madagascar to maritime Southeast Asia all the way to Oceania. It includes such languages as Malagasy, Māori, Samoan, and many of the indigenous languages of Indonesia and Taiwan. The Austronesian languages are considered to have originated in Taiwan around 3000 BC. and spread through the Oceanic region through island-hopping, based on an level advanced nautical technology. Other populous language families are the Dravidian languages of South Asia (among them Tamil and Telugu), the Turkic languages of Central Asia (such as Turkish), and the Austro-Asiatic (Among them Khmer) and Tai–Kadai languages of Southeast Asia (including Thai).^[113]

The areas of the world where there is greatest linguistic diversity such as the Americas, Papua-New Guinea, West Africa and South-Asia contain hundreds of small language families. These areas together account for the majority of the world's languages, though not the majority of speakers. In the Americas some of the largest language families include the Quechumaran, Arawak, and Tupi-Guarani families of South America, the Uto-Aztecán, Oto-Manguean, Mayan of Mesoamerica, and the Na-Dene and Algonquian language families of North America. In Australia, most indigenous languages belong to the Pama-Nyungan family, whereas Papua-New Guinea is home to a large number of small families and isolates, as well as a number of Austronesian languages.^[114]

Language endangerment

Language endangerment occurs when a language is at risk of falling out of use as its speakers die out or shift to speaking another language. Language loss occurs when the language has no more native speakers, and becomes a *dead language*. If eventually no one speaks the language at all, it becomes an *extinct language*. While languages have always gone extinct throughout human history, they are currently disappearing at an accelerated rate due to the processes of globalization and neo-colonialism, where the economically powerful languages dominate other languages.^[1]



The more commonly spoken languages dominate the less commonly spoken languages and therefore, the less commonly spoken languages eventually disappear from populations. The total number of languages in the world is not known. Estimates vary depending on many factors. The general consensus is that there are between 6,000^[2] and 7,000 languages currently spoken, and that between 50-90% of those will have become extinct by the year 2100.^[1] The top 20 languages spoken by more than 50 million speakers each, are spoken by 50% of the world's population, whereas many of the other languages are spoken by small communities, most of them with less than 10,000 speakers.^[1]

The United Nations Educational, Scientific and Cultural Organization (UNESCO) operates with five levels of language endangerment: "safe", "vulnerable" (not spoken by children outside the home), "definitely endangered" (not spoken by children), "severely endangered" (only spoken by the oldest generations), "critically endangered" (spoken by few members of the oldest generation, often semi-speakers). Notwithstanding claims that the world would be better off if most adopted a single common *lingua franca* such as English or Esperanto, there is a general consensus that the loss of languages harms the cultural diversity of the world. It is a common belief, going back to the biblical narrative of the tower of Babel that linguistic diversity causes political conflict,^[24] but this belief is contradicted by the facts that many of the world's major episodes of violence have taken place in situations with low linguistic diversity, such as the Yugoslav and American Civil Wars, or the genocides of Nazi Germany and Rwanda, whereas many of the most stable political units have been highly multilingual.^[115]

Many projects under way are aimed at preventing or slowing this loss by revitalizing endangered languages and promoting education and literacy in minority languages. Across the world many countries have enacted specific legislation aimed at protecting and stabilizing the language of indigenous speech communities. A minority of linguists have argued that language loss is a natural process that should not be counteracted, and that documenting endangered languages for posterity is sufficient.^[116]

Notes

- [1] Austin & Sallabank (2011)
- [2] Moseley (2010)
- [3] "language". *The American Heritage Dictionary of the English Language* (3rd ed.). Boston: Houghton Mifflin Company. 1992.
- [4] Lyons (1981:2)
- [5] Lyons (1981:1–8)
- [6] Trask (2007:129–31)
- [7] Hauser & Fitch (2003)
- [8] Pinker (1994)
- [9] Saussure & Harris (1983)
- [10] Campbell (2001:96)
- [11] Chomsky (1957)
- [12] Trask (2007:93, 130)
- [13] Newmeyer (1998:3–6)
- [14] Evans & Levinson (2009)
- [15] Van Valin (2001)
- [16] Hockett (1960); Deacon (1997)
- [17] Trask (1999:1–5)
- [18] Trask (1999:9)
- [19] Tomasello (2008)
- [20] The Gorilla Koko reportedly uses as many as 1000 words in American Sign Language, and understands 2000 words of spoken English.
There are some doubts about whether her use of signs is based in complex understanding or in simple conditioning.
- [21] Deacon (1997)
- [22] Hauser, Chomsky & Fitch (2002)
- [23] Trask (2007:165–66)
- [24] Haugen (1973)
- [25] Ulbaek (1998)
- [26] Chomsky (2000:4)
- [27] Fitch (2010:466–507)
- [28] Fitch (2010:250–92)
- [29] Foley (1997:70–74)
- [30] Fitch (2010:292–3)
- [31] Newmeyer (2005)
- [32] Trask (2007)
- [33] Campbell (2001:82–83)
- [34] Bloomfield 1914, p. 310
- [35] Clarke (1990:143–144)
- [36] Foley (1997:82–83)
- [37] Nichols (1984) "*Functional grammar analyzes grammatical structure, as do formal and structural grammar; but it also analyzes the entire communicative situation: the purpose of the speech event, its participants, its discourse context. Functionalists maintain that the communicative situation motivates, constrains, explains, or otherwise determines grammatical structure, and that a structural or formal approaches not merely limited to an artificially restricted data base, but is inadequate even as a structural account. Functional grammar, then, differs from formal and structural grammar in that it purports not to model but to explain; and the explanation is grounded in the communicative situation.*"
- [38] Croft & Cruse (2004:1)
- [39] Trask (1999:11–14; 105–113)
- [40] Fisher, Lai & Monaco (2003)
- [41] Lesser (1989:205–6)
- [42] Trask (1999:105–7)
- [43] Trask (1999:108)
- [44] Sandler & Lillo-Martin (2001:554)
- [45] MacMahon (1989:2)
- [46] MacMahon (1989:3)
- [47] International Phonetic Association (1999:3–8)
- [48] MacMahon (1989:11–15)
- [49] MacMahon (1989:6–11)
- [50] Ladefoged & Maddieson (1996)
- [51] Lyons (1981:17–24)

- [52] Trask (1999:35)
- [53] Lyons (1981:218–24)
- [54] Levinson (1983)
- [55] Levinson (1983)
- [56] Goldsmith (1995)
- [57] International Phonetic Association (1999)
- [58] Ladefoged & Maddieson (1996)
- [59] International Phonetic Association (1999:27)
- [60] Trask (2007:214)
- [61] International Phonetic Association (1999:4)
- [62] Sandler & Lillo-Martin (2001:539–40)
- [63] Trask (2007:326)
- [64] Coulmas (2002)
- [65] Trask (2007:123)
- [66] Lyons (1981:103)
- [67] Allerton (1989)
- [68] Payne (1997)
- [69] Trask (2007:208)
- [70] Trask (2007:305)
- [71] Aronoff & Fudeman (2011:1–2)
- [72] Bauer (2003)
- [73] Haspelmath (2002)
- [74] Payne (1997:28–29)
- [75] Trask (2007:11)
- [76] The prefixed asterisk * conventionally indicates that the sentence is ungrammatical, i.e. syntactically incorrect
- [77] Baker (2001:265)
- [78] Trask (2007:179)
- [79] Baker (2001:269–70)
- [80] Trask (2007:218–19)
- [81] Trask (2007:218–19)
- [82] Nichols (1992); Comrie (1989)
- [83] Croft (2001:340)
- [84] Greenberg (1966)
- [85] Croft (2001:355)
- [86] Campbell (2004)
- [87] Austin & Sallabank (2011)
- [88] Levinson (1983:226–78)
- [89] Levinson (1983:100–169)
- [90] Bonvillian, John D.; Michael D. Orlansky and Leslie Lazin Novack (December 1983). "Developmental milestones: Sign language acquisition and motor development". *Child Development* 54 (6): 1435–1445.
- [91] O'Grady, William; Cho, Sook Whan (2001). "First language acquisition". *Contemporary Linguistics: An Introduction* (fourth ed.). Boston: Bedford St. Martin's.
- [92] Duranti (2003)
- [93] Foley (1997)
- [94] Foley (1997:311–28)
- [95] Olson (1996)
- [96] Aitchison (2001)
- [97] Trask (1999:70)
- [98] Clackson (2007:27–33)
- [99] Aitchison (2001:112)
- [100] Zentella (2002:178)
- [101] Labov (1994)
- [102] Labov (2001)
- [103] Thomason (2001:1)
- [104] Romaine (2001:513)
- [105] Campbell (2002)
- [106] Aikhenvald (2001)
- [107] Thomason & Kaufman (1988); Thomason (2001)
- [108] Matras & Bakker (2003)

- [109] Lewis (2009)
- [110] Ethnologue's figure is based on numbers from before 1995. A more recent figure is 420 million ("Primer estudio conjunto del Instituto Cervantes y el British Council sobre el peso internacional del español y del inglés" (http://www.cervantes.es/sobre_instituto_cervantes/prensa/2012/noticias/nota-londres-palabra-por-palabra.htm). Instituto Cervantes (www.cervantes.es). .)
- [111] Rickerson, E.M.. "What's the difference between dialect and language?" (http://spinner.cofc.edu/linguist/archives/2005/08/whats_the_diffe.html?referrer=webcluster&). *The Five Minute Linguist*. College of Charleston. . Retrieved 17 July 2011.
- [112] Lyons (1981:26)
- [113] Katzner (1999); Comrie (2009); Brown & Ogilvie (2008)
- [114] Katzner (1999); Comrie (2009); Brown & Ogilvie (2008)
- [115] Austin & Sallabank (2011:10–11)
- [116] Ladefoged (1992)

References

- Aikhenvald, Alexandra (2001). "Introduction". In Alexandra Y. Aikhenvald. *Areal diffusion and genetic inheritance: problems in comparative linguistics*. Oxford: Oxford University Press. pp. 1–26.
- Aitchison, Jean (2001). *Language Change: Progress or Decay?* (3rd (1st edition 1981) ed.). Cambridge, New York, Melbourne: Cambridge University Press.
- Allerton, D. J. (1989). "Language as Form and Pattern: Grammar and its Categories". In Collinge, N.E.. *An Encyclopedia of Language*. London:NewYork: Routledge.
- Aronoff, Mark; Fudeman, Kirsten (2011). *What is Morphology*. John Wiley & Sons.
- Austin, Peter K; Sallabank, Julia (2011). "Introduction". In Austin, Peter K; Sallabank, Julia. *Cambridge Handbook of Endangered Languages*. Cambridge University Press. ISBN 978-0-521-88215-6.
- Baker, Mark C. (2001). "Syntax". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 265–295.
- Bauer, Laurie (2003). *Introducing linguistic morphology* (2nd ed.). Washington, D.C.: Georgetown University Press. ISBN 0-87840-343-4.
- Bloomfield, Leonard (1914). *An introduction to the study of language*. New York: Henry Holt and Company.
- Brown, Keith; Ogilvie, Sarah, eds. (2008). *Concise Encyclopedia of Languages of the World*. Elsevier Science. ISBN 0080877745.
- Clackson, James (2007). *Indo-European Linguistics: An Introduction*. Cambridge University press.
- Campbell, Lyle (2002). "Areal linguistics". In Bernard Comrie, Neil J. Smelser and Paul B. Balte. *International Encyclopedia of Social and Behavioral Sciences*. Oxford: Pergamon. pp. 729–733.
- Campbell, Lyle (2004). *Historical Linguistics: an Introduction* (2nd ed.). Edinburgh and Cambridge, MA: Edinburgh University Press and MIT Press.
- Campbell, Lyle (2001). "The History of Linguistics". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 81–105.
- Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton.
- Chomsky, Noam (2000). *The Architecture of Language*. Oxford: Oxford University Press.
- Clarke, David S. (1990). *Sources of semiotic: readings with commentary from antiquity to the present*. Carbondale: Southern Illinois University Press.
- Comrie, Bernard (1989). *Language universals and linguistic typology: Syntax and morphology*. (2nd ed.). Oxford: Blackwell. ISBN 0-226-11433-3.
- Comrie, Bernard, ed. (2009). *The World's Major Languages*. New York: Routledge. ISBN 978-0-415-35339-7.
- Coulmas, Florian (2002). *Writing Systems: An Introduction to Their Linguistic Analysis*. Cambridge University Press.

- Croft, William; Cruse, D. Alan (2004). *Cognitive Linguistics*. Cambridge: Cambridge University Press.
- Croft, William (2001). "Typology". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 81–105.
- Crystal, David (1997). *The Cambridge Encyclopedia of Language*. Cambridge: Cambridge University Press.
- Deacon, Terrence (1997). *The Symbolic Species: The Co-evolution of Language and the Brain..* New York: W.W. Norton & Company. ISBN 978-0-393-31754-1.
- Duranti, Alessandro (2003). "Language as Culture in U.S. Anthropology: Three Paradigms". *Current Anthropology* **44** (3): 323–348.
- Evans, Nicholas; Levinson, Stephen C. (2009). *The myth of language universals: Language diversity and its importance for cognitive science*. **32**. Behavioral and Brain Sciences. pp. 429–492.
- Fisher, Simon E.; Lai, Cecilia S.L.; Monaco, Anthony P. (2003). "Deciphering the Genetic Basis of Speech and Language Disorders". *Annual Review of Neuroscience* **26**: 57–80. doi:10.1146/annurev.neuro.26.041002.131144. PMID 12524432.
- Fitch, W. Tecumseh (2010). *The Evolution of Language*. Cambridge: Cambridge University Press.
- Foley, William A. (1997). *Anthropological Linguistics: An Introduction*. Blackwell.
- Goldsmith, John A (1995). "Phonological Theory". In John A. Goldsmith. *The Handbook of Phonological Theory*. Blackwell Handbooks in Linguistics. Blackwell Publishers. ISBN 1-4051-5768-2.
- Greenberg, Joseph (1966). *Language Universals: With Special Reference to Feature Hierarchies*. The Hague: Mouton & Co.
- Haspelmath, Martin (2002). *Understanding morphology*. London: Arnold, Oxford University Press. (pbk)
- Haugen, Einar (1973). "The Curse of Babel". *Daedalus* **102** (3, Language as a Human Problem): 47–57.
- Hauser, Marc D.; Chomsky, Noam; Fitch, W. Tecumseh (2002). "The Faculty of Language: What Is It, Who Has It, and How Did It Evolve?". *Science* **298** (5598): 1569–1579.
- Hauser, Marc D.; Fitch, W. Tecumseh (2003). "What are the uniquely human components of the language faculty?" (<http://www.isrl.uiuc.edu/~amag/langev/paper/hauser03whatAre.html>). In M.H. Christiansen and S. Kirby. *Language Evolution: The States of the Art*. Oxford University Press.
- Hockett, Charles F. (1960). "Logical considerations in the study of animal communication". In W.E. Lanyon. *Animals sounds and animal communication*. pp. 392–430.
- International Phonetic Association (1999). *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge: Cambridge University Press. ISBN 0-521-65236-7 (hb); ISBN 0-521-63751-1 (pb).
- Katzner, K (1999). *The Languages of the World*. New York: Routledge.
- Labov, William (1994). Principles of Linguistic Change vol.I Internal Factors. Blackwell.
- Labov, William (2001). Principles of Linguistic Change vol.II Social Factors. Blackwell.
- Ladefoged, Peter (1992). "Another view of endangered languages". *Language* **68** (4): 809–811.
- Ladefoged, Ian; Maddieson (1996). *The sounds of the world's languages*. Oxford: Blackwell. pp. 329–330. ISBN 0-631-19815-6.
- Lesser, Ruth (1989). "Language in the Brain: Neurolinguistics". In Collinge, N.E.. *An Encyclopedia of Language*. London:NewYork: Routledge.
- Levinson, Stephen C. (1983). *Pragmatics*. Cambridge: Cambridge University Press.
- Lewis, M. Paul (ed.) (2009). "Ethnologue: Languages of the World, Sixteenth edition" (http://www.ethnologue.com/ethno_docs/distribution.asp?by=size). Dallas, Tex.: SIL International.
- Lyons, John (1981). *Language and Linguistics*. Cambridge University Press. ISBN 0-521-29775-3.

- MacMahon, M.K.C. (1989). "Language as available sound:Phonetics". In Collinge, N.E.. *An Encyclopedia of Language*. London:NewYork: Routledge.
- Matras, Yaron; Bakker, Peter, eds. (2003). *The Mixed Language Debate: Theoretical and Empirical Advances*. Berlin: Walter de Gruyter. ISBN 3-11-017776-5.
- Moseley, Christopher, ed. (2010). *Atlas of the World's Languages in Danger, 3rd edition*. (<http://www.unesco.org/culture/en/endangeredlanguages/atlas>). Paris: UNESCO Publishing.
- Newmeyer, Frederick J. (2005). *The History of Linguistics* (<http://www.lsadc.org/info/ling-fields-history.cfm>). Linguistic Society of America. ISBN 0-415-11553-1.
- Newmeyer, Frederick J. (1998). *Language Form and Language Function* (http://www.isc.cnrs.fr/FN_chapter1.pdf). Cambridge,MA: MIT Press.
- Nichols, Johanna (1992). *Linguistic diversity in space and time*. Chicago: University of Chicago Press. ISBN 0-226-58057-1.
- Nichols, Johanna (1984). "Functional Theories of Grammar". *Annual Review of Anthropology* **13**: 97–117.
- Olson, David R. (1996). "Language and Literacy: what writing does to Language and Mind". *Annual Review of Applied Linguistics* **16**: 3–13. doi:10.1017/S0267190500001392.
- Payne, Thomas Edward (1997). *Describing morphosyntax: a guide for field linguists* (<http://books.google.com/books?id=LC3DfjWfCiwC&pg=PA239&lpg=PA239&dq=%22perfect+aspect%22+&ct=result#PPA238,M1>). Cambridge University Press. pp. 238–241.
- Pinker, Steven (1994). *The Language Instinct: How the Mind Creates Language*. Perennial.
- Romaine, Suzanne (2001). "Multilingualism". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 512–533.
- Saussure, Ferdinand de; Harris, Roy, Translator (1983) [1913]. Bally, Charles; Sechehaye, Albert. eds. *Course in General Linguistics*. La Salle, Illinois: Open Court. ISBN 0-8126-9023-0.
- Sandler, Wendy; Lillo-Martin, Diane (2001). "Natural Sign Languages". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 533–563.
- Swadesh, Morris (1934). "The phonemic principle". *Language* **10** (2): 117–129. doi:10.2307/409603. JSTOR 409603.
- Tomasello, Michael (2008). *Origin of Human Communication*. MIT Press.
- Thomason, Sarah G.; Kaufman, Terrence (1988). *Language Contact, Creolization and Genetic Linguistics*. University of California Press.
- Thomason, Sarah G. (2001). *Language Contact - An Introduction*. Edinburgh University Press.
- Trask, Robert Lawrence (1999). *Language: The Basics* (2nd ed.). Psychology Press.
- Trask, Robert Lawrence (2007). Stockwell, Peter. ed. *Language and Linugistics: The Key Concepts* (2nd ed.). Routledge.
- Ulbaek, Ib (1998). "The Origin of Language and Cognition". In J. R. Hurford & C. Knight. *Approaches to the evolution of language*. Cambridge University Press. pp. 30–43.
- Van Valin, jr, Robert D. (2001). "Functional Linguistics". In Mark Aronoff. *The Handbook of Linguistics*. Blackwell. pp. 319–337.
- Zentella, Ana Celia (2002). "Spanish in New York". In García, Ofelia; Fishman, Joshua. *The Multilingual Apple: Languages in New York City*.

External links

- World Atlas of Language Structures: a large database of structural (phonological, grammatical, lexical) properties of languages (<http://wals.info/>)

Natural language

In the philosophy of language, a **natural language** (or **ordinary language**) is any language which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect. A natural language is typically used for communication, and may be spoken, signed, or written. Natural language is distinguished from constructed languages and formal languages such as computer-programming languages or the "languages" used in the study of formal logic, especially mathematical logic.

Defining natural language

Though the exact definition varies between scholars, natural language can broadly be defined in contrast on the one hand to artificial or constructed languages, such as **computer programming languages** like Python and international auxiliary languages like Esperanto, and on the other hand to other communication systems in nature, such as the waggle dance of bees. Although there are a variety of natural languages, any cognitively normal human infant is able to learn any natural language. By comparing the different natural languages, scholars hope to learn something about the nature of human intelligence and the innate biases and constraints that shape natural language, which are sometimes called universal grammar.

Linguists have an incomplete understanding of all aspects of the rules underlying natural languages, and these rules are therefore objects of study. The understanding of natural languages reveals much about not only how language works (in terms of syntax, semantics, phonetics, phonology, etc.), but also about how the human mind and the human brain process language. In linguistic terms, *natural language* only applies to a language that has developed naturally, and the study of natural language primarily involves native (first language) speakers.^[1]

While grammarians, writers of dictionaries, and language policy-makers all have a certain influence on the evolution of language, their ability to influence what people think they *ought* to say is distinct from what people actually say. The term *natural language* refers to actual linguistic behavior, and is aligned with descriptive linguistics rather than linguistic prescription. Thus non-standard language varieties (such as African American Vernacular English) are considered to be natural while standard language varieties (such as Standard American English) which are more prescribed can be considered to be at least somewhat artificial or constructed.^[2]

Native language learning

The learning of one's own native language, typically that of one's parents, normally occurs spontaneously in early human childhood and is biologically, socially and ecologically driven. A crucial role of this process is the ability of humans from an early age to engage in speech repetition and so quickly acquire a spoken vocabulary from the pronunciation of words spoken around them. This together with other aspects of speech involves the neural activity of parts of the human brain such as the Wernicke's and Broca's areas.^[3]

There are approximately 7,000 current human languages, and many, if not most seem to share certain properties, leading to the hypothesis of Universal Grammar, as argued by the generative grammar studies of Noam Chomsky and his followers. Recently, it has been demonstrated that a dedicated network in the human brain (crucially involving Broca's area, a portion of the left inferior frontal gyrus), is selectively activated by complex verbal structures (but not simple ones) of those languages that meet the Universal Grammar requirements.^{[4][5]}

While it is clear that there are innate mechanisms that enable the learning of language and define the range of languages that can be learned, it is not clear that these mechanisms in anyway resemble a human language or universal grammar. The study of language acquisition is the domain of psycholinguistics and Chomsky always declined to engage in questions of how his putative language organ, the Language Acquisition Device or Universal Grammar, might have evolved.^[6] During a period (the 1970s and 80s) when nativist Transformational Generative Grammar was becoming dominant in Linguistics, and called "Standard Theory", linguists who questioned these tenets were disenfranchised and Cognitive Linguistics and Computational Psycholinguistics were born and the more general term Emergentism developed for the anti-nativist view that language is emergent from more fundamental cognitive processes that are not specifically linguistic in nature.

Origins of natural language

There is disagreement among anthropologists on when language was first used by humans (or their ancestors). Estimates range from about two million (2,000,000) years ago, during the time of *Homo habilis*, to as recently as forty thousand (40,000) years ago, during the time of Cro-Magnon man. However recent evidence suggests modern human language was invented or evolved in Africa prior to the dispersal of humans from Africa around 50,000 years ago. Since all people including the most isolated indigenous groups such as the Andamanese or the Tasmanian aborigines possess language, then it was presumably present in the ancestral populations in Africa before the human population split into various groups to inhabit the rest of the world.^{[7][8]}

Controlled languages

Controlled natural languages are subsets of natural languages whose grammars and dictionaries have been restricted in order to reduce or eliminate both ambiguity and complexity (for instance, by cutting down on rarely used superlative or adverbial forms or irregular verbs). The purpose behind the development and implementation of a controlled natural language typically is to aid non-native speakers of a natural language in understanding it, or to ease computer processing of a natural language. An example of a widely used controlled natural language is Simplified English, which was originally developed for aerospace industry maintenance manuals.

Constructed languages and international auxiliary languages

Constructed international auxiliary languages such as Esperanto and Interlingua (even those that have native speakers) are not generally considered natural languages.^[9] The problem is that other languages have been used to communicate and evolve in a natural way, while Esperanto was selectively designed by L.L. Zamenhof from natural languages, not grown from the natural fluctuations in vocabulary and syntax. Some natural languages have become naturally "standardized" by children's natural tendency to correct for illogical grammar structures in their parents' language, which can be seen in the development of pidgin languages into creole languages (as explained by Steven Pinker in *The Language Instinct*), but this is not the case in many languages, including constructed languages such as Esperanto, where strict rules are in place as an attempt to consciously remove such irregularities. The possible exception to this are true native speakers of such languages.^[10] More substantive basis for this designation is that the vocabulary, grammar, and orthography of Interlingua are natural; they have been standardized and presented by a linguistic research body, but they predated it and are not themselves considered a product of human invention.^[11] Most experts, however, consider Interlingua to be naturalistic rather than natural.^[9] Latino Sine Flexione, a second naturalistic auxiliary language, is also naturalistic in content but is no longer widely spoken.^[12]

Modalities

Natural language manifests itself in modalities other than speech.

Sign languages

A sign language is a language which conveys meaning through visual rather than acoustic patterns—simultaneously combining hand shapes, orientation and movement of the hands, arms or body, and facial expressions to express a speaker's thoughts. Sign languages are natural languages which have developed in Deaf communities, which can include interpreters and friends and families of deaf people as well as people who are deaf or hard of hearing themselves.

In contrast, a manually coded language (or signed oral language) is a constructed sign system combining elements of a sign language and an oral language. For example, Signed Exact English (SEE) did not develop naturally in any population, but was "created by a committee of individuals".^[13]

Written languages

In a sense, written language should be distinguished from natural language. Until recently in the developed world, it was common for many people to be fluent in spoken and yet remain illiterate; this is still the case in poor countries today. Furthermore, natural language acquisition during childhood is largely spontaneous, while literacy must usually be intentionally acquired.^[14]

Notes

- [1] Witzany G (2011). Can mathematics explain the evolution of human language. *Communicative & Integrative Biology* 4(5):1-5.
- [2] Language: Journal of the Linguistic Society of America ([http://books.google.se/books?id=OgHXBP3SoBsC&pg=PA56&lpg=PA56&dq=While+grammarians,+writers+of+dictionaries,+and+language+policy-makers+all+have+a+certain+influence+on+the+evolution+of+language,+their+ability+to+influence+what+people+think+they+ought+to+say+is+distinct+from+what+people+actually+say.+The+term+natural+language+refers+to+actual+linguistic+behavior,+and+is+aligned+with+descriptive+linguistics+rather+than+linguistic+prescription.+Thus+non-standard+language+varieties+\(such+as+African+American+Vernacular+English\)+are+considered+to+be+natural+while+standard+language+varieties+\(such+as+Standard+American+English\)+which+are+more+prescribed+can+be+considered+to+be+at+least+somewhat+artificial+or+constructed.&source=bl&ots=z_7xv60wBM&sig=HR0KcvvHD-dGWqa_CqyiFhXBa2A&hl=sv&sa=X&ei=zUieULqmBlbZ4QTgt4GYAg&ved=0CCkQ6AEwAQ#v=onepage&q&f=false](http://books.google.se/books?id=OgHXBP3SoBsC&pg=PA56&lpg=PA56&dq=While+grammarians,+writers+of+dictionaries,+and+language+policy-makers+all+have+a+certain+influence+on+the+evolution+of+language,+their+ability+to+influence+what+people+think+they+ought+to+say+is+distinct+from+what+people+actually+say.+The+term+natural+language+refers+to+actual+linguistic+behavior,+and+is+aligned+with+descriptive+linguistics+rather+than+linguistic+prescription.+Thus+non-standard+language+varieties+(such+as+African+American+Vernacular+English)+are+considered+to+be+natural+while+standard+language+varieties+(such+as+Standard+American+English)+which+are+more+prescribed+can+be+considered+to+be+at+least+somewhat+artificial+or+constructed.&source=bl&ots=z_7xv60wBM&sig=HR0KcvvHD-dGWqa_CqyiFhXBa2A&hl=sv&sa=X&ei=zUieULqmBlbZ4QTgt4GYAg&ved=0CCkQ6AEwAQ#v=onepage&q&f=false))
- [3] Kendra A. Palmer (2009). "Understanding Human Language: An In-Depth Exploration of the Human Facility for Language" (<http://www.studentpulse.com/articles/82/2/understanding-human-language-an-in-depth-exploration-of-the-human-facility-for-language>). StudentPulse.com. . Retrieved 22 August 2012.
- [4] A. Moro, M. Tettamanti, D. Perani, C. Donati, S. F. Cappa, F. Fazio "Syntax and the brain: disentangling grammar by selective anomalies", *NeuroImage*, 13, January 2001, Academic Press, Chicago, pp. 110-118
- [5] Musso, M., Moro, A. , Glauche. V., Rijntjes, M., Reichenbach, J., Büchel, C., Weiller, C. "Broca's area and the language instinct," *Nature neuroscience*, 2003, vol. 6, pp. 774-781.
- [6] Piattelli-Palmarini M., *Language and Learning: The Debate between Jean Piaget & Noam Chomsky*, Routledge and Kegan Paul, 1980.
- [7] Early Voices: The Leap to Language *nytimes* article by (<http://query.nytimes.com/gst/fullpage.html?res=9503E0DF173CF936A25754C0A9659C8B63&sec=health&spon=&pagewanted=1>) Nicholas Wade
- [8] (http://www.benjamins.com/cgi-bin/t_bookview.cgi?bookid=CELCR 5)
- [9] Gopsill, F. P., "A historical overview of international languages". In *International languages: A matter for Interlingua*. Sheffield, England: British Interlingua Society, 1990.
- [10] Proponents contend that there are 200-2000 native speakers of Esperanto.
- [11] Gode, Alexander, *Interlingua-English: A dictionary of the international language*. New York: Storm Publishers, 1951. (Original edition)
- [12] Gopsill, F. P., "Naturalistic international languages". In *International languages: A matter for Interlingua*. Sheffield, England: British Interlingua Society, 1990.
- [13] Emmorey, Karen. *Language, cognition, and the brain: insights from sign language research* (2001), p. 11.
- [14] Pinker, Steven. 1994. *The Language Instinct*

References

- ter Meulen, Alice, 2001, "Logic and Natural Language," in Goble, Lou, ed., *The Blackwell Guide to Philosophical Logic*. Blackwell.

Theoretical linguistics

Theoretical linguistics

Theoretical linguistics is the branch of linguistics that is most concerned with developing models of linguistic knowledge. The fields that are generally considered the core of theoretical linguistics are syntax, phonology, morphology, and semantics. Although phonetics often informs phonology, it is often excluded from the purview of theoretical linguistics, along with psycholinguistics and sociolinguistics. Theoretical linguistics also involves the search for an explanation of linguistic universals, that is, properties all languages have in common.

Major fields

Phonetics

Phonetics is the study of speech sounds with concentration on three main points :

- Articulation : the production of speech sounds in human speech organs.
- Perception : the way human ears respond to speech signals, how the human brain analyses them.
- Acoustic features : physical characteristics of speech sounds such as color, loudness, amplitude, frequency etc.

According to this definition, phonetics can also be called linguistic analysis of human speech at the surface level. That is one obvious difference from phonology, which concerns the structure and organisation of speech sounds in natural languages, and furthermore has a theoretical and abstract nature. One example can be made to illustrate this distinction: In English, the suffix *-s* can represent either /s/, /z/, or can be silent (written Ø) depending on context.

Orthographic representation : S, s

Phonetic features:

Phonetic representations: [s], [z], Ø

Perception through the ear: high frequency sounds accompanied by a hissing noise.

Acoustic features:

Frequency : 8000 – 11000 Hz

Color : similar to the hissing noise made by snakes.

Phonological characteristics :

Occurrence : beginning, middle or end of words.

Accompanied by vowels or consonants.

Distinguishes meanings of words depending on context: **s''low ≠ g''low**

Articulatory phonetics

The field of articulatory phonetics is a subfield of phonetics. In studying articulation, phoneticians attempt to document how humans produce speech sounds (vowels and consonants). That is, articulatory phoneticians are interested in how the different structures of the vocal tract, called the articulators (tongue, lips, jaw, palate, teeth etc.), interact to create the specific sounds.

Auditory phonetics

Auditory phonetics is a branch of phonetics concerned with the hearing, acquisition and comprehension of phonetic sounds of words of a language. As articulatory phonetics explores the methods of sound production, auditory phonetics explores the methods of reception—the ear to the brain, and those processes.

Acoustic phonetics

Acoustic phonetics is a subfield of phonetics which deals with acoustic aspects of speech sounds. Acoustic phonetics investigates properties like the mean squared amplitude of a waveform, its duration, its fundamental frequency, or other properties of its frequency spectrum, and the relationship of these properties to other branches of phonetics (e.g. articulatory or auditory phonetics), and to abstract linguistic concepts like phones, phrases, or utterances.

Phonology

Phonology is the study of language sounds.^[1] Phonology is divided into two separate studies, phonetics and phonemics. Phonetics is what depicts the sounds we hear. It calls attention to the smallest details in language sounds. There are three kinds of phonetics: acoustic phonetics, auditory phonetics, and articulatory phonetics. Acoustic phonetics deals with the physical properties of sound, what sounds exactly are coming from the person speaking. Auditory phonetics deals with how the sounds are perceived, exactly what the person hearing the sounds is perceiving. Finally, articulatory phonetics studies how the speech sounds are produced. This is what describes the actual sounds in detail. It is also known as descriptive phonetics.^[2]

Phonemics studies how the sounds are used. It analyzes the way sounds are arranged in languages and helps you to hear what sounds are important in a language.^[3] The unit of analysis for phonemics is called phonemes. "A phoneme is a sound that functions to distinguish one word from another in a language."^[4] For example, the English word 'tie' sounds different from the word 'die': the sounds that differentiate the words are [t] and [d].^[4]

Morphology

Morphology is the study of word structure. For example, in the sentences *The dog runs* and *The dogs run*, the word forms *runs* and *dogs* have an affix -s added, distinguishing them from the base forms *dog* and *run*. Adding this suffix to a nominal stem gives plural forms, adding it to verbal stems restricts the subject to third person singular. Some morphological theories operate with two distinct suffixes -s, called allomorphs of the morphemes Plural and Third person singular, respectively. Languages differ with respect to their morphological structure. Along one axis, we may distinguish analytic languages, with few or no affixes or other morphological processes from synthetic languages with many affixes. Along another axis, we may distinguish agglutinative languages, where affixes express one grammatical property each, and are added neatly one after another, from fusional languages, with non-concatenative morphological processes (infixation, umlaut, ablaut, etc.) and/or with less clear-cut affix boundaries.

Syntax

Syntax is the study of language structure and phrasal hierarchies, depicted in parse tree format. It is concerned with the relationship between units at the level of words or morphology. Syntax seeks to delineate exactly all and only those sentences which make up a given language, using native speaker intuition. Syntax seeks to describe formally exactly how structural relations between elements (lexical items/words and operators) in a sentence contribute to its interpretation. Syntax uses principles of formal logic and Set Theory to formalize and represent accurately the hierarchical relationship between elements in a sentence. Abstract syntax trees are often used to illustrate the hierarchical structures that are posited. Thus, in active declarative sentences in English the subject is followed by the main verb which in turn is followed by the object (SVO). This order of elements is crucial to its correct interpretation and it is exactly this which syntacticians try to capture. They argue that there must be a formal computational component contained within the language faculty of normal speakers of a language and seek to describe it.

Semantics

Semantics is the study of intension, that is, the intrinsic meanings of words and phrases. Much of the work in the field of philosophy of language is concerned with the relation between meanings and the world, and this concern cross-cuts formal semantics in several ways. For example, both philosophers of language and semanticists make use of propositional, predicate and modal logics to express their ideas about word meaning; what Frege termed 'sense'.

References

- Ottenheimer, H.J. (2006). *The Anthropology of Language: An Introduction to Linguistic Anthropology*. Canada: Thomas Wadsworth.
 - [1] Ottenheimer, 34
 - [2] Ottenheimer, 36-37
 - [3] Ottenheimer, 46-47
 - [4] Ottenheimer, 47

Psycholinguistics

Psycholinguistics or **psychology of language** is the study of the psychological and neurobiological factors that enable humans to acquire, use, comprehend and produce language. Initial forays into psycholinguistics were largely philosophical ventures, due mainly to a lack of cohesive data on how the human brain functioned. Modern research makes use of biology, neuroscience, cognitive science, linguistics, and information theory to study how the brain processes language. There are a number of subdisciplines with non-invasive techniques for studying the neurological workings of the brain; for example, *neurolinguistics* has become a field in its own right.

Psycholinguistics covers the cognitive processes that make it possible to generate a grammatical and meaningful sentence out of vocabulary and grammatical structures, as well as the processes that make it possible to understand utterances, words, text, etc. Developmental psycholinguistics studies children's ability to learn language.

Areas of study

Psycholinguistics is an interdisciplinary field. Hence, it is studied by researchers from a variety of different backgrounds, such as psychology, cognitive science, linguistics, and speech and language pathology. Psycholinguists study many different topics, but these topics can generally be divided into answering the following questions: (1) how do children acquire language (language acquisition)?; (2) how do people process and comprehend language (language comprehension)?; (3) how do people produce language (language production)?; and (4) how do adults acquire a new language (second language acquisition)?

Subdivisions in psycholinguistics are also made based on the different components that make up human language.

Linguistics-related areas:

- Phonetics and phonology are concerned with the study of speech sounds. Within psycholinguistics, research focuses on how the brain processes and understands these sounds.
- Morphology is the study of word structures, especially the relationships between related words (such as *dog* and *dogs*) and the formation of words based on rules (such as plural formation).
- Syntax is the study of the patterns which dictate how words are combined to form sentences.
- Semantics deals with the meaning of words and sentences. Where syntax is concerned with the formal structure of sentences, semantics deals with the actual meaning of sentences.
- Pragmatics is concerned with the role of context in the interpretation of meaning.

A researcher interested in language comprehension may study word recognition during reading to examine the processes involved in the extraction of orthographic, morphological, phonological, and semantic information from patterns in printed text. A researcher interested in language production might study how words are prepared to be spoken starting from the conceptual or semantic level. Developmental psycholinguistics study infants' and children's ability to learn and process language.^[1]

Theories

In this section, some influential theories are discussed for each of the fundamental questions listed in the section above.

Language acquisition

There are essentially two schools of thought as to how children acquire or learn language, and there is still much debate as to which theory is the correct one. The first theory states that all language must be learned by the child. The second view states that the abstract system of language cannot be learned, but that humans possess an innate language faculty, or an access to what has been called universal grammar. The view that language must be learned

was especially popular before 1960 and is well represented by the mentalistic theories of Jean Piaget and the empiricist Rudolf Carnap. Likewise, the school of psychology known as behaviorism (see *Verbal Behavior* (1957) by B.F. Skinner) puts forth the point of view that language is a behavior shaped by conditioned response, hence it is learned.

The innatist perspective began with Noam Chomsky's highly critical review of Skinner's book in 1959.^[2] This review helped to start what has been termed "the cognitive revolution" in psychology. Chomsky posited humans possess a special, innate ability for language and that complex syntactic features, such as recursion, are "hard-wired" in the brain. These abilities are thought to be beyond the grasp of the most intelligent and social non-humans. According to Chomsky, children acquiring a language have a vast search space to explore among all possible human grammars, yet at the time there was no evidence that children receive sufficient input to learn all the rules of their language (see poverty of the stimulus). Hence, there must be some other innate mechanism that endows a language ability to humans. Such a language faculty is, according to the innateness hypothesis, what defines human language and makes it different from even the most sophisticated forms of animal communication.

The field of linguistics and psycholinguistics since then has been defined by reactions to Chomsky, pro and con. The pro view still holds that the human ability to use language (specifically the ability to use recursion) is qualitatively different from any sort of animal ability.^[3] This ability may have resulted from a favorable mutation or from an adaptation of skills evolved for other purposes. The view that language can be learned has had a recent resurgence inspired by emergentism. This view challenges the "innate" view as scientifically unfalsifiable; that is to say, it can't be tested. With the amount of computer power increasing since the 1980s, researchers have been able to simulate language acquisition using neural network models.^[4] These models provide evidence that there may, in fact, be sufficient information contained in the input to learn language, even syntax. If this is true, then an innate mechanism is no longer necessary to explain language acquisition.

Language comprehension

One question in the realm of language comprehension is how people understand sentences as they read (also known as sentence processing). Experimental research has spawned a number of theories about the architecture and mechanisms of sentence comprehension. Typically these theories are concerned with what types of information contained in the sentence the reader can use to build meaning, and at what point in reading does that information become available to the reader. Issues such as "modular" versus "interactive" processing have been theoretical divides in the field.

A modular view of sentence processing assumes that the stages involved in reading a sentence function independently in separate modules. These modules have limited interaction with one another. For example, one influential theory of sentence processing, the garden-path theory,^[5] states that syntactic analysis takes place first. Under this theory as the reader is reading a sentence, he or she creates the simplest structure possible in order to minimize effort and cognitive load. This is done without any input from semantic analysis or context-dependent information. Hence, in the sentence "The evidence examined by the lawyer turned out to be unreliable," by the time the reader gets to the word "examined" he or she has committed to a reading of the sentence in which the evidence is examining something because it is the simplest parse. This commitment is made despite the fact that it results in an implausible situation; we know from experience that evidence can rarely if ever examine something. Under this "syntax first" theory, semantic information is processed at a later stage. It is only later that the reader will recognize that he or she needs to revise the initial parse into one in which "the evidence" is being examined. In this example, readers typically recognize their misparse by the time they reach "by the lawyer" and must go back and re-parse the sentence.^[6] This reanalysis is costly and contributes to slower reading times.

In contrast to a modular account, an interactive theory of sentence processing, such as a constraint-based lexical approach^[7] assumes that all available information contained within a sentence can be processed at any time. Under an interactive account, for example, the semantics of a sentence (such as plausibility) can come into play early on in

order to help determine the structure of a sentence. Hence, in the sentence above, the reader would be able to make use of plausibility information in order to assume that "the evidence" is being examined instead of doing the examining. There are data to support both modular and interactive accounts; which account is the correct one is still up for debate.

Methodologies

Behavioral tasks

Many of the experiments conducted in psycholinguistics, especially earlier on, are behavioral in nature. In these types of studies, subjects are presented with linguistic stimuli and asked to perform an action. For example, they may be asked to make a judgment about a word (lexical decision), reproduce the stimulus, or name a visually presented word aloud. Reaction times to respond to the stimuli (usually on the order of milliseconds) and proportion of correct responses are the most often employed measures of performance in behavioral tasks. Such experiments often take advantage of priming effects, whereby a "priming" word or phrase appearing in the experiment can speed up the lexical decision for a related "target" word later.^[8]

As an example of how behavioral methods can be used in psycholinguistics research, Fischler (1977) investigated word encoding using the lexical decision task. He asked participants to make decisions about whether two strings of letters were English words. Sometimes the strings would be actual English words requiring a "yes" response, and other times they would be nonwords requiring a "no" response. A subset of the licit words were related semantically (e.g., cat-dog) while others were unrelated (e.g., bread-stem). Fischler found that related word pairs were responded to faster when compared to unrelated word pairs. This facilitation suggests that semantic relatedness can facilitate word encoding.^[9]

Eye-movements

Recently, eye tracking has been used to study online language processing. Beginning with Rayner (1978)^[10] the importance and informativity of eye-movements during reading was established. Later, Tanenhaus et al. (1995)^[11] used the visual-world paradigm to study the cognitive processes related to spoken language. Assuming that eye movements are closely linked to the current focus of attention, language processing can be studied by monitoring eye movements while a subject is presented auditorily with linguistic input.

Language Production Errors

The analysis of systematic errors in speech, writing and typing of language as it is produced can provide evidence of the process which has generated it.

Neuroimaging

Until the recent advent of non-invasive medical techniques, brain surgery was the preferred way for language researchers to discover how language works in the brain. For example, severing the corpus callosum (the bundle of nerves that connects the two hemispheres of the brain) was at one time a treatment for some forms of epilepsy. Researchers could then study the ways in which the comprehension and production of language were affected by such drastic surgery. Where an illness made brain surgery necessary, language researchers had an opportunity to pursue their research.

Newer, non-invasive techniques now include brain imaging by positron emission tomography (PET); functional magnetic resonance imaging (fMRI); event-related potentials (ERPs) in electroencephalography (EEG) and magnetoencephalography (MEG); and transcranial magnetic stimulation (TMS). Brain imaging techniques vary in their spatial and temporal resolutions (fMRI has a resolution of a few thousand neurons per pixel, and ERP has millisecond accuracy). Each type of methodology presents a set of advantages and disadvantages for studying a

particular problem in psycholinguistics.

Computational modeling

Computational modeling—e.g. the DRC model of reading and word recognition proposed by Coltheart and colleagues^[12]—is another methodology. It refers to the practice of setting up cognitive models in the form of executable computer programs. Such programs are useful because they require theorists to be explicit in their hypotheses and because they can be used to generate accurate predictions for theoretical models that are so complex that they render discursive analysis unreliable. Another example of computational modeling is McClelland and Elman's TRACE model of speech perception.^[13]

Issues and areas of research

Psycholinguistics is concerned with the nature of the computations and processes that the brain undergoes to comprehend and produce language. For example, the cohort model seeks to describe how words are retrieved from the mental lexicon when an individual hears or sees linguistic input.^{[8][14]}

Recent research using new non-invasive imaging techniques seeks to shed light on just where certain language processes occur in the brain.

There are a number of unanswered questions in psycholinguistics, such as whether the human ability to use syntax is based on innate mental structures or emerges from interaction with other humans, and whether some animals can be taught the syntax of human language.

Two other major subfields of psycholinguistics investigate first language acquisition, the process by which infants acquire language, and second language acquisition. In addition, it is much more difficult for adults to acquire second languages than it is for infants to learn their first language (bilingual infants are able to learn both of their native languages easily). Thus, sensitive periods may exist during which language can be learned readily.^[15] A great deal of research in psycholinguistics focuses on how this ability develops and diminishes over time. It also seems to be the case that the more languages one knows, the easier it is to learn more.^[16]

The field of aphasiology deals with language deficits that arise because of brain damage. Studies in aphasiology can both offer advances in therapy for individuals suffering from aphasia, and further insight into how the brain processes language.

References

- [1] Houston, D.M.; Jusczyk, P.W. (2000). "The ROle of Talker-Specific Information in Word Segmentation by Infants" (http://www.iupui.edu/~babytalk/pdfs/Houston_2011.pdf). *Journal of Experimental Psychology: Human Perception and Performance* **26** (5): 1570–1582. doi:10.1037/0096-1523.26.5.1570. . Retrieved 1 March 2012.
- [2] Chomsky, N; Skinner, B. F. (1959). "A Review of B. F. Skinner's Verbal Behavior". *Language* (Linguistic Society of America) **35** (1): 26–58. doi:10.2307/411334. ISSN 0097-8507. JSTOR 411334.
- [3] Hauser M.D., Chomsky N., Fitch W. (2002). *Science* **298** (5598): 1569–79. doi:10.1126/science.298.5598.1569. PMID 12446899.
- [4] Elman, Jeffrey; Bates Elizabeth, Johnson Mark, Karmiloff-Smith Annette, Parisi Domenico, Plunkett Kim (1998). *Rethinking innateness: A connectionist perspective on development*. The MIT Press.
- [5] Frazier L., Rayner, K. (1982). "Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences". *Cognitive psychology* **14** (2): 178–210.
- [6] Rayner K., Carlson M., Frazier L. (1983). "The interaction of syntax and semantics during sentence processing: Eye movements in the analysis of semantically biased sentences". *Journal of verbal learning and verbal behavior* **22** (3): 358–374.
- [7] Trueswell J., Tanenhaus M. (1994). "Toward a lexical framework of constraint-based syntactic ambiguity resolution". *Perspectives on sentence processing*: 155–179.
- [8] Packard, Jerome L (2000). "Chinese words and the lexicon." *The Morphology of Chinese: A Linguistic and Cognitive Approach*. Cambridge: Cambridge University Press. pp. 284-309.
- [9] Fischler I. (1977). "Semantic facilitation without association in a lexical decision task". *Memory & Cognition* **5** (3): 335-339.
- [10] Rayner K. (1978). "Eye movements in reading and information processing". *Psychological Bulletin* **85** (3): 618–660. doi:10.1037/0033-2909.85.3.618. PMID 353867.

- [11] Tanenhaus M. K., Spivey-Knowlton M. J., Eberhard K. M., Sedivy J. E. (1995). "Integration of visual and linguistic information in spoken language comprehension". *Science* **268** (5217): 1632–1634. doi:10.1126/science.7777863. PMID 7777863.
- [12] Coltheart M., Rastle K., Perry C., Langdon R., Ziegler J. (2001). "DRC: "A dual route cascaded of visual word recognition and reading aloud". *Psychological Review* **108**: 204–256.
- [13] McClelland, J.L., & Elman, J.L. (1986). The TRACE model of speech perception. *Cognitive Psychology*, 18, 1-86.
- [14] Altmann, Gerry T.M. (1997). "Words, and how we (eventually) find them." *The Ascent of Babel: An Exploration of Language, Mind, and Understanding*. Oxford: Oxford University Press. pp. 65-83.
- [15] Seidner, Stanley S.(1982). *Ethnicity, Language, and Power from a Psycholinguistic Perspective*. Bruxelles: Centre de recherche sur le pluralinguisme. pp. 4-7.
- [16] Seidner, Stanley S.(1982). *Ethnicity, Language, and Power from a Psycholinguistic Perspective*. Bruxelles: Centre de recherche sur le pluralinguisme. p. 11.

Further reading

A short list of books that deal with psycholinguistics, written in language accessible to the non-expert, includes:

- Belyanin V.P. *Foundations of Psycholinguistic Diagnostics (Models of the World)*. Moscow, 2000 (in Russian) (http://www.textology.ru/belyanin/bel_ann1.html)
- Chomsky, Noam. (2000) *New Horizons in the Study of Language and Mind*. Cambridge: Cambridge University Press.
- Harley, Trevor. (2008) *The Psychology of Language: From data to theory (3rd. ed.)* (<http://www.psypress.com/harley/>) Hove: Psychology Press.
- Harley, Trevor. (2009) *Talking the talk: Language, psychology and science*. Hove: Psychology Press.
- Lakoff, George. (1987) *Women, fire, and dangerous things: what categories reveal about the mind*. Chicago: University of Chicago Press.
- Piattelli-Palmarini, Massimo. (ed.) (1980) *Language and learning: the debate between Jean Piaget and Noam Chomsky*. Cambridge, Mass.: Harvard University Press.
- Pinker, Steven. (1994) *The Language Instinct*. New York: William Morrow.
- Rayner, K. and Pollatsek, A. (1989) *The Psychology of Reading*. New York:Prentice Hall.
- Steinberg, Danny D., Hiroshi Nagata, and David P. Aline, ed. (2001) *Psycholinguistics: Language, Mind and World*, 2nd ed. Longman (<http://www.ling.ed.ac.uk/linguist/issues/13/13-55.html>)
- Steinberg, Danny D. & Sciarini, Natalia. (2006) *Introduction to Psycholinguistics 2nd edition*. London: Longman.
- Aitchison, Jean. (1998). *The Articulate Mammal: An Introduction to Psycholinguistics*. Routledge.
- Scovel, Thomas. (1998). *Psycholinguistics*. Oxford University Press.

External links

- Psycholinguistics (http://www.dmoz.org/Science/Social_Sciences/Linguistics/Psycholinguistics/) at the Open Directory Project

Sociolinguistics

Sociolinguistics is the descriptive study of the effect of any and all aspects of society, including cultural norms, expectations, and context, on the way language is used, and the effects of language use on society. Sociolinguistics differs from sociology of language in that the focus of sociolinguistics is the effect of the society on the language, while the sociology of language focuses on language's effect on the society. Sociolinguistics overlaps to a considerable degree with pragmatics. It is historically closely related to linguistic anthropology and the distinction between the two fields has even been questioned recently.^[1]

It also studies how language varieties differ between groups separated by certain social variables, e.g., ethnicity, religion, status, gender, level of education, age, etc., and how creation and adherence to these rules is used to categorize individuals in social or socioeconomic classes. As the usage of a language varies from place to place, language usage also varies among social classes, and it is these *sociolects* that sociolinguistics studies.

The social aspects of language were in the modern sense first studied by Indian and Japanese linguists in the 1930s, and also by Gauchat in Switzerland in the early 1900s, but none received much attention in the West until much later. The study of the social motivation of language change, on the other hand, has its foundation in the wave model of the late 19th century. The first attested use of the term *sociolinguistics* was by Thomas Callan Hodson in the title of a 1939 paper.^[2] Sociolinguistics in the West first appeared in the 1960s and was pioneered by linguists such as William Labov in the US and Basil Bernstein in the UK. In the 1960s, William Stewart^[3] and Heinz Kloss introduced the basic concepts for the sociolinguistic theory of pluricentric languages, which describes how standard language varieties differ between nations (e.g. American/British/Canadian/Australian English;^[4] Austrian/German/Swiss German;^[5] Bosnian/Croatian/Montenegrin/Serbian Serbo-Croatian^[6]).

Applications of sociolinguistics

For example, a sociolinguist might determine through study of social attitudes that a particular vernacular would not be considered appropriate language use in a business or professional setting. Sociolinguists might also study the grammar, phonetics, vocabulary, and other aspects of this sociolect much as dialectologists would study the same for a regional dialect.

The study of language variation is concerned with social constraints determining language in its contextual environment. Code-switching is the term given to the use of different varieties of language in different social situations.

William Labov is often regarded as the founder of the study of sociolinguistics. He is especially noted for introducing the quantitative study of language variation and change,^[7] making the sociology of language into a scientific discipline.

Traditional sociolinguistic interview

Sociolinguistic interviews are an integral part of collecting data for sociolinguistic studies. There is an interviewer, who is conducting the study, and a subject, or informant, who is the interviewee. In order to get a grasp on a specific linguistic form and how it is used in the dialect of the subject, a variety of methods are used to elicit certain registers of speech. There are five different styles, ranging from formal to casual. The most formal style would be elicited by having the subject read a list of minimal pairs (MP). Minimal pairs are pairs of words that differ in only one phoneme, such as cat and bat. Having the subject read a word list (WL) will elicit a formal register, but generally not as formal as MP. The reading passage (RP) style is next down on the formal register, and the interview style (IS) is when an interviewer can finally get into eliciting a more casual speech from the subject. During the IS the interviewer can converse with the subject and try to draw out of them an even more casual sort of speech by asking him to recall childhood memories or maybe a near death experience, in which case the subject will get deeply

involved with the story since strong emotions are often attached to these memories. Of course, the most sought after type of speech is the casual style (CS). This type of speech is difficult if not impossible to elicit because of the Observer's Paradox. The closest one might come to CS in an interview is when the subject is interrupted by a close friend or family member, or perhaps must answer the phone. CS is used in a completely unmonitored environment where the subject feels most comfortable and will use their natural vernacular without overtly thinking about it.

Fundamental concepts in sociolinguistics

While the study of sociolinguistics is very broad, there are a few fundamental concepts on which many sociolinguistic inquiries depend.

Speech community

Speech community is a concept in sociolinguistics that describes a more or less discrete group of people who use language in a unique and mutually accepted way among themselves. This is sometimes referred to as a Sprechbund.

Speech communities can be members of a profession with a specialized jargon, distinct social groups like high school students or hip hop fans, or even tight-knit groups like families and friends. Members of speech communities will often develop slang or jargon to serve the group's special purposes and priorities.

High prestige and low prestige varieties

Crucial to sociolinguistic analysis is the concept of prestige; certain speech habits are assigned a positive or a negative value, which is then applied to the speaker. This can operate on many levels. It can be realised on the level of the individual sound/phoneme, as Labov discovered in investigating pronunciation of the post-vocalic /r/ in the North-Eastern USA, or on the macro scale of language choice, as realised in the various diglossias that exist throughout the world, where Swiss-German/High German is perhaps most well known. An important implication of sociolinguistic theory is that speakers 'choose' a variety when making a speech act, whether consciously or subconsciously.

Social network

Understanding language in society means that one also has to understand the social networks in which language is embedded. A social network is another way of describing a particular speech community in terms of relations between individual members in a community. A network could be *loose* or *tight* depending on how members interact with each other.^[8] For instance, an office or factory may be considered a tight community because all members interact with each other. A large course with 100+ students would be a looser community because students may only interact with the instructor and maybe 1–2 other students. A *multiplex* community is one in which members have multiple relationships with each other.^[8] For instance, in some neighborhoods, members may live on the same street, work for the same employer and even intermarry.

The looseness or tightness of a social network may affect speech patterns adopted by a speaker. For instance, Sylvie Dubois and Barbara Horvath found that speakers in one Cajun Louisiana community were more likely to pronounce English "th" [θ] as [t] (or [ð] as [d]) if they participated in a relatively dense social network (i.e. had strong local ties and interacted with many other speakers in the community), and less likely if their networks were looser (i.e. fewer local ties).^[9]

A social network may apply to the macro level of a country or a city, but also to the inter-personal level of neighborhoods or a single family. Recently, social networks have been formed by the Internet, through chat rooms, MySpace groups, organizations, and online dating services.

Internal vs. external language

In Chomskyan linguistics, a distinction is drawn between I-language (internal language) and E-language (external language). In this context, internal language is linguistic knowledge that a native speaker of language has. It applies to the study of syntax and semantics on the abstract level. External language applies to language in social contexts, i.e. behavioral habits shared by a community. Internal language analyses operate on the assumption that all native speakers of a language are quite homogeneous in how they process and perceive language. External language fields, such as sociolinguistics, attempt to explain why this is in fact not the case. Many sociolinguists reject the distinction between I- and E-language on the grounds that it is based on a mentalist view of language. On this view, grammar is first and foremost an interactional (social) phenomenon (e.g. Elinor Ochs, Emanuel Schegloff, Sandra Thompson).

Differences according to class

Sociolinguistics as a field distinct from dialectology was pioneered through the study of language variation in urban areas. Whereas dialectology studies the geographic distribution of language variation, sociolinguistics focuses on other sources of variation, among them class. Class and occupation are among the most important linguistic markers found in society. One of the fundamental findings of sociolinguistics, which has been hard to disprove, is that class and language variety are related. Members of the working class tend to speak less standard language, while the lower, middle, and upper middle class will in turn speak closer to the standard. However, the upper class, even members of the upper middle class, may often speak 'less' standard than the middle class. The looseness or tightness of a social network may affect speech patterns adopted by a speaker. For instance, Sylvie Dubois and Barbara Horvath found that speakers in one Cajun Louisiana community were more likely to pronounce English "th" [θ] as [t] (or [ð] as [d]) if they participated in a relatively dense social network (i.e. had strong local ties and interacted with many other speakers in the community), and less likely if their networks were looser (i.e. fewer local ties). This is because not only class, but class aspirations, are important.

Class aspiration

Studies, such as those by William Labov in the 1960s, have shown that social aspirations influence speech patterns. This is also true of class aspirations. In the process of wishing to be associated with a certain class (usually the upper class and upper middle class) people who are moving in that direction socio-economically will adjust their speech patterns to sound like them. However, not being native upper class speakers, they often hypercorrect, which involves overcorrecting their speech to the point of introducing new errors. The same is true for individuals moving down in socio-economic status.

Social language codes

Basil Bernstein, a well-known British socio-linguist, devised in his book, 'Elaborated and restricted codes: their social origins and some consequences,' a social code system he used to classify the various speech patterns for different social classes. He claimed that members of the middle class have ways of organizing their speech that are fundamentally very different from the ways adopted by the working class.

Restricted code

In Basil Bernstein's theory, the restricted code was an example of the speech patterns used by the working class. He stated that this type of code allows strong bonds between group members, who tend to behave largely on the basis of distinctions such as 'male', 'female', 'older', and 'younger'. This social group also uses language in a way that brings unity between people, and members often do not need to be explicit about meaning, as their shared knowledge and common understanding often bring them together in a way that other social language groups do not experience. The difference with the restricted code is the emphasis on 'we' as a social group, which fosters greater solidarity than an emphasis on 'I'. The time when "restricted-code" matters is the day when children start school where the standard

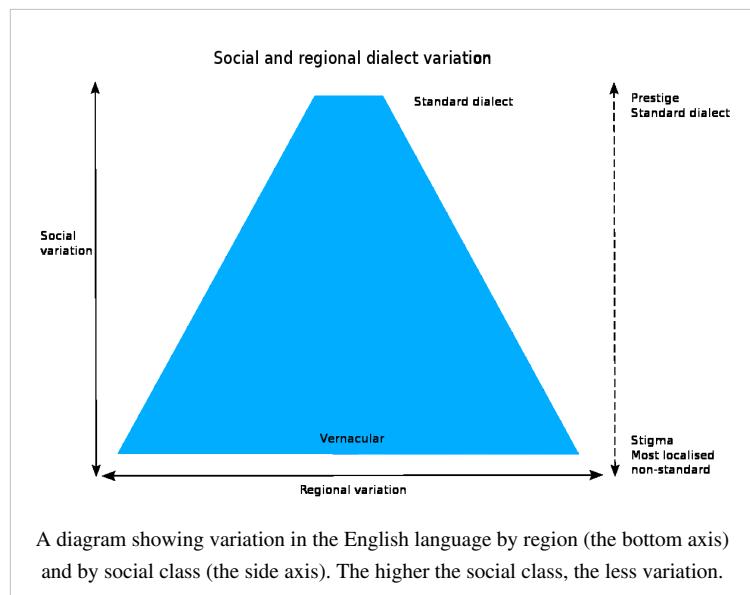
variety of language is used. Moreover, the written form of a language is already very different from the everyday form. Children with restricted-code, therefore, struggle at school more than those who speak an "elaborated-code". The type of communication used by the working class reminds Paivio's dual code theory. According to Paivio, there are two types of codes; verbal and non-verbal. The dual coding theory proposed by Paivio attempts to give equal weight to verbal and non-verbal processing. Paivio (1986) states: "Human cognition is unique in that it has become specialized for dealing simultaneously with language and with nonverbal objects and events. Moreover, the language system is peculiar in that it deals directly with linguistic input and output (in the form of speech or writing) while at the same time serving a symbolic function with respect to nonverbal objects, events, and behaviors. Any representational theory must accommodate this dual functionality." (p. 53). The use of context by members of working class to imply what they mean, therefore, may be a "non-verbal code". However, this type of communicative skills may not be understood by other children who belong to other classes. What's more, children with restricted-code may have difficulty in understanding the teacher, the only source of information for them at school. Therefore, it is suggested that working-class children should have pre-school training within their early childhood period. Early schooling may provide them with opportunities to acquire the way of speaking valid at school.

Elaborated code

Basil Bernstein also studied what he named the 'elaborated code' explaining that in this type of speech pattern the middle and upper classes use this language style to gain access to education and career advancement. Bonds within this social group are not as well defined and people achieve their social identity largely on the basis of individual disposition and temperament. There is no obvious division of tasks according to sex or age and generally, within this social formation members negotiate and achieve their roles, rather than have them there ready-made in advance. Due to the lack of solidarity the elaborated social language code requires individual intentions and viewpoints to be made explicit as the T has a greater emphasis with this social group than the working class.

Deviation from standard language varieties

The existence of differences in language between social classes can be illustrated by the following table:



Bristolian Dialect (lower class)	...	Standard English (higher class)
I ain't done nothing	...	I haven't done anything
I done it yesterday	...	I did it yesterday
It weren't me that done it	...	I didn't do it

Any native speaker of English would immediately be able to guess that *speaker 1* was likely of a different social class than *speaker 2*, namely from a lower social class, probably from a working class pedigree. The differences in grammar between the two examples of speech is referred to as differences between social class dialects or sociolects.

It is also notable that, at least in England and Australia, the closer to standard English a dialect gets, the less the lexicon varies by region, and vice versa.

Covert prestige

It is generally assumed that non-standard language is low-prestige language. However, in certain groups, such as traditional working-class neighborhoods, standard language may be considered undesirable in many contexts. This is because the working class dialect is a powerful in-group marker, and especially for non-mobile individuals, the use of non-standard varieties (even exaggeratedly so) expresses neighborhood pride and group and class solidarity. There will thus be a considerable difference in use of non-standard varieties when going to the pub or having a neighborhood barbecue (high), and going to the bank (lower) for the same individual.

Sociolinguistic variables

Studies in the field of sociolinguistics typically take a sample population and interview them, assessing the realisation of certain sociolinguistic variables.

A commonly studied source of variation is regional dialects. Dialectology studies variations in language based primarily on geographic distribution and their associated features. Sociolinguists concerned with grammatical and phonological features that correspond to regional areas are often called dialectologists.

There are several different types of age-based variation one may see within a population. They are: vernacular of a subgroup with membership typically characterized by a specific age range, age-graded variation, and indications of linguistic change in progress.

Variation may also be associated with gender. Men and women, on average, tend to use slightly different language styles. These differences tend to be quantitative rather than qualitative. That is, to say that women use a particular speaking style more than men do is akin to saying that men are taller than women (i.e., men are on average taller than women, but some women are taller than some men).

References

- [1] John J. Gumperz and Jenny Cook-Gumperz, "Studying language, culture, and society: Sociolinguistics or linguistic anthropology?". (<http://doi.wiley.com/10.1111/j.1467-9841.2008.00378.x>) *Journal of Sociolinguistics* 12(4), 2008: 532–545.
- [2] T. C. Hodson and the Origins of British Socio-linguistics by John E. Joseph (http://www.ncl.ac.uk/ss15/papers/paper_details.php?id=304) Sociolinguistics Symposium 15, Newcastle-upon-Tyne, April 2004
- [3] Stewart, William A (1968). "A Sociolinguistic Typology for Describing National Multilingualism". In Fishman, Joshua A. *Readings in the Sociology of Language*. The Hague, Paris: Mouton. p. 534. OCLC 306499.
- [4] Kloss, Heinz (1976). "Abstandssprachen und Ausbausprachen [Abstand-languages and Ausbau-languages]". In Göschel, Joachim; Nail, Norbert; van der Els, Gaston. *Zur Theorie des Dialekts: Aufsätze aus 100 Jahren Forschung*. Zeitschrift für Dialektologie und Linguistik, Beihefte, n.F., Heft 16. Wiesbaden: F. Steiner. p. 310. OCLC 2598722.
- [5] Ammon, Ulrich (1995) (in German). *Die deutsche Sprache in Deutschland, Österreich und der Schweiz: das Problem der nationalen Varietäten [German Language in Germany, Austria and Switzerland: The Problem of National Varieties]*. Berlin & New York: Walter de Gruyter. pp. 1–11. OCLC 33981055.

- [6] Kordić, Snježana (2010) (in Serbo-Croatian). *Jezik i nacionalizam [Language and Nationalism]* (<http://www.webcitation.org/690BiBe4T>). Rotulus Universitas. Zagreb: Durieux. pp. 77–90. ISBN 978-953-188-311-5. LCCN 2011520778. OCLC 729837512. OL{{1}}. Archived from the original (http://bib.irb.hr/datoteka/475567.Jezik_i_nacionalizam.pdf) on 8 July 2012.. Retrieved 17 July 2012.
- [7] Paolillo, John C. *Analyzing Linguistic Variation: Statistical Models and Methods* CSLI Press 2001, Tagliamonte, Sali *Analysing Sociolinguistic Variation* Cambridge, 2006
- [8] Wardhaugh, Ronald (2006), *An Introduction to Sociolinguistics*, New York: Wiley-Blackwell
- [9] Dubois, Sylvie and Horvath, Barbara. (1998). "Let's tink about dat: Interdental Fricatives in Cajun English." *Language Variation and Change* 10 (3), pp 245–61.

Further reading

- Chambers, J.K. (2010). *Sociolinguistic Theory. Linguistic Variation and its Social Significance*. Malden, MA: Blackwell Publishers. (A very elaborate book that brings the reader a detailed overview of the most important investigations that have been done in sociolinguistics, their results and the tendencies that can be derived from those. It also offers an overview of the structure vs. variability debate.)
- Kordić, Snježana (2009). "Plurizentrische Sprachen, Ausbausprachen, Abstandssprachen und die Serbokroatistik [Pluricentric languages, Ausbau languages, Abstand languages and the Serbo-Croatistics]" (<http://www.webcitation.org/69f5bCgpH>) (in German). *Zeitschrift für Balkanologie* (<http://www.zeitschrift-fuer-balkanologie.de/index.php/zfb/index>) **45** (2): 210–215. ISSN 0044-2356. Archived from the original (<http://www.zeitschrift-fuer-balkanologie.de/index.php/zfb/article/view/203/203>) on 4 August 2012. Retrieved 31 August 2012.
- Labov, W. (2001). *Principles of Linguistic Changes: Social Factors*. Malden, MA: Blackwell Publishers.
- Lakoff, Robin T. (2000). *The Language War*. Berkely, CA: University of California Press. ISBN 0-520-21666-0
- Meyerhoff, Miriam. (2006). *Introducing Sociolinguistics*. New York: Routledge. ISBN 0-415-39948-3
- Milroy, Lesley and Gordon. Matthew. (2003) *Sociolinguistics: Method and Interpretation* London: Blackwell Publishing. ISBN 0-631-22225-1. (More advanced, but has lots of good examples and describes research methodologies to use.)
- Paulston, Christina Bratt and G. Richard Tucker, editors. 1997. *The early days of sociolinguistics: memories and reflections*. (Publications in Sociolinguistics, 2.) Dallas: Summer Institute of Linguistics.
- Tagliamonte, S. (2006). *Analysing Sociolinguistic Variation*. Cambridge: Cambridge University Press. (An entry-level introduction to sociolinguistics that features a practical how-to guide for setting up an investigation.)
- Trudgill, Peter. (2000). *Sociolinguistics: An Introduction to Language and Society*(4th Ed.). London: Penguin Books. ISBN 0-14-028921-6 This book is a very readable, if Anglo-centric, introduction for the non-linguist.
- Watts, Richard J. (2003). *Politeness*. Cambridge: Cambridge University Press. ISBN 978-0-521-79406-0. A sociolinguistics book specializing in the research in politeness. It's a little tough at times, but very helpful and informational.

External links

- Applied Linguistics (http://www.dmoz.org/Science/Social_Sciences/Linguistics/Applied_Linguistics/) at the Open Directory Project
- <http://www.ncsu.edu/linguistics/ncllp/aboutfieldwork.php> About sociolinguistic fieldwork
- Sociolinguistics: an interview with William Labov (http://www.revel.inf.br/site2007/_pdf/9/entrevistas/revel_9_interview_labov.pdf) ReVEL, vol. 5, n. 9, 2007.

Linguistic universal

A **linguistic universal** is a pattern that occurs systematically across natural languages, potentially true for all of them. For example, *All languages have nouns and verbs*, or *If a language is spoken, it has consonants and vowels*. Research in this area of linguistics is closely tied to the study of linguistic typology, and intends to reveal generalizations across languages, likely tied to cognition, perception, or other abilities of the mind. The field was largely pioneered by the linguist Joseph Greenberg, who derived a set of forty-five basic universals, mostly dealing with syntax (see the list), from a study of some thirty languages.

Terminology

Linguists distinguish between two kinds of universals: **absolute** (opposite: **statistical**, often called **tendencies**) and **implicational** (opposite **non-implicational**). Absolute universals apply to every known language and are quite few in number; an example is *All languages have pronouns*. An implicational universal applies to languages with a particular feature that is always accompanied by another feature, such as *If a language has trial grammatical number, it also has dual grammatical number*, while non-implicational universals just state the existence (or non-existence) of one particular feature.

Also in contrast to absolute universals are **tendencies**, statements that may not be true for all languages, but nevertheless are far too common to be the result of chance.^[1] They also have implicational and non-implicational forms. An example of the latter would be *The vast majority of languages have nasal consonants*.^[2] However, most tendencies, like their universal counterparts, are implicational. For example, *With overwhelmingly-greater-than-chance frequency, languages with normal SOV order are postpositional*. Strictly speaking, a tendency is not a kind of universal, but exceptions to most statements called universals can be found. For example, Latin is an SOV language with prepositions. Often it turns out that these exceptional languages are undergoing a shift from one type of language to another. In the case of Latin, its descendant Romance languages switched to SVO, which is a much more common order among prepositional languages.

Universals may also be **bidirectional** or **unidirectional**. In a bidirectional universal two features each imply the existence of each other. For example, languages with postpositions usually have SOV order, and likewise SOV languages usually have postpositions. The implication works both ways, and thus the universal is bidirectional. By contrast, in a unidirectional universal the implication works only one way. Languages that place relative clauses before the noun they modify again usually have SOV order, so pre-nominal relative clauses imply SOV. On the other hand, SOV languages worldwide show little preference for pre-nominal relative clauses, and thus SOV implies little about the order of relative clauses. As the implication works only one way, the proposed universal is a unidirectional one.

Linguistic universals in syntax are sometimes held up as evidence for universal grammar (although epistemological arguments are more common). Other explanations for linguistic universals have been proposed, for example, that linguistic universals tend to be properties of language that aid communication. If a language were to lack one of these properties, it has been argued, it would probably soon evolve into a language having that property.

Michael Halliday has argued for a distinction between **descriptive** and **theoretical** categories in resolving the matter of the existence of linguistic universals, a distinction he takes from J.R. Firth and Louis Hjelmslev. He argues that "theoretical categories, and their inter-relations construe an abstract model of language...; they are interlocking and mutually defining". Descriptive categories, by contrast, are those set up to describe particular languages. He argues that "When people ask about "universals", they usually mean descriptive categories that are assumed to be found in all languages. The problem is there is no mechanism for deciding how much alike descriptive categories from different languages have to be before they are said to be "the same thing"^[3]

In semantics

In the domain of semantics, research into linguistic universals has taken place in a number of ways. Some linguists, starting with Leibniz, have pursued the search for a hypothetic irreducible semantic core of all languages. A modern variant of this approach can be found in the Natural Semantic Metalanguage of Wierzbicka and associates.^[4] Other lines of research suggest cross-linguistic tendencies to use body part terms metaphorically as adpositions,^[5] or tendencies to have morphologically simple words for cognitively salient concepts.^[6] The human body, being a physiological universal, provides an ideal domain for research into semantic and lexical universals. In a seminal study, Cecil H. Brown (1976) proposed a number of universals in the semantics of body part terminology, including the following: in any language, there will be distinct terms for BODY, HEAD, ARM, EYES, NOSE, and MOUTH; if there is a distinct term for FOOT, there will be a distinct term for HAND; similarly, if there are terms for INDIVIDUAL TOES, then there are terms for INDIVIDUAL FINGERS. Subsequent research has shown that most of these features have to be considered cross-linguistic tendencies rather than true universals. Several languages, for example Tidore and Kuuk Thaayorre, lack a general term meaning 'body'. On the basis of such data it has been argued that the highest level in the partonomy of body part terms would be the word for 'person'.^[7]

Notes

- [1] Dryer (1998)
- [2] Lushootseed and Rotokas are examples of the rare languages which truly lack nasal consonants as normal speech sounds.
- [3] Halliday, M.A.K. 2002. A personal perspective. In *On Grammar*, Volume 1 in the Collected Works of M.A.K. Halliday. London and New York: Continuum p12.
- [4] see for example Goddard & Wierzbicka (1994) and Goddard (2002).
- [5] Heine (1997)
- [6] Rosch et al. (1976)
- [7] Wilkins (1993), Enfield et al. 2006:17.

References

- Brown, Cecil H. (1976) "General principles of human anatomical partonomy and speculations on the growth of partonomic nomenclature." *American Ethnologist* 3, no. 3, Folk Biology, pp. 400–424
- Comrie, Bernard (1981) *Language Universals and Linguistic Typology*. Chicago: University of Chicago Press.
- Croft, W. (2002). *Typology and Universals*. Cambridge: Cambridge UP. 2nd ed. ISBN 0-521-00499-3
- Dryer, Matthew S. (1998) "Why Statistical Universals are Better Than Absolute Universals" Chicago Linguistic Society 33: The Panels, pp. 123-145.
- Enfield, Nick J. & Asifa Majid & Miriam van Staden (2006) 'Cross-linguistic categorisation of the body: Introduction' (special issue of *Language Sciences*).
- Ferguson, Charles A. (1968) 'Historical background of universals research'. In: Greenberg, Ferguson, & Moravcsik, *Universals of human languages*, pp. 7–31.
- Goddard, Cliff and Wierzbicka, Anna (eds.). 1994. *Semantic and Lexical Universals - Theory and Empirical Findings*. Amsterdam/Philadelphia: John Benjamins.
- Goddard, Cliff (2002) 'The search for the shared semantic core of all languages (http://www.une.edu.au/lcl/nsm/pdf/Goddard_Ch1_2002.pdf)'. In Goddard & Wierzbicka (eds.) *Meaning and Universal Grammar - Theory and Empirical Findings* volume 1, pp. 5–40, Amsterdam/Philadelphia: John Benjamins.
- Greenberg, Joseph H. (ed.) (1963) *Universals of Languages*. Cambridge, Mass.: MIT Press.
- Greenberg, Joseph H. (ed.) (1978a) *Universals of Human Language* Vol. 4: *Syntax*. Stanford, California: Stanford University Press.
- Greenberg, Joseph H. (ed.) (1978b) *Universals of Human Language* Vol. 3: *Word Structure*. Stanford, California: Stanford University Press.
- Heine, Bernd (1997) *Cognitive Foundations of Grammar*. New York/Oxford: Oxford University Press.

- Song, Jae Jung (2001) *Linguistic Typology: Morphology and Syntax*. Harlow, UK: Pearson Education (Longman).
- Rosch, E. & Mervis, C.B. & Gray, W.D. & Johnson, D.M. & Boyes-Braem, P. (1976) 'Basic Objects In Natural Categories', *Cognitive Psychology* 8-3, 382-439.
- Wilkins, David P. (1993) 'From part to person: natural tendencies of semantic change and the search for cognates', *Working paper No. 23*, Cognitive Anthropology Research Group at the Max Planck Institute for Psycholinguistics.

External links

- Some Universals of Grammar with Particular Reference to the Order of Meaningful Elements (<http://angli02.kgw.tu-berlin.de/Korean/Artikel02/>) by Joseph H. Greenberg
- The Universals Archive (<http://typo.uni-konstanz.de/archive/>) by the University of Konstanz

Grammar

Grammar

In linguistics, **grammar** is the set of structural rules that governs the composition of clauses, phrases, and words in any given natural language. The term refers also to the study of such rules, and this field includes morphology, syntax, and phonology, often complemented by phonetics, semantics, and pragmatics. Linguists do not normally use the term to refer to orthographical rules, although usage books and style guides that call themselves grammars may also refer to spelling and punctuation.

Use of the term

The term *grammar* is often used by non-linguists with a very broad meaning. As Jeremy Butterfield puts it: "Grammar is often a generic way of referring to any aspect of English that people object to."^[1] However, linguists use it in a much more specific sense. Speakers of a language have in their heads a set of rules^[2] for using that language. This is a grammar, and the vast majority of the information in it is acquired—at least in the case of one's native language—not by conscious study or instruction, but by observing other speakers; much of this work is done during infancy. Learning a language later in life usually involves a greater degree of explicit instruction.^[3]

The term "grammar" can also be used to describe the rules that govern the linguistic behaviour of a group of speakers. The term "English grammar", therefore, may have several meanings. It may refer to the whole of English grammar—that is, to the grammars of all the speakers of the language—in which case, the term encompasses a great deal of variation.^[4] Alternatively, it may refer only to what is common to the grammars of all, or of the vast majority of English speakers (such as subject–verb–object word order in simple declarative sentences). Or it may refer to the rules of a particular, relatively well-defined variety of English (such as Standard English).

"An English grammar" is a specific description, study or analysis of such rules. A reference book describing the grammar of a language is called a "reference grammar" or simply "a grammar." A fully explicit grammar that exhaustively describes the grammatical constructions of a language is called a descriptive grammar. This kind of linguistic description contrasts with linguistic prescription, an attempt to discourage or suppress some grammatical constructions, while promoting others. For example, preposition stranding occurs widely in Germanic languages and has a long history in English. John Dryden, however, objected to it (without explanation),^[5] leading other English speakers to avoid the construction and discourage its use.^[6]

Etymology

The word *grammar* derives from Greek γραμματικὴ τέχνη (*grammatikē technē*), which means "art of letters", from γράμμα (*gramma*), "letter", itself from γράφειν (*graphein*), "to draw, to write".^[7]

History

The first systematic grammars originated in Iron Age India, with Yaska (6th c. BC), Pāṇini (4th c. BC) and his commentators Pingala (ca. 200 BC), Katyayana, and Patanjali (2nd c. BC). In the West, grammar emerged as a discipline in Hellenism from the 3rd c. BC forward with authors like Rhyanus and Aristarchus of Samothrace, the oldest extant work being the *Art of Grammar* (Τέχνη Γραμματική), attributed to Dionysius Thrax (ca. 100 BC). Latin grammar developed by following Greek models from the 1st century BC, due to the work of authors such as Orbilius Pupillus, Remmius Palaemon, Marcus Valerius Probus, Verrius Flaccus, and Aemilius Asper.

Tolkāppiyam is the earliest Tamil grammar; it has been dated variously between 1st CE and 10th CE.

A grammar of Irish originated in the 7th century with the *Auraicept na n-Éces*.

Arabic grammar emerged with Abu al-Aswad al-Du'ali from the 7th century who in-turn was taught the discipline by Ali ibn Abi talib, the fourth historical caliph of Islam and first Imam for Shi'i Muslims.

The first treatises on Hebrew grammar appeared in the High Middle Ages, in the context of Mishnah (exegesis of the Hebrew Bible). The Karaite tradition originated in Abbasid Baghdad. The *Diqduq* (10th century) is one of the earliest grammatical commentaries on the Hebrew Bible.^[8] Ibn Barun in the 12th century compares the Hebrew language with Arabic in the Islamic grammatical tradition.^[9]

Belonging to the *trivium* of the seven liberal arts, grammar was taught as a core discipline throughout the Middle Ages, following the influence of authors from Late Antiquity, such as Priscian. Treatment of vernaculars began gradually during the High Middle Ages, with isolated works such as the First Grammatical Treatise, but became influential only in the Renaissance and Baroque periods. In 1486, Antonio de Nebrija published *Las introducciones Latinas contrapuesto el romance al Latin*, and the first Spanish grammar, *Gramática de la lengua castellana*, in 1492. During the 16th century Italian Renaissance, the *Questione della lingua* was the discussion on the status and ideal form of the Italian language, initiated by Dante's *de vulgari eloquentia* (Pietro Bembo, *Prose della volgar lingua* Venice 1525). The first grammar of Slovene language was written in 1584 by Adam Bohorič.

Grammars of non-European languages began to be compiled for the purposes of evangelization and Bible translation from the 16th century onward, such as *Grammatica o Arte de la Lengua General de los Indios de los Reynos del Perú* (1560), and a Quechua grammar by Fray Domingo de Santo Tomás.

In 1643 there appeared Ivan Uzhevych's *Grammatica sclavonica* and, in 1762, the *Short Introduction to English Grammar* of Robert Lowth was also published. The *Grammatisch-Kritisches Wörterbuch der hochdeutschen Mundart*, a High German grammar in five volumes by Johann Christoph Adelung, appeared as early as 1774.

From the latter part of the 18th century, grammar came to be understood as a subfield of the emerging discipline of modern linguistics. The Serbian grammar by Vuk Stefanović Karadžić arrived in 1814, while the *Deutsche Grammatik* of the Brothers Grimm was first published in 1818. The *Comparative Grammar* of Franz Bopp, the starting point of modern comparative linguistics, came out in 1833.

Development of grammars

Grammars evolve through usage and also due to separations of the human population. With the advent of written representations, formal rules about language usage tend to appear also. **Formal grammars** are codifications of usage that are developed by repeated documentation over time, and by observation as well. As the rules become established and developed, the prescriptive concept of grammatical correctness can arise. This often creates a discrepancy between contemporary usage and that which has been accepted, over time, as being correct. Linguists tend to view prescriptive grammars as having little justification beyond their authors' aesthetic tastes, although style guides may give useful advice about *standard language employment*, based on descriptions of usage in contemporary writings of the same language. **Linguistic prescriptions** also form part of the explanation for variation in speech, particularly variation in the speech of an individual speaker (an explanation, for example, for why some people say, "I didn't do nothing"; some say, "I didn't do anything"; and some say one or the other depending on social context).

The formal study of grammar is an important part of education for children from a young age through advanced learning, though the rules taught in schools are not a "grammar" in the sense most linguists use the term, particularly as they are often prescriptive rather than descriptive.

Constructed languages (also called *planned languages* or *conlangs*) are more common in the modern day. Many have been designed to aid human communication (for example, naturalistic Interlingua, schematic Esperanto, and the highly logic-compatible artificial language Lojban). Each of these languages has its own grammar.

Syntax refers to linguistic structure above the word level (e.g. how sentences are formed)—though without taking into account intonation, which is the domain of phonology. Morphology, by contrast, refers to structure at and below the word level (e.g. how compound words are formed), but above the level of individual sounds, which, like intonation, are in the domain of phonology.^[10] No clear line can be drawn, however, between syntax and morphology. Analytic languages use syntax to convey information that is encoded via inflection in synthetic languages. In other words, word order is not significant and morphology is highly significant in a purely synthetic language, whereas morphology is not significant and syntax is highly significant in an analytic language. Chinese and Afrikaans, for example, are highly analytic, and meaning is therefore very context-dependent. (Both do have some inflections, and have had more in the past; thus, they are becoming even less synthetic and more "purely" analytic over time.) Latin, which is highly synthetic, uses affixes and inflections to convey the same information that Chinese does with syntax. Because Latin words are quite (though not completely) self-contained, an intelligible Latin sentence can be made from elements that are placed in a largely arbitrary order. Latin has a complex affixation and simple syntax, while Chinese has the opposite.

Grammar frameworks

Various "grammar frameworks" have been developed in theoretical linguistics since the mid 20th century, in particular under the influence of the idea of a "universal grammar" in the United States. Of these, the main divisions are:

- Transformational grammar (TG)
- Systemic functional grammar (SFG)
- Principles and Parameters Theory (P&P)
- Lexical-functional Grammar (LFG)
- Generalized Phrase Structure Grammar (GPSG)
- Head-Driven Phrase Structure Grammar (HPSG)
- Dependency grammars (DG)
- Role and reference grammar (RRG)

Education

Prescriptive grammar is taught in primary school (elementary school). The term "grammar school" historically refers to a school teaching Latin grammar to future Roman citizens, orators, and, later, Catholic priests. In its earliest form, "grammar school" referred to a school that taught students to read, scan, interpret, and declaim Greek and Latin poets (including Homer, Virgil, Euripides, Ennius, and others). These should not be confused with the related, albeit distinct, modern British grammar schools.

A standard language is a particular dialect of a language that is promoted above other dialects in writing, education, and broadly speaking in the public sphere; it contrasts with vernacular dialects, which may be the objects of study in descriptive grammar but which are rarely taught prescriptively. The standardized "first language" taught in primary education may be subject to political controversy, since it establishes a standard defining nationality or ethnicity.

Recently, efforts have begun to update grammar instruction in primary and secondary education. The primary focus has been to prevent the use of outdated prescriptive rules in favor of more accurate descriptive ones and to change perceptions about relative "correctness" of standard forms in comparison to non standard dialects.

The pre-eminence of Parisian French has reigned largely unchallenged throughout the history of modern French literature. Standard Italian is not based on the speech of the capital, Rome, but on the speech of Florence because of the influence Florentines had on early Italian literature. Similarly, standard Spanish is not based on the speech of Madrid, but on the one of educated speakers from more northerly areas like Castile and León. In Argentina and Uruguay the Spanish standard is based on the local dialects of Buenos Aires and Montevideo (Rioplatense Spanish).

Portuguese has for now two official written standards, respectively Brazilian Portuguese and European Portuguese, but in a short term it will have a unified orthography.^[11]

The Serbian language is divided in a similar way; Serbia and the Republika Srpska use their own separate standards. The existence of a third standard is a matter of controversy, some consider Montenegrin as a separate language, and some think it's merely another variety of Serbian.

Norwegian has two standards, *Bokmål* and *Nynorsk*, the choice between which is subject to controversy: Each Norwegian municipality can declare one of the two its official language, or it can remain "language neutral". *Nynorsk* is endorsed by a minority of 27 percent of the municipalities. The main language used in primary schools normally follows the official language of its municipality, and is decided by referendum within the local school district. Standard German emerged out of the standardized chancellery use of High German in the 16th and 17th centuries. Until about 1800, it was almost entirely a written language, but now it is so widely spoken that most of the former German dialects are nearly extinct.

Standard Chinese has official status as the standard spoken form of the Chinese language in the People's Republic of China (PRC), the Republic of China (ROC) and the Republic of Singapore. Pronunciation of Standard Chinese is based on the Beijing dialect of Mandarin Chinese, while grammar and syntax are based on modern vernacular written Chinese. Modern Standard Arabic is directly based on Classical Arabic, the language of the Qur'an. The Hindustani language has two standards, Hindi and Urdu.

In the United States, the Society for the Promotion of Good Grammar designated March 4 as National Grammar Day in 2008.^[12]

Notes and references

- [1] Jeremy Butterfield, (2008) *Damp Squid: The English Language Laid Bare*, Oxford University Press, Oxford. 978-0-19-923906. p. 142.
- [2] Traditionally, the mental information used to produce and process linguistic utterances is referred to as "rules." However, other frameworks employ different terminology, with theoretical implications. Optimality theory, for example, talks in terms of "constraints", while Construction grammar, Cognitive grammar, and other "usage-based" theories make reference to patterns, constructions, and "schemata"
- [3] O'Grady, William; Dobrovolsky, Michael; Katamba, Francis (1996). *Contemporary Linguistics: An Introduction* (<http://books.google.co.uk/books?id=djhsAAAAIAAJ&q=Contemporary+Linguistics&dq=Contemporary+Linguistics>). Harlow, Essex: Longman. pp. 4–7; 464–539. .
- [4] Holmes, Janet (2001). *An Introduction to Sociolinguistics* (<http://books.google.co.uk/books?id=qjdqxecifHcC&printsec=frontcover&dq=Introduction+to+Sociolinguistics+Holmes>) (second ed.). Harlow, Essex: Longman. pp. 73–94. ; for more discussion of sets of grammars as populations, see: Croft, William (2000). *Explaining Language Change: An Evolutionary Approach* (http://books.google.co.uk/books?id=5_Ka7zLi9HQc&printsec=frontcover&dq=Explaining+Language+Change+Croft). Harlow, Essex: Longman. pp. 13–20. .
- [5] Rodney Huddleston and Geoffrey K. Pullum, 2002, The Cambridge Grammar of the English Language. Cambridge (UK): Cambridge University Press, p. 627f.
- [6] Lundin, Leigh (2007-09-23). "The Power of Prepositions" (<http://criminalbrief.com/?p=216>). *On Writing*. Cairo: Criminal Brief. .
- [7] Harper, Douglas, "Grammar" (<http://www.etymonline.com/index.php?term=grammar>), *Online Etymological Dictionary*, , retrieved 8 April 2010
- [8] G. Khan, J. B. Noah, *The Early Karaite Tradition of Hebrew Grammatical Thought* (2000)
- [9] Pinchas Wechter, Ibn Barūn's Arabic Works on Hebrew Grammar and Lexicography (1964)
- [10] Gussenoven, Carlos; Jacobs, Haike (2005). *Understanding Phonology* (http://books.google.co.uk/books?id=gHp_QgAACAAJ&dq=Understanding+Phonology&cd=1) (second ed.). London: Hodder Arnold.
- [11] (<http://www.languagesandnumbers.com/how-to-count-in-portuguese-brazil/en/por-bra/>)
- [12] National Grammar Day (<http://nationalgrammarday.com/>)
- American Academic Press, The (ed.). William Strunk, Jr., et al. *The Classics of Style: The Fundamentals of Language Style From Our American Craftsmen*. Cleveland: The American Academic Press, 2006. ISBN 0-9787282-0-3.
- Rundle, Bede. *Grammar in Philosophy*. Oxford: Clarendon Press; New York: Oxford University Press, 1979. ISBN 0-19-824612-9.

External links

- Archibald Henry Sayce (1911). "Grammar". In Chisholm, Hugh. *Encyclopædia Britannica* (11th ed.). Cambridge University Press.
- GrammarBank : Grammar rules explanations with examples and exercises online (<http://www.grammarbank.com>)
- The syntax of natural language: An online introduction using the Trees program (<http://www.ling.upenn.edu/~beatrice/syntax-textbook>) -- Beatrice Santorini & Anthony Kroch, University of Pennsylvania, 2007
- The Grammar Vandal (Funny, informative blog that fixes bad grammar.) (<http://thegrammarvandal.wordpress.com/>)
- The "Blog" of "Unnecessary" Quotes (Another educational, still funny poke at people who incorrectly use quote marks.) (<http://www.unnecessaryquotes.com/>)

English grammar

English grammar is the body of rules that describe the structure of expressions in the English language. This includes the structure of words, phrases, clauses, and sentences.

There are historical, social, and regional variations of English. Divergences from the grammar described here occur in some dialects of English. This article describes a generalized present-day Standard English, the form of speech found in types of public discourse including broadcasting, education, entertainment, government, and news reporting, including both formal and informal speech. There are certain differences in grammar between the standard forms of British English, American English, and Australian English, although these are inconspicuous compared with the lexical and pronunciation differences.

Word classes and phrases

There are eight word classes, or parts of speech, that are distinguished in English: nouns, determiners, pronouns, verbs, adjectives, adverbs, prepositions, and conjunctions. (Determiners, traditionally classified along with adjectives, have not always been regarded as a separate part of speech.) Interjections are another word class, but these are not described here as they do not form part of the clause and sentence structure of the language.^[1]

Nouns, verbs, adjectives, and adverbs form open classes – word classes that readily accept new members, such as the noun *celebutante* (a celebrity who frequents the fashion circles), the adverb *24/7* (as in *I am working on it 24/7*), and similar relatively new words.^[1] The others are regarded as closed classes. For example, it is rare for a new pronoun to be admitted to the language.

English words are not generally marked for word class. It is not usually possible to tell from the form of a word which class it belongs to except, to some extent, in the case of words with inflectional endings or derivational suffixes. On the other hand, some words belong to more than one word class. For example *run* can serve as either a verb or a noun (these are regarded as two different lexemes).^[1] Lexemes may be inflected to express different grammatical categories. The lexeme *run* has the forms *runs*, *ran*, and *running*.^[1] Words in one class can sometimes be derived from those in another. This has the potential to give rise to new words. The noun *aerobics* has recently given rise to the adjective *aerobicized*.^[1]

Words combine to form phrases. A phrase typically serves the same function as a word from some particular word class.^[1] For example, *my very good friend Peter* is a phrase that can be used in a sentence as if it were a noun, and is therefore called a noun phrase. Similarly, adjective phrases and adverb phrases function as if they were adjectives or adverbs, but with other types of phrases the terminology has different implications. For example, a verb phrase consists of a verb together with any objects and other dependents; a prepositional phrase consists of a preposition together with its complement (and is therefore usually a type of adverb phrase); and a determiner phrase is a type of

noun phrase containing a determiner.

Nouns

Nouns form the largest English word class. There are many common suffixes used to form nouns from other nouns or from other types of words, such as *-age* (as in *shrinkage*), *-hood* (as in *sisterhood*), and so on,^[1] although many nouns are base forms not containing any such suffix (such as *cat*, *grass*, *France*). Nouns are also often created by conversion of verbs or adjectives, as with the words *talk* and *reading* (*a boring talk*, *the assigned reading*).

Unlike in many related languages, English nouns do not have grammatical gender (although many nouns refer specifically to male or female persons or animals, like *mother*, *father*, *bull*, *tigress*; see Gender in English). Nouns are sometimes classified semantically (by their meanings) as proper nouns and common nouns (*Cyrus*, *China* vs. *frog*, *milk*) or as concrete nouns and abstract nouns (*book*, *laptop* vs. *heat*, *prejudice*).^[1] A grammatical distinction is often made between count (countable) nouns such as *clock* and *city*, and non-count (uncountable) nouns such as *milk* and *decor*.^[1] Some nouns can function to be either countable or uncountable such the word "wine" (*This is a good wine*, *I prefer red wine*).

Countable nouns generally have singular and plural forms.^[1] In most cases the plural is formed from the singular by adding *-[e]s* (as in *dogs*, *bushes*), although there are also irregular forms (*woman/women*, *medium/media*, etc.), including cases where the two forms are identical (*sheep*, *series*). For more details, see English plural.

Certain nouns can take plural verbs even though they are singular in form, as in *The government were ...* (where *the government* is considered to refer to the people constituting the government). This, a form of synesis, is more common in British than American English. See English plural: Singulars with collective meaning treated as plural.

English nouns are not marked for case as they are in some languages, but they have possessive forms, formed by the addition of *'s* (as in *John's*, *children's*), or just an apostrophe (with no change in pronunciation) in the case of *-[e]s* plurals and sometimes other words ending with *-s* (*the dogs' owners*, *Jesus' love*). More generally, the ending can be applied to noun phrases (as in *the man you saw yesterday's sister*); see below. The possessive form can be used either as a determiner (*John's cat*) or as a noun phrase (*John's is the one next to Jane's*). For details, see English possessive.

Noun phrases

Noun phrases are phrases that function grammatically as nouns within sentences, for example as the subject or object of a verb. Most noun phrases have a noun as their head.^[1]

An English noun phrase typically takes the following form (not all elements need be present):

Determiner + Pre-modifiers + NOUN + Postmodifiers/Complement

In this structure:

- the *determiner* may be an article (*the*, *a[n]*) or other equivalent word, as described in the following section. In many contexts it is required for a noun phrase to include some determiner.
- *pre-modifiers* include adjectives and some adjective phrases (such as *red*, *really lovely*), and noun adjuncts (such as *college* in the phrase *the college student*). Adjectival modifiers usually come before noun adjuncts.
- a *complement* or *postmodifier*^[1] may be a prepositional phrase (... of *London*), a relative clause (like ...which we saw yesterday), certain adjective or participial phrases (... sitting on the beach), or a dependent clause or infinitive phrase appropriate to the noun (like ... that the world is round after a noun such as *fact* or *statement*, or ... to travel widely after a noun such as *desire*).

An example of a noun phrase that includes all of the above-mentioned elements is *that rather attractive young college student that you were talking to*. Here *that* is the determiner, *rather attractive* and *young* are adjectival pre-modifiers, *college* is a noun adjunct, *student* is the noun serving as the head of the phrase, and *that you were talking to* is a post-modifier (a relative clause in this case). Notice the order of the pre-modifiers; the determiner *that*

must come first and the noun adjunct *college* must come after the adjectival modifiers.

Coordinating conjunctions such as *and*, *or*, and *but* can be used at various levels in noun phrases, as in *John, Paul, and Mary; the matching green coat and hat; a dangerous but exciting ride; a person sitting down or standing up*. See Conjunctions below for more explanation.

Noun phrases can also be placed in *apposition* (where two consecutive phrases refer to the same thing), as in *that president, Abraham Lincoln, ...* (where *that president* and *Abraham Lincoln* are in apposition). In some contexts the same can be expressed by a prepositional phrase, as in *the twin curses of famine and pestilence* (meaning "the twin curses" that are "famine and pestilence").

Particular forms of noun phrases include:

- phrases formed by the determiner *the* with an adjective, as in *the homeless, the English* (these are plural phrases referring to homeless people or English people in general);
- phrases with a pronoun rather than a noun as the head (see below);
- phrases consisting just of a possessive;
- infinitive and gerund phrases, in certain positions;
- certain clauses, such as *that* clauses and relative clauses like *what he said*, in certain positions.

Determiners

English determiners constitute a relatively small class of words. They include the articles *the, a[n]* (and in some contexts *some*), certain demonstrative and interrogative words such as *this, that, and which*, possessives such as *my* and *whose* (the role of determiner can also be played by noun possessive forms such as *John's* and *the girl's*), various quantifying words like *all, many, various*, and numerals (*one, two, etc.*). There are also many phrases (such as *a couple of*) that can play the role of determiners.

Determiners are used in the formation of noun phrases (see above). Many words that serve as determiners can also be used as pronouns (*this, that, many, etc.*)

Determiners can be used in certain combinations, such as *all the water* and *the many problems*.

In many contexts, it is required for a noun phrase to be completed with an article or some other determiner. It is not grammatical to say just *cat sat on table*; one must say *my cat sat on the table*. The most common situations in which a complete noun phrase can be formed without a determiner are when it refers generally to a whole class or concept (as in *dogs are dangerous* and *beauty is subjective*) and when it is a name (*Jane, Spain, etc.*) This is discussed in more detail at English articles and Zero article in English.

Pronouns

Pronouns are a relatively small, closed class of words that function in the place of nouns or noun phrases. They include personal pronouns, demonstrative pronouns, relative pronouns, interrogative pronouns, and some others, mainly indefinite pronouns.

Personal pronouns

The personal pronouns of modern standard English, and the corresponding possessive forms, are as follows:

	Nominative	Oblique	Reflexive	Possessive determiner	Possessive pronoun
1st pers. sing.	<i>I</i>	<i>me</i>	<i>myself</i>	<i>my</i>	<i>mine</i>
2nd pers. sing./pl.	<i>you</i>	<i>you</i>	<i>yourself/yourselves</i>	<i>your</i>	<i>yours</i>
3rd pers. sing.	<i>she, he, it</i>	<i>her, him, it</i>	<i>herself, himself, itself</i>	<i>her, his, its</i>	<i>hers, his, (rare: its)</i>
1st pers. pl.	<i>we</i>	<i>us</i>	<i>ourselves</i>	<i>our</i>	<i>ours</i>
3rd pers. pl.	<i>they</i>	<i>them</i>	<i>themselves</i>	<i>their</i>	<i>theirs</i>

The second-person forms such as *you* are used with both singular and plural reference. (An archaic set of pronouns used for singular reference is *thou, thee, thyself, thy, thine*.) *You* can also be used as an indefinite pronoun, referring to a person in general (see generic *you*) compared to the more formal alternative, *one* (reflexive *oneself*, possessive *one's*).

The third-person singular forms are differentiated according to the sex of the referent. For example, *she* can be used to refer to a female person, sometimes a female animal, and sometimes an object to which female characteristics are attributed, such as a ship or a country. A male person, and sometimes a male animal, is referred to using *he*. In other cases *it* can be used. (See Gender in English.) The word *it* can also be used as a dummy subject, in sentences like *It is going to be sunny this afternoon*.

The third-person plural forms such as *they* are sometimes used with singular reference, as a gender-neutral pronoun, as in *each employee should ensure they tidy their desk*. This usage is sometimes considered ungrammatical. (See singular *they*.)

The possessive determiners such as *my* are used as determiners together with nouns, as in *my old man, some of his friends*. The second possessive forms like *mine* are used when they do not qualify a noun: as pronouns, as in *mine is bigger than yours*, and as predicates, as in *this one is mine*. Note also the construction *a friend of mine* (meaning "someone who is my friend"). See English possessive for more details.

Demonstrative and interrogative pronouns

The demonstrative pronouns of English are *this* (plural *these*), and *that* (plural *those*), as in *these are good, I like that*. Note that all four words can also be used as determiners (followed by a noun), as in *those cars*. They can also then form the alternative pronominal expressions *this/that one, these/those ones*.

The interrogative pronouns are *who, what, and which* (all of them can take the suffix *-ever* for emphasis). The pronoun *who* refers to a person or people; it has an oblique form *whom* (though in informal contexts this is usually replaced by *who*), and a possessive form (pronoun or determiner) *whose*. The pronoun *what* refers to things or abstracts. The word *which* is used to ask about alternatives from what is seen as a closed set: *which (of the books) do you like best?* (It can also be an interrogative determiner: *which book?*; this can form the alternative pronominal expressions *which one* and *which ones*.) *Which, who, and what* can be either singular or plural, although *who* and *what* often take a singular verb regardless of any supposed number. For more information see *who*.

All the interrogative pronouns can also be used as relative pronouns; see below for more details.

Relative pronouns

The main relative pronouns in English are *who* (with its derived forms *whom* and *whose*), *which*, and *that*.^[2]

The relative pronoun *which* refers to things rather than persons, as in *the shirt, which used to be red, is faded*. For persons, *who* is used (*the man who saw me was tall*). The oblique case form of *who* is *whom*, as in *the man whom I saw was tall*, although in informal registers *who* is commonly used in place of *whom*.

The possessive form of *who* is *whose* (*the man whose car is missing ...*); however the use of *whose* is not restricted to persons (one can say *an idea whose time has come*).

The word *that* as a relative pronoun is normally found only in restrictive relative clauses (unlike *which* and *who*, which can be used in both restrictive and unrestricted clauses). It can refer to either persons or things, and cannot follow a preposition. For example, one can say *the song that [or which] I listened to yesterday*, but *the song to which [not to that] I listened yesterday*. The relative pronoun *that* is usually pronounced with a reduced vowel (schwa), and hence differently from the demonstrative *that* (see Weak and strong forms in English). If *that* is not the subject of the relative clause, it can be omitted (*the song I listened to yesterday*).

The word *what* can be used to form a free relative clause – one that has no antecedent and that serves as a complete noun phrase in itself, as in *I like what he likes*. The words *whatever* and *whichever* can be used similarly, in the role of either pronouns (*whatever he likes*) or determiners (*whatever book he likes*). When referring to persons, *who(ever)* (and *whom(ever)*) can be used in a similar way (but not as determiners).

There as pronoun

The word *there* is used as a pronoun in some sentences, playing the role of a dummy subject, normally of an intransitive verb. The "logical subject" of the verb then appears as a complement after the verb.

This use of *there* occurs most commonly with forms of the verb *be* in existential clauses, to refer to the presence or existence of something. For example: *There is a heaven; There are two cups on the table; There have been a lot of problems lately*. It can also be used with other verbs: *There exist two major variants; There occurred a very strange incident*.

The dummy subject takes the number (singular or plural) of the logical subject (complement), hence it takes a plural verb if the complement is plural. In colloquial English, however, the contraction *there's* is often used where *there are* would be expected.

The dummy subject can undergo inversion, *Is there a test today?* and *Never has there been a man such as this*. It can also appear without a corresponding logical subject, in short sentences and question tags: *There wasn't a discussion, was there? There was*.

The word *there* in such sentences has sometimes been analyzed as an adverb, or as a dummy predicate, rather than as a pronoun.^[3] However, its identification as a pronoun is most consistent with its behavior in inverted sentences and question tags as described above.

Because the word *there* can also be a deictic adverb (meaning "at/to that place"), a sentence like *There is a river* could have either of two meanings: "a river exists" (with *there* as a pronoun), and "a river is in that place" (with *there* as an adverb). In speech, the adverbial *there* would be given stress, while the pronoun would not – in fact the pronoun is often pronounced as a weak form, /ðə(r)/.

Other pronouns

Other pronouns in English are often identical in form to determiners (especially quantifiers), such as *many*, *a little*, etc. Sometimes the pronoun form is different, as with *none* (corresponding to the determiner *no*), *nothing*, *everyone*, *somebody*, etc. Many examples are listed at Indefinite pronoun.

Verbs

Verbs form the second largest word class after nouns. The basic form of an English verb is not generally marked by any ending, although there are certain suffixes that are frequently used to form verbs, such as *-ate* (*formulate*), *-fy* (*electrify*), and *-ise/ize* (*realise/realize*).^[1] Many verbs also contain prefixes, such *un-* (*unmask*), *out-* (*outlast*), *over-* (*overtake*), and *under-* (*undervalue*).^[1] Verbs can also be formed from nouns and adjectives by conversion, as with the verbs *snare*, *nose*, *dry*, and *calm*.

Most verbs have three or four inflected forms: a third-person singular present tense form in *-(e)s* (*writes*, *botches*), a present participle and gerund form in *-ing* (*writing*), a past tense (*wrote*), and – though often identical to the past tense form – a past participle (*written*). Regular verbs have identical past tense and past participle forms in *-ed*, but there are 100 or so irregular English verbs with different forms (see list). The verbs *have*, *do* and *say* also have irregular third-person present tense forms (*has*, *does* /dʌz/, *says* /sεz/). The verb *be* has the largest number of irregular forms (*am*, *is*, *are* in the present tense, *was*, *were* in the past tense, *been* for the past participle).

Most of what are often referred to as verb tenses (or sometimes aspects) in English are formed using auxiliary verbs. Apart from what are called the simple present (*write*, *writes*) and simple past (*wrote*), there are also continuous (progressive) forms (*am/is/are/was/were writing*), perfect forms (*have/has/had written*, and the perfect continuous *have/has/had been writing*), future forms (*will write*, *will be writing*, *will have written*, *will have been writing*), and conditionals (also called "future in the past") with *would* in place of *will*. The auxiliaries *shall* and *should* sometimes replace *will* and *would* in the first person. For the uses of these various verb forms, see English verbs and English clause syntax.

The infinitive is the basic form of the verb (*be*, *write*, *play*), although there is also a "to-infinitive" (*to be*, *to write*, *to play*) used in many syntactical constructions. There are also infinitives corresponding to other aspects: (*to*) *have written*, (*to*) *be writing*, (*to*) *have been writing*. The second-person imperative is identical to the (basic) infinitive; other imperative forms may be made with *let* (*let us go*, or *let's go*; *let them eat cake*).

A form identical to the infinitive can be used as a present subjunctive in certain contexts: *It is important that he follow them or ... that he be committed to the cause*. There is also a past subjunctive (distinct from the simple past only in the possible use of *were* instead of *was*), used in some conditional sentences and similar: *if I were* (or *was*) *rich ...; were he to arrive now ...; I wish she were* (or *was*) *here*. For details see English subjunctive.

The passive voice is formed using the verb *be* (in the appropriate tense or form) with the past participle of the verb in question: *cars are driven*, *he was killed*, *I am being tickled*, *it is nice to be pampered*, etc. The performer of the action may be introduced in a prepositional phrase with *by* (as in *they were killed by the invaders*).

The English modal verbs consist of the core modals *can*, *could*, *may*, *might*, *must*, *shall*, *should*, *will*, *would*, as well as *ought* (*to*), *had better*, and in some uses *dare* and *need*.^[1] These do not inflect for person or number,^[1] and do not have infinitive or participle forms (except synonyms, as with *be/being/been able* (*to*) for the modals *can/could*). The modals are used with the basic infinitive form of a verb (*I can swim*, *he may be killed*, *we dare not move*, *need they go?*), except for *ought*, which takes *to* (*you ought to go*).

The copula *be*, along with the modal verbs and the other auxiliaries, form a distinct class, sometimes called "special verbs" or simply "auxiliaries".^[4] These have different syntax from ordinary lexical verbs, especially in that they make their interrogative forms by plain inversion with the subject, and their negative forms by adding *not* after the verb (*could I ...? I could not ...*). Apart from those already mentioned, this class may also include *used to* (although the forms *did he use to?* and *he didn't use to* are also found), and sometimes *have* even when not an auxiliary (forms

like *have you a sister?* and *he hadn't a clue* are possible, though becoming less common). It also includes the auxiliary *do* (*does, did*); this is used with the basic infinitive of other verbs (those not belonging to the "special verbs" class) to make their question and negation forms, as well as emphatic forms (*do I like you?; he doesn't speak English; we did close the fridge*). For more details of this, see *do-support*.

Some forms of the copula and auxiliaries often appear as contractions, as in *I'm* for *I am*, *you'd* for *you would* or *you had*, and *John's* for *John is*. Their negated forms with following *not* are also often contracted (see Negation below). For detail see English auxiliaries and contractions.

Verb phrases

A verb together with its dependents, excluding its subject, may be identified as a verb phrase (although this concept is not acknowledged in all theories of grammar^[5]). A verb phrase headed by a finite verb may also be called a predicate. The dependents may be objects, complements, and modifiers (adverbs or adverbial phrases). In English, objects and complements nearly always come after the verb; a direct object precedes other complements such as prepositional phrases, but if there is an indirect object as well, expressed without a preposition, then that precedes the direct object: *give me the book*, but *give the book to me*. Adverbial modifiers generally follow objects, although other positions are possible (see under Adverbs below). Certain verb-modifier combinations, particularly when they have independent meaning (such as *take on* and *get up*), are known as "phrasal verbs".

For details of possible patterns, see English clause syntax. See the Non-finite clauses section of that article for verb phrases headed by non-finite verb forms, such as infinitives and participles.

Adjectives

English adjectives, as with other word classes, cannot in general be identified as such by their form,^[1] although many of them are formed from nouns or other words by the addition of a suffix, such as *-al* (*habitual*), *-ful* (*blissful*), *-ic* (*atomic*), *-ish* (*impish, youngish*), *-ous* (*hazardous*), etc.; or from other adjectives using a prefix: *disloyal, irredeemable, unforeseen, overtired*.

Adjectives may be used attributively, as part of a noun phrase (nearly always preceding the noun they modify), as in *the big house*, or predicatively, as in *the house is big*. Certain adjectives are restricted to one or other use; for example, *drunken* is attributive (*a drunken sailor*), while *drunk* is usually predicative (*the sailor was drunk*).

Comparison

Many adjectives have comparative and superlative forms in *-er* and *-est*,^[1] such as *faster* and *fastest* (from the positive form *fast*). Spelling rules which maintain pronunciation apply to suffixing adjectives just as they do for similar treatment of regular past tense formation; these cover consonant doubling (as in *bigger* and *biggest*, from *big*) and the change of *y* to *i* after consonants (as in *happier* and *happiest*, from *happy*).

The adjectives *good* and *bad* have the irregular forms *better, best* and *worse, worst*; also *far* becomes *farther, farthest* or *further, furthest*. The adjective *old* (for which the regular *older* and *oldest* are usual) also has the irregular forms *elder* and *eldest*, these generally being restricted to use in comparing siblings and in certain independent uses. For the comparison of adverbs, see Adverbs below.

Many adjectives, however, particularly those that are longer and less common, do not have inflected comparative and superlative forms. Instead, they can be qualified with *more* and *most*, as in *beautiful, more beautiful, most beautiful* (this construction is also sometimes used even for adjectives for which inflected forms do exist).

Certain adjectives are classed as ungradable.^[1] These represent properties that cannot be compared on a scale; they simply apply or do not, as with *pregnant, dead, unique*. Consequently, comparative and superlative forms of such adjectives are not normally used, except in a figurative, humorous or imprecise context. Similarly, such adjectives are not normally qualified with modifiers of degree such as *very* and *fairly*, although with some of them it is idiomatic to use adverbs such as *completely*. Another type of adjectives sometimes considered ungradable is those

that represent an extreme degree of some property, such as *delicious* and *terrified*; the same point about modifiers applies to these, although it is not rare to find them in comparative or superlative form.

Adjective phrases

An adjective phrase is a group of words that plays the role of an adjective in a sentence. It usually has a single adjective as its head, to which modifiers and complements may be added.^[1]

Adjectives can be modified by a preceding adverb or adverb phrase, as in *very warm*, *truly imposing*, *more than a little excited*. Some can also be preceded by a noun or quantitative phrase, as in *fat-free*, *two-metre-long*.

Complements following the adjective may include:

- prepositional phrases: *proud of him*, *angry at the screen*, *keen on breeding toads*;
- infinitive phrases: *anxious to solve the problem*, *easy to pick up*;
- content clauses, i.e. *that* clauses and certain others: *certain that he was right*, *unsure where they are*;
- after comparatives, phrases or clauses with *than*: *better than you*, *smaller than I had imagined*.

An adjective phrase may include both modifiers before the adjective and a complement after it, as in *very difficult to put away*.

Adjective phrases containing complements after the adjective cannot normally be used as attributive adjectives *before* a noun. Sometimes they are used attributively after the noun, as in *a woman proud of being a midwife* (where they may be converted into relative clauses: *a woman who is proud of being a midwife*), but it is wrong to say **a proud of being a midwife woman*. Exceptions include very brief and often established phrases such as *easy-to-use*. (Certain complements can be moved to after the noun, leaving the adjective before the noun, as in *a better man than you*, *a hard nut to crack*.)

Certain attributive adjective phrases are formed from other parts of speech, without any adjective as their head, as in *a two-bedroom house*, *a no-jeans policy*.

Adverbs

Adverbs perform a wide range of functions, typically modifying verbs (or verb phrases), adjectives (or adjective phrases), or other adverbs (or adverb phrases),^[1] although they also sometimes qualify noun phrases (*only the boss*, *quite a lovely place*), pronouns and determiners (*almost all*) and prepositional phrases (*halfway through the movie*), indicate an attitude or comment on a whole sentence (*frankly*, *I don't believe you*),^[1] or indicate the relation between clauses or sentences (*he died*, *and consequently I inherited the estate*).^[1]

Many English adverbs are formed from adjectives, by adding the ending *-ly*, as in *hopefully*, *widely*, *theoretically* (for details of spelling and etymology, see *-ly*). Certain words can be used as both adjectives and adverbs, such as *fast*, *straight*, and *hard*. The adverb corresponding to the adjective *good* is *well* (note that *bad* forms the regular *badly*, although *ill* is occasionally used in some phrases).

There are also a large number of adverbs that are not derived from adjectives,^[1] including adverbs of time (*today*, *soon*, *then*), *of place* (*here*, *there*, *everywhere*), of degree (*very*, *much*, *quite*, *so*, *too*), and with other meanings (*only*, *just*, *however*, *therefore*). Some suffixes that are fairly commonly used to form adverbs from nouns are *-ward[s]* (as in *homeward[s]*) and *-wise* (as in *lengthwise*).

A few adverbs retain irregular inflection for comparative and superlative forms:^[1] *much*, *more*, *most*; *a little*, *less*, *least*; *soon*, *sooner*, *soonest*; *well*, *better*, *best*; *badly*, *worse*, *worst*; *far*, *further* (*farther*), *furthest* (*farthest*); or follow the regular adjectival inflection: *fast*, *faster*, *fastest*, etc. However most adverbs form comparatives and superlatives by modification with *more* and *most*: *often*, *more often*, *most often*; *smoothly*, *more smoothly*, *most smoothly* (see also comparison of adjectives, above).

Adverbs indicating the manner of an action are most usually placed after the verb and its objects (*We considered the proposal carefully*), although other positions are often possible. Many adverbs of frequency, degree, certainty, etc.

(such as *often*, *always*, *almost*, *probably*, and various others such as *just*) tend to be placed before the verb (*they usually have chips*), although if there is an auxiliary or other "special verb" (see Verbs above), then the normal position for such adverbs is after the special verb (or after the first of them, if there is more than one): *I have just finished the crossword, she can usually manage a pint, we are never late, you might possibly have been unconscious.* Adverbs that provide a connection with previous information (such as *next*, *then*, *however*), and those that provide a context (such as time or place) for the sentence, often come at the start of the sentence: *Yesterday we went on a shopping expedition.*^[6]

A special type of adverb is the adverbial particle used to form phrasal verbs (such as *up* in *pick up*, *on* in *get on*, etc.) If such a verb also has an object, then the particle may precede or follow the object, although it will normally follow the object if the object is a pronoun (*pick the pen up* or *pick up the pen*, but *pick it up*).

Adverb phrases

An adverb phrase is a phrase that acts as an adverb within a sentence.^[1] An adverb phrase may have an adverb as its head, together with any modifiers (other adverbs or adverb phrases) and complements, analogously to the adjective phrases described above. For example: *very sleepily, all too suddenly, oddly enough, perhaps shockingly for us.*

Another very common type of adverb phrase is the prepositional phrase, which consists of a preposition and its object: *in the pool, after two years, for the sake of harmony.*

Prepositions

Prepositions form a closed word class,^[1] although there are also certain phrases that serve as prepositions, such as *in front of*. A single preposition may have a variety of meanings, often including temporal, spatial and abstract. Many words that are prepositions can also serve as adverbs. Examples of common English prepositions (including phrasal instances) are *of, in, on, over, under, to, from, with, in front of, behind, opposite, by, before, after, during, through, in spite of or despite, between, among, etc.*

A preposition is usually used with a noun phrase as its complement. A preposition together with its complement is called a prepositional phrase.^[1] Examples are *in England, under the table, after six pleasant weeks, between the land and the sea.* A prepositional phrase can be used as a complement or post-modifier of a noun in a noun phrase, as in *the man in the car, the start of the fight; as a complement of a verb or adjective, as in deal with the problem, proud of oneself; or generally as an adverb phrase (see above).*

English allows the use of "stranded" prepositions. This can occur in interrogative and relative clauses, where the interrogative or relative pronoun that is the preposition's complement is moved to the start (fronted), leaving the preposition in place. This kind of structure is avoided in some kinds of formal English. For example:

- *What are you talking about?* (Possible alternative version: *About what are you talking?*)
- *The song that you were listening to ...* (more formal: *The song to which you were listening ...*)

Notice that in the second example the relative pronoun *that* could be omitted.

Stranded prepositions can also arise in passive voice constructions and other uses of passive past participle phrases, where the complement in a prepositional phrase can become zero in the same way that a verb's direct object would: *it was looked at; I will be operated on; get your teeth seen to.* The same can happen in certain uses of infinitive phrases: *he is nice to talk to; this is the page to make copies of.*

Conjunctions

Conjunctions express a variety of logical relations between items, phrases, clauses and sentences.^[7] The principal coordinating conjunctions in English are *and*, *or*, and *but*, as well as *nor*, *so*, *yet* and *for*. These can be used in many grammatical contexts to link two or more items of equal grammatical status,^[1] for example:

- Noun phrases combined into a longer noun phrase, such as *John, Eric, and Jill, the red coat or the blue one*. When *and* is used, the resulting noun phrase is plural. A determiner does not need to be repeated with the individual elements: *the cat, the dog, and the mouse* and *the cat, dog, and mouse* are both correct. The same applies to other modifiers. (The word *but* can be used here in the sense of "except": *nobody but you*.)
- Adjective or adverb phrases combined into a longer adjective or adverb phrase: *tired but happy, over the fields and far away*.
- Verbs or verb phrases combined as in *he washed, peeled, and diced the turnips* (verbs conjoined, object shared); *he washed the turnips, peeled them, and diced them* (full verb phrases, including objects, conjoined).
- Other equivalent items linked, such as prefixes linked in *pre- and post-test counselling*,^[8] numerals as in *two or three buildings*, etc.
- Clauses or sentences linked, as in *We came but they wouldn't let us in. They wouldn't let us in, nor would they explain what we had done wrong*.

There are also correlative conjunctions, where as well as the basic conjunction, an additional element appears before the first of the items being linked.^[1] The common correlatives in English are:

- *either ... or* (*either a man or a woman*);
- *neither ... nor* (*neither clever nor funny*);
- *both ... and* (*they both punished and rewarded them*);
- *not ... but*, particularly in *not only ... but also* (*not exhausted but exhilarated, not only football but also many other sports*).

Subordinating conjunctions make relations between clauses, making the clause in which they appear into a subordinate clause.^[1] Some common subordinating conjunctions in English are:

- conjunctions of time, including *after, before, since, until, when, while*;
- conjunctions of cause and effect, including *because, since, now that, as, in order that, so*;
- conjunctions of opposition or concession, such as *although, though, even though, whereas, while*;
- conjunctions of condition: such as *if, unless, only if, whether or not, even if, in case (that)*;
- the conjunction *that*, which produces content clauses, as well as words that produce interrogative content clauses: *whether, where, when, how*, etc.

A subordinating conjunction generally comes at the very start of its clause, although many of them can be preceded by qualifying adverbs, as in *probably because ..., especially if* The conjunction *that* can be omitted after certain verbs, as in *she told us (that) she was ready*. (For the use of *that* in relative clauses, see Relative pronouns above.)

Negation

As noted above under Verbs, a finite indicative verb (or its clause) is negated by placing the word *not* after an auxiliary, modal or other "special" verb such as *do, can* or *be*. For example, the clause *I go* is negated with the appearance of the auxiliary *do*, as *I do not go* (see *do-support*). When the affirmative already uses auxiliary verbs (*I am going*), no other auxiliary verbs are added to negate the clause (*I am not going*). (Until the period of early Modern English, negation was effected without additional auxiliary verbs: *I go not*.)

Most combinations of auxiliary verbs etc. with *not* have contracted forms: *don't, can't, isn't*, etc. (Also the uncontracted negated form of *can* is written as a single word *cannot*.) On inversion of subject and verb (such as in questions; see below), the subject may be placed after a contracted negated form: *Should he not pay?* or *Shouldn't he pay?*

Other elements, such as noun phrases, adjectives, adverbs, infinitive and participial phrases, etc., can be negated by placing the word *not* before them: *not the right answer*, *not interesting*, *not to enter*, *not noticing the train*, etc.

When other negating words such as *never*, *nobody*, etc. appear in a sentence, the negating *not* is omitted (unlike its equivalents in many languages): *I saw nothing* or *I didn't see anything*, but not (except in non-standard speech) **I didn't see nothing*. Sometimes, multiple negations are used for humorous purposes, and are understood to follow the rules of multiplying negative numbers in math: an even number of negations, including zero, means a positive, while an odd number means a negative. For example, *There is not no cow over there* is understood to mean there is at least one cow, which is positive. *He didn't never not bow after a performance* is understood to mean there has been at least one instance of him not bowing, which is negative.

Clause and sentence structure

A typical sentence contains one independent clause and possibly one or more dependent clauses, although it is also possible to link together sentences of this form into longer sentences, using coordinating conjunctions (see above).

A clause typically contains a subject (a noun phrase) and a predicate (a verb phrase in the terminology used above; that is, a verb together with its objects and complements). A dependent clause also normally contains a subordinating conjunction (or in the case of relative clauses, a relative pronoun or phrase containing one). English syntax is essentially of SVO (subject–verb–object) type; the verb precedes its object in the verb phrase, and the subject of the clause precedes the verb.

Questions

Like many other Western European languages, English historically allowed questions to be asked by inverting the positions of verb and subject. Modern English requires the use of the auxiliary verb *do*, along with inversion of the word order, to form a question from a simple (one-word) affirmative (*I go* → *Do I go?*, *Where do I go?*), apart from when the main verb is "be" (*I am here* → *Am I here?*, *Why am I here?*). When the affirmative verb is compound, a question is formed by inverting the auxiliary verb with the subject (*John is going* → *Is John going?*).

Combining the formation of a question with negation involves both insertion of *do*, if the verb is not already compound, and inversion of the auxiliary verb with the subject: *John is going* → *Is John not going?*; *John goes* → *Does John not go?*. However, the word *not* can optionally (especially in informal English) be contracted with the auxiliary verb, in which case the word order is further changed: *Isn't John going?*, *Doesn't John go?*.

Dependent clauses

The syntax of a dependent clause is generally the same as that of an independent clause, except that the dependent clause usually begins with a subordinating conjunction or relative pronoun (or phrase containing such). In some situations (as already described) the conjunction or relative pronoun *that* can be omitted. Another type of dependent clause with no subordinating conjunction is the conditional clause formed by inversion (see below).

Other uses of inversion

The clause structure with inverted subject and verb, used to form questions as described above, is also used in certain types of declarative sentence. This occurs mainly when the sentence begins with an adverbial or other phrase that is essentially negative or contains words such as *only*, *hardly*, etc.: *Never have I known someone so stupid*; *Only in France can such food be tasted*.

In elliptical sentences (see below), inversion takes place after *so* (meaning "also") as well as after the negative *neither*: *so do I*, *neither does she*.

Inversion can also be used to form conditional clauses, beginning with *should*, *were* (subjunctive), or *had*, in the following ways:

- *should I win the race* (equivalent to *if I win the race*);
- *were he a soldier* (equivalent to *if he were a soldier*);
- *were he to win the race* (equivalent to *if he were to win the race*, i.e. *if he won the race*);
- *had he won the race* (equivalent to *if he had won the race*).

Other similar forms sometimes appear, but are less common. There is also a construction with subjunctive *be*, as in *be he alive or dead* (meaning "no matter whether he is alive or dead").

Use of inversion to express a third-person imperative is now mostly confined to the expression *long live X*, meaning "let X live long".

Imperatives

In an imperative sentence (one giving an order), there is usually no subject in the independent clause: *Go away until I call you*. It is possible, however, to include *you* as the subject for emphasis: *You stay away from me*.

Elliptical constructions

Many types of elliptical construction are possible in English, resulting in sentences that omit certain redundant elements. Various examples are given in the article on Ellipsis.

Some notable elliptical forms found in English include:

- Short statements of the form *I can, he isn't, we mustn't*. Here the verb phrase (understood from the context) is reduced to a single auxiliary or other "special" verb, negated if appropriate. If there is no special verb in the original verb phrase, it is replaced by *do/does/did: he does, they didn't*.
- Clauses that omit the verb, in particular those like *me too, nor me, me neither*. The latter forms are used after negative statements. (Equivalents including the verb: *I do too or so do I; I don't either or neither do I*.)
- Tag questions, formed with a special verb and pronoun subject: *isn't it?; were there?; am I not?*

History of English grammars

The first published English grammar was a *Pamphlet for Grammar* of 1586, written by William Bullokar with the stated goal of demonstrating that English was just as rule-based as Latin. Bullokar's grammar was faithfully modeled on William Lily's Latin grammar, *Rudimenta Grammatices* (1534), used in English schools at that time, having been "prescribed" for them in 1542 by Henry VIII. Bullokar wrote his grammar in English and used a "reformed spelling system" of his own invention; but many English grammars, for much of the century after Bullokar's effort, were written in Latin, especially by authors who were aiming to be scholarly. John Wallis's *Grammatica Linguae Anglicanae* (1685) was the last English grammar written in Latin.

Even as late as the early 19th century, Lindley Murray, the author of one of the most widely used grammars of the day, was having to cite "grammatical authorities" to bolster the claim that grammatical cases in English are different from those in Ancient Greek or Latin.

Notes and references

- [1] Carter & McCarthy 2006, p. 296
- [2] Some linguists consider *that* in such sentences to be a complementizer rather than a relative pronoun. See English relative clauses: Status of *that*.
- [3] For a treatment of *there* as a dummy predicate, based on the analysis of the copula, see Moro, A., *The Raising of Predicates. Predicative Noun Phrases and the Theory of Clause Structure*, Cambridge Studies in Linguistics, 80, Cambridge University Press, 1997.
- [4] C.D. Sidhu, *An Intensive Course in English*, Orient Blackswan, 1976, p. 5.
- [5] Dependency grammars reject the concept of finite verb phrases as clause constituents, regarding the subject as a dependent of the verb as well. See the verb phrase article for more information.
- [6] esl.about.com (http://esl.about.com/od/grammarstructures/a/adverb_placement.htm)
- [7] Carter & McCarthy 2006, p. 315
- [8] British Medical Association, *Misuse of Drugs*, Chapter 4, "Constraints of current practice."

Bibliography

Grammar books

- Aarts, Bas (2011). *Oxford modern English grammar*. Oxford University Press. p. 410. ISBN 978-0-19-953319-0.
- Biber, Douglas; Johansson, Stig; Leech, Geoffrey; Conrad, Susan; Finegan, Edward (1999). *Longman grammar of spoken and written English*. Pearson Education Limited. p. 1203. ISBN 0-582-23725-4.
- Biber, Douglas; Leech, Geoffrey; Conrad, Susan; (2002). *Longman student grammar of spoken and written English*. Pearson Education Limited. p. 487. ISBN 0-582-23726-2.
- Bryant, Margaret (1945). *A functional English grammar*. D.C. Heath and company. p. 326.
- Bryant, Margaret; Momozawa, Chikara (1976). *Modern English Syntax*. Seibido. p. 157.
- Carter, Ronald; McCarthy, Michael (2006). *Cambridge Grammar of English: A Comprehensive Guide*. Cambridge University Press. p. 984. ISBN 0-521-67439-5. A CD-Rom version is included.
- Celce-Murcia, Marianne; Larsen-Freeman, Diane (1999). *The Grammar Book: An ESL/EFL teacher's course*, 2nd ed.. Heinle & Heinle. p. 854. ISBN 0-8384-4725-2.
- Chalker, Sylvia; Weiner, Edmund, ed. *The Oxford Dictionary of English Grammar*. Oxford University Press. p. 464. ISBN 0-19-280087-6.
- Cobbett, William (1883). *A Grammar of the English Language, In a Series of Letters: Intended for the Use of Schools and of Young Persons in General, but more especially for the use of Soldiers, Sailors, Apprentices, and Plough-Boys* (<http://books.google.com/books?id=LIgAAAAAYAAJ&printsec=titlepage&cad=0#PPR1,M1>). New York and Chicago: A. S. Barnes and Company.
- Cobbett, William (2003, originally 1818). *A Grammar of the English Language (Oxford Language Classics)*. Oxford University Press. p. 256. ISBN 0-19-860508-0.
- Curme, George O., College English Grammar, Richmond, VA, 1925, Johnson Publishing company, 414 pages . A revised edition *Principles and Practice of English Grammar* was published by Barnes & Noble, in 1947.
- Curme, George O. (1978; original 1931, 1935). *A Grammar of the English Language: Volumes I (Parts of Speech) & II (Syntax)*. Verbatim Books. p. 1045. ISBN 0-930454-03-0.
- Declerck, Renaat (1990). *A Comprehensive Descriptive Grammar of English*. Kaitakusha,Tokyo. p. 595. ISBN 4-7589-0538-X. Declerck in his introduction (p.vi) states that almost half his grammar is taken up by the topics of tense, aspect and modality. This he contrasts with the 71 pages devoted to these subjects in *The Comprehensive Grammar of English*. Huddleston and Pullman say they profited from consulting this grammar in their *Cambridge Grammar of the English Language*. (p. 1765)
- Dekeyser, Xavier; Devriendt, Betty; Tops, Guy A. J.; Guekens, Steven; (2004). *Foundations of English Grammar For University Students and Advanced Learners*. Uitgeverij Acco, Leuven, Belgium. p. 449. ISBN 978-90-334-5637-4.
- Greenbaum, Sidney (1996). *Oxford English Grammar*. Oxford and New York: Oxford University Press. p. 672. ISBN 0-19-861250-8.

- Greenbaum, Sidney (1990). *A Student's Grammar of the English Language*. Addison Wesley Publishing Company. p. 496. ISBN 0-582-05971-2.
- Halliday, M. A. K.; Matthiessen, Christian M. I. M. (revised by) (2004). *An Introduction to Functional Grammar, 3rd. edition*. London: Hodder Arnold. p. 700. ISBN 0-340-76167-9.
- Huddleston, Rodney D. (1984) *Introduction to the grammar of English*. Cambridge: Cambridge University Press.
- Huddleston, Rodney D. (1988) *English grammar: An outline*. Cambridge: Cambridge University Press.
- Huddleston, Rodney D.; Pullum, Geoffrey K., eds. (2002). *The Cambridge grammar of the English language*. Cambridge University Press. p. 1860. ISBN 0-521-43146-8.
- Huddleston, Rodney D.; Pullum, Geoffrey K. (2005). *A student's introduction to English grammar*. Cambridge University Press. p. 320. ISBN 0-521-61288-8.
- Jespersen, Otto. (1937). *Analytic Syntax*. Copenhagen: Levin & Munksgaard, 1937. 170 p.
- Jespersen, Otto. (1909–1949). *A modern English grammar on historical principles* (Vols. 1-7). Heidelberg: C. Winter.
- Jespersen, Otto (1933). *Essentials of English Grammar: 25th impression, 1987*. London: Routledge. p. 400. ISBN 0-415-10440-8.
- Jonson, Ben (1756). "The English grammar: Made by Ben Jonson for the benefit of all strangers, out of his observation of the English language now spoken and in use" (http://books.google.com/books?id=SaM_AAAAYAAJ&printsec=titlepage&cad=0#PPA205,M1). *The Works of Ben Jonson: Volume 7*. London: D. Midwinter et al.
- Kolln, Martha J. (2006). *Rhetorical Grammar: Grammatical Choices, Rhetorical Effects, 5th edition*. Longman. p. 336. ISBN 0-321-39723-1.
- Kolln, Martha J.; Funk, Robert W. (2008). *Understanding English Grammar (8th Edition)*. Longman. p. 453. ISBN 0-205-62690-4.
- Korsakov, A. K. (Andreĭ Konstantinovich). 1969. The use of tenses in English. Korsakov, A. K. Structure of Modern English pt. 1. oai:gial.edu:26766 at <http://www.language-archives.org/item/oai:gial.edu:26766>
- Maetzner, Eduard Adolf Ferdinand, 1805–1892. (1873). *An English grammar; methodical, analytical, and historical*. J. Murray, London. Three Volumes, translated by Clair James Grece from the German edition *Englische Grammatik: Die Lehre von der Wort- und Satzfügung*. Professor Whitney in his *Essentials of English Grammar* recommends the German original stating "there is an English version, but it is hardly to be used." (p. vi)
- Meyer-Myklestad, J., (1967). *An Advanced English Grammar for Students and Teachers*. Universitetsforlaget-Oslo. p. 627.
- Morenberg, Max (2002). *Doing Grammar, 3rd edition*. New York: Oxford University Press. p. 352. ISBN 0-19-513840-6.
- Poutsma, Hendrik. A grammar of late modern English, Groningen, P. Noordhoff, 1914–29, 2 pt. in 5 v. Contents: pt. I. The sentence: 1st half. The elements of the sentence, 1928. 2d half. The composite sentence, 1929.--pt. II. The parts of speech: section I, A. Nouns, adjectives and articles, 1914. section I, B. Pronouns and numerals, 1916. section II. The verb and the particles, 1926.
- Quirk, Randolph; Greenbaum, Sidney; Leech, Geoffrey; & Svartvik, Jan. (1972). *A grammar of contemporary English*. Harlow: Longman.
- Quirk, Randolph (1985). *A comprehensive grammar of the English language*. Harlow: Longman. p. 1779. ISBN 0-582-51734-6.
- Schibsbye, Knud (1970). *A Modern English Grammar: Second Edition*. London: Oxford University Press. p. 390. ISBN 0-19-431327-1. This book is a translation of Schibsbye's three volume *Engelsk Grammatik* published between 1957 and 1961. Schibsbye was a student of Jespersen's and co-author of the sixth volume –Morphology –of Jespersen's seven volume *Modern English Grammar*.
- Sinclair, John, ed. (1991) *Collins COBUILD – English Grammar* London: Collins ISBN 0-00-370257-X second edition, 2005 ISBN 0-00-718387-9. Huddleston and Pullman say they found this grammar 'useful' in their

Cambridge Grammar of the English Language. (p. 1765) A CD-Rom version of the 1st edition is available on the Collins COBUILD Resource Pack ISBN 0-00-716921-3

- Sledd, James. (1959) *A short introduction to English grammar* Chicago: Scott, Foresman.
- Strang, Barbara M. H. (1968) *Modern English structure* (2nd ed.) London: Arnold.
- Thomson, A. J. (Audrey Jean); Martinet, A. V. (Agnes V.) (1986). *A practical English grammar:Fourth Edition.* Oxford University Press. p. 384. ISBN 0-19-431342-5.
- Visser, F. Th. (Fredericus Theodorus) (2003). *An historical syntax of the English language*. Brill. ISBN 90-04-07142-3 (set). 4th impression. pts. 1-2. Syntactical units with one verb.--pt.3. 1st half. Syntactical units with two verbs.--pt.3. 2d half. Syntactical units with two and more verbs.
- Whitney, William Dwight, (1877) *Essentials of English Grammar*, Boston: Ginn & Heath.
- Zandvoort, R. W. (1972) *A handbook of English grammar* (2nd ed.) London: Longmans.

Monographs

- Adams, Valerie. (1973). *An introduction to modern English word-formation*. London: Longman.
- Bauer, Laurie. (1983). *English word-formation*. Cambridge: Cambridge University Press.
- Fries, Charles Carpenter. (1952). *The structure of English; an introduction to the construction of English sentences*. New York: Harcourt, Brace.
- Halliday, M. A. K. (1985/94). *Spoken and written language*. Deakin University Press.
- Huddleston, Rodney D. (1976). *An introduction to English transformational syntax*. Longman.
- Huddleston, Rodney D. (2009). *The Sentence in Written English: A Syntactic Study Based on an Analysis of Scientific Texts*. Cambridge University Press.. p. 352. ISBN 0-521-11395-4.
- Jespersen, Otto (1982). *Growth and Structure of the English Language*. Chicago and London: University of Chicago Press. p. 244. ISBN 0-226-39877-3.
- Jespersen, Otto (1992). *Philosophy of Grammar*. Chicago and London: University of Chicago Press. p. 363. ISBN 0-226-39881-1.
- Jespersen, Otto (1962). *Selected Writings*. London: Allen & Unwin. p. 820.--includes Jespersen's monographs *Negation in English and Other Languages*, and *A System of Grammar*.
- Kruisinga, E. (1925). *A handbook of present-day English*. Utrecht: Kemink en Zoon.
- Leech, Geoffrey N. (1971). *Meaning and the English verb*. London: Longman.
- Marchand, Hans. (1969). *The categories and types of present-day English word-formation* (2nd ed.). München: C. H. Beck.
- McCawley, James D. (1998). *The syntactic phenomena of English* (2nd ed.). Chicago: The University of Chicago Press.
- Onions, C. T. (Charles Talbut), (1904, 1st edition) *An advanced English syntax based on the principles and requirements of the Grammatical society*. London: Keegan Paul, Trench, Trubner & co. A new edition of *An advanced English syntax*, prepared from the author's materials by B. D. H. Miller, was published as *Modern English syntax* in 1971.
- Palmer, F. R. (1974). *The English verb*. London: Longman.
- Palmer, F. R. (1979). *Modality and the English modals*. London: Longman.
- Plag, Ingo. (2003). *Word-formation in English*. Cambridge: Cambridge University Press.
- Scheurweghs, Gustave. (1959). *Present-day English syntax: A survey of sentence patterns*. London: Longmans.

External links

- English Grammar, wikibook in English
- GrammarBank – English Grammar Resources for Teachers and Learners (<http://www.grammarbank.com>)
- A Friendly Grammar of English (<http://www.beaugrande.com/UPLOADGRAMMARHEADER.htm>) by Robert de Beaugrande
- Modern English Grammar (<http://papyr.com/hypertextbooks/grammar/>) by Daniel Kies
- Grammar lessons, rules, and news for everyday use (<http://www.englishgrammar.org/>) by Jennifer Frost
- The Internet Grammar of English (<http://www.ucl.ac.uk/internet-grammar/home.htm>).
- A Short Overview of English Syntax (<http://www.lel.ed.ac.uk/grammar/overview.html>) by Rodney Huddleston (based on *The Cambridge Grammar of the English Language*)
- (<http://grammar.ccc.commnet.edu/grammar/>) Basic Grammar Rules

Clause

In grammar, a **clause** is the smallest grammatical unit that can express a complete proposition.^[1] A typical clause in English contains minimally a subject and a predicate.^[2] In other languages, the subject is often omitted if it is retrievable from context. A simple sentence usually consists of a single finite clause with a finite verb that is independent. More complex sentences may contain multiple clauses. Main clauses (= *matrix clauses, independent clauses*) are those that could stand as a sentence by themselves. Subordinate clauses (= *embedded clauses, dependent clauses*) are those that would be awkward or nonsensical if used alone.

Two major distinctions

A primary division for the discussion of clauses is the distinction between main clauses (= *matrix clauses, independent clauses*) and subordinate clauses (= *embedded clauses, dependent clauses*).^[3] A main clause can stand alone, i.e. it can constitute a complete sentence by itself. A subordinate clause (= *embedded clause*), in contrast, is reliant on the appearance of a main clause; it depends on the main clause and is therefore a dependent clause, whereas the main clause is an independent clause.

A second major distinction concerns the difference between finite and non-finite clauses. A finite clause contains a structurally central finite verb, whereas the structurally central word of a non-finite clause is often a non-finite verb. Traditional grammar focuses on finite clauses, the awareness of non-finite clauses having arisen much later in connection with the modern study of syntax. The discussion here also focuses on finite clauses, although some aspects of non-finite clauses are considered further below.

Clauses according to a distinctive syntactic trait

Clauses can be classified according to a distinctive trait that is a prominent characteristic of their syntactic form. The position of the finite verb is one major trait used for classification, and the appearance of a specific type of focusing word (e.g. *wh-word*) is another. These two criteria overlap to an extent, which means that often no single aspect of syntactic form is always decisive in determining how the clause functions. There are, however, strong tendencies.

Standard SV-clauses

Standard SV-clauses (subject-verb) are the norm in English. They are usually declarative (as opposed to exclamative, imperative, or interrogative); they express information in a neutral manner, e.g.

The pig has not yet been fed. - Declarative clause, standard SV order

I've been hungry for two hours. - Declarative clause, standard SV order

...that I've been hungry for two hours. - Declarative clause, standard SV order, but functioning as a subordinate clause due to the appearance of the subordinator *that*

Declarative clauses like these are by far the most frequently occurring type of clause in any language. They can be viewed as basic, other clause types being derived from them. Standard SV-clauses can also be interrogative or exclamative, however, given the appropriate intonation contour and/or the appearance of a question word, e.g.

- a. The pig has not yet been fed? - Rising intonation on *fed* makes the clause a yes/no-question.
- b. The pig has not yet been fed! - Spoken forcefully, this clause is exclamative.
- c. You've been hungry for how long? - Appearance of interrogative word *how* and rising intonation make the clause a constituent question

Examples like these demonstrate that how a clause functions cannot be known based entirely on a single distinctive syntactic criterion. SV-clauses are usually declarative, but intonation and/or the appearance of a question word can render them interrogative or exclamative.

Verb first clauses

Verb first clauses in English usually play one of three roles: 1. They express a yes/no-question via subject-auxiliary inversion, 2. they express a condition as an embedded clause, or they express a command via imperative mood, e.g.

- a. He **should** stop laughing. - Standard declarative SV-clause (verb second order)
- b. **Should** he stop laughing? - Yes/no-question expressed by verb first order
- c. **Had** he stopped laughing,... - Condition expressed by verb first order
- d. **Stop** laughing! - Imperative formed with verb first order

- a. They **have** done the job. - Standard declarative SV-clause (verb second order)
- b. **Have** they done the job? - Yes/no-question expressed by verb first order
- c. **Had** they done the job,... - Condition expressed by verb first order
- d. **Do** the job! - Imperative formed with verb first order

Most verb first clauses are main clauses. Verb first conditional clauses, however, must be classified as embedded clauses because they cannot stand alone.

Wh-clauses

Wh-clauses contain a *wh*-word. *Wh*-words often serve to help express a constituent question. They are also prevalent, though, as relative pronouns, in which case they serve to introduce a relative clause and are not part of a question. The *wh*-word focuses a particular constituent and most of the time, it appears in clause-initial position. The following examples illustrate standard interrogative *wh*-clauses. The b-sentences are direct questions (main clauses), and the c-sentences contain the corresponding indirect questions (embedded clauses):

- a. Sam likes the meat. - Standard declarative SV-clause
- b. **Who** likes the meat? - Matrix interrogative *wh*-clause focusing on the subject
- c. They asked **who likes the meat**. - Embedded interrogative *wh*-clause focusing on the subject

- a. Larry sent Susan to the store. - Standard declarative SV-clause
- b. **Who** did Larry send to the store? - Matrix interrogative *wh*-clause focusing on the object, subject-auxiliary inversion present
- c. We know **who Larry sent to the store**. - Embedded *wh*-clause focusing on the object, subject-auxiliary inversion absent

- a. Larry sent Susan to the store. - Standard declarative SV-clause

- b. **Where** did Larry send Susan? - Matrix interrogative *wh*-clause focusing on the oblique object, subject-auxiliary inversion present
- c. Someone is wondering where Larry sent Susan. - Embedded *wh*-clause focusing on the oblique object, subject-auxiliary inversion absent

One important aspect of matrix *wh*-clauses is that subject-auxiliary inversion is obligatory when something other than the subject is focused. When it is the subject (or something embedded in the subject) that is focused, however, subject-auxiliary inversion does not occur.

- a. **Who** called you? - Subject focused, no subject-auxiliary inversion
- b. **Who** did you call? - Object focused, subject-auxiliary inversion occurs

Another important aspect of *wh*-clauses concerns the absence of subject-auxiliary inversion in embedded clauses, as illustrated in the c-examples just produced. Subject-auxiliary inversion is obligatory in matrix clauses when something other than the subject is focused, but it never occurs in embedded clauses regardless of the constituent that is focused. A systematic distinction in word order emerges across matrix *wh*-clauses, which can have VS order, and embedded *wh*-clauses, which always maintain SV order, e.g.

- a. **Why** are they doing that? - Subject-auxiliary inversion results in VS order in matrix *wh*-clause.
- b. They told us why they are doing that. - Subject-auxiliary inversion is absent in embedded *wh*-clause.
- c. *They told us why are they doing that. - Subject-auxiliary inversion is blocked in embedded *wh*-clause.
- a. **Who** is he trying to avoid? - Subject-auxiliary inversion results in VS order in matrix *wh*-clause.
- b. We know who he is trying to avoid? - Subject-auxiliary inversion is absent in embedded *wh*-clause.
- c. *We know who is he trying to avoid. - Subject-auxiliary inversion is blocked in embedded *wh*-clause.

Relative clauses

Relative clauses are a mixed group. In English they can be standard SV-clauses if they are introduced by *that* or lack a relative pronoun entirely, or they can be *wh*-clauses if they are introduced by a *wh*-word that serves as a relative pronoun.

- a. Something happened twice. - Standard declarative SV-clause
- b. something that happened twice - Relative clause introduced by the relative pronoun *that* and modifying the indefinite pronoun *something*
- a. I know everyone. - Standard declarative SV-clause
- b. everyone I know - Relative clause lacking a relative pronoun entirely and modifying the indefinite pronoun *everyone*
- a. They left early - Standard declarative clause
- b. the time when they left early - Relative clause introduced by the relative proform *when* and modifying the noun *time*
- a. The woman sang a song. - Standard declarative SV-clause
- b. the woman who sang a song. - Relative clause introduced by the relative pronoun *who* and modifying the noun *woman*

Being embedded clauses, relative clauses in English cannot display subject-auxiliary inversion.

A particular type of *wh*-relative-clause is the so-called *free relative clause*. Free relatives typically function as arguments, e.g.

- a. What he did was unexpected. - Free relative clause functioning as subject argument
- b. He will flatter whoever is present. - Free relative clause functioning as object argument

These relative clauses are "free" because they can appear in a variety of syntactic positions; they are not limited to appearing as modifiers of nominals. The suffix *-ever* is often employed to render a standard relative pronoun as a pronoun that can introduce a free relative clause.

Clauses according to semantic predicate-argument function

Embedded clauses can be categorized according to their syntactic function in terms of predicate-argument structures. They can function as arguments, as adjuncts, or as predicative expressions. That is, embedded clauses can be an argument of a predicate, an adjunct on a predicate, or (part of) the predicate itself. The predicate in question is usually the matrix predicate of a main clause, but embedding of predicates is also frequent.

Argument clauses

A clause that functions as the argument of a given predicate is known as an *argument clause*. Argument clauses can appear as subjects, as objects, and as obliques. They can also modify a noun predicate, in which case they are known as *content clauses*.

That they actually helped was really appreciated. - SV-clause functioning as the subject argument

They mentioned that they had actually helped. - SV-clause functioning as the object argument

What he said was ridiculous. - Wh-clause functioning as the subject argument

We know what he said. - Wh-clause functioning as an object argument

He talked about what he had said. - Wh-clause functioning as an oblique object argument

The following examples illustrate argument clauses that provide the content of a noun. Such argument clauses are content clauses:

- a. the claim that he was going to change it - Argument clause that provides the content of a noun (= content clause)
- b. the claim that he expressed - Adjunct clause (relative clause) that modifies a noun
- a. the idea that we should alter the law - Argument clause that provides the content of a noun (= content clause)
- b. the idea that came up - Adjunct clause (relative clause) that modifies a noun

The content clauses like these in the a-sentences are arguments. Relative clauses introduced by the relative pronoun *that* as in the b-clauses here have an outward appearance that is closely similar to that of content clauses. The relative clauses are adjuncts, however, not arguments.

Adjunct clauses

Adjunct clauses are embedded clauses that modify an entire predicate-argument structure. All clause types (SV-, verb first, wh-) can function as adjuncts, although the stereotypical adjunct clause is SV and introduced by a subordinator (= subordinate conjunction, e.g. *after*, *because*, *before*, *when*, etc.), e.g.

- a. Fred arrived before you did. - Adjunct clause modifying matrix clause
- b. After Fred arrived, the party started. - Adjunct clause modifying matrix clause
- c. Susan skipped the meal because she is fasting. - Adjunct clause modifying matrix clause

These adjunct clauses modify the entire matrix clause. Thus *before you did* in the first example modifies the matrix clause *Fred arrived*. Adjunct clauses can also modify a nominal predicate. The typical instance of this type of adjunct is a relative clause, e.g.

- a. We like the music that you brought. - Relative clause functioning as an adjunct that modifies the noun *music*
- b. The people who brought music were singing loudly. - Relative clause functioning as an adjunct that modifies the noun *people*
- c. They are waiting for some food that will not come. - Relative clause functioning as an adjunct that modifies the noun *people*

Predicative clauses

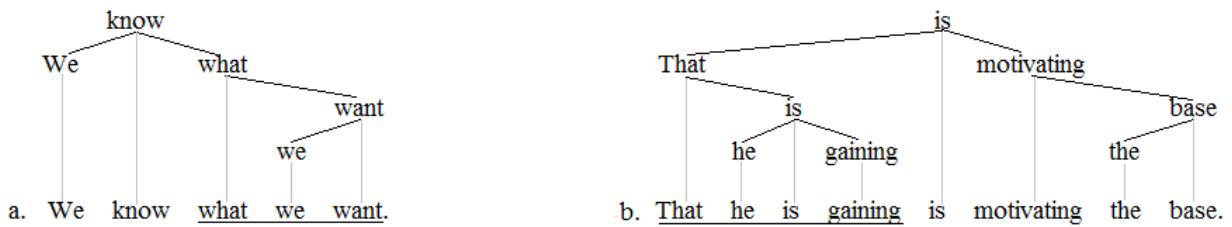
An embedded clause can also function as a predicative expression. That is, it can form (part of) the predicate of a greater clause.

- a. That was when they laughed. - Predicative SV-clause, i.e. a clause that functions as (part of) the main predicate
- b. He became what he always wanted to be. - Predicative wh-clause, i.e. wh-clause that functions as (part of) the main predicate

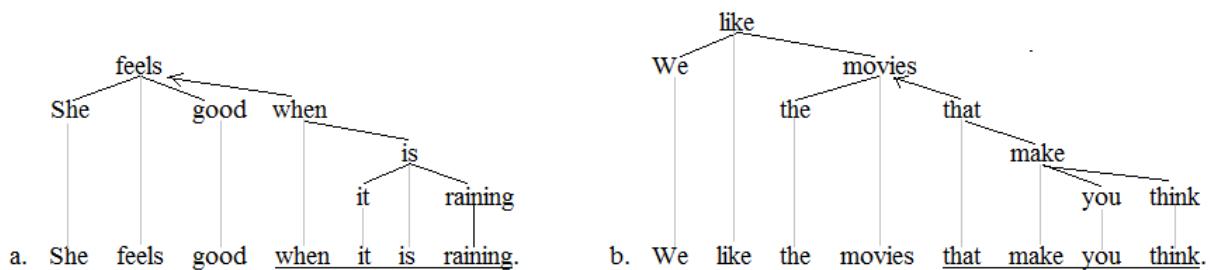
These predicative clauses are functioning just like other predicative expressions, e.g. predicative adjectives (*That was good*) and predicative nominals (*That was the truth*). They form the matrix predicate together with the copula.

Representing clauses

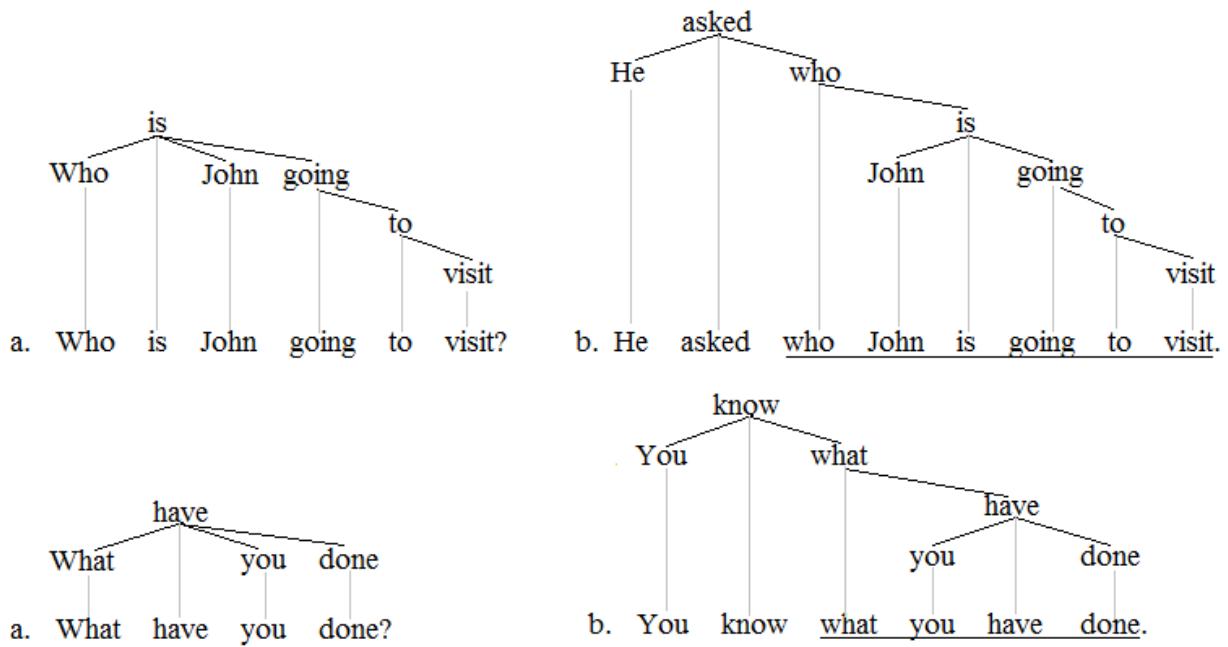
Some of the distinctions presented above are represented in syntax trees. These trees make the difference between main and subordinate clauses very clear, and they also illustrate well the difference between argument and adjunct clauses. The following dependency grammar trees show that embedded clauses are dependent on an element in the main clause, often on a verb:^[4]



The main clause encompasses the entire tree each time, whereas the embedded clause is contained within the main clause. These two embedded clauses are arguments. The embedded wh-clause *what he wanted* is the object argument of the predicate *know*. The embedded clause *that he is gaining* is the subject argument of the predicate *is motivating*. Both of these argument clauses are directly dependent on the main verb of the matrix clause. The following trees identify adjunct clauses using an arrow dependency edge:



These two embedded clauses are adjunct clauses because they provide circumstantial information that modifies a superordinate expression. The first is a dependent of the main verb of the matrix clause and the second is a dependent of the object noun. The arrow dependency edges identify them as adjuncts. The arrow points away from the adjunct towards its governor to indicate that semantic selection is running counter to the direction of the syntactic dependency; the adjunct is selecting its governor. The next four trees illustrate the distinction mentioned above between matrix wh-clauses and embedded wh-clauses



The embedded *wh*-clause is an object argument each time. The position of the *wh*-word across the matrix clauses (a-trees) and the embedded clauses (b-trees) captures the difference in word order. Matrix *wh*-clauses have V2 word order, whereas embedded *wh*-clauses have (what amounts to) V3 word order. In the matrix clauses, the *wh*-word is a dependent of the finite verb, whereas it is the head over the finite verb in the embedded *wh*-clauses.

Clauses vs. phrases

There has been confusion about the distinction between clauses and phrases. This confusion is due in part to how these concepts are employed in the phrase structure grammars of the chomskyan tradition. In the 1970s, chomskyan grammars began labeling many clauses as CPs (= complementizer phrases) or as IPs (= inflection phrases), and then later as TPs (= tense phrases), etc. The choice of labels was influenced by the theory-internal desire to use the labels consistently. The X-bar schema acknowledged at least three projection levels for every lexical head: a minimal projection (e.g. N, V, P, etc.), an intermediate projection (e.g. N', V', P', etc.), and a phrase level projection (e.g. NP, VP, PP, etc.). Extending this convention to the clausal categories occurred in the interest of the consistent use of labels.

This use of labels should not, however, be confused with the actual status of the syntactic units to which the labels are attached. A more traditional understanding of clauses and phrases maintains that phrases are not clauses, and clauses are not phrases. There is a progression in the size and status of syntactic units: *words < phrases < clauses*. The characteristic trait of clauses, i.e. the presence of a subject and a (finite) verb, is absent from phrases. Clauses can be, however, embedded inside phrases.

Non-finite clauses

The central word of a non-finite clause is usually a non-finite verb (as opposed to a finite verb). There are various types of non-finite clauses that can be acknowledged based in part on the type of non-finite verb at hand. Gerunds are widely acknowledged to constitute non-finite clauses, and some modern grammars also judge many *to*-infinitives to be the structural locus of non-finite clauses. Finally, some modern grammars also acknowledge so-called small clauses, which often lack a verb altogether. It should be apparent that non-finite clauses are (by and large) embedded clauses.

Gerund clauses

The underlined words in the following examples are judged to be non-finite clauses, e.g.

- a. Bill stopping the project was a big disappointment. - Non-finite gerund clause
- b. Bill's stopping the project was a big disappointment. - Gerund with noun status
- a. We've heard about Susan attempting a solution. - Non-finite gerund clause
- b. We've heard about Susan's attempting a solution. - Gerund with noun status
- a. They mentioned him cheating on the test. - Non-finite gerund clause
- b. They mentioned his cheating on the test. - Gerund with noun status

Each of the gerunds in the a-sentences (*stopping*, *attempting*, and *cheating*) constitutes a non-finite clause. The subject-predicate relationship that has long been taken as the defining trait of clauses is fully present in the a-sentences. The fact that the b-sentences are also acceptable illustrates the enigmatic behavior of gerunds. They seem to straddle two syntactic categories: they can function as non-finite verbs or as nouns. When they function as nouns as in the b-sentences, it is debatable whether they constitute clauses, since nouns are not generally taken to be constitutive of clauses.

to-infinitive clauses

Some modern theories of syntax take many *to*-infinitives to be constitutive of non-finite clauses.^[5] This stance is supported by the clear predicate status of many *to*-infinitives. It is challenged, however, by the fact that *to*-infinitives do not take an overt subject, e.g.

- a. She refuses to consider the issue.
- a. He attempted to explain his concerns.

The *to*-infinitives *to consider* and *to explain* clearly qualify as predicates (because they can be negated). They do not, however, take overt subjects. The subjects *she* and *he* are dependents of the matrix verbs *refuses* and *attempted*, respectively, not of the *to*-infinitives. Data like these are often addressed in terms of control. The matrix predicates *refuses* and *attempted* are control verbs; they control the embedded predicates *consider* and *explain*, which means they determine which of their arguments serves as the subject argument of the embedded predicate. Some theories of syntax posit the null subject PRO (=pronoun) to help address the facts of control constructions, e.g.

- b. She refuses PRO to consider the issue.
- b. He attempted PRO to explain his concerns.

With the presence of PRO as a null subject, *to*-infinitives can be construed as complete clauses, since both subject and predicate are present.

One must keep in mind, though, that PRO-theory is particular to one tradition in the study of syntax and grammar (Government and Binding Theory, Minimalist Program). Other theories of syntax and grammar (e.g. Head-Driven Phrase Structure Grammar, Construction Grammar, dependency grammar) reject the presence of null elements such as PRO, which means they are likely to reject the stance that *to*-infinitives constitute clauses.

Small clauses

Another type of construction that some schools of syntax and grammar view as non-finite clauses is the so-called small clause. A typical small clause consists of a noun phrase and a predicative expression,^[6] e.g.

We consider that a joke. - Small clause with the predicative noun phrase *a joke*

Something made him angry. - Small clause with the predicative adjective *angry*

She wants us to stay. - Small clause with the predicative non-finite *to-infinitive* *to stay*

The subject-predicate relationship is clearly present in the underlined strings. The expression on the right is a predication over the noun phrase immediately to its left. While the subject-predicate relationship is indisputably present, the underlined strings do not behave as single constituents, a fact that undermines their status as clauses. Hence one can debate whether the underlined strings in these examples should qualify as clauses. The layered structures of the chomskyan tradition are again likely to view the underlined strings as clauses, whereas the schools of syntax that posit flatter structures are likely to reject clause status for them.

Notes

- [1] For this basic definition in terms of a proposition, see Kroeger (2005:32).
- [2] For a definition of the clause that emphasizes the subject-predicate relationship, see Radford (2004:327f.).
- [3] Most basic discussions of the clause emphasize the distinction between main and subordinate clauses. See for instance Crystal (1997:62).
- [4] Numerous dependency grammar trees like the ones produced here can be found, for instance, in Osborne and Groß (2012).
- [5] For an example of a grammar that acknowledges non-finite *to-infinitive* clauses, see Radford (2004:23).
- [6] For the basic characteristics of small clauses, see Crystal (1997:62).

References

- Crystal, D. 1997. A dictionary of linguistics and phonetics, fourth edition. Oxford, UK: Blackwell Publishers.
- Kroeger, P. 2005. Analysing Grammar: An Introduction. Cambridge, UK: Cambridge University Press.
- Osborne, T. and T. Groß 2012. Constructions are catenae: Construction Grammar meets Dependency Grammar. Cognitive Linguistics 23, 1, 163-214.
- Radford, A. 2004. English syntax: An introduction. Cambridge, UK: Cambridge University Press.

Phrase

In everyday speech, a **phrase** may refer to any group of words. In linguistics, a phrase is a group of words (or sometimes a single word) that form a constituent and so function as a single unit in the syntax of a sentence. A phrase is lower on the grammatical hierarchy than a clause.^[1]

Examples

Examine the following sentence:

The house at the end of the street is not red.

The words in bold form a phrase; together they act like a noun. This phrase can be further broken down; a prepositional phrase functioning as an adjective can be identified:

at the end of the street

Further, a smaller prepositional phrase can be identified inside this greater prepositional phrase:

of the street

And within the greater prepositional phrase, one can identify a noun phrase:

the end of the street

Phrases can be identified by constituency tests such as proform substitution (=replacement). For instance, the prepositional phrase *at the end of the street* could be replaced by an adjective such as *nearby*: *the nearby house* or even *the house nearby*. *The end of the street* could also be replaced by another noun phrase, such as *the crossroads* to produce *the house at the crossroads*.

Heads and dependents

Most phrases have an important word defining the type and linguistic features of the phrase. This word is the head of the phrase and gives its name to the phrase category.^[2] The heads in the following phrases are in bold:

too slowly - Adverb phrase (AdvP)

very happy - Adjective phrase (AP)

the massive dinosaur - Noun phrase (NP)

at lunch - Preposition phrase (PP)

watch TV - Verb phrase (VP)

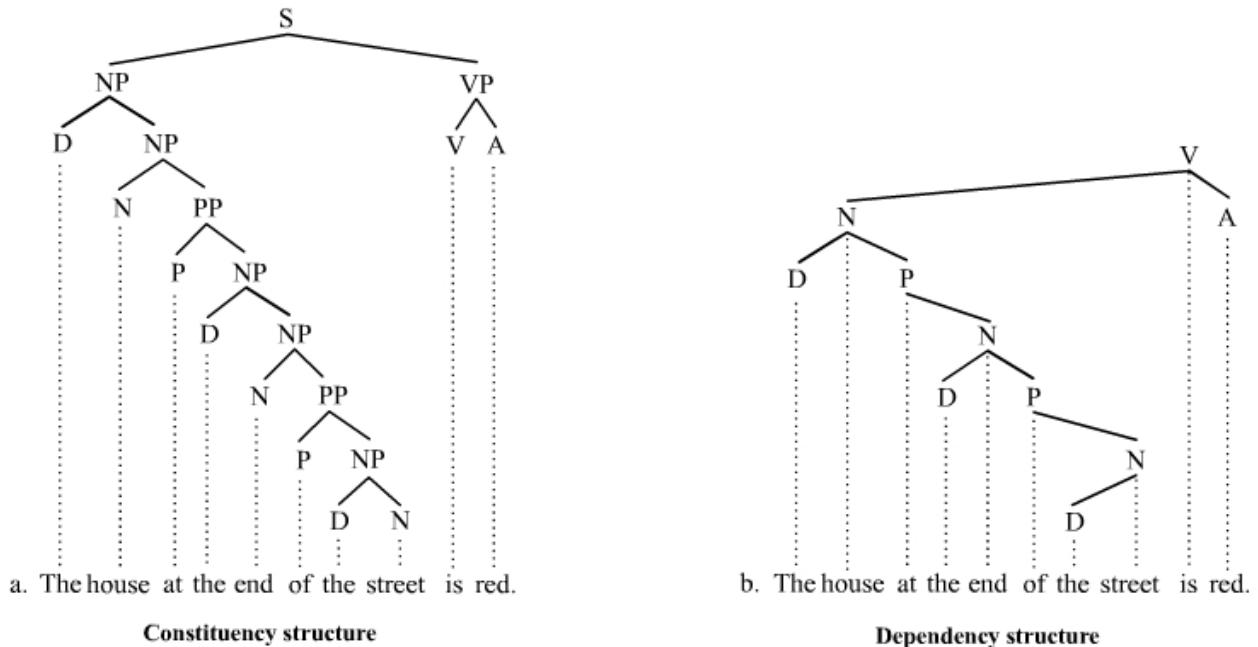
The head can be distinguished from its *dependents* (the rest of the phrase other than the head) because the head of the phrase determines many of the grammatical features of the phrase as a whole. The examples just given show the five most commonly acknowledged types of phrases. Further phrase types can be assumed, although doing so is not common. For instance one might acknowledge subordinator phrases:

before that happened - Subordinator phrase (SP)

This "phrase" is more commonly classified as a full subordinate clause and therefore many grammars would not label it as a phrase. If one follows the reasoning of heads and dependents, however, then subordinate clauses should indeed qualify as phrases. Most theories of syntax see most if not all phrases as having a head. Sometimes, however, non-headed phrases are acknowledged. If a phrase lacks a head, it is known as exocentric, whereas phrases with heads are endocentric.

Representing phrases

Many theories of syntax and grammar represent sentence structure using trees. The trees provide schematic illustrations of how the words of sentences are grouped. These representations show the words, phrases, and at times clauses that make up sentences.^[3] Any word combination that corresponds to a complete subtree can be seen as a phrase. There are two competing principles for producing trees, constituency and dependency. Both of these principles are illustrated here using the example sentence from above. The constituency-based tree is on the left, and the dependency-based tree on the right:



The constituency-based tree on the left is associated with a traditional phrase structure grammar, and the tree on the right is one of a dependency grammar. The node labels in the trees (e.g. N, NP, V, VP) mark the syntactic category of the constituents. Both trees take a phrase to be any combination of words that corresponds to a complete subtree. In the constituency tree on the left, each phrasal node (marked with P) identifies a phrase; there are therefore 8 phrases in the constituency tree. In the dependency tree on the right, each node that dominates one or more other nodes corresponds to a phrase; there are therefore 5 (or 6 if the whole sentence is included) phrases in the dependency tree. What the trees and the numbers demonstrate is that theories of syntax differ in what they deem to qualify as a phrase. The constituency tree takes three word combinations to be phrases (*house at the end of the street*, *end of the street*, and *is red*) that the dependency tree does not judge to be phrases. Which of the two tree structures is more plausible can be determined in part by empirical considerations, such as those delivered by constituency tests.

Confusion: phrases in theories of syntax

The common use of the term "phrase" is different from that employed by some phrase structure theories of syntax. The everyday understanding of the phrase is that it consists of two or more words, whereas depending on the theory of syntax that one employs, individual words may or may not qualify as phrases.^[4] The trees in the previous section, for instance, do not view individual words as phrases. Theories of syntax that employ X-bar theory, in contrast, will acknowledge many individual words as phrases. This practice is due to the fact that sentence structure is analyzed in terms of a universal schema, the X-bar schema, which sees each head as projecting at least three levels of structure: a minimal level, an intermediate level, and a maximal level. Thus an individual noun, such as *Susan* in *Susan laughed*, will project up to an intermediate level and a maximal level, which means that *Susan* qualifies as a phrase. This concept of the phrase is a source of confusion for students of syntax.

Many other theories of syntax do not employ the X-bar schema and are therefore less likely to encounter this confusion. For instance, dependency grammars do not acknowledge phrase structure in the manner associated with phrase structure grammars and therefore do not acknowledge individual words as phrases, a fact that is evident in the dependency grammar trees above and below.

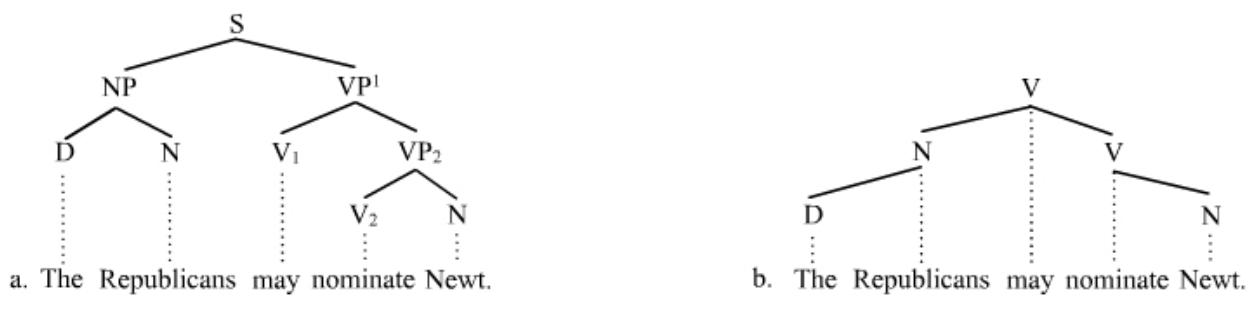
The verb phrase (VP) as a source of controversy

Most if not all theories of syntax acknowledge verb phrases (VPs), but they can diverge greatly in the types of verb phrases that they posit. Phrase structure grammars acknowledge both finite verb phrases and non-finite verb phrases as constituents. Dependency grammars, in contrast, acknowledge just non-finite verb phrases as constituents. The distinction is illustrated with the following examples:

The Republicans **may nominate Newt**. - Finite VP in bold

The Republicans may **nominate Newt**. - Non-finite VP in bold

The syntax trees of this sentence are next:



The constituency tree on the left shows the finite verb string *may nominate Newt* as a phrase (= constituent); it corresponds to VP₁. In contrast, this same string is not shown as a phrase in the dependency tree on the right. Observe that both trees, however, take the non-finite VP string *nominate Newt* to be a phrase, since in both trees *nominate Newt* corresponds to a complete subtree.

Since there is disagreement concerning the status of finite VPs (whether they are constituents or not), empirical considerations are needed. Grammarians can (again) employ constituency tests to shed light on the controversy. Constituency tests are diagnostics for identifying the constituents of sentences and they are thus essential for identifying phrases. The results of most constituency tests do not support the existence of a finite VP constituent.^[5]

Notes

[1] Kroeger 2005:35

[2] Kroeger 2005:37

[3] For a good introduction and discussion of phrases and the tree structures that represent phrases, see Slobin (2011:29ff.).

[4] Finch (2000:112) sees a phrase consisting of two or more words; individual words do not count as phrases.

[5] Concerning the inability of most constituency tests to identify finite VP as a constituent, see Miller (2011:54f.) and Osborne (2011:323f.).

References

- Finch, G. 2000. Linguistic terms and concepts. New York: St. Martin's Press.
- Kroeger, Paul 2005. Analyzing grammar: An introduction. Cambridge University Press.
- Miller, J. 2011. A critical introduction to syntax. London: continuum.
- Osborne, Timothy, Michael Putnam, and Thomas Gross 2011. Bare phrase structure, label-less structures, and specifier-less syntax: Is Minimalism becoming a dependency grammar? *The Linguistic Review* 28: 315-364.
- Slobin, N. 2011. Syntactic analysis: The basics. Malden, MA: Wiley-Blackwell.

External links

- The Phrase Finder (<http://www.phrases.org.uk/meanings/>) - The meanings and origins of phrases, sayings, and idioms
- Phrases.net (<http://www.phrases.net/>) - A large collection of common phrases that can be heard and translated to several languages.
- Phras.in (<http://phras.in>) - An online tool that helps choosing the correct phrasing, based on web results frequency.
- phraseup* (<http://www.phraseup.com>) - A writing assistant that helps with completing sentences by finding the missing words we can't recall.
- Fraze.it (<http://www.fraze.it>) - A search engine for sentences and phrases. Supports six languages, filtered by form, zone, context, etc.

Word

In language, a **word** is the smallest element that may be uttered in isolation with semantic or pragmatic content (with literal or practical meaning). This contrasts with a morpheme, which is the smallest unit of meaning but will not necessarily stand on its own. A word may consist of a single morpheme (for example: *oh!*, *rock*, *red*, *quick*, *run*, *expect*), or several (*rocks*, *redness*, *quickly*, *running*, *unexpected*), whereas a morpheme may not be able to stand on its own as a word (in the words just mentioned, these are *-s*, *-ness*, *-ly*, *-ing*, *un-*, *-ed*). A complex word will typically include a root and one or more affixes (*rock-s*, *red-ness*, *quick-ly*, *run-ning*, *un-expect-ed*), or more than one root in a compound (*black-board*, *rat-race*). Words can be put together to build larger elements of language, such as phrases (*a red rock*), clauses (*I threw a rock*), and sentences (*He threw a rock too but he missed*).

Words are merely Symbols which represents the intended meaning behind, which can be known from the context. The term *word* may refer to a spoken word or to a written word, or sometimes to the abstract concept behind either. Spoken words are made up of units of sound called phonemes, and written words of symbols called graphemes, such as the letters of the English alphabet.

Definitions

The ease or difficulty of deciphering a word depends on the language. Dictionaries categorize a language's lexicon (i.e., its vocabulary) into lemmas. These can be taken as an indication of what constitutes a "word" in the opinion of the writers of that language.

Semantic definition

Leonard Bloomfield introduced the concept of "Minimal Free Forms" in 1926. Words are thought of as the smallest meaningful unit of speech that can stand by themselves.^[1] This correlates phonemes (units of sound) to lexemes (units of meaning). However, some written words are not minimal free forms, as they make no sense by themselves (for example, *the* and *of*).^[2]

Some semanticists have proposed a theory of so-called semantic primitives or semantic primes, indefinable words representing fundamental concepts that are intuitively meaningful. According to this theory, semantic primes serve as the basis for describing the meaning, without circularity, of other words and their associated conceptual denotations.^[3]

Features

In the Minimalist school of theoretical syntax, words (also called *lexical items* in the literature) are construed as "bundles" of linguistic features that are united into a structure with form and meaning.^[4] For example, the word "bears" has semantic features (it denotes real-world objects, bears), category features (it is a noun), number features (it is plural and must agree with verbs, pronouns, and demonstratives in its domain), phonological features (it is pronounced a certain way), etc.

Word boundaries

The task of defining what constitutes a "word" involves determining where one word ends and another word begins—in other words, identifying word boundaries. There are several ways to determine where the word boundaries of spoken language should be placed:

- **Potential pause:** A speaker is told to repeat a given sentence slowly, allowing for pauses. The speaker will tend to insert pauses at the word boundaries. However, this method is not foolproof: the speaker could easily break up polysyllabic words, or fail to separate two or more closely related words.
- **Indivisibility:** A speaker is told to say a sentence out loud, and then is told to say the sentence again with extra words added to it. Thus, *I have lived in this village for ten years* might become *My family and I have lived in this little village for about ten or so years*. These extra words will tend to be added in the word boundaries of the original sentence. However, some languages have infixes, which are put inside a word. Similarly, some have separable affixes; in the German sentence "Ich **komme** gut zu Hause **an**", the verb *ankommen* is separated.
- **Phonetic boundaries:** Some languages have particular rules of pronunciation that make it easy to spot where a word boundary should be. For example, in a language that regularly stresses the last syllable of a word, a word boundary is likely to fall after each stressed syllable. Another example can be seen in a language that has vowel harmony (like Turkish):^[5] the vowels within a given word share the same *quality*, so a word boundary is likely to occur whenever the vowel quality changes. Nevertheless, not all languages have such convenient phonetic rules, and even those that do present the occasional exceptions.
- **Orthographic boundaries:** See below.

Orthography

In languages with a literary tradition, there is interrelation between orthography and the question of what is considered a single word. Word separators (typically spaces) are common in modern orthography of languages using alphabetic scripts, but these are (excepting isolated precedents) a relatively modern development (see also history of writing).

In English orthography, compound expressions may contain spaces. Examples are *ice cream*, *air raid shelter*, *get up*, and these must thus be considered as more than one word. (*Ice*, *cream*, *air* etc. indisputably exist as free forms, the case of *get* is less clear.) In contrast, *brownstone* is spelt as a single word and would thus be considered as such for most purposes even though *brown* and *stone* are free forms.

Vietnamese orthography, although using the Latin alphabet, delimits monosyllabic morphemes, not words. East Asian orthographies (languages using CJK characters) also tend to delimit syllables (in the case of Chinese characters) or morae (in the case of kana) rather than full words. Hangul the Korean alphabet, delimits both syllables and words, by grouping graphemes into syllabic blocks but also adds spaces between words. Conversely, synthetic languages often combine many lexical morphemes into single words, making it difficult to boil them down to the traditional sense of words found more easily in analytic languages; this is especially difficult for polysynthetic languages, such as Inuktitut and Ubykh, where entire sentences may consist of a single word.

Morphology

In synthetic languages, a single word stem (for example, *love*) may have a number of different forms (for example, *loves*, *loving*, and *loved*). However for some purposes these are not usually considered to be different words, but rather different forms of the same word. In these languages, words may be considered to be constructed from a number of morphemes. In Indo-European languages in particular, the morphemes distinguished are

- the root
- optional suffixes
- a desinence, or inflectional suffix.

Thus, the Proto-Indo-European **wṛdhom* would be analyzed as consisting of

1. **wṛ-*, the zero grade of the root **wer-*
2. a root-extension *-dh-* (diachronically a suffix), resulting in a complex root **wṛdh-*
3. The thematic suffix **-o-*
4. the neuter gender nominative or accusative singular desinence **-m*.

Philosophy

Philosophers have found words objects of fascination since at least the 5th century BC, with the foundation of the philosophy of language. Plato analyzed words in terms of their origins and the sounds making them up, concluding that there was some connection between sound and meaning, though words change a great deal over time. John Locke wrote that the use of words "is to be sensible marks of ideas", though they are chosen "not by any natural connexion that there is between particular articulate sounds and certain ideas, for then there would be but one language amongst all men; but by a voluntary imposition, whereby such a word is made arbitrarily the mark of such an idea".^[6] Wittgenstein's thought transitioned from a word as representation of meaning to "the meaning of a word is its use in the language."^[7]

Classes

Grammar classifies a language's lexicon into several groups of words. The basic bipartite division possible for virtually every natural language is that of nouns vs. verbs.

The classification into such classes is in the tradition of Dionysius Thrax, who distinguished eight categories: noun, verb, adjective, pronoun, preposition, adverb, conjunction and interjection.

In Indian grammatical tradition, Pāṇini introduced a similar fundamental classification into a nominal (nāma, suP) and a verbal (ākhyāta, tiN) class, based on the set of desinences taken by the word.

Notes

[1] Katamba 11

[2] Fleming 77

[3] Wierzbicka 1996; Goddard 2002

[4] Adger (2003), pp. 36–7.

[5] Bauer 9

[6] "Locke ECHU BOOK III Chapter II Of the Signification of Words" (<http://www.rbjones.com/rbjpub/philos/classics/locke/ctb3c02.htm>). Rbjones.com. . Retrieved 13 March 2012.

[7] "Ludwig Wittgenstein (Stanford Encyclopedia of Philosophy)" (<http://plato.stanford.edu/entries/wittgenstein>). Plato.stanford.edu. . Retrieved 13 March 2012.

References

- Adger, David (2003). *Core Syntax: A Minimalist Approach*. Oxford: Oxford University Press. ISBN 0-19-924370-0.
- Barton, David (1994). *Literacy: An Introduction to the Ecology of Written Language*. Blackwell Publishing. p. 96.
- Bauer, Laurie (1983). *English Word-formation*. Cambridge: Cambridge University Press. ISBN 0-521-28492-9.
- Brown, Keith R. (Ed.) (2005) Encyclopedia of Language and Linguistics (2nd ed.). Elsevier. 14 vols.
- Crystal, David (1995). *The Cambridge Encyclopedia of the English Language* (1 ed.). Cambridge: Cambridge University Press. ISBN 0-521-40179-8.
- Fleming, Michael et al. (2001). *Meeting the Standards in Secondary English: A Guide to the ITT NC*. Routledge. p. 77. ISBN 0-415-23377-1.
- Goddard, Cliff (2002). "The search for the shared semantic core of all languages" (http://www.une.edu.au/lcl/nsm/pdf/Goddard_Ch1_2002.pdf). In Cliff Goddard and Anna Wierzbicka. *Meaning and Universal Grammar: Theory and Empirical Findings. Volume I*. Amsterdam: John Benjamins. pp. 5–40
- Katamba, Francis (2005). *English Words: Structure, History, Usage*. Routledge. ISBN 0-415-29893-8.
- Plag, Ingo (2003). *Word-formation in English*. Cambridge: Cambridge University Press. ISBN 0-521-52563-2.
- Simpson, J.A. and E.S.C. Weiner, ed. (1989). *Oxford English Dictionary* (2 ed.). Clarendon Press. ISBN 0-19-861186-2.
- Wierzbicka, Anna (1996). *Semantics: Primes and Universals*. Oxford University Press. ISBN 0-19-870002-4.

External links

- What Is a Word? (<http://www.sussex.ac.uk/english/documents/essay---what-is-a-word.pdf>) – a working paper by Larry Trask (see (<http://www.sussex.ac.uk/english/research/projects/linguisticspapers>) for attribution), Department of Linguistics and English Language, University of Sussex.
- For a different type of views cf. Wor(L)d- Spaces: The Definition of 'Word' (<http://ssrn.com/abstract=2016136>) Encyclopedia of Science of Language, Polimetrica Onlus, 2007. Milan, Italy.

Morphology (linguistics)

In linguistics, **morphology** is the identification, analysis and description of the structure of a given language's morphemes and other linguistic units, such as root words, affixes, parts of speech, intonation/stress, or implied context (words in a *lexicon* are the subject matter of *lexicology*). Morphological typology represents a method for classifying languages according to the ways by which morphemes are used in a language —from the analytic that use only isolated morphemes, through the agglutinative ("stuck-together") and fusional languages that use bound morphemes (affixes), up to the polysynthetic, which compress many separate morphemes into single words.

While words are generally accepted as being (with clitics) the smallest units of syntax, it is clear that in most languages, if not all, words can be related to other words by rules (grammars). For example, English speakers recognize that the words *dog* and *dogs* are closely related — differentiated only by the *plurality morpheme* "-s", which is only found bound to nouns, and is never separate. Speakers of English (a fusional language) recognize these relations from their tacit knowledge of the rules of word formation in English. They infer intuitively that *dog* is to *dogs* as *cat* is to *cats*; similarly, *dog* is to *dog catcher* as *dish* is to *dishwasher*, in one sense. The rules understood by the speaker reflect specific patterns, or regularities, in the way words are formed from smaller units and how those smaller units interact in speech. In this way, morphology is the branch of linguistics that studies patterns of word formation within and across languages, and attempts to formulate rules that model the knowledge of the speakers of those languages.

A language like Classical Chinese instead uses unbound ("free") morphemes, but depends on post-phrase affixes, and word order to convey meaning. However, this cannot be said of present-day Mandarin, in which most words are compounds (around 80%), and most roots are bound.

In the Chinese languages, these are understood as grammars that represent the morphology of the language. Beyond the agglutinative languages, a polysynthetic language like Chukchi will have words composed of many morphemes: The word "*təmeyŋəlevtpəyterkən*" is composed of eight morphemes *t-θ-meyŋ-θ-levt-pəγt-θ-rkən*, that can be glossed 1.SG.SUBJ-great-head-hurt-PRES.1, meaning 'I have a fierce headache.' The morphology of such languages allows for each consonant and vowel to be understood as morphemes, just as the grammars of the language key the usage and understanding of each morpheme.

The discipline that deals specifically with the sound changes occurring within morphemes is called *morphophonology*.

History

The history of morphological analysis dates back to the ancient Indian linguist Pāṇini, who formulated the 3,959 rules of Sanskrit morphology in the text *Aṣṭādhyāyī* by using a constituency grammar. The Greco-Roman grammatical tradition also engaged in morphological analysis. Studies in Arabic morphology, conducted by Marāḥ al-arwāḥ and Ahmad b. ‘alī Mas‘ūd, date back to at least 1200 CE.^[1]

The term *morphology* was coined by August Schleicher in 1859.^[2]

Fundamental concepts

Lexemes and word forms

The distinction between these two senses of "word" is arguably the most important one in morphology. The first sense of "word", the one in which *dog* and *dogs* are "the same word", is called a lexeme. The second sense is called *word form*. We thus say that *dog* and *dogs* are different forms of the same lexeme. *Dog* and *dog catcher*, on the other hand, are different lexemes, as they refer to two different kinds of entities. The form of a word that is chosen conventionally to represent the canonical form of a word is called a lemma, or citation form.

Prosodic word vs. morphological word

Here are examples from other languages of the failure of a single phonological word to coincide with a single morphological word form. In Latin, one way to express the concept of 'NOUN-PHRASE₁ and NOUN-PHRASE₂' (as in "apples and oranges") is to suffix '-que' to the second noun phrase: "apples oranges-and", as it were. An extreme level of this theoretical quandary posed by some phonological words is provided by the Kwak'wala language.^[3] In Kwak'wala, as in a great many other languages, meaning relations between nouns, including possession and "semantic case", are formulated by affixes instead of by independent "words". The three-word English phrase, "with his club", where 'with' identifies its dependent noun phrase as an instrument and 'his' denotes a possession relation, would consist of two words or even just one word in many languages. Unlike most languages, Kwak'wala semantic affixes phonologically attach not to the lexeme they pertain to semantically, but to the *preceding* lexeme. Consider the following example (in Kwak'wala, sentences begin with what corresponds to an English verb):^[4]

kwix?id-i-da bəgwanəma_i-χ-a q'asa-s-is_i t'alwagwayu

Morpheme by morpheme translation:

kwix?id-i-da = clubbed-PIVOT-DETERMINER
 bəgwanəma-χ-a = man-ACCUSATIVE-DETERMINER
 q'asa-s-is = otter-INSTRUMENTAL-3SG-POSSESSIVE
 t'alwagwayu = club.

"the man clubbed the otter with his club"

(Notation notes:

1. accusative case marks an entity that something is done to.
2. determiners are words such as "the", "this", "that".
3. the concept of "pivot" is a theoretical construct that is not relevant to this discussion.)

That is, to the speaker of Kwak'wala, the sentence does not contain the "words" 'him-the-otter' or 'with-his-club'. Instead, the markers *-i-da* (PIVOT-'the'), referring to *man*, attaches not to *bəgwanəma* ('man'), but instead to the "verb"; the markers *-χ-a* (ACCUSATIVE-'the'), referring to *otter*, attach to *bəgwanəma* instead of to *q'asa* ('otter'), etc. To summarize differently: a speaker of Kwak'wala does *not* perceive the sentence to consist of these phonological words:

kwix?id i-da-bəgwanəma χ-a-q'asa s-is_i-t'alwagwayu

clubbed PIVOT-the-man_i hit-the-otter with-his_i-club

A central publication on this topic is the recent volume edited by Dixon and Aikhenvald (2007), examining the mismatch between prosodic-phonological and grammatical definitions of "word" in various Amazonian, Australian Aboriginal, Caucasian, Eskimo, Indo-European, Native North American, West African, and sign languages. Apparently, a wide variety of languages make use of the hybrid linguistic unit clitic, possessing the grammatical features of independent words but the prosodic-phonological lack of freedom of bound morphemes. The intermediate status of clitics poses a considerable challenge to linguistic theory.

Inflection vs. word formation

Given the notion of a lexeme, it is possible to distinguish two kinds of morphological rules. Some morphological rules relate to different forms of the same lexeme; while other rules relate to different lexemes. Rules of the first kind are called *inflectional rules*, while those of the second kind are called *word formation*. The English plural, as illustrated by *dog* and *dogs*, is an inflectional rule; compound phrases and words like *dog catcher* or *dishwasher* provide an example of a word formation rule. Informally, word formation rules form "new words" (that is, new lexemes), while inflection rules yield variant forms of the "same" word (lexeme).

There is a further distinction between two kinds of word formation: derivation and compounding. Compounding is a process of word formation that involves combining complete word forms into a single *compound* form; *dog catcher* is therefore a compound, because both *dog* and *catcher* are complete word forms in their own right before the compounding process has been applied, and are subsequently treated as one form. Derivation involves affixing bound (non-independent) forms to existing lexemes, whereby the addition of the affix *derives* a new lexeme. One example of derivation is clear in this case: the word *independent* is derived from the word *dependent* by prefixing it with the derivational prefix *in-*, while *dependent* itself is derived from the verb *depend*.

The distinction between inflection and word formation is not at all clear cut. There are many examples where linguists fail to agree whether a given rule is inflection or word formation. The next section will attempt to clarify this distinction.

Word formation is a process, as we have said, where you combine two complete words, whereas with inflection you can combine a suffix with some verb to change its form to subject of the sentence. For example: in the present indefinite, we use 'go' with subject I/we/you/they and plural nouns, whereas for third person singular pronouns (he/she/it) and singular nouns we use 'goes'. So this '-es' is an inflectional marker and is used to match with its subject. A further difference is that in word formation, the resultant word may differ from its source word's grammatical category whereas in the process of inflection the word never changes its grammatical category.

Paradigms and morphosyntax

A linguistic paradigm is the complete set of related word forms associated with a given lexeme. The familiar examples of paradigms are the conjugations of verbs, and the declensions of nouns. Accordingly, the word forms of a lexeme may be arranged conveniently into tables, by classifying them according to shared inflectional categories such as tense, aspect, mood, number, gender or case. For example, the personal pronouns in English can be organized into tables, using the categories of person (first, second, third), number (singular vs. plural), gender (masculine, feminine, neuter), and case (subjective, objective, and possessive). See English personal pronouns for the details.

The inflectional categories used to group word forms into paradigms cannot be chosen arbitrarily; they must be categories that are relevant to stating the syntactic rules of the language. For example, person and number are categories that can be used to define paradigms in English, because English has grammatical agreement rules that require the verb in a sentence to appear in an inflectional form that matches the person and number of the subject. In other words, the syntactic rules of English care about the difference between *dog* and *dogs*, because the choice between these two forms determines which form of the verb is to be used. In contrast, however, no syntactic rule of English cares about the difference between *dog* and *dog catcher*, or *dependent* and *independent*. The first two are just nouns, and the second two just adjectives, and they generally behave like any other noun or adjective behaves.

An important difference between inflection and word formation is that inflected word forms of lexemes are organized into paradigms, which are defined by the requirements of syntactic rules, whereas the rules of word formation are not restricted by any corresponding requirements of syntax. Inflection is therefore said to be relevant to syntax, and word formation is not. The part of morphology that covers the relationship between syntax and morphology is called morphosyntax, and it concerns itself with inflection and paradigms, but not with word formation or compounding.

Allomorphy

In the exposition above, morphological rules are described as analogies between word forms: *dog* is to *dogs* as *cat* is to *cats*, and as *dish* is to *dishes*. In this case, the analogy applies both to the form of the words and to their meaning: in each pair, the first word means "one of X", while the second "two or more of X", and the difference is always the plural form *-s* affixed to the second word, signaling the key distinction between singular and plural entities.

One of the largest sources of complexity in morphology is that this one-to-one correspondence between meaning and form scarcely applies to every case in the language. In English, there are word form pairs like *ox/oxen*, *goose/geese*, and *sheep/sheep*, where the difference between the singular and the plural is signaled in a way that departs from the regular pattern, or is not signaled at all. Even cases considered "regular", with the final *-s*, are not so simple; the *-s* in *dogs* is not pronounced the same way as the *-s* in *cats*, and in a plural like *dishes*, an "extra" vowel appears before the *-s*. These cases, where the same distinction is effected by alternative forms of a "word", are called allomorphy.

Phonological rules constrain which sounds can appear next to each other in a language, and morphological rules, when applied blindly, would often violate phonological rules, by resulting in sound sequences that are prohibited in the language in question. For example, to form the plural of *dish* by simply appending an *-s* to the end of the word would result in the form *[dɪʃs], which is not permitted by the phonotactics of English. In order to "rescue" the word, a vowel sound is inserted between the root and the plural marker, and [dɪʃɪz] results. Similar rules apply to the pronunciation of the *-s* in *dogs* and *cats*: it depends on the quality (voiced vs. unvoiced) of the final preceding phoneme.

Lexical morphology

Lexical morphology is the branch of morphology that deals with the lexicon, which, morphologically conceived, is the collection of lexemes in a language. As such, it concerns itself primarily with word formation: derivation and compounding.

Models

There are three principal approaches to morphology, which each try to capture the distinctions above in different ways. These are,

- Morpheme-based morphology, which makes use of an Item-and-Arrangement approach.
- Lexeme-based morphology, which normally makes use of an Item-and-Process approach.
- Word-based morphology, which normally makes use of a Word-and-Paradigm approach.

Note that while the associations indicated between the concepts in each item in that list is very strong, it is not absolute.

Morpheme-based morphology

In morpheme-based morphology, word forms are analyzed as arrangements of morphemes. A morpheme is defined as the minimal meaningful unit of a language. In a word like *independently*, we say that the morphemes are *in-*, *depend*, *-ent*, and *ly*; *depend* is the root and the other morphemes are, in this case, derivational affixes.^[5] In a word like *dogs*, we say that *dog* is the root, and that *-s* is an inflectional morpheme. In its simplest (and most naïve) form, this way of analyzing word forms treats words as if they were made of morphemes put after each other like beads on a string, is called Item-and-Arrangement. More modern and sophisticated approaches seek to maintain the idea of the morpheme while accommodating non-concatenative, analogical, and other processes that have proven problematic for Item-and-Arrangement theories and similar approaches.

Morpheme-based morphology presumes three basic axioms (cf. Beard 1995 for an overview and references):

- Baudoin's **single morpheme hypothesis**: Roots and affixes have the same status as *morphemes*.

- Bloomfield's **sign base morpheme hypothesis**: As morphemes, they are dualistic signs, since they have both (phonological) form and meaning.
- Bloomfield's **lexical morpheme hypothesis**: The morphemes, affixes and roots alike, are stored in the lexicon.

Morpheme-based morphology comes in two flavours, one Bloomfieldian and one Hockettian. (cf. Bloomfield 1933 and Charles F. Hockett 1947). For Bloomfield, the *morpheme* was the minimal form with meaning, but it was not meaning itself. For Hockett, morphemes are *meaning elements*, not *form elements*. For him, there is a *morpheme plural*, with the *allomorphs* -s, -en, -ren etc. Within much morpheme-based morphological theory, these two views are mixed in unsystematic ways, so that a writer may talk about "the morpheme *plural*" and "the morpheme -s" in the same sentence, although these are different things.

Lexeme-based morphology

Lexeme-based morphology is (usually) an Item-and-Process approach. Instead of analyzing a word form as a set of morphemes arranged in sequence, a word form is said to be the result of applying rules that *alter* a word form or stem in order to produce a new one. An inflectional rule takes a stem, changes it as is required by the rule, and outputs a word form; a derivational rule takes a stem, changes it as per its own requirements, and outputs a derived stem; a compounding rule takes word forms, and similarly outputs a compound stem.

Word-based morphology

Word-based morphology is (usually) a Word-and-paradigm approach. This theory takes paradigms as a central notion. Instead of stating rules to combine morphemes into word forms, or to generate word forms from stems, word-based morphology states generalizations that hold between the forms of inflectional paradigms. The major point behind this approach is that many such generalizations are hard to state with either of the other approaches. The examples are usually drawn from fusional languages, where a given "piece" of a word, which a morpheme-based theory would call an inflectional morpheme, corresponds to a combination of grammatical categories, for example, "third person plural." Morpheme-based theories usually have no problems with this situation, since one just says that a given morpheme has two categories. Item-and-Process theories, on the other hand, often break down in cases like these, because they all too often assume that there will be two separate rules here, one for third person, and the other for plural, but the distinction between them turns out to be artificial. Word-and-Paradigm approaches treat these as whole words that are related to each other by analogical rules. Words can be categorized based on the pattern they fit into. This applies both to existing words and to new ones. Application of a pattern different from the one that has been used historically can give rise to a new word, such as *older* replacing *elder* (where *older* follows the normal pattern of adjectival superlatives) and *cows* replacing *kine* (where *cows* fits the regular pattern of plural formation).

Morphological typology

In the 19th century, philologists devised a now classic classification of languages according to their morphology. According to this typology, some languages are isolating, and have little to no morphology; others are agglutinative, and their words tend to have lots of easily separable morphemes; while others yet are inflectional or fusional, because their inflectional morphemes are "fused" together. This leads to one bound morpheme conveying multiple pieces of information. The classic example of an isolating language is Chinese; the classic example of an agglutinative language is Turkish; both Latin and Greek are classic examples of fusional languages.

Considering the variability of the world's languages, it becomes clear that this classification is not at all clear cut, and many languages do not neatly fit any one of these types, and some fit in more than one way. A continuum of complex morphology of language may be adapted when considering languages.

The three models of morphology stem from attempts to analyze languages that more or less match different categories in this typology. The Item-and-Arrangement approach fits very naturally with agglutinative languages;

while the Item-and-Process and Word-and-Paradigm approaches usually address fusional languages.

The reader should also note that the classical typology mostly applies to inflectional morphology. There is very little fusion going on with word formation. Languages may be classified as synthetic or analytic in their word formation, depending on the preferred way of expressing notions that are not inflectional: either by using word formation (synthetic), or by using syntactic phrases (analytic).

References

- [1] Arabic Morphology and Phonology (<http://www.brill.nl/Default.aspx?partid=75&pid=9506>)
- [2] *Für die Lehre von der Wortform wähle ich das Wort "Morphologie"* ("for the science of word formation, I choose the term 'morphology'", *Mémoires Acad. Impériale* 7/1/7, 35)
- [3] Formerly known as Kwakiutl, Kwak'wala belongs to the Northern branch of the Wakashan language family. "Kwakiutl" is still used to refer to the tribe itself, along with other terms.
- [4] Example taken from Foley 1998, using a modified transcription. This phenomenon of Kwak'wala was reported by Jacobsen as cited in van Valin and La Polla 1997.
- [5] The existence of words like *appendix* and *pending* in English does not mean that the English word *depend* is analyzed into a derivational prefix *de-* and a root *pend*. While all those were indeed once related to each other by morphological rules, this was so only in Latin, not in English. English borrowed the words from French and Latin, but not the morphological rules that allowed Latin speakers to combine *de-* and the verb *pendere* 'to hang' into the derivative *dependere*.

Further reading

(Abbreviations: CUP = Cambridge: Cambridge University Press)

- Anderson, Stephen R. (1992). *A-Morphous Morphology*. Cambridge: CUP.
- Aronoff, Mark. (1993). " Morphology by Itself (http://books.google.de/books?id=PFMH1qg-Z5kC&dq=morphology+by+itself&pg=PP1&ots=HKANzPIrQZ&sig=CXtHtx_Gr_cEW8Rz5hLglnw17f4&prev=http://www.google.de/search?q=Morphology+by+Itself&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:de:official&client=firefox-a&sa=X&oi=print&ct=title&cad=one-book-with-thumbnail"). Cambridge, MA: MIT Press.
- Aronoff, Mark. (2009). " Morphology: an interview with Mark Aronoff (http://www.revel.inf.br/site2007/_pdf/14/entrevistas/revel_12_interview_aronoff.pdf)". ReVEL, v. 7, n. 12, ISSN 1678-8931.
- Beard, Robert. (1995). *Lexeme-Morpheme Base Morphology* (<http://www.facstaff.bucknell.edu/rbeard/>). Albany, N.Y.: State University of New York Press. ISBN 0-7914-2471-5.
- Bauer, Laurie. (2003). *Introducing linguistic morphology* (2nd ed.). Washington, D.C.: Georgetown University Press. ISBN 0-87840-343-4.
- Bauer, Laurie. (2004). *A glossary of morphology*. Washington, D.C.: Georgetown UP.
- Bubenik, Vit. (1999). *An introduction to the study of morphology*. LINCON coursebooks in linguistics, 07. Muenchen: LINCOM Europa. ISBN 3-89586-570-2.
- Bybee, J. L. (1985). *Morphology: A Study of the Relation between Meaning and Form*. Amsterdam: John Benjamins.
- Dixon, R. M. W. and Aikhenvald, Alexandra Y. (Eds). (2007). *Word: A cross-linguistic typology*. Cambridge: Cambridge University Press
- Foley, William A. (1998). " Symmetrical Voice Systems and Precategoriality in Philippine Languages (<http://www.sultry.arts.usyd.edu.au/LFG98/austro/foley/frames/foley4.htm>)". Workshop: Voice and Grammatical Functions in Austronesian. University of Sydney.
- Haspelmath, Martin. (2002). *Understanding morphology*. London: Arnold (co-published by Oxford University Press). ISBN 0-340-76025-7 (hb); ISBN 0-340-76026-5 (pbk).
- Fabrega, Antonio & Sergio Scalise (2012). *Morphology: from Data to Theory*. Edinburgh: Edinburgh University Pres.

- Katamba, Francis. (1993). *Morphology*. Modern linguistics series. New York: St. Martin's Press. ISBN 0-312-10101-5 (hb). ISBN 0-312-10356-5 (pbk).
- Korsakov, A. K. (Andreĭ Konstantinovich). (1969) The use of tenses in English. Korsakov, A. K. Structure of Modern English pt. 1. oai:gial.edu:26766 at <http://www.language-archives.org/item/oai:gial.edu:26766>
- Matthews, Peter. (1991). *Morphology* (2nd ed.). CUP. ISBN 0-521-41043-6 (hb). ISBN 0-521-42256-6 (pbk).
- Mel'čuk, Igor A. (1993–2000). *Cours de morphologie générale*, vol. 1-5. Montreal: Presses de l'Université de Montréal.
- Mel'čuk, Igor A. (2006). *Aspects of the theory of morphology*. Berlin: Mouton.
- Scalise, Sergio. (1983). *Generative Morphology*, Dordrecht, Foris.
- Singh, Rajendra and Stanley Starosta (eds). (2003). *Explorations in Seamless Morphology*. SAGE Publications. ISBN 0-7619-9594-3 (hb).
- Spencer, Andrew. (1991). *Morphological theory: an introduction to word structure in generative grammar*. No. 2 in Blackwell textbooks in linguistics. Oxford: Blackwell. ISBN 0-631-16143-0 (hb); ISBN 0-631-16144-9 (pb)
- Spencer, Andrew and Zwicky, Arnold M. (Eds.) (1998). *The handbook of morphology*. Blackwell handbooks in linguistics. Oxford: Blackwell. ISBN 0-631-18544-5.
- Stump, Gregory T. (2001). *Inflectional morphology: a theory of paradigm structure*. No. 93 in Cambridge studies in linguistics. CUP. ISBN 0-521-78047-0 (hb).
- van Valin, Robert D., and LaPolla, Randy. (1997). *Syntax : Structure, Meaning And Function*. CUP
- Zuckermann, Ghil'ad. (2009). Hybridity versus Revivability: Multiple Causation, Forms and Patterns (http://www.zuckermann.org/pdf/Hybridity_vs_Revivability.pdf), *Journal of Language Contact*, Varia 2: 40-67.

Syntax

In linguistics, **syntax** (from Ancient Greek σύνταξις "arrangement" from σύν syn, "together", and τάξις táxis, "an ordering") is "the study of the principles and processes by which sentences are constructed in particular languages".^[1]

In addition to referring to the overarching discipline, the term *syntax* is also used to refer directly to the rules and principles that govern the sentence structure of any individual language, for example in "the syntax of Modern Irish." Modern research in syntax attempts to describe languages in terms of such rules. Many professionals in this discipline attempt to find general rules that apply to all natural languages.

The term *syntax* is also used to refer to the rules governing the behavior of mathematical systems, such as formal languages used in logic. (See Logical syntax).

Early history

Works on grammar were written long before modern syntax came about; the *Aṣṭādhyāyī* of Pāṇini is often cited as an example of a premodern work that approaches the sophistication of a modern syntactic theory.^[2] In the West, the school of thought that came to be known as "traditional grammar" began with the work of Dionysius Thrax.

For centuries, work in syntax was dominated by a framework known as *grammaire générale*, first expounded in 1660 by Antoine Arnauld in a book of the same title. This system took as its basic premise the assumption that language is a direct reflection of thought processes and therefore there is a single, most natural way to express a thought. (That *natural way*, coincidentally, was exactly the way it was expressed in French.)

However, in the 19th century, with the development of historical-comparative linguistics, linguists began to realize the sheer diversity of human language, and to question fundamental assumptions about the relationship between language and logic. It became apparent that there was no such thing as the most natural way to express a thought, and therefore logic could no longer be relied upon as a basis for studying the structure of language.

The Port-Royal grammar modeled the study of syntax upon that of logic (indeed, large parts of the Port-Royal Logic were copied or adapted from the *Grammaire générale*^[3]). Syntactic categories were identified with logical ones, and all sentences were analyzed in terms of "Subject – Copula – Predicate". Initially, this view was adopted even by the early comparative linguists such as Franz Bopp.

The central role of syntax within theoretical linguistics became clear only in the 20th century, which could reasonably be called the "century of syntactic theory" as far as linguistics is concerned. For a detailed and critical survey of the history of syntax in the last two centuries, see the monumental work by Giorgio Graffi (2001).^[4]

Modern theories

There are a number of theoretical approaches to the discipline of syntax. One school of thought, founded in the works of Derek Bickerton,^[5] sees syntax as a branch of biology, since it conceives of syntax as the study of linguistic knowledge as embodied in the human mind. Other linguists (e.g. Gerald Gazdar) take a more Platonic view, since they regard syntax to be the study of an abstract formal system.^[6] Yet others (e.g. Joseph Greenberg) consider grammar a taxonomical device to reach broad generalizations across languages.

Generative grammar

The hypothesis of generative grammar is that language is a structure of the human mind. The goal of generative grammar is to make a complete model of this inner language (known as *i-language*). This model could be used to describe all human language and to predict the grammaticality of any given utterance (that is, to predict whether the utterance would sound correct to native speakers of the language). This approach to language was pioneered by Noam Chomsky. Most generative theories (although not all of them) assume that syntax is based upon the constituent structure of sentences. Generative grammars are among the theories that focus primarily on the form of a sentence, rather than its communicative function.

Among the many generative theories of linguistics, the Chomskyan theories are:

- Transformational grammar (TG) (Original theory of generative syntax laid out by Chomsky in *Syntactic Structures* in 1957)^[7]
- Government and binding theory (GB) (revised theory in the tradition of TG developed mainly by Chomsky in the 1970s and 1980s)^[8]
- Minimalist program (MP) (a reworking of the theory out of the GB framework published by Chomsky in 1995)^[9]

Other theories that find their origin in the generative paradigm are:

- Generative semantics (now largely out of date)
- Relational grammar (RG) (now largely out of date)
- Arc pair grammar
- Generalized phrase structure grammar (GPSG; now largely out of date)
- Head-driven phrase structure grammar (HPSG)
- Lexical functional grammar (LFG)
- Nanosyntax

Categorial grammar

Categorial grammar is an approach that attributes the syntactic structure not to rules of grammar, but to the properties of the syntactic categories themselves. For example, rather than asserting that sentences are constructed by a rule that combines a noun phrase (NP) and a verb phrase (VP) (e.g. the phrase structure rule $S \rightarrow NP VP$), in categorial grammar, such principles are embedded in the category of the head word itself. So the syntactic category for an intransitive verb is a complex formula representing the fact that the verb acts as a function word requiring an NP as an input and produces a sentence level structure as an output. This complex category is notated as (NPS)

instead of V. NP\|S is read as "a category that searches to the left (indicated by \) for a NP (the element on the left) and outputs a sentence (the element on the right)". The category of transitive verb is defined as an element that requires two NPs (its subject and its direct object) to form a sentence. This is notated as (NP/(NP\|S)) which means "a category that searches to the right (indicated by /) for an NP (the object), and generates a function (equivalent to the VP) which is (NP\|S), which in turn represents a function that searches to the left for an NP and produces a sentence".

Tree-adjoining grammar is a categorial grammar that adds in partial tree structures to the categories.

Dependency grammar

Dependency grammar is an approach to sentence structure where syntactic units are arranged according to the dependency relation, as opposed to the constituency relation of phrase structure grammars. Dependencies are directed links between words. The (finite) verb is seen as the root of all clause structure and all the other words in the clause are either directly or indirectly dependent on this root. Some prominent dependency-based theories of syntax:

- Algebraic syntax
- Word grammar
- Operator grammar
- Meaning–text theory
- Functional generative description

Lucien Tesnière (1893–1954) is widely seen as the father of modern dependency-based theories of syntax and grammar. He argued vehemently against the binary division of the clause into subject and predicate that is associated with the grammars of his day ($S \rightarrow NP\ VP$) and which remains at the core of all phrase structure grammars, and in the place of this division, he positioned the verb as the root of all clause structure.^[10]

Stochastic/probabilistic grammars/network theories

Theoretical approaches to syntax that are based upon probability theory are known as stochastic grammars. One common implementation of such an approach makes use of a neural network or connectionism. Some theories based within this approach are:

- Optimality theory
- Stochastic context-free grammar

Functional grammars

Functional grammars, although focused upon form, are driven by explanation based upon the function of a sentence (i.e. its communicative function). Some typical functionalist theories include:

- Functional discourse grammar (Dik)
- Prague linguistic circle
- Systemic functional grammar
- Cognitive grammar
- Construction grammar (CxG)
- Role and reference grammar (RRG)
- Emergent grammar

Notes

- [1] Chomsky, Noam (2002) [1957]. *Syntactic Structures*. p. 11 (<http://books.google.co.uk/books?id=SNeHkMXHcd8C&pg=PA11&dq=%22syntax+is+the+study+of+the+principles+and+processes+by+which+sentences+are+constructed+in+particular+languages%22>).
- [2] Fortson IV, Benjamin W. (2004). *Indo-European Language and Culture: An Introduction*. Blackwell. p. 186. ISBN 1-4051-0315-9 (hb); 1-4051-0316-7 (pb). "[The *Aṣṭādhyāyī*] is a highly precise and thorough description of the structure of Sanskrit somewhat resembling modern generative grammar...[it] remained the most advanced linguistic analysis of any kind until the twentieth century."
- [3] Arnauld, Antoine (1683). *La logique* (<http://visualiseur.bnf.fr/Visualiseur?Destination=Gallica&O=NUMM-57444>) (5th ed.). Paris: G. Desprez. pp. 137. . "Nous avons emprunté...ce que nous avons dit...d'un petit Livre...sous le titre de *Grammaire générale*."
- [4] Giorgio, Graffi (2001). "200 Years of Syntax: A Critical Survey" (http://books.google.com/books/about/200_Years_of_Syntax.html?id=mydolrE-PPkC) (googlebook preview). John Benjamins Publishing. .
- [5] See Bickerton, Derek (1990). *Language and Species*. University of Chicago Press. ISBN 0-226-04610-9. and, for more recent advances, Derek Bickerton; Eörs Szathmáry, ed. (2009). *Biological foundations and origin of syntax*. MIT Press. ISBN 978-0-262-01356-7.
- [6] Ted Briscoe, 2 May 2001, Interview with Gerald Gazdar (<http://www.informatics.susx.ac.uk/research/nlp/gazdar/briscoe/gpsg.html#SECTION00040000000000000000>). Retrieved 2008-06-04.
- [7] Chomsky, Noam. 1957. *Syntactic Structures*. The Hague/Paris: Mouton, p. 15.
- [8] Chomsky, Noam (1981/1993). Lectures on Government and Binding: The Pisa Lectures. Mouton de Gruyter.
- [9] Chomsky, Noam (1995). The Minimalist Program. MIT Press.
- [10] Concerning Tesnière's rejection of the binary division of the clause into subject and predicate and in favor of the verb as the root of all structure, see Tesnière (1969:103–105).

References

- Brown, Keith; Jim Miller (eds.) (1996). *Concise Encyclopedia of Syntactic Theories*. New York: Elsevier Science. ISBN 0-08-042711-1.
- Carnie, Andrew (2006). *Syntax: A Generative Introduction* (2nd ed.). Oxford: Wiley-Blackwell. ISBN 1-4051-3384-8.
- Freidin, Robert; Howard Lasnik (eds.) (2006). *Syntax*. Critical Concepts in Linguistics. New York: Routledge. ISBN 0-415-24672-5.
- Graffi, Giorgio (2001). *200 Years of Syntax. A Critical Survey*. Studies in the History of the Language Sciences 98. Amsterdam: Benjamins. ISBN 90-272-4587-8.
- Mieszko Talasiewicz (2009). *Philosophy of Syntax—Foundational Topics*. Springer. ISBN 978-90-481-3287-4. An interdisciplinary essay on the interplay between logic and linguistics on syntactic theories.
- Tesnière, Lucien 1969. Éléments de syntaxe structurale. 2nd edition. Paris: Klincksieck.

Further reading

- Martin Everaert, Henk Van Riemsdijk, Rob Goedemans and Bart Hollebrandse, ed. (2006). *The Blackwell companion to syntax*. Blackwell. ISBN 978-1-4051-1485-1. 5 Volumes; 77 case studies of syntactic phenomena.
- Brian Roark; Richard William Sproat (2007). *Computational approaches to morphology and syntax*. Oxford University Press. ISBN 978-0-19-927477-2. part II: Computational approaches to syntax.
- Isac, Daniela; Charles Reiss (2008). *I-language: An Introduction to Linguistics as Cognitive Science* (<http://linguistics.concordia.ca/i-language/>). Oxford University Press. ISBN 978-0-19-953420-3.
- Edith A. Moravcsik (2006). *An introduction to syntax: fundamentals of syntactic analysis*. Continuum International Publishing Group. ISBN 978-0-8264-8945-6. Attempts to be a theory-neutral introduction. The companion Edith A. Moravcsik (2006). *An introduction to syntactic theory*. Continuum International Publishing Group. ISBN 978-0-8264-8943-2. surveys the major theories. Jointly reviewed in *The Canadian Journal of Linguistics* 54(1), March 2009, pp. 172–175 (http://muse.jhu.edu/login?url=/journals/canadian_journal_of_linguistics/v054/54.1.hewson.html)

External links

- The syntax of natural language: An online introduction using the Trees program (<http://www.ling.upenn.edu/~beatrice/syntax-textbook>)—Beatrice Santorini & Anthony Kroch, University of Pennsylvania, 2007

Phonology

Phonology is a branch of linguistics concerned with the systematic organization of sounds in languages. It has traditionally focused largely on study of the systems of phonemes in particular languages, but it may also cover any linguistic analysis either at a level beneath the word (including syllable, onset and rhyme, articulatory gestures, articulatory features, mora, etc.) or at all levels of language where sound is considered to be structured for conveying linguistic meaning. Phonology also includes the study of equivalent organizational systems in sign languages.

The word *phonology* (as in *the phonology of English*) can also refer to the phonological system (sound system) of a given language. This is one of the fundamental systems which a language is considered to comprise, like its syntax and its vocabulary.

Phonology is often distinguished from *phonetics*. While phonetics concerns the physical production, acoustic transmission and perception of the sounds of speech,^{[1][2]} phonology describes the way sounds function within a given language or across languages to encode meaning. In other words, phonetics belongs to descriptive linguistics, and phonology to theoretical linguistics. Note that this distinction was not always made, particularly before the development of the modern concept of phoneme in the mid 20th century. Some subfields of modern phonology have a crossover with phonetics in descriptive disciplines such as psycholinguistics and speech perception, resulting in specific areas like articulatory phonology or laboratory phonology.

Derivation and definitions

The word *phonology* comes from Greek φωνή, *phōnē*, "voice, sound", and the suffix *-logy* (which is from Greek λόγος, *lógos*, "word, speech, subject of discussion").

Definitions of the term vary. Nikolai Trubetzkoy in *Grundzüge der Phonologie* (1939) defines phonology as "the study of sound pertaining to the system of language", as opposed to phonetics, which is "the study of sound pertaining to the act of speech" (the distinction between *language* and *speech* being basically Saussure's distinction between *langue* and *parole*).^[3] More recently, Lass (1998) writes that phonology refers broadly to the subdiscipline of linguistics concerned with the sounds of language, while in more narrow terms, "phonology proper is concerned with the function, behaviour and organization of sounds as linguistic items".^[1] According to Clark *et al.* (2007) it means the systematic use of sound to encode meaning in any spoken human language, or the field of linguistics studying this use.^[4]

Development of phonology

The history of phonology may be traced back to the *Ashtadhyayi*, the Sanskrit grammar composed by Pāṇini in the 4th century BC. In particular the *Shiva Sutras*, an auxiliary text to the *Ashtadhyayi*, introduces what can be considered a list of the phonemes of the Sanskrit language, with a notational system for them that is used throughout the main text, which deals with matters of morphology, syntax and semantics.

The Polish scholar Jan Baudouin de Courtenay (together with his former student Mikołaj Kruszewski) introduced the concept of the *phoneme* in 1876, and his work, though often unacknowledged, is considered to be the starting point of modern phonology. He also worked on the theory of phonetic alternations (what is now called allophony and morphophonology), and had a significant influence on the work of Ferdinand de Saussure.

An influential school of phonology in the interwar period was the Prague School. One of its leading members was Prince Nikolai Trubetzkoy, whose *Grundzüge der Phonologie* (Principles of Phonology),^[3] published posthumously in 1939, is among the most important works in the field from this period. Directly influenced by Baudouin de Courtenay, Trubetzkoy is considered the founder of morphophonology, although this concept had also been recognized by de Courtenay. Trubetzkoy also developed the concept of the *archiphoneme*. Another important figure in the Prague School was Roman Jakobson, who was one of the most prominent linguists of the 20th century.

In 1968 Noam Chomsky and Morris Halle published *The Sound Pattern of English* (SPE), the basis for Generative Phonology. In this view, phonological representations are sequences of segments made up of distinctive features. These features were an expansion of earlier work by Roman Jakobson, Gunnar Fant, and Morris Halle. The features describe aspects of articulation and perception, are from a universally fixed set, and have the binary values + or -. There are at least two levels of representation: underlying representation and surface phonetic representation. Ordered phonological rules govern how underlying representation is transformed into the actual pronunciation (the so-called surface form). An important consequence of the influence SPE had on phonological theory was the downplaying of the syllable and the emphasis on segments. Furthermore, the Generativists folded morphophonology into phonology, which both solved and created problems.

Natural Phonology was a theory based on the publications of its proponent David Stampe in 1969 and (more explicitly) in 1979. In this view, phonology is based on a set of universal phonological processes which interact with one another; which ones are active and which are suppressed are language-specific. Rather than acting on segments, phonological processes act on distinctive features within prosodic groups. Prosodic groups can be as small as a part of a syllable or as large as an entire utterance. Phonological processes are unordered with respect to each other and apply simultaneously (though the output of one process may be the input to another). The second-most prominent Natural Phonologist is Stampe's wife, Patricia Donegan; there are many Natural Phonologists in Europe, though also a few others in the U.S., such as Geoffrey Nathan. The principles of Natural Phonology were extended to morphology by Wolfgang U. Dressler, who founded Natural Morphology.

In 1976 John Goldsmith introduced autosegmental phonology. Phonological phenomena are no longer seen as operating on *one* linear sequence of segments, called phonemes or feature combinations, but rather as involving *some parallel sequences* of features which reside on multiple tiers. Autosegmental phonology later evolved into Feature Geometry, which became the standard theory of representation for the theories of the organization of phonology as different as Lexical Phonology and Optimality Theory.

Government Phonology, which originated in the early 1980s as an attempt to unify theoretical notions of syntactic and phonological structures, is based on the notion that all languages necessarily follow a small set of principles and vary according to their selection of certain binary parameters. That is, all languages' phonological structures are essentially the same, but there is restricted variation that accounts for differences in surface realizations. Principles are held to be inviolable, though parameters may sometimes come into conflict. Prominent figures include Jonathan Kaye, Jean Lowenstamm, Jean-Roger Vergnaud, Monik Charette, John Harris, and many others.

In a course at the LSA summer institute in 1991, Alan Prince and Paul Smolensky developed Optimality Theory — an overall architecture for phonology according to which languages choose a pronunciation of a word that best satisfies a list of constraints which is ordered by importance: a lower-ranked constraint can be violated when the violation is necessary in order to obey a higher-ranked constraint. The approach was soon extended to morphology by John McCarthy and Alan Prince, and has become a dominant trend in phonology. Though this usually goes



Nikolai Trubetzkoy, 1920s.

unacknowledged, Optimality Theory was strongly influenced by Natural Phonology; both view phonology in terms of constraints on speakers and their production, though these constraints are formalized in very different ways. The appeal to phonetic grounding of constraints in various approaches has been criticized by proponents of 'substance-free phonology'.^[5]

Broadly speaking Government Phonology (or its descendant, Strict-CV Phonology) has a greater following in the United Kingdom, whereas Optimality Theory is predominant in North America.

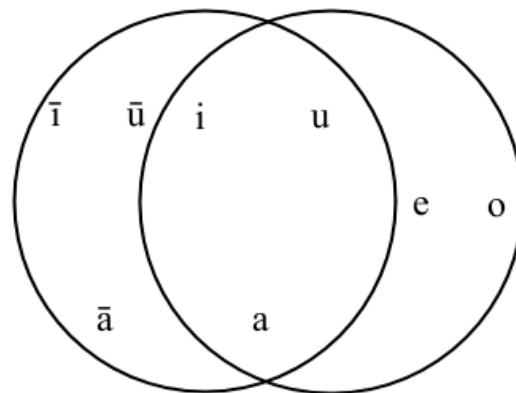
Analysis of phonemes

An important part of traditional, pre-generative, schools of phonology is studying which sounds can be grouped into distinctive units within a language; these units are known as phonemes. For example, in English, the "p" sound in *pot* is aspirated (pronounced [p^h]), while that in *spot* is not aspirated (pronounced [p]). However, English speakers intuitively treat both sounds as variations (allophones) of the same phonological category, that is, of the phoneme /p/. (Traditionally, it would be argued that if a word-initial aspirated [p^h] were interchanged with the unaspirated [p] in *spot*, native speakers of English would still hear the same words; that is, the two sounds are perceived as "the same" /p/.) In some other languages, however, these two sounds are perceived as different, and they are consequently assigned to different phonemes in those languages. For example, in Thai, Hindi, and Quechua, there are minimal pairs of words for which aspiration is the only contrasting feature (two words with different meanings that are identical except that one has an aspirated sound where the other has an unaspirated one).

Part of the phonological study of a language therefore involves looking at data (phonetic transcriptions of the speech of native speakers) and trying to deduce what the underlying phonemes are and what the sound inventory of the language is. The presence or absence of minimal pairs, as mentioned above, is a frequently used criterion for deciding whether two sounds should be assigned to the same phoneme. However other considerations often need to be taken into account as well.

The particular sounds which are phonemic in a language can change over time. At one time, [f] and [v] were allophones in English, but these later changed into separate phonemes. This is one of the main factors of historical change of languages as described in historical linguistics.

The findings and insights of speech perception and articulation research complicates the traditional and somewhat intuitive idea of interchangeable allophones being perceived as the same phoneme. First, interchanged allophones of the same phoneme can result in unrecognizable words. Second, actual speech, even at a word level, is highly co-articulated, so it is problematic to expect to be able to splice words into simple segments without affecting speech perception.



The vowels of modern (Standard) Arabic and (Israeli) Hebrew from the phonemic point of view. Note the intersection of the two circles—the distinction between short *a*, *i* and *u* is made by both speakers, but Arabic lacks the mid articulation of short vowels, while Hebrew lacks the distinction of vowel length.

Different linguists therefore take different approaches to the problem of assigning sounds to phonemes. For example, they differ in the extent to which they require allophones to be phonetically similar. There are also differing ideas as to whether this grouping of sounds is purely a tool for linguistic analysis, or reflects an actual process in the way the human brain processes a language.

Since the early 1960s, theoretical linguists have moved away from the traditional concept of a phoneme, preferring to consider basic units at a more abstract level, as a component of morphemes; these units can be called *morphophonemes*, and analysis using this approach is called morphophonology.

Other topics in phonology

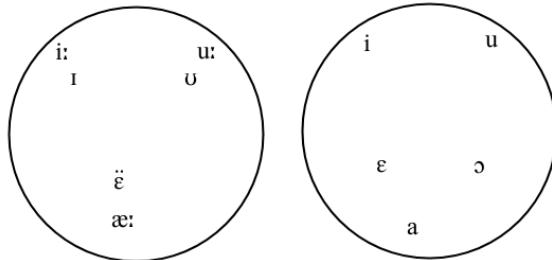
In addition to the minimal units that can serve the purpose of differentiating meaning (the phonemes), phonology studies how sounds alternate, i.e. replace one another in different forms of the same morpheme (allomorphs), as well as, for example, syllable structure, stress, accent, and intonation.

Phonology also includes topics such as phonotactics (the phonological constraints on what sounds can appear in what positions in a given language) and phonological alternation (how the pronunciation of a sound changes through the application of phonological rules, sometimes in a given order which can be feeding or bleeding,^[6]) as well as prosody, the study of suprasegmentals and topics such as stress and intonation.

The principles of phonological analysis can be applied independently of modality because they are designed to serve as general analytical tools, not language-specific ones. The same principles have been applied to the analysis of sign languages (see Phonemes in sign languages), even though the sub-lexical units are not instantiated as speech sounds.

Notes

- [1] Lass, Roger (1998. Digitized 2000). *Phonology: An Introduction to Basic Concepts* (<http://books.google.com/books?id=aTsAt3D6H58C&printsec=frontcover&dq=phonology#v=onepage&q&f=false>). Cambridge, UK; New York; Melbourne, Australia: Cambridge University Press. p. 1. ISBN 0-521-23728-9. . Retrieved 8 January 2011 Paperback ISBN 0-521-28183-0
 - [2] Carr, Philip (2003). *English Phonetics and Phonology: An Introduction* (<http://books.google.com/books?id=p5a7mmppqbt0C&printsec=frontcover&dq=introduction+phonetics+phonology#v=onepage&q&f=false>). Massachusetts, USA; Oxford, UK; Victoria, Australia; Berlin, Germany: Blackwell Publishing. ISBN 0-631-19775-3. . Retrieved 8 January 2011 Paperback ISBN 0-631-19776-1
 - [3] Trubetzkoy N., *Grundzüge der Phonologie* (published 1939), translated by C. Baltaxe as *Principles of Phonology* (http://books.google.pl/books?id=Ej6EndUGS-UC&dq=%E2%80%9EGrundz%C3%BCge+der+Phonologie%E2%80%9C&hl=pl&source=gbs_navlinks_s), University of California Press, 1969
 - [4] Clark, John; Yallop, Colin; Fletcher, Janet (2007). *An Introduction to Phonetics and Phonology* (<http://books.google.com/books?id=dX5P5mxtYYIC&printsec=frontcover&dq=introduction+phonetics+phonology#v=onepage&q&f=false>) (3rd ed.). Massachusetts, USA; Oxford, UK; Victoria, Australia: Blackwell Publishing. ISBN 978-1-4051-3083-7. . Retrieved 8 January 2011 Alternative ISBN 1-4051-3083-0
 - [5] Hale, Mark; Reiss, Charles (2008). *The Phonological Enterprise*. Oxford, UK: Oxford University Press. ISBN 0-19-953397-0.
 - [6] Goldsmith 1995:1.



The vowels of Modern Standard Arabic and Israeli Hebrew from the phonetic point of view. Note that the two circles are totally separate—none of the vowel-sounds made by speakers of one language is made by speakers of the other. One modern theory is that Israeli Hebrew's phonology reflects Yiddish elements, not Semitic ones.

Bibliography

- Anderson, John M.; and Ewen, Colin J. (1987). *Principles of dependency phonology*. Cambridge: Cambridge University Press.
- Bloch, Bernard (1941). "Phonemic overlapping". *American Speech* **16** (4): 278–284. doi:10.2307/486567. JSTOR 486567.
- Bloomfield, Leonard. (1933). *Language*. New York: H. Holt and Company. (Revised version of Bloomfield's 1914 *An introduction to the study of language*).
- Brentari, Diane (1998). *A prosodic model of sign language phonology*. Cambridge, MA: MIT Press.
- Chomsky, Noam. (1964). Current issues in linguistic theory. In J. A. Fodor and J. J. Katz (Eds.), *The structure of language: Readings in the philosophy language* (pp. 91–112). Englewood Cliffs, NJ: Prentice-Hall.
- Chomsky, Noam; and Halle, Morris. (1968). *The sound pattern of English*. New York: Harper & Row.
- Clements, George N. (1985). "The geometry of phonological features". *Phonology Yearbook* **2**: 225–252. doi:10.1017/S0952675700000440.
- Clements, George N.; and Samuel J. Keyser. (1983). *CV phonology: A generative theory of the syllable*. Linguistic inquiry monographs (No. 9). Cambridge, MA: MIT Press. ISBN 0-262-53047-3 (pbk); ISBN 0-262-03098-5 (hbk).
- de Lacy, Paul, ed. (2007). *The Cambridge Handbook of Phonology* (<http://books.google.com/?id=7sxLaeZAhOAC&printsec=frontcover&q=Cambridge+Handbook+of+Phonology#v=onepage&q&f=false>). Cambridge University Press. ISBN 0-521-84879-2 (hbk). Retrieved 8 January 2011
- Donegan, Patricia. (1985). On the Natural Phonology of Vowels. New York: Garland. ISBN 0-8240-5424-5.
- Firth, J. R. (1948). "Sounds and prosodies". *Transactions of the Philological Society* **47** (1): 127–152. doi:10.1111/j.1467-968X.1948.tb00556.x.
- Gilbers, Dicky; de Hoop, Helen (1998). "Conflicting constraints: An introduction to optimality theory". *Lingua* **104**: 1–12. doi:10.1016/S0024-3841(97)00021-1.
- Goldsmith, John A. (1979). The aims of autosegmental phonology. In D. A. Dinnsen (Ed.), *Current approaches to phonological theory* (pp. 202–222). Bloomington: Indiana University Press.
- Goldsmith, John A. (1989). *Autosegmental and metrical phonology: A new synthesis*. Oxford: Basil Blackwell.
- Goldsmith, John A (1995). "Phonological Theory". In John A. Goldsmith. *The Handbook of Phonological Theory*. Blackwell Handbooks in Linguistics. Blackwell Publishers. ISBN 1-4051-5768-2.
- Gussenhoven, Carlos & Jacobs, Haire. "Understanding Phonology", Hodder & Arnold, 1998. 2nd edition 2005.
- Hale, Mark; Reiss, Charles (2008). *The Phonological Enterprise*. Oxford, UK: Oxford University Press. ISBN 0-19-953397-0.
- Halle, Morris (1954). "The strategy of phonemics". *Word* **10**: 197–209.
- Halle, Morris. (1959). *The sound pattern of Russian*. The Hague: Mouton.
- Harris, Zellig. (1951). *Methods in structural linguistics*. Chicago: Chicago University Press.
- Hockett, Charles F. (1955). *A manual of phonology*. Indiana University publications in anthropology and linguistics, memoirs II. Baltimore: Waverley Press.
- Hooper, Joan B. (1976). *An introduction to natural generative phonology*. New York: Academic Press.
- Jakobson, Roman (1949). "On the identification of phonemic entities". *Travaux du Cercle Linguistique de Copenhague* **5**: 205–213.
- Jakobson, Roman; Fant, Gunnar; and Halle, Morris. (1952). *Preliminaries to speech analysis: The distinctive features and their correlates*. Cambridge, MA: MIT Press.
- Kaisse, Ellen M.; and Shaw, Patricia A. (1985). On the theory of lexical phonology. In E. Colin and J. Anderson (Eds.), *Phonology Yearbook* **2** (pp. 1–30).
- Kenstowicz, Michael. *Phonology in generative grammar*. Oxford: Basil Blackwell.
- Ladefoged, Peter. (1982). *A course in phonetics* (2nd ed.). London: Harcourt Brace Jovanovich.
- Martinet, André. (1949). *Phonology as functional phonetics*. Oxford: Blackwell.

- Martinet, André. (1955). *Économie des changements phonétiques: Traité de phonologie diachronique*. Berne: A. Francke S.A.
- Napoli, Donna Jo (1996). *Linguistics: An Introduction*. New York: Oxford University Press.
- Pike, Kenneth. (1947). *Phonemics: A technique for reducing languages to writing*. Ann Arbor: University of Michigan Press.
- Sandler, Wendy and Lillo-Martin, Diane. 2006. *Sign language and linguistic universals*. Cambridge: Cambridge University Press
- Sapir, Edward (1925). "Sound patterns in language". *Language* **1** (2): 37–51. doi:10.2307/409004. JSTOR 409004.
- Sapir, Edward (1933). "La réalité psychologique des phonèmes". *Journal de Psychologie Normale et Pathologique* **30**: 247–265.
- de Saussure, Ferdinand. (1916). *Cours de linguistique générale*. Paris: Payot.
- Stampe, David. (1979). *A dissertation on natural phonology*. New York: Garland.
- Swadesh, Morris (1934). "The phonemic principle". *Language* **10** (2): 117–129. doi:10.2307/409603. JSTOR 409603.
- Trager, George L.; Bloch, Bernard (1941). "The syllabic phonemes of English". *Language* **17** (3): 223–246. doi:10.2307/409203. JSTOR 409203.
- Trubetzkoy, Nikolai. (1939). *Grundzüge der Phonologie*. Travaux du Cercle Linguistique de Prague 7.
- Twaddell, William F. (1935). On defining the phoneme. Language monograph no. 16. *Language*.

External links

- Phonetics and phonology (http://www.dmoz.org/Science/Social_Sciences/Linguistics/Phonetics_and_Phonology/) at the Open Directory Project

Morphophonology

Morphophonology (also **morphophonemics**, **morphonology**) is a branch of linguistics which studies the interaction between morphological and phonological or phonetic processes. Its chief focus is the sound changes that take place in morphemes (minimal meaningful units) when they combine to form words.

Morphophonological analysis often involves an attempt to give a series of formal rules that successfully predict the regular sound changes occurring in the morphemes of a given language. Such a series of rules converts a theoretical underlying representation into a surface form that is actually heard. The units of which the underlying representations of morphemes are composed are sometimes called **morphophonemes**. The surface form produced by the morphophonological rules may consist of phonemes (which are then subject to ordinary phonological rules to produce speech sounds or *phones*), or else the morphophonological analysis may bypass the phoneme stage and produce the phones itself.

Morphophonemes and morphophonological rules

When morphemes combine, they influence each other's sound structure (whether analyzed at a phonetic or phonemic level), resulting in different variant pronunciations for the same morpheme. Morphophonology attempts to analyze these processes. A language's morphophonological structure is generally described with a series of rules which, ideally, can predict every morphophonological alternation that takes place in the language.

An example of a morphophonological alternation in English is provided by the plural morpheme, written as "-s" or "-es". Its pronunciation alternates between [s], [z], and [ɪz], as in *cats*, *dogs*, and *horses* respectively. A purely phonological analysis would most likely assign to these three endings the phonemic representations /s/, /z/, /ɪz/. On a morphophonological level, however, they may all be considered to be forms of the underlying object //z//, which is a **morphophoneme**. The different forms it takes are dependent on the segment at the end of the morpheme to which it attaches – these dependencies are described by morphophonological rules. (The behaviour of the English past tense ending "-ed" is similar – it can be pronounced [t], [d] or [ɪd], as in *hoped*, *bobbed* and *added*.)

Note that the plural suffix "-s" can also influence the form taken by the preceding morpheme, as in the case of the words *leaf* and *knife*, which end with [f] in the singular, but have [v] in the plural (*leaves*, *knives*). On a morphophonological level these morphemes may be analyzed as ending in a morphophoneme //F//, which becomes voiced when a voiced consonant (in this case the //z// of the plural ending) is attached to it. This rule may be written symbolically as: /F/ -> [əvoice] / __ [əvoice].

In the International Phonetic Alphabet, pipes (| |) are often used to indicate a morphophonemic rather than phonemic representation. Another common convention is double slashes (// //), as above, implying that the transcription is 'more phonemic than simply phonemic'. Other conventions sometimes seen are double pipes (|| ||) and curly brackets ({ }).

Types of morphophonological changes

Inflected and agglutinating languages may have extremely complicated systems of morphophonemics. Examples of complex morphophonological systems include:

- Sandhi, the phenomenon behind the English examples of plural and past tense above, is found in virtually all languages to some degree. Even Mandarin, which is sometimes said to display no morphology, nonetheless displays tone sandhi, a morphophonemic alternation.
- Consonant gradation, found in some Uralic languages such as Finnish, Estonian, Northern Sámi, and Nganasan.
- Vowel harmony, which occurs in varying degrees in languages all around the world, notably Turkic languages.
- Ablaut, found in English and other Germanic languages. Ablaut is the phenomenon wherein stem vowels change form depending on context, as in English *sing*, *sang*, *sung*.

Relation between phonology and morphophonology

Until the 1950s, many phonologists assumed that neutralizing rules generally applied before allophonic rules. Thus phonological analysis was split into two parts: a morphophonological part, where neutralizing rules were developed to derive phonemes from morphophonemes; and a purely phonological part, where phones were derived from the phonemes. Since the 1960s (in particular with the work of the generative school, such as Chomsky and Halle's *The Sound Pattern of English*) many linguists have moved away from making such a split, instead regarding the surface phones as being derived from the underlying morphophonemes (which may be referred to using various terminology) through a single system of (morpho)phonological rules.

The purpose of both phonemic and morphophonemic analysis is to produce simpler underlying descriptions for what appear on the surface to be complicated patterns. In purely phonemic analysis the data is just a set of words in a language, while for the purposes of morphophonemic analysis the words must be considered in grammatical paradigms to take account of the underlying morphemes. It is postulated that morphemes are recorded in the speaker's "lexicon" in an invariant (morphophonemic) form, which, in a given environment, is converted by rules into a surface form. The analyst attempts to present as completely as possible a system of underlying units (morphophonemes) and a series of rules that act on them, so as to produce surface forms consistent with the linguistic data.

Isolation forms

The **isolation form** of a morpheme is the form in which that morpheme appears in isolation (when not subject to the effects of any other morpheme). In the case of a bound morpheme, such as the English past tense ending "-ed", it will generally not be possible to identify an isolation form, since such a morpheme does not occur in isolation.

It is often reasonable to assume that the isolation form of a morpheme provides its underlying representation. For example, in some American English, *plant* is pronounced [plænt], while *planting* is ['plænɪŋ], where the morpheme "plant-" appears in the form [plæn]. Here the underlying form can be assumed to be //plænt//, corresponding to the isolation form, since rules can be set up to derive the reduced form [plæn] from this (while it would be difficult or impossible to set up rules that would derive the isolation form [plænt] from an underlying //plæn//).

This is not always the case, however; sometimes the isolation form itself is subject to neutralization that does not apply to some other instances of the morpheme. For example, the French word *petit* ("small") is pronounced in isolation without the final [t] sound, although in certain derived forms (such as the feminine *petite*) the [t] is heard. If the isolation form were adopted as the underlying form, the information that there is a final "t" would be lost, and it would be hard to explain the appearance of the "t" in the inflected forms.

Rule ordering

Morphophonological rules are generally considered to apply in a set order. This means that the application of one rule may sometimes either prevent or enable the application of another rule, provided the rules are appropriately ordered.

If the ordering of two rules is such that the application of the first rule can have the effect of making it possible to apply the second, then the rules are said to be in *feeding order*. For example, if a language has an apocope rule (A) which deletes a final vowel, and a cluster reduction rule (CR) that reduces a final consonant cluster, then the rules are in feeding order if A precedes CR, since the application of A can enable application of CR (for example, a word ending /-rpa/ is not itself subject to CR, since the consonant cluster is not final, but if A is applied to it first, leaving /-rp/, then CR can apply). Here rule A is said to *feed* rule CR. If the rules are ordered such as to avoid possible feeding (in this case, if CR applies before A) then they are said to be in *counter-feeding order*.

On the other hand, if rules are ordered such that the application of the first rule can have the effect of *preventing* application of the second, then the rules are said to be in *bleeding order*. For example, if a language has an

epenthesis rule (E) that inserts a /w/ before certain vowels, and a vowel deletion rule (D) that deletes one of two consecutive vowels, then the rules are in bleeding order if E precedes D, since the application of E can prevent application of D (for example, a word containing /-iu-/ would be subject to D, but if E is applied to it first, leaving /-iwu-, then D can no longer apply). Here rule E is said to *bleed* rule D. If the rules are ordered such as to avoid possible bleeding (in this case, if D applies before E) then they are said to be in *counter-bleeding order*.

The terminology of feeding and bleeding is also applied to other linguistic rules, such as those of historical sound changes.

Morphophonology and orthography

The principle behind alphabetic writing systems is that the letters (graphemes) represent phonemes. However in many orthographies based on such systems the correspondences between graphemes and phonemes are not exact, and it is sometimes the case that certain spellings better represent a word's morphophonological structure rather than the purely phonological. An example of this is that the English plural morpheme is written -s regardless of whether it is pronounced as /s/ or /z/; we write *cats* and *dogs*, not *dogz*.

The above example involves active morphology (inflection), and morphophonemic spellings are common in this context in many languages. Another type of spelling that can be described as morphophonemic is the kind that reflects the etymology of words. Such spellings are particularly common in English; examples include *science* /saɪ/ vs. *unconscious* /ʃ/, *prejudice* /prɛdɪs/ vs. *prequel* /pri:əl/, *sign* /saɪn/ *signature* /saɪgn/, *nation* /neɪtʃn/ vs. *nationalism* /nætlɪzəm/, and *special* /spɛl/ vs. *species* /spi:əl/.

For more detail on this topic, see Phonemic orthography, in particular the section on Morphophonemic features.

References

- Hayes, Bruce (2009). "Morphophonemic Analysis" *Introductory Phonology*, pp. 161–185. Blackwell

Phonetics

Phonetics (pronounced /fə'netɪks/, from the Greek: φωνή, *phōnē*, 'sound, voice') is a branch of linguistics that comprises the study of the sounds of human speech, or—in the case of sign languages—the equivalent aspects of sign.^[1] It is concerned with the physical properties of speech sounds or signs (phones): their physiological production, acoustic properties, auditory perception, and neurophysiological status. Phonology, on the other hand, is concerned with the abstract, grammatical characterization of systems of sounds or signs.

The field of phonetics is a multiple layered subject of linguistics that focuses on speech. In the case of oral languages there are three basic areas of study:

- Articulatory phonetics: the study of the production of speech sounds by the articulatory and vocal tract by the speaker
- Acoustic phonetics: the study of the physical transmission of speech sounds from the speaker to the listener
- Auditory phonetics: the study of the reception and perception of speech sounds by the listener

These areas are inter-connected through the common mechanism of sound, such as wavelength (pitch), amplitude, and harmonics.

History

Phonetics was studied as early as 500 BC in ancient India, with Pāṇini's account of the place and manner of articulation of consonants in his 5th century BC treatise on Sanskrit. The major Indic alphabets today order their consonants according to Pāṇini's classification. The Ancient Greeks are credited as the first to base a writing system on a phonetic alphabet.

Modern phonetics begins with attempts — such as those of Joshua Steele (in *Prosodia Rationalis*, 1779) and Alexander Melville Bell (in *Visible Speech*, 1867) — to introduce systems of precise notation for speech sounds.^{[2][3]}

Phonetic transcription

The International Phonetic Alphabet (IPA) is used as the basis for the phonetic transcription of speech. It is based on the Latin alphabet and is able to transcribe most features of speech such as consonants, vowels, and suprasegmental features. Every documented phoneme available within the known languages in the world is assigned its own corresponding symbol.

The difference between phonetics and phonology

Phonology concerns itself with systems of phonemes, abstract *cognitive* units of speech sound or sign which distinguish the words of a language. Phonetics, on the other hand, concerns itself with the production, transmission, and perception of the *physical* phenomena which are abstracted in the mind to constitute these speech sounds or signs.

Using an Edison phonograph, Ludimar Hermann investigated the spectral properties of vowels and consonants. It was in these papers that the term *formant* was first introduced. Hermann also played back vowel recordings made with the Edison phonograph at different speeds in order to test Willis' and Wheatstone's theories of vowel production.

Relation to phonology

In contrast to phonetics, phonology is the study of how sounds and gestures pattern in and across languages, relating such concerns with other levels and aspects of language. Phonetics deals with the articulatory and acoustic properties of speech sounds, how they are produced, and how they are perceived. As part of this investigation, phoneticians may concern themselves with the physical properties of meaningful sound contrasts or the social meaning encoded in the speech signal (socio-phonetics) (e.g. gender, sexuality, ethnicity, etc.). However, a substantial portion of research in phonetics is not concerned with the meaningful elements in the speech signal.

While it is widely agreed that phonology is grounded in phonetics, phonology is a distinct branch of linguistics, concerned with sounds and gestures as abstract units (e.g., distinctive features, phonemes, mora, syllables, etc.) and their conditioned variation (via, e.g., allophonic rules, constraints, or derivational rules).^[4] Phonology relates to phonetics via the set of distinctive features, which map the abstract representations of speech units to articulatory gestures, acoustic signals, and/or perceptual representations.^{[5][6][7]}

Subfields

Phonetics as a research discipline has three main branches:

- articulatory phonetics is concerned with the articulation of speech: The position, shape, and movement of *articulators* or speech organs, such as the lips, tongue, and vocal folds.
- acoustic phonetics is concerned with acoustics of speech: The spectro-temporal properties of the sound waves produced by speech, such as their frequency, amplitude, and harmonic structure.
- auditory phonetics is concerned with speech perception: the perception, categorization, and recognition of speech sounds and the role of the auditory system and the brain in the same.

Transcription

Phonetic transcription is a system for transcribing sounds that occur in a language, whether oral or sign. The most widely known system of phonetic transcription, the International Phonetic Alphabet (IPA), provides a standardized set of symbols for oral phones.^{[8][9]} The standardized nature of the IPA enables its users to transcribe accurately and consistently the phones of different languages, dialects, and idiolects.^{[8][10][11]} The IPA is a useful tool not only for the study of phonetics, but also for language teaching, professional acting, and speech pathology.^[10]

Applications

Application of phonetics include:

- forensic phonetics: the use of phonetics (the science of speech) for forensic (legal) purposes.
- Speech Recognition: the analysis and transcription of recorded speech by a computer system.

Notes

[1] O'Grady (2005) p.15

[2] T.V.F. Brogan: *English Versification, 1570–1980* (<http://www.arsversificandi.net/resources/evrg/index.html>). Baltimore: Johns Hopkins University Press, 1981. E394.

[3] Alexander Melville Bell 1819-1905 (http://www.acsu.buffalo.edu/~duchan/new_history/hist19c/subpages/mbell.html). University at Buffalo, The State University of New York.

[4] Kingston, John. 2007. *The Phonetics-Phonology Interface*, in The Cambridge Handbook of Phonology (ed. Paul DeLacy), Cambridge University Press.

[5] Halle, Morris. 1983. *On Distinctive Features and their articulatory implementation*, Natural Language and Linguistic Theory, p. 91 - 105

[6] Jakobson, Roman, Gunnar Fant, and Morris Halle. 1976. Preliminaries to Speech Analysis: The Distinctive Features and their Correlates, MIT Press.

[7] Hall, T. Allen. 2001. Phonological representations and phonetic implementation of distinctive features, Mouton de Gruyter.

- [8] O'Grady (2005) p.17
- [9] International Phonetic Association (1999) *Handbook of the International Phonetic Association*. Cambridge University Press.
- [10] Ladefoged, Peter (1975) *A Course in Phonetics*. Orlando: Harcourt Brace. 5th ed. Boston: Thomson/Wadsworth 2006.
- [11] Ladefoged, Peter & Ian Maddieson (1996) *The Sounds of the World's Languages*. Oxford: Blackwell.

References

- O'Grady, William, et al. (2005). *Contemporary Linguistics: An Introduction* (5th ed.). Bedford/St. Martin's. ISBN 0-312-41936-8.

External links

- the Web Site of the Phonetic Sciences Laboratory of the Université de Montréal. (<http://www.phonetique.info>)
- The International Society of Phonetic Sciences (ISPhS) (<http://www.isphs.org/main.htm>)
- A little encyclopedia of phonetics (<http://www.cambridge.org/elt/peterroach/resources/Glossary.pdf>), Peter Roach, Professor of Phonetics, University of Reading, UK. (pdf)
- The sounds and sound patterns of language (http://www.ling.upenn.edu/courses/Summer_2004/ling001/lecture2.html) U Penn
- UCLA lab data (<http://hctv.humnet.ucla.edu/departments/linguistics/VowelsandConsonants/>)
- UCLA Phonetics Lab Archive (<http://archive.phonetics.ucla.edu/>)
- EGG and Voice Quality (<http://www.ims.uni-stuttgart.de/phonetik/EGG/page1.htm>) (electroglossography, phonation, etc.)
- IPA handbook (<http://web.uvic.ca/ling/resources/ipa/handbook.htm>)
- IPA-SAM Phonetic Fonts (<http://www.phon.ucl.ac.uk/home/wells/fonts.htm>)
- Speech Analysis Tutorial (<http://www.ling.lu.se/research/speechtutorial/tutorial.html>)
- Lecture materials in German on phonetics & phonology, university of Erfurt (http://www.uni-erfurt.de/sprachwissenschaft/personal/lehmann/CL_Lehr/PhonPhon/Phon_Index.html)
- Real-time MRI video of the articulation of speech sounds, from the USC Speech Articulation and kNowledge (SPAN) Group (<http://sail.usc.edu/span/video.php>)
- Beginner's course in phonetics, with some exercises (<http://www.ic.arizona.edu/~lsp/Phonetics.html>)
- Praat - Phonetic analysis software (<http://www.fon.hum.uva.nl/praat/>)
- SID- Speech Internet Dictionary (<http://www.phon.ucl.ac.uk/home/johnm/sid/sidhome.htm>)
- Extensive collection of phonetics resources on the Web (<http://www.unc.edu/~jlsmith/pht-url.html>) (University of North Carolina)
- Phonetics and Phonology (<http://www.elloandfriends.uni-osnabrueck.de/wikis/1/show?n=PhoneticsandPhonology.PhoneticsandPhonology>) (University of Osnabrueck)

Semantics

Semantics (from Greek: *sēmantikós*)^{[1][2]} is the study of meaning. It focuses on the relation between *signifiers*, such as words, phrases, signs, and symbols, and what they stand for, their denotata.

Linguistic semantics is the study of meaning that is used to understand human expression through language. Other forms of semantics include the semantics of programming languages, formal logics, and semiotics.

The word *semantics* itself denotes a range of ideas, from the popular to the highly technical. It is often used in ordinary language to denote a problem of understanding that comes down to word selection or connotation. This problem of understanding has been the subject of many formal enquiries, over a long period of time, most notably in the field of formal semantics. In linguistics, it is the study of interpretation of signs or symbols as used by agents or communities within particular circumstances and contexts.^[3] Within this view, sounds, facial expressions, body language, and proxemics have semantic (meaningful) content, and each has several branches of study. In written language, such things as paragraph structure and punctuation have semantic content; in other forms of language, there is other semantic content.^[3]

The formal study of semantics intersects with many other fields of inquiry, including lexicology, syntax, pragmatics, etymology and others, although semantics is a well-defined field in its own right, often with synthetic properties.^[4] In philosophy of language, semantics and reference are closely connected. Further related fields include philology, communication, and semiotics. The formal study of semantics is therefore complex.

Semantics contrasts with syntax, the study of the combinatorics of units of a language (without reference to their meaning), and pragmatics, the study of the relationships between the symbols of a language, their meaning, and the users of the language.^[5]

In international scientific vocabulary semantics is also called *semasiology*.

Linguistics

In linguistics, **semantics** is the subfield that is devoted to the study of meaning, as inherent at the levels of words, phrases, sentences, and larger units of discourse (termed *texts*). The basic area of study is the meaning of signs, and the study of relations between different linguistic units and compounds: homonymy, synonymy, antonymy, hypernymy, hyponymy, meronymy, metonymy, holonymy, paronyms. A key concern is how meaning attaches to larger chunks of text, possibly as a result of the composition from smaller units of meaning. Traditionally, semantics has included the study of *sense* and denotative *reference*, truth conditions, argument structure, thematic roles, discourse analysis, and the linkage of all of these to syntax.

Montague grammar

In the late 1960s, Richard Montague proposed a system for defining semantic entries in the lexicon in terms of the lambda calculus. In these terms, the syntactic parse of the sentence *John ate every bagel* would consist of a subject (*John*) and a predicate (*ate every bagel*); Montague showed that the meaning of the sentence as a whole could be decomposed into the meanings of its parts and relatively few rules of combination. The logical predicate thus obtained would be elaborated further, e.g. using truth theory models, which ultimately relate meanings to a set of Tarskian universals, which may lie outside the logic. The notion of such meaning atoms or primitives is basic to the language of thought hypothesis from the 1970s.

Despite its elegance, Montague grammar was limited by the context-dependent variability in word sense, and led to several attempts at incorporating context, such as:

- Situation semantics (1980s): truth-values are incomplete, they get assigned based on context
- Generative lexicon (1990s): categories (types) are incomplete, and get assigned based on context

Dynamic turn in semantics

In Chomskyan linguistics there was no mechanism for the learning of semantic relations, and the nativist view considered all semantic notions as inborn. Thus, even novel concepts were proposed to have been dormant in some sense. This view was also thought unable to address many issues such as metaphor or associative meanings, and semantic change, where meanings within a linguistic community change over time, and qualia or subjective experience. Another issue not addressed by the nativist model was how perceptual cues are combined in thought, e.g. in mental rotation.^[6]

This view of semantics, as an innate finite meaning inherent in a lexical unit that can be composed to generate meanings for larger chunks of discourse, is now being fiercely debated in the emerging domain of cognitive linguistics^[7] and also in the non-Fodoran camp in philosophy of language.^[8] The challenge is motivated by:

- factors internal to language, such as the problem of resolving indexical or anaphora (e.g. *this x, him, last week*). In these situations *context* serves as the input, but the interpreted utterance also modifies the context, so it is also the output. Thus, the interpretation is necessarily dynamic and the meaning of sentences is viewed as context change potentials instead of propositions.
- factors external to language, i.e. language is not a set of labels stuck on things, but "a toolbox, the importance of whose elements lie in the way they function rather than their attachments to things."^[8] This view reflects the position of the later Wittgenstein and his famous *game* example, and is related to the positions of Quine, Davidson, and others.

A concrete example of the latter phenomenon is semantic underspecification – meanings are not complete without some elements of context. To take an example of one word, *red*, its meaning in a phrase such as *red book* is similar to many other usages, and can be viewed as compositional.^[9] However, the colours implied in phrases such as *red wine* (very dark), and *red hair* (coppery), or *red soil*, or *red skin* are very different. Indeed, these colours by themselves would not be called *red* by native speakers. These instances are contrastive, so *red wine* is so called only in comparison with the other kind of wine (which also is not *white* for the same reasons). This view goes back to de Saussure:

Each of a set of synonyms like *redouter* ('to dread'), *craindre* ('to fear'), *avoir peur* ('to be afraid') has its particular value only because they stand in contrast with one another. No word has a value that can be identified independently of what else is in its vicinity.^[10]

and may go back to earlier Indian views on language, especially the Nyaya view of words as indicators and not carriers of meaning.^[11]

An attempt to defend a system based on propositional meaning for semantic underspecification can be found in the generative lexicon model of James Pustejovsky, who extends contextual operations (based on type shifting) into the lexicon. Thus meanings are generated on the fly based on finite context.

Prototype theory

Another set of concepts related to fuzziness in semantics is based on prototypes. The work of Eleanor Rosch in the 1970s led to a view that natural categories are not characterizable in terms of necessary and sufficient conditions, but are graded (fuzzy at their boundaries) and inconsistent as to the status of their constituent members. One may compare it with Jung's archetype, though the concept of archetype sticks to static concept. Some post-structuralists are against the fixed or static meaning of the words. Derrida, following Nietzsche, talked about slippages in fixed meanings. Here are some examples from Bangla fuzzy words^{[12][13]}

Systems of categories are not objectively *out there* in the world but are rooted in people's experience. These categories evolve as learned concepts of the world – meaning is not an objective truth, but a subjective construct, learned from experience, and language arises out of the "grounding of our conceptual systems in shared embodiment and bodily experience".^[14] A corollary of this is that the conceptual categories (i.e. the lexicon) will not be identical

for different cultures, or indeed, for every individual in the same culture. This leads to another debate (see the Sapir–Whorf hypothesis or Eskimo words for snow).

Theories in semantics

Model theoretic semantics

Originates from Montague's work (see above). A highly formalized theory of natural language semantics in which expressions are assigned denotations (meanings) such as individuals, truth values, or functions from one of these to another. The truth of a sentence, and more interestingly, its logical relation to other sentences, is then evaluated relative to a model.

Formal (or truth-conditional) semantics

Pioneered by the philosopher Donald Davidson, another formalized theory, which aims to associate each natural language sentence with a meta-language description of the conditions under which it is true, for example: 'Snow is white' is true if and only if snow is white. The challenge is to arrive at the truth conditions for any sentences from fixed meanings assigned to the individual words and fixed rules for how to combine them. In practice, truth-conditional semantics is similar to model-theoretic semantics; conceptually, however, they differ in that truth-conditional semantics seeks to connect language with statements about the real world (in the form of meta-language statements), rather than with abstract models.

Lexical and conceptual semantics

This theory is an effort to explain properties of argument structure. The assumption behind this theory is that syntactic properties of phrases reflect the meanings of the words that head them.^[15] With this theory, linguists can better deal with the fact that subtle differences in word meaning correlate with other differences in the syntactic structure that the word appears in.^[15] The way this is gone about is by looking at the internal structure of words.^[16] These small parts that make up the internal structure of words are termed *semantic primitives*.^[16]

Lexical semantics

A linguistic theory that investigates word meaning. This theory understands that the meaning of a word is fully reflected by its context. Here, the meaning of a word is constituted by its contextual relations.^[17] Therefore, a distinction between degrees of participation as well as modes of participation are made.^[17] In order to accomplish this distinction any part of a sentence that bears a meaning and combines with the meanings of other constituents is labeled as a semantic constituent. Semantic constituents that cannot be broken down into more elementary constituents are labeled minimal semantic constituents.^[17]

Computational semantics

Computational semantics is focused on the processing of linguistic meaning. In order to do this concrete algorithms and architectures are described. Within this framework the algorithms and architectures are also analyzed in terms of decidability, time/space complexity, data structures they require and communication protocols.^[18]

Computer science

In computer science, the term *semantics* refers to the meaning of languages, as opposed to their form (syntax). According to Euzenat, semantics "provides the rules for interpreting the syntax which do not provide the meaning directly but constrains the possible interpretations of what is declared."^[19] In other words, semantics is about interpretation of an expression. Additionally, the term is applied to certain types of data structures specifically designed and used for representing information content.

Programming languages

The semantics of programming languages and other languages is an important issue and area of study in computer science. Like the syntax of a language, its semantics can be defined exactly.

For instance, the following statements use different syntaxes, but cause the same instructions to be executed:

Statement	Programming languages
x += y	C, C++, C#, Java, Perl, Python, Ruby, PHP, etc.
x := x + y	ALGOL, BCPL, Simula, ALGOL 68, SETL, Pascal, Smalltalk, Modula-2, Ada, Standard ML, OCaml, Eiffel, Object Pascal (Delphi), Oberon, Dylan, VHDL, etc.
ADD x, y	Assembly languages: Intel 8086
LET X = X + Y	BASIC: early
x = x + y	BASIC: most dialects; Fortran, MATLAB
Set x = x + y	Caché ObjectScript
ADD Y TO X GIVING X	COBOL
(incf x y)	Common Lisp

Generally these operations would all perform an arithmetical addition of 'y' to 'x' and store the result in a variable called 'x'.

Various ways have been developed to describe the semantics of programming languages formally, building on mathematical logic:^[20]

- Operational semantics: The meaning of a construct is specified by the computation it induces when it is executed on a machine. In particular, it is of interest *how* the effect of a computation is produced.
- Denotational semantics: Meanings are modelled by mathematical objects that represent the effect of executing the constructs. Thus *only* the effect is of interest, not how it is obtained.
- Axiomatic semantics: Specific properties of the effect of executing the constructs are expressed as *assertions*. Thus there may be aspects of the executions that are ignored.

Semantic models

Terms such as *semantic network* and *semantic data model* are used to describe particular types of data models characterized by the use of directed graphs in which the vertices denote concepts or entities in the world, and the arcs denote relationships between them.

The Semantic Web refers to the extension of the World Wide Web via embedding added semantic metadata, using semantic data modelling techniques such as Resource Description Framework (RDF) and Web Ontology Language (OWL).

Psychology

In psychology, *semantic memory* is memory for meaning – in other words, the aspect of memory that preserves only the *gist*, the general significance, of remembered experience – while episodic memory is memory for the ephemeral details – the individual features, or the unique particulars of experience. Word meaning is measured by the company they keep, i.e. the relationships among words themselves in a semantic network. The memories may be transferred intergenerationally or isolated in one generation due to a cultural disruption. Different generations may have different experiences at similar points in their own time-lines. This may then create a vertically heterogeneous semantic net for certain words in an otherwise homogeneous culture.^[21] In a network created by people analyzing their understanding of the word (such as Wordnet) the links and decomposition structures of the network are few in

number and kind, and include *part of*, *kind of*, and similar links. In automated ontologies the links are computed vectors without explicit meaning. Various automated technologies are being developed to compute the meaning of words: latent semantic indexing and support vector machines as well as natural language processing, neural networks and predicate calculus techniques.

Ideasthesia is a rare psychological phenomenon that in certain individuals associates semantic and sensory representations. Activation of a concept (e.g., that of the letter A) evokes sensory-like experiences (e.g., of red color).

References

- [1] σημαντικός[[Category:Articles containing Ancient Greek language text (<http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=shmantiko/s>)]]. Liddell, Henry George; Scott, Robert; *A Greek–English Lexicon* at Perseus Project
- [2] The word is derived from the Ancient Greek word *σημαντικός* (*semantikos*), *related to meaning, significant*, from *σημαίω* (*semaino*), *to signify, to indicate*, which is from *σῆμα* (*sema*), *sign, mark, token*. The plural is used in analogy with words similar to *physics*, which was in the neuter plural in Ancient Greek and meant "things relating to nature".
- [3] Neurath, Otto; Carnap, Rudolf; Morris, Charles F. W. (Editors) (1955). *International Encyclopedia of Unified Science*. Chicago, IL: University of Chicago Press.
- [4] Cruse, Alan; *Meaning and Language: An introduction to Semantics and Pragmatics*, Chapter 1, Oxford Textbooks in Linguistics, 2004; Kearns, Kate; *Semantics*, Palgrave MacMillan 2000; Cruse, D. A.; *Lexical Semantics*, Cambridge, MA, 1986.
- [5] Kitcher, Philip; Salmon, Wesley C. (1989). *Scientific Explanation*. Minneapolis, MN: University of Minnesota Press. p. 35.
- [6] Barsalou, L.; *Perceptual Symbol Systems*, Behavioral and Brain Sciences, 22(4), 1999
- [7] Langacker, Ronald W. (1999). *Grammar and Conceptualization*. Berlin/New York: Mouton de Gruyter. ISBN 3-11-016603-8.
- [8] Peregrin, Jaroslav (2003). *Meaning: The Dynamic Turn. Current Research in the Semantics/Pragmatics Interface*. London: Elsevier.
- [9] Gärdenfors, Peter (2000). *Conceptual Spaces: The Geometry of Thought* (<http://www.lucs.lu.se/people/Peter.Gardenfors/Abstracts/conceptualspaces.html>). MIT Press/Bradford Books. ISBN 978-0-585-22837-2. .
- [10] de Saussure, Ferdinand (1916). *The Course of General Linguistics (Cours de linguistique générale)*.
- [11] Matilal, Bimal Krishna (1990). *The Word and the World: India's Contribution to the Study of Language*. Oxford. The Nyaya and Mimamsa schools in Indian *vyākaraṇa* tradition conducted a centuries-long debate on whether sentence meaning arises through composition on word meanings, which are primary; or whether word meanings are obtained through analysis of sentences where they appear. (Chapter 8).
- [12] Bangla Numerals and Problems of Computability (<http://linguistlist.org/pubs/papers/browse-papers-action.cfm?PaperID=7802>)
- [13] Computational Linguistics: A Dissenter's Voice (<http://ssrn.com/abstract=2015944>)
- [14] Lakoff, George; Johnson, Mark (1999). *Philosophy in the Flesh: The embodied mind and its challenge to Western thought. Chapter 1..* New York, NY: Basic Books. OCLC 93961754.
- [15] Levin, Beth; Pinker, Steven; *Lexical & Conceptual Semantics*, Blackwell, Cambridge, MA, 1991
- [16] Jackendoff, Ray; *Semantic Structures*, MIT Press, Cambridge, MA, 1990
- [17] Cruse, D.; *Lexical Semantics*, Cambridge University Press, Cambridge, MA, 1986
- [18] Nerbonne, J.; *The Handbook of Contemporary Semantic Theory* (ed. Lappin, S.), Blackwell Publishing, Cambridge, MA, 1996
- [19] Euzenat, Jerome. *Ontology Matching*. Springer-Verlag Berlin Heidelberg, 2007, p. 36
- [20] Nielson, Hanne Riis; Nielson, Flemming (1995). *Semantics with Applications, A Formal Introduction* (1st ed.). Chichester, England: John Wiley & Sons. ISBN 0-471-92980-8.
- [21] Giannini, A. J.; *Semiotic and Semantic Implications of "Authenticity"*, Psychological Reports, 106(2):611–612, 2010

External links

- semanticsarchive.net (<http://www.semanticsarchive.net/>)
- Teaching page for A-level semantics (<http://www.universalteacher.org.uk/lang/semantics.htm>)
- Chomsky, Noam; *On Referring*, Harvard University, 30 October 2007 (video) (<http://blip.tv/file/471951>)
- Jackendoff, Ray; *Conceptual Semantics*, Harvard University, 13 November 2007(video) (<http://blip.tv/file/509192>)
- *Semantics: an interview with Jerry Fodor* (http://www.revel.inf.br/site2007/_pdf/8/entrevistas/revel_8_interview_jerry_fodor.pdf), ReVEL, vol. 5, n. 8, 2007

Pragmatics

Pragmatics is a subfield of linguistics which studies the ways in which context contributes to meaning. Pragmatics encompasses speech act theory, conversational implicature, talk in interaction and other approaches to language behavior in philosophy, sociology, and linguistics and anthropology.^[1] Unlike semantics, which examines meaning that is conventional or "coded" in a given language, pragmatics studies how the transmission of meaning depends not only on structural and linguistic knowledge (e.g., grammar, lexicon, etc.) of the speaker and listener, but also on the context of the utterance, any preexisting knowledge about those involved, the inferred intent of the speaker, and other factors.^[2] In this respect, pragmatics explains how language users are able to overcome apparent ambiguity, since meaning relies on the manner, place, time etc. of an utterance.^[1]

The ability to understand another speaker's intended meaning is called *pragmatic competence*.^{[3][4][5]}

Structural ambiguity

The sentence "You have a green light" is ambiguous. Without knowing the context, the identity of the speaker, and his or her intent, it is difficult to infer the meaning with confidence. For example:

- It could mean that you have green ambient lighting.
- It could mean that you have a green light while driving your car.
- It could mean that you can go ahead with the project.
- It could mean that your body has a green glow.
- It could mean that you possess a light bulb that is tinted green.

Similarly, the sentence "Sherlock saw the man with binoculars" could mean that Sherlock observed the man by using binoculars, or it could mean that Sherlock observed a man who was holding binoculars.^[6] The meaning of the sentence depends on an understanding of the context and the speaker's intent. As defined in linguistics, a sentence is an abstract entity — a string of words divorced from non-linguistic context — as opposed to an utterance, which is a concrete example of a speech act in a specific context. The closer conscious subjects stick to common words, idioms, phrasings, and topics, the more easily others can surmise their meaning; the further they stray from common expressions and topics, the wider the variations in interpretations. This suggests that sentences do not have meaning intrinsically; there is not a meaning associated with a sentence or word, they can only symbolically represent an idea. *The cat sat on the mat* is a sentence in English; if you say to your sister on Tuesday afternoon, "The cat sat on the mat," this is an example of an utterance. Thus, there is no such thing as a sentence, term, expression or word symbolically representing a single true meaning; it is underspecified (which cat sat on which mat?) and potentially ambiguous. The meaning of an utterance, on the other hand, is inferred based on linguistic knowledge and knowledge of the non-linguistic context of the utterance (which may or may not be sufficient to resolve ambiguity). In mathematics with Berry's paradox there arose a systematic ambiguity with the word "definable". The ambiguity with words shows that the descriptive power of any human language is limited.

Etymology

The word *pragmatics* derives via Latin *pragmaticus* from the Greek πραγματικός (*pragmatikos*), meaning amongst others "fit for action",^[7] which comes from πρᾶγμα (*pragma*), "deed, act",^[8] and that from πράσσω (*prassō*), "to pass over, to practise, to achieve".^[9]

Origins

Pragmatics was a reaction to structuralist linguistics as outlined by Ferdinand de Saussure. In many cases, it expanded upon his idea that language has an analyzable structure, composed of parts that can be defined in relation to others. Pragmatics first engaged only in synchronic study, as opposed to examining the historical development of language. However, it rejected the notion that all meaning comes from signs existing purely in the abstract space of *langue*. Meanwhile, historical pragmatics has also come into being.

Areas of interest

- The study of the speaker's meaning, not focusing on the phonetic or grammatical form of an utterance, but instead on what the speaker's intentions and beliefs are.
- The study of the meaning in context, and the influence that a given context can have on the message. It requires knowledge of the speaker's identities, and the place and time of the utterance.
- The study of implicatures, i.e. the things that are communicated even though they are not explicitly expressed.
- The study of relative distance, both social and physical, between speakers in order to understand what determines the choice of what is said and what is not said.
- The study of what is not meant, as opposed to the intended meaning, i.e. that which is unsaid and unintended, or unintentional.
- **Information Structure**, the study of how utterances are marked in order to efficiently manage the common ground of referred entities between speaker and hearer
- Formal Pragmatics, the study of those aspects of meaning and use, for which context of use is an important factor, by using the methods and goals of formal semantics.

Referential uses of language

When we speak of the referential uses of language we are talking about how we use signs to refer to certain items. Below is an explanation of, first, what a sign is, second, how meanings are accomplished through its usage.

A sign is the link or relationship between a signified and the signifier as defined by Saussure and Huguenin. The signified is some entity or concept in the world. The signifier represents the signified. An example would be:

Signified: the concept cat

Signifier: the word "cat"

The relationship between the two gives the sign meaning. This relationship can be further explained by considering what we mean by "meaning." In pragmatics, there are two different types of meaning to consider: **semantico-referential meaning** and **indexical meaning**. Semantico-referential meaning refers to the aspect of meaning, which describes events in the world that are independent of the circumstance they are uttered in. An example would be propositions such as:

"Santa Claus eats cookies."

In this case, the proposition is describing that Santa Claus eats cookies. The meaning of this proposition does not rely on whether or not Santa Claus is eating cookies at the time of its utterance. Santa Claus could be eating cookies at any time and the meaning of the proposition would remain the same. The meaning is simply describing something that is the case in the world. In contrast, the proposition, "Santa Claus is eating a cookie right now," describes events

that are happening at the time the proposition is uttered.

Semantico-referential meaning is also present in meta-semantical statements such as:

Tiger: omnivorous, a mammal

If someone were to say that a tiger is an omnivorous animal in one context and a mammal in another, the definition of tiger would still be the same. The meaning of the sign tiger is describing some animal in the world, which does not change in either circumstance.

Indexical meaning, on the other hand, is dependent on the context of the utterance and has rules of use. By rules of use, it is meant that indexicals can tell you when they are used, but not what they actually mean.

Example: "I"

Whom "I" refers to depends on the context and the person uttering it.

As mentioned, these meanings are brought about through the relationship between the signified and the signifier. One way to define the relationship is by placing signs in two categories: **referential indexical signs**, also called "shifters," and **pure indexical signs**.

Referential indexical signs are signs where the meaning shifts depending on the context hence the nickname "shifters." 'I' would be considered a referential indexical sign. The referential aspect of its meaning would be '1st person singular' while the indexical aspect would be the person who is speaking (refer above for definitions of semantico-referential and indexical meaning). Another example would be:

"This"

Referential: singular count

Indexical: Close by

A pure indexical sign does not contribute to the meaning of the propositions at all. It is an example of a ""non-referential use of language.""

A second way to define the signified and signifier relationship is C.S. Peirce's **Peircean Trichotomy**. The components of the trichotomy are the following:

1. **Icon:** the signified resembles the signifier (signified: a dog's barking noise, signifier: bow-wow)
2. **Index:** the signified and signifier are linked by proximity or the signifier has meaning only because it is pointing to the signified
3. **Symbol:** the signified and signifier are arbitrarily linked (signified: a cat, signifier: the word cat)

These relationships allow us to use signs to convey what we want to say. If two people were in a room and one of them wanted to refer to a characteristic of a chair in the room he would say "this chair has four legs" instead of "a chair has four legs." The former relies on context (indexical and referential meaning) by referring to a chair specifically in the room at that moment while the latter is independent of the context (semantico-referential meaning), meaning the concept chair.

Non-referential uses of language

Silverstein's "pure" indexes

Michael Silverstein has argued that "nonreferential" or "pure" indices do not contribute to an utterance's referential meaning but instead "signal some particular value of one or more contextual variables."^[10] Although nonreferential indexes are devoid of semantico-referential meaning, they do encode "pragmatic" meaning.

The sorts of contexts that such indexes can mark are varied. Examples include:

- **Sex indexes** are affixes or inflections that index the sex of the speaker, e.g. the verb forms of female Koasati speakers take the suffix "-s".
- **Deference indexes** are words that signal social differences (usually related to status or age) between the speaker and the addressee. The most common example of a deference index is the V form in a language with a T-V

distinction, the widespread phenomenon in which there are multiple second-person pronouns that correspond to the addressee's relative status or familiarity to the speaker. Honorifics are another common form of deference index and demonstrate the speaker's respect or esteem for the addressee via special forms of address and/or self-humbling first-person pronouns.

- An **Affinal taboo index** is an example of avoidance speech and produces and reinforces sociological distance, as seen in the Aboriginal Dyirbal language of Australia. In this language and some others, there is a social taboo against the use of the everyday lexicon in the presence of certain relatives (mother-in-law, child-in-law, paternal aunt's child, and maternal uncle's child). If any of those relatives are present, a Dyirbal speaker has to switch to a completely separate lexicon reserved for that purpose.

In all of these cases, the semantico-referential meaning of the utterances is unchanged from that of the other possible (but often impermissible) forms, but the pragmatic meaning is vastly different.

The performative

Main articles: Performative utterance, Speech act theory

J.L. Austin introduced the concept of the performative, contrasted in his writing with "constative" (i.e. descriptive) utterances. According to Austin's original formulation, a performative is a type of utterance characterized by two distinctive features:

- It is not truth-evaluable (i.e. it is neither true nor false)
- Its uttering *performs* an action rather than simply describing one

However, a performative utterance must also conform to a set of felicity conditions.

Examples:

- "I hereby pronounce you man and wife."
- "I accept your apology."
- "This meeting is now adjourned."

Jakobson's six functions of language

Roman Jakobson, expanding on the work of Karl Bühler, described six "constitutive factors" of a speech event, each of which represents the privileging of a corresponding function, and only one of which is the referential (which corresponds to the **context** of the speech event). The six constitutive factors and their corresponding functions are diagrammed below.

The six constitutive factors of a speech event

Context

Message

Addresser-----Addressee

Contact

Code

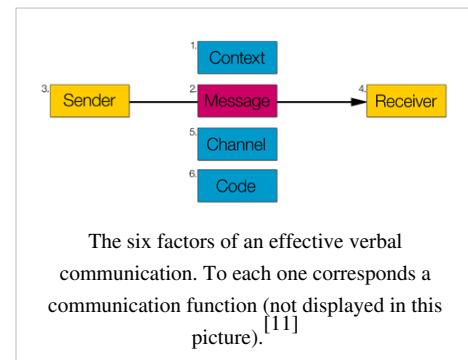
The six functions of language

Referential

Poetic

Emotive-----Conative

Phatic



Metalinguual

- The Referential Function corresponds to the factor of Context and describes a situation, object or mental state. The descriptive statements of the referential function can consist of both definite descriptions and deictic words, e.g. "The autumn leaves have all fallen now."
- The Expressive (alternatively called "emotive" or "affective") Function relates to the Addresser and is best exemplified by interjections and other sound changes that do not alter the denotative meaning of an utterance but do add information about the Addresser's (speaker's) internal state, e.g. "Wow, what a view!"
- The Conative Function engages the Addressee directly and is best illustrated by vocatives and imperatives, e.g. "Tom! Come inside and eat!"
- The Poetic Function focuses on "the message for its own sake"^[12] and is the operative function in poetry as well as slogans.
- The Phatic Function is language for the sake of interaction and is therefore associated with the Contact factor. The Phatic Function can be observed in greetings and casual discussions of the weather, particularly with strangers.
- The Metalinguual (alternatively called "metalinguistic" or "reflexive") Function is the use of language (what Jakobson calls "Code") to discuss or describe itself.

Related fields

There is considerable overlap between pragmatics and sociolinguistics, since both share an interest in linguistic meaning as determined by usage in a speech community. However, sociolinguists tend to be more interested in variations in language within such communities.

Pragmatics helps anthropologists relate elements of language to broader social phenomena; it thus pervades the field of linguistic anthropology. Because pragmatics describes generally the forces in play for a given utterance, it includes the study of power, gender, race, identity, and their interactions with individual speech acts. For example, the study of code switching directly relates to pragmatics, since a switch in code effects a shift in pragmatic force.^[12]

According to Charles W. Morris, pragmatics tries to understand the relationship between signs and their users, while semantics tends to focus on the actual objects or ideas to which a word refers, and syntax (or "syntactics") examines relationships among signs or symbols. Semantics is the literal meaning of an idea whereas pragmatics is the implied meaning of the given idea.

Speech Act Theory, pioneered by J.L. Austin and further developed by John Searle, centers around the idea of the performative, a type of utterance that performs the very action it describes. Speech Act Theory's examination of Illocutionary Acts has many of the same goals as pragmatics, as outlined above.

Pragmatics in literary theory

Pragmatics (more specifically, Speech Act Theory's notion of the performative) underpins Judith Butler's theory of gender performativity. In *Gender Trouble*, she claims that gender and sex are not natural categories, but socially constructed roles produced by "reiterative acting."

In *Excitable Speech* she extends her theory of performativity to hate speech and censorship, arguing that censorship necessarily strengthens any discourse it tries to suppress and therefore, since the state has sole power to define hate speech legally, it is the state that makes hate speech performative.

Jacques Derrida remarked that some work done under Pragmatics aligned well with the program he outlined in his book *Of Grammatology*.

Emile Benveniste argued that the pronouns "I" and "you" are fundamentally distinct from other pronouns because of their role in creating the subject.

Gilles Deleuze and Félix Guattari discuss linguistic pragmatics in the fourth chapter of *A Thousand Plateaus* ("November 20, 1923--Postulates of Linguistics"). They draw three conclusions from Austin: (1) A performative

utterance does not communicate information about an act second-hand—it is the act; (2) Every aspect of language ("semantics, syntax, or even phonetics") functionally interacts with pragmatics; (3) There is no distinction between language and speech. This last conclusion attempts to refute Saussure's division between *langue* and *parole* and Chomsky's distinction between surface structure and deep structure simultaneously.^[13]

Significant works

- J. L. Austin's *How To Do Things With Words*
- Paul Grice's cooperative principle and conversational maxims
- Brown & Levinson's Politeness Theory
- Geoffrey Leech's politeness maxims
- Levinson's Presumptive Meanings
- Jürgen Habermas's universal pragmatics
- Dan Sperber and Deirdre Wilson's relevance theory
- Dallin D. Oaks's *Structural Ambiguity in English: An Applied Grammatical Inventory*

Notes

- [1] Mey, Jacob L. (1993) *Pragmatics: An Introduction*. Oxford: Blackwell (2nd ed. 2001).
- [2] Shaozhong, Liu. "What is pragmatics?" (<http://www.gxnu.edu.cn/Personal/szliu/definition.html>). . Retrieved 18 March 2009.
- [3] Daejin Kim *et al.* (2002) "The Role of an Interactive Book Reading Program in the Development of Second Language Pragmatic Competence", *The Modern Language Journal*, Vol. 86, No. 3 (Autumn, 2002), pp. 332-348
- [4] Masahiro Takimoto (2008) "The Effects of Deductive and Inductive Instruction on the Development of Language Learners' Pragmatic Competence", *The Modern Language Journal*, Vol. 92, No. 3 (Fall, 2008), pp. 369-386
- [5] Dale April Koike (1989) "Pragmatic Competence and Adult L2 Acquisition: Speech Acts in Interlanguage", *The Modern Language Journal*, Vol. 73, No. 3 (Autumn, 1989), pp. 279-289
- [6] <http://ocw.mit.edu/OcwWeb/Linguistics-and-Philosophy/24-903Spring-2005/CourseHome/>
- [7] πραγματικός (<http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=pragmatiko/s>), Henry George Liddell, Robert Scott, *A Greek-English Lexicon*, on Perseus
- [8] πρᾶγμα (<http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=pra=gma>), Henry George Liddell, Robert Scott, *A Greek-English Lexicon*, on Perseus
- [9] πράσσω (<http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=pra/ssw>), Henry George Liddell, Robert Scott, *A Greek-English Lexicon*, on Perseus
- [10] Silverstein 1976
- [11] Middleton, Richard (1990/2002). *Studying Popular Music*, p. 241. Philadelphia: Open University Press. ISBN 0-335-15275-9.
- [12] Duranti 1997
- [13] Deleuze, Gilles and Félix Guattari (1987) [1980]. *A Thousand Plateaus*. University of Minnesota Press.

References

- Austin, J. L. (1962) *How to Do Things With Words*. Oxford University Press.
- Brown, Penelope, and Stephen C. Levinson. (1978) *Politeness: Some Universals in Language Usage*. Cambridge University Press.
- Carston, Robyn (2002) *Thoughts and Utterances: The Pragmatics of Explicit Communication*. Oxford: Blackwell.
- Clark, Herbert H. (1996) "Using Language". Cambridge University Press.
- Cole, Peter, ed.. (1978) *Pragmatics*. (Syntax and Semantics, 9). New York: Academic Press.
- Dijk, Teun A. van. (1977) *Text and Context. Explorations in the Semantics and Pragmatics of Discourse*. London: Longman.
- Grice, H. Paul. (1989) *Studies in the Way of Words*. Cambridge (MA): Harvard University Press.
- Laurence R. Horn and Gregory Ward. (2005) *The Handbook of Pragmatics*. Blackwell.
- Leech, Geoffrey N. (1983) *Principles of Pragmatics*. London: Longman.
- Levinson, Stephen C. (1983) *Pragmatics*. Cambridge University Press.

- Levinson, Stephen C. (2000). Presumptive meanings: The theory of generalized conversational implicature. MIT Press.
- Lin, G. H. C., & Perkins, L. (2005). Cross-cultural discourse of giving and accepting gifts. International Journal of Communication, 16,1-2, 103-12 (ERIC Collections in ED 503685 <http://www.eric.ed.gov/PDFS/ED503685.pdf>)
- Lin, G. H. C. (2007). The significant of pragmatics. Mingdao Journal, Vol, 3, 91-102 ERIC Collection in ED503682<<http://www.eric.ed.gov/PDFS/ED503685.pdf>>
- Lin, G. H. C., Su, S. C. F., & Ho, M. M. H. (2009). Pragmatics and communicative competences. Proceedings of the景文科技大學應用英語系2009研討會, 54-60 (ERIC Collections in ED514939<<http://www.eric.ed.gov/PDFS/ED514939.pdf>>)
- Mey, Jacob L. (1993) *Pragmatics: An Introduction*. Oxford: Blackwell (2nd ed. 2001).
- Kepa Korta and John Perry. (2006) *Pragmatics* (<http://plato.stanford.edu/entries/pragmatics/>). The Stanford Encyclopedia of Philosophy
- Potts, Christopher. (2005) *The Logic of Conventional Implicatures*. Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.
- Robinson, Douglas. (2003). *Performative Linguistics: Speaking and Translating as Doing Things With Words*. London and New York: Routledge.
- Robinson, Douglas. (2006). *Introducing Performative Pragmatics*. London and New York: Routledge.
- Sperber, Dan and Wilson, Deirdre. (2005) Pragmatics (<http://www.dan.sperber.com/pragmatics.htm>). In F. Jackson and M. Smith (eds.) Oxford Handbook of Contemporary Philosophy. OUP, Oxford, 468-501. (Also available here (<http://www.phon.ucl.ac.uk/home/deirdre/papers/Pragmatics2005.doc>)).
- Thomas, Jenny (1995) *Meaning in Interaction: An Introduction to Pragmatics*. Longman.
- Verschueren, Jef. (1999) *Understanding Pragmatics*. London, New York: Arnold Publishers.
- Verschueren, Jef, Jan-Ola Östman, Jan Blommaert, eds. (1995) *Handbook of Pragmatics*. Amsterdam: Benjamins.
- Watzlawick, Paul, Janet Helmick Beavin and Don D. Jackson (1967) *Pragmatics of Human Communication: A Study of Interactional Patterns, Pathologies, and Paradoxes*. New York: Norton.
- Wierzbicka, Anna (1991) *Cross-cultural Pragmatics. The Semantics of Human Interaction*. Berlin, New York: Mouton de Gruyter.
- Yule, George (1996) *Pragmatics* (Oxford Introductions to Language Study). Oxford University Press.
- Silverstein, Michael. 1976. "Shifters, Linguistic Categories, and Cultural Description," in Meaning and Anthropology, Basso and Selby, eds. New York: Harper & Row
- Wardhaugh, Ronald. (2006). "An Introduction to Sociolinguistics". Blackwell.
- Duranti, Alessandro. (1997). "Linguistic Anthropology". Cambridge University Press.
- Carbaugh, Donal. (1990). "Cultural Communication and Intercultural Contact." LEA.
- Mira Ariel (2010). *Defining Pragmatics*. Cambridge University Press. ISBN 978-0-521-73203-1.

External links

- Journal of Pragmatics (http://www.elsevier.com/wps/find/journaldescription.cws_home/505593/description#description), An Interdisciplinary Journal of Language Studies
- Liu, Shaozhong, "What is Pragmatics?", Eprint (<http://www.gxnu.edu.cn/Personal/szliu/definition.html>)
- wiki project in comparative pragmatics: European Communicative Strategies (ECSTRA) (http://www1.ku-eichstaett.de/SLF/EngluVglSW/mediawiki/index.php/ELIX_Wiki:Projects/ECSTRA) (directed by Joachim Grzega)

Formal grammar

Formal grammar

In formal language theory, a **grammar** (when the context isn't given, often called a **formal grammar** for clarity) is a set of formation rules for strings in a formal language. The rules describe how to form strings from the language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings or what can be done with them in whatever context—only their form.

Formal language theory, the discipline which studies formal grammars and languages, is a branch of applied mathematics. Its applications are found in theoretical computer science, theoretical linguistics, formal semantics, mathematical logic, and other areas.

A formal grammar is a set of rules for rewriting strings, along with a "start symbol" from which rewriting must start. Therefore, a grammar is usually thought of as a language generator. However, it can also sometimes be used as the basis for a "recognizer"—a function in computing that determines whether a given string belongs to the language or is grammatically incorrect. To describe such recognizers, formal language theory uses separate formalisms, known as automata theory. One of the interesting results of automata theory is that it is not possible to design a recognizer for certain formal languages.

Parsing is the process of recognizing an utterance (a string in natural languages) by breaking it down to a set of symbols and analyzing each one against the grammar of the language. Most languages have the meanings of their utterances structured according to their syntax—a practice known as compositional semantics. As a result, the first step to describing the meaning of an utterance in language is to break it down part by part and look at its analyzed form (known as its parse tree in computer science, and as its deep structure in generative grammar).

Introductory example

A grammar mainly consists of a set of rules for transforming strings. (If it *only* consisted of these rules, it would be a semi-Thue system.) To generate a string in the language, one begins with a string consisting of only a single *start symbol*. The *production rules* are then applied in any order, until a string that contains neither the start symbol nor designated *nonterminal symbols* is produced. A production rule is applied to a string by replacing one occurrence of its left-hand side in the string by its right-hand side. The language formed by the grammar consists of all distinct strings that can be generated in this manner. Any particular sequence of production rules on the start symbol yields a distinct string in the language. If there are multiple ways of generating the same single string, the grammar is said to be ambiguous.

For example, assume the alphabet consists of *a* and *b*, the start symbol is *S*, and we have the following production rules:

1. $S \rightarrow aSb$
2. $S \rightarrow ba$

then we start with *S*, and can choose a rule to apply to it. If we choose rule 1, we obtain the string *aSb*. If we then choose rule 1 again, we replace *S* with *aSb* and obtain the string *aaSbb*. If we now choose rule 2, we replace *S* with *ba* and obtain the string *aababb*, and are done. We can write this series of choices more briefly, using symbols: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aababb$. The language of the grammar is then the infinite set , where *a*^{*k*} is *a* repeated *k* times (and *n* in particular represents the number of times production rule 1 has been applied).

Formal definition

The syntax of grammars

In the classic formalization of generative grammars first proposed by Noam Chomsky in the 1950s,^{[1][2]} a grammar G consists of the following components:

- A finite set N of *nonterminal symbols*, none of which appear in strings formed from G .
- A finite set Σ of *terminal symbols* that is disjoint from N .
- A finite set P of *production rules*, each rule of the form

$$(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$$

where $*$ is the Kleene star operator and \cup denotes set union. That is, each production rule maps from one string of symbols to another, where the first string (the "head") contains an arbitrary number of symbols provided at least one of them is a nonterminal. In the case that the second string (the "body") consists solely of the empty string – i.e., that it contains no symbols at all – it may be denoted with a special notation (often Λ , e or ϵ) in order to avoid confusion.

- A distinguished symbol $S \in N$ that is the *start symbol*.

A grammar is formally defined as the tuple (N, Σ, P, S) . Such a formal grammar is often called a rewriting system or a phrase structure grammar in the literature.^{[3][4]}

The semantics of grammars

The operation of a grammar can be defined in terms of relations on strings:

- Given a grammar $G = (N, \Sigma, P, S)$, the binary relation \Rightarrow_G (pronounced as "G derives in one step") on strings in $(\Sigma \cup N)^*$ is defined by:
 $x \Rightarrow_G y$ iff $\exists u, v, p, q \in (\Sigma \cup N)^* : (x = upv) \wedge (p \rightarrow q \in P) \wedge (y = uqv)$
- the relation \Rightarrow_G^* (pronounced as G derives in zero or more steps) is defined as the reflexive transitive closure of \Rightarrow_G
- a *sentential form* is a member of $(\Sigma \cup N)^*$ that can be derived in a finite number of steps from the start symbol S ; that is, a sentential form is a member of $\{w \in (\Sigma \cup N)^* \mid S \Rightarrow_G^* w\}$. A sentential form that contains no nonterminal symbols (i.e. is a member of Σ^*) is called a *sentence*.^[5]
- the *language* of G , denoted as $L(G)$, is defined as all those sentences that can be derived in a finite number of steps from the start symbol S ; that is, the set $\{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$.

Note that the grammar $G = (N, \Sigma, P, S)$ is effectively the semi-Thue system $(N \cup \Sigma, P)$, rewriting strings in exactly the same way; the only difference is in that we distinguish specific *nonterminal* symbols which must be rewritten in rewrite rules, and are only interested in rewritings from the designated start symbol S to strings without nonterminal symbols.

Example

For these examples, formal languages are specified using set-builder notation.

Consider the grammar G where $N = \{S, B\}$, $\Sigma = \{a, b, c\}$, S is the start symbol, and P consists of the following production rules:

1. $S \rightarrow aBSc$
2. $S \rightarrow abc$
3. $Ba \rightarrow aB$
4. $Bb \rightarrow bb$

This grammar defines the language $L(G) = \{a^n b^n c^n | n \geq 1\}$ where a^n denotes a string of n consecutive a 's.

Thus, the language is the set of strings that consist of 1 or more a 's, followed by the same number of b 's, followed by the same number of c 's.

Some examples of the derivation of strings in $L(G)$ are:

- $S \Rightarrow_2 abc$
- $S \Rightarrow_1 aBSc \Rightarrow_2 aBabcc \Rightarrow_3 aaBbcc \Rightarrow_4 aabbcc$
- $S \Rightarrow_1 aBSc \Rightarrow_1 aBaBSc \Rightarrow_2 aBaBabcc \Rightarrow_3 aaBBabcc \Rightarrow_3 aaBaBbcc \Rightarrow_3 aaaBBbcc \Rightarrow_4 aaa$

(Note on notation: $P \Rightarrow_i Q$ reads "String P generates string Q by means of production i ", and the generated part is each time indicated in bold type.)

The Chomsky hierarchy

When Noam Chomsky first formalized generative grammars in 1956,^[1] he classified them into types now known as the Chomsky hierarchy. The difference between these types is that they have increasingly strict production rules and can express fewer formal languages. Two important types are *context-free grammars* (Type 2) and *regular grammars* (Type 3). The languages that can be described with such a grammar are called *context-free languages* and *regular languages*, respectively. Although much less powerful than unrestricted grammars (Type 0), which can in fact express any language that can be accepted by a Turing machine, these two restricted types of grammars are most often used because parsers for them can be efficiently implemented.^[6] For example, all regular languages can be recognized by a finite state machine, and for useful subsets of context-free grammars there are well-known algorithms to generate efficient LL parsers and LR parsers to recognize the corresponding languages those grammars generate.

Context-free grammars

A *context-free grammar* is a grammar in which the left-hand side of each production rule consists of only a single nonterminal symbol. This restriction is non-trivial; not all languages can be generated by context-free grammars. Those that can are called *context-free languages*.

The language defined above is not a context-free language, and this can be strictly proven using the pumping lemma for context-free languages, but for example the language $\{a^n b^n | n \geq 1\}$ (at least 1 a followed by the same number of b 's) is context-free, as it can be defined by the grammar G_2 with $N = \{S\}$, $\Sigma = \{a, b\}$, S the start symbol, and the following production rules:

1. $S \rightarrow aSb$
2. $S \rightarrow ab$

A context-free language can be recognized in $O(n^3)$ time (see Big O notation) by an algorithm such as Earley's algorithm. That is, for every context-free language, a machine can be built that takes a string as input and determines in $O(n^3)$ time whether the string is a member of the language, where n is the length of the string.^[7] Deterministic context-free languages is a subset of context-free languages that can be recognized in linear time.^[8] There exist

various algorithms that target either this set of languages or some subset of it.

Regular grammars

In regular grammars, the left hand side is again only a single nonterminal symbol, but now the right-hand side is also restricted. The right side may be the empty string, or a single terminal symbol, or a single terminal symbol followed by a nonterminal symbol, but nothing else. (Sometimes a broader definition is used: one can allow longer strings of terminals or single nonterminals without anything else, making languages easier to denote while still defining the same class of languages.)

The language defined above is not regular, but the language $\{a^n b^m \mid m, n \geq 1\}$ (at least 1 a followed by at least 1 b , where the numbers may be different) is, as it can be defined by the grammar G_3 with $N = \{S, A, B\}$,

$\Sigma = \{a, b\}$, S the start symbol, and the following production rules:

1. $S \rightarrow aA$
2. $A \rightarrow aA$
3. $A \rightarrow bB$
4. $B \rightarrow bB$
5. $B \rightarrow \epsilon$

All languages generated by a regular grammar can be recognized in linear time by a finite state machine. Although, in practice, regular grammars are commonly expressed using regular expressions, some forms of regular expression used in practice do not strictly generate the regular languages and do not show linear recognitional performance due to those deviations.

Other forms of generative grammars

Many extensions and variations on Chomsky's original hierarchy of formal grammars have been developed, both by linguists and by computer scientists, usually either in order to increase their expressive power or in order to make them easier to analyze or parse. Some forms of grammars developed include:

- Tree-adjoining grammars increase the expressiveness of conventional generative grammars by allowing rewrite rules to operate on parse trees instead of just strings.^[9]
- Affix grammars^[10] and attribute grammars^{[11][12]} allow rewrite rules to be augmented with semantic attributes and operations, useful both for increasing grammar expressiveness and for constructing practical language translation tools.

Analytic grammars

Though there is a tremendous body of literature on parsing algorithms, most of these algorithms assume that the language to be parsed is initially *described* by means of a *generative* formal grammar, and that the goal is to transform this generative grammar into a working parser. Strictly speaking, a generative grammar does not in any way correspond to the algorithm used to parse a language, and various algorithms have different restrictions on the form of production rules that are considered well-formed.

An alternative approach is to formalize the language in terms of an analytic grammar in the first place, which more directly corresponds to the structure and semantics of a parser for the language. Examples of analytic grammar formalisms include the following:

- The Language Machine^[13] directly implements unrestricted analytic grammars. Substitution rules are used to transform an input to produce outputs and behaviour. The system can also produce the lm-diagram^[14] which shows what happens when the rules of an unrestricted analytic grammar are being applied.
- Top-down parsing language (TDPL): a highly minimalist analytic grammar formalism developed in the early 1970s to study the behavior of top-down parsers.^[15]

- Link grammars: a form of analytic grammar designed for linguistics, which derives syntactic structure by examining the positional relationships between pairs of words.^{[16][17]}
- Parsing expression grammars (PEGs): a more recent generalization of TDPL designed around the practical expressiveness needs of programming language and compiler writers.^[18]

References

- [1] Chomsky, Noam (1956). "Three Models for the Description of Language". *IRE Transactions on Information Theory* **2** (2): 113–123. doi:10.1109/TIT.1956.1056813.
- [2] Chomsky, Noam (1957). *Syntactic Structures*. The Hague: Mouton.
- [3] Ginsburg, Seymour (1975). *Algebraic and automata theoretic properties of formal languages*. North-Holland. pp. 8–9. ISBN 0-7204-2506-9.
- [4] Harrison, Michael A. (1978). *Introduction to Formal Language Theory*. Reading, Mass.: Addison-Wesley Publishing Company. pp. 13. ISBN 0-201-02955-3.
- [5] Sentential Forms (<http://www.seas.upenn.edu/~cit596/notes/dave/cfg7.html>), Context-Free Grammars, David Matuszek
- [6] Grune, Dick & Jacobs, Ceriel H., *Parsing Techniques – A Practical Guide*, Ellis Horwood, England, 1990.
- [7] Earley, Jay, "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, Vol. 13 No. 2, pp. 94-102, February 1970.
- [8] Knuth, Donald (July 1965). "On the Translation of Languages from Left to Right" (<http://www.cs.dartmouth.edu/~mckeeeman/cs48/mxcom/doc/knuth65.pdf>). *Information and Control* **8**: 707 - 639. . Retrieved 29 May 2011.
- [9] Joshi, Aravind K., et al., "Tree Adjunct Grammars," *Journal of Computer Systems Science*, Vol. 10 No. 1, pp. 136-163, 1975.
- [10] Koster , Cornelis H. A., "Affix Grammars," in *ALGOL 68 Implementation*, North Holland Publishing Company, Amsterdam, p. 95-109, 1971.
- [11] Knuth, Donald E., "Semantics of Context-Free Languages," *Mathematical Systems Theory*, Vol. 2 No. 2, pp. 127-145, 1968.
- [12] Knuth, Donald E., "Semantics of Context-Free Languages (correction)," *Mathematical Systems Theory*, Vol. 5 No. 1, pp 95-96, 1971.
- [13] <http://languagemachine.sourceforge.net/>
- [14] <http://languagemachine.sourceforge.net/picturebook.html>
- [15] Birman, Alexander, *The TMG Recognition Schema*, Doctoral thesis, Princeton University, Dept. of Electrical Engineering, February 1970.
- [16] Sleator, Daniel D. & Temperly, Davy, "Parsing English with a Link Grammar," Technical Report CMU-CS-91-196, Carnegie Mellon University Computer Science, 1991.
- [17] Sleator, Daniel D. & Temperly, Davy, "Parsing English with a Link Grammar," *Third International Workshop on Parsing Technologies*, 1993. (Revised version of above report.)
- [18] Ford, Bryan, *Packrat Parsing: a Practical Linear-Time Algorithm with Backtracking*, Master's thesis, Massachusetts Institute of Technology, Sept. 2002.

External links

- Yearly Formal Grammar conference (<http://cs.haifa.ac.il/~shuly/fg/>)

Formal language

In mathematics, computer science, and linguistics, a **formal language** is a set of strings of symbols.

The alphabet of a formal language is the set of symbols, letters, or tokens from which the strings of the language may be formed; frequently it is required to be finite. The strings formed from this alphabet are called words, and the words that belong to a particular formal language are sometimes called *well-formed words* or *well-formed formulas*. A formal language is often defined by means of a formal grammar such as a regular grammar or context-free grammar, also called its formation rule.

The field of **formal language theory** studies the purely syntactical aspects of such languages—that is, their internal structural patterns. Formal language theory sprang out of linguistics, as a way of understanding the syntactic regularities of natural languages. In computer science, formal languages are often used as the basis for defining programming languages and other systems in which the words of the language are associated with particular meanings or semantics. In computational complexity theory, decision problems are typically defined as formal languages, and complexity classes are defined as the sets of the formal languages that can be parsed by machines with limited computational power. In logic and the foundations of mathematics, formal languages are used to represent the syntax of axiomatic systems, and mathematical formalism is the philosophy that all of mathematics can be reduced to the syntactic manipulation of formal languages in this way.

History

The first formal language is thought to be the one used by Gottlob Frege in his *Begriffsschrift* (1879), literally meaning "concept writing", and which Frege described as a "formal language of pure thought."^[1]

Axel Thue's early Semi-Thue system which can be used for rewriting strings was influential on formal grammars.

Words over an alphabet

An **alphabet**, in the context of formal languages, can be any set, although it often makes sense to use an alphabet in the usual sense of the word, or more generally a character set such as ASCII. Alphabets can also be infinite; e.g. first-order logic is often expressed using an alphabet which, besides symbols such as \wedge , \neg , \forall and parentheses, contains infinitely many elements x_0, x_1, x_2, \dots that play the role of variables. The elements of an alphabet are called its **letters**.

A **word** over an alphabet can be any finite sequence, or string, of letters. The set of all words over an alphabet Σ is usually denoted by Σ^* (using the Kleene star). The length of a word is the number of letters it is composed of. For any alphabet there is only one word of length 0, the *empty word*, which is often denoted by e , ϵ or λ . By concatenation one can combine two words to form a new word, whose length is the sum of the lengths of the original words. The result of concatenating a word with the empty word is the original word.

In some applications, especially in logic, the alphabet is also known as the *vocabulary* and words are known as *formulas* or *sentences*; this breaks the letter/word metaphor and replaces it by a word/sentence metaphor.

Definition

A **formal language** L over an alphabet Σ is a subset of Σ^* , that is, a set of words over that alphabet.

In computer science and mathematics, which do not usually deal with natural languages, the adjective "formal" is often omitted as redundant.

While formal language theory usually concerns itself with formal languages that are described by some syntactical rules, the actual definition of the concept "formal language" is only as above: a (possibly infinite) set of finite-length strings, no more nor less. In practice, there are many languages that can be described by rules, such as regular languages or context-free languages. The notion of a formal grammar may be closer to the intuitive concept of a "language," one described by syntactic rules. By an abuse of the definition, a particular formal language is often thought of as being equipped with a formal grammar that describes it.

Examples

The following rules describe a formal language L over the alphabet $\Sigma = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, = \}$:

- Every nonempty string that does not contain "+" or "=" and does not start with "0" is in L .
- The string "0" is in L .
- A string containing "=" is in L if and only if there is exactly one "=", and it separates two valid strings of L .
- A string containing "+" but not "=" is in L if and only if every "+" in the string separates two valid strings of L .
- No string is in L other than those implied by the previous rules.

Under these rules, the string "23+4=555" is in L , but the string "=234=+" is not. This formal language expresses natural numbers, well-formed addition statements, and well-formed addition equalities, but it expresses only what they look like (their syntax), not what they mean (semantics). For instance, nowhere in these rules is there any indication that "0" means the number zero, or that "+" means addition.

Constructions

For finite languages one can explicitly enumerate all well-formed words. For example, we can describe a language L as just $L = \{"a", "b", "ab", "cba"\}$. The degenerate case of this construction is the **empty language**, which contains no words at all ($L = \emptyset$).

However, even over a finite (non-empty) alphabet such as $\Sigma = \{a, b\}$ there are infinitely many words: "a", "abb", "ababba", "aaababbbbaab", Therefore formal languages are typically infinite, and describing an infinite formal language is not as simple as writing $L = \{"a", "b", "ab", "cba"\}$. Here are some examples of formal languages:

- $L = \Sigma^*$, the set of *all* words over Σ ;
- $L = \{"a"\}^* = \{"a^n\}$, where n ranges over the natural numbers and " a^n " means "a" repeated n times (this is the set of words consisting only of the symbol "a");
- the set of syntactically correct programs in a given programming language (the syntax of which is usually defined by a context-free grammar);
- the set of inputs upon which a certain Turing machine halts; or
- the set of maximal strings of alphanumeric ASCII characters on this line, i.e., the set { "the", "set", "of", "maximal", "strings", "alphanumeric", "ASCII", "characters", "on", "this", "line", "i", "e" }.

Language-specification formalisms

Formal language theory rarely concerns itself with particular languages (except as examples), but is mainly concerned with the study of various types of formalisms to describe languages. For instance, a language can be given as

- those strings generated by some formal grammar;
- those strings described or matched by a particular regular expression;
- those strings accepted by some automaton, such as a Turing machine or finite state automaton;
- those strings for which some decision procedure (an algorithm that asks a sequence of related YES/NO questions) produces the answer YES.

Typical questions asked about such formalisms include:

- What is their expressive power? (Can formalism X describe every language that formalism Y can describe? Can it describe other languages?)
- What is their recognizability? (How difficult is it to decide whether a given word belongs to a language described by formalism X ?)
- What is their comparability? (How difficult is it to decide whether two languages, one described in formalism X and one in formalism Y , or in X again, are actually the same language?).

Surprisingly often, the answer to these decision problems is "it cannot be done at all", or "it is extremely expensive" (with a characterization of how expensive). Therefore, formal language theory is a major application area of computability theory and complexity theory. Formal languages may be classified in the Chomsky hierarchy based on the expressive power of their generative grammar as well as the complexity of their recognizing automaton. Context-free grammars and regular grammars provide a good compromise between expressivity and ease of parsing, and are widely used in practical applications.

Operations on languages

Certain operations on languages are common. This includes the standard set operations, such as union, intersection, and complement. Another class of operation is the element-wise application of string operations.

Examples: suppose L_1 and L_2 are languages over some common alphabet.

- The *concatenation* $L_1 L_2$ consists of all strings of the form vw where v is a string from L_1 and w is a string from L_2 .
- The *intersection* $L_1 \cap L_2$ of L_1 and L_2 consists of all strings which are contained in both languages
- The *complement* $\neg L$ of a language with respect to a given alphabet consists of all strings over the alphabet that are not in the language.
- The *Kleene star*: the language consisting of all words that are concatenations of 0 or more words in the original language;
- *Reversal*:
 - Let e be the empty word, then $e^R = e$, and
 - for each non-empty word $w = x_1 \dots x_n$ over some alphabet, let $w^R = x_n \dots x_1$,
 - then for a formal language L , $L^R = \{w^R \mid w \in L\}$.
- String homomorphism

Such string operations are used to investigate closure properties of classes of languages. A class of languages is closed under a particular operation when the operation, applied to languages in the class, always produces a language in the same class again. For instance, the context-free languages are known to be closed under union, concatenation, and intersection with regular languages, but not closed under intersection or complement. The theory of trios and abstract families of languages studies the most common closure properties of language families in their own right.^[2]

Closure properties of language families (

Operation		Regular	DCFL	CFL	IND	CSL	recursive	RE
Union	$\{w \mid w \in L_1 \vee w \in L_2\}$	Yes	No	Yes	Yes	Yes	Yes	Yes
Intersection	$\{w \mid w \in L_1 \wedge w \in L_2\}$	Yes	No	No	No	Yes	Yes	Yes
Complement	$\{w \mid w \notin L_1\}$	Yes	Yes	No	No	Yes	Yes	No
Concatenation	$L_1 \cdot L_2 = \{w \cdot z \mid w \in L_1 \wedge z \in L_2\}$	Yes	No	Yes	Yes	Yes	Yes	Yes
Kleene star	$L_1^* = \{\epsilon\} \cup \{w \cdot z \mid w \in L_1 \wedge z \in L_1^*\}$	Yes	No	Yes	Yes	Yes	Yes	Yes
Homomorphism		Yes	No	Yes	Yes	No	No	Yes
e-free Homomorphism		Yes	No	Yes	Yes	Yes	Yes	Yes
Substitution		Yes	No	Yes	Yes	Yes	No	Yes
Inverse Homomorphism		Yes	Yes	Yes	Yes	Yes	Yes	Yes
Reverse	$\{w^R \mid w \in L\}$	Yes	No	Yes	Yes	Yes	Yes	Yes
Intersection with a regular language	$\{w \mid w \in L_1 \wedge w \in R\}, R \text{ regular}$	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Applications

Programming languages

A compiler usually has two distinct components. A lexical analyzer, generated by a tool like `lex`, identifies the tokens of the programming language grammar, e.g. identifiers or keywords, which are themselves expressed in a simpler formal language, usually by means of regular expressions. At the most basic conceptual level, a parser, usually generated by a parser generator like `yacc`, attempts to decide if the source program is valid, that is if it belongs to the programming language for which the compiler was built. Of course, compilers do more than just parse the source code—they usually translate it into some executable format. Because of this, a parser usually outputs more than a yes/no answer, typically an abstract syntax tree, which is used by subsequent stages of the compiler to eventually generate an executable containing machine code that runs directly on the hardware, or some intermediate code that requires a virtual machine to execute.

Formal theories, systems and proofs

In mathematical logic, a *formal theory* is a set of sentences expressed in a formal language.

A *formal system* (also called a *logical calculus*, or a *logical system*) consists of a formal language together with a deductive apparatus (also called a *deductive system*). The deductive apparatus may consist of a set of transformation rules which may be interpreted as valid rules of inference or a set of axioms, or have both. A formal system is used to derive one expression from one or more other expressions. Although a formal language can be identified with its formulas, a formal system cannot be likewise identified by its theorems. Two formal systems \mathcal{FS} and \mathcal{FS}' may have all the same theorems and yet differ in some significant proof-theoretic way (a formula A may be a syntactic consequence of a formula B in one but not another for instance).

A *formal proof* or *derivation* is a finite sequence of well-formed formulas (which may be interpreted as propositions) each of which is an axiom or follows from the preceding formulas in the sequence by a rule of inference. The last sentence in the sequence is a theorem of a formal system. Formal proofs are useful because their theorems can be interpreted as true propositions.

Interpretations and models

Formal languages are entirely syntactic in nature but may be given semantics that give meaning to the elements of the language. For instance, in mathematical logic, the set of possible formulas of a particular logic is a formal language, and an interpretation assigns a meaning to each of the formulas—usually, a truth value.

The study of interpretations of formal languages is called formal semantics. In mathematical logic, this is often done in terms of model theory. In model theory, the terms that occur in a formula are interpreted as mathematical structures, and fixed compositional interpretation rules determine how the truth value of the formula can be derived from the interpretation of its terms; a *model* for a formula is an interpretation of terms such that the formula becomes true.

Symbols and strings of symbols

Well-formed formulas

Theorems

This diagram shows the syntactic divisions within a formal system. Strings of symbols may be broadly divided into nonsense and well-formed formulas. The set of well-formed formulas is divided into theorems and non-theorems.

References

Notes

- [1] Martin Davis (1995). "Influences of Mathematical Logic on Computer Science" (<http://books.google.com/books?id=YafIDVd1Z68C&pg=PA290>). In Rolf Herken. *The universal Turing machine: a half-century survey*. Springer. p. 290. ISBN 978-3-211-82637-9. .
- [2] Hopcroft & Ullman (1979), Chapter 11: Closure properties of families of languages.

General references

- A. G. Hamilton, *Logic for Mathematicians*, Cambridge University Press, 1978, ISBN 0-521-21838-1.
- Seymour Ginsburg, *Algebraic and automata theoretic properties of formal languages*, North-Holland, 1975, ISBN 0-7204-2506-9.
- Michael A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley, 1978.
- John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing, Reading Massachusetts, 1979. ISBN 81-7808-347-7.
- Rautenberg, Wolfgang (2010), *A Concise Introduction to Mathematical Logic* (<http://www.springerlink.com/content/978-1-4419-1220-6/>) (3rd ed.), New York: Springer Science+Business Media, doi:10.1007/978-1-4419-1221-3, ISBN 978-1-4419-1220-6.
- Grzegorz Rozenberg, Arto Salomaa, *Handbook of Formal Languages: Volume I-III*, Springer, 1997, ISBN 3-540-61486-9.
- Patrick Suppes, *Introduction to Logic*, D. Van Nostrand, 1957, ISBN 0-442-08072-7.

External links

- Alphabet (<http://planetmath.org/?op=getobj&from=objects&id=1681>), PlanetMath.org.
- Language (<http://planetmath.org/?op=getobj&from=objects&id=1767>), PlanetMath.org.
- University of Maryland, Formal Language Definitions (http://www.csee.umbc.edu/help/theory/lang_def.shtml)
- James Power, "Notes on Formal Language Theory and Parsing" (<http://www.cs.nuim.ie/~jpower/Courses/parsing/>), 29 November 2002.
- Drafts of some chapters in the "Handbook of Formal Language Theory", Vol. 1-3, G. Rozenberg and A. Salomaa (eds.), Springer Verlag, (1997):t
 - Alexandru Mateescu and Arto Salomaa, "Preface" in Vol.1, pp. v-viii, and "Formal Languages: An Introduction and a Synopsis", Chapter 1 in Vol. 1, pp.1-39 (<http://www.cs.cmu.edu/~lkontor/noam/Mateescu-Salomaa.pdf>)
 - Sheng Yu, "Regular Languages", Chapter 2 in Vol. 1 (<http://www.csd.uwo.ca/~syu/public/draft.ps>)
 - Jean-Michel Autebert, Jean Berstel, Luc Boasson, "Context-Free Languages and Push-Down Automata", Chapter 3 in Vol. 1 (<http://citeseer.ist.psu.edu/248295.html>)
 - Christian Choffrut and Juhani Karhumäki, "Combinatorics of Words", Chapter 6 in Vol. 1 (<http://www.liafa.jussieu.fr/~cc/PUBLICATIONS/CKTUCS.PS.gz>)
 - Tero Harju and Juhani Karhumäki, "Morphisms", Chapter 7 in Vol. 1, pp. 439 - 510 (<http://users.utu.fi/harju/articles/morph.pdf>)
 - Jean-Eric Pin, "Syntactic semigroups", Chapter 10 in Vol. 1, pp. 679-746 (<http://www.liafa.jussieu.fr/~jep/PDF/HandBook.pdf>)
 - M. Crochemore and C. Hancart, "Automata for matching patterns", Chapter 9 in Vol. 2 (<http://www-igm.univ-mlv.fr/~mac/REC/DOC/B4.ps>)
 - Dora Giammarresi, Antonio Restivo, "Two-dimensional Languages", Chapter 4 in Vol. 3, pp. 215 - 267 (<http://bruno.maitresdumonde.com/optinfo/Spe-MP/dmds1998/2dlang/giammaresi-restivo-paper.ps>)

Formation rule

In mathematical logic, **formation rules** are rules for describing which strings of symbols formed from the alphabet of a formal language are syntactically valid within the language. These rules only address the location and manipulation of the strings of the language. It does not describe anything else about a language, such as its semantics (i.e. what the strings mean). (See also formal grammar).

Formal language

A *formal language* is an organized set of symbols the essential feature of which is that it can be precisely defined in terms of just the shapes and locations of those symbols. Such a language can be defined, then, without any reference to any meanings of any of its expressions; it can exist before any interpretation is assigned to it—that is, before it has any meaning. A formal grammar determines which symbols and sets of symbols are formulas in a formal language.

Formal systems

A *formal system* (also called a *logical calculus*, or a *logical system*) consists of a formal language together with a deductive apparatus (also called a *deductive system*). The deductive apparatus may consist of a set of transformation rules (also called *inference rules*) or a set of axioms, or have both. A formal system is used to derive one expression from one or more other expressions. Propositional and predicate calculi are examples of formal systems.

Propositional and predicate logic

The formation rules of a propositional calculus may, for instance, take a form such that;

- if we take Φ to be a propositional formula we can also take $\neg\Phi$ to be a formula;
- if we take Φ and Ψ to be a propositional formulas we can also take $(\Phi \& \Psi)$, $(\Phi \rightarrow \Psi)$, $(\Phi \vee \Psi)$ and $(\Phi \leftrightarrow \Psi)$ to also be formulas.

A predicate calculus will usually include all the same rules as a propositional calculus, with the addition of quantifiers such that if we take Φ to be a formula of propositional logic and α as a variable then we can take $(\forall \alpha)\Phi$ and $(\exists \alpha)\Phi$ each to be formulas of our predicate calculus.

String (computer science)

In computer programming, a **string** is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and/or the length changed, or it may be fixed (after creation). A string is generally understood as a data type and is often implemented as an array of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. A string may also denote more general array data types and/or other sequential data types and structures; terms such as *<data type> string* or *string of <data types>* are sometimes used to denote strings in which the stored data represents other data types.

Depending on programming language and/or precise data type used, a variable declared to be a string may either cause storage in memory to be statically allocated for a predetermined *max length* or employ dynamic allocation to allow it to hold *chronologically* variable number of elements.

When a string appears literally in source code, it is known as a string literal and has a representation that denotes it as such.

In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set called an alphabet.

Formal theory

Let Σ be an *alphabet*, a non-empty finite set. Elements of Σ are called *symbols* or *characters*. A **string** (or **word**) over Σ is any finite sequence of characters from Σ . For example, if $\Sigma = \{0, 1\}$, then 0101 is a string over Σ .

The *length* of a string is the number of characters in the string (the length of the sequence) and can be any non-negative integer. The *empty string* is the unique string over Σ of length 0, and is denoted ε or λ .

The set of all strings over Σ of length n is denoted Σ^n . For example, if $\Sigma = \{0, 1\}$, then $\Sigma^2 = \{00, 01, 10, 11\}$. Note that $\Sigma^0 = \{\varepsilon\}$ for any alphabet Σ .

The set of all strings over Σ of any length is the Kleene closure of Σ and is denoted Σ^* . In terms of Σ^n ,

$$\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$$

For example, if $\Sigma = \{0, 1\}$, $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \dots\}$. Although Σ^* itself is countably infinite, all elements of Σ^* have finite length.

A set of strings over Σ (i.e. any subset of Σ^*) is called a *formal language* over Σ . For example, if $\Sigma = \{0, 1\}$, the set of strings with an even number of zeros ($\{\varepsilon, 1, 00, 11, 001, 010, 100, 111, 0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111, \dots\}$) is a formal language over Σ .

Concatenation and substrings

Concatenation is an important binary operation on Σ^* . For any two strings s and t in Σ^* , their concatenation is defined as the sequence of characters in s followed by the sequence of characters in t , and is denoted st . For example, if $\Sigma = \{a, b, \dots, z\}$, $s = \text{bear}$, and $t = \text{hug}$, then $st = \text{bearhug}$ and $ts = \text{hugbear}$.

String concatenation is an associative, but non-commutative operation. The empty string serves as the identity element; for any string s , $\varepsilon s = s\varepsilon = s$. Therefore, the set Σ^* and the concatenation operation form a monoid, the free monoid generated by Σ . In addition, the length function defines a monoid homomorphism from Σ^* to the non-negative integers (that is, a function $L : \Sigma^* \mapsto \mathbb{N}_0$, such that $L(st) = L(s) + L(t) \quad \forall s, t \in \Sigma^*$).

A string s is said to be a *substring* or *factor* of t if there exist (possibly empty) strings u and v such that $t = usv$. The relation "is a substring of" defines a partial order on Σ^* , the least element of which is the empty string.

Prefixes and suffixes

A string s is said to be a prefix of t if there exists a string u such that $t = su$. If u is nonempty, s is said to be a *proper* prefix of t . Symmetrically, a string s is said to be a suffix of t if there exists a string u such that $t = us$. If u is nonempty, s is said to be a *proper* suffix of t . Suffixes and prefixes are substrings of t .

Rotations

A string $s = uv$ is said to be a rotation of t if $t = vu$. For example, if $\Sigma = \{0, 1\}$ the string 0011001 is a rotation of 0100110.

Reversal

The reverse of a string is a string with the same symbols but in reverse order. For example, if $s = abc$ (where a, b, and c are symbols of the alphabet), then the reverse of s is cba. A string that is the reverse of itself (e.g., $s = madam$) is called a palindrome, which also includes the empty string and all strings of length 1.

Lexicographical ordering

It is often useful to define an ordering on a set of strings. If the alphabet Σ has a total order (cf. alphabetical order) one can define a total order on Σ^* called lexicographical order. For example, if $\Sigma = \{0, 1\}$ and $0 < 1$, then the lexicographical order on Σ^* includes the relationships $\epsilon < 0 < 00 < 000 < \dots < 0001 < 001 < 01 < 010 < 011 < 0110 < 0111 < 1 < 10 < 100 < 101 < 111 < 1111 < 11111 \dots$

String operations

A number of additional operations on strings commonly occur in the formal theory. These are given in the article on string operations.

Topology

Strings admit the following interpretation as nodes on a graph:

- Fixed-length strings can be viewed as nodes on a hypercube
- Variable-length strings (of finite length) can be viewed as nodes on the k -ary tree, where k is the number of symbols in Σ
- Infinite strings can be viewed as infinite paths on the k -ary tree.

The natural topology on the set of fixed-length strings or variable length strings is the discrete topology, but the natural topology on the set of infinite strings is the limit topology, viewing the set of infinite strings as the inverse limit of the sets of finite strings. This is the construction used for the p -adic numbers and some constructions of the Cantor set, and yields the same topology.

Isomorphisms between string representations of topologies can be found by normalizing according to the lexicographically minimal string rotation.

String datatypes

A **string datatype** is a datatype modeled on the idea of a formal string. Strings are such an important and useful datatype that they are implemented in nearly every programming language. In some languages they are available as primitive types and in others as composite types. The syntax of most high-level programming languages allows for a string, usually quoted in some way, to represent an instance of a string datatype; such a meta-string is called a *literal* or *string literal*.

String length

Although formal strings can have an arbitrary (but finite) length, the length of strings in real languages is often constrained to an artificial maximum. In general, there are two types of string datatypes: *fixed-length strings*, which have a fixed maximum length and which use the same amount of memory whether this maximum is reached or not, and *variable-length strings*, whose length is not arbitrarily fixed and which use varying amounts of memory depending on their actual size. Most strings in modern programming languages are variable-length strings. Despite the name, even variable-length strings are limited in length, although, in general, the limit depends only on the amount of memory available. The string length can be stored as a separate integer (which puts a theoretical limit on the length) or implicitly through a termination character, usually a character value with all bits zero. See also "Null-terminated" below.

Character encoding

String datatypes have historically allocated one byte per character, and, although the exact character set varied by region, character encodings were similar enough that programmers could often get away with ignoring this, since characters a program treated specially (such as period and space and comma) were in the same place in all the encodings a program would encounter. These character sets were typically based on ASCII or EBCDIC.

Logographic languages such as Chinese, Japanese, and Korean (known collectively as CJK) need far more than 256 characters (the limit of a one 8-bit byte per-character encoding) for reasonable representation. The normal solutions involved keeping single-byte representations for ASCII and using two-byte representations for CJK ideographs. Use of these with existing code led to problems with matching and cutting of strings, the severity of which depended on how the character encoding was designed. Some encodings such as the EUC family guarantee that a byte value in the ASCII range will represent only that ASCII character, making the encoding safe for systems that use those characters as field separators. Other encodings such as ISO-2022 and Shift-JIS do not make such guarantees, making matching on byte codes unsafe. These encodings also were not "self-synchronizing", so that locating character boundaries required backing up to the start of a string, and pasting two strings together could result in corruption of the second string (these problems were much less with EUC as any ASCII character did synchronize the encoding).

Unicode has simplified the picture somewhat. Most programming languages now have a datatype for Unicode strings. Unicode's preferred byte stream format UTF-8 is designed not to have the problems described above for older multibyte encodings. All UTF-8, UTF-16 and UTF-32 require the programmer to know that the fixed-size code units are different than the "characters", the main difficulty currently is incorrectly designed API's that attempt to hide this difference.

Implementations

Some languages like C++ implement strings as templates that can be used with any datatype, but this is the exception, not the rule.

Some languages, such as C++ and Ruby, normally allow the contents of a string to be changed after it has been created; these are termed *mutable* strings. In other languages, such as Java and Python, the value is fixed and a new string must be created if any alteration is to be made; these are termed *immutable* strings.

Strings are typically implemented as arrays of characters (or code units), in order to allow fast access to individual characters. A few languages such as Haskell implement them as linked lists instead.

Some languages, such as Prolog and Erlang, avoid implementing a dedicated string datatype at all, instead adopting the convention of representing strings as lists of character codes.

Representations

Representations of strings depend heavily on the choice of character repertoire and the method of character encoding. Older string implementations were designed to work with repertoire and encoding defined by ASCII, or more recent extensions like the ISO 8859 series. Modern implementations often use the extensive repertoire defined by Unicode along with a variety of complex encodings such as UTF-8 and UTF-16.

Most string implementations are very similar to variable-length arrays with the entries storing the character codes of corresponding characters. The principal difference is that, with certain encodings, a single logical character may take up more than one entry in the array. This happens for example with UTF-8, where single characters can take anywhere from one to four bytes. In these cases, the logical length of the string (number of characters) differs from the logical length of the array (number of bytes in use). UTF-32 is the only Unicode encoding that avoids this problem.

Null-terminated

The length of a string can be stored implicitly by using a special terminating character; often this is the null character (NUL), which has all bits zero, a convention used and perpetuated by the popular C programming language.^[1] Hence, this representation is commonly referred to as C string. The length of a string can also be stored explicitly, for example by prefixing the string with the length as a byte value (a convention used in many Pascal dialects): as a consequence, some people call it a P-string. Storing the string length as byte limits the maximum string length to 255. To avoid such limitations, improved implementations of P-strings use 16-, 32-, or 64-bit words to store the string length. When the *length* field covers the address space, strings are limited only by the available memory.

In terminated strings, the terminating code is not an allowable character in any string. Strings with *length* field do not have this limitation and can also store arbitrary binary data. In C two things are needed to handle binary data, a character pointer and the length of the data.

The term *bytestring* usually indicates a general-purpose string of bytes, rather than strings of only (readable) characters, strings of bits, or such. Byte strings often imply that bytes can take any value and any data can be stored as-is, meaning that there should be no value interpreted as a termination value.

An example of a *null-terminated string* stored in a 10-byte buffer, along with its ASCII (or more modern UTF-8) representation as 8-bit hexadecimal numbers is:

F	R	A	N	K	NUL	k	e	f	w
46 _h	52 _h	41 _h	4E _h	4B _h	00 _h	6B _h	65 _h	66 _h	77 _h

The length of the string in the above example, "FRANK", is 5 characters, but it occupies 6 bytes. Characters after the terminator do not form part of the representation; they may be either part of another string or just garbage. (Strings of this form are sometimes called *ASCIZ strings*, after the original assembly language directive used to declare them.)

Length-prefixed

Here is the equivalent **Pascal string** stored in a 10-byte buffer, along with its ASCII / UTF-8 representation:

length	F	R	A	N	K	k	e	f	w
5 _{dec}	46 _h	52 _h	41 _h	4E _h	4B _h	6B _h	65 _h	66 _h	77 _h

Object oriented

An object oriented language will typically implement a string like this:

```
class string {
    int length;
    char *text;
};
```

although this implementation is hidden, and accessed through member functions. The "text" will be a dynamically allocated memory area, that might be expanded if needed. See also `string` (C++).

Linked-list

Both character termination and length codes limit strings: For example, C character arrays that contain null (NUL) characters cannot be handled directly by C string library functions: Strings using a length code are limited to the maximum value of the length code.

Both of these limitations can be overcome by clever programming, of course, but such workarounds are by definition not standard.

Rough equivalents of the C termination method have historically appeared in both hardware and software. For example, "data processing" machines like the IBM 1401 used a special word mark bit to delimit strings at the left, where the operation would start at the right. This meant that, while the IBM 1401 had a seven-bit word in "reality", almost no-one ever thought to use this as a feature, and override the assignment of the seventh bit to (for example) handle ASCII codes.

It is possible to create data structures and functions that manipulate them that do not have the problems associated with character termination and can in principle overcome length code bounds. It is also possible to optimize the string represented using techniques from run length encoding (replacing repeated characters by the character value and a length) and Hamming encoding.

While these representations are common, others are possible. Using ropes makes certain string operations, such as insertions, deletions, and concatenations more efficient.

Text file strings

In computer readable text files, for example programming language source files or configuration files, strings can be represented. The NUL byte is normally not used as terminator since that does not correspond to the ASCII text standard, and the length is usually not stored, since the file should be human editable without bugs.

Two common representations are:

- Surrounded by quotation marks (ASCII 22_h), used by most programming languages. To be able to include quotation marks, newline characters etc., escape sequences are often available, usually using the backslash character (ASCII 5C_h).
- Terminated by a newline sequence, for example in Windows INI files.

Non-text strings

While character strings are very common uses of strings, a string in computer science may refer generically to any sequence of homogeneously typed data. A string of bits or bytes, for example, may be used to represent non-textual binary data retrieved from a communications medium. This data may or may not be represented by a string-specific datatype, depending on the needs of the application, the desire of the programmer, and the capabilities of the programming language being used. If the programming language's string implementation is not 8-bit clean, data corruption may ensue.

String processing algorithms

There are many algorithms for processing strings, each with various trade-offs. Some categories of algorithms include:

- String searching algorithms for finding a given substring or pattern
- String manipulation algorithms
- Sorting algorithms
- Regular expression algorithms
- Parsing a string
- Sequence mining

Advanced string algorithms often employ complex mechanisms and data structures, among them suffix trees and finite state machines.

Character string-oriented languages and utilities

Character strings are such a useful datatype that several languages have been designed in order to make string processing applications easy to write. Examples include the following languages:

- awk
- Icon
- MUMPS
- Perl
- Rexx
- Ruby
- sed
- SNOBOL
- Tcl

Many Unix utilities perform simple string manipulations and can be used to easily program some powerful string processing algorithms. Files and finite streams may be viewed as strings.

Some APIs like Multimedia Control Interface, embedded SQL or printf use strings to hold commands that will be interpreted.

Recent scripting programming languages, including Perl, Python, Ruby, and Tcl employ regular expressions to facilitate text operations.

Some languages such as Perl and Ruby support string interpolation, which permits arbitrary expressions to be evaluated and included in string literals.

Character string functions

String functions are used to manipulate a string or change or edit the contents of a string. They also are used to query information about a string. They are usually used within the context of a computer programming language.

The most basic example of a string function is the `length(string)` function, which returns the length of a string (not counting any terminator characters or any of the string's internal structural information) and does not modify the string. For example, `length("hello world")` returns 11.

There are many string functions that exist in other languages with similar or exactly the same syntax or parameters. For example, in many languages, the length function is usually represented as `len(string)`. Even though string functions are very useful to a computer programmer, a computer programmer using these functions should be mindful that a string function in one language could in another language behave differently or have a similar or completely different function name, parameters, syntax, and results.

References

- [1] Bryant, Randal E.; David, O'Hallaron (2003), *Computer Systems: A Programmer's Perspective* (<http://csapp.cs.cmu.edu/>) (2003 ed.), Upper Saddle River, NJ: Pearson Education, p. 40, ISBN 0-13-034074-X,

Formal semantics (logic)

In logic, **formal semantics** is the study of the semantics, or interpretations, of formal and (idealizations of) natural languages usually trying to capture the pre-theoretic notion of entailment. (Although both linguistics and logic lay claim to providing theories of natural language, according to Geach, logic generally ignores the "idiotism of idiom", and sees natural languages as cluttered with idioms of no logical interest.)^[1]

A formal language can be defined apart from any interpretation of it. This is done by designating a set of symbols (also called an alphabet) and a set of formation rules (also called a *formal grammar*) which determine which strings of symbols are well-formed formulas. When transformation rules (also called *rules of inference*) are added, and certain sentences are accepted as axioms (together called a deductive system or a *deductive apparatus*) a logical system is formed. An interpretation of a formal language is (roughly) an assignment of meanings to its symbols and truth-conditions to its sentences.^[2]

The truth conditions of various sentences we may encounter in arguments will depend upon their meaning, and so conscientious logicians cannot completely avoid the need to provide some treatment of the meaning of these sentences. The **semantics of logic** refers to the approaches that logicians have introduced to understand and determine that part of meaning in which they are interested; the logician traditionally is not interested in the sentence as uttered but in the proposition, an idealised sentence suitable for logical manipulation.

Until the advent of modern logic, Aristotle's *Organon*, especially *De Interpretatione*, provided the basis for understanding the significance of logic. The introduction of quantification, needed to solve the problem of multiple generality, rendered impossible the kind of subject-predicate analysis that governed Aristotle's account, although there is a renewed interest in term logic, attempting to find calculi in the spirit of Aristotle's syllogistic but with the generality of modern logics based on the quantifier.

The main modern approaches to semantics for formal languages are the following:

- **Model-theoretic semantics** is the archetype of Alfred Tarski's semantic theory of truth, based on his T-schema, and is one of the founding concepts of model theory. This is the most widespread approach, and is based on the idea that the meaning of the various parts of the propositions are given by the possible ways we can give a recursively specified group of interpretation functions from them to some predefined mathematical domains: an interpretation of first-order predicate logic is given by a mapping from terms to a universe of individuals, and a

mapping from propositions to the truth values "true" and "false". Model-theoretic semantics provides the foundations for an approach to the theory of meaning known as Truth-conditional semantics, which was pioneered by Donald Davidson. Kripke semantics introduces innovations, but is broadly in the Tarskian mold.

- **Proof-theoretic semantics** associates the meaning of propositions with the roles that they can play in inferences. Gerhard Gentzen, Dag Prawitz and Michael Dummett are generally seen as the founders of this approach; it is heavily influenced by Ludwig Wittgenstein's later philosophy, especially his aphorism "meaning is use".
- **Truth-value semantics** (also commonly referred to as *substitutional quantification*) was advocated by Ruth Barcan Marcus for modal logics in the early 1960s and later championed by Dunn, Belnap, and Leblanc for standard first-order logic. James Garson has given some results in the areas of adequacy for intensional logics outfitted with such a semantics. The truth conditions for quantified formulas are given purely in terms of truth with no appeal to domains whatsoever (and hence its name *truth-value semantics*).
- **Game-theoretical semantics** has made a resurgence lately mainly due to Jaakko Hintikka for logics of (finite) partially ordered quantification which were originally investigated by Leon Henkin, who studied Henkin quantifiers.
- **Probabilistic semantics** originated from H. Field and has been shown equivalent to and a natural generalization of truth-value semantics. Like truth-value semantics, it is also non-referential in nature.

Notes

[1] Mieszko Talasiewicz (2009). *Philosophy of Syntax - Foundational Topics*. Springer. p. 12. ISBN 978-90-481-3287-4.

[2] The Cambridge Dictionary of Philosophy, *Formal semantics*

Finite-state machine

A **finite-state machine (FSM)** or **finite-state automaton** (plural: *automata*), or simply a **state machine**, is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of *states*. The machine is in only one state at a time; the state it is in at any given time is called the *current state*. It can change from one state to another when initiated by a triggering event or condition, this is called a *transition*. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

The behavior of state machines can be observed in many devices in modern society which perform a predetermined sequence of actions depending on a sequence of events they are presented with. Simple examples are vending machines which dispense products when the proper combination of coins are deposited, elevators which drop riders off at upper floors before going down, traffic lights which change sequence when cars are waiting, and combination locks which require the input of combination numbers in the proper order.

Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, language parsing and other engineering applications. In biology and artificial intelligence research, state machines or hierarchies of state machines have been used to describe neurological systems and in linguistics—to describe the grammars of natural languages.

Considered as an abstract model of computation, the finite state machine is weak; it has less computational power than some other models of computation such as the Turing machine.^[1] That is, there are tasks which no FSM can do but a Turing machine can do. This is because the FSM has limited memory. The memory is limited by the number of states.

FSMs are studied in the more general field of automata theory.

Example: a turnstile

An example of a very simple mechanism that can be modeled by a state machine is a turnstile.^{[2][3]} A turnstile, used to control access to subways and amusement park rides, is a gate with three rotating arms at waist height, one across the entryway. Initially the arms are locked, barring the entry, preventing customers from passing through. Depositing a coin or token in a slot on the turnstile unlocks the arms, allowing them to rotate by one-third of a complete turn, allowing a single customer to push through. After the customer passes through, the arms are locked again until another coin is inserted.

The turnstile has two states: **Locked** and **Unlocked**.^[2] There are two inputs that affect its state: putting a coin in the slot (**coin**) and pushing on the arm (**push**). In the locked state, pushing on the arm has no effect; no matter how many times the input **push** is given it stays in the locked state. Putting a coin in, that is giving the machine a **coin**

input, shifts the state from **Locked** to **Unlocked**. In the unlocked state, putting additional coins in has no effect; that is, giving additional **coin** inputs does not change the state. However, a customer pushing through the arms, giving a **push** input, shifts the state back to **Locked**.

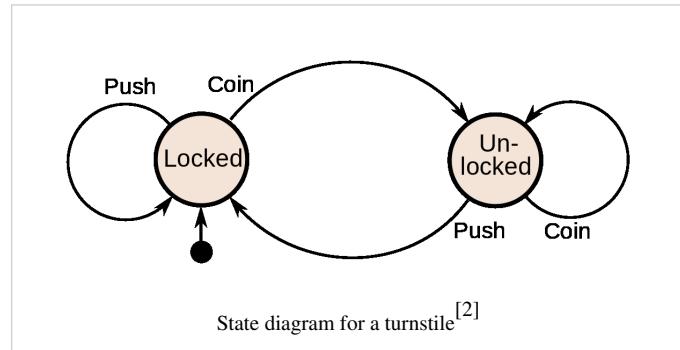
The turnstile state machine can be represented by a state transition table, showing for each state the new state and the output (action) resulting from each input

Current State	Input	Next State	Output
Locked	coin	Unlocked	Release turnstile so customer can push through
	push	Locked	None
Unlocked	coin	Unlocked	None
	push	Locked	When customer has pushed through lock turnstile

It can also be represented by a directed graph called a state diagram (*above*). Each of the states is represented by a node (*circle*). Edges (*arrows*) show the transitions from one state to another. Each arrow is labeled with the input that triggers that transition. Inputs that don't cause a change of state (such as a **coin** input in the **Unlocked** state) are represented by a circular arrow returning to the original state. The arrow into the **Locked** node from the black dot indicates it is the initial state.

Concepts and vocabulary

A *state* is a description of the status of a system that is waiting to execute a *transition*. A transition is a set of actions to be executed when a condition is fulfilled or when an event is received. For example, when using an audio system to listen to the radio (the system is in the "radio" state), receiving a "next" stimulus results in moving to the next station. When the system is in the "CD" state, the "next" stimulus results in moving to the next track. Identical stimuli trigger different actions depending on the current state.



In some finite-state machine representations, it is also possible to associate actions with a state:

- Entry action: performed *when entering* the state,
- Exit action: performed *when exiting* the state.

Representations

For an introduction, see *State diagram*.

State/Event table

Several state transition table types are used. The most common representation is shown below: the combination of current state (e.g. B) and input (e.g. Y) shows the next state (e.g. C). The complete actions information is not directly described in the table and can only be added using footnotes. A FSM definition including the full actions information is possible using state tables (see also VFSM).

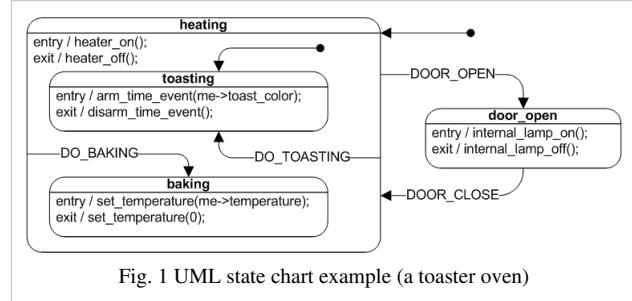


Fig. 1 UML state chart example (a toaster oven)

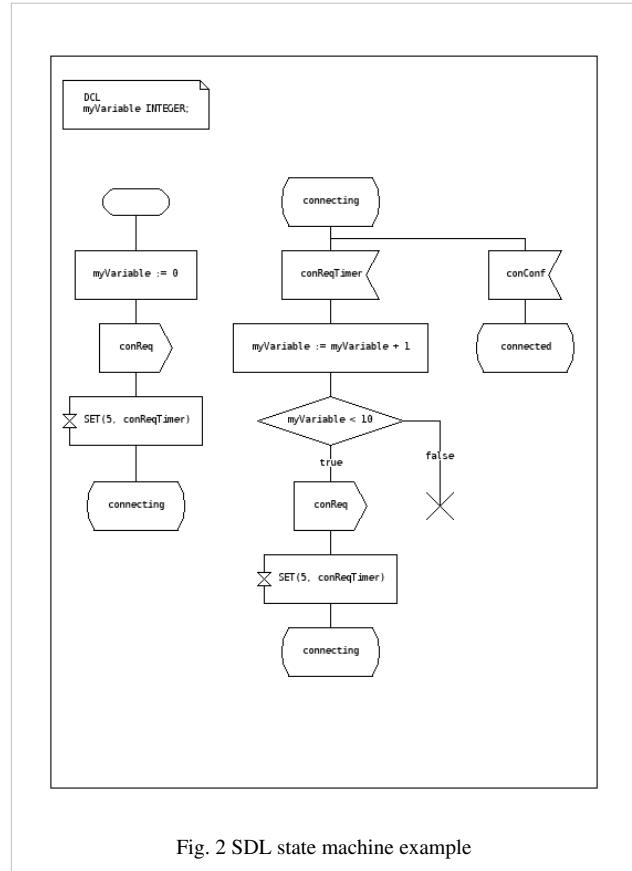
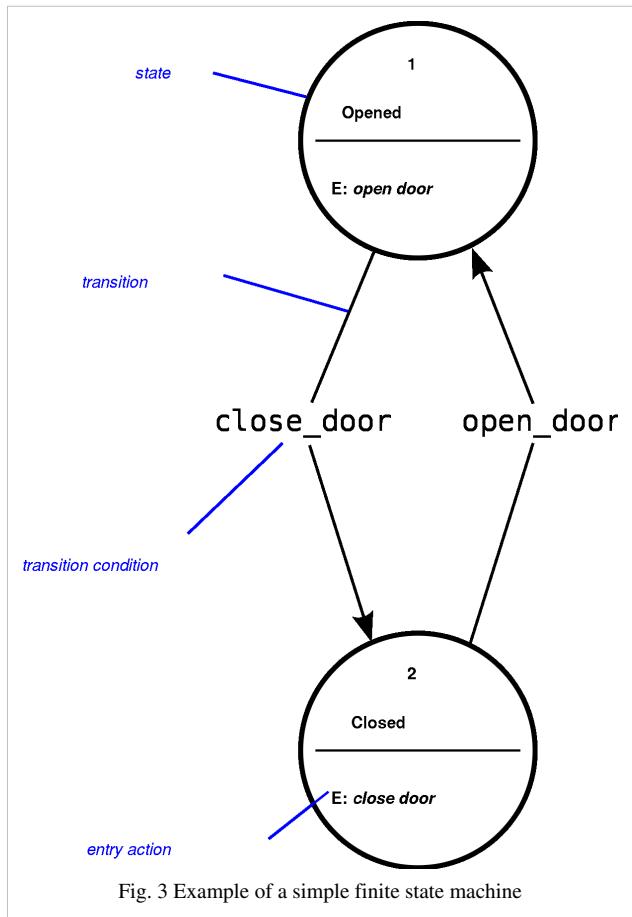


Fig. 2 SDL state machine example



State transition table

Current state → Input ↓	State A	State B	State C
Input X
Input Y	...	State C	...
Input Z

UML state machines

The Unified Modeling Language has a notation for describing state machines. UML state machines overcome the limitations of traditional finite state machines while retaining their main benefits. UML state machines introduce the new concepts of hierarchically nested states and orthogonal regions, while extending the notion of actions. UML state machines have the characteristics of both Mealy machines and Moore machines. They support actions that depend on both the state of the system and the triggering event, as in Mealy machines, as well as entry and exit actions, which are associated with states rather than transitions, as in Moore machines.

SDL state machines

The Specification and Description Language is a standard from ITU which includes graphical symbols to describe actions in the transition:

- send an event
- receive an event
- start a timer
- cancel a timer
- start another concurrent state machine
- decision

SDL embeds basic data types called Abstract Data Types, an action language, and an execution semantic in order to make the finite state machine executable.

Other state diagrams

There are a large number of variants to represent an FSM such as the one in figure 3.

Usage

In addition to their use in modeling reactive systems presented here, finite state automata are significant in many different areas, including electrical engineering, linguistics, computer science, philosophy, biology, mathematics, and logic. Finite state machines are a class of automata studied in automata theory and the theory of computation. In computer science, finite state machines are widely used in modeling of application behavior, design of hardware digital systems, software engineering, compilers, network protocols, and the study of computation and languages.

Classification

The state machines can be subdivided into Transducers, Acceptors, Classifiers and Sequencers.^[4]

Acceptors and recognizers

Acceptors (also **recognizers** and **sequence detectors**) produce a binary output, saying either *yes* or *no* to answer whether the input is accepted by the machine or not. All states of the FSM are said to be either accepting or not accepting. At the time when all input is processed, if the current state is an accepting state, the input is accepted; otherwise it is rejected. As a rule the input are symbols (characters); actions are not used. The example in figure 4 shows a finite state machine which accepts the word "nice". In this FSM the only accepting state is number 7.

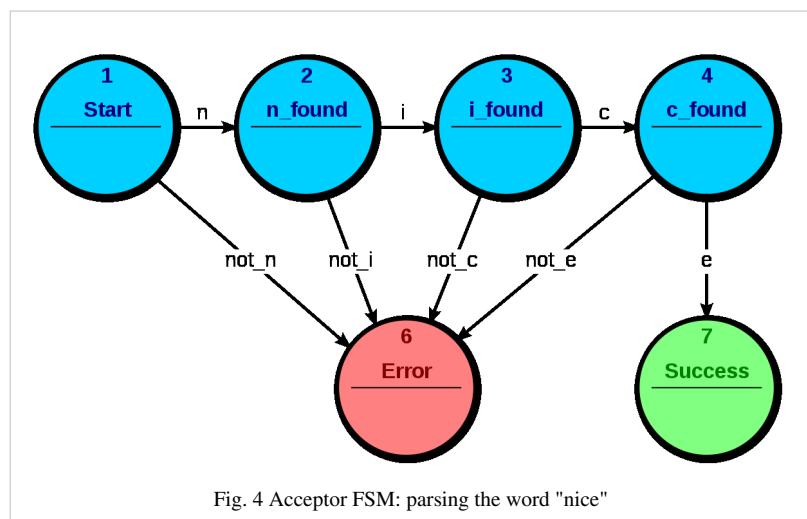


Fig. 4 Acceptor FSM: parsing the word "nice"

The machine can also be described as defining a language, which would contain every word accepted by the machine but none of the rejected ones; we say then that the language is *accepted* by the machine. By definition, the languages accepted by FSMs are the regular languages—that is, a language is regular if there is some FSM that accepts it.

Start state

The start state is usually shown drawn with an arrow "pointing at it from anywhere" (Sipser (2006) p. 34).

Accept (or final) states

Accept states (also referred to as **accepting** or **final** states) are those at which the machine reports that the input string, as processed so far, is a member of the language it accepts. It is usually represented by a double circle.

An example of an accepting state appears in the diagram to the right: a deterministic finite automaton (DFA) that detects whether the binary input string contains an even number of 0's.

S_1 (which is also the start state) indicates the state at which an even number of 0's has been input. S_1 is therefore an accepting state. This machine will finish in an accept state, if the binary string contains an even number of 0's (including any binary string containing no 0's). Examples of strings accepted by this DFA are epsilon (the empty string), 1, 11, 11..., 00, 010, 1010, 10110, etc...

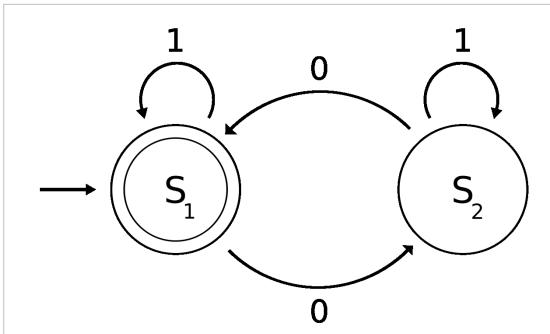


Fig. 5: Representation of a finite-state machine; this example shows one that determines whether a binary number has an odd or even number of 0's, where S_1 is an **accepting state**.

Classifier is a generalization that similarly to acceptor produces single output when terminates but has more than two terminal states.

Transducers

Transducers generate output based on a given input and/or a state using actions. They are used for control applications and in the field of computational linguistics. Here two types are distinguished:

Moore machine

The FSM uses only entry actions, i.e., output depends only on the state. The advantage of the Moore model is a simplification of the behaviour. Consider an elevator door. The state machine recognizes two commands: "command_open" and "command_close" which trigger state changes. The entry action (E:) in state "Opening" starts a motor opening the door, the entry action in state "Closing" starts a motor in the other direction closing the door. States "Opened" and "Closed" stop the motor when fully opened or closed. They signal to the outside world (e.g., to other state machines) the situation: "door is open" or "door is closed".

Mealy machine

The FSM uses only input actions, i.e., output depends on input and state. The use of a Mealy FSM leads often to a reduction of the number of states. The example in figure 7 shows a Mealy FSM implementing the same behaviour as in the Moore example (the behaviour depends on the implemented FSM execution model and will work, e.g., for virtual FSM but not for event driven FSM). There are two input actions (I:): "start motor to close the door if

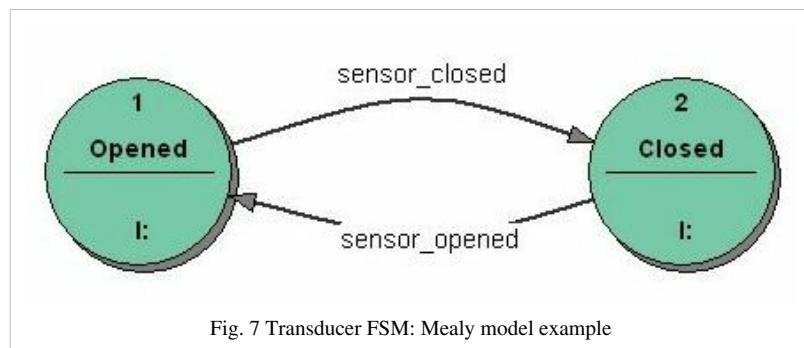


Fig. 7 Transducer FSM: Mealy model example

"command_close arrives" and "start motor in the other direction to open the door if command_open arrives".

The "opening" and "closing" intermediate states are not shown.

In practice^[of what?] mixed models are often used.

More details about the differences and usage of Moore and Mealy models, including an executable example, can be found in the external technical note "Moore or Mealy model?"^[5]

Generators

The **sequencers** or **generators** are a subclass of aforementioned types that have a single-letter input alphabet. They produce only one sequence, which can be interpreted as output sequence of transducer or classifier outputs.

Determinism

A further distinction is between **deterministic** (DFA) and **non-deterministic** (NFA, GNFA) automata. In deterministic automata, every state has exactly one transition for each possible input. In non-deterministic automata, an input can lead to one, more than one or no transition for a given state. This distinction is relevant in practice, but not in theory, as there exists an algorithm (the powerset construction) which can transform any NFA into a more complex DFA with identical functionality.

The FSM with only one state is called a combinatorial FSM and uses only input actions. This concept is useful in cases where a number of FSM are required to work together, and where it is convenient to consider a purely combinatorial part as a form of FSM to suit the design tools.

Alternative semantics

There are other sets of semantics available to represent state machines. For example, there are tools for modeling and designing logic for embedded controllers.^[6] They combine hierarchical state machines, flow graphs, and truth tables into one language, resulting in a different formalism and set of semantics.^[7] Figure 8 illustrates this mix of state machines and flow graphs with a set of states to represent the state of a stopwatch and a flow graph to control the ticks of the watch. These charts, like Harel's original state machines,^[8] support hierarchically nested states, orthogonal regions, state actions, and transition actions.^[9]

FSM logic

The next state and output of an FSM is a function of the input and of the current state. The FSM logic is shown in Figure 8.

Mathematical model

In accordance with the general classification, the following formal definitions are found:

- A *deterministic finite state machine* or *acceptor deterministic finite state machine* is a quintuple $(\Sigma, S, s_0, \delta, F)$, where:
 - Σ is the input alphabet (a finite, non-empty set of symbols).
 - S is a finite, non-empty set of states.
 - s_0 is an initial state, an element of S .
 - δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$ (in a nondeterministic finite automaton it would be $\delta : S \times \Sigma \rightarrow \mathcal{P}(S)$, i.e., δ would return a set of states).
 - F is the set of final states, a (possibly empty) subset of S .

For both deterministic and non-deterministic FSMs, it is conventional to allow δ to be a partial function, i.e. $\delta(q, x)$ does not have to be defined for every combination of $q \in S$ and $x \in \Sigma$. If an FSM M is in a state q , the next symbol is x and $\delta(q, x)$ is not defined, then M can announce an error (i.e. reject the input). This is useful in definitions of general state machines, but less useful when transforming the machine. Some algorithms in their default form may require total functions.

A finite-state machine is a restricted Turing machine where the head can only perform "read" operations, and always moves from left to right.^[10]

- A *finite state transducer* is a sextuple $(\Sigma, \Gamma, S, s_0, \delta, \omega)$, where:
 - Σ is the input alphabet (a finite non empty set of symbols).
 - Γ is the output alphabet (a finite, non-empty set of symbols).
 - S is a finite, non-empty set of states.
 - s_0 is the initial state, an element of S . In a nondeterministic finite automaton, s_0 is a set of initial states.
 - δ is the state-transition function: $\delta : S \times \Sigma \rightarrow S$.
 - ω is the output function.

If the output function is a function of a state and input alphabet ($\omega : S \times \Sigma \rightarrow \Gamma$) that definition corresponds to the **Mealy model**, and can be modelled as a Mealy machine. If the output function depends only on a state ($\omega : S \rightarrow \Gamma$) that definition corresponds to the **Moore model**, and can be modelled as a Moore machine. A finite-state machine with no output function at all is known as a semiautomaton or transition system.

If we disregard the first output symbol of a Moore machine, $\omega(s_0)$, then it can be readily converted to an output-equivalent Mealy machine by setting the output function of every Mealy transition (i.e. labeling every edge) with the output symbol given of the destination Moore state. The converse transformation is less straightforward because a Mealy machine state may have different output labels on its incoming transitions (edges). Every such state needs to be split in multiple Moore machine states, one for every incident output symbol.^[11]

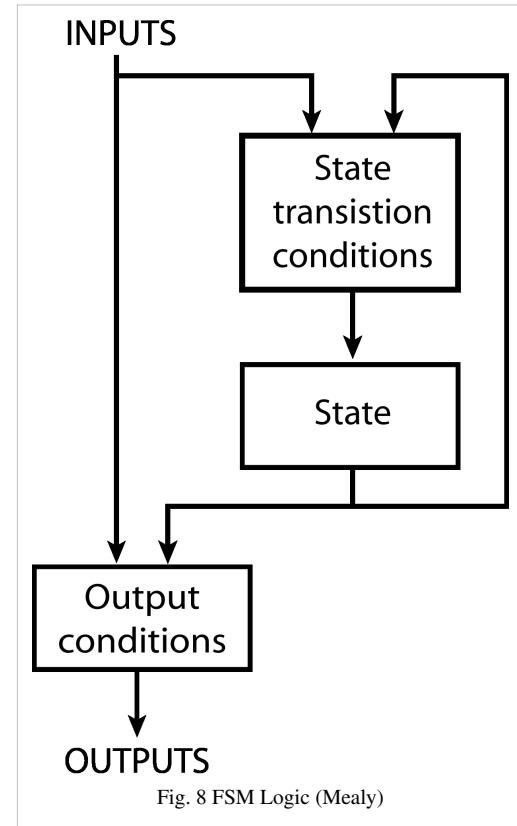


Fig. 8 FSM Logic (Mealy)

Optimization

Optimizing an FSM means finding the machine with the minimum number of states that performs the same function. The fastest known algorithm doing this is the Hopcroft minimization algorithm.^{[12][13]} Other techniques include using an implication table, or the Moore reduction procedure. Additionally, acyclic FSAs can be minimized in linear time Revuz (1992).^[14]

Implementation

Hardware applications

In a digital circuit, an FSM may be built using a programmable logic device, a programmable logic controller, logic gates and flip flops or relays. More specifically, a hardware implementation requires a register to store state variables, a block of combinational logic which determines the state transition, and a second block of combinational logic that determines the output of an FSM. One of the classic hardware implementations is the Richards controller.

A particular case of Moore FSM, when output is directly connected to the state flip-flops, that is when output function is simple identity, is known as Medvedev FSM.^[15] It is advised in chip design that no logic is placed between primary I/O and registers to minimize interchip delays, which are usually long and limit the FSM frequencies.

Software applications

The following concepts are commonly used to build software applications with finite state machines:

- Automata-based programming
- Event-driven FSM
- Virtual FSM (VFSM)

References

- [1] Belzer, Jack; Albert George Holzman, Allen Kent (1975). *Encyclopedia of Computer Science and Technology*, Vol. 25 (<http://books.google.com/books?id=W2YLBIdelIEC&printsec=frontcover&dq=%22finite+state+machine%22&source=bl&ots=7lcYvqsRvY&sig=19Ez1HGwvAm2HdaWYmbKDvod3Ec&hl=en&sa=X&ei=auwAUIWOfoezqQGglNGxDA&ved=0CFsQ6AEwBg#v=onepage&q=%22finite+state+machine%22&f=false>). USA: CRC Press. pp. 73. ISBN 0824722752..
- [2] Koshy, Thomas (2004). *Discrete Mathematics With Applications* (http://books.google.com/books?id=90KApidK5NwC&pg=PA762&dq=state+machine+turnstile&hl=en&sa=X&ei=4s70T_HtKlj1qAGr1cjbAw&ved=0CFgQ6AEwBA#v=onepage&q=state+machine+turnstile&f=false). Academic Press. pp. 762. ISBN 0124211801..
- [3] Wright, David R. (2005). "Finite State Machines" (<http://www4.ncsu.edu/~drwright3/docs/courses/csc216/fsm-notes.pdf>). *CSC215 Class Notes*. Prof. David R. Wright website, N. Carolina State Univ... Retrieved July 14, 2012.
- [4] M. Keller, Robert (2001). "Classifiers, Acceptors, Transducers, and Sequencers" (http://www.cs.hmc.edu/~keller/cs60book/12_Finite-State-Machines.pdf). *Computer Science: Abstraction to Implementation* (<http://www.cs.hmc.edu/~keller/cs60book/All.pdf>). Harvey Mudd College. p. 480..
- [5] <http://www.stateworks.com/technology/TN10-Moore-Or-Mealy-Model/>
- [6] Tiwari, A. (2002). Formal Semantics and Analysis Methods for Simulink Stateflow Models. (<http://www.csl.sri.com/users/tiwari/papers/stateflow.pdf>)
- [7] Hamon, G. (2005). "A Denotational Semantics for Stateflow". International Conference on Embedded Software. Jersey City, NJ: ACM. pp. 164–172. CiteSeerX: 10.1.1.89.8817 (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.8817>).

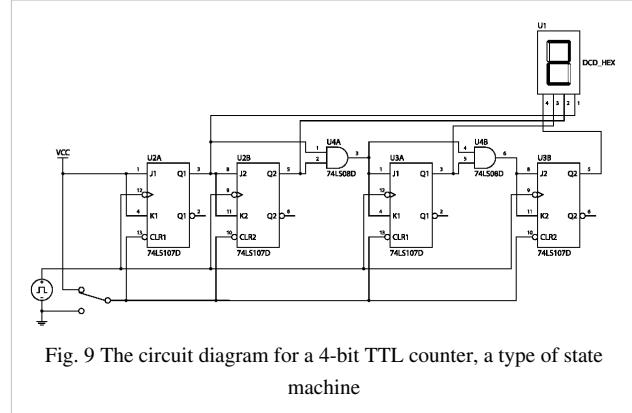


Fig. 9 The circuit diagram for a 4-bit TTL counter, a type of state machine

- [8] Harel, D. (1987). A Visual Formalism for Complex Systems. *Science of Computer Programming* , 231–274. (<http://www.fceia.unr.edu.ar/asist/harel01.pdf>)
- [9] Alur, R., Kanade, A., Ramesh, S., & Shashidhar, K. C. (2008). Symbolic analysis for improving simulation coverage of Simulink/Stateflow models. *Internation Conference on Embedded Software* (pp. 89–98). Atlanta, GA: ACM. (http://drona.csa.iisc.ernet.in/~kanade/publications/symbolic_analysis_for_improving_simulation_coverage_of_simulink_stateflow_models.pdf)
- [10] Black, Paul E (12 May 2008). "Finite State Machine" (<http://www.nist.gov/dads/HTML/finiteStateMachine.html>). *Dictionary of Algorithms and Data Structures* (U.S. National Institute of Standards and Technology).
- [11] James Andrew Anderson; Thomas J. Head (2006). *Automata theory with modern applications* (<http://books.google.com/books?id=ikS8BLdLDxIC&pg=PA105>). Cambridge University Press. pp. 105–108. ISBN 978-0-521-84887-9.
- [12] Hopcroft, John E (1971). An $n \log n$ algorithm for minimizing states in a finite automaton (<ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/71/190/CS-TR-71-190.pdf>)
- [13] Almeida, Marco; Moreira, Nelma; Reis, Rogerio (2007). On the performance of automata minimization algorithms. (<http://www.dcc.fc.up.pt/dcc/Pubs/TReports/TR07/dcc-2007-03.pdf>)
- [14] Revuz D. Minimization of Acyclic automata in Linear Time. *Theoretical Computer Science* 92 (1992) 181-189 181 Elsevier
- [15] Kaeslin, Hubert (2008). "Mealy, Moore, Medvedev-type and combinatorial output bits" (http://books.google.ee/books?id=gdRStcYgf2oC&pg=PA787&dq=medvedev+fsm&hl=en&sa=X&ei=qTG1UKfgIq344QTqhYHgDA&redir_esc=y#v=onepage&q=medvedev fsm&f=false). *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*. Cambridge University Press. p. 787..

Further reading

General

- Wagner, F., "Modeling Software with Finite State Machines: A Practical Approach", Auerbach Publications, 2006, ISBN 0-8493-8086-3.
- ITU-T, *Recommendation Z.100 Specification and Description Language (SDL)* (<http://www.itu.int/rec/T-REC-Z.100-200711-I/en>)
- Samek, M., *Practical Statecharts in C/C++* (<http://www.state-machine.com/psicc/index.php>), CMP Books, 2002, ISBN 1-57820-110-1.
- Samek, M., *Practical UML Statecharts in C/C++, 2nd Edition* (<http://www.state-machine.com/psicc2/index.php>), Newnes, 2008, ISBN 0-7506-8706-1.
- Gardner, T., *Advanced State Management* (<http://www.troyworks.com/cogs/>), 2007
- Cassandras, C., Lafourche, S., "Introduction to Discrete Event Systems". Kluwer, 1999, ISBN 0-7923-8609-4.
- Timothy Kam, *Synthesis of Finite State Machines: Functional Optimization*. Kluwer Academic Publishers, Boston 1997, ISBN 0-7923-9842-4
- Tiziano Villa, *Synthesis of Finite State Machines: Logic Optimization*. Kluwer Academic Publishers, Boston 1997, ISBN 0-7923-9892-0
- Carroll, J., Long, D., *Theory of Finite Automata with an Introduction to Formal Languages*. Prentice Hall, Englewood Cliffs, 1989.
- Kohavi, Z., *Switching and Finite Automata Theory*. McGraw-Hill, 1978.
- Gill, A., *Introduction to the Theory of Finite-state Machines*. McGraw-Hill, 1962.
- Ginsburg, S., *An Introduction to Mathematical Machine Theory*. Addison-Wesley, 1962.

Finite state machines (automata theory) in theoretical computer science

- Arbib, Michael A. (1969). *Theories of Abstract Automata* (1st ed.). Englewood Cliffs, N.J.: Prentice-Hall, Inc.. ISBN 0-13-913368-2.
- Bobrow, Leonard S.; Michael A. Arbib (1974). *Discrete Mathematics: Applied Algebra for Computer and Information Science* (1st ed.). Philadelphia: W. B. Saunders Company, Inc.. ISBN 0-7216-1768-9.
- Booth, Taylor L. (1967). *Sequential Machines and Automata Theory* (1st ed.). New York: John Wiley and Sons, Inc.. Library of Congress Card Catalog Number 67-25924.
- Boolos, George; Richard Jeffrey (1989, 1999). *Computability and Logic* (3rd ed.). Cambridge, England: Cambridge University Press. ISBN 0-521-20402-X.
- Brookshear, J. Glenn (1989). *Theory of Computation: Formal Languages, Automata, and Complexity*. Redwood City, California: Benjamin/Cummings Publish Company, Inc.. ISBN 0-8053-0143-7.
- Davis, Martin; Ron Sigal, Elaine J. Weyuker (1994). *Computability, Complexity, and Languages and Logic: Fundamentals of Theoretical Computer Science* (2nd ed.). San Diego: Academic Press, Harcourt, Brace & Company. ISBN 0-12-206382-1.
- Hopcroft, John; Jeffrey Ullman (1979). *Introduction to Automata Theory, Languages, and Computation* (1st ed.). Reading Mass: Addison-Wesley. ISBN 0-201-02988-X.
- Hopcroft, John E.; Rajeev Motwani, Jeffrey D. Ullman (2001). *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Reading Mass: Addison-Wesley. ISBN 0-201-44124-1.
- Hopkin, David; Barbara Moss (1976). *Automata*. New York: Elsevier North-Holland. ISBN 0-444-00249-9.
- Kozen, Dexter C. (1997). *Automata and Computability* (1st ed.). New York: Springer-Verlag. ISBN 0-387-94907-0.
- Lewis, Harry R.; Christos H. Papadimitriou (1998). *Elements of the Theory of Computation* (2nd ed.). Upper Saddle River, New Jersey: Prentice-Hall. ISBN 0-13-262478-8.
- Linz, Peter (2006). *Formal Languages and Automata* (4th ed.). Sudbury, MA: Jones and Bartlett. ISBN 978-0-7637-3798-6.
- Minsky, Marvin (1967). *Computation: Finite and Infinite Machines* (1st ed.). New Jersey: Prentice-Hall.
- Christos Papadimitriou (1993). *Computational Complexity* (1st ed.). Addison Wesley. ISBN 0-201-53082-1.
- Pippenger, Nicholas (1997). *Theories of Computability* (1st ed.). Cambridge, England: Cambridge University Press. ISBN 0-521-55380-6 (hc).
- Rodger, Susan; Thomas Finley (2006). *JFLAP: An Interactive Formal Languages and Automata Package* (1st ed.). Sudbury, MA: Jones and Bartlett. ISBN 0-7637-3834-4.
- Sipser, Michael (2006). *Introduction to the Theory of Computation* (2nd ed.). Boston Mass: Thomson Course Technology. ISBN 0-534-95097-3.
- Wood, Derick (1987). *Theory of Computation* (1st ed.). New York: Harper & Row, Publishers, Inc.. ISBN 0-06-047208-1.

Abstract state machines in theoretical computer science

- Yuri Gurevich (2000), *Sequential Abstract State Machines Capture Sequential Algorithms*, ACM Transactions on Computational Logic, v1. no. 1 (July 2000), pages 77–111. <http://research.microsoft.com/~gurevich/Opera/141.pdf>

Machine learning using finite-state algorithms

- Mitchell, Tom M. (1997). *Machine Learning* (1st ed.). New York: WCB/McGraw-Hill Corporation. ISBN 0-07-042807-7.

Hardware engineering: state minimization and synthesis of sequential circuits

- Booth, Taylor L. (1967). *Sequential Machines and Automata Theory* (1st ed.). New York: John Wiley and Sons, Inc.. Library of Congress Card Catalog Number 67-25924.
- Booth, Taylor L. (1971). *Digital Networks and Computer Systems* (1st ed.). New York: John Wiley and Sons, Inc.. ISBN 0-471-08840-4.
- McCluskey, E. J. (1965). *Introduction to the Theory of Switching Circuits* (1st ed.). New York: McGraw-Hill Book Company, Inc.. Library of Congress Card Catalog Number 65-17394.
- Hill, Fredrick J.; Gerald R. Peterson (1965). *Introduction to the Theory of Switching Circuits* (1st ed.). New York: McGraw-Hill Book Company. Library of Congress Card Catalog Number 65-17394.

Finite Markov chain processes

"We may think of a Markov chain as a process that moves successively through a set of states s_1, s_2, \dots, s_r if it is in state s_i it moves on to the next stop to state s_j with probability p_{ij} . These probabilities can be exhibited in the form of a transition matrix" (Kemeny (1959), p. 384)

Finite Markov-chain processes are also known as subshifts of finite type.

- Booth, Taylor L. (1967). *Sequential Machines and Automata Theory* (1st ed.). New York: John Wiley and Sons, Inc.. Library of Congress Card Catalog Number 67-25924.
- Kemeny, John G.; Hazleton Mirkil, J. Laurie Snell, Gerald L. Thompson (1959). *Finite Mathematical Structures* (1st ed.). Englewood Cliffs, N.J.: Prentice-Hall, Inc.. Library of Congress Card Catalog Number 59-12841. Chapter 6 "Finite Markov Chains".

External links

- Modeling a Simple AI behavior using a Finite State Machine (<http://blog.manuvra.com/modeling-a-simple-ai-behavior-using-a-finite-state-machine/>) Example of usage in Video Games
- Free On-Line Dictionary of Computing (<http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=finite+state+machine>) description of Finite State Machines
- NIST Dictionary of Algorithms and Data Structures (<http://www.nist.gov/dads/HTML/finiteStateMachine.html>) description of Finite State Machines
- Video lecture (<http://video.google.com/videoplay?docid=-5837841629284334824>)

Automata theory

In theoretical computer science, **automata theory** is the study of mathematical objects called *abstract machines* or *automata* and the computational problems that can be solved using them. *Automata* comes from the Greek word αὐτόματο meaning "self-acting".

The figure at right illustrates a finite state machine, which belongs to one well-known variety of automaton. This automaton consists of states (represented in the figure by circles), and transitions (represented by arrows). As the automaton sees a symbol of input, it makes a *transition* (or *jump*) to another state, according to its *transition function* (which takes the current state and the recent symbol as its inputs).

Automata theory is also closely related to formal language theory. An automaton is a finite representation of a formal language that may be an infinite set. Automata are often classified by the class of formal languages they are able to recognize.

Automata play a major role in theory of computation, compiler design, parsing and formal verification.

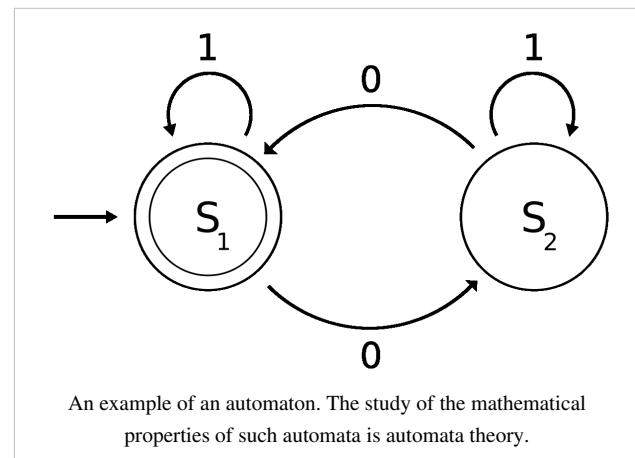
Automata

Following is an introductory definition of one type of automaton, which attempts to help one grasp the essential concepts involved in automata theory.

Informal description

An automaton is supposed to *run* on some given sequence of *inputs* in discrete time steps. At each time step, an automaton gets one input that is picked up from a set of *symbols* or *letters*, which is called an *alphabet*. At any time, the symbols so far fed to the automaton as input form a finite sequence of symbols, which is called a *word*. An automaton contains a finite set of *states*. At each instance in time of some run, the automaton is *in* one of its states. At each time step when the automaton reads a symbol, it *jumps* or *transits* to a next state that is decided by a function that takes current state and the symbol currently read as parameters. This function is called *transition function*. The automaton *reads* the symbols of the input word one after another and *transits* from state to state according to the transition function, until the word is read completely. Once the input word has been read, the automaton is said to have been *stopped* and the state at which automaton has stopped is called *final state*. Depending on the final state, it's said that the automaton either *accepts* or *rejects* an input word. There is a subset of states of the automaton, which is defined as the set of *accepting states*. If the final state is an accepting state, then the automaton *accepts* the word. Otherwise, the word is *rejected*. The set of all the words accepted by an automaton is called the *language recognized by the automaton*.

In short, an automaton is a mathematical object that takes a word as input and decides either to accept it or reject it. Since all computational problems are reducible into the accept/reject question on words (all problem instances can be represented in a finite length of symbols), automata theory plays a crucial role in computational theory.



Formal definition

Automaton

An **automaton** is represented formally by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of *states*.
- Σ is a finite set of *symbols*, called the *alphabet* of the automaton.
- δ is the **transition function**, that is, $\delta: Q \times \Sigma \rightarrow Q$.
- q_0 is the *start state*, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
- F is a set of states of Q (i.e. $F \subseteq Q$) called **accept states**.

Input word

An automaton reads a finite string of symbols a_1, a_2, \dots, a_n , where $a_i \in \Sigma$, which is called an *input word*. The set of all words is denoted by Σ^* .

Run

A sequence of states $q_0, q_1, q_2, \dots, q_n$, where $q_i \in Q$ such that q_0 is the start state and $q_i = \delta(q_{i-1}, a_i)$ for $0 < i \leq n$, is a *run* of the automaton on an input word $w = a_1, a_2, \dots, a_n \in \Sigma^*$. In other words, at first the automaton is at the start state q_0 , and then the automaton reads symbols of the input word in sequence. When the automaton reads symbol a_i it jumps to state $q_i = \delta(q_{i-1}, a_i)$. q_n is said to be the *final state* of the run.

Accepting word

A word $w \in \Sigma^*$ is accepted by the automaton if $q_n \in F$.

Recognized language

An automaton can recognize a formal language. The language $L \subseteq \Sigma^*$ recognized by an automaton is the set of all the words that are accepted by the automaton.

Recognizable languages

The recognizable languages are the set of languages that are recognized by some automaton. For the above definition of automata the recognizable languages are regular languages. For different definitions of automata, the recognizable languages are different.

Variant definitions of automata

Automata are defined to study useful machines under mathematical formalism. So, the definition of an automaton is open to variations according to the "real world machine", which we want to model using the automaton. People have studied many variations of automata. The most standard variant, which is described above, is called a deterministic finite automaton. The following are some popular variations in the definition of different components of automata.

Input

- *Finite input*: An automaton that accepts only finite sequence of symbols. The above introductory definition only encompasses finite words.
- *Infinite input*: An automaton that accepts infinite words (ω -words). Such automata are called ω -*automata*.
- *Tree word input*: The input may be a *tree of symbols* instead of sequence of symbols. In this case after reading each symbol, the automaton *reads* all the successor symbols in the input tree. It is said that the automaton *makes one copy* of itself for each successor and each such copy starts running on one of the successor symbol from the state according to the transition relation of the automaton. Such an automaton is called tree automaton.
- *Infinite tree input* : The two extensions above can be combined, so the automaton reads a tree structure with (in)finite branches. Such an automaton is called infinite tree automaton

States

- *Finite states*: An automaton that contains only a finite number of states. The above introductory definition describes automata with finite numbers of states.
- *Infinite states*: An automaton that may not have a finite number of states, or even a countable number of states. For example, the quantum finite automaton or topological automaton has uncountable infinity of states.
- *Stack memory*: An automaton may also contain some extra memory in the form of a stack in which symbols can be pushed and popped. This kind of automaton is called a *pushdown automaton*

Transition function

- *Deterministic*: For a given current state and an input symbol, if an automaton can only jump to one and only one state then it is a *deterministic automaton*.
- *Nondeterministic*: An automaton that, after reading an input symbol, may jump into any of a number of states, as licensed by its transition relation. Notice that the term transition function is replaced by transition relation: The automaton *non-deterministically* decides to jump into one of the allowed choices. Such automata are called *nondeterministic automata*.
- *Alternation*: This idea is quite similar to tree automaton, but orthogonal. The automaton may run its *multiple copies* on the *same* next read symbol. Such automata are called *alternating automata*. Acceptance condition must satisfy all runs of such *copies* to accept the input.

Acceptance condition

- *Acceptance of finite words*: Same as described in the informal definition above.
- *Acceptance of infinite words*: an *omega automaton* cannot have final states, as infinite words never terminate. Rather, acceptance of the word is decided by looking at the infinite sequence of visited states during the run.
- *Probabilistic acceptance*: An automaton need not strictly accept or reject an input. It may accept the input with some probability between zero and one. For example, quantum finite automaton, geometric automaton and *metric automaton* have probabilistic acceptance.

Different combinations of the above variations produce many classes of automaton.

Automata theory

Automata theory is a subject matter that studies properties of various types of automata. For example, the following questions are studied about a given type of automata.

- Which class of formal languages is recognizable by some type of automata? (Recognizable languages)
- Are certain automata *closed* under union, intersection, or complementation of formal languages? (Closure properties)
- How much is a type of automata expressive in terms of recognizing class of formal languages? And, their relative expressive power? (Language Hierarchy)

Automata theory also studies if there exist any effective algorithm or not to solve problems similar to the following list.

- Does an automaton accept any input word? (emptiness checking)
- Is it possible to transform a given non-deterministic automaton into deterministic automaton without changing the recognizable language? (Determinization)
- For a given formal language, what is the smallest automaton that recognizes it? (Minimization).

Classes of automata

The following is an incomplete list of types of automata.

Automaton	Recognizable language
Nondeterministic/Deterministic Finite state machine (FSM)	regular languages
Deterministic pushdown automaton (DPDA)	deterministic context-free languages
Pushdown automaton (PDA)	context-free languages
Linear bounded automaton (LBA)	context-sensitive languages
Turing machine	recursively enumerable languages
Deterministic Büchi automaton	ω -limit languages
Nondeterministic Büchi automaton	ω -regular languages
Rabin automaton, Streett automaton, Parity automaton, Muller automaton	ω -regular languages

Discrete, continuous, and hybrid automata

Normally automata theory describes the states of abstract machines but there are analog automata or continuous automata or hybrid discrete-continuous automata, which use analog data, continuous time, or both.

Applications

Each model in automata theory plays important roles in several applied areas. Finite automata are used in text processing, compilers, and hardware design. Context-free grammar (CFGs) are used in programming languages and artificial intelligence. Originally, CFGs were used in the study of the human languages. Cellular automata are used in the field of biology, the most common example being John Conway's Game of Life. Some other examples which could be explained using automata theory in biology include mollusk and pine cones growth and pigmentation patterns. Going further, a theory suggesting that the whole universe is computed by some sort of a discrete automaton, is advocated by some scientists. The idea originated in the work of Konrad Zuse, and was popularized in America by Edward Fredkin.

Automata Simulators

Automata simulators are pedagogical tools used to teach, learn and research automata theory. An automata simulator takes as input the description of an automaton and then simulates its working for an arbitrary input string. The description of the automaton can be entered in several ways. An automaton can be defined in a symbolic language or its specification may be entered in a predesigned form or its transition diagram may be drawn by clicking and dragging the mouse. Well known automata simulators include Turing's World, JFLAP, VAS, TAGS and SimStudio.^[1]

Connection to Category theory

One can define several distinct categories of automata^[2] following the automata classification into different types described in the previous section. The mathematical category of deterministic automata, sequential machines or *sequential automata*, and Turing machines with automata homomorphisms defining the arrows between automata is a Cartesian closed category,^{[3][4]} it has both categorical limits and colimits. An automata homomorphism maps a quintuple of an automaton A_i onto the quintuple of another automaton A_j .^[5] Automata homomorphisms can also be considered as *automata transformations* or as semigroup homomorphisms, when the state space, S , of the automaton is defined as a semigroup S_g . Monoids are also considered as a suitable setting for automata in monoidal

categories.^{[6][7][8]}

Categories of variable automata

One could also define a *variable automaton*, in the sense of Norbert Wiener in his book on "*Human Use of Human Beings*" via the endomorphisms $A_i \rightarrow A_i$. Then, one can show that such variable automata homomorphisms form a mathematical group. In the case of non-deterministic, or other complex kinds of automata, the latter set of endomorphisms may become, however, a *variable automaton groupoid*. Therefore, in the most general case, categories of variable automata of any kind are categories of groupoids^[9] or groupoid categories. Moreover, the category of reversible automata is then a 2-category, and also a subcategory of the 2-category of groupoids, or the groupoid category.

References

- [1] Chakraborty, P., Saxena, P. C., Katti, C. P. 2011. Fifty Years of Automata Simulation: A Review. *ACM Inroads*, **2**(4):59–70. <http://dl.acm.org/citation.cfm?id=2038893&dl=ACM&coll=DL&CFID=65021406&CFTOKEN=86634854>
- [2] Jirí Adámek and Vera Trnková. 1990. *Automata and Algebras in Categories*. Kluwer Academic Publishers:Dordrecht and Prague
- [3] S. Mac Lane, Categories for the Working Mathematician, Springer, New York (1971)
- [4] <http://planetmath.org/encyclopedia/CartesianClosedCategory.html> Cartesian closed category
- [5] <http://planetmath.org/encyclopedia/SequentialMachine3.html> The Category of Automata
- [6] <http://www.csee.wvu.edu/~jwething/asl2010.pdf> James Worthington.2010.Determinizing, Forgetting, and Automata in Monoidal Categories. ASL North American Annual Meeting,March 17, 2010
- [7] Aguiar, M. and Mahajan, S.2010. "Monoidal Functors, Species, and Hopf Algebras".
- [8] Meseguer, J., Montanari, U.: 1990 Petri nets are monoids. *Information and Computation* **88**:105–155
- [9] http://en.wikipedia.org/wiki/Groupoid#Category_of_groupoids Category of groupoids

Further reading

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman (2000). *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Pearson Education. ISBN 0-201-44124-1.
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X. Part One: Automata and Languages, chapters 1–2, pp. 29–122. Section 4.1: Decidable Languages, pp. 152–159. Section 5.1: Undecidable Problems from Language Theory, pp. 172–183.
- Salomaa, Arto (1985). *Computation and automata*. Encyclopedia of Mathematics and Its Applications. **25**. Cambridge University Press. ISBN 0-521-30245-5. Zbl 0565.68046.
- Anderson, James A. (2006). *Automata theory with modern applications*. With contributions by Tom Head. Cambridge: Cambridge University Press. ISBN 0-521-61324-8. Zbl 1127.68049.
- Conway, J.H. (1971). *Regular algebra and finite machines*. Chapman and Hall Mathematics Series. London: Chapman & Hall. Zbl 0231.94041.
- Sakarovitch, Jacques (2009). *Elements of automata theory*. Translated from the French by Reuben Thomas. Cambridge University Press. ISBN 978-0-521-84425-3. Zbl 1188.68177.
- James P. Schmeiser, David T. Barnard (1995). *Producing a top-down parse order with bottom-up parsing*. Elsevier North-Holland.
- D'Souza, D. and Shankar, P. (2012). *Modern Applications of Automata Theory*. World Scientific Publishing, Singapore.

External links

- Visual Automata Simulator (<http://www.cs.usfca.edu/~jbovet/vas.html>), A tool for simulating, visualizing and transforming finite state automata and Turing Machines, by Jean Bovet
- JFLAP (<http://www.jflap.org>)
- dk.brics.automaton (<http://www.brics.dk/automaton>)
- libfa (<http://www.augeas.net/libfa/index.html>)

Principle of compositionality

In mathematics, semantics, and philosophy of language, the **Principle of Compositionality** is the principle that the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them. This principle is also called **Frege's Principle**, because Gottlob Frege is widely credited for the first modern formulation of it. However, the idea appears already among Indian philosophers of grammar such as Yāska, and also in Plato's work such as in *Theaetetus*.

The principle of compositionality states that in a meaningful sentence, if the lexical parts are taken out of the sentence, what remains will be the rules of composition. Take, for example, the sentence "Socrates was a man". Once the meaningful lexical items are taken away—"Socrates" and "man"—what is left is the pseudo-sentence, "S was a M". The task becomes a matter of describing what the connection is between S and M.

It is frequently taken to mean that every operation of the syntax should be associated with an operation of the semantics that acts on the meanings of the constituents combined by the syntactic operation. As a guideline for constructing semantic theories, this is generally taken, as in the influential work on the philosophy of language by Donald Davidson, to mean that every construct of the syntax should be associated by a clause of the T-schema with an operator in the semantics that specifies how the meaning of the whole expression is built from constituents combined by the syntactic rule. In some general mathematical theories (especially those in the tradition of Montague grammar) this guideline is taken to mean that the interpretation of a language is essentially given by a homomorphism between an algebra of syntactic representations and an algebra of semantic objects.

The Principle of Compositionality also exists in a similar form in the compositionality of programming languages.

Critiques

The principle of compositionality has been the subject of intense debate. Indeed, there is no general agreement as to how the principle is to be interpreted, although there have been several attempts to provide formal definitions of it. (Szabó, 2012)

Scholars are also divided as to whether the principle should be regarded as a factual claim, open to empirical testing; a analytic truth, obvious from the nature of language and meaning; or a methodological principle to guide the development of theories of syntax and semantics. The principle has been attacked in all three spheres, although so far none of the criticisms brought against it have been generally regarded as compelling. Most proponents of the principle, however, make certain exceptions for idiomatic expressions in natural language. (Szabó, 2012)

Further, in the context of the philosophy of language, the principle of compositionality does not explain all of meaning. For example, you cannot infer sarcasm purely on the basis of words and their composition, yet a phrase used sarcastically means something completely different from the same phrase uttered straightforwardly. The principle of compositionality, then, some theorists thus think that the principle has to be revised to take into account linguistic and extralinguistic context, which includes the tone of voice used, common ground between the speakers, the intentions of the speaker, and so on. (Szabó, 2012)

External links

- Compositionality [1] entry by Zoltán Gendler Szabó in the *Stanford Encyclopedia of Philosophy*
- The Oxford Handbook of Compositionality (OUP, 2012, edited by Markus Werning, Wolfram Hinzen, & Edouard Machery) [2]
- The Compositionality of Meaning and Content (Vol. I & II, Ontos, 2004, edited by Markus Werning, Edouard Machery, & Gerhard Schurz) [3]

References

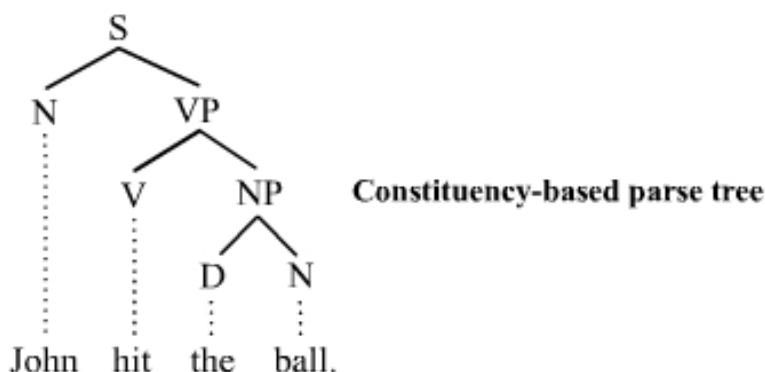
- [1] <http://plato.stanford.edu/entries/compositionality>
 [2] <http://ukcatalogue.oup.com/product/9780199541072.do>
 [3] http://ontosverlag.de/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=86&category_id=10&option=com_virtuemart&Itemid=1

Parse tree

A **concrete syntax tree** or **parse tree** or **parsing tree**^[1] is an ordered, rooted tree that represents the syntactic structure of a string according to some formal grammar. Parse trees are usually constructed according to one of two competing relations, either in terms of the constituency relation of constituency grammars (= phrase structure grammars) or in terms of the dependency relation of dependency grammars. Parse trees are distinct from abstract syntax trees (also known simply as syntax trees), in that their structure and elements more concretely reflect the syntax of the input language. Parse trees may be generated for sentences in natural languages (see natural language processing), as well as during processing of computer languages, such as programming languages.

Constituency-based parse trees

The constituency-based parse trees of constituency grammars (= phrase structure grammars) distinguish between terminal and non-terminal nodes. The interior nodes are labeled by non-terminal categories of the grammar, while the leaf nodes are labeled by terminal categories. The image below represents a constituency-based parse tree; it shows the syntactic structure of the English sentence *John hit the ball*:



This parse tree is simplified; for more information, see X-bar theory. The parse tree is the entire structure, starting from S and ending in each of the leaf nodes (*John*, *hit*, *the*, *ball*). The following abbreviations are used in the tree:

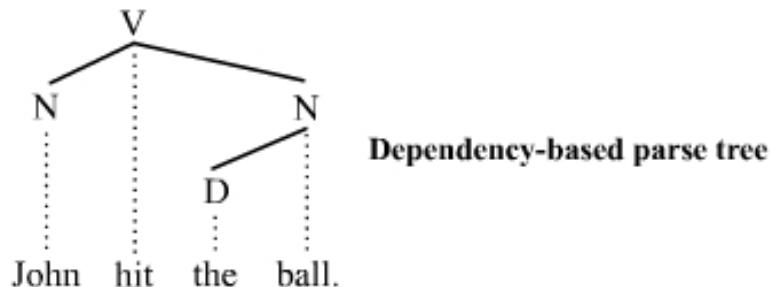
- S for sentence, the top-level structure in this example
- NP for noun phrase. The first (leftmost) NP, a single noun "John", serves as the subject of the sentence. The second one is the object of the sentence.
- VP for verb phrase, which serves as the predicate
- V for verb. In this case, it's a transitive verb *hit*.

- D for determiner, in this instance the definite article "the"
- N for noun

Each node in the tree is either a **root** node, a **branch** node, or a **leaf** node. S is the root node, NP and VP are branch nodes, and *John*, *hit*, *the*, and *ball* are all leaf nodes. The leaves are the lexical tokens of the sentence.^[2] A node can also be referred to as parent node or a child node. A **parent** node is one that has at least one other node linked by a branch under it. In the example, S is a parent of both NP and VP. A **child** node is one that has at least one node directly above it to which it is linked by a branch of the tree. From the example, *hit* is a child node of V. The terms **mother** and **daughter** are also sometimes used for this relationship.

Dependency-based parse trees

The dependency-based parse trees of dependency grammars^[3] see all nodes as terminal, which means they do not acknowledge the distinction between terminal and non-terminal categories. They are simpler on average than constituency-based parse trees because they contain many fewer nodes. The dependency-based parse tree for the example sentence above is as follows:



This parse tree lacks the phrasal categories (S, VP, and NP) seen in the constituency-based counterpart above. Like the constituency-based tree however, constituent structure is acknowledged. Any complete subtree of the tree is a constituent. Thus this dependency-based parse tree acknowledges the subject noun *John* and the object noun phrase *the ball* as constituents just like the constituency-based parse tree does.

The constituency vs. dependency distinction is far-reaching. Whether the additional syntactic structure associated with constituency-based parse trees is necessary or beneficial is a matter of debate.

Notes

[1] See Chiswell and Hodges 2007: 34.

[2] See Alfred et al. 2007.

[3] See for example Ágel et al. 2003/2006.

References

- Ágel, V., Ludwig Eichinger, Hans-Werner Eroms, Peter Hellwig, Hans Heringer, and Hennig Lobin (eds.) 2003/6. Dependency and valency: An international handbook of contemporary research. Berlin: Walter de Gruyter.
- Chiswell, Ian and Wilfrid Hodges 2007. Mathematical logic. Oxford: Oxford University Press.
- Aho, Alfred et al. 2007. Compilers: Principles, techniques, & tools. Boston: Pearson/Addison Wesley.

External links

- Syntax Tree Editor (<http://www.ductape.net/~eppie/tree/>)
- Linguistic Tree Constructor (<http://ltc.sourceforge.net/>)
- phpSyntaxTree (<http://www.ironcreek.net/phpsyntaxtree/>) – Online parse tree drawing site
- phpSyntaxTree (Unicode) (<http://lrv.bplaced.net/syntaxtree/>) – Online parse tree drawing site (improved version that supports Unicode)
- Qtree (<http://www.ling.upenn.edu/advice/latex/qtree/>) – LaTeX package for drawing parse trees
- TreeForm Syntax Tree Drawing Software (<http://www.ece.ubc.ca/~donaldd/treeform.htm>)
- rSyntaxTree (<http://yohasebe.com/rsyntaxtree/>) Enhanced version of phpSyntaxTree in Ruby with Unicode and Vectorized graphics

Deep structure

In linguistics, specifically in the study of syntax in the tradition of generative grammar (also known as transformational grammar), the **deep structure** of a linguistic expression is a theoretical construct that seeks to unify several related structures. For example, the sentences "Pat loves Chris" and "Chris is loved by Pat" mean roughly the same thing and use similar words. Some linguists, in particular Noam Chomsky, have tried to account for this similarity by positing that these two sentences are distinct *surface forms* that derive from a common *deep structure*.^[1]

The concept of deep structure plays an important role in transformational grammar. In early transformational syntax, deep structures are derivation trees of a context free language. These trees are then transformed by a sequence of tree rewriting operations ("transformations") into surface structures. The terminal yield of a surface structure tree, the surface form, is then predicted to be a grammatical sentence of the language being studied. The role and significance of deep structure changed a great deal as Chomsky developed his theories, and since the mid 1990s deep structure no longer features at all (see Transformational grammar).

It is tempting to regard deep structures as representing meanings and surface structures as representing sentences that express those meanings, but this is not the concept of deep structure favoured by Chomsky. Rather, a sentence more closely corresponds to a deep structure paired with the surface structure derived from it, with an additional phonetic form obtained from processing of the surface structure. It has been variously suggested that the interpretation of a sentence is determined by its deep structure alone, by a combination of its deep and surface structures, or by some other level of representation altogether (logical form), as argued in 1977 by Chomsky's student Robert May. Chomsky may have tentatively entertained the first of these ideas in the early 1960s, but quickly moved away from it to the second, and finally the third. Throughout the 1960s and 1970s, the generative semantics movement put up a vigorous defence of the first option, sparking an acrimonious debate, the "Linguistics Wars".^[2]

Chomsky noted in his early years that by dividing deep structures from surface structures, one could understand "slip of the tongue" moments (where one says something that they did not intend) as instances where deep structures do not translate into the intended surface structure.^[3]

The "surface" appeal of the deep structure concept soon led people from unrelated fields (architecture, music, politics, and even ritual studies) to use the term to express various concepts in their own work. In common usage, the term is often used as a synonym for universal grammar—the constraints which Chomsky claims govern the overall forms of linguistic expression available to the human species. This is probably due to the importance of deep structure in Chomsky's earlier work on universal grammar, though his concept of universal grammar is logically independent of any particular theoretical construct, including deep structure.

According to Middleton (1990), Schenkerian analysis of music corresponds to the Chomskyan notion of deep structure, applying to a two-level generative structure for melody, harmony, and rhythm, of which the analysis by

Lee (1985) of rhythmical structure is an instance. See also Chord progression.

References

Notes

- [1] In the first formulations of transformational grammar, active and passive pairs had identical deep structures. As the theory developed, it became necessary to mark whether a sentence was active or passive in the deep structure itself, with the result that active/passive pairs had almost-but-not-quite identical deep structures.
- [2] Harris, Randy Allen. (1995). *The linguistics wars*. Oxford University Press. ISBN 0-19-509834-X.
- [3] Carlson [et al.], Neil R. (2005). Psychology: The Science of Behaviour 3rd Canadian Edition. Pearson. pp. 310-311. ISBN 0-205-45769-X.

General references

1. Noam Chomsky (1957). *Syntactic Structures*. Mouton.
2. Noam Chomsky (1965). *Aspects of the Theory of Syntax*. MIT Press.
3. Noam Chomsky (1981). *Lectures on Government and Binding*. Mouton.
4. Noam Chomsky (1986). *Barriers*. Linguistic Inquiry Monographs. MIT Press.
5. C. S. Lee (1985). "The rhythmic interpretation of simple musical sequences: towards a perceptual model", in P. Howell, I. Cross and R. West (eds.), *Musical Structure and Cognition* (Academic Press), pp. 53–69.
6. Richard Middleton (1990). *Studying Popular Music*. Open University Press.
7. Samovar, L, & Porter, R (August 2003). Communication between Cultures .Wadsworth Publishing.

Ambiguous grammar

In computer science, an **ambiguous grammar** is a formal grammar for which there exists a string that can have more than one leftmost derivation.

Example

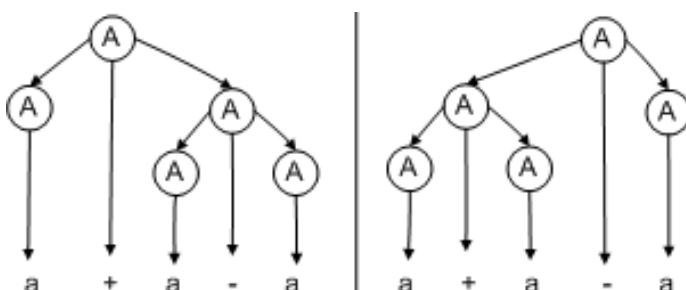
The context free grammar

$$A \rightarrow A + A \mid A - A \mid a$$

is ambiguous since there are two leftmost derivations for the string a + a + a:

$A \rightarrow A + A$	$A \rightarrow A + A$
$\rightarrow a + A$	$\rightarrow A + A + A$ (First A is replaced by A+A. Replacement of the second A would yield a similar derivation)
$\rightarrow a + A + A$	$\rightarrow a + A + A$
$\rightarrow a + a + A$	$\rightarrow a + a + A$
$\rightarrow a + a + a$	$\rightarrow a + a + a$

As another example, the grammar is ambiguous since there are two parse trees for the string a + a - a:



The language that it generates, however, is not inherently ambiguous; the following is a non-ambiguous grammar generating the same language:

$$A \rightarrow A + a \mid A - a \mid a$$

Recognizing ambiguous grammars

The general decision problem of whether a grammar is ambiguous is undecidable because it can be shown that it is equivalent to the Post correspondence problem. At least, there are tools implementing some semi-decision procedure for detecting ambiguity of context-free grammars.^[1]

The efficiency of context-free grammar parsing is determined by the automaton that accepts it. Deterministic context-free grammars are accepted by deterministic pushdown automata and can be parsed in linear time, for example by the LR parser.^[2] This is a subset of the context-free grammars which are accepted by the pushdown automaton and can be parsed in polynomial time, for example by the CYK algorithm. Unambiguous context-free grammars can be nondeterministic. For example, the language of even-length palindromes on the alphabet of 0 and 1 has the unambiguous context-free grammar $S \rightarrow 0S0 \mid 1S1 \mid \epsilon$. An arbitrary string of this language cannot be parsed without reading all its letters first which means that a pushdown automaton has to try alternative state transitions to accommodate for the different possible lengths of a semi-parsed string.^[3] Nevertheless, removing grammar ambiguity may produce a deterministic context-free grammar and thus allow for more efficient parsing. Compiler generators such as YACC include features for resolving some kinds of ambiguity, such as by using the precedence and associativity constraints.

Inherently ambiguous languages

Inherent ambiguity was proven with Parikh's theorem in 1961 by Rohit Parikh in an MIT research report.^[4]

While some context-free languages (the set of strings that can be generated by a grammar) have both ambiguous and unambiguous grammars, there exist context-free languages for which no unambiguous context-free grammar can exist. An example of an inherently ambiguous language is the union of $\{a^n b^m c^m d^n | n, m > 0\}$ with $\{a^n b^n c^m d^m | n, m > 0\}$. This set is context-free, since the union of two context-free languages is always context-free. But Hopcroft & Ullman (1979) give a proof that there is no way to unambiguously parse strings in the (non-context-free) subset $\{a^n b^n c^n d^n | n > 0\}$ which is the intersection of these two languages.

References

- [1] Axelsson, Roland; Heljanko, Keijo; Lange, Martin (2008). "Analyzing Context-Free Grammars Using an Incremental SAT Solver". *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08), Reykjavik, Iceland*. Lecture Notes in Computer Science. **5126**. Springer-Verlag. pp. 410–422.
- [2] Knuth, Donald (July 1965). "On the Translation of Languages from Left to Right" (<http://www.cs.dartmouth.edu/~mckeeeman/cs48/mxcom/doc/knuth65.pdf>). *Information and Control* **8**: 707 - 639. . Retrieved 29 May 2011.
- [3] Hopcroft, John; Rajeev Motwani & Jeffrey Ullman (2001). *Introduction to automata theory, languages, and computation 2nd edition*. Addison-Wesley. pp. 249–253.
- [4] Parikh, Rohit (January 1961). *Language-generating devices*. Quarterly Progress Report, Research Laboratory of Electronics, MIT.
- Gross, Maurice (1964). *Inherent ambiguity of minimal linear grammars*. *7:3*. Information and Control. pp. 366–368.
- Michael, Harrison. (1978). *Introduction to Formal Language Theory*. Addison-Wesley.
- Hopcroft, John E.; Ullman, Jeffrey D. (1979). *Introduction to Automata Theory, Languages, and Computation* (1st ed.). Addison-Wesley.
- Hopcroft, John.; Mowani, Rajeev.; Ullman, Jeffrey. (2001). *Introduction to Automata Theory, Languages and Computation (second edition)*. Addison Wesley. pp. 217.

- Brabrand, Claus; Giegerich, Robert; Møller, Anders (2010). "Analyzing Ambiguity of Context-Free Grammars". *Science of Computer Programming*. 75. Elsevier. pp. 176–191.

External links

- dk.brics.grammar (<http://www.brics.dk/grammar>) - a grammar ambiguity analyzer.
- CFGAnalyzer (<http://www2.tcs.ifl.lmu.de/~mlange/cfganalyzer/index.html>) - tool for analyzing context-free grammars with respect to language universality, ambiguity, and similar properties.

Phrase structure grammar

The term **phrase structure grammar** was originally introduced by Noam Chomsky as the term for grammars as defined by phrase structure rules,^[1] i.e. rewrite rules of the type studied previously by Emil Post and Axel Thue (see Post canonical systems). Some authors, however, reserve the term for more restricted grammars in the Chomsky hierarchy: context-sensitive grammars, or context-free grammars. In a broader sense, phrase structure grammars are also known as *constituency grammars*. The defining trait of phrase structure grammars is thus their adherence to the constituency relation, as opposed to the dependency relation of dependency grammars.

Constituency relation

In linguistics, phrase structure grammars are all those grammars that are based on the constituency relation, as opposed to the dependency relation associated with dependency grammars; hence phrase structure grammars are also known as *constituency grammars*.^[2] Any of several related theories for the parsing of natural language qualify as constituency grammars, and most of them have been developed from Chomsky's work, including

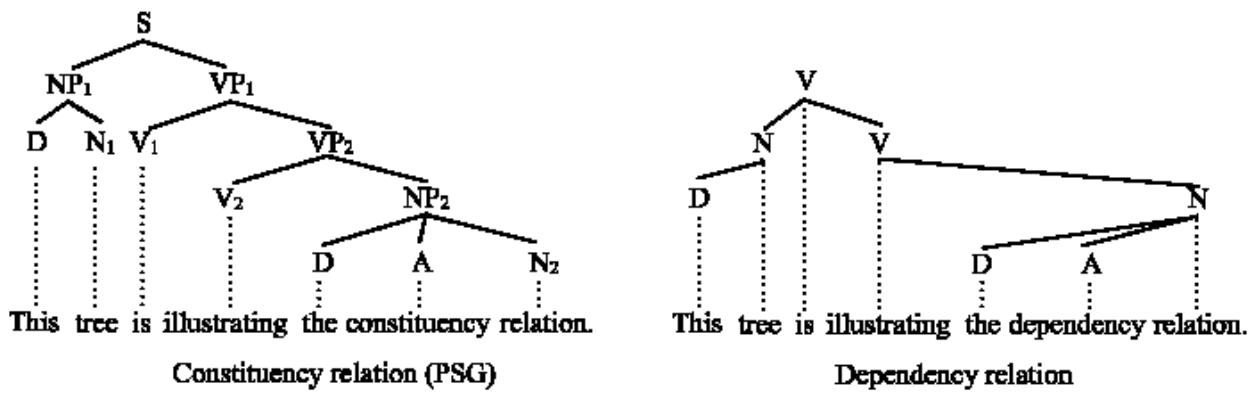
- Government and Binding Theory,
- Generalized Phrase Structure Grammar,
- Head-Driven Phrase Structure Grammar,
- Lexical Functional Grammar,
- The Minimalist Program, and
- Nanosyntax.

Further grammar frameworks and formalisms also qualify as constituency-based, although they may not think of themselves as having spawned from Chomsky's work, e.g.

- Arc Pair Grammar and
- Categorial Grammar.

The fundamental trait that these frameworks all share is that they view sentence structure in terms of the constituency relation. The constituency relation derives from the subject-predicate division of Latin and Greek grammars that is based on term logic and reaches back to Aristotle in antiquity. Basic clause structure is understood in terms of a binary division of the clause into subject (noun phrase NP) and predicate (verb phrase VP).

The binary division of the clause results in a one-to-one-or-more correspondence. For each element in a sentence, there are one or more nodes in the tree structure that one assumes for that sentence. A two word sentence such as *Luke laughed* necessarily implies three (or more) nodes in the syntactic structure: one for the noun *Luke* (subject NP), one for the verb *laughed* (predicate VP), and one for the entirety *Luke laughed* (sentence S). The constituency grammars listed above all view sentence structure in terms of this one-to-one-or-more correspondence.



Dependency relation

By the time of Gottlob Frege, a competing understanding of the logic of sentences had arisen. Frege rejected the binary division of the sentence and replaced it with an understanding of sentence logic in terms of predicates and their arguments. On this alternative conception of sentence logic, the binary division of the clause into subject and predicate was not possible. It therefore opened the door to the dependency relation (although the dependency relation had also existed in a less obvious form in traditional grammars long before Frege). The dependency relation was first acknowledged concretely and developed as the basis for a comprehensive theory of syntax and grammar by Lucien Tesnière in his posthumously published work *Éléments de syntaxe structurale* (Elements of Structural Syntax).^[3]

The dependency relation is a one-to-one correspondence: for every element (word or morph) in a sentence, there is just one node in the syntactic structure. The distinction is thus a graph-theoretical distinction. The dependency relation restricts the number of nodes in the syntactic structure of a sentence to the exact number of syntactic units (usually words) that that sentence contains. Thus the two word sentence *Luke laughed* implies just two syntactic nodes, one for *Luke* and one for *laughed*. Some prominent dependency grammars are listed here:

- Algebraic Syntax
- Functional Generative Description
- Lexicase
- Meaning-Text Theory
- Operator Grammar
- Word Grammar

Since these grammars are all based on the dependency relation, they are by definition NOT phrase structure grammars.

Non-descript grammars

Other grammars generally avoid attempts to group syntactic units into clusters in a manner that would allow classification in terms of the constituency vs. dependency distinction. In this respect, the following grammar frameworks do not come down solidly on either side of the dividing line:

- Construction grammar
- Cognitive grammar

Notes

- [1] See Chomsky (1957).
- [2] Matthews (1981:71ff.) provides an insightful discussion of the distinction between constituency- and dependency-based grammars. See also Allerton (1979:238f.), McCawley (1988:13), Mel'cuk (1988:12-14), Borsley (1991:30f.), Sag and Wasow (1999:421f.), van Valin (2001:86ff.).
- [3] See Tesnière (1959).

References

- Allerton, D. 1979. Essentials of grammatical theory. London: Routledge & Kegan Paul.
- Borsley, R. 1991. Syntactic theory: A unified approach. London: Edward Arnold.
- Chomsky, Noam 1957. Syntactic structures. The Hague/Paris: Mouton.
- Matthews, P. Syntax. 1981. Cambridge, UK: Cambridge University Press.
- McCawley, T. 1988. The syntactic phenomena of English, Vol. 1. Chicago: The University of Chicago Press.
- Mel'cuk, I. 1988. Dependency syntax: Theory and practice. Albany: SUNY Press.
- Sag, I. and T. Wasow. 1999. Syntactic theory: A formal introduction. Stanford, CA: CSLI Publications.
- Tesnière, Lucien 1959. *Éléments de syntaxe structurale*. Paris: Klincksieck.
- van Valin, R. 2001. An introduction to syntax. Cambridge, UK: Cambridge University Press.

Chomsky hierarchy

Chomsky hierarchy

Within the field of computer science, specifically in the area of formal languages, the **Chomsky hierarchy** (occasionally referred to as **Chomsky–Schützenberger hierarchy**) is a containment hierarchy of classes of formal grammars. This hierarchy of grammars was described by Noam Chomsky in 1956.^[1] It is also named after Marcel-Paul Schützenberger, who played a crucial role in the development of the theory of formal languages. The Chomsky Hierarchy, in essence, allows the possibility for the understanding and use of a computer science model which enables a programmer to accomplish meaningful linguistic goals systematically.

Formal grammars

A formal grammar of this type consists of:

- a finite set of *production rules* (*left-hand side* \rightarrow *right-hand side*) where each side consists of a sequence of these symbols
- a finite set of *nonterminal symbols* (indicating that some production rule can yet be applied)
- a finite set of *terminal symbols* (indicating that no production rule can be applied)
- a *start symbol* (a distinguished nonterminal symbol)

A formal grammar defines (or *generates*) a *formal language*, which is a (usually infinite) set of finite-length sequences of symbols (i.e. strings) that may be constructed by applying production rules to another sequence of symbols which initially contains just the start symbol. A rule may be applied to a sequence of symbols by replacing an occurrence of the symbols on the left-hand side of the rule with those that appear on the right-hand side. A sequence of rule applications is called a *derivation*. Such a grammar defines the formal language: all words consisting solely of terminal symbols which can be reached by a derivation from the start symbol.

Nonterminals are usually represented by uppercase letters, terminals by lowercase letters, and the start symbol by S . For example, the grammar with terminals $\{a, b\}$, nonterminals $\{S, A, B\}$, production rules

$$S \rightarrow ABS$$

$$S \rightarrow \epsilon \text{ (where } \epsilon \text{ is the empty string)}$$

$$BA \rightarrow AB$$

$$BS \rightarrow b$$

$$Bb \rightarrow bb$$

$$Ab \rightarrow ab$$

$$Aa \rightarrow aa$$

and start symbol S , defines the language of all words of the form $a^n b^n$ (i.e. n copies of a followed by n copies of b). The following is a simpler grammar that defines the same language: Terminals $\{a, b\}$, Nonterminals $\{S\}$, Start symbol S , Production rules

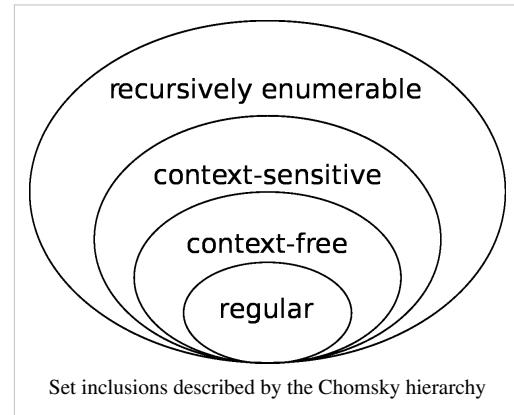
$$S \rightarrow aSb$$

$$S \rightarrow \epsilon$$

The hierarchy

The Chomsky hierarchy consists of the following levels:

- Type-0 grammars (unrestricted grammars) include all formal grammars. They generate exactly all languages that can be recognized by a Turing machine. These languages are also known as the recursively enumerable languages. Note that this is different from the recursive languages which can be *decided* by an always-halting Turing machine.
- Type-1 grammars (context-sensitive grammars) generate the context-sensitive languages. These grammars have rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ with A a nonterminal and α, β and γ strings of terminals and nonterminals. The strings α and β may be empty, but γ must be nonempty. The rule $S \rightarrow \epsilon$ is allowed if S does not appear on the right side of any rule. The languages described by these grammars are exactly all languages that can be recognized by a linear bounded automaton (a nondeterministic Turing machine whose tape is bounded by a constant times the length of the input.)
- Type-2 grammars (context-free grammars) generate the context-free languages. These are defined by rules of the form $A \rightarrow \gamma$ with A a nonterminal and γ a string of terminals and nonterminals. These languages are exactly all languages that can be recognized by a non-deterministic pushdown automaton. Context-free languages are the theoretical basis for the syntax of most programming languages.
- Type-3 grammars (regular grammars) generate the regular languages. Such a grammar restricts its rules to a single nonterminal on the left-hand side and a right-hand side consisting of a single terminal, possibly followed (or preceded, but not both in the same grammar) by a single nonterminal. The rule $S \rightarrow \epsilon$ is also allowed here if S does not appear on the right side of any rule. These languages are exactly all languages that can be decided by a finite state automaton. Additionally, this family of formal languages can be obtained by regular expressions. Regular languages are commonly used to define search patterns and the lexical structure of programming languages.



Note that the set of grammars corresponding to recursive languages is not a member of this hierarchy.

Every regular language is context-free, every context-free language, not containing the empty string, is context-sensitive and every context-sensitive language is recursive and every recursive language is recursively enumerable. These are all proper inclusions, meaning that there exist recursively enumerable languages which are not context-sensitive, context-sensitive languages which are not context-free and context-free languages which are not regular.

The following table summarizes each of Chomsky's four types of grammars, the class of language it generates, the type of automaton that recognizes it, and the form its rules must have.

Grammar	Languages	Automaton	Production rules (constraints)
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$ (no restrictions)
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A\beta \rightarrow \alpha\gamma\beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow a$ and $A \rightarrow aB$

However, there are further categories of formal languages, some of which are given in the following table:

References

- [1] Chomsky, Noam (1956). "Three models for the description of language" (<http://www.chomsky.info/articles/195609--.pdf>). *IRE Transactions on Information Theory* (2): 113–124. doi:10.1109/TIT.1956.1056813. .
- Chomsky, Noam (1959). "On certain formal properties of grammars" (<http://www.diku.dk/hjemmesider/ansatte/henglein/papers/chomsky1959.pdf>). *Information and Control* 2 (2): 137–167. doi:10.1016/S0019-9958(59)90362-6.
- Chomsky, Noam; Schützenberger, Marcel P. (1963). "The algebraic theory of context free languages". In Bräffort, P.; Hirschberg, D.. *Computer Programming and Formal Languages*. Amsterdam: North Holland. pp. 118–161.
- Davis, Martin E.; Sigal, Ron; Weyuker, Elaine J. (1994). *Computability, complexity, and languages: Fundamentals of theoretical computer science*. Boston: Academic Press, Harcourt, Brace. pp. 327. ISBN 0-12-206382-1.

External links

- <http://www.staff.ncl.ac.uk/hermann.moisl/ell236/lecture5.htm>

Unrestricted grammar

In formal language theory, an **unrestricted grammar** is a formal grammar on which no restrictions are made on the left and right sides of the grammar's productions. This is the most general class of grammars in the Chomsky–Schützenberger hierarchy, and can generate arbitrary recursively enumerable languages.

Formal definition

An **unrestricted grammar** is a formal grammar $G = (N, \Sigma, P, S)$, where N is a set of nonterminal symbols, Σ is a set of terminal symbols, N and Σ are disjoint (actually, this is not strictly necessary, because unrestricted grammars make no real distinction between nonterminal and terminal symbols, the designation exists purely so that one knows when to stop when trying to generate sentential forms of the grammar), P is a set of production rules of the form $\alpha \rightarrow \beta$ where α and β are strings of symbols in $N \cup \Sigma$ and α is not the empty string, and $S \in N$ is a specially designated start symbol. As the name implies, there are no real restrictions on the types of production rules that unrestricted grammars can have.

Unrestricted grammars and Turing machines

It may be shown that unrestricted grammars characterize the recursively enumerable languages. This is the same as saying that for every unrestricted grammar G there exists some Turing machine capable of recognizing $L(G)$ and vice-versa. Given an unrestricted grammar, such a Turing machine is simple enough to construct, as a two-tape nondeterministic Turing machine. The first tape contains the input word w to be tested, and the second tape is used by the machine to generate sentential forms from G . The Turing machine then does the following:

1. Start at the left of the second tape and repeatedly choose to move right or select the current position on the tape.
2. Nondeterministically choose a production $\beta \rightarrow \gamma$ from the productions in G .
3. If β appears at some position on the second tape, replace β by γ at that point, possibly shifting the symbols on the tape left or right depending on the relative lengths of β and γ (e.g. if β is longer than γ , shift the tape symbols left).
4. Compare the resulting sentential form on tape 2 to the word on tape 1. If they match, then the Turing machine accepts the word. If they don't go back to step 1.

It is easy to see that this Turing machine will generate all and only the sentential forms of G on its second tape after the last step is executed an arbitrary number of times, thus the language $L(G)$ must be recursively enumerable.

The reverse construction is also possible. Given some Turing machine, it is possible to create an unrestricted grammar.

Computational properties

The decision problem of whether a given string s can be generated by a given unrestricted grammar is equivalent to the problem of whether it can be accepted by the Turing machine equivalent to the grammar. The latter problem is called the Halting problem and is undecidable.

The equivalence of unrestricted grammars to Turing machines implies the existence of a universal unrestricted grammar, a grammar capable of accepting any other unrestricted grammar's language given a description of the language. For this reason, it is theoretically possible to build a programming language based on unrestricted grammars (e.g. Thue).

References

- John Hopcroft and Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation* (1st edition ed.). Addison-Wesley. ISBN 0-201-44124-1.

Turing machine

A **Turing machine** is a device that manipulates symbols on a strip of tape according to a table of rules. Despite its simplicity, a Turing machine can be adapted to simulate the logic of any computer algorithm, and is particularly useful in explaining the functions of a CPU inside a computer.

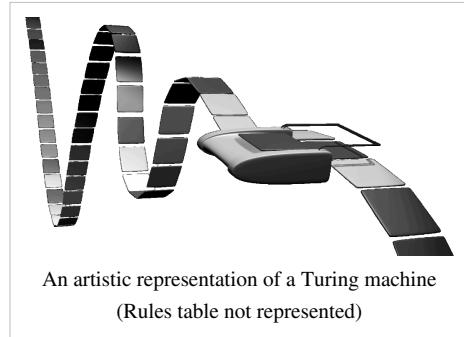
The "Turing" machine was described in 1936 by Alan Turing^[1] who called it an "a-machine" (automatic machine). The Turing machine is not intended as practical computing technology, but rather as a hypothetical device representing a computing machine. Turing machines help computer scientists understand the limits of mechanical computation.

Turing gave a succinct definition of the experiment in his 1948 essay, "Intelligent Machinery". Referring to his 1936 publication, Turing wrote that the Turing machine, here called a Logical Computing Machine, consisted of:

...an unlimited memory capacity obtained in the form of an infinite tape marked out into squares, on each of which a symbol could be printed. At any moment there is one symbol in the machine; it is called the scanned symbol. The machine can alter the scanned symbol and its behavior is in part determined by that symbol, but the symbols on the tape elsewhere do not affect the behaviour of the machine. However, the tape can be moved back and forth through the machine, this being one of the elementary operations of the machine. Any symbol on the tape may therefore eventually have an innings.^[2] (Turing 1948, p. 61)

A Turing machine that is able to simulate any other Turing machine is called a universal Turing machine (**UTM**, or simply a **universal machine**). A more mathematically oriented definition with a similar "universal" nature was introduced by Alonzo Church, whose work on lambda calculus intertwined with Turing's in a formal theory of computation known as the Church–Turing thesis. The thesis states that Turing machines indeed capture the informal notion of effective method in logic and mathematics, and provide a precise definition of an algorithm or 'mechanical procedure'.

Studying their abstract properties yields many insights into computer science and complexity theory.



An artistic representation of a Turing machine
(Rules table not represented)

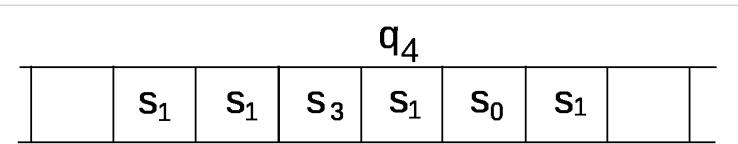
Informal description

For visualizations of Turing machines, see Turing machine gallery.

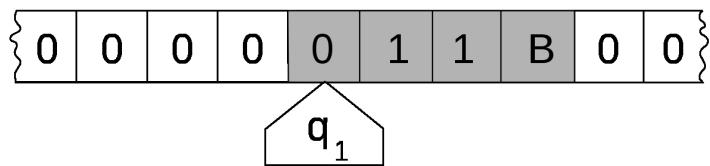
The Turing machine mathematically models a machine that mechanically operates on a tape. On this tape are symbols which the machine can read and write, one at a time, using a tape head. Operation is fully determined by a finite set of elementary instructions such as "in state 42, if the symbol seen is 0, write a 1; if the symbol seen is 1, change into state 17; in state 17, if the symbol seen is 0, write a 1 and change to state 6;" etc. In the original article ("On computable numbers, with an application to the Entscheidungsproblem", see also references below), Turing imagines not a mechanism, but a person whom he calls the "computer", who executes these deterministic mechanical rules slavishly (or as Turing puts it, "in a desultory manner").

More precisely, a Turing machine consists of:

1. A **tape** which is divided into cells, one next to the other. Each cell contains a symbol from some finite alphabet. The alphabet contains a special *blank* symbol (here written as '0') and one or more other symbols. The tape is assumed to be arbitrarily extendable to the left and to the right, i.e., the Turing machine is always supplied with as much tape as it needs for its computation. Cells that have not been written to before are assumed to be filled with the blank symbol. In some models the tape has a left end marked with a special symbol; the tape extends or is indefinitely extensible to the right.



The head is always over a particular square of the tape; only a finite stretch of squares is shown. The instruction to be performed (q₄) is shown over the scanned square. (Drawing after Kleene (1952) p.375.)



Here, the internal state (q₁) is shown inside the head, and the illustration describes the tape as being infinite and pre-filled with "0", the symbol serving as blank. The system's full state (its *complete configuration*) consists of the internal state, any non-blank symbols on the tape (in this illustration "11B"), and the position of the head relative to those symbols including blanks, i.e. "011B". (Drawing after Minsky (1967) p. 121).

2. A **head** that can read and write symbols on the tape and move the tape left and right one (and only one) cell at a time. In some models the head moves and the tape is stationary.
3. A **state register** that stores the state of the Turing machine, one of finitely many. There is one special *start state* with which the state register is initialized. These states, writes Turing, replace the "state of mind" a person performing computations would ordinarily be in.
4. A finite **table** (occasionally called an **action table** or **transition function**) of instructions (usually quintuples [5-tuples] : q_ia_j → q_{i1}a_{j1}d_k, but sometimes 4-tuples) that, given the *state*(q_i) the machine is currently in *and the symbol*(a_j) it is reading on the tape (symbol currently under the head) tells the machine to do the following in sequence (for the 5-tuple models):

- Either erase or write a symbol (replacing a_j with a_{j1}), *and then*
- Move the head (which is described by d_k and can have values: 'L' for one step left *or* 'R' for one step right *or* 'N' for staying in the same place), *and then*
- Assume the same or a *new state* as prescribed (go to state q_{i1}).

In the 4-tuple models, erasing or writing a symbol (a_{j1}) and moving the head left or right (d_k) are specified as separate instructions. Specifically, the table tells the machine to (ia) erase or write a symbol *or* (ib) move the head left or right, *and then* (ii) assume the same or a new state as prescribed, but not both actions (ia) and (ib) in the same instruction. In some models, if there is no entry in the table for the current combination of symbol and state then the machine will halt; other models require all entries to be filled.

Note that every part of the machine—its state and symbol-collections—and its actions—printing, erasing and tape motion—is *finite, discrete* and *distinguishable*; it is the potentially unlimited amount of tape that gives it an unbounded amount of storage space.

Formal definition

Hopcroft and Ullman (1979, p. 148) formally define a (one-tape) Turing machine as a 7-tuple $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ where

- Q is a finite, non-empty set of *states*
- Γ is a finite, non-empty set of the *tape alphabet/symbols*
- $b \in \Gamma$ is the *blank symbol* (the only symbol allowed to occur on the tape infinitely often at any step during the computation)
- $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of *input symbols*
- $q_0 \in Q$ is the *initial state*
- $F \subseteq Q$ is the set of *final or accepting states*.
- $\delta : Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a partial function called the *transition function*, where L is left shift, R is right shift. (A relatively uncommon variant allows "no shift", say N, as a third element of the latter set.)

Anything that operates according to these specifications is a Turing machine.

The 7-tuple for the 3-state busy beaver looks like this (see more about this busy beaver at Turing machine examples):

- $Q = \{A, B, C, \text{HALT}\}$
- $\Gamma = \{0, 1\}$
- $b = 0$ ("blank")
- $\Sigma = \{1\}$
- $q_0 = A$ (the initial state)
- $F = \{\text{HALT}\}$
- δ = see state-table below

Initially all tape cells are marked with 0.

State table for 3 state, 2 symbol busy beaver

Tape symbol	Current state A			Current state B			Current state C		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state
0	1	R	B	1	L	A	1	L	B
1	1	L	C	1	R	B	1	R	HALT

Additional details required to visualize or implement Turing machines

In the words of van Emde Boas (1990), p. 6: "The set-theoretical object [his formal seven-tuple description similar to the above] provides only partial information on how the machine will behave and what its computations will look like."

For instance,

- There will need to be many decisions on what the symbols actually look like, and a failproof way of reading and writing symbols indefinitely.
- The shift left and shift right operations may shift the tape head across the tape, but when actually building a Turing machine it is more practical to make the tape slide back and forth under the head instead.
- The tape can be finite, and automatically extended with blanks as needed (which is closest to the mathematical definition), but it is more common to think of it as stretching infinitely at both ends and being pre-filled with blanks except on the explicitly given finite fragment the tape head is on. (This is, of course, not implementable in practice.) The tape *cannot* be fixed in length, since that would not correspond to the given definition and would seriously limit the range of computations the machine can perform to those of a linear bounded automaton.

Alternative definitions

Definitions in literature sometimes differ slightly, to make arguments or proofs easier or clearer, but this is always done in such a way that the resulting machine has the same computational power. For example, changing the set $\{L, R\}$ to $\{L, R, N\}$, where N ("None" or "No-operation") would allow the machine to stay on the same tape cell instead of moving left or right, does not increase the machine's computational power.

The most common convention represents each "Turing instruction" in a "Turing table" by one of nine 5-tuples, per the convention of Turing/Davis (Turing (1936) in *Undecidable*, p. 126-127 and Davis (2000) p. 152):

(definition 1): $(q_i, S_j, S_k/E/N, L/R/N, q_m)$

(current state q_i , symbol scanned S_j , print symbol S_k /erase E /none N , move_tape_one_square left L /right R /none N , new state q_m)

Other authors (Minsky (1967) p. 119, Hopcroft and Ullman (1979) p. 158, Stone (1972) p. 9) adopt a different convention, with new state q_m listed immediately after the scanned symbol S_j :

(definition 2): $(q_i, S_j, q_m, S_k/E/N, L/R/N)$

(current state q_i , symbol scanned S_j , new state q_m , print symbol S_k /erase E /none N , move_tape_one_square left L /right R /none N)

For the remainder of this article "definition 1" (the Turing/Davis convention) will be used.

Example: state table for the 3-state 2-symbol busy beaver reduced to 5-tuples

Current state	Scanned symbol	Print symbol	Move tape	Final (i.e. next) state	5-tuples
A	0	1	R	B	(A, 0, 1, R, B)
A	1	1	L	C	(A, 1, 1, L, C)
B	0	1	L	A	(B, 0, 1, L, A)
B	1	1	R	B	(B, 1, 1, R, B)
C	0	1	L	B	(C, 0, 1, L, B)
C	1	1	N	H	(C, 1, 1, N, H)

In the following table, Turing's original model allowed only the first three lines that he called N1, N2, N3 (cf Turing in *Undecidable*, p. 126). He allowed for erasure of the "scanned square" by naming a 0th symbol S_0 = "erase" or "blank", etc. However, he did not allow for non-printing, so every instruction-line includes "print symbol S_k " or "erase" (cf footnote 12 in Post (1947), *Undecidable* p. 300). The abbreviations are Turing's (*Undecidable* p. 119). Subsequent to Turing's original paper in 1936–1937, machine-models have allowed all nine possible types of five-tuples:

	Current m-configuration (Turing state)	Tape symbol	Print-operation	Tape-motion	Final m-configuration (Turing state)	5-tuple	5-tuple comments	4-tuple
N1	q_i	S_j	Print(S_k)	Left L	q_m	(q_i, S_j, S_k, L, q_m)	"blank" = S_0 , $1=S_1$, etc.	
N2	q_i	S_j	Print(S_k)	Right R	q_m	(q_i, S_j, S_k, R, q_m)	"blank" = S_0 , $1=S_1$, etc.	
N3	q_i	S_j	Print(S_k)	None N	q_m	(q_i, S_j, S_k, N, q_m)	"blank" = S_0 , $1=S_1$, etc.	(q_i, S_j, S_k, q_m)
4	q_i	S_j	None N	Left L	q_m	(q_i, S_j, N, L, q_m)		(q_i, S_j, L, q_m)
5	q_i	S_j	None N	Right R	q_m	(q_i, S_j, N, R, q_m)		(q_i, S_j, R, q_m)

6	q_i	S_j	None N	None N	q_m	(q_i, S_j, N, N, q_m)	Direct "jump"	(q_i, S_j, N, q_m)
7	q_i	S_j	Erase	Left L	q_m	(q_i, S_j, E, L, q_m)		
8	q_i	S_j	Erase	Right R	q_m	(q_i, S_j, E, R, q_m)		
9	q_i	S_j	Erase	None N	q_m	(q_i, S_j, E, N, q_m)		(q_i, S_j, E, q_m)

Any Turing table (list of instructions) can be constructed from the above nine 5-tuples. For technical reasons, the three non-printing or "N" instructions (4, 5, 6) can usually be dispensed with. For examples see Turing machine examples.

Less frequently the use of 4-tuples are encountered: these represent a further atomization of the Turing instructions (cf Post (1947), Boolos & Jeffrey (1974, 1999), Davis-Sigal-Weyuker (1994)); also see more at Post–Turing machine.

The "state"

The word "state" used in context of Turing machines can be a source of confusion, as it can mean two things. Most commentators after Turing have used "state" to mean the name/designator of the current instruction to be performed—i.e. the contents of the state register. But Turing (1936) made a strong distinction between a record of what he called the machine's "m-configuration", (its internal state) and the machine's (or person's) "state of progress" through the computation - the current state of the total system. What Turing called "the state formula" includes both the current instruction and *all* the symbols on the tape:

Thus the state of progress of the computation at any stage is completely determined by the note of instructions and the symbols on the tape. That is, the **state of the system** may be described by a single expression (sequence of symbols) consisting of the symbols on the tape followed by Δ (which we suppose not to appear elsewhere) and then by the note of instructions. This expression is called the 'state formula'.

—*Undecidable*, p.139–140, emphasis added

Earlier in his paper Turing carried this even further: he gives an example where he places a symbol of the current "m-configuration"—the instruction's label—beneath the scanned square, together with all the symbols on the tape (*Undecidable*, p. 121); this he calls "the *complete configuration*" (*Undecidable*, p. 118). To print the "complete configuration" on one line he places the state-label/m-configuration to the *left* of the scanned symbol.

A variant of this is seen in Kleene (1952) where Kleene shows how to write the Gödel number of a machine's "situation": he places the "m-configuration" symbol q_4 over the scanned square in roughly the center of the 6 non-blank squares on the tape (see the Turing-tape figure in this article) and puts it to the *right* of the scanned square. But Kleene refers to " q_4 " itself as "the machine state" (Kleene, p. 374-375). Hopcroft and Ullman call this composite the "instantaneous description" and follow the Turing convention of putting the "current state" (instruction-label, m-configuration) to the *left* of the scanned symbol (p. 149).

Example: total state of 3-state 2-symbol busy beaver after 3 "moves" (taken from example "run" in the figure below):

1A1

This means: after three moves the tape has ... 000110000 ... on it, the head is scanning the right-most 1, and the state is **A**. Blanks (in this case represented by "0"s) can be part of the total state as shown here: **B01**; the tape has a single 1 on it, but the head is scanning the 0 ("blank") to its left and the state is **B**.

"State" in the context of Turing machines should be clarified as to which is being described: (i) the current instruction, or (ii) the list of symbols on the tape together with the current instruction, or (iii) the list of symbols on

the tape together with the current instruction placed to the left of the scanned symbol or to the right of the scanned symbol.

Turing's biographer Andrew Hodges (1983: 107) has noted and discussed this confusion.

Turing machine "state" diagrams

The table for the 3-state busy beaver ("P" = print/write a "1")

Tape symbol	Current state A			Current state B			Current state C		
	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state	Write symbol	Move tape	Next state
0	P	R	B	P	L	A	P	L	B
1	P	L	C	P	R	B	P	R	HALT

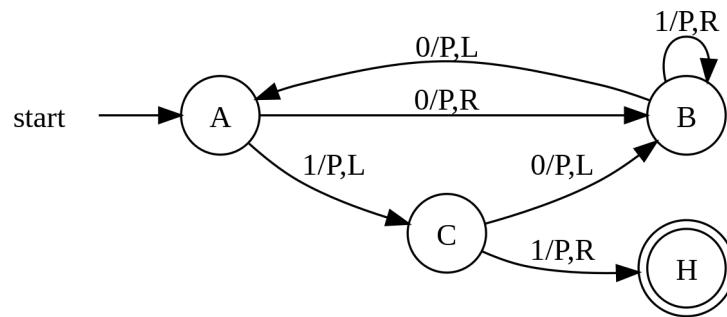
To the right: the above TABLE as expressed as a "state transition" diagram.

Usually large TABLES are better left as tables (Booth, p. 74). They are more readily simulated by computer in tabular form (Booth, p. 74). However, certain concepts—e.g. machines with "reset" states and machines with repeating patterns (cf Hill and Peterson p. 244ff)—can be more readily seen when viewed as a drawing.

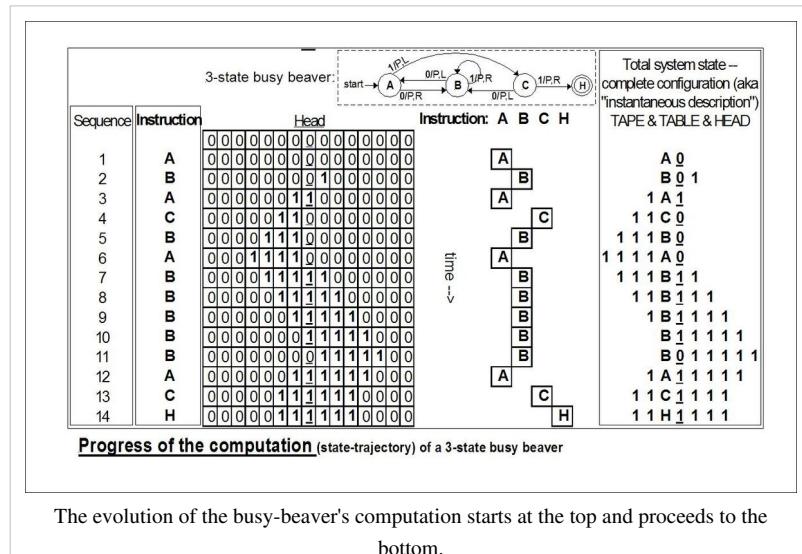
Whether a drawing represents an improvement on its TABLE must be decided by the reader for the particular context. See Finite state machine for more.

The reader should again be cautioned that such diagrams represent a snapshot of their TABLE frozen in time, *not* the course ("trajectory") of a computation *through* time and/or space. While every time the busy beaver machine "runs" it will always follow the same state-trajectory, this is not true for the "copy" machine that can be provided with variable input "parameters".

The diagram "Progress of the computation" shows the 3-state busy beaver's "state" (instruction) progress through its computation from start to



The "3-state busy beaver" Turing machine in a finite state representation. Each circle represents a "state" of the TABLE—an "m-configuration" or "instruction". "Direction" of a state *transition* is shown by an arrow. The label (e.g., 0/P,R) near the outgoing state (at the "tail" of the arrow) specifies the scanned symbol that causes a particular transition (e.g. 0) followed by a slash /, followed by the subsequent "behaviors" of the machine, e.g. "P Print" then move tape "R Right". No general accepted format exists. The convention shown is after McClusky (1965), Booth (1967), Hill, and Peterson (1974).



The evolution of the busy-beaver's computation starts at the top and proceeds to the bottom.

finish. On the far right is the Turing "complete configuration" (Kleene "situation", Hopcroft–Ullman "instantaneous description") at each step. If the machine were to be stopped and cleared to blank both the "state register" and entire tape, these "configurations" could be used to rekindle a computation anywhere in its progress (cf Turing (1936) *Undecidable* pp. 139–140).

Models equivalent to the Turing machine model

Many machines that might be thought to have more computational capability than a simple universal Turing machine can be shown to have no more power (Hopcroft and Ullman p. 159, cf Minsky (1967)). They might compute faster, perhaps, or use less memory, or their instruction set might be smaller, but they cannot compute more powerfully (i.e. more mathematical functions). (Recall that the Church–Turing thesis *hypothesizes* this to be true for any kind of machine: that anything that can be "computed" can be computed by some Turing machine.)

A Turing machine is equivalent to a pushdown automaton that has been made more flexible and concise by relaxing the last-in-first-out requirement of its stack.

At the other extreme, some very simple models turn out to be Turing-equivalent, i.e. to have the same computational power as the Turing machine model.

Common equivalent models are the multi-tape Turing machine, multi-track Turing machine, machines with input and output, and the *non-deterministic* Turing machine (NDTM) as opposed to the *deterministic* Turing machine (DTM) for which the action table has at most one entry for each combination of symbol and state.

Read-only, right-moving Turing machines are equivalent to NDFAs (as well as DFAs by conversion using the NDFA to DFA conversion algorithm).

For practical and didactical intentions the equivalent register machine can be used as a usual assembly programming language.

Choice c-machines, Oracle o-machines

Early in his paper (1936) Turing makes a distinction between an "automatic machine"—its "motion ... completely determined by the configuration" and a "choice machine":

...whose motion is only partially determined by the configuration ... When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems.

—*Undecidable*, p. 118

Turing (1936) does not elaborate further except in a footnote in which he describes how to use an a-machine to "find all the provable formulae of the [Hilbert] calculus" rather than use a choice machine. He "suppose[s] that the choices are always between two possibilities 0 and 1. Each proof will then be determined by a sequence of choices i_1, i_2, \dots, i_n ($i_1 = 0$ or 1, $i_2 = 0$ or 1, ..., $i_n = 0$ or 1), and hence the number $2^n + i_1 2^{n-1} + i_2 2^{n-2} + \dots + i_n$ completely determines the proof. The automatic machine carries out successively proof 1, proof 2, proof 3, ..." (Footnote ‡, *Undecidable*, p. 138)

This is indeed the technique by which a deterministic (i.e. a-) Turing machine can be used to mimic the action of a nondeterministic Turing machine; Turing solved the matter in a footnote and appears to dismiss it from further consideration.

An oracle machine or o-machine is a Turing a-machine that pauses its computation at state "**o**" while, to complete its calculation, it "awaits the decision" of "the oracle"—an unspecified entity "apart from saying that it cannot be a machine" (Turing (1939), *Undecidable* p. 166–168). The concept is now actively used by mathematicians.

Universal Turing machines

As Turing wrote in *Undecidable*, p. 128 (italics added):

It is possible to invent a *single machine* which can be used to compute *any* computable sequence. If this machine **U** is supplied with the tape on the beginning of which is written the string of quintuples separated by semicolons of some computing machine **M**, then **U** will compute the same sequence as **M**.

This finding is now taken for granted, but at the time (1936) it was considered astonishing. The model of computation that Turing called his "universal machine"—"U" for short—is considered by some (cf Davis (2000)) to have been the fundamental theoretical breakthrough that led to the notion of the Stored-program computer.

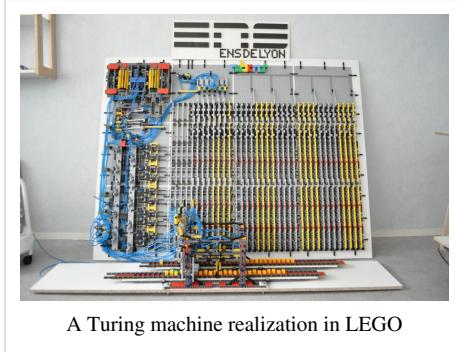
Turing's paper ... contains, in essence, the invention of the modern computer and some of the programming techniques that accompanied it.

—Minsky (1967), p. 104

In terms of computational complexity, a multi-tape universal Turing machine need only be slower by logarithmic factor compared to the machines it simulates. This result was obtained in 1966 by F. C. Hennie and R. E. Stearns. (Arora and Barak, 2009, theorem 1.9)

Comparison with real machines

It is often said that Turing machines, unlike simpler automata, are as powerful as real machines, and are able to execute any operation that a real program can. What is missed in this statement is that, because a real machine can only be in finitely many *configurations*, in fact this "real machine" is nothing but a linear bounded automaton. On the other hand, Turing machines are equivalent to machines that have an unlimited amount of storage space for their computations. In fact, Turing machines are not intended to model computers, but rather they are intended to model computation itself; historically, computers, which compute only on their (fixed) internal storage, were developed only later.



A Turing machine realization in LEGO

There are a number of ways to explain why Turing machines are useful models of real computers:

1. Anything a real computer can compute, a Turing machine can also compute. For example: "A Turing machine can simulate any type of subroutine found in programming languages, including recursive procedures and any of the known parameter-passing mechanisms" (Hopcroft and Ullman p. 157). A large enough FSA can also model any real computer, disregarding IO. Thus, a statement about the limitations of Turing machines will also apply to real computers.
2. The difference lies only with the ability of a Turing machine to manipulate an unbounded amount of data. However, given a finite amount of time, a Turing machine (like a real machine) can only manipulate a finite amount of data.
3. Like a Turing machine, a real machine can have its storage space enlarged as needed, by acquiring more disks or other storage media. If the supply of these runs short, the Turing machine may become less useful as a model. But the fact is that neither Turing machines nor real machines need astronomical amounts of storage space in order to perform useful computation. The processing time required is usually much more of a problem.
4. Descriptions of real machine programs using simpler abstract models are often much more complex than descriptions using Turing machines. For example, a Turing machine describing an algorithm may have a few hundred states, while the equivalent deterministic finite automaton on a given real machine has quadrillions. This makes the DFA representation infeasible to analyze.

5. Turing machines describe algorithms independent of how much memory they use. There is a limit to the memory possessed by any current machine, but this limit can rise arbitrarily in time. Turing machines allow us to make statements about algorithms which will (theoretically) hold forever, regardless of advances in *conventional* computing machine architecture.
6. Turing machines simplify the statement of algorithms. Algorithms running on Turing-equivalent abstract machines are usually more general than their counterparts running on real machines, because they have arbitrary-precision data types available and never have to deal with unexpected conditions (including, but not limited to, running out of memory).

One way in which Turing machines are a poor model for programs is that many real programs, such as operating systems and word processors, are written to receive unbounded input over time, and therefore do not halt. Turing machines do not model such ongoing computation well (but can still model portions of it, such as individual procedures).

Limitations of Turing machines

Computational complexity theory

A limitation of Turing machines is that they do not model the strengths of a particular arrangement well. For instance, modern stored-program computers are actually instances of a more specific form of abstract machine known as the random access stored program machine or RASP machine model. Like the Universal Turing machine the RASP stores its "program" in "memory" external to its finite-state machine's "instructions". Unlike the universal Turing machine, the RASP has an infinite number of distinguishable, numbered but unbounded "registers"—memory "cells" that can contain any integer (cf. Elgot and Robinson (1964), Hartmanis (1971), and in particular Cook-Rechow (1973); references at random access machine). The RASP's finite-state machine is equipped with the capability for indirect addressing (e.g. the contents of one register can be used as an address to specify another register); thus the RASP's "program" can address any register in the register-sequence. The upshot of this distinction is that there are computational optimizations that can be performed based on the memory indices, which are not possible in a general Turing machine; thus when Turing machines are used as the basis for bounding running times, a 'false lower bound' can be proven on certain algorithms' running times (due to the false simplifying assumption of a Turing machine). An example of this is binary search, an algorithm that can be shown to perform more quickly when using the RASP model of computation rather than the Turing machine model.

Concurrency

Another limitation of Turing machines is that they do not model concurrency well. For example, there is a bound on the size of integer that can be computed by an always-halting nondeterministic Turing machine starting on a blank tape. (See article on unbounded nondeterminism.) By contrast, there are always-halting concurrent systems with no inputs that can compute an integer of unbounded size. (A process can be created with local storage that is initialized with a count of 0 that concurrently sends itself both a stop and a go message. When it receives a go message, it increments its count by 1 and sends itself a go message. When it receives a stop message, it stops with an unbounded number in its local storage.)

History

They were described in 1936 by Alan Turing.

Historical background: computational machinery

Robin Gandy (1919–1995)—a student of Alan Turing (1912–1954) and his lifelong friend—traces the lineage of the notion of "calculating machine" back to Babbage (circa 1834) and actually proposes "Babbage's Thesis":

That the whole of development and operations of analysis are now capable of being executed by machinery.

—(italics in Babbage as cited by Gandy, p. 54)

Gandy's analysis of Babbage's Analytical Engine describes the following five operations (cf p. 52–53):

1. The arithmetic functions $+$, $-$, \times where $-$ indicates "proper" subtraction $x - y = 0$ if $y \geq x$
2. Any sequence of operations is an operation
3. Iteration of an operation (repeating n times an operation P)
4. Conditional iteration (repeating n times an operation P conditional on the "success" of test T)
5. Conditional transfer (i.e. conditional "goto").

Gandy states that "the functions which can be calculated by (1), (2), and (4) are precisely those which are Turing computable." (p. 53). He cites other proposals for "universal calculating machines" included those of Percy Ludgate (1909), Leonardo Torres y Quevedo (1914), Maurice d'Ocagne (1922), Louis Couffignal (1933), Vannevar Bush (1936), Howard Aiken (1937). However:

... the emphasis is on programming a fixed iterable sequence of arithmetical operations. The fundamental importance of conditional iteration and conditional transfer for a general theory of calculating machines is not recognized ...

—Gandy p. 55

The Entscheidungsproblem (the "decision problem"): Hilbert's tenth question of 1900

With regards to Hilbert's problems posed by the famous mathematician David Hilbert in 1900, an aspect of problem #10 had been floating about for almost 30 years before it was framed precisely. Hilbert's original expression for #10 is as follows:

10. Determination of the solvability of a Diophantine equation. Given a Diophantine equation with any number of unknown quantities and with rational integral coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

The Entscheidungsproblem [decision problem for first-order logic] is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity or satisfiability ... The Entscheidungsproblem must be considered the main problem of mathematical logic.

—quoted, with this translation and the original German, in Dershowitz and Gurevich, 2008

By 1922, this notion of "Entscheidungsproblem" had developed a bit, and H. Behmann stated that

... most general form of the Entscheidungsproblem [is] as follows:

A quite definite generally applicable prescription is required which will allow one to decide in a finite number of steps the truth or falsity of a given purely logical assertion ...

—Gandy p. 57, quoting Behmann

Behmann remarks that ... the general problem is equivalent to the problem of deciding which mathematical propositions are true.

—*ibid.*

If one were able to solve the Entscheidungsproblem then one would have a "procedure for solving many (or even all) mathematical problems".

—*ibid.*, p. 92

By the 1928 international congress of mathematicians Hilbert "made his questions quite precise. First, was mathematics *complete* ... Second, was mathematics *consistent* ... And thirdly, was mathematics *decidable*?" (Hodges p. 91, Hawking p. 1121). The first two questions were answered in 1930 by Kurt Gödel at the very same meeting where Hilbert delivered his retirement speech (much to the chagrin of Hilbert); the third—the Entscheidungsproblem—had to wait until the mid-1930s.

The problem was that an answer first required a precise definition of "*definite general applicable prescription*", which Princeton professor Alonzo Church would come to call "*effective calculability*", and in 1928 no such definition existed. But over the next 6–7 years Emil Post developed his definition of a worker moving from room to room writing and erasing marks per a list of instructions (Post 1936), as did Church and his two students Stephen Kleene and J. B. Rosser by use of Church's lambda-calculus and Gödel's recursion theory (1934). Church's paper (published 15 April 1936) showed that the Entscheidungsproblem was indeed "*undecidable*" and beat Turing to the punch by almost a year (Turing's paper submitted 28 May 1936, published January 1937). In the meantime, Emil Post submitted a brief paper in the fall of 1936, so Turing at least had priority over Post. While Church refereed Turing's paper, Turing had time to study Church's paper and add an Appendix where he sketched a proof that Church's lambda-calculus and his machines would compute the same functions.

But what Church had done was something rather different, and in a certain sense weaker. ... the Turing construction was more direct, and provided an argument from first principles, closing the gap in Church's demonstration.

—Hodges p. 112

And Post had only proposed a definition of calculability and criticized Church's "definition", but had proved nothing.

Alan Turing's a- (automatic-)machine

In the spring of 1935 Turing as a young Master's student at King's College Cambridge, UK, took on the challenge; he had been stimulated by the lectures of the logician M. H. A. Newman "and learned from them of Gödel's work and the Entscheidungsproblem ... Newman used the word 'mechanical' ... In his obituary of Turing 1955 Newman writes:

To the question 'what is a "mechanical" process?' Turing returned the characteristic answer 'Something that can be done by a machine' and he embarked on the highly congenial task of analysing the general notion of a computing machine.

—Gandy, p. 74

Gandy states that:

I suppose, but do not know, that Turing, right from the start of his work, had as his goal a proof of the undecidability of the Entscheidungsproblem. He told me that the 'main idea' of the paper came to him when he was lying in Grantchester meadows in the summer of 1935. The 'main idea' might have either been his analysis of computation or his realization that there was a universal machine, and so a diagonal argument to prove unsolvability.

—*ibid.*, p. 76

While Gandy believed that Newman's statement above is "misleading", this opinion is not shared by all. Turing had a lifelong interest in machines: "Alan had dreamt of inventing typewriters as a boy; [his mother] Mrs. Turing had a typewriter; and he could well have begun by asking himself what was meant by calling a typewriter 'mechanical'" (Hodges p. 96). While at Princeton pursuing his PhD, Turing built a Boolean-logic multiplier (see below). His PhD thesis, titled "Systems of Logic Based on Ordinals", contains the following definition of "a computable function":

It was stated above that 'a function is effectively calculable if its values can be found by some purely mechanical process'. We may take this statement literally, understanding by a purely mechanical process one which could be carried out by a machine. It is possible to give a mathematical description, in a certain normal form, of the structures of these machines. The development of these ideas leads to the author's definition of a computable function, and to an identification of computability with effective calculability. It is not difficult, though somewhat laborious, to prove that these three definitions [the 3rd is the λ -calculus] are equivalent.

—Turing (1939) in *The Undecidable*, p. 160

When Turing returned to the UK he ultimately became jointly responsible for breaking the German secret codes created by encryption machines called "The Enigma"; he also became involved in the design of the ACE (Automatic Computing Engine), "[Turing's] ACE proposal was effectively self-contained, and its roots lay not in the EDVAC [the USA's initiative], but in his own universal machine" (Hodges p. 318). Arguments still continue concerning the origin and nature of what has been named by Kleene (1952) Turing's Thesis. But what Turing *did prove* with his computational-machine model appears in his paper *On Computable Numbers, With an Application to the Entscheidungsproblem* (1937):

[that] the Hilbert Entscheidungsproblem can have no solution ... I propose, therefore to show that there can be no general process for determining whether a given formula U of the functional calculus K is provable, i.e. that there can be no machine which, supplied with any one U of these formulae, will eventually say whether U is provable.

—from Turing's paper as reprinted in *The Undecidable*, p. 145

Turing's example (his second proof): If one is to ask for a general procedure to tell us: "Does this machine ever print 0?", the question is "undecidable".

1937–1970: The "digital computer", the birth of "computer science"

In 1937, while at Princeton working on his PhD thesis, Turing built a digital (Boolean-logic) multiplier from scratch, making his own electromechanical relays (Hodges p. 138). "Alan's task was to embody the logical design of a Turing machine in a network of relay-operated switches ..." (Hodges p. 138). While Turing might have been just initially curious and experimenting, quite-earnest work in the same direction was going in Germany (Konrad Zuse (1938)), and in the United States (Howard Aiken) and George Stibitz (1937); the fruits of their labors were used by the Axis and Allied military in World War II (cf Hodges p. 298–299). In the early to mid-1950s Hao Wang and Marvin Minsky reduced the Turing machine to a simpler form (a precursor to the Post-Turing machine of Martin Davis); simultaneously European researchers were reducing the new-fangled electronic computer to a computer-like theoretical object equivalent to what was now being called a "Turing machine". In the late 1950s and early 1960s, the coincidentally parallel developments of Melzak and Lambek (1961), Minsky (1961), and Shepherdson and Sturgis (1961) carried the European work further and reduced the Turing machine to a more friendly, computer-like abstract model called the counter machine; Elgot and Robinson (1964), Hartmanis (1971), Cook and Reckhow (1973) carried this work even further with the register machine and random access machine models—but basically all are just multi-tape Turing machines with an arithmetic-like instruction set.

1970–present: the Turing machine as a model of computation

Today the counter, register and random-access machines and their sire the Turing machine continue to be the models of choice for theorists investigating questions in the theory of computation. In particular, computational complexity theory makes use of the Turing machine:

Depending on the objects one likes to manipulate in the computations (numbers like nonnegative integers or alphanumeric strings), two models have obtained a dominant position in machine-based complexity theory:

the off-line multitape Turing machine..., which represents the standard model for string-oriented computation, and

the random access machine (RAM) as introduced by Cook and Reckhow ..., which models the idealized Von Neumann style computer.

—van Emde Boas 1990:4

Only in the related area of analysis of algorithms this role is taken over by the RAM model.

—van Emde Boas 1990:16

Notes

- [1] The idea came to him in mid-1935 (perhaps, see more in the History section) after a question posed by M. H. A. Newman in his lectures: "Was there a definite method, or as Newman put it, a *mechanical process* which could be applied to a mathematical statement, and which would come up with the answer as to whether it was provable" (Hodges 1983:93). Turing submitted his paper on 31 May 1936 to the London Mathematical Society for its *Proceedings* (cf Hodges 1983:112), but it was *published* in early 1937 and offprints were available in February 1937 (cf Hodges 1983:129).
- [2] See the definition of "innings" on Wiktionary

References

Primary literature, reprints, and compilations

- B. Jack Copeland ed. (2004), *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*, Clarendon Press (Oxford University Press), Oxford UK, ISBN 0-19-825079-7. Contains the Turing papers plus a draft letter to Emil Post re his criticism of "Turing's convention", and Donald W. Davies' *Corrections to Turing's Universal Computing Machine*
- Martin Davis (ed.) (1965), *The Undecidable*, Raven Press, Hewlett, NY.
- Emil Post (1936), "Finite Combinatory Processes—Formulation 1", *Journal of Symbolic Logic*, 1, 103–105, 1936. Reprinted in *The Undecidable* pp. 289ff.
- Emil Post (1947), "Recursive Unsolvability of a Problem of Thue", *Journal of Symbolic Logic*, vol. 12, pp. 1–11. Reprinted in *The Undecidable* pp. 293ff. In the Appendix of this paper Post comments on and gives corrections to Turing's paper of 1936–1937. In particular see the footnotes 11 with corrections to the universal computing machine coding and footnote 14 with comments on Turing's first and second proofs.
- Turing, A.M. (1936). "On Computable Numbers, with an Application to the Entscheidungs problem". *Proceedings of the London Mathematical Society*. 2 **42**: 230–65. 1937. doi:10.1112/plms/s2-42.1.230. (and Turing, A.M. (1938). "On Computable Numbers, with an Application to the Entscheidungsproblem: A correction". *Proceedings of the London Mathematical Society*. 2 **43** (6): 544–6. 1937. doi:10.1112/plms/s2-43.6.544.). Reprinted in many collections, e.g. in *The Undecidable* pp. 115–154; available on the web in many places, e.g. at Scribd (<http://www.scribd.com/doc/2937039/Alan-M-Turing-On-Computable-Numbers>).
- Alan Turing, 1948, "Intelligent Machinery." Reprinted in "Cybernetics: Key Papers." Ed. C.R. Evans and A.D.J. Robertson. Baltimore: University Park Press, 1968. p. 31.
- F. C. Hennie and R. E. Stearns. *Two-tape simulation of multitape Turing machines*. JACM, 13(4):533–546, 1966.

Computability theory

- Boolos, George; Richard Jeffrey (1989, 1999). *Computability and Logic* (3rd ed.). Cambridge UK: Cambridge University Press. ISBN 0-521-20402-X.
- Boolos, George; John Burgess, Richard Jeffrey, (2002). *Computability and Logic* (4th ed.). Cambridge UK: Cambridge University Press. ISBN 0-521-00758-5 (pb.). Some parts have been significantly rewritten by Burgess. Presentation of Turing machines in context of Lambek "abacus machines" (cf Register machine) and recursive functions, showing their equivalence.
- Taylor L. Booth (1967), *Sequential Machines and Automata Theory*, John Wiley and Sons, Inc., New York. Graduate level engineering text; ranges over a wide variety of topics, Chapter IX *Turing Machines* includes some recursion theory.
- Martin Davis (1958). *Computability and Unsolvability*. McGraw-Hill Book Company, Inc, New York.. On pages 12–20 he gives examples of 5-tuple tables for Addition, The Successor Function, Subtraction ($x \geq y$), Proper Subtraction (0 if $x < y$), The Identity Function and various identity functions, and Multiplication.
- Davis, Martin; Ron Sigal, Elaine J. Weyuker (1994). *Computability, Complexity, and Languages and Logic: Fundamentals of Theoretical Computer Science* (2nd ed.). San Diego: Academic Press, Harcourt, Brace & Company. ISBN 0-12-206382-1.
- Hennie, Fredrick (1977). *Introduction to Computability*. Addison–Wesley, Reading, Mass.. On pages 90–103 Hennie discusses the UTM with examples and flow-charts, but no actual 'code'.
- John Hopcroft and Jeffrey Ullman, (1979). *Introduction to Automata Theory, Languages and Computation* (1st ed.). Addison–Wesley, Reading Mass. ISBN 0-201-02988-X.. A difficult book. Centered around the issues of machine-interpretation of "languages", NP-completeness, etc.
- Hopcroft, John E.; Rajeev Motwani, Jeffrey D. Ullman (2001). *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Reading Mass: Addison–Wesley. ISBN 0-201-44124-1. Distinctly different and less intimidating than the first edition.
- Stephen Kleene (1952), *Introduction to Metamathematics*, North–Holland Publishing Company, Amsterdam Netherlands, 10th impression (with corrections of 6th reprint 1971). Graduate level text; most of Chapter XIII *Computable functions* is on Turing machine proofs of computability of recursive functions, etc.
- Knuth, Donald E. (1973). *Volume I/Fundamental Algorithms: The Art of computer Programming* (2nd ed.). Reading, Mass.: Addison–Wesley Publishing Company.. With reference to the role of Turing machines in the development of computation (both hardware and software) see 1.4.5 *History and Bibliography* pp. 225ff and 2.6 *History and Bibliography* pp. 456ff.
- Zohar Manna, 1974, *Mathematical Theory of Computation*. Reprinted, Dover, 2003. ISBN 978-0-486-43238-0
- Marvin Minsky, *Computation: Finite and Infinite Machines*, Prentice–Hall, Inc., N.J., 1967. See Chapter 8, Section 8.2 "Unsolvability of the Halting Problem." Excellent, i.e. relatively readable, sometimes funny.
- Christos Papadimitriou (1993). *Computational Complexity* (1st ed.). Addison Wesley. ISBN 0-201-53082-1. Chapter 2: Turing machines, pp. 19–56.
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X. Chapter 3: The Church–Turing Thesis, pp. 125–149.
- Stone, Harold S. (1972). *Introduction to Computer Organization and Data Structures* (1st ed.). New York: McGraw–Hill Book Company. ISBN 0-07-061726-0.
- Peter van Emde Boas 1990, *Machine Models and Simulations*, pp. 3–66, in Jan van Leeuwen, ed., *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, The MIT Press/Elsevier, [place?], ISBN 0-444-88071-2 (Volume A). QA76.H279 1990. Valuable survey, with 141 references.

Church's thesis

- Nachum Dershowitz; Yuri Gurevich (September 2008). "A natural axiomatization of computability and proof of Church's Thesis" (<http://research.microsoft.com/en-us/um/people/gurevich/Opera/188.pdf>). *Bulletin of Symbolic Logic* 14 (3). Retrieved 2008-10-15.
- Roger Penrose (1989, 1990). *The Emperor's New Mind* (2nd ed.). Oxford University Press, New York. ISBN 0-19-851973-7.

Small Turing machines

- Rogozhin, Yurii, 1998, " A Universal Turing Machine with 22 States and 2 Symbols (http://web.archive.org/web/20050308141040/http://www.imt.ro/Romjist/Volum1/Vol1_3/turing.htm)", *Romanian Journal Of Information Science and Technology*, 1(3), 259–265, 1998. (surveys known results about small universal Turing machines)
- Stephen Wolfram, 2002, *A New Kind of Science* (<http://www.wolframscience.com/nksonline/page-707>), Wolfram Media, ISBN 1-57955-008-8
- Brunfiel, Geoff, Student snags maths prize (<http://www.nature.com/news/2007/071024/full/news.2007.190.html>), *Nature*, October 24. 2007.
- Jim Giles (2007), Simplest 'universal computer' wins student \$25,000 (<http://technology.newscientist.com/article/dn12826-simplest-universal-computer-wins-student-25000.html>), *New Scientist*, October 24, 2007.
- Alex Smith, Universality of Wolfram's 2, 3 Turing Machine (<http://www.wolframscience.com/prizes/tm23/TM23Proof.pdf>), Submission for the Wolfram 2, 3 Turing Machine Research Prize.
- Vaughan Pratt, 2007, " Simple Turing machines, Universality, Encodings, etc. (<http://cs.nyu.edu/pipermail/fom/2007-October/012156.html>)", FOM email list. October 29, 2007.
- Martin Davis, 2007, " Smallest universal machine (<http://cs.nyu.edu/pipermail/fom/2007-October/012132.html>)", and Definition of universal Turing machine (<http://cs.nyu.edu/pipermail/fom/2007-October/012145.html>) FOM email list. October 26–27, 2007.
- Alasdair Urquhart, 2007 " Smallest universal machine (<http://cs.nyu.edu/pipermail/fom/2007-October/012140.html>)", FOM email list. October 26, 2007.
- Hector Zenil (Wolfram Research), 2007 " smallest universal machine (<http://cs.nyu.edu/pipermail/fom/2007-October/012163.html>)", FOM email list. October 29, 2007.
- Todd Rowland, 2007, " Confusion on FOM (<http://forum.wolframscience.com/showthread.php?s=&threadid=1472>)", Wolfram Science message board, October 30, 2007.

Other

- Martin Davis (2000). *Engines of Logic: Mathematicians and the origin of the Computer* (1st ed.). W. W. Norton & Company, New York. ISBN 0-393-32229-7 pbk..
- Robin Gandy, "The Confluence of Ideas in 1936", pp. 51–102 in Rolf Herken, see below.
- Stephen Hawking (editor), 2005, *God Created the Integers: The Mathematical Breakthroughs that Changed History*, Running Press, Philadelphia, ISBN 978-0-7624-1922-7. Includes Turing's 1936–1937 paper, with brief commentary and biography of Turing as written by Hawking.
- Rolf Herken (1995). *The Universal Turing Machine—A Half-Century Survey*. Springer Verlag. ISBN 3-211-82637-8.
- Andrew Hodges, *Alan Turing: The Enigma*, Simon and Schuster, New York. Cf Chapter "The Spirit of Truth" for a history leading to, and a discussion of, his proof.
- Ivars Peterson (1988). *The Mathematical Tourist: Snapshots of Modern Mathematics* (1st ed.). W. H. Freeman and Company, New York. ISBN 0-7167-2064-7 (pbk.).

- Paul Strathern (1997). *Turing and the Computer—The Big Idea*. Anchor Books/Doubleday. ISBN 0-385-49243-X.
- Hao Wang, "A variant to Turing's theory of computing machines", *Journal of the Association for Computing Machinery* (JACM) 4, 63–92 (1957).
- Charles Petzold, Petzold, Charles, *The Annotated Turing* (<http://www.theannotatedturing.com/>), John Wiley & Sons, Inc., ISBN 0-470-22905-5
- Arora, Sanjeev; Barak, Boaz, "Complexity Theory: A Modern Approach" (<http://www.cs.princeton.edu/theory/complexity/>), Cambridge University Press, 2009, ISBN 978-0-521-42426-4, section 1.4, "Machines as strings and the universal Turing machine" and 1.7, "Proof of theorem 1.9"
- Isaiah Pinchas Kantorovitz, "A note on turing machine computability of rule driven systems", ACM SIGACT News December 2005.

External links

- Turing Machine on Stanford Encyclopedia of Philosophy (<http://plato.stanford.edu/entries/turing-machine/>)
- Detailed info on the Church–Turing Hypothesis (<http://plato.stanford.edu/entries/church-turing/>) (Stanford Encyclopedia of Philosophy)
- Turing Machine-Like Models (http://www.weizmann.ac.il/mathusers/lbn/new_pages/Research_Turing.html) in Molecular Biology, to understand life mechanisms with a DNA-tape processor.
- The Turing machine (http://www.SaschaSeidel.de/html/programmierung/download_The_Turing_machine.php)—Summary about the Turing machine, its functionality and historical facts
- The Wolfram 2,3 Turing Machine Research Prize (<http://www.wolframscience.com/prizes/tm23/>)—Stephen Wolfram's \$25,000 prize for the proof or disproof of the universality of the potentially smallest universal Turing Machine. The contest has ended, with the proof affirming the machine's universality.
- " Turing Machine Causal Networks (<http://demonstrations.wolfram.com/TuringMachineCausalNetworks/>)" by Enrique Zeleny, Wolfram Demonstrations Project.
- Turing Machines (http://www.dmoz.org/Computers/Computer_Science/Theoretical/Automata_Theory/Turing_Machines/) at the Open Directory Project
- Purely mechanical Turing Machine (<http://www.turing2012.fr/?p=530&lang=en>)

Recursively enumerable language

In mathematics, logic and computer science, a formal language is called **recursively enumerable** (also **recognizable**, **partially decidable** or **Turing-acceptable**) if it is a recursively enumerable subset in the set of all possible words over the alphabet of the language, i.e., if there exists a Turing machine which will enumerate all valid strings of the language.

Recursively enumerable languages are known as **type-0** languages in the Chomsky hierarchy of formal languages. All regular, context-free, context-sensitive and recursive languages are recursively enumerable.

The class of all recursively enumerable languages is called **RE**.

Definitions

There exist three equivalent major definitions for the concept of a recursively enumerable language.

1. A recursively enumerable language is a recursively enumerable subset in the set of all possible words over the alphabet of the language.
2. A recursively enumerable language is a formal language for which there exists a Turing machine (or other computable function) which will enumerate all valid strings of the language. Note that, if the language is infinite, the enumerating algorithm provided can be chosen so that it avoids repetitions, since we can test whether the string produced for number n is "already" produced for a number which is less than n . If it already is produced, use the output for input $n+1$ instead (recursively), but again, test whether it is "new".
3. A recursively enumerable language is a formal language for which there exists a Turing machine (or other computable function) that will halt and accept when presented with any string in the language as input but may either halt and reject or loop forever when presented with a string not in the language. Contrast this to recursive languages, which require that the Turing machine halts in all cases.

All regular, context-free, context-sensitive and recursive languages are recursively enumerable.

Post's theorem shows that **RE**, together with its complement co-**RE**, correspond to the first level of the arithmetical hierarchy.

Example

The Halting problem is r.e. but not recursive. Indeed one can run the Turing Machine and accept if the machine halts, hence it is r.e. On the other hand the problem is undecidable.

Some other r.e. languages that are not recursive:

- Post correspondence problem
- Mortality (computability theory)
- Entscheidungsproblem

Closure properties

Recursively enumerable languages are closed under the following operations. That is, if L and P are two recursively enumerable languages, then the following languages are recursively enumerable as well:

- the Kleene star L^* of L
- the concatenation $L \circ P$ of L and P
- the union $L \cup P$
- the intersection $L \cap P$.

Note that recursively enumerable languages are not closed under set difference or complementation. The set difference $L - P$ may or may not be recursively enumerable. If L is recursively enumerable, then the complement of L is recursively enumerable if and only if L is also recursive.

References

- Sipser, M. (1996), *Introduction to the Theory of Computation*, PWS Publishing Co.
- Kozen, D.C. (1997), *Automata and Computability*, Springer.

External links

- Complexity Zoo: Class RE ^[1]
- Lecture slides ^[2]

References

- [1] http://complexity-zoo.net/zoo/wiki/Complexity_Zoo:R#re
[2] <http://www.cs.colostate.edu/~massey/Teaching/cs301/RestrictedAccess/Slides/301lecture23.pdf>

Recursive language

In mathematics, logic and computer science, a formal language (a set of finite sequences of symbols taken from a fixed alphabet) is called **recursive** if it is a recursive subset of the set of all possible finite sequences over the alphabet of the language. Equivalently, a formal language is recursive if there exists a total Turing machine (a Turing machine which halts for every given input) which when given a finite sequence of symbols from the alphabet of the language as input accepts exactly those words from the alphabet of the language that are part of the language and rejects all other words. Recursive languages are also called **decidable**.

The concept of **decidability** may be extended to other models of computation. For example one may speak of languages decidable on a non-deterministic Turing machine. Therefore whenever an ambiguity is possible, the synonym for "recursive language" used is **Turing-decidable language**, rather than simply "decidable".

The class of all recursive languages is often called **R**, although this name is also used for the class RP.

This type of language was not defined in the Chomsky hierarchy of (Chomsky 1959). All recursive languages are also recursively enumerable. All regular, context-free and context-sensitive languages are recursive.

Definitions

There are two equivalent major definitions for the concept of a recursive language:

1. A recursive formal language is a recursive subset in the set of all possible words over the alphabet of the language.
2. A recursive language is a formal language for which there exists a Turing machine which will, when presented with any finite input string, halt and accept if the string is in the language, and halt and reject otherwise. The Turing machine always halts; it is known as a decider and is said to *decide* the recursive language.

By the second definition, any decision problem can be shown to be decidable by exhibiting an algorithm for it that terminates on all inputs. An undecidable problem is a problem that is not decidable.

Closure properties

Recursive languages are closed under the following operations. That is, if L and P are two recursive languages, then the following languages are recursive as well:

- The Kleene star L^*
- The image $\varphi(L)$ under an e-free homomorphism φ
- The concatenation $L \circ P$
- The union $L \cup P$
- The intersection $L \cap P$
- The complement of L
- The set difference $L - P$

The last property follows from the fact that the set difference can be expressed in terms of intersection and complement.

References

- Michael Sipser (1997). "Decidability". *Introduction to the Theory of Computation*. PWS Publishing. pp. 151–170. ISBN 0-534-94728-X
- Chomsky, Noam (1959). "On certain formal properties of grammars". *Information and Control* 2 (2): 137–167. doi:10.1016/S0019-9958(59)90362-6.

Machine that always halts

In computability theory, a **machine that always halts**—also called a **decider** (Sipser, 1996) or a **total Turing machine** (Kozen, 1997)—is a Turing machine that halts for every input.

Because it always halts, the machine is able to *decide* whether a given string is a member of a formal language. The class of languages which can be decided by such machines is exactly the set of recursive languages. However, due to the Halting Problem, determining whether an arbitrary Turing machine halts on an arbitrary input is itself an undecidable decision problem.

Functions computable by total Turing machines

In practice, many functions of interest are computable by machines that always halt. A machine that uses only finite memory on any particular input can be forced to halt for every input by restricting its flow control capabilities so that no input will ever cause the machine to enter an infinite loop. As a trivial example, a machine implementing a finitary decision tree will always halt.

It is not required that the machine be entirely free of looping capabilities, however, to guarantee halting. If we restrict loops to be of a predictably finite size (not unlike the FOR loop in BASIC), we can express all of the primitive recursive functions (Meyer and Ritchie, 1967). An example of such a machine is provided by the toy programming language PL-{GOTO} of Brainerd and Landweber (1974).

We can further define a programming language in which we can ensure that even more sophisticated functions always halt. For example, the Ackermann function, which is not primitive recursive, nevertheless is a total computable function computable by a term rewriting system with a reduction ordering on its arguments (Ohlebusch, 2002, pp.67).

Relationship to partial Turing machines

A general Turing machine will compute a partial function. Two questions can be asked about the relationship between partial Turing machines and total Turing machines:

1. Can every partial function computable by a partial Turing machine be extended (that is, have its domain enlarged) to become a total computable function?
2. Is it possible to change the definition of a Turing machine so that a particular class of total Turing machines, computing all the total computable functions, can be found?

The answer to each of these questions is no.

The following theorem shows that the functions computable by machines that always halt do not include extensions of all partial computable functions, which implies the first question above has a negative answer. This fact is closely related to the algorithmic unsolvability of the Halting problem.

Theorem. There are Turing computable partial functions that have no extension to a total Turing computable function. In particular, the partial function f defined so that $f(n) = m$ if and only if the Turing machine with index n halts on input 0 with output m has no extension to a total computable function.

The proof proceeds by contradiction. If g were a total computable function extending f then g would be computable by some Turing machine; fix e as the index of such a machine. Build a Turing machine M , using Kleene's recursion theorem, which on input 0 simulates the machine with index e running on an index n_M for M (thus the machine M can produce an index of itself; this is the role of the recursion theorem). By assumption, this simulation will eventually return an answer. Define M so that if $g(n_M) = m$ then the return value of M is $m + 1$. Thus $f(n_M)$, the true return value of M on input 0 , will not equal $g(n_M)$. This contradicts the assumption that g extends f .

The second question asks, in essence, whether there is another reasonable model of computation which computes only total functions and computes all the total computable functions. Informally, if such a model existed then each of its computers could be simulated by a Turing machine. Thus if this new model of computation consisted of a sequence M_1, M_2, \dots of machines, there would be a recursively enumerable sequence T_1, T_2, \dots of Turing machines that compute total functions and so that every total computable function is computable by one of the machines T_i . This is impossible, because a machine T could be constructed such that on input i the machine T returns $T_i(i) + 1$. This machine cannot be equivalent to any machine T on the list: suppose it were on the list at index j . Then $T_j(j) = T_j(j) + 1$, which does not return an integer result. Therefore it can't be total, but the function by construction must be total (if total functions are recursively enumerable, then this function can be constructed), so we have a contradiction. This shows that the second question has a negative answer.

The set of indices of total Turing machines

The decision problem of whether the Turing machine with index e will halt on every input is not decidable. In fact, this problem is at level Π_2^0 of the arithmetical hierarchy. Thus this problem is strictly more difficult than the Halting problem, which asks whether the machine with index e halts on input 0 . Intuitively, this difference in unsolvability is because each instance of the "total machine" problem represents infinitely many instances of the Halting problem.

References

- Brainerd, W.S., Landweber, L.H. (1974), *Theory of Computation*, Wiley.
- Meyer, A.R., Ritchie, D.M. (1967), *The complexity of loop programs*, Proc. of the ACM National Meetings, 465.
- Sipser, M. (1996), *Introduction to the Theory of Computation*, PWS Publishing Co.
- Kozen, D.C. (1997), *Automata and Computability*, Springer.
- Ohlebusch, E. (2002), *Advanced Topics in Term Rewriting*, Springer.

Context-sensitive grammar

A **context-sensitive grammar (CSG)** is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and nonterminal symbols. Context-sensitive grammars are more general than context-free grammars but still orderly enough to be parsed by a linear bounded automaton.

The concept of context-sensitive grammar was introduced by Noam Chomsky in the 1950s as a way to describe the syntax of natural language where it is indeed often the case that a word may or may not be appropriate in a certain place depending upon the context. A formal language that can be described by a context-sensitive grammar is called a context-sensitive language.

Formal definition

A formal grammar $G = (N, \Sigma, P, S)$ (this is the same as $G = (V, T, P, S)$, where N/V is the Non-terminal Variable, and Σ/T is the Terminal) is context-sensitive if all rules in P are of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

where $A \in N$ (i.e., A is a single nonterminal), $\alpha, \beta \in (N \cup \Sigma)^*$ (i.e., α and β are strings of nonterminals and terminals) and $\gamma \in (N \cup \Sigma)^+$ (i.e., γ is a nonempty string of nonterminals and terminals).

Some definitions also add that for any production rule of the form $u \rightarrow v$ of a context-sensitive grammar, it shall be true that $|u| \leq |v|$. Here $|u|$ and $|v|$ denote the length of the strings respectively.

In addition, a rule of the form

$$S \rightarrow \lambda \text{ provided } S \text{ does not appear on the right side of any rule}$$

where λ represents the empty string is permitted. The addition of the empty string allows the statement that the context sensitive languages are a proper superset of the context free languages, rather than having to make the weaker statement that all context free grammars with no $\rightarrow \lambda$ productions are also context sensitive grammars.

The name *context-sensitive* is explained by the α and β that form the context of A and determine whether A can be replaced with γ or not. This is different from a context-free grammar where the context of a nonterminal is not taken into consideration. (Indeed, every production of a context free grammar is of the form $V \rightarrow w$ where V is a *single* nonterminal symbol, and w is a string of terminals and/or nonterminals (w can be empty)).

If the possibility of adding the empty string to a language is added to the strings recognized by the noncontracting grammars (which can never include the empty string) then the languages in these two definitions are identical.

Examples

- This grammar generates the canonical non-context-free language $\{a^n b^n c^n \mid n \geq 1\}$:

1. $S \rightarrow aSBC$
2. $S \rightarrow aBC$
3. $CB \rightarrow HB$
4. $HB \rightarrow HC$
5. $HC \rightarrow BC$
6. $aB \rightarrow ab$
7. $bB \rightarrow bb$
8. $bC \rightarrow bc$
9. $cC \rightarrow cc$

The generation chain for aaa bbb ccc is:

S

$$\begin{aligned}
 &\Rightarrow_1 aSBC \\
 &\Rightarrow_1 aaSBCBC \\
 &\Rightarrow_2 aaaBCBCBC \\
 &\Rightarrow_3 aaaB\mathbf{H}BCBC \\
 &\Rightarrow_4 aaaB\mathbf{HCC}BC \\
 &\Rightarrow_5 aaaB\mathbf{BCC}BC \\
 &\Rightarrow_3 aaaBB\mathbf{CHBC} \\
 &\Rightarrow_4 aaaBB\mathbf{CHCC} \\
 &\Rightarrow_5 aaaBBC\mathbf{BCC} \\
 &\Rightarrow_3 aaaBB\mathbf{HBCC} \\
 &\Rightarrow_4 aaaBB\mathbf{HCCC} \\
 &\Rightarrow_5 aaaBB\mathbf{BBCCC} \\
 &\Rightarrow_6 aaabBB\mathbf{CCC} \\
 &\Rightarrow_7 aaabb\mathbf{BCCC} \\
 &\Rightarrow_7 aaabb\mathbf{bCCC} \\
 &\Rightarrow_8 aaabb\mathbf{bcCC} \\
 &\Rightarrow_9 aaabb\mathbf{bccC} \\
 &\Rightarrow_9 aaabb\mathbf{ccc}
 \end{aligned}$$

More complicated grammars can be used to parse $\{a^n b^n c^n d^n \mid n \geq 1\}$, and other languages with even more letters.

Normal forms

Every context-sensitive grammar which does not generate the empty string can be transformed into an equivalent one in Kuroda normal form. "Equivalent" here means that the two grammars generate the same language. The normal form will not in general be context-sensitive, but will be a noncontracting grammar.

Computational properties and uses

The decision problem that asks whether a certain string s belongs to the language of a certain context-sensitive grammar G , is PSPACE-complete. There are even some context-sensitive grammars whose fixed grammar recognition problem is PSPACE-complete.

The emptiness problem for context-sensitive grammars (given a context-sensitive grammar G , is $L(G) = \emptyset$?) is undecidable.

It has been shown that nearly all natural languages may in general be characterized by context-sensitive grammars, but the whole class of CSG's seems to be much bigger than natural languages. Worse yet, since the aforementioned decision problem for CSG's is PSPACE-complete, that makes them totally unworkable for practical use, as a polynomial-time algorithm for a PSPACE-complete problem would imply P=NP. Ongoing research on computational linguistics has focused on formulating other classes of languages that are "mildly context-sensitive" whose decision problems are feasible, such as tree-adjoining grammars, combinatory categorial grammars, coupled context-free languages, and linear context-free rewriting systems. The languages generated by these formalisms properly lie between the context-free and context-sensitive languages.

References

- Introduction to Languages and the Theory of Computation by John C. Martin McGraw Hill 1996 (2nd edition)

External links

- Earley Parsing for Context-Sensitive Grammars [1]

References

[1] <http://danielmattosroberts.com/earley/context-sensitive-earley.pdf>

Context-sensitive language

In theoretical computer science, a **context-sensitive language** is a formal language that can be defined by a context-sensitive grammar. That is one of the four types of grammars in the Chomsky hierarchy. Of the four, this is the least often used, in both theory and practice.

Computational properties

Computationally, a context-sensitive language is equivalent with a linear bounded nondeterministic Turing machine, also called a linear bounded automaton. That is a non-deterministic Turing machine with a tape of only kn cells, where n is the size of the input and k is a constant associated with the machine. This means that every formal language that can be decided by such a machine is a context-sensitive language, and every context-sensitive language can be decided by such a machine.

This set of languages is also known as **NLIN-SPACE**, because they can be accepted using linear space on a non-deterministic Turing machine. The class **LIN-SPACE** is defined the same, except using a deterministic Turing machine. Clearly **LIN-SPACE** is a subset of **NLIN-SPACE**, but it is not known whether **LIN-SPACE=NLIN-SPACE**. It is widely suspected they are not equal .

Examples

An example of a context-sensitive language that is not context-free is $L = \{ a^p : p \text{ is a prime number} \}$. L can be shown to be a context-sensitive language by constructing a linear bounded automaton which accepts L . The language can easily be shown to be neither regular nor context free by applying the respective pumping lemmas for each of the language classes to L .

Simpler example would be $L = \{ a^n b^n c^n : n \geq 1 \}$.

An example of recursive language that is not context-sensitive is any recursive language whose decision is an EXPSPACE-hard problem, say, the set of pairs of equivalent regular expressions with exponentiation.

Properties of context-sensitive languages

- The union, intersection, concatenation and kleene star of two context-sensitive languages is context-sensitive.
- The complement of a context-sensitive language is itself context-sensitive.
- Every context-free language is context-sensitive.
- Membership of a string in a language defined by an arbitrary context-sensitive grammar, or by an arbitrary deterministic context-sensitive grammar, is a PSPACE-complete problem.

References

- Sipser, M. (1996), *Introduction to the Theory of Computation*, PWS Publishing Co.
- Immerman, Neil (1988). "Nondeterministic space is closed under complementation" ^[1]. *SIAM J. Comput.* **17** (5): 935–938. doi:10.1137/0217058.

References

[1] <http://www.cs.umass.edu/~immerman/pub/space.pdf>

Context-free grammar

In formal language theory, a **context-free grammar (CFG)** is a formal grammar in which every production rule is of the form

$$V \rightarrow w$$

where V is a *single* nonterminal symbol, and w is a string of terminals and/or nonterminals (w can be empty).

The languages generated by context-free grammars are known as the context-free languages.

A formal grammar is considered "context free" when its production rules can be applied regardless of the context of a nonterminal.

Context-free grammars are important in linguistics for describing the structure of sentences and words in natural language, and in computer science for describing the structure of programming languages and other formal languages.

In linguistics, some authors use the term **phrase structure grammar** to refer to context-free grammars, whereby phrase structure grammars are distinct from dependency grammars. In computer science, a popular notation for context-free grammars is Backus–Naur Form, or *BNF*.

Background

Since the time of Pāṇini, at least, linguists have described the grammars of languages in terms of their block structure, and described how sentences are recursively built up from smaller phrases, and eventually individual words or word elements. An essential property of these block structures is that logical units never overlap. For example, the sentence:

John, whose blue car was in the garage, walked to the grocery store.

can be logically parenthesized as follows:

(John, ((whose blue car) (was (in the garage))), (walked (to (the grocery store)))).

A context-free grammar provides a simple and mathematically precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the "block structure" of sentences in a natural way. Its simplicity makes the formalism amenable to rigorous mathematical study. Important features of

natural language syntax such as agreement and reference are not part of the context-free grammar, but the basic recursive structure of sentences, the way in which clauses nest inside other clauses, and the way in which lists of adjectives and adverbs are swallowed by nouns and verbs, is described exactly.

The formalism of context-free grammars was developed in the mid-1950s by Noam Chomsky,^[1] and also their classification as a special type of formal grammar (which he called phrase-structure grammars).^[2] What Chomsky called a phrase structure grammar is also known now as a constituency grammar, whereby constituency grammars stand in contrast to dependency grammars. In Chomsky's generative grammar framework, the syntax of natural language was described by context-free rules combined with transformation rules.

Block structure was introduced into computer programming languages by the Algol project (1957–1960), which, as a consequence, also featured a context-free grammar to describe the resulting Algol syntax. This became a standard feature of computer languages, and the notation for grammars used in concrete descriptions of computer languages came to be known as Backus-Naur Form, after two members of the Algol language design committee.^[1] The "block structure" aspect that context-free grammars capture is so fundamental to grammar that the terms syntax and grammar are often identified with context-free grammar rules, especially in computer science. Formal constraints not captured by the grammar are then considered to be part of the "semantics" of the language.

Context-free grammars are simple enough to allow the construction of efficient parsing algorithms which, for a given string, determine whether and how it can be generated from the grammar. An Earley parser is an example of such an algorithm, while the widely used LR and LL parsers are simpler algorithms that deal only with more restrictive subsets of context-free grammars.

Formal definitions

A context-free grammar G is defined by the 4-tuple:^[3]

$G = (V, \Sigma, R, S)$ where

1. V is a finite set; each element $v \in V$ is called a *non-terminal character* or a *variable*. Each variable represents a different type of phrase or clause in the sentence. Variables are also sometimes called syntactic categories. Each variable defines a sub-language of the language defined by G .
2. Σ is a finite set of *terminals*, disjoint from V , which make up the actual content of the sentence. The set of terminals is the alphabet of the language defined by the grammar G .
3. R is a finite relation from V to $(V \cup \Sigma)^*$, where the asterisk represents the Kleene star operation. The members of R are called the (*rewrite*) *rules* or *productions* of the grammar. (also commonly symbolized by a P)
4. S is the start variable (or start symbol), used to represent the whole sentence (or program). It must be an element of V .

Production rule notation

A production rule in R is formalized mathematically as a pair $(\alpha, \beta) \in R$, where $\alpha \in V$ is a non-terminal and $\beta \in (V \cup \Sigma)^*$ is a string of variables and/or terminals; rather than using ordered pair notation, production rules are usually written using an arrow operator with α as its left hand side and β as its right hand side: $\alpha \rightarrow \beta$. It is allowed for β to be the empty string, and in this case it is customary to denote it by ϵ . The form $\alpha \rightarrow \epsilon$ is called an ϵ -production.^[4]

Rule application

For any strings $u, v \in (V \cup \Sigma)^*$, we say u yields v , written as $u \Rightarrow v$, if $\exists(\alpha, \beta) \in R$ with $\alpha \in V$ and $u_1, u_2 \in (V \cup \Sigma)^*$ such that $u = u_1\alpha u_2$ and $v = u_1\beta u_2$. Thus, v is the result of applying the rule (α, β) to u .

Repetitive rule application

For any $u, v \in (V \cup \Sigma)^*$, $u \xrightarrow{*} v$ (or $u \Rightarrow \Rightarrow v$ in some textbooks), if $u_1, u_2, \dots, u_k \in (V \cup \Sigma)^*$, $k \geq 0$ such that $u \Rightarrow u_1 \Rightarrow u_2 \dots \Rightarrow u_k \Rightarrow v$

Context-free language

The language of a grammar $G = (V, \Sigma, R, S)$ is the set

$$L(G) = \{w \in \Sigma^* : S \xrightarrow{*} w\}$$

A language L is said to be a context-free language (CFL), if there exists a CFG G , such that $L = L(G)$.

Proper CFGs

A context-free grammar is said to be *proper*, if it has

- no *inaccessible* symbols: $\forall N \in V : \exists \alpha, \beta \in (V \cup \Sigma)^* : S \xrightarrow{*} \alpha N \beta$
- no *unproductive* symbols: $\forall N \in V : \exists w \in \Sigma^* : N \xrightarrow{*} w$
- no ϵ -productions: $\forall N \in V, w \in \Sigma^* : (N, w) \in R \Rightarrow w \neq \epsilon$
(the right-arrow in this case denotes logical "implies" and not grammatical "yields")
- no cycles: $\neg \exists N \in V : N \xrightarrow{*} N$

Example

The grammar $G = (\{S\}, \{a, b\}, P, S)$, with productions

$$S \rightarrow aSa,$$

$$S \rightarrow bSb,$$

$$S \rightarrow \epsilon,$$

is context-free. It is not proper since it includes an ϵ -production. A typical derivation in this grammar is

$$S \rightarrow aSa \rightarrow aaSaa \rightarrow aabSbaa \rightarrow aabbaa.$$

This makes it clear that $L(G) = \{ww^R : w \in \{a, b\}^*\}$. The language is context-free, however it can be proved that it is not regular.

Examples

Well-formed parentheses

The canonical example of a context free grammar is parenthesis matching, which is representative of the general case. There are two terminal symbols "(" and ")" and one nonterminal symbol S. The production rules are

$$S \rightarrow SS$$

$$S \rightarrow (S)$$

$$S \rightarrow ()$$

The first rule allows Ss to multiply; the second rule allows Ss to become enclosed by matching parentheses; and the third rule terminates the recursion.

Well-formed nested parentheses and square brackets

A second canonical example is two different kinds of matching nested parentheses, described by the productions:

S → SS
S → ()
S → (S)
S → []
S → [S]

with terminal symbols [] () and nonterminal S.

The following sequence can be derived in that grammar:

([[[()() [][]]]([])])

However, there is no context-free grammar for generating all sequences of two different types of parentheses, each separately balanced disregarding the other, but where the two types need not nest inside one another, for example:

[[[[((())]]])]))(([))(([))([)(])([]))

A regular grammar

Every regular grammar is context-free, but not all context-free grammars are regular. The following context-free grammar, however, is also regular.

$$\begin{array}{l} S \rightarrow a \\ S \rightarrow aS \\ S \rightarrow bS \end{array}$$

The terminals here are a and b , while the only non-terminal is S . The language described is all nonempty strings of a s and b s that end in a .

This grammar is regular: no rule has more than one nonterminal in its right-hand side, and each of these nonterminals is at the same end of the right-hand side.

Every regular grammar corresponds directly to a nondeterministic finite automaton, so we know that this is a regular language.

It is common to list all right-hand sides for the same left-hand side on the same line, using | (the pipe symbol) to separate them. Hence the grammar above can be described more tersely as follows:

$$S \rightarrow a \mid aS \mid bS$$

Matching pairs

In a context-free grammar, we can pair up characters the way we do with brackets. The simplest example:

$$S \rightarrow aSb$$

This grammar generates the language $\{a^n b^n : n \geq 1\}$, which is not regular (according to the Pumping Lemma for regular languages).

The special character ϵ stands for the empty string. By changing the above grammar to

$$S \rightarrow aSb \mid \varepsilon$$

we obtain a grammar generating the language $\{a^n b^n : n \geq 0\}$ instead. This differs only in that it contains the empty string while the original grammar did not.

Algebraic expressions

Here is a context-free grammar for syntactically correct infix algebraic expressions in the variables x, y and z:

1. $S \rightarrow x$
2. $S \rightarrow y$
3. $S \rightarrow z$
4. $S \rightarrow S + S$
5. $S \rightarrow S - S$
6. $S \rightarrow S * S$
7. $S \rightarrow S / S$
8. $S \rightarrow (S)$

This grammar can, for example, generate the string

$(x + y) * x - z * y / (x + x)$

as follows:

```

S (the start symbol)
→ S - S (by rule 5)
→ S * S - S (by rule 6, applied to the leftmost S)
→ S * S - S / S (by rule 7, applied to the rightmost S)
→ ( S ) * S - S / S (by rule 8, applied to the leftmost S)
→ ( S ) * S - S / ( S ) (by rule 8, applied to the rightmost S)
→ ( S + S ) * S - S / ( S ) (etc.)
→ ( S + S ) * S - S * S / ( S )
→ ( S + S ) * S - S * S / ( S + S )
→ ( x + S ) * S - S * S / ( S + S )
→ ( x + y ) * S - S * S / ( S + S )
→ ( x + y ) * x - S * y / ( S + S )
→ ( x + y ) * x - S * y / ( x + S )
→ ( x + y ) * x - z * y / ( x + S )
→ ( x + y ) * x - z * y / ( x + x )

```

Note that many choices were made underway as to which rewrite was going to be performed next. These choices look quite arbitrary. As a matter of fact, they are, in the sense that the string finally generated is always the same. For example, the second and third rewrites

```

→ S * S - S (by rule 6, applied to the leftmost S)
→ S * S - S / S (by rule 7, applied to the rightmost S)

```

could be done in the opposite order:

```

→ S - S / S (by rule 7, applied to the rightmost S)
→ S * S - S / S (by rule 6, applied to the leftmost S)

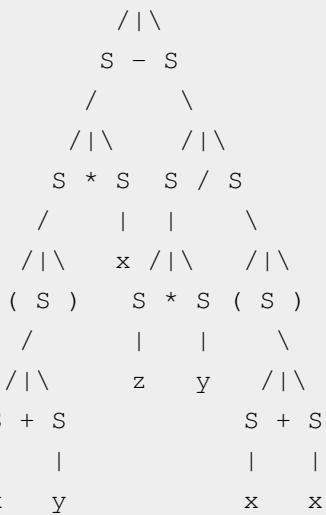
```

Also, many choices were made on which rule to apply to each selected S. Changing the choices made and not only the order they were made in usually affects which terminal string comes out at the end.

Let's look at this in more detail. Consider the parse tree of this derivation:

S

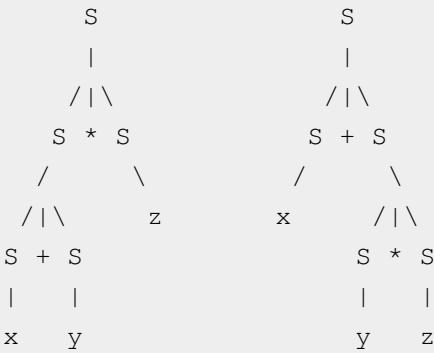
|



Starting at the top, step by step, an S in the tree is expanded, until no more unexpanded Ses (non-terminals) remain. Picking a different order of expansion will produce a different derivation, but the same parse tree. The parse tree will only change if we pick a different rule to apply at some position in the tree.

But can a different parse tree still produce the same terminal string, which is $(x + y)^* x - z^* y / (x + x)$ in this case? Yes, for this particular grammar, this is possible. Grammars with this property are called ambiguous.

For example, $x + y^* z$ can be produced with these two different parse trees:



However, the *language* described by this grammar is not inherently ambiguous: an alternative, unambiguous grammar can be given for the language, for example:

$$\begin{aligned}
 T &\rightarrow x \\
 T &\rightarrow y \\
 T &\rightarrow z \\
 S &\rightarrow S + T \\
 S &\rightarrow S - T \\
 S &\rightarrow S * T \\
 S &\rightarrow S / T \\
 T &\rightarrow (S) \\
 S &\rightarrow T
 \end{aligned}$$

(once again picking S as the start symbol). This alternative grammar will produce $x + y^* z$ with a parse tree similar to the left one above, i.e. implicitly assuming the association $(x + y)^* z$, which is not according to standard operator precedence. More elaborate, unambiguous and context-free grammars can be constructed that

produce parse trees that obey all desired operator precedence and associativity rules.

Further examples

Example 1

A context-free grammar for the language consisting of all strings over {a,b} containing an unequal number of a's and b's:

$$\begin{aligned} S &\rightarrow U \mid V \\ U &\rightarrow T_a U \mid T_a T \\ V &\rightarrow T_b V \mid T_b T \\ T &\rightarrow aT_b T \mid bT_a T \mid \epsilon \end{aligned}$$

Here, the nonterminal T can generate all strings with the same number of a's as b's, the nonterminal U generates all strings with more a's than b's and the nonterminal V generates all strings with fewer a's than b's.

Example 2

Another example of a non-regular language is $\{b^n a^m b^{2n} : n \geq 0, m \geq 0\}$. It is context-free as it can be generated by the following context-free grammar:

$$\begin{aligned} S &\rightarrow bSbb \mid A \\ A &\rightarrow aA \mid \epsilon \end{aligned}$$

Other examples

The formation rules for the terms and formulas of formal logic fit the definition of context-free grammar, except that the set of symbols may be infinite and there may be more than one start symbol.

Derivations and syntax trees

A *derivation* of a string for a grammar is a sequence of grammar rule applications that transforms the start symbol into the string. A derivation proves that the string belongs to the grammar's language.

A derivation is fully determined by giving, for each step:

- the rule applied in that step
- the occurrence of its left hand side to which it is applied

For clarity, the intermediate string is usually given as well.

For instance, with the grammar:

$$\begin{aligned} (1) \quad S &\rightarrow S + S \\ (2) \quad S &\rightarrow 1 \\ (3) \quad S &\rightarrow a \end{aligned}$$

the string

1 + 1 + a

can be derived with the derivation:

$$\begin{aligned} S \\ \rightarrow & (\text{rule 1 on first } S) \\ S+S \\ \rightarrow & (\text{rule 1 on second } S) \\ S+S+S \end{aligned}$$

```

→ (rule 2 on second S)
S+1+S
    → (rule 3 on third S)
S+1+a
    → (rule 2 on first S)
1+1+a

```

Often, a strategy is followed that deterministically determines the next nonterminal to rewrite:

- in a *leftmost derivation*, it is always the leftmost nonterminal;
- in a *rightmost derivation*, it is always the rightmost nonterminal.

Given such a strategy, a derivation is completely determined by the sequence of rules applied. For instance, the leftmost derivation

```

S
→ (rule 1 on first S)
S+S
    → (rule 2 on first S)
1+S
    → (rule 1 on first S)
1+S+S
    → (rule 2 on first S)
1+1+S
    → (rule 3 on first S)
1+1+a

```

can be summarized as

```
rule 1, rule 2, rule 1, rule 2, rule 3
```

The distinction between leftmost derivation and rightmost derivation is important because in most parsers the transformation of the input is defined by giving a piece of code for every grammar rule that is executed whenever the rule is applied. Therefore it is important to know whether the parser determines a leftmost or a rightmost derivation because this determines the order in which the pieces of code will be executed. See for an example LL parsers and LR parsers.

A derivation also imposes in some sense a hierarchical structure on the string that is derived. For example, if the string "1 + 1 + a" is derived according to the leftmost derivation:

```

S → S + S (1)
→ 1 + S (2)
→ 1 + S + S (1)
→ 1 + 1 + S (2)
→ 1 + 1 + a (3)

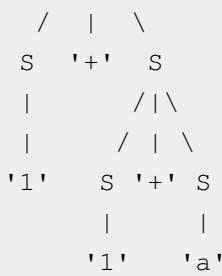
```

the structure of the string would be:

$$\{ \{ 1 \}_S + \{ \{ 1 \}_S + \{ a \}_S \}_S \}_S$$

where $\{ \dots \}_S$ indicates a substring recognized as belonging to S. This hierarchy can also be seen as a tree:

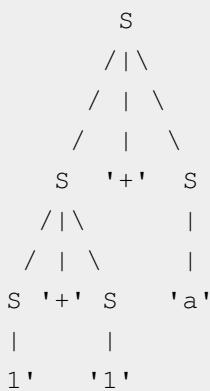




This tree is called a *concrete syntax tree* (see also abstract syntax tree) of the string. In this case the presented leftmost and the rightmost derivations define the same syntax tree; however, there is another (rightmost) derivation of the same string

$$\begin{aligned} S &\rightarrow S + S (1) \\ &\rightarrow S + a (3) \\ &\rightarrow S + S + a (1) \\ &\rightarrow S + 1 + a (2) \\ &\rightarrow 1 + 1 + a (2) \end{aligned}$$

and this defines the following syntax tree:



If, for certain strings in the language of the grammar, there is more than one parsing tree, then the grammar is said to be an *ambiguous grammar*. Such grammars are usually hard to parse because the parser cannot always decide which grammar rule it has to apply. Usually, ambiguity is a feature of the grammar, not the language, and an unambiguous grammar can be found that generates the same context-free language. However, there are certain languages that can only be generated by ambiguous grammars; such languages are called inherently ambiguous.

Normal forms

Every context-free grammar that does not generate the empty string can be transformed into one in which no rule has the empty string as a product [a rule with ϵ as a product is called an ϵ -production]. If it does generate the empty string, it will be necessary to include the rule $S \rightarrow \epsilon$, but there need be no other ϵ -rule. Every context-free grammar with no ϵ -production has an equivalent grammar in Chomsky normal form or Greibach normal form. "Equivalent" here means that the two grammars generate the same language.

Because of the especially simple form of production rules in Chomsky Normal Form grammars, this normal form has both theoretical and practical implications. For instance, given a context-free grammar, one can use the Chomsky Normal Form to construct a polynomial-time algorithm that decides whether a given string is in the language represented by that grammar or not (the CYK algorithm).

Undecidable problems

Some questions that are undecidable for wider classes of grammars become decidable for context-free grammars; e.g. the emptiness problem (whether the grammar generates any terminal strings at all), is undecidable for context-sensitive grammars, but decidable for context-free grammars.

Still, many problems remain undecidable. Examples:

Universality

Given a CFG, does it generate the language of all strings over the alphabet of terminal symbols used in its rules?^{[5][6]}

A reduction can be demonstrated to this problem from the well-known undecidable problem of determining whether a Turing machine accepts a particular input (the Halting problem). The reduction uses the concept of a *computation history*, a string describing an entire computation of a Turing machine. We can construct a CFG that generates all strings that are not accepting computation histories for a particular Turing machine on a particular input, and thus it will accept all strings only if the machine doesn't accept that input.

Language equality

Given two CFGs, do they generate the same language?^{[6][7]}

The undecidability of this problem is a direct consequence of the previous: we cannot even decide whether a CFG is equivalent to the trivial CFG defining the language of all strings.

Language inclusion

Given two CFGs, can the first generate all strings that the second can generate?^{[6][7]}

If this problem were decidable, then we could use it to determine whether two CFGs G1 and G2 generate the same language by checking whether L(G1) is a subset of L(G2) and L(G2) is a subset of L(G1).

Being in a lower or higher level of the Chomsky hierarchy

Using Greibach's theorem, it can be shown that the two following problems are undecidable:

- Given a context-sensitive grammar, does it describe a context-free language?
- Given a context-free grammar, does it describe a regular language?^{[6][7]}

Extensions

An obvious way to extend the context-free grammar formalism is to allow nonterminals to have arguments, the values of which are passed along within the rules. This allows natural language features such as agreement and reference, and programming language analogs such as the correct use and definition of identifiers, to be expressed in a natural way. E.g. we can now easily express that in English sentences, the subject and verb must agree in number. In computer science, examples of this approach include affix grammars, attribute grammars, indexed grammars, and Van Wijngaarden two-level grammars. Similar extensions exist in linguistics.

An **extended context-free grammar** is one in which the right-hand side of the production rules is allowed to be a regular expression over the grammar's terminals and nonterminals. Extended context-free grammars describe exactly the context-free languages.^[8]

Another extension is to allow additional terminal symbols to appear at the left hand side of rules, constraining their application. This produces the formalism of context-sensitive grammars.

Subclasses

There are a number of important subclasses of the context-free grammars:

- LR(k) grammars (also known as deterministic context-free grammars) allow parsing (string recognition) with deterministic pushdown automata, but they can only describe deterministic context-free languages.
- Simple LR, Look-Ahead LR grammars are subclasses that allow further simplification of parsing.
- LL(k) and LL(*) grammars allow parsing by direct construction of a leftmost derivation as described above, and describe even fewer languages.
- Simple grammars are a subclass of the LL(1) grammars mostly interesting for its theoretical property that language equality of simple grammars is decidable, while language inclusion is not.
- Bracketed grammars have the property that the terminal symbols are divided into left and right bracket pairs that always match up in rules.
- Linear grammars have no rules with more than one nonterminal in the right hand side.
- Regular grammars are a subclass of the linear grammars and describe the regular languages, i.e. they correspond to finite automata and regular expressions.

LR parsing extends LL parsing to support a larger range of grammars; in turn, generalized LR parsing extends LR parsing to support arbitrary context-free grammars. On LL grammars and LR grammars, it essentially performs LL parsing and LR parsing, respectively, while on nondeterministic grammars, it is as efficient as can be expected. Although GLR parsing was developed in the 1980s, many new language definitions and parser generators continue to be based on LL, LALR or LR parsing up to the present day.

Linguistic applications

Chomsky initially hoped to overcome the limitations of context-free grammars by adding transformation rules.^[2]

Such rules are another standard device in traditional linguistics; e.g. passivization in English. Much of generative grammar has been devoted to finding ways of refining the descriptive mechanisms of phrase-structure grammar and transformation rules such that exactly the kinds of things can be expressed that natural language actually allows. Allowing arbitrary transformations doesn't meet that goal: they are much too powerful, being Turing complete unless significant restrictions are added (e.g. no transformations that introduce and then rewrite symbols in a context-free fashion).

Chomsky's general position regarding the non-context-freeness of natural language has held up since then,^[9] although his specific examples regarding the inadequacy of context-free grammars (CFGs) in terms of their weak generative capacity were later disproved.^[10] Gerald Gazdar and Geoffrey Pullum have argued that despite a few context-sensitive constructions in natural language (such as cross-serial dependencies in Swiss German^[9] and reduplication in Bambara^[11]), the vast majority of forms in natural language are indeed context-free.^[10]

Notes

- [1] Hopcroft & Ullman (1979), p. 106.
- [2] Chomsky, Noam (Sep 1956), "Three models for the description of language" (<http://ieeexplore.ieee.org/iel5/18/22738/01056813.pdf?isnumber=22738&prod=STD&arnumber=1056813&arnumber=1056813&arSt=+113&ared=+124&arAuthor=+Chomsky,+N.>), *Information Theory, IEEE Transactions* **2** (3): 113–124, doi:10.1109/TIT.1956.1056813, , retrieved 2007-06-18
- [3] The notation here is that of Sipser (1997), p. 94. Hopcroft & Ullman (1979) (p. 79) define context-free grammars as 4-tuples in the same way, but with different variable names.
- [4] Hopcroft & Ullman (1979), pp. 90–92.
- [5] Sipser (1997), Theorem 5.10, p. 181.
- [6] Hopcroft & Ullman (1979), p. 281.
- [7] Hazewinkel, Michiel (1994), *Encyclopaedia of mathematics: an updated and annotated translation of the Soviet "Mathematical Encyclopaedia"* (<http://books.google.com/books?id=s9F71NJxwzoC&pg=PA56>), Springer, Vol. IV, p. 56, ISBN 978-1-55608-003-6. .
- [8] Norvell, Theodore. "A Short Introduction to Regular Expressions and Context-Free Grammars" (<http://www.engr.mun.ca/~theo/Courses/fm/pub/context-free.pdf>). pp. 4. . Retrieved August 24, 2012.
- [9] Shieber, Stuart (1985), "Evidence against the context-freeness of natural language" (<http://www.eecs.harvard.edu/~shieber/Biblio/Papers/shieber85.pdf>), *Linguistics and Philosophy* **8** (3): 333–343, doi:10.1007/BF00630917, ..
- [10] Pullum, Geoffrey K.; Gerald Gazdar (1982), "Natural languages and context-free languages", *Linguistics and Philosophy* **4** (4): 471–504, doi:10.1007/BF00360802.
- [11] Culy, Christopher (1985), "The Complexity of the Vocabulary of Bambara", *Linguistics and Philosophy* **8** (3): 345–351, doi:10.1007/BF00630918.

References

- Hopcroft, John E.; Ullman, Jeffrey D. (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley. Chapter 4: Context-Free Grammars, pp. 77–106; Chapter 6: Properties of Context-Free Languages, pp. 125–137.
- Sipser, Michael (1997), *Introduction to the Theory of Computation*, PWS Publishing, ISBN 0-534-94728-X. Chapter 2: Context-Free Grammars, pp. 91–122; Section 4.1.2: Decidable problems concerning context-free languages, pp. 156–159; Section 5.1.1: Reductions via computation histories: pp. 176–183.

Context-free language

In formal language theory, a **context-free language** is a language generated by some context-free grammar. The set of all context-free languages is identical to the set of languages accepted by pushdown automata.

Examples

An archetypical context-free language is $L = \{a^n b^n : n \geq 1\}$, the language of all non-empty even-length strings, the entire first halves of which are a 's, and the entire second halves of which are b 's. L is generated by the grammar $S \rightarrow aSb \mid ab$, and is accepted by the pushdown automaton $M = (\{q_0, q_1, q_f\}, \{a, b\}, \{a, z\}, \delta, q_0, z, \{q_f\})$ where δ is defined as follows:

$$\begin{aligned}\delta(q_0, a, z) &= (q_0, az) \\ \delta(q_0, a, a) &= (q_0, aa) \\ \delta(q_0, b, a) &= (q_1, \varepsilon) \\ \delta(q_1, b, a) &= (q_1, \varepsilon) \\ \delta(\text{state}_1, \text{read}, \text{pop}) &= (\text{state}_2, \text{push})\end{aligned}$$

Context-free languages have many applications in programming languages; for example, the language of all properly matched parentheses is generated by the grammar $S \rightarrow SS \mid (S) \mid \varepsilon$. Also, most arithmetic expressions are generated by context-free grammars.

Closure properties

Context-free languages are closed under the following operations. That is, if L and P are context-free languages, the following languages are context-free as well:

- the union $L \cup P$ of L and P
- the reversal of L
- the concatenation $L \cdot P$ of L and P
- the Kleene star L^* of L
- the image $\varphi(L)$ of L under a homomorphism φ
- the image $\varphi^{-1}(L)$ of L under an inverse homomorphism φ^{-1}
- the cyclic shift of L (the language $\{vu : uv \in L\}$)

Context-free languages are not closed under complement, intersection, or difference. However, if L is a context-free language and D is a regular language then both their intersection $L \cap D$ and their difference $L \setminus D$ are context-free languages.

Nonclosure under intersection and complement

The context-free languages are not closed under intersection. This can be seen by taking the languages $A = \{a^n b^n c^m \mid m, n \geq 0\}$ and $B = \{a^m b^n c^n \mid m, n \geq 0\}$, which are both context-free. Their intersection is $A \cap B = \{a^n b^n c^n \mid n \geq 0\}$, which can be shown to be non-context-free by the pumping lemma for context-free languages.

Context-free languages are also not closed under complementation, as for any languages A and B : $A \cap B = \overline{\overline{A} \cup \overline{B}}$.

Decidability properties

The following problems are undecidable for arbitrary context-free grammars A and B:

- Equivalence: is $L(A) = L(B)$?
- is $L(A) \cap L(B) = \emptyset$? (However, the intersection of a context-free language and a *regular* language is context-free, so if B were a regular language, this problem becomes decidable.)
- is $L(A) = \Sigma^*$?
- is $L(A) \subseteq L(B)$?

The following problems are decidable for arbitrary context-free languages:

- is $L(A) = \emptyset$?
- is $L(A)$ finite?
- Membership: given any word w , does $w \in L(A)$? (membership problem is even polynomially decidable - see CYK algorithm and Earley's Algorithm)

Properties of context-free languages

- The reverse of a context-free language is context-free, but the complement need not be.
- Every regular language is context-free because it can be described by a context-free grammar.
- The intersection of a context-free language and a regular language is always context-free.
- There exist context-sensitive languages which are not context-free.
- To prove that a given language is not context-free, one may employ the pumping lemma for context-free languages or a number of other methods, such as Ogden's lemma, Parikh's theorem, or using closure properties.^[1]
- Context Free Languages are closed under Union, Concatenation, and Kleene star.^[2]

Parsing

Determining an instance of the membership problem; i.e. given a string w , determine whether $w \in L(G)$ where L is the language generated by some grammar G ; is also known as *parsing*.

Formally, the set of all context-free languages is identical to the set of languages accepted by pushdown automata (PDA). Parser algorithms for context-free languages include the CYK algorithm and the Earley's Algorithm.

A special subclass of context-free languages are the deterministic context-free languages which are defined as the set of languages accepted by a deterministic pushdown automaton and can be parsed by a LR(k) parser.^[3]

See also parsing expression grammar as an alternative approach to grammar and parser.

References

- [1] How to prove that a language is not context-free? (<http://cs.stackexchange.com/questions/265/how-to-prove-that-a-language-is-not-context-free>)
- [2] (<http://infolab.stanford.edu/~ullman/ialc/spr10/slides/cfl5.pdf>)
- [3] Knuth, Donald (July 1965). "On the Translation of Languages from Left to Right" (<http://www.cs.dartmouth.edu/~mckeeeman/cs48/mxcom/doc/knuth65.pdf>). *Information and Control* 8: 707 - 639. . Retrieved 29 May 2011.
- Seymour Ginsburg (1966). *The Mathematical Theory of Context-Free Languages*. New York, NY, USA: McGraw-Hill, Inc..
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X. Chapter 2: Context-Free Languages, pp. 91–122.
- Jean-Michel Autebert, Jean Berstel, Luc Boasson, Context-Free Languages and Push-Down Automata (<http://www-igm.univ-mlv.fr/~berstel/Articles/1997CFLPDA.pdf>), in: G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1, Springer-Verlag, 1997, 111-174.

Pushdown automaton

In computer science, a **pushdown automaton** (PDA) is a type of automaton that employs a stack.

The PDA is used in theories about what can be computed by machines. It is more capable than a finite-state machine but less capable than a Turing machine. Because its input can be described with a formal grammar, it can be used in parser design. The deterministic pushdown automaton can handle all deterministic context-free languages while the nondeterministic version can handle all context-free languages.

The term "pushdown" refers to the fact that the stack can be regarded as being "pushed down" like a tray dispenser at a cafeteria, since the operations never work on elements other than the top element. A **stack automaton**, by contrast, does allow access to and operations on deeper elements. Stack automata can recognize a strictly larger set of languages than deterministic pushdown automata. A nested stack automaton allows full access, and also allows stacked values to be entire sub-stacks rather than just single finite symbols.

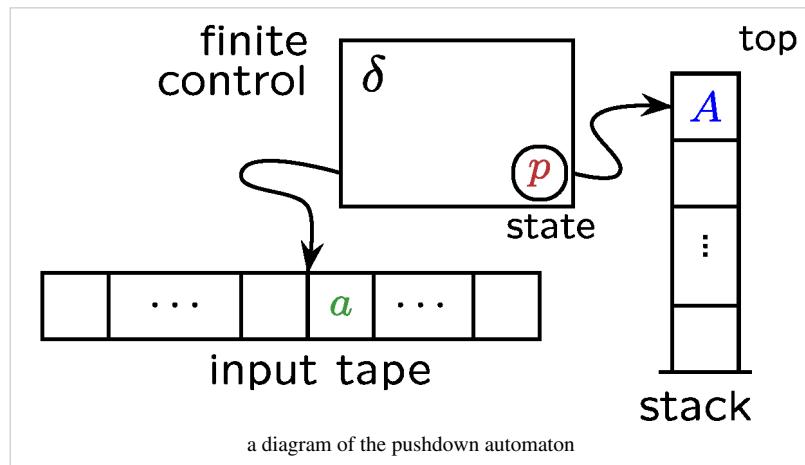
The remainder of this article describes the nondeterministic pushdown automaton.

Operation

Pushdown automata differ from finite state machines in two ways:

1. They can use the top of the stack to decide which transition to take.
2. They can manipulate the stack as part of performing a transition.

Pushdown automata choose a transition by indexing a table by input signal, current state, and the symbol at the top of the stack. This means that those three parameters completely determine the transition path that is chosen. Finite state machines just look at the input signal and the current state: they have no stack to work with. Pushdown automata add the stack as a parameter for choice.



Pushdown automata can also manipulate the stack, as part of performing a transition. Finite state machines choose a new state, the result of following the transition. The manipulation can be to push a particular symbol to the top of the stack, or to pop off the top of the stack. The automaton can alternatively ignore the stack, and leave it as it is. The choice of manipulation (or no manipulation) is determined by the transition table.

Put together: Given an input signal, current state, and stack symbol, the automaton can follow a transition to another state, and optionally manipulate (push or pop) the stack.

In general, pushdown automata may have several computations on a given input string, some of which may be halting in accepting configurations. If only one computation exists for all accepted strings, the result is a deterministic pushdown automaton (DPDA) and the language of these strings is a deterministic context-free language. Not all context-free languages are deterministic. The problem of deciding whether a context-free language is deterministic is unsolvable.^[1] As a consequence of the above the DPDA is a strictly weaker variant of the PDA and there exists no algorithm for converting a PDA to an equivalent DPDA, if such a DPDA exists.

If we allow a finite automaton access to two stacks instead of just one, we obtain a more powerful device, equivalent in power to a Turing machine. A linear bounded automaton is a device which is more powerful than a pushdown automaton but less so than a Turing machine.

Relation to backtracking

Nondeterministic PDAs are able to handle situations where more than one choices of action are available. In principle it is enough to create in every such case new automaton instances that will handle the extra choices. The problem with this approach is that in practice most of these instances quickly fail. This can severely affect the automaton's performance as the execution of multiple instances is a costly operation. Situations such as these can be identified in the design phase of the automaton by examining the grammar the automaton uses. This makes possible the use of backtracking in every such case in order to improve performance.

Formal Definition

We use standard formal language notation: Γ^* denotes the set of strings over alphabet Γ and ε denotes the empty string.

A PDA is formally defined as a 7-tuple:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F) \text{ where}$$

- Q is a finite set of *states*
- Σ is a finite set which is called the *input alphabet*
- Γ is a finite set which is called the *stack alphabet*
- δ is a finite subset of $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$, the *transition relation*.
- $q_0 \in Q$ is the *start state*
- $Z \in \Gamma$ is the *initial stack symbol*
- $F \subseteq Q$ is the set of *accepting states*

An element $(p, a, A, q, \alpha) \in \delta$ is a transition of M . It has the intended meaning that M , in state $p \in Q$, with $a \in \Sigma \cup \{\varepsilon\}$ on the input and with $A \in \Gamma$ as topmost stack symbol, may read a , change the state to q , pop A , replacing it by pushing $\alpha \in \Gamma^*$. The $(\Sigma \cup \{\varepsilon\})$ component of the transition relation is used to formalize that the PDA can either read a letter from the input, or proceed leaving the input untouched. In many texts the transition relation is replaced by an (equivalent) formalization, where

- δ is the *transition function*, mapping $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ into finite subsets of $Q \times \Gamma^*$.

Here $\delta(p, a, A)$ contains all possible actions in state p with A on the stack, while reading a on the input. One writes $(q, \alpha) \in \delta(p, a, A)$ for the function precisely when $(p, a, A, q, \alpha) \in \delta$ for the relation. Note that *finite* in this definition is essential.

Computations

In order to formalize the semantics of the pushdown automaton a description of the current situation is introduced. Any 3-tuple $(p, w, \beta) \in Q \times \Sigma^* \times \Gamma^*$ is called an instantaneous description (ID) of M , which includes the current state, the part of the input tape that has not been read, and the contents of the stack (topmost symbol written first). The transition relation δ defines the step-relation \vdash_M of M on instantaneous descriptions. For instruction $(p, a, A, q, \alpha) \in \delta$ there exists a step $(p, ax, A\gamma) \vdash_M (q, x, \alpha\gamma)$, for every $x \in \Sigma^*$ and every $\gamma \in \Gamma^*$.

In general pushdown automata are nondeterministic meaning that in a given instantaneous description (p, w, β) there may be several possible steps. Any of these steps can be chosen in a computation. With the above definition in each step always a single symbol (top of the stack) is popped, replacing it with as many symbols as necessary. As a consequence no step is defined when the stack is empty.

Computations of the pushdown automaton are sequences of steps. The computation starts in the initial state q_0 with the initial stack symbol Z on the stack, and a string w on the input tape, thus with initial description (q_0, w, Z) .

There are two modes of accepting. The pushdown automaton either accepts by final state, which means after reading its input the automaton reaches an accepting state (in F), or it accepts by empty stack (ε), which means after reading its input the automaton empties its stack. The first acceptance mode uses the internal memory (state), the second the external memory (stack).

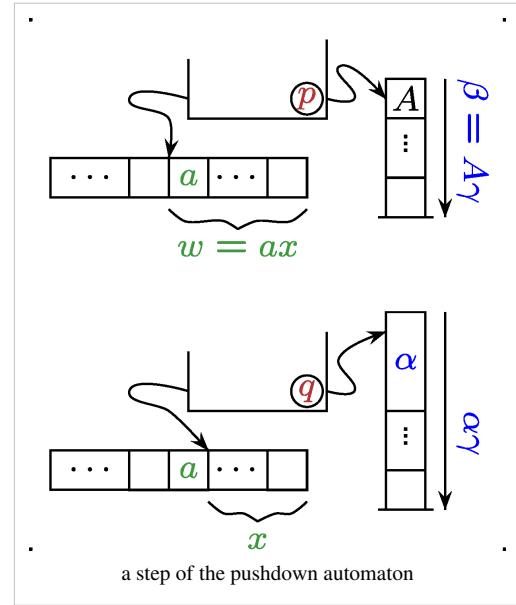
Formally one defines

1. $L(M) = \{w \in \Sigma^* | (q_0, w, Z) \vdash_M^* (f, \varepsilon, \gamma) \text{ with } f \in F \text{ and } \gamma \in \Gamma^*\}$ (final state)
2. $N(M) = \{w \in \Sigma^* | (q_0, w, Z) \vdash_M^* (q, \varepsilon, \varepsilon) \text{ with } q \in Q\}$ (empty stack)

Here \vdash_M^* represents the reflexive and transitive closure of the step relation \vdash_M meaning any number of consecutive steps (zero, one or more).

For each single pushdown automaton these two languages need to have no relation: they may be equal but usually this is not the case. A specification of the automaton should also include the intended mode of acceptance. Taken over all pushdown automata both acceptance conditions define the same family of languages.

Theorem. For each pushdown automaton M one may construct a pushdown automaton M' such that $L(M) = N(M')$, and vice versa, for each pushdown automaton M one may construct a pushdown automaton M' such that $N(M) = L(M')$



Example

The following is the formal description of the PDA which recognizes the language $\{0^n 1^n \mid n \geq 0\}$ by final state:

$M = (Q, \Sigma, \Gamma, \delta, p, Z, F)$, where

$Q = \{p, q, r\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{A, Z\}$

$q_0 = p$

$F = \{r\}$

δ consists of the following six instructions:

$(p, 0, Z, p, AZ), (p, 0, A, p, AA),$

$(p, \epsilon, Z, q, Z), (p, \epsilon, A, q, A), (q, 1, A, q, \epsilon), \text{ and } (q, \epsilon, Z, r, Z).$

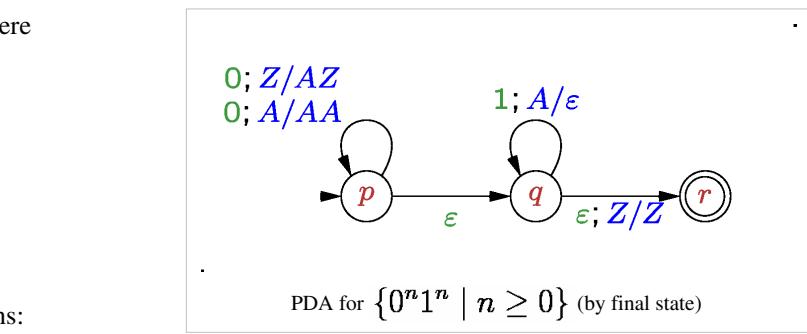
In words, in state p for each symbol 0 read, one A is pushed onto the stack. Pushing symbol A on top of another A is formalized as replacing top A by AA . In state q for each symbol 1 read one A is popped. At any moment the automaton may move from state p to state q , while it may move from state q to accepting state r only when the stack consists of a single Z .

There seems to be no generally used representation for PDA. Here we have depicted the instruction (p, a, A, q, α) by an edge from state p to state q labelled by $a; A/\alpha$ (read a ; replace A by α).

Understanding the computation process

The following illustrates how the above PDA computes on different input strings. The subscript M from the step symbol \vdash is here omitted.

(a) Input string = 0011. There are various computations, depending on the moment the move from state p to state q is made. Only one of these is accepting.



$(p, 0011, Z) \vdash$	$(p, 0011, AZ) \vdash$	$(p, 0011, Z) \vdash$
$(p, 011, Z) \vdash$	$(p, 011, AZ) \vdash$	$(p, 011, Z) \vdash$
$(p, 11, AAZ) \vdash$	$(p, 11, AAZ) \vdash$	$(p, 11, AAZ) \vdash$
$(q, 11, AAZ) \vdash$	$(q, 11, AAZ) \vdash$	$(q, 11, AAZ) \vdash$
$(q, 1, AZ) \vdash$	$(q, 1, AZ) \vdash$	$(q, 1, AZ) \vdash$
$(q, \epsilon, Z) \vdash$	$(q, \epsilon, Z) \vdash$	$(q, \epsilon, Z) \vdash$
$(r, \epsilon, Z) \vdash$	$(r, \epsilon, Z) \vdash$	$(r, \epsilon, Z) \vdash$

accepting computation for 0011

(i) $(p, 0011, Z) \vdash (q, 0011, Z) \vdash (r, 0011, Z)$. The final state is accepting, but the input is not accepted this way as it has not been read.

(ii) $(p, 0011, Z) \vdash (p, 011, AZ) \vdash (q, 011, AZ)$. No further steps possible.

(iii) $(p, 0011, Z) \vdash (p, 011, AZ) \vdash (p, 11, AAZ) \vdash (q, 11, AAZ) \vdash (q, 1, AZ) \vdash (q, \epsilon, Z) \vdash (r, \epsilon, Z)$. Accepting computation: ends in accepting state, while complete input has been read.

(b) Input string = 00111. Again there are various computations. None of these is accepting.

(i) $(p, 00111, Z) \vdash (q, 00111, Z) \vdash (r, 00111, Z)$. The final state is accepting, but the input is not accepted this way as it has not been read.

(ii) $(p, 00111, Z) \vdash (p, 0111, AZ) \vdash (q, 0111, AZ)$. No further steps possible.

(iii) $(p, 00111, Z) \vdash (p, 0111, AZ) \vdash (p, 111, AAZ) \vdash (q, 111, AAZ) \vdash (q, 11, AZ) \vdash (q, 1, Z) \vdash (r, 1, Z)$. The final state is accepting, but the input is not accepted this way as it has not been (completely) read.

PDA and Context-free Languages

Every context-free grammar can be transformed into an equivalent pushdown automaton. The derivation process of the grammar is simulated in a leftmost way. Where the grammar rewrites a nonterminal, the PDA takes the topmost nonterminal from its stack and replaces it by the right-hand part of a grammatical rule (expand). Where the grammar generates a terminal symbol, the PDA reads a symbol from input when it is the topmost symbol on the stack (match). In a sense the stack of the PDA contains the unprocessed data of the grammar, corresponding to a pre-order traversal of a derivation tree.

Technically, given a context-free grammar, the PDA is constructed as follows.

1. $(1, \epsilon, A, 1, \alpha)$ for each rule $A \rightarrow \alpha$ (*expand*)
2. $(1, a, a, 1, \epsilon)$ for each terminal symbol a (*match*)

As a result we obtain a single state pushdown automaton, the state here is 1 , accepting the context-free language by empty stack. Its initial stack symbol equals the axiom of the context-free grammar.

The converse, finding a grammar for a given PDA, is not that easy. The trick is to code two states of the PDA into the nonterminals of the grammar.

Theorem. For each pushdown automaton M one may construct a context-free grammar G such that $N(M) = L(G)$.

Generalized Pushdown Automaton (GPDA)

A GPDA is a PDA which writes an entire string of some known length to the stack or removes an entire string from the stack in one step.

A GPDA is formally defined as a 6-tuple:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

where Q, Σ, Γ, q_0 and F are defined the same way as a PDA.

$$\delta : Q \times \Sigma_\epsilon \times \Gamma^* \longrightarrow P(Q \times \Gamma^*)$$

is the transition function.

Computation rules for a GPDA are the same as a PDA except that the a_{i+1} 's and b_{i+1} 's are now strings instead of symbols.

GPDA's and PDA's are equivalent in that if a language is recognized by a PDA, it is also recognized by a GPDA and vice versa.

One can formulate an analytic proof for the equivalence of GPDA's and PDA's using the following simulation:

Let $\delta(q_1, w, x_1 x_2 \dots x_m) \longrightarrow (q_2, y_1 y_2 \dots y_n)$ be a transition of the GPDA

where $q_1, q_2 \in Q, w \in \Sigma_\epsilon, x_1, x_2, \dots, x_m \in \Gamma^*, m \geq 0, y_1, y_2, \dots, y_n \in \Gamma^*, n \geq 0$.

Construct the following transitions for the PDA:

$$\begin{aligned} \delta'(q_1, w, x_1) &\longrightarrow (p_1, \epsilon) \\ \delta'(p_1, \epsilon, x_2) &\longrightarrow (p_2, \epsilon) \\ &\vdots \\ \delta'(p_{m-1}, \epsilon, x_m) &\longrightarrow (p_m, \epsilon) \\ \delta'(p_m, \epsilon, \epsilon) &\longrightarrow (p_{m+1}, y_n) \\ \delta'(p_{m+1}, \epsilon, \epsilon) &\longrightarrow (p_{m+2}, y_{n-1}) \\ &\vdots \\ \delta'(p_{m+n-1}, \epsilon, \epsilon) &\longrightarrow (q_2, y_1) \end{aligned}$$

References

- [1] Greibach, Sheila (October 1966). "The Unsolvability of the Recognition of Linear Context-Free Languages". *Journal of the ACM* **13** (4).
- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X.
Section 2.2: Pushdown Automata, pp.101–114.
 - Jean-Michel Autebert, Jean Berstel, Luc Boasson, Context-Free Languages and Push-Down Automata (<http://www-igm.univ-mlv.fr/~berstel/Articles/1997CFLPDA.pdf>), in: G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, Vol. 1, Springer-Verlag, 1997, 111-174.

External links

- non-deterministic pushdown automaton (<http://planetmath.org/encyclopedia/PushdownAutomaton.html>), on Planet Math.
- JFLAP (<http://www.jflap.org>), simulator for several types of automata including nondeterministic pushdown automata

Regular grammar

In computer science, a **regular grammar** is a formal grammar that describes a regular language.

Strictly regular grammars

A **right regular grammar** (also called right linear grammar) is a formal grammar (N, Σ, P, S) such that all the production rules in P are of one of the following forms:

1. $B \rightarrow a$ - where B is a non-terminal in N and a is a terminal in Σ
2. $B \rightarrow aC$ - where B and C are in N and a is in Σ
3. $B \rightarrow \epsilon$ - where B is in N and ϵ denotes the empty string, i.e. the string of length 0.

In a **left regular grammar** (also called left linear grammar), all rules obey the forms

1. $A \rightarrow a$ - where A is a non-terminal in N and a is a terminal in Σ
2. $A \rightarrow Ba$ - where A and B are in N and a is in Σ
3. $A \rightarrow \epsilon$ - where A is in N and ϵ is the empty string.

An example of a right regular grammar G with $N = \{S, A\}$, $\Sigma = \{a, b, c\}$, P consists of the following rules

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow bA \\ A &\rightarrow \epsilon \\ A &\rightarrow cA \end{aligned}$$

and S is the start symbol. This grammar describes the same language as the regular expression a^*bc^* .

A **regular grammar** is a left or right regular grammar.

Some textbooks and articles disallow empty production rules, and assume that the empty string is not present in languages.

Extended regular grammars

An *extended right regular grammar* is one in which all rules obey one of

1. $B \rightarrow a$ - where B is a non-terminal in N and a is a terminal in Σ
2. $A \rightarrow wB$ - where A and B are in N and w is in Σ^*
3. $A \rightarrow \epsilon$ - where A is in N and ϵ is the empty string.

Some authors call this type of grammar a *right regular grammar* (or *right linear grammar*) and the type above a *strictly right regular grammar* (or *strictly right linear grammar*).

An *extended left regular grammar* is one in which all rules obey one of

1. $A \rightarrow a$ - where A is a non-terminal in N and a is a terminal in Σ
2. $A \rightarrow Bw$ - where A and B are in N and w is in Σ^*
3. $A \rightarrow \epsilon$ - where A is in N and ϵ is the empty string.

Some authors call this type of grammar a *left regular grammar* and the type above a *strictly left regular grammar*.

Expressive power

There is a direct one-to-one correspondence between the rules of a (strictly) left regular grammar and those of a nondeterministic finite automaton, such that the grammar generates exactly the language the automaton accepts. Hence, the left regular grammars generate exactly all regular languages. The right regular grammars describe the reverses of all such languages, that is, exactly the regular languages as well.

Every strict right regular grammar is extended right regular, while every extended right regular grammar can be made strict by inserting new nonterminals, such that the result generates the same language; hence, extended right regular grammars generate the regular languages as well. Analogously, so do the extended left regular grammars.

If empty productions are disallowed, only all regular languages that do not include the empty string can be generated.

Mixing left and right regular rules

If mixing of left-regular and right-regular rules is allowed, we still have a linear grammar, but not necessarily a regular one. What is more, such a grammar need not generate a regular language: all linear grammars can be easily brought into this form, and hence, such grammars can generate exactly all linear languages, including nonregular ones.

For instance, the grammar G with $N = \{S, A\}$, $\Sigma = \{a, b\}$, P with start symbol S and rules

$$S \rightarrow aA$$

$$A \rightarrow Sb$$

$$S \rightarrow \epsilon$$

generates $\{a^i b^i : i \geq 0\}$, the paradigmatic non-regular linear language.

Regular language

In theoretical computer science and formal language theory, a **regular language** is a formal language that can be expressed using a regular expression. Note that the "regular expression" features provided with many programming languages are augmented with features that make them capable of recognizing languages that can not be expressed by the formal regular expressions (*as formally defined below*).

In the Chomsky hierarchy, regular languages are defined to be the languages that are generated by Type-3 grammars (regular grammars). Regular languages are very useful in input parsing and programming language design.

Formal definition

The collection of regular languages over an alphabet Σ is defined recursively as follows:

- The empty language \emptyset is a regular language.
- For each $a \in \Sigma$ (a belongs to Σ), the singleton language $\{a\}$ is a regular language.
- If A and B are regular languages, then $A \cup B$ (union), $A \cdot B$ (concatenation), and A^* (Kleene star) are regular languages.
- No other languages over Σ are regular.

See regular expression for its syntax and semantics. Note that the above cases are in effect the defining rules of regular expression.

Examples

All finite languages are regular; in particular the empty string language $\{\epsilon\} = \emptyset^*$ is regular. Other typical examples include the language consisting of all strings over the alphabet $\{a, b\}$ which contain an even number of a s, or the language consisting of all strings of the form: several a s followed by several b s.

A simple example of a language that is not regular is the set of strings $\{a^n b^n \mid n \geq 0\}$.^[1] Intuitively, it cannot be recognized with a finite automaton, since a finite automaton has finite memory and it cannot remember the exact number of a 's. Techniques to prove this fact rigorously are given below.

Equivalence to other formalisms

A regular language satisfies the following equivalent properties:

- it can be accepted by a (nondeterministic) finite automaton, but also by the restricted deterministic finite automaton, or the more general alternating finite automaton
- it can be generated by a regular grammar
- it can be generated by a prefix grammar
- it can be accepted by a read-only Turing machine
- it can be defined in monadic second-order logic (Büchi-Elgot-Trakhtenbrot theorem^[2])
- it is recognized by some finite monoid, meaning it is the preimage of a subset of a finite monoid under a homomorphism from the free monoid on its alphabet (see Myhill–Nerode theorem).

The above properties are sometimes used as alternative definition of regular languages.

Closure properties

The regular languages are closed under the various operations, that is, if the languages K and L are regular, so is the result of the following operations:

- the set theoretic Boolean operations: union $K \cup L$, intersection $K \cap L$, and complement \bar{L} . From this also relative complement $K - L$ follows.
- the regular operations: union $K \cup L$, concatenation $K \circ L$, and Kleene star L^* .
- the trio operations: string homomorphism, inverse string homomorphism, and intersection with regular languages. As a consequence they are closed under arbitrary finite state transductions, like quotient K/L with a regular language. Even more, regular languages are closed under quotients with *arbitrary* languages: If L is regular then L/K is regular for any K .
- the reverse (or mirror image) L^R .

Deciding whether a language is regular

To locate the regular languages in the Chomsky hierarchy, one notices that every regular language is context-free. The converse is not true: for example the language consisting of all strings having the same number of a 's as b 's is context-free but not regular. To prove that a language such as this is regular, one often uses the Myhill–Nerode theorem or the pumping lemma among other methods.^[3]

There are two purely algebraic approaches to define regular languages. If:

- Σ is a finite alphabet,
- Σ^* denotes the free monoid over Σ consisting of all strings over Σ ,
- $f: \Sigma^* \rightarrow M$ is a monoid homomorphism where M is a *finite* monoid,
- S is a subset of M

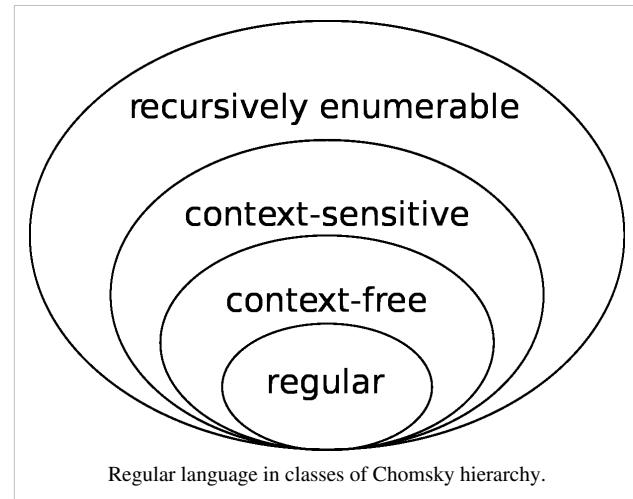
then the set $\{w \in \Sigma^* \mid f(w) \in S\}$ is regular. Every regular language arises in this fashion.

If L is any subset of Σ^* , one defines an equivalence relation \sim (called the syntactic relation) on Σ^* as follows: $u \sim v$ is defined to mean

$$uw \in L \text{ if and only if } vw \in L \text{ for all } w \in \Sigma^*$$

The language L is regular if and only if the number of equivalence classes of \sim is finite (A proof of this is provided in the article on the syntactic monoid). When a language is regular, then the number of equivalence classes is equal to the number of states of the minimal deterministic finite automaton accepting L .

A similar set of statements can be formulated for a monoid $M \subset \Sigma^*$. In this case, equivalence over M leads to the concept of a recognizable language.



Complexity results

In computational complexity theory, the complexity class of all regular languages is sometimes referred to as **REGULAR** or **REG** and equals $\text{DSPACE}(O(1))$, the decision problems that can be solved in constant space (the space used is independent of the input size). **REGULAR** $\neq \text{AC}^0$, since it (trivially) contains the parity problem of determining whether the number of 1 bits in the input is even or odd and this problem is not in AC^0 .^[4] On the other hand, **REGULAR** does not contain AC^0 , because the nonregular language of palindromes, or the nonregular language $\{0^n 1^n : n \in \mathbb{N}\}$ can both be recognized in AC^0 .^[5]

If a language is *not* regular, it requires a machine with at least $\Omega(\log \log n)$ space to recognize (where n is the input size).^[6] In other words, $\text{DSPACE}(\Theta(\log \log n))$ equals the class of regular languages. In practice, most nonregular problems are solved by machines taking at least logarithmic space.

Subclasses

Important subclasses of regular languages include

- Finite languages - those containing only a finite number of words. These are regular languages, as one can create a regular expression that is the union of every word in the language.
- Star-free languages, those that can be described by a regular expression constructed from the empty symbol, letters, concatenation and all boolean operators including complementation but not the Kleene star: this class includes all finite languages.^[7]
- **Cyclic languages**, satisfying the conditions $uv \in L \Leftrightarrow vu \in L$ and $w \in L \Leftrightarrow w^n \in L$.^[8]

The number of words in a regular language

Let $s_L(n)$ denote the number of words of length n in L . The ordinary generating function for L is the formal power series

$$S_L(z) = \sum_{n \geq 0} s_L(n) z^n .$$

The generating function of a language L is a rational function if and only if L is regular.^[8] Hence for any infinite regular language L there exist constants $\lambda_1, \dots, \lambda_k$ and polynomials $p_1(x), \dots, p_k(x)$ such that for every n the number $s_L(n)$ of words of length n in L satisfies the equation $s_L(n) = p_1(n)\lambda_1^n + \dots + p_k(n)\lambda_k^n$.^{[9][10]}

Thus, non-regularity of certain infinite languages L' can be proved by counting the words of a given length in L' . Consider, for example, the Dyck language of strings of balanced parentheses. The number of words of length $2n$ in the Dyck language is equal to the Catalan number $C_n \sim \frac{4^n}{n^{3/2} \sqrt{\pi}}$, which is not of the form $p(n)\lambda^n$, witnessing the non-regularity of the Dyck language.

The *zeta function* of a language L is^[8]

$$\zeta_L(z) = \exp \left(\sum_{n \geq 0} s_L(n) \frac{z^n}{n} \right) .$$

The zeta function of a regular language is not in general rational, but that of a cyclic language is.^[11]

Generalizations

The notion of a regular language has been generalized to infinite words (see ω -automata) and to trees (see tree automaton).

References

- Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS Publishing. ISBN 0-534-94728-X.
Chapter 1: Regular Languages, pp. 31–90. Subsection "Decidable Problems Concerning Regular Languages" of section 4.1: Decidable Languages, pp. 152–155.
 - Eilenberg, Samuel (1974). *Automata, Languages, and Machines*. A. New York: Academic Press.
- [1] Eilenberg (1974), p. 16 (Example II, 2.8) and p. 25 (Example II, 5.2).
- [2] M. Weyer: Chapter 12 - Decidability of S1S and S2S, p. 219, Theorem 12.26. In: Erich Grädel, Wolfgang Thomas, Thomas Wilke (Eds.): Automata, Logics, and Infinite Games: A Guide to Current Research. Lecture Notes in Computer Science 2500, Springer 2002.
- [3] How to prove that a language is not regular? (<http://cs.stackexchange.com/questions/1031/how-to-prove-that-a-language-is-not-regular>)
- [4] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Systems Theory*, 17:13–27, 1984.
- [5] Cook, Stephen; Nguyen, Phuong (2010). *Logical foundations of proof complexity* (1. publ. ed.). Ithaca, NY: Association for Symbolic Logic. pp. 75. ISBN 0-521-51729-X.
- [6] J. Hartmanis, P. L. Lewis II, and R. E. Stearns. Hierarchies of memory-limited computations. *Proceedings of the 6th Annual IEEE Symposium on Switching Circuit Theory and Logic Design*, pp. 179–190. 1965.
- [7] Volker Diekert, Paul Gastin (2008). "First-order definable languages" (<http://www.lsv.ens-cachan.fr/Publications/PAPERS/PDF/DG-WT08.pdf>). In Jörg Flum, Erich Grädel, Thomas Wilke. *Logic and automata: history and perspectives*. Amsterdam University Press. ISBN 978-90-5356-576-6. .
- [8] Honkala, Juha (1989). "A necessary condition for the rationality of the zeta function of a regular language". *Theor. Comput. Sci.* **66** (3): 341–347. Zbl 0675.68034.
- [9] Proof of theorem for irreducible DFAs (<http://cs.stackexchange.com/a/1048/55>)
- [10] Number of words of a given length in a regular language (<http://cs.stackexchange.com/q/1045/55>)
- [11] Berstel, Christophe; Reutenauer (1990). "Zeta functions of formal languages". *Trans. Am. Math. Soc.* **321** (2): 533–546. Zbl 0797.68092.

External links

- Complexity Zoo: Class REG (http://complexity-zoo.net/zoo/wiki/Complexity_Zoo:R#reg)

Regular expression

In computing, a **regular expression** provides a concise and flexible means to "match" (specify and recognize) strings of text, such as particular characters, words, or patterns of characters. Common abbreviations for "regular expression" include **regex** and **regexp**.

The concept of regular expressions was first popularized by utilities provided by Unix distributions, in particular the editor ed and the filter grep. A regular expression is written in a formal language that can be interpreted by a regular expression processor, which is a program that either serves as a parser generator or examines text and identifies parts that match the provided specification. Historically, the concept of regular expressions is associated with Kleene's formalism of regular sets, introduced in the 1950s.

The following are examples of specifications which can be expressed as a regular expression:

- The sequence of characters "car" appearing consecutively, such as in "car", "cartoon", or "bicarbonate"
- The word "car" when it appears as an isolated word (and delimited from other words, typically through whitespace characters)
- The word "car" when preceded by the word "motor" (and separated by a named delimiter, or multiple.)

Regular expressions are used by many text editors, utilities, and programming languages to search and manipulate text based on patterns. Some of these languages, including Perl, Ruby, AWK, and Tcl, integrate regular expressions into the syntax of the core language itself. Other programming languages like .NET languages, Java, and Python instead provide regular expressions through standard libraries. For yet other languages, such as Object Pascal (Delphi) and C and C++, non-core libraries are available (however, version C++11 provides regular expressions in its Standard Libraries).

As an example of the syntax, the regular expression `\bex` can be used to search for all instances of the string "ex" occurring after "word boundaries". Thus `\bex` will find the matching string "ex" in two possible locations, (1) at the beginning of words, and (2) between two characters in a string, where the first is not a word character and the second is a word character. For instance, in the string "Texts for experts", `\bex` matches the "ex" in "experts" but not in "Texts" (because the "ex" occurs inside a word and not immediately after a word boundary).

Many modern computing systems provide wildcard characters in matching filenames from a file system. This is a core capability of many command-line shells and is also known as globbing. Wildcards differ from regular expressions in generally expressing only limited forms of patterns.

History

The origins of regular expressions lie in automata theory and formal language theory, both of which are part of theoretical computer science. These fields study models of computation (automata) and ways to describe and classify formal languages. In the 1950s, mathematician Stephen Cole Kleene described these models using his mathematical notation called *regular sets*.^[1] The SNOBOL language was an early implementation of pattern matching, but not identical to regular expressions. Ken Thompson built Kleene's notation into the editor QED as a means to match patterns in text files. He later added this capability to the Unix editor ed, which eventually led to the popular search tool grep's use of regular expressions ("grep" is a word derived from the command for regular expression searching in the ed editor: `g/re/p` where *re* stands for regular expression^[2]). Since that time, many variations of Thompson's original adaptation of regular expressions have been widely used in Unix and Unix-like utilities including expr, AWK, Emacs, vi, and lex.

Perl and Tcl regular expressions were derived from a regex library written by Henry Spencer, though Perl later expanded on Spencer's library to add many new features.^[3] Philip Hazel developed PCRE (Perl Compatible Regular Expressions), which attempts to closely mimic Perl's regular expression functionality and is used by many modern tools including PHP and Apache HTTP Server. Part of the effort in the design of Perl 6 is to improve Perl's regular

expression integration, and to increase their scope and capabilities to allow the definition of parsing expression grammars.^[4] The result is a mini-language called Perl 6 rules, which are used to define Perl 6 grammar as well as provide a tool to programmers in the language. These rules maintain existing features of Perl 5.x regular expressions, but also allow BNF-style definition of a recursive descent parser via sub-rules.

The use of regular expressions in structured information standards for document and database modeling started in the 1960s and expanded in the 1980s when industry standards like ISO SGML (precursored by ANSI "GCA 101-1983") consolidated. The kernel of the structure specification language standards consists of regular expressions. Its use is evident in the DTD element group syntax.

Basic concepts

A regular expression, often called a pattern, is an expression that specifies a set of strings. To specify such sets of strings, rules are often more concise than lists of a set's members. For example, the set containing the three strings "Handel", "Händel", and "Haendel" can be specified by the pattern H (ä | ae?) ndel (or alternatively, it is said that the pattern *matches* each of the three strings). In most formalisms, if there exists at least one regex that matches a particular set then there exist an infinite number of such expressions. Most formalisms provide the following operations to construct regular expressions.

Boolean "or"

A vertical bar separates alternatives. For example, gray | grey can match "gray" or "grey".

Grouping

Parentheses are used to define the scope and precedence of the operators (among other uses). For example, gray | grey and gr (a | e) y are equivalent patterns which both describe the set of "gray" and "grey".

Quantification

A quantifier after a token (such as a character) or group specifies how often that preceding element is allowed to occur. The most common quantifiers are the question mark ?, the asterisk * (derived from the Kleene star), and the plus sign + (Kleene cross).

- ? The question mark indicates there is *zero or one* of the preceding element. For example, colou?r matches both "color" and "colour".
- * The asterisk indicates there is *zero or more* of the preceding element. For example, ab*c matches "ac", "abc", "abbc", "abbcc", and so on.
- + The plus sign indicates there is *one or more* of the preceding element. For example, ab+c matches "abc", "abbc", "abbbc", and so on, but not "ac".

These constructions can be combined to form arbitrarily complex expressions, much like one can construct arithmetical expressions from numbers and the operations +, -, ×, and ÷. For example, H (ae? | ä) ndel and H (a | ae | ä) ndel are both valid patterns which match the same strings as the earlier example, H (ä | ae?) ndel.

The precise syntax for regular expressions varies among tools and with context; more detail is given in the *Syntax* section.

Formal language theory

Regular expressions describe regular languages in formal language theory. They have the same expressive power as regular grammars.

Formal definition

Regular expressions consist of constants and operator symbols that denote sets of strings and operations over these sets, respectively. The following definition is standard, and found as such in most textbooks on formal language theory.^{[5][6]} Given a finite alphabet Σ , the following constants are defined as regular expressions:

- (*empty set*) \emptyset denoting the set \emptyset .
- (*empty string*) ϵ denoting the set containing only the "empty" string, which has no characters at all.
- (*literal character*) a in Σ denoting the set containing only the character a .

Given regular expressions R and S , the following operations over them are defined to produce regular expressions:

- (*concatenation*) RS denoting the set $\{ \alpha\beta \mid \alpha \text{ in set described by expression } R \text{ and } \beta \text{ in set described by } S \}$. For example $\{ "ab", "c" \} \{ "d", "ef" \} = \{ "abd", "abef", "cd", "cef" \}$.
- (*alternation*) $R \mid S$ denoting the set union of sets described by R and S . For example, if R describes $\{ "ab", "c" \}$ and S describes $\{ "ab", "d", "ef" \}$, expression $R \mid S$ describes $\{ "ab", "c", "d", "ef" \}$.
- (*Kleene star*) R^* denoting the smallest superset of set described by R that contains ϵ and is closed under string concatenation. This is the set of all strings that can be made by concatenating any finite number (including zero) of strings from set described by R . For example, $\{ "0", "1" \}^*$ is the set of all finite binary strings (including the empty string), and $\{ "ab", "c" \}^* = \{ \epsilon, "ab", "c", "abab", "abc", "cab", "cc", "ababab", "abcab", ... \}$.

To avoid parentheses it is assumed that the Kleene star has the highest priority, then concatenation and then alternation. If there is no ambiguity then parentheses may be omitted. For example, $(ab)c$ can be written as abc , and $a \mid (b(c^*))$ can be written as $a \mid bc^*$. Many textbooks use the symbols \cup , $+$, or \vee for alternation instead of the vertical bar.

Examples:

- $a \mid b^*$ denotes $\{ \epsilon, "a", "b", "bb", "bbb", ... \}$
- $(a \mid b)^*$ denotes the set of all strings with no symbols other than "a" and "b", including the empty string: $\{ \epsilon, "a", "b", "aa", "ab", "ba", "bb", "aaa", ... \}$
- $ab^*(c \mid \epsilon)$ denotes the set of strings starting with "a", then zero or more "b"s and finally optionally a "c": $\{ "a", "ac", "ab", "abc", "abb", "abbc", ... \}$

Expressive power and compactness

The formal definition of regular expressions is purposely parsimonious and avoids defining the redundant quantifiers $?$ and $+$, which can be expressed as follows: $a^+ = aa^*$, and $a? = (a \mid \epsilon)$. Sometimes the complement operator is added, to give a *generalized regular expression*; here R^c matches all strings over Σ^* that do not match R . In principle, the complement operator is redundant, as it can always be circumscribed by using the other operators. However, the process for computing such a representation is complex, and the result may require expressions of a size that is double exponentially larger.^{[7][8]}

Regular expressions in this sense can express the regular languages, exactly the class of languages accepted by deterministic finite automata. There is, however, a significant difference in compactness. Some classes of regular languages can only be described by deterministic finite automata whose size grows exponentially in the size of the shortest equivalent regular expressions. The standard example here is the languages L_k consisting of all strings over the alphabet $\{a,b\}$ whose k^{th} -from-last letter equals a . On one hand, a regular expression describing L_4 is given by $(a|b)^*a(a|b)(a|b)(a|b)$. Generalizing this pattern to L_k gives the expression

$$(a|b)^* \underbrace{a(a|b)(a|b)\cdots(a|b)}_{k-1 \text{ times}}.$$

On the other hand, it is known that every deterministic finite automaton accepting the language L_k must have at least 2^k states. Luckily, there is a simple mapping from regular expressions to the more general nondeterministic finite automata (NFAs) that does not lead to such a blowup in size; for this reason NFAs are often used as alternative representations of regular languages. NFAs are a simple variation of the type-3 grammars of the Chomsky hierarchy.^[5]

Finally, it is worth noting that many real-world "regular expression" engines implement features that cannot be described by the regular expressions in the sense of formal language theory; see below for more on this.

Deciding equivalence of regular expressions

As seen in many of the examples above, there is more than one way to construct a regular expression to achieve the same results.

It is possible to write an algorithm which for two given regular expressions decides whether the described languages are essentially equal, reduces each expression to a minimal deterministic finite state machine, and determines whether they are isomorphic (equivalent).

The redundancy can be eliminated by using Kleene star and set union to find an interesting subset of regular expressions that is still fully expressive, but perhaps their use can be restricted. This is a surprisingly difficult problem. As simple as the regular expressions are, there is no method to systematically rewrite them to some normal form. The lack of axiom in the past led to the star height problem. In 1991, Dexter Kozen axiomatized regular expressions with Kleene algebra.^[9]

Syntax

A number of special characters or meta characters are used to denote actions or delimit groups; but it is possible to force these special characters to be interpreted as normal characters by preceding them with a defined escape character, usually the backslash "\". For example, a dot is normally used as a "wild card" metacharacter to denote any character, but if preceded by a backslash it represents the dot character itself. The pattern `c.t` matches "cat", "cot", "cut", and non-words such as "czt" and "c.t"; but `c\.t` matches only "c.t". The backslash also escapes itself, i.e., two backslashes are interpreted as a literal backslash character.

POSIX

POSIX Basic Regular Expressions

Traditional Unix regular expression syntax followed common conventions but often differed from tool to tool. The IEEE POSIX Basic Regular Expressions (BRE) standard (ISO/IEC 9945-2:1993 *Information technology -- Portable Operating System Interface (POSIX) -- Part 2: Shell and Utilities*, successively revised as ISO/IEC 9945-2:2002 *Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces*, ISO/IEC 9945-2:2003, and currently ISO/IEC/IEEE 9945:2009 *Information technology -- Portable Operating System Interface (POSIX®) Base Specifications, Issue 7*) was designed mostly for backward compatibility with the traditional (Simple Regular Expression) syntax but provided a common standard which has since been adopted as the default syntax of many Unix regular expression tools, though there is often some variation or additional features.

BRE was released alongside an alternative flavor called Extended Regular Expressions or ERE. Many Unix tools also provide support for ERE syntax with command line arguments.

In the BRE syntax, most characters are treated as literals — they match only themselves (e.g., `a` matches "a"). The exceptions, listed below, are called metacharacters or metasequences.

Metacharacter	Description
.	Matches any single character (many applications exclude newlines, and exactly which characters are considered newlines is flavor-, character-encoding-, and platform-specific, but it is safe to assume that the line feed character is included). Within POSIX bracket expressions, the dot character matches a literal dot. For example, <code>a . c</code> matches "abc", etc., but <code>[a . c]</code> matches only "a", ".", or "c".
[]	A bracket expression. Matches a single character that is contained within the brackets. For example, <code>[abc]</code> matches "a", "b", or "c". <code>[a-z]</code> specifies a range which matches any lowercase letter from "a" to "z". These forms can be mixed: <code>[abcx-z]</code> matches "a", "b", "c", "x", "y", or "z", as does <code>[a-cx-z]</code> . The <code>-</code> character is treated as a literal character if it is the last or the first (after the <code>^</code>) character within the brackets: <code>[abc-]</code> , <code>[-abc]</code> . Note that backslash escapes are not allowed. The <code>]</code> character can be included in a bracket expression if it is the first (after the <code>^</code>) character: <code>[] abc</code> .
[^]	Matches a single character that is not contained within the brackets. For example, <code>[^abc]</code> matches any character other than "a", "b", or "c". <code>[^a-z]</code> matches any single character that is not a lowercase letter from "a" to "z". Likewise, literal characters and ranges can be mixed.
^	Matches the starting position within the string. In line-based tools, it matches the starting position of any line.
\$	Matches the ending position of the string or the position just before a string-ending newline. In line-based tools, it matches the ending position of any line.
BRE: \(\)	Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, <code>\n</code>). A marked subexpression is also called a block or capturing group.
ERE: ()	
\n	Matches what the <i>n</i> th marked subexpression matched, where <i>n</i> is a digit from 1 to 9. This construct is theoretically irregular and was not adopted in the POSIX ERE syntax. Some tools allow referencing more than nine capturing groups.
*	Matches the preceding element zero or more times. For example, <code>ab*c</code> matches "ac", "abc", "abbcc", etc. <code>[xyz]*</code> matches "", "x", "y", "z", "zx", "zyx", "xyzzy", and so on. <code>(ab)*</code> matches "", "ab", "abab", "ababab", and so on.
BRE: \{m, n\}	Matches the preceding element at least <i>m</i> and not more than <i>n</i> times. For example, <code>a{3,5}</code> matches only "aaa", "aaaa", and "aaaaaa". This is not found in a few older instances of regular expressions.
ERE: {m, n}	

Examples:

- `.at` matches any three-character string ending with "at", including "hat", "cat", and "bat".
- `[hc]at` matches "hat" and "cat".
- `[^b]at` matches all strings matched by `.at` except "bat".
- `[^hc]at` matches all strings matched by `.at` other than "hat" and "cat".
- `^ [hc]at` matches "hat" and "cat", but only at the beginning of the string or line.
- `[hc]at$` matches "hat" and "cat", but only at the end of the string or line.
- `\[. \]` matches any single character surrounded by "[" and "]" since the brackets are escaped, for example: "[a]" and "[b]".

POSIX Extended Regular Expressions

The meaning of metacharacters escaped with a backslash is reversed for some characters in the POSIX Extended Regular Expression (ERE) syntax. With this syntax, a backslash causes the metacharacter to be treated as a literal character. So, for example, `\(\)` is now `()` and `\{ \}` is now `{ }`. Additionally, support is removed for `\n` backreferences and the following metacharacters are added:

Metacharacter	Description
?	Matches the preceding element zero or one time. For example, <code>ba?</code> matches "b" or "ba".
+	Matches the preceding element one or more times. For example, <code>ba+</code> matches "ba", "baa", "baaa", and so on.
	The choice (aka alternation or set union) operator matches either the expression before or the expression after the operator. For example, <code>abc def</code> matches "abc" or "def".

Examples:

- `[hc]+at` matches "hat", "cat", "hhat", "chat", "hcat", "ccchat", and so on, but not "at".
- `[hc]?at` matches "hat", "cat", and "at".
- `[hc]*at` matches "hat", "cat", "hhat", "chat", "hcat", "ccchat", "at", and so on.
- `cat|dog` matches "cat" or "dog".

POSIX Extended Regular Expressions can often be used with modern Unix utilities by including the command line flag `-E`.

POSIX character classes

Since many ranges of characters depend on the chosen locale setting (i.e., in some settings letters are organized as *abc...zABC...Z*, while in some others as *aAbBcC...zZ*), the POSIX standard defines some classes or categories of characters as shown in the following table:

POSIX	Non-standard	Perl	Vim	ASCII	Description
<code>[:alnum:]</code>				<code>[A-Za-z0-9]</code>	Alphanumeric characters
	<code>[:word:]</code>	<code>\w</code>	<code>\w</code>	<code>[A-Za-z0-9_]</code>	Alphanumeric characters plus "_"
		<code>\W</code>	<code>\W</code>	<code>[^A-Za-z0-9_]</code>	Non-word characters
<code>[:alpha:]</code>			<code>\a</code>	<code>[A-Za-z]</code>	Alphabetic characters
<code>[:blank:]</code>				<code>[\t]</code>	Space and tab
		<code>\b</code>	<code>\< \></code>	<code>(?=<\w) (=?=\w) (?=<=\w) (=?=\w)</code>	Word boundaries
<code>[:cntrl:]</code>				<code>[\x00-\x1F\x7F]</code>	Control characters
<code>[:digit:]</code>		<code>\d</code>	<code>\d</code>	<code>[0-9]</code>	Digits
		<code>\D</code>	<code>\D</code>	<code>[^0-9]</code>	Non-digits
<code>[:graph:]</code>				<code>[\x21-\x7E]</code>	Visible characters
<code>[:lower:]</code>			<code>\l</code>	<code>[a-z]</code>	Lowercase letters
<code>[:print:]</code>			<code>\p</code>	<code>[\x20-\x7E]</code>	Visible characters and the space character
<code>[:punct:]</code>				<code>[\!] [\!#"#\$%&' () *+, ./:; <=>?@^\^` { } ~-]</code>	Punctuation characters
<code>[:space:]</code>		<code>\s</code>	<code>\s</code>	<code>[\t\r\n\f]</code>	Whitespace characters
		<code>\S</code>	<code>\S</code>	<code>[^ \t\r\n\f]</code>	Non-whitespace characters
<code>[:upper:]</code>			<code>\u</code>	<code>[A-Z]</code>	Uppercase letters
<code>[:xdigit:]</code>			<code>\x</code>	<code>[A-Fa-f0-9]</code>	Hexadecimal digits

POSIX character classes can only be used within bracket expressions. For example, `[[:upper:] ab]` matches the uppercase letters and lowercase "a" and "b".

An additional non-POSIX class understood by some tools is [:word:], which is usually defined as [:alnum:] plus underscore. This reflects the fact that in many programming languages these are the characters that may be used in identifiers. The editor Vim further distinguishes *word* and *word-head* classes (using the notation \w and \h) since in many programming languages the characters that can begin an identifier are not the same as those that can occur in other positions.

Note that what the POSIX regular expression standards call *character classes* are commonly referred to as *POSIX character classes* in other regular expression flavors which support them. With most other regular expression flavors, the term *character class* is used to describe what POSIX calls *bracket expressions*.

Perl-derived regular expressions

Perl has a more consistent and richer syntax than the POSIX basic (BRE) and extended (ERE) regular expression standards. An example of its consistency is that \ always escapes a non-alphanumeric character. Other examples of functionality possible with Perl but not POSIX-compliant regular expressions is the concept of lazy quantification (see the next section), possessive quantifiers to control backtracking, named capture groups, and recursive patterns.

Due largely to its expressive power, many other utilities and programming languages have adopted syntax similar to Perl's — for example, Java, JavaScript, Python, Ruby, Microsoft's .NET Framework, and the W3C's XML Schema all use regular expression syntax similar to Perl's. Some languages and tools such as Boost and PHP support multiple regular expression flavors. Perl-derivative regular expression implementations are not identical, and all implement no more than a subset of Perl's features, usually those of Perl 5.0, released in 1994. With Perl 5.10, this process has come full circle with Perl incorporating syntactic extensions originally developed in Python and PCRE "Perl Regular Expression Documentation" [10]. perldoc.perl.org. Retrieved January 8, 2012.

Simple Regular Expressions

Simple Regular Expressions is a syntax that may be used by historical versions of application programs, and may be supported within some applications for the purpose of providing backward compatibility. It is deprecated.^[11]

Lazy quantification

The standard quantifiers in regular expressions are greedy, meaning they match as much as they can. For example, to find the first instance of an item between the angled bracket symbols <> in this example:

```
Another whale sighting occurred on <January 26>, <2004>.
```

someone new to regexes would likely come up with the pattern <. *> or similar. However, instead of the "<January 26>" that might be expected, this pattern will actually return "<January 26>, <2004>" because the * quantifier is greedy — it will consume as many characters as possible from the input, and "<January 26>, <2004>" has more characters than "<January 26>".

Though this problem can be avoided in a number of ways (e.g., by specifying the text that is *not* to be matched: <[^>]*>), modern regular expression tools allow a quantifier to be specified as *lazy* (also known as *non-greedy*, *reluctant*, *minimal*, or *ungreedy*) by putting a question mark after the quantifier (e.g., <. *?>), or by using a modifier which reverses the greediness of quantifiers (though changing the meaning of the standard quantifiers can be confusing). By using a lazy quantifier, the expression tries the minimal match first. Though in the previous example lazy matching is used to select one of many matching results, in some cases it can also be used to improve performance when greedy matching would require more backtracking.

Patterns for non-regular languages

Many features found in modern regular expression libraries provide an expressive power that far exceeds the regular languages. For example, many implementations allow grouping subexpressions with parentheses and recalling the value they match in the same expression (**backreferences**). This means that a pattern can match strings of repeated words like "papa" or "WikiWiki", called *squares* in formal language theory. The pattern for these strings is `(. *) \1`. The language of squares is not regular, nor is it context-free. Pattern matching with an unbounded number of back references, as supported by numerous modern tools, is NP-complete.^[12]

However, many tools, libraries, and engines that provide such constructions still use the term *regular expression* for their patterns. This has led to a nomenclature where the term regular expression has different meanings in formal language theory and pattern matching. For this reason, some people have taken to using the term *regex* or simply *pattern* to describe the latter. Larry Wall, author of the Perl programming language, writes in an essay about the design of Perl 6:

“Regular expressions' [...] are only marginally related to real regular expressions. Nevertheless, the term has grown with the capabilities of our pattern matching engines, so I'm not going to try to fight linguistic necessity here. I will, however, generally call them "regexes" (or "regexen",^[4] when I'm in an Anglo-Saxon mood).”

Fuzzy Regular Expressions

Variants of regular expressions can be used for working with text in natural language, when it is necessary to take into account possible typos and spelling variants. For example, the text "Julius Caesar" might be a fuzzy match for:

- Gaius Julius Caesar
- Yulius Cesar
- G. Juliy Caezar

In such cases the mechanism implements some fuzzy string matching algorithm and possibly some algorithm for finding the similarity between text fragment and pattern.

This task is closely related to both full text search and named entity recognition.

Some software libraries work with fuzzy regular expressions:

- TRE - well-developed portable free project in C, which uses syntax similar to POSIX
- FREJ - open source project in Java with non-standard syntax (which utilizes prefix, Lisp-like notation), targeted to allow easy use of substitutions of inner matched fragments in outer blocks, but lacks many features of standard regular expressions.
- agrep - command-line utility (proprietary, but free for non-commercial usage).

Implementations and running times

There are at least three different algorithms that decide if and how a given regular expression matches a string.

The oldest and fastest rely on a result in formal language theory that allows every nondeterministic finite automaton (NFA) to be transformed into a deterministic finite automaton (DFA). The DFA can be constructed explicitly and then run on the resulting input string one symbol at a time. Constructing the DFA for a regular expression of size m has the time and memory cost of $O(2^m)$, but it can be run on a string of size n in time $O(n)$. An alternative approach is to simulate the NFA directly, essentially building each DFA state on demand and then discarding it at the next step. This keeps the DFA implicit and avoids the exponential construction cost, but running cost rises to $O(m^2n)$. The explicit approach is called the DFA algorithm and the implicit approach the NFA algorithm. Adding caching to the NFA algorithm is often called the "lazy DFA" algorithm, or just the DFA algorithm without making a distinction. These algorithms are fast, but using them for recalling grouped subexpressions, lazy quantification, and similar

features is tricky.^{[13][14]}

The third algorithm is to match the pattern against the input string by backtracking. This algorithm is commonly called NFA, but this terminology can be confusing. Its running time can be exponential, which simple implementations exhibit when matching against expressions like $(a|aa)^*b$ that contain both alternation and unbounded quantification and force the algorithm to consider an exponentially increasing number of sub-cases. This behavior can cause a security problem called Regular expression Denial of Service.

Although backtracking implementations only give an exponential guarantee in the worst case, they provide much greater flexibility and expressive power. For example, any implementation which allows the use of backreferences, or implements the various extensions introduced by Perl, must include some kind of backtracking. Some implementations try to provide the best of both algorithms by first running a fast DFA algorithm, and revert to a potentially slower backtracking algorithm only when a backreference is encountered during the match.

Unicode

In theoretical terms, any token set can be matched by regular expressions as long as it is pre-defined. In terms of historical implementations, regular expressions were originally written to use ASCII characters as their token set though regular expression libraries have supported numerous other character sets. Many modern regular expression engines offer at least some support for Unicode. In most respects it makes no difference what the character set is, but some issues do arise when extending regular expressions to support Unicode.

- Supported encoding. Some regular expression libraries expect to work on some particular encoding instead of on abstract Unicode characters. Many of these require the UTF-8 encoding, while others might expect UTF-16, or UTF-32. In contrast, Perl and Java are agnostic on encodings, instead operating on decoded characters internally.
- Supported Unicode range. Many regular expression engines support only the Basic Multilingual Plane, that is, the characters which can be encoded with only 16 bits. Currently, only a few regular expression engines (e.g., Perl's and Java's) can handle the full 21-bit Unicode range.
- Extending ASCII-oriented constructs to Unicode. For example, in ASCII-based implementations, character ranges of the form $[x-y]$ are valid wherever x and y are codepoints in the range $[0x00,0x7F]$ and $\text{codepoint}(x) \leq \text{codepoint}(y)$. The natural extension of such character ranges to Unicode would simply change the requirement that the endpoints lie in $[0x00,0x7F]$ to the requirement that they lie in $[0,0x10FFFF]$. However, in practice this is often not the case. Some implementations, such as that of gawk, do not allow character ranges to cross Unicode blocks. A range like $[0x61,0x7F]$ is valid since both endpoints fall within the Basic Latin block, as is $[0x0530,0x0560]$ since both endpoints fall within the Armenian block, but a range like $[0x0061,0x0532]$ is invalid since it includes multiple Unicode blocks. Other engines, such as that of the Vim editor, allow block-crossing but limit the number of characters in a range to 128.
- Case insensitivity. Some case-insensitivity flags affect only the ASCII characters. Other flags affect all characters. Some engines have two different flags, one for ASCII, the other for Unicode. Exactly which characters belong to the POSIX classes also varies.
- Cousins of case insensitivity. As ASCII has case distinction, case insensitivity became a logical feature in text searching. Unicode introduced alphabetic scripts without case like Devanagari. For these, case sensitivity is not applicable. For scripts like Chinese, another distinction seems logical: between traditional and simplified. In Arabic scripts, insensitivity to initial, medial, final, and isolated position may be desired. In Japanese, insensitivity between hiragana and katakana is sometimes useful.
- Normalization. Unicode has combining characters. Like old typewriters, plain letters can be followed by one or more non-spacing symbols (usually diacritics like accent marks) to form a single printing character, but also provides precomposed characters, i.e. characters that already include one or more combining characters. A sequence of a character + combining character should be matched with the identical single precomposed character. The process of standardizing sequences of characters + combining characters is called normalization.

- New control codes. Unicode introduced amongst others, byte order marks and text direction markers. These codes might have to be dealt with in a special way.
- Introduction of character classes for Unicode blocks, scripts, and numerous other character properties. Block properties are much less useful than script properties, because a block can have code points from several different scripts, and a script can have code points from several different blocks.^[15] In Perl and the `java.util.regex` library, properties of the form `\p{InX}` or `\p{Block=X}` match characters in block X and `\P{InX}` or `\P{Block=X}` matches code points not in that block. Similarly, `\p{Armenian}`, `\p{IsArmenian}`, or `\p{Script=Armenian}` matches any character in the Armenian script. In general, `\p{X}` matches any character with either the binary property X or the general category X. For example, `\p{Lu}`, `\p{Uppercase_Letter}`, or `\p{GC=Lu}` matches any upper-case letter. Binary properties that are *not* general categories include `\p{White_Space}`, `\p{Alphabetic}`, `\p{Math}`, and `\p{Dash}`. Examples of non-binary properties are `\p{Bidi_Class=Right_to_Left}`, `\p{Word_Break=A_Letter}`, and `\p{Numeric_Value=10}`.

Uses

Regular expressions are useful in the production of syntax highlighting systems, data validation, and many other tasks.

While regular expressions would be useful on Internet search engines, processing them across the entire database could consume excessive computer resources depending on the complexity and design of the regex. Although in many cases system administrators can run regex-based queries internally, most search engines do not offer regex support to the public. Notable exceptions: Google Code Search, Exalead.

Examples

A regular expression is a string that is used to describe or match a set of strings according to certain syntax rules. The specific syntax rules vary depending on the specific implementation, programming language, or library in use. Additionally, the functionality of regex implementations can vary between versions.

Despite this variability, and because regular expressions can be difficult to both explain and understand without examples, this article provides a basic description of some of the properties of regular expressions by way of illustration.

The following conventions are used in the examples.^[16]

```
metacharacter(s) ;; the metacharacters column specifies the regex syntax being demonstrated
 =~ m// ;; indicates a regex match operation in Perl
 =~ s/// ;; indicates a regex substitution operation in Perl
```

Also worth noting is that these regular expressions are all Perl-like syntax. Standard POSIX regular expressions are different.

Unless otherwise indicated, the following examples conform to the Perl programming language, release 5.8.8, January 31, 2006. This means that other implementations may lack support for some parts of the syntax shown here (e.g. basic vs. extended regex, `\(\)` vs. `()`, or lack of `\d` instead of POSIX `[:digit:]`).

The syntax and conventions used in these examples coincide with that of other programming environments as well (e.g., see *Java in a Nutshell* — Page 213, *Python Scripting for Computational Science* — Page 320, Programming PHP — Page 106).

Character(s)	Description	Example Note that all the if statements return a TRUE value
.	Normally matches any character except a newline. Within square brackets the dot is literal.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/...../) { print "\$string1 has length >= 5\n"; } </pre>
()	Groups a series of pattern elements to a single element. When you match a pattern within parentheses, you can use any of \$1, \$2, ... later to refer to the previously matched pattern.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/(H..).(\o..)/) { print "We matched '\$1' and '\$2'\n"; } </pre> <p>Output:</p> <pre> We matched 'Hel' and 'o W'; </pre>
*	Matches the preceding pattern element one or more times.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/l+/) { print "There are one or more consecutive letter \"l\" in \$string1\n"; } </pre> <p>Output:</p> <pre> There are one or more consecutive letter "l"'s in Hello </pre>
?	Matches the preceding pattern element zero or one times.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/H.?e/) { print "There is an 'H' and a 'e' separated by "; print "0-1 characters (Ex: He Hoe)\n"; } </pre>
{}{M,N}	Modifies the *, +, or {M,N}'d regex that comes before to match as few times as possible.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/(l.+?o)/) { print "The non-greedy match with 'l' followed by one o print "more characters is 'llo' rather than 'llo wo'.\\n" } </pre>
*	Matches the preceding pattern element zero or more times.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/el*o/) { print "There is an 'e' followed by zero to many "; print "'l' followed by 'o' (eo, elo, ello, elllo)\n"; } </pre>

Denotes the minimum M and the maximum N match count.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/l{1,2}/) { print "There exists a substring with at least 1 "; print "and at most 2 l's in \$string1\n"; } </pre>
Denotes a set of possible character matches.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/[aeiou]+/) { print "\$string1 contains one or more vowels.\n"; } </pre>
Separates alternate possibilities.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/(Hello Hi Pogo)/) { print "At least one of Hello, Hi, or Pogo is "; print "contained in \$string1.\n"; } </pre>
Matches a zero-width boundary between a word-class character (see next) and either a non-word class character or an edge.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/llo\b/) { print "There is a word that ends with 'llo'\n"; } </pre>
Matches an alphanumeric character, including "_"; same as [A-Za-z0-9_] in ASCII. In Unicode ^[15] same as \p{Alphabetic}\p{GC=Mark}\p{GC=Decimal_Number}\p{GC=Connector_Punctuation}, where the Alphabetic property contains more than just Letters, and the Decimal_Number property contains more than [0-9].	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/\w/) { print "There is at least one alphanumeric "; print "character in \$string1 (A-Z, a-z, 0-9, _) \n"; } </pre>
Matches a non -alphanumeric character, excluding "_"; same as [^A-Za-z0-9_] in ASCII, and [^\p{Alphabetic}\p{GC=Mark}\p{GC=Decimal_Number}\p{GC=Connector_Punctuation} in Unicode.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/\W/) { print "The space between Hello and "; print "World is not alphanumeric\n"; } </pre>
Matches a whitespace character, which in ASCII are tab, line feed, form feed, carriage return, and space; in Unicode, also matches no-break spaces, next line, and the variable-width spaces (amongst others).	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/\s.*\s/) { print "There are TWO whitespace characters, which may "; print "be separated by other characters, in \$string1"; } </pre>
Matches anything BUT a whitespace.	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/\S.*\S/) { print "There are TWO non-whitespace characters, which "; print "may be separated by other characters, in \$string1"; } </pre>

<p>Matches a digit; same as [0-9] in ASCII; in Unicode, same as the \p{Digit} or \p{GC=Decimal_Number} property, which itself the same as the \p{Numeric_Type=Decimal} property.</p>	<pre> \$string1 = "99 bottles of beer on the wall."; if (\$string1 =~ m/(\d+)/) { print "\$1 is the first number in '\$string1'\n"; } </pre> <p>Output:</p> <pre> 99 is the first number in '99 bottles of beer on the wal </pre>
<p>Matches a non-digit; same as [^0-9] in ASCII or \P{Digit} in Unicode.</p>	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/\D/) { print "There is at least one character in \$string1"; print " that is not a digit.\n"; } </pre>
<p>Matches the beginning of a line or string.</p>	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/^He/) { print "\$string1 starts with the characters 'He'\n"; } </pre>
<p>Matches the end of a line or string.</p>	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/rld\$/) { print "\$string1 is a line or string "; print "that ends with 'rld'\n"; } </pre>
<p>Matches the beginning of a string (but not an internal line).</p>	<pre> \$string1 = "Hello\nWorld\n"; if (\$string1 =~ m/\AH/) { print "\$string1 is a string "; print "that starts with 'H'\n"; } </pre>
<p>Matches the end of a string (but not an internal line). see Perl Best Practices — Page 240</p>	<pre> \$string1 = "Hello\nWorld\n"; if (\$string1 =~ m/d\n\z/) { print "\$string1 is a string "; print "that ends with 'd\\n'\n"; } </pre>
<p>Matches every character except the ones inside brackets.</p>	<pre> \$string1 = "Hello World\n"; if (\$string1 =~ m/[^abc]/) { print "\$string1 contains a character other than "; print "a, b, and c\n"; } </pre>

Notes

- [1] Kleene (1956)
- [2] Raymond, Eric S. citing Dennis Ritchie (2003). "Jargon File 4.4.7: grep" (<http://catb.org/jargon/html/G/grep.html>). .
- [3] Wall, Larry and the Perl 5 development team (2006). "perlre: Perl regular expressions" (<http://perldoc.perl.org/perlre.html>). .
- [4] Wall (2002)
- [5] Hopcroft, Motwani & Ullman (2000)
- [6] Sipser (1998)
- [7] Gelade & Neven (2008)
- [8] Gruber & Holzer (2008)
- [9] Kozen (1991)
- [10] <http://perldoc.perl.org/perlre.html#PCRE%2fPython-Support>
- [11] The Single Unix Specification (Version 2)
- [12] see Aho (1990) Theorem 6.2
- [13] Cox (2007)
- [14] Laurikari (2009)
- [15] "UTS#18 on Unicode Regular Expressions, Annex A: Character Blocks" (http://unicode.org/reports/tr18/#Character_Blocks). . Retrieved 2010-02-05.
- [16] The character 'm' is not always required to specify a Perl match operation. For example, m/[[^]abc]/ could also be rendered as /[^abc]/. The 'm' is only necessary if the user wishes to specify a match operation without using a forward-slash as the regex delimiter. Sometimes it is useful to specify an alternate regex delimiter in order to avoid "delimiter collision". See ' perldoc perlre (<http://perldoc.perl.org/perlre.html>)' for more details.

References

- Aho, Alfred V. (1990). "Algorithms for finding patterns in strings". In van Leeuwen, Jan. *Handbook of Theoretical Computer Science, volume A: Algorithms and Complexity*. The MIT Press. pp. 255–300
- "Regular Expressions" (<http://pubs.opengroup.org/onlinepubs/007908799/xbd/re.html>). *The Single UNIX® Specification, Version 2*. The Open Group. 1997
- "Chapter 9: Regular Expressions" (http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html). *The Open Group Base Specifications Issue 6, IEEE Std 1003.1, 2004 Edition*. The Open Group. 2004
- Cox, Russ (2007). "Regular Expression Matching Can Be Simple and Fast" (<http://swtch.com/~rsc/regexp/regexp1.html>)
- Forta, Ben (2004). *Sams Teach Yourself Regular Expressions in 10 Minutes*. Sams. ISBN 0-672-32566-7.
- Friedl, Jeffrey (2002). *Mastering Regular Expressions* (<http://regex.info/>). O'Reilly. ISBN 0-596-00289-0.
- Gelade, Wouter; Neven, Frank (2008). "Succinctness of the Complement and Intersection of Regular Expressions" (<http://drops.dagstuhl.de/opus/volltexte/2008/1354>). *Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008)*. pp. 325–336
- Gruber, Hermann; Holzer, Markus (2008). "Finite Automata, Digraph Connectivity, and Regular Expression Size" (<http://www.hermann-gruber.com/data/icalp08.pdf>). *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)*. **5126**. pp. 39–50. doi:10.1007/978-3-540-70583-3_4
- Habibi, Mehran (2004). *Real World Regular Expressions with Java 1.4*. Springer. ISBN 1-59059-107-0.
- Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2000). *Introduction to Automata Theory, Languages, and Computation* (2nd ed.). Addison-Wesley.
- Kleene, Stephen C. (1956). "Representation of Events in Nerve Nets and Finite Automata". In Shannon, Claude E.; McCarthy, John. *Automata Studies*. Princeton University Press. pp. 3–42
- Kozen, Dexter (1991). "A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events". *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science (LICS 1991)*. pp. 214–225
- Laurikari, Ville (2009). "TRE library 0.7.6" (<http://www.laurikari.net/tre/>).
- Liger, Francois; Craig McQueen, Paul Wilton (2002). *Visual Basic .NET Text Manipulation Handbook*. Wrox Press. ISBN 1-86100-730-2.

- Sipser, Michael (1998). "Chapter 1: Regular Languages". *Introduction to the Theory of Computation*. PWS Publishing. pp. 31–90. ISBN 0-534-94728-X.
- Stubblebine, Tony (2003). *Regular Expression Pocket Reference*. O'Reilly. ISBN 0-596-00415-X.
- Wall, Larry (2002). "Apocalypse 5: Pattern Matching" (<http://dev.perl.org/perl6/doc/design/apo/A05.html>).
- Goyvaerts, Jan; [Jan Goyvaerts], [Steven Levithan] (2009). *Regular Expressions Cookbook*. [O'reilly]. ISBN 978-0-596-52068-7.

External links

- ISO/IEC 9945-2:1993 *Information technology -- Portable Operating System Interface (POSIX) -- Part 2: Shell and Utilities* (http://www.iso.org/iso/catalogue_detail.htm?csnumber=17841)
- ISO/IEC 9945-2:2002 *Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces* (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=37313)
- ISO/IEC 9945-2:2003 *Information technology -- Portable Operating System Interface (POSIX) -- Part 2: System Interfaces* (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=38790)
- ISO/IEC/IEEE 9945:2009 *Information technology -- Portable Operating System Interface (POSIX®) Base Specifications, Issue 7* (http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=50516)
- Java Tutorials: Regular Expressions (<http://java.sun.com/docs/books/tutorial/essential/regex/index.html>)
- Perl Regular Expressions documentation (<http://perldoc.perl.org/perlre.html>)
- VBScript and Regular Expressions (<http://msdn2.microsoft.com/en-us/library/ms974570.aspx>)
- .NET Framework Regular Expressions (<http://msdn.microsoft.com/en-us/library/hs600312.aspx>)
- Regular Expressions (http://www.dmoz.org/Computers/Programming/Languages/Regular_Expressions/) at the Open Directory Project
- Pattern matching tools and libraries (<http://billposer.org/Linguistics/Computation/Resources.html#patterns>)
- Structural Regular Expressions by Rob Pike (http://doc.cat-v.org/bell_labs/structural_regexps/)
- JavaScript Regular Expressions Chapter (https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Regular_Expressions) and RegExp Object Reference (https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Global_Objects/RegExp) at the Mozilla Developer Center

Generative grammar

Generative grammar

In theoretical linguistics, **generative grammar** refers to a particular approach to the study of syntax. A generative grammar of a language attempts to give a set of rules that will correctly predict which combinations of words will form grammatical sentences. In most approaches to generative grammar, the rules will also predict the morphology of a sentence. Generative grammar arguably originates in the work of Noam Chomsky, beginning in the late 1950s. However, Chomsky has said that the first generative grammar in the modern sense was Panini's Sanskrit grammar.^[1] Chomsky also acknowledges other historical antecedents.^[2]

Early versions of Chomsky's theory were called transformational grammar, and this term is still used as a general term that includes his subsequent theories. There are a number of competing versions of generative grammar currently practiced within linguistics. Chomsky's current theory is known as the Minimalist program. Other prominent theories include or have included dependency grammar, head-driven phrase structure grammar, lexical functional grammar, categorial grammar, relational grammar, link grammar, and tree-adjoining grammar.

Chomsky has argued that many of the properties of a generative grammar arise from an "innate" universal grammar. Proponents of generative grammar have argued that most grammar is not the result of communicative function and is not simply learned from the environment (see poverty of the stimulus argument). In this respect, generative grammar takes a point of view different from cognitive grammar, functional, and behaviorist theories.

Most versions of generative grammar characterize sentences as either grammatically correct (also known as *well formed*) or not. The rules of a generative grammar typically function as an algorithm to predict grammaticality as a discrete (yes-or-no) result. In this respect, it differs from stochastic grammar, which considers grammaticality as a probabilistic variable. However, some work in generative grammar (e.g. recent work by Joan Bresnan) uses stochastic versions of optimality theory.

Frameworks

There are a number of different approaches to generative grammar. Common to all is the effort to come up with a set of rules or principles that formally defines each and every one of the members of the set of well-formed expressions of a natural language. The term *generative grammar* has been associated with at least the following schools of linguistics:

- Transformational grammar (TG)
 - Standard Theory (ST)
 - Extended Standard Theory (EST)
 - Revised Extended Standard Theory (REST)
 - Principles and Parameters Theory (P&P)
 - Government and Binding Theory (GB)
 - Minimalist Program (MP)
- Monostratal (or non-transformational) grammars
 - Relational Grammar (RG)
 - Lexical-Functional Grammar (LFG)
 - Generalized Phrase Structure Grammar (GPSG)
 - Head-Driven Phrase Structure Grammar (HPSG)

- Categorial Grammar
- Tree-Adjoining Grammar

Historical development of models of transformational grammar

Chomsky, in an award acceptance speech delivered in India in 2001, claimed "The first generative grammar in the modern sense was Panini's grammar".^[1] This work, called the Ashtadhyayi, was composed by the middle of the 1st millennium BCE.

Generative grammar has been under development since the late 1950s, and has undergone many changes in the types of rules and representations that are used to predict grammaticality. In tracing the historical development of ideas within generative grammar, it is useful to refer to various stages in the development of the theory.

Standard Theory (1957–1965)

The so-called Standard Theory corresponds to the original model of generative grammar laid out in Chomsky (1965). A core aspect of Standard Theory is a distinction between two different representations of a sentence, called Deep structure and Surface structure. The two representations are linked to each other by transformational grammar.

Extended Standard Theory (1965–1973)

The so-called Extended Standard Theory was formulated in the late 1960s to early 1970s. Features are:

- syntactic constraints
- generalized phrase structures (X-bar theory)

Revised Extended Standard Theory (1973–1976)

The so-called Revised Extended Standard Theory was formulated between 1973 and 1976. It contains

- restrictions upon X-bar theory (Jackendoff (1977)).
- assumption of the COMP position.
- Move α

Relational grammar (ca. 1975–1990)

An alternative model of syntax based on the idea that notions like Subject, Direct Object, and Indirect Object play a primary role in grammar.

Government and binding/Principles and parameters theory (1981–1990)

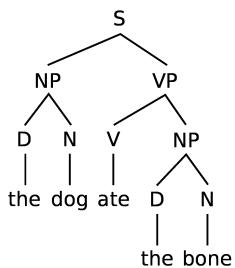
Chomsky's *Lectures on Government and Binding* (1981) and *Barriers* (1986).

Context-free grammars

Generative grammars can be described and compared with the aid of the Chomsky hierarchy proposed by Noam Chomsky in the 1950s. This sets out a series of types of formal grammars with increasing expressive power. Among the simplest types are the regular grammars (type 3); Chomsky claims that regular grammars are not adequate as models for human language, because all human languages allow the center-embedding of strings within strings.

At a higher level of complexity are the context-free grammars (type 2). The derivation of a sentence by a grammar can be depicted as a derivation tree. Linguists working in generative grammar often view such derivation trees as a primary object of study. According to this view, a sentence is not merely a string of words, but rather a tree with subordinate and superordinate branches connected at nodes.

Essentially, the tree model works something like this example, in which S is a sentence, D is a determiner, N a noun, V a verb, NP a noun phrase and VP a verb phrase:



The resulting sentence could be *The dog ate the bone*. Such a tree diagram is also called a phrase marker. They can be represented more conveniently in text form, (though the result is less easy to read); in this format the above sentence would be rendered as:

[S [NP [D The] [N dog]] [VP [V ate] [NP [D the] [N bone]]]]]

Chomsky has argued that phrase structure grammars are also inadequate for describing natural languages, and has formulated the more complex system of transformational grammar.^[3]

Grammaticality judgments

When generative grammar was first proposed, it was widely hailed as a way of formalizing the implicit set of rules a person "knows" when they know their native language and produce grammatical utterances in it (grammaticality intuitions). However Chomsky has repeatedly rejected that interpretation; according to him, the grammar of a language is a statement of what it is that a person has to know in order to recognize an utterance as grammatical, but not a hypothesis about the processes involved in either understanding or producing language.

Music

Generative grammar has been used to a limited extent in music theory and analysis since the 1980s.^{[4][5]} The most well-known approaches were developed by Mark Steedman^[6] as well as Fred Lerdahl and Ray Jackendoff,^[7] who formalised and extended ideas from Schenkerian analysis.^[8] More recently, such early generative approaches to music were further developed and extended by several scholars.^{[9][10][11][12]}

References

- [1] S.S. Chattopadhyay, event in Kolkata (<http://www.frontlineonnet.com/fl1825/18250150.htm>"An), *Frontline*
- [2] Another example is Humboldt. Chomsky quotes Humboldt's description of language as a system which "makes infinite use of finite means".
- [3] Chomsky, Noam (1956). "Three models for the description of language" (<http://www.chomsky.info/articles/195609--.pdf>). *IRE Transactions on Information Theory* 2 (3): 113–124. doi:10.1109/TIT.1956.1056813..
- [4] Baroni, M., Maguire, S., and Drabkin, W. (1983). The Concept of Musical Grammar. *Music Analysis*, 2:175–208.
- [5] Baroni, M. and Callegari, L. (1982) Eds., Musical grammars and computer analysis. Leo S. Olschki Editore: Firenze, 201–218.
- [6] Steedman, M.J. (1989). "A Generative Grammar for Jazz Chord Sequences". *Music Perception* 2 (1): 52–77. JSTOR 40285282.
- [7] Lerdahl, Fred; Ray Jackendoff (1996). *A Generative Theory of Tonal Music*. Cambridge: MIT Press. ISBN 978-0-262-62107-6.
- [8] Heinrich Schenker, Free Composition. (Der Freie Satz) translated and edited by Ernst Ostler. New York: Longman, 1979.
- [9] Tojo, O. Y. & Nishida, M. (2006). Analysis of chord progression by HPSG. In Proceedings of the 24th IASTED international conference on Artificial intelligence and applications, 305–310.
- [10] Rohrmeier, Martin (2007). A generative grammar approach to diatonic harmonic structure. In Spyridis, Georgaki, Kouroupetroglou, Anagnostopoulou (Eds.), Proceedings of the 4th Sound and Music Computing Conference, 97–100. <http://smc07.uoa.gr/SMC07%20Proceedings/SMC07%20Paper%202015.pdf>
- [11] Giblin, Iain (2008). Music and the generative enterprise. Doctoral dissertation. University of New South Wales.
- [12] Katz, Jonah; David Pesetsky (2009) "The Identity Thesis for Language and Music". <http://ling.auf.net/lingBuzz/000959>

Further reading

- Chomsky, Noam. 1965. Aspects of the theory of syntax. Cambridge, Massachusetts: MIT Press.
- Hurford, J. (1990) *Nativist and functional explanations in language acquisition*. In I. M. Roca (ed.), Logical Issues in Language Acquisition, 85–136. Foris, Dordrecht.
- Isac, Daniela; Charles Reiss (2008). *I-language: An Introduction to Linguistics as Cognitive Science* (<http://linguistics.concordia.ca/i-language/>). Oxford University Press. ISBN 978-0-19-953420-3.

Transformational grammar

In linguistics, a **transformational grammar** or **transformational-generative grammar (TGG)** is a generative grammar, especially of a natural language, that has been developed in the Chomskyan tradition of phrase structure grammars (as opposed to dependency grammars). Additionally, transformational grammar is the tradition that gives rise to specific transformational grammars. Much current research in transformational grammar is inspired by Chomsky's Minimalist Program.^[1]

Deep structure and surface structure

In 1957, Noam Chomsky published *Syntactic Structures*, in which he developed the idea that each sentence in a language has two levels of representation — a deep structure and a *surface structure*.^{[2][3]} The deep structure represented the core semantic relations of a sentence, and was mapped on to the surface structure (which followed the phonological form of the sentence very closely) via *transformations*. Chomsky believed there are considerable similarities between languages' deep structures, and that these structures reveal properties, common to all languages that surface structures conceal. However, this may not have been the central motivation for introducing deep structure. Transformations had been proposed prior to the development of deep structure as a means of increasing the mathematical and descriptive power of context-free grammars. Similarly, deep structure was devised largely for technical reasons relating to early semantic theory. Chomsky emphasizes the importance of modern formal mathematical devices in the development of grammatical theory:

But the fundamental reason for [the] inadequacy of traditional grammars is a more technical one. Although it was well understood that linguistic processes are in some sense "creative," the technical devices for expressing a system of recursive processes were simply not available until much more recently. In fact, a real understanding of how a language can (in Humboldt's words) "make infinite use of finite means" has developed only within the last thirty years, in the course of studies in the foundations of mathematics.

—*Aspects of the Theory of Syntax*

Formal definition

Chomsky's advisor, Zellig Harris, took transformations to be relations between sentences such as "I finally met this talkshow host you always detested" and simpler (kernel) sentences "I finally met this talkshow host" and "You always detested this talkshow host". Chomsky developed a formal theory of grammar where transformations manipulated not just the surface strings, but the parse tree associated to them, making transformational grammar a system of tree automata.^[4] This definition proved adequate for subsequent versions including the 'extended', 'revised extended', and 'Government-Binding' (GB) versions of generative grammar, but may no longer be sufficient for the current minimalist grammar in that merge may require a formal definition that goes beyond the tree manipulation characteristic of Move α .

Development of basic concepts

Though transformations continue to be important in Chomsky's current theories, he has now abandoned the original notion of Deep Structure and Surface Structure. Initially, two additional levels of representation were introduced (LF — Logical Form, and PF — Phonetic Form), and then in the 1990s Chomsky sketched out a new program of research known as *Minimalism*, in which Deep Structure and Surface Structure no longer featured and PF and LF remained as the only levels of representation.

To complicate the understanding of the development of Noam Chomsky's theories, the precise meanings of Deep Structure and Surface Structure have changed over time — by the 1970s, the two were normally referred to simply as D-Structure and S-Structure by Chomskyan linguists. In particular, the idea that the meaning of a sentence was determined by its Deep Structure (taken to its logical conclusions by the generative semanticists during the same period) was dropped for good by Chomskyan linguists when LF took over this role (previously, Chomsky and Ray Jackendoff had begun to argue that meaning was determined by both Deep and Surface Structure).^{[5][6]}

Innate linguistic knowledge

Terms such as "transformation" can give the impression that theories of transformational generative grammar are intended as a model for the processes through which the human mind constructs and understands sentences. Chomsky is clear that this is not in fact the case: a generative grammar models only the knowledge that underlies the human ability to speak and understand. One of the most important of Chomsky's ideas is that most of this knowledge is innate, with the result that a baby can have a large body of prior knowledge about the structure of language in general, and need only actually *learn* the idiosyncratic features of the language(s) it is exposed to. Chomsky was not the first person to suggest that all languages had certain fundamental things in common (he quotes philosophers writing several centuries ago who had the same basic idea), but he helped to make the innateness theory respectable after a period dominated by more behaviorist attitudes towards language. Perhaps more significantly, he made concrete and technically sophisticated proposals about the structure of language, and made important proposals regarding how the success of grammatical theories should be evaluated.

Grammatical theories

In the 1960s, Chomsky introduced two central ideas relevant to the construction and evaluation of grammatical theories. The first was the distinction between *competence* and *performance*. Chomsky noted the obvious fact that people, when speaking in the real world, often make linguistic errors (e.g., starting a sentence and then abandoning it midway through). He argued that these errors in linguistic performance were irrelevant to the study of linguistic competence (the knowledge that allows people to construct and understand grammatical sentences). Consequently, the linguist can study an idealised version of language, greatly simplifying linguistic analysis (see the "Grammaticality" section below). The second idea related directly to the evaluation of theories of grammar. Chomsky distinguished between grammars that achieve *descriptive adequacy* and those that go further and achieved *explanatory adequacy*. A descriptively adequate grammar for a particular language defines the (infinite) set of grammatical sentences in that language; that is, it describes the language in its entirety. A grammar that achieves explanatory adequacy has the additional property that it gives an insight into the underlying linguistic structures in the human mind; that is, it does not merely describe the grammar of a language, but makes predictions about how linguistic knowledge is mentally represented. For Chomsky, the nature of such mental representations is largely innate, so if a grammatical theory has explanatory adequacy it must be able to explain the various grammatical nuances of the languages of the world as relatively minor variations in the universal pattern of human language. Chomsky argued that, even though linguists were still a long way from constructing descriptively adequate grammars, progress in terms of descriptive adequacy will only come if linguists hold explanatory adequacy as their goal. In other words, real insight into the structure of individual languages can only be gained through comparative study of a wide range of languages, on the assumption that they are all cut from the same cloth.

"I-Language" and "E-Language"

In 1986, Chomsky proposed a distinction between *I-Language* and *E-Language*, similar but not identical to the competence/performance distinction.^[7] (*I-language*) refers to Internal language and is contrasted with External Language (or *E-language*). *I-Language* is taken to be the object of study in linguistic theory; it is the mentally represented linguistic knowledge that a native speaker of a language has, and is therefore a mental object — from this perspective, most of theoretical linguistics is a branch of psychology. *E-Language* encompasses all other notions of what a language is, for example that it is a body of knowledge or behavioural habits shared by a community. Thus, *E-Language* is not itself a coherent concept,^[8] and Chomsky argues that such notions of language are not useful in the study of innate linguistic knowledge, i.e., competence, even though they may seem sensible and intuitive, and useful in other areas of study. Competence, he argues, can only be studied if languages are treated as mental objects.

Grammaticality

Chomsky argued that the notions "grammatical" and "ungrammatical" could be defined in a meaningful and useful way. In contrast, an extreme behaviorist linguist would argue that language can only be studied through recordings or transcriptions of actual speech, the role of the linguist being to look for patterns in such observed speech, but not to hypothesize about why such patterns might occur, nor to label particular utterances as either "grammatical" or "ungrammatical." Although few linguists in the 1950s actually took such an extreme position, Chomsky was at an opposite extreme, defining grammaticality in an unusually mentalistic way (for the time).^[9] He argued that the intuition of a native speaker is enough to define the grammaticalness of a sentence; that is, if a particular string of English words elicits a double take, or feeling of wrongness in a native English speaker, and when various extraneous factors affecting intuitions are controlled for, it can be said that the string of words is ungrammatical. This, according to Chomsky, is entirely distinct from the question of whether a sentence is meaningful, or can be understood. It is possible for a sentence to be both grammatical and meaningless, as in Chomsky's famous example "colorless green ideas sleep furiously." But such sentences manifest a linguistic problem distinct from that posed by meaningful but ungrammatical (non)-sentences such as "man the bit sandwich the," the meaning of which is fairly clear, but no native speaker would accept as well formed.

The use of such intuitive judgments permitted generative syntacticians to base their research on a methodology in which studying language through a corpus of observed speech became downplayed, since the grammatical properties of constructed sentences were considered to be appropriate data to build a grammatical model on.

Minimalism

From the mid-1990s onwards, much research in transformational grammar has been inspired by Chomsky's *Minimalist Program*.^[10] The "Minimalist Program" aims at the further development of ideas involving *economy of derivation* and *economy of representation*, which had started to become significant in the early 1990s, but were still rather peripheral aspects of Transformational-generative grammar theory.

- Economy of derivation is a principle stating that movements (i.e., transformations) only occur in order to match *interpretable features* with *uninterpretable features*. An example of an interpretable feature is the plural inflection on regular English nouns, e.g., *dogs*. The word *dogs* can only be used to refer to several dogs, not a single dog, and so this inflection contributes to meaning, making it *interpretable*. English verbs are inflected according to the number of their subject (e.g., "Dogs bite" vs "A dog bites"), but in most sentences this inflection just duplicates the information about number that the subject noun already has, and it is therefore *uninterpretable*.
- Economy of representation is the principle that grammatical structures must exist for a purpose, i.e., the structure of a sentence should be no larger or more complex than required to satisfy constraints on grammaticality.

Both notions, as described here, are somewhat vague, and indeed the precise formulation of these principles is controversial.^{[11][12]} An additional aspect of minimalist thought is the idea that the derivation of syntactic structures should be *uniform*; that is, rules should not be stipulated as applying at arbitrary points in a derivation, but instead apply throughout derivations. Minimalist approaches to phrase structure have resulted in "Bare Phrase Structure," an attempt to eliminate X-bar theory. In 1998, Chomsky suggested that derivations proceed in phases. The distinction of Deep Structure vs. Surface Structure is not present in Minimalist theories of syntax, and the most recent phase-based theories also eliminate LF and PF as unitary levels of representation.

Mathematical representation

Returning to the more general mathematical notion of a grammar, an important feature of all transformational grammars is that they are more powerful than context-free grammars.^[13] This idea was formalized by Chomsky in the Chomsky hierarchy. Chomsky argued that it is impossible to describe the structure of natural languages using context-free grammars.^[14] His general position regarding the non-context-freeness of natural language has held up since then, although his specific examples regarding the inadequacy of CFGs in terms of their weak generative capacity were later disproven.^{[15][16]}

Transformations

The usual usage of the term 'transformation' in linguistics refers to a rule that takes an input typically called the Deep Structure (in the Standard Theory) or D-structure (in the extended standard theory or government and binding theory) and changes it in some restricted way to result in a Surface Structure (or S-structure). In TGG, Deep structures were generated by a set of phrase structure rules.

For example, a typical transformation in TG is the operation of subject-auxiliary inversion (SAI). This rule takes as its input a declarative sentence with an auxiliary: "John has eaten all the heirloom tomatoes." and transforms it into "Has John eaten all the heirloom tomatoes?" In their original formulation (Chomsky 1957), these rules were stated as rules that held over strings of either terminals or constituent symbols or both.

$$X \text{ NP AUX Y} \Rightarrow X \text{ AUX NP Y}$$

(where NP = Noun Phrase and AUX = Auxiliary)

In the 1970s, by the time of the Extended Standard Theory, following the work of Joseph Emonds on structure preservation, transformations came to be viewed as holding over trees. By the end of government and binding theory in the late 1980s, transformations are no longer structure changing operations at all; instead they add information to already existing trees by copying constituents.

The earliest conceptions of transformations were that they were construction-specific devices. For example, there was a transformation that turned active sentences into passive ones. A different transformation raised embedded subjects into main clause subject position in sentences such as "John seems to have gone"; and yet a third reordered arguments in the dative alternation. With the shift from rules to principles and constraints that was found in the 1970s, these construction-specific transformations morphed into general rules (all the examples just mentioned being instances of NP movement), which eventually changed into the single general rule of move alpha or Move.

Transformations actually come of two types: (i) the post-Deep structure kind mentioned above, which are string or structure changing, and (ii) Generalized Transformations (GTs). Generalized transformations were originally proposed in the earliest forms of generative grammar (e.g., Chomsky 1957). They take small structures, either atomic or generated by other rules, and combine them. For example, the generalized transformation of embedding would take the kernel "Dave said X" and the kernel "Dan likes smoking" and combine them into "Dave said Dan likes smoking." GTs are thus structure building rather than structure changing. In the Extended Standard Theory and government and binding theory, GTs were abandoned in favor of recursive phrase structure rules. However, they are still present in tree-adjoining grammar as the Substitution and Adjunction operations, and they have recently

re-emerged in mainstream generative grammar in Minimalism, as the operations Merge and Move.

In generative phonology, another form of transformation is the phonological rule, which describes a mapping between an underlying representation (the phoneme) and the surface form that is articulated during natural speech.^[17]

References

- [1] Chomsky, Noam (1995). *The Minimalist Program*. MIT Press.
- [2] Chomsky, Noam (1965). *Aspects of the Theory of Syntax*. MIT Press. ISBN 0-262-53007-4.
- [3] The Port-Royal Grammar of 1660 identified similar principles; Chomsky, Noam (1972). *Language and Mind*. Harcourt Brace Jovanovich. ISBN 0-15-147810-4.
- [4] Stockwell, Robert P.; Partee, Barbara Hall; Schacter, Paul (1973). *The major syntactic structures of English*. Holt, Rinehart and Winston. ISBN 978-0-03-088042-1.
- [5] Jackendoff, Ray (1974). *Semantic Interpretation in Generative Grammar*. MIT Press. ISBN 0-262-10013-4.
- [6] May, Robert C. (1977). *The Grammar of Quantification*. MIT Phd Dissertation. ISBN 0-8240-1392-1. (Supervised by Noam Chomsky, this dissertation introduced the idea of "logical form.")
- [7] Chomsky, Noam (1986). *Knowledge of Language*. New York:Praeger. ISBN 0-275-90025-8.
- [8] Chomsky, Noam (2001). "Derivation by Phase." In other words, in algebraic terms, *the I-Language is the actual function, whereas the E-Language is the extension of this function*. In Michael Kenstowicz (ed.) *Ken Hale: A Life in Language*. MIT Press. Pages 1-52. (See p. 49 fn. 2 for comment on E-Language.)
- [9] Newmeyer, Frederick J. (1986). *Linguistic Theory in America (Second Edition)*. Academic Press.
- [10] Chomsky, Noam (1995). *The Minimalist Program*. MIT Press. ISBN 0-262-53128-3.
- [11] Lappin, Shalom; Robert Levine and David Johnson (2000). "Topic ... Comment". *Natural Language & Linguistic Theory* **18** (3): 665–671. doi:10.1023/A:1006474128258.
- [12] Lappin, Shalom; Robert Levine and David Johnson (2001). "The Revolution Maximally Confused". *Natural Language & Linguistic Theory* **19** (4): 901–919. doi:10.1023/A:1013397516214.
- [13] Peters, Stanley; R. Ritchie (1973). "On the generative power of transformational grammars". *Information Sciences* **6**: 49–83. doi:10.1016/0020-0255(73)90027-3.
- [14] Chomsky, Noam (1956). "Three models for the description of language" (<http://www.chomsky.info/articles/195609--.pdf>). *IRE Transactions on Information Theory* **2** (3): 113–124. doi:10.1109/TIT.1956.1056813. .
- [15] Shieber, Stuart (1985). "Evidence against the context-freeness of natural language" (<http://www.eecs.harvard.edu/~shieber/Biblio/Papers/shieber85.pdf>). *Linguistics and Philosophy* **8** (3): 333–343. doi:10.1007/BF00630917. .
- [16] Pullum, Geoffrey K.; Gerald Gazdar (1982). "Natural languages and context-free languages". *Linguistics and Philosophy* **4** (4): 471–504. doi:10.1007/BF00360802.
- [17] Goldsmith, John A (1995). "Phonological Theory". In John A. Goldsmith. *The Handbook of Phonological Theory*. Blackwell Handbooks in Linguistics. Blackwell Publishers. p. 2. ISBN 1-4051-5768-2.

External links

- What is I-language? (<http://linguistics.concordia.ca/i-language/>) - Chapter 1 of I-language: An Introduction to Linguistics as Cognitive Science.
- The Syntax of Natural Language (<http://www.ling.upenn.edu/~beatrice/syntax-textbook/index.html>) – an online textbook on transformational grammar.

Principles and parameters

Principles and parameters is a framework within generative linguistics in which the syntax of a natural language is described in accordance with general *principles* (i.e. abstract rules or grammars) and specific *parameters* (i.e. markers, switches) that for particular languages are either turned *on* or *off*. For example, the position of heads in phrases is determined by a parameter. Whether a language is *head-initial* or *head-final* is regarded as a parameter which is either on or off for particular languages (i.e. English is *head-initial*, whereas Japanese is *head-final*). Principles and parameters was largely formulated by the linguists Noam Chomsky and Howard Lasnik. Many linguists have worked within this framework, and for a period of time it was considered the dominant form of mainstream generative linguistics.^[1]

Principles and Parameters as a grammar framework is also known as Government and Binding theory. That is, the two terms *Principles and Parameters* and *Government and Binding* refer to the same school in the generative tradition of phrase structure grammars (as opposed to dependency grammars).

Framework

The central idea of principles and parameters is that a person's syntactic knowledge can be modelled with two formal mechanisms:

- A finite set of fundamental **principles** that are common to all languages; e.g., that a sentence must always have a subject, even if it is not overtly pronounced.
- A finite set of **parameters** that determine syntactic variability amongst languages; e.g., a binary parameter that determines whether or not the subject of a sentence must be overtly pronounced (this example is sometimes referred to as the Pro-drop parameter).

Within this framework, the goal of linguistics is to identify all of the principles and parameters that are universal to human language (called: Universal Grammar). As such, any attempt to explain the syntax of a particular language using a principle or parameter is cross-examined with the evidence available in other languages. This leads to continual refinement of the theoretical machinery of generative linguistics in an attempt to account for as much syntactic variation in human language as possible.

Language acquisition

The Principles and Parameters approach is the postulated answer to Plato's Problem: how can children with different linguistic environments arrive at an accurate grammar that exhibits universal and non-obvious similarities, relatively rapidly, and with finite input. According to this framework, principles and parameters are part of a genetically innate universal grammar (UG) which all humans possess, barring any genetic disorders. As such, principles and parameters do not need to be learned by exposure to language. Rather, exposure to language merely triggers the parameters to adopt the correct setting. The problem is simplified considerably if children are innately equipped with mental apparatus that reduces and in a sense directs the search space amongst possible grammars. The P&P approach is an attempt to provide a precise and testable characterization of this innate endowment which consists of universal "Principles" and language-specific, binary "Parameters" that can be set in various ways. The interaction of the principles and the parameter settings produces all known languages while excluding non-natural languages.

Criticisms

Criticism of the P&P approach has come from a number of quarters, but with varying impact. These can be subdivided into three main groups.

- Theory internal critique
- The lack of consensus on a set of parameters
- Inter-paradigm critiques not specific to P&P

Perhaps the most influential criticisms of P&P have been theory internal. By its very nature, research published within the P&P paradigm often suggests reformulations and variations of the basic P&P premises. This is the norm for any developing field of enquiry. Notable debates emerged within P&P including (a) derivationalism vs representationalism (b) the locus of morphology (e.g. lexicalism vs derived morphology) and (c) the tension between a production model and a competence model amongst others. The development of head-driven phrase structure grammar (HPSG) and lexical functional grammar (LFG) reflect these debates: these are both strongly lexicalist and representational systems. Nevertheless, perhaps the most coherent and substantial critique of P&P is the Minimalist Program, Noam Chomsky's most recent proposal.^[2] This program of research utilizes conceptions of economy to enhance the search for universal principles and parameters. Linguists in this program assume that humans use as economic a system as possible in their innate syntactic knowledge. The Minimalist Program takes issue with the large number of independent postulations in P&P. and either (a) reduces them to more fundamental principles (e.g. Merge, Move, Agree), (b) derives them from 'reasonable' interface constraints on derivations (e.g. bottom-up Merge and requirement that no derivation be counter-cyclic derives Relativized Minimality effects) or (c) programmatically suggests that they be either derived from more basic principles or eliminated subject to future research (e.g. Binding Principles). Note that there is debate about whether the Minimalist Program is motivated by the empirical shortcomings of P&P^[3] or whether it is motivated by ideological concerns with 'elegance' etc.^[4] (see main article on the Minimalist Program).

Aside from this major move within the discipline, it seems that consensus has not been achieved over a list of universal parameters.^[5] Certainly, there is no publicly available list of these parameters and textbooks tend to cite the same ones: the interrelated verb-movement parameters (V-v, V-T, T-C), noun-movement parameters (N-D), subject-related parameters (pro-drop and EPP) and headedness parameters. This is not to say that the theory has not been fruitful (e.g. Holmberg and Platzak's comprehensive analysis of parametric variation in Scandinavian languages), or that the theory is not descriptively adequate, but rather that the accomplishments of this line of thinking have been less than anticipated in terms of explanatory adequacy. Notably lacking is the development of a systematic, predictive system of parameters, their properties and interactions along the lines of the periodic table in chemistry. This is not to say that such a system is impossible in principle, merely that it has not yet been developed. Generally, theorists have moved to regarding parameters as varying feature specifications on lexical items within languages and derivations rather than parameters which are globally defined.

For example, while formal linguistics takes the sentence to be the canonical unit of analysis, conversation analysis (CA) takes the turn at talk as canonical. Speakers in conversation often do not use complete sentences or even complete words to converse. Rather, discourse is composed of sequences of turns which are composed of turn constructional units (e.g. a word, phrase, clause, sentence).^[6] In CA, the form and meaning of an utterance is a product of situated activity- which is to say meaning is highly contextual (within a social, interactive context) and contingent upon how participants respond to each other regardless of grammatical completeness of an utterance.

Similarly, other discourse and corpus linguistic analyses have found recursion and other forms of grammatical complexity to be rather rare in spoken discourse (especially in preliterate societies) but common in written discourse suggesting that much of grammatical complexity may in fact be a product of literacy training.^{[7][8][9][10]}

Other critics point out that there is little if anything that can unequivocally be called universal across the world's languages.^[11] Discourse analyses have focused on the dynamic, dialogic, and social nature of language use in social situations.^{[12][13][14][15]} These critics argue that P&P and discourse analysis differ in the same way that chemistry

and cookery differ: one is the study of fundamental interactions at a micro-scale in a deterministic model that attempts to be scientific in the broad sense, the other is a more macro-scale, non-deterministic, non-scientific model focussing on use of chemicals in everyday situations in the real world. What these critiques have in common is the claim that the analysis of I-language does not carry over to E-language. From a Chomskyan perspective, this is a truism because the two objects of study are fundamentally different.

There is a tendency for inter-paradigm critiques to focus on a number of assumptions that are commonly associated with P&P, but which actually are common to Chomskyan generative linguistics as a whole. These include innateness, modularity, the poverty of the stimulus, language universals,^[16] binarity, etc. See for example, Connectionist, Functional, and Cognitivist critiques. As another example, the linguist Larry Trask argues that the ergative case system of the Basque language is not a simple binary parameter, and that different languages can have different levels of ergativity.^[17] Also, some have argued using evidence from historical linguistics that grammar is an emergent property of language use.^{[18][19][20]} Language evolution theorist, Terrence Deacon notes that it is problematic to consider language structure as innate - that is, as having been subject to the forces of natural selection, because languages change much too quickly for natural selection to act upon them. There are many more critiques. There is debate about the validity of these arguments, but since these are not specific to P&P they will not be dealt with here.

Examples

Examples of theorized principles are:

- Structure preservation principle
- Trace erasure principle
- Projection principle

Examples of theorized parameters are:

- Ergative case parameter
- Head directionality parameter
- Nominal mapping parameter
- Null subject parameter
- Polysynthesis parameter
- Pro-drop parameter
- Serial verb parameter
- Subject placement parameter
- Subject side parameter
- Topic prominent parameter
- Verb attraction parameter

Notes

- [1] Newmeyer, F.J. (2004). Against a parameter-setting approach to language variation. *Linguistic Variation Yearbook* 4:181-234.
- [2] Chomsky, Noam. (1995). *The Minimalist Program*. MIT Press, Cambridge MA
- [3] Holmberg, Anders (2002). Natural Language and Linguistic Theory 18: 837–842
- [4] Lappin, Shalom. Levine, Robert. Johnson, David. (2000). *Natural Language and Linguistic Theory* 18: 665–671.
- [5] Haspelmath, Martin. (2008). Parametric versus functional explanations of syntactic universals. Pp75--107 in *The limits of syntactic variation*. Biberauer, Theresa (Ed.) John Benjamins, Amsterdam.
- [6] Sacks, H., E. Schegloff, G. Jefferson (1974). "A Simplest Systematics for the Organization of Turn-taking for Conversation." *Language* 50(4): 696-735.
- [7] Chafe, W. L. (1985). Linguistic differences produced by differences between speaking and writing. *Literacy, language, and learning: The nature and consequences of reading and writing*. D. R. Olson, N. Torrence and A. Hildyard. Cambridge, Cambridge University Press
- [8] Croft, W. (2000). *Explaining Language Change*. New York, Longman.
- [9] Kalmar, I. (1985). *Are There Really No Primitive Languages? Literacy, Language, and Learning*. D. R. Olson, N. Torrence and A. Hildyard, Cambridge U Press.
- [10] Thompson, S. A. and P. J. Hopper (2001). *Transitivity, Clause Structure, and Argument Structure: Evidence from Conversation. Frequency and the Emergence of Linguistics Structure*. J. L. Bybee and P. J. Hopper. Amsterdam, Benjamins.
- [11] Tomasello, M. (2004). "What kind of evidence could refute the UG hypothesis? Commentary on Wunderlich." *Studies in Language* 28(3)
- [12] Goodwin, C. (1979). *The Interactive Construction of a Sentence in Natural Conversation*. *Everyday Language:Studies in Ethnomethodology*. G. Psathas. New York, Irvington Publishers: 97-121
- [13] Goodwin, C. (2003b). *The Semiotic Body in its Environment. Discourses of the Body*. J. Coupland and R. Gwyn. Oxford, Oxford University Press
- [14] Heritage, J. (1987). *Ethnomethodology. Social Theory Today*. A. Giddens and J. Turner. Cambridge, Polity Press.
- [15] Duranti, A., Ed. (2001). *Linguistic Anthropology: A Reader*, Blackwell Publishing.
- [16] Evans, N and Levinson, Stephen. (2009). "The Myth of language universals: language diversity and its importance for cognitive science. Behavioral and Brain Sciences. 32. pp429--492.
- [17] Larry Trask reviews *The Atoms of Language: The Mind's Hidden Rules of Grammar* by Mark C. Baker (<http://human-nature.com/nibbs/02/trask.html>)
- [18] Hopper, P. (1987). "Emergent Grammar." *Berkeley Linguistics Society* 13: 139-57.
- [19] Hopper, P. and E. Traugott (2003). *Grammaticalization*, Cambridge U Press.
- [20] Heine, B. and T. Kuteva (2007). *The Genesis of Gramma: A Reconstruction*, Oxford U Press.

References

- Baker, M. (2001). *The Atoms of Language: The Mind's Hidden Rules of Grammar*. Basic Bks.
- Chomsky, N. (1981). *Lectures on Government and Binding*. Mouton de Gruyter.
- Chomsky, N. and Lasnik, H. (1993) Principles and Parameters Theory, in *Syntax: An International Handbook of Contemporary Research*, Berlin: de Gruyter.
- Chomsky, N. (1995) *The Minimalist Program (Current Studies in Linguistics)*. MIT Press.
- Lightfoot, D. (1982). *The Language Lottery: Towards a Biology of Grammars*. MIT Press.

External links

- Three short talks on Principles & Parameters: by Luigi Rizzi, Mark Baker and Richard S. Kayne (http://www.youtube.com/watch?v=dXHSR_qpCEM&context=C3aa2591AD0EgsToPDskIFU52R9GmpbMiQ8Wx-hwPm)
- Why Chomskyan Linguistics is Mentalistic (<http://www.princeton.edu/~harman/Papers/Ling-Ment.html>)

Government and binding theory

Government and binding is a theory of syntax and a phrase structure grammar (as opposed to a dependency grammar) in the tradition of transformational grammar developed principally by Noam Chomsky in the 1980s.^{[1][2][3]} This theory is a radical revision of his earlier theories^{[4][5][6]} and was later revised in *The Minimalist Program* (1995)^[7] and several subsequent papers, the latest being *Three Factors in Language Design* (2005).^[8] Although there is a large literature on government and binding theory which is not written by Chomsky, Chomsky's papers have been foundational in setting the research agenda.

The name refers to two central subtheories of the theory: **government**, which is an abstract syntactic relation, and **binding**, which deals with the referents of pronouns, anaphors, and referential expressions. GB was the first theory to be based on the principles and parameters model of language, which also underlies the later developments of the Minimalist Program.

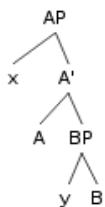
Government

The main application of the **government** relation concerns the assignment of case. Government is defined as follows:

A **governs** B if and only if

- A is a **governor** and
- A **m-commands** B and
- no **barrier** intervenes between A and B.

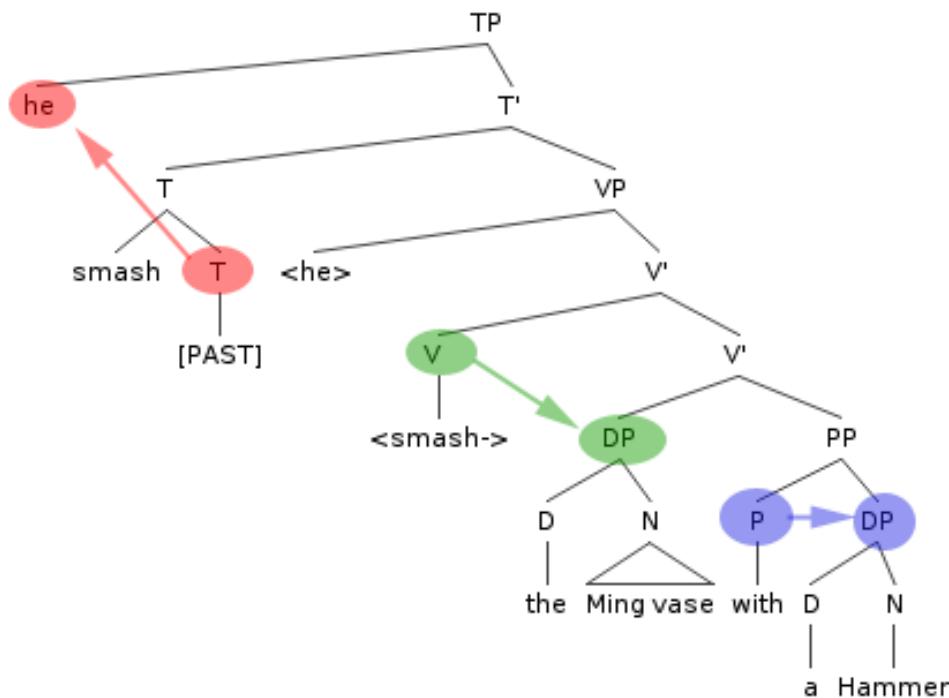
Governors are heads of the lexical categories (V, N, A, P) and tensed I (T). A m-commands B if A does not dominate B and B does not dominate A and the first maximal projection of A dominates B. The maximal projection of a head X is XP. This means that for example in a structure like the following, A m-commands B, but B does not m-command A:



In addition, **barrier** is defined as follows:^[9] A barrier is any node Z such that

- Z is a potential governor for B and
- Z c-commands B and
- Z does not c-command A

The government relation makes case assignment unambiguous. The tree diagram below illustrates how DPs are governed and assigned case by their governing heads:



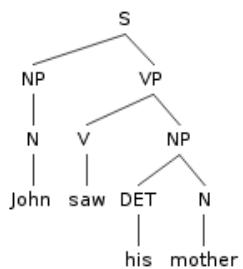
Another important application of the government relation constrains the occurrence and identity of traces as the Empty Category Principle requires them to be properly governed.

Binding

Binding can be defined as follows:

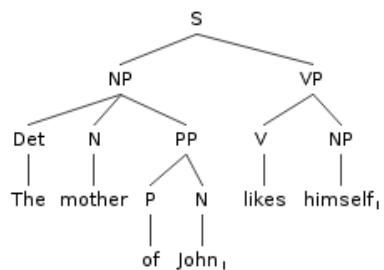
- An element α binds an element β if and only if α c-commands β , and α and β corefer.

Consider the sentence "John_i saw his_i mother." which is diagrammed below using simple phrase structure rules.



The NP "John" c-commands "his" because the first parent of the NP, S, contains "his". "John" and "his" are also coreferential (they refer to the same person), therefore "John" binds "his".

On the other hand, in the ungrammatical sentence "*The mother of John_i likes himself_i", "John" does not c-command "himself", so they have no binding relationship despite the fact that they corefer.



The importance of binding is shown in the grammaticality of the following sentences:

1. *John_i saw him_i.
2. John_i saw himself_i.
3. *Himself_i saw John_i.
4. *John_i saw John_i.

Binding is used, along with particular binding principles, to explain the ungrammaticality of those statements. The applicable rules are called Binding Principle A, Binding Principle B, and Binding Principle C.

- **Principle A:** an anaphor (reflexive or reciprocal, such as "each other") must be bound in its governing category (roughly, the clause).

Since "himself" is not c-commanded by "John" in sentence [3], Principle A is violated.

- **Principle B:** a pronoun must be free (i.e., not bound) within its governing category (roughly, the clause).

In sentence [1], "him" is bound by "John", violating Principle B.

- **Principle C:** an R-expression must be free (i.e., not bound). R-expressions (e.g. "the dog" or "John") are referential expressions: unlike pronouns and anaphora, they independently refer, i.e., pick out entities in the world.

In sentence [4], the first instance of "John" binds the second, violating Principle C.

Note that Principles A and B refer to "governing categories"--domains which limit the scope of binding. The definition of a governing category laid out in *Lectures on Government and Binding*^[1] is complex, but in most cases the governing category is essentially the minimal clause or complex NP.

Further reading

- Liliane Haegeman (1994). *Introduction to Government and Binding Theory* (Second Edition). Blackwell.

References

- [1] Chomsky, Noam (1981/1993). *Lectures on Government and Binding: The Pisa Lectures*. Mouton de Gruyter.
- [2] Chomsky, Noam (1982). *Some Concepts and Consequences of the Theory of Government and Binding*. Linguistic Inquiry Monograph 6. MIT Press.
- [3] Chomsky, Noam (1986). *Barriers*. Linguistic Inquiry Monograph 13. MIT Press.
- [4] Chomsky, Noam (1957/2002). *Syntactic Structures (Second Edition)*. Mouton de Gruyter.
- [5] Chomsky, Noam (1965). *Aspects of the Theory of Syntax*. MIT Press.
- [6] Chomsky, Noam (1970). Remarks on Nominalization. In *Studies on Semantics in Generative Grammar* (1972). The Hague: Mouton. Pages 11-61.
- [7] Chomsky, Noam (1995). *The Minimalist Program*. MIT Press.
- [8] Chomsky, Noam (2005). "Three Factors in Language Design" (http://muse.jhu.edu/journals/linguistic_inquiry/v036/36.1chomsky.pdf). *Linguistic Inquiry* 36 (36): 1–22. doi:10.1162/0024389052993655. .
- [9] see "Minimality" in Haegeman 1994:163f.

External links

- A step-by-step introduction to the Government and Binding theory of syntax (<http://www.sil.org/mexico/ling/E002-IntroGB.pdf>)
- Introduction to GB (<http://www.ifi.unizh.ch/CL/gschneid/dreitaegig.pdf>)

Minimalist program

In linguistics, the **Minimalist Program** (MP) is a major line of inquiry that has been developing inside generative grammar since the early 1990s, starting with a 1993 paper by Noam Chomsky.^[1]

Chomsky presents MP as a program, not as a theory, following Imre Lakatos's distinction.^[2] The MP seeks to be a mode of inquiry characterized by the flexibility of the multiple directions that its minimalism enables. Ultimately, the MP provides a conceptual framework used to guide the development of grammatical theory. For Chomsky, there are minimalist questions, but the answers can be framed in any theory. Of all these questions, the one that plays the most crucial role is this: why language has the properties it has.^[3] The MP lays out a very specific view of the basis of syntactic grammar that, when compared to other formalisms, is often taken to look very much like a theory.

Theoretical goals

Perfection

The MP appeals to the idea that the language ability in humans shows signs of being incorporated under an optimal design with exquisite organization, which seems to suggest that the inner workings conform to a very simple computational law or a particular mental organ. In other words, the MP works on the assumption that Universal Grammar constitutes a perfect design in the sense that it contains only what is necessary to meet our conceptual and physical (phonological) needs.^[4]

From a theoretical standpoint, and in the context of generative grammar, the MP draws on the minimalist approach of the Principles and Parameters program, considered to be the ultimate standard theoretical model that generative linguistics has developed since the 1980s. What this approach suggests is the existence of a fixed set of principles valid for all languages, which, when combined with settings for a finite set of binary switches (parameters), may describe the specific properties that characterize the language system a child eventually comes to attain.^[5]

The MP aims to get to know how much of the Principles and Parameters model can be taken as a result of this hypothetical optimal and computationally efficient design of the human language faculty. In turn, more developed versions of the Principles and Parameters approach provide technical principles from which the MP can be seen to follow.^[6]

Economy

The MP aims at the further development of ideas involving *economy of derivation* and *economy of representation*, which had started to become significant in the early 1990s, but were still peripheral aspects of transformational grammar.^[7]

- Economy of derivation is a principle stating that movements (i.e. transformations) only occur in order to match *interpretable features* with *uninterpretable features*. An example of an interpretable feature is the plural inflection on regular English nouns, e.g. *dogs*. The word *dogs* can only be used to refer to several dogs, not a single dog, and so this inflection contributes to meaning, making it *interpretable*. English verbs are inflected according to the number of their subject (e.g. "Dogs bite" vs "A dog bites"), but this information is only interpretable once a relationship is formed between the subject and the verb, so movement of the subject is required.
- Economy of representation is the principle that grammatical structures must exist for a purpose, i.e. the structure of a sentence should be no larger or more complex than required to satisfy constraints on grammaticality, which are equivalent to constraints on the mapping between the conceptual/intensional and sensori-motor interfaces in the optimal system that minimalism seeks to explore.

Technical innovations

The exploration of minimalist questions has led to several radical changes in the technical apparatus of transformational generative grammatical theory. Some of the most important are:^[8]

- The generalization of X-bar theory into Bare Phrase Structure (see below).
- The simplification of representational levels in the grammatical model, eliminating the distinction between deep structure and surface structure in favor of more explicitly derivational approach.
- The elimination of the notion of government.
- The inclusion of a single point of interaction between syntax and the interfaces (conceptual/intensional and sensori-motor), commonly called the point of *Spell-Out*.
- The idea that syntactic derivations proceed by clearly delineated stages called *phases* (see below).

Bare Phrase Structure

A major development of MP inquiry is Bare Phrase Structure (BPS), a theory of phrase structure (sentence building prior to movement) developed by Noam Chomsky.^[9] Interestingly, the introduction of BPS has moved the Chomskyan tradition toward the dependency grammar tradition, which operates with significantly less structure than most phrase structure grammars.^[10]

This theory contrasts with X-bar theory, which preceded it, in four important ways:

1. BPS is explicitly derivational. That is, it is built from the bottom up, bit by bit. In contrast, X-Bar Theory is representational—a structure for a given construction is built in one fell swoop, and lexical items are inserted into the structure.
2. BPS does not have a preconceived phrasal structure, while in X-Bar Theory, every phrase has a specifier, a head, and a complement.
3. BPS permits only binary branching, while X-Bar Theory permits both binary and unary branching.
4. BPS does not distinguish between a "head" and a "terminal", while some versions of X-Bar Theory require such a distinction.

BPS incorporates two basic operations: Merge and Move. Although there is active debate on exactly how Move should be formulated, the differences between the current proposals are relatively minute. The following description follows Chomsky's original proposal.

Merge is a function that takes two objects (say α and β) and merges them into an unordered set with a label (either α or β , in this case α). The label identifies the properties of the phrase.

$$\text{Merge } (\alpha, \beta) \rightarrow \{\alpha, \{\alpha, \beta\}\}$$

For example, Merge can operate on the lexical items 'drink' and 'water' to give 'drink water'. Note that the phrase 'drink water' behaves more like the verb 'drink' than like the noun 'water'. That is, wherever we can put the verb 'drink' we can usually put the phrase 'drink water':

I like to _____ (drink)/(drink water).

(Drinking/Drinking water) _____ is fun.

Furthermore, we typically can't put the phrase 'drink water' in places where we can put the noun 'water':

We can say "There's some *water* on the table", but not "There's some *drink water* on the table".

So, we identify the phrase with a label. In the case of 'drink water', the label is 'drink' since the phrase acts as a verb. For simplicity, we call this phrase a verb phrase or VP. Now if we were to Merge 'cold' and 'water' to get 'cold water', then we would have a noun phrase or NP with the label 'water'. The reader can verify that the phrase 'cold water' can appear in the same environments as the noun 'water' in the three test sentences above. So, for 'drink water' we have the following:

$$\text{Merge } (\text{drink}, \text{water}) \rightarrow \{\text{drink}, \{\text{drink}, \text{water}\}\}$$

We can represent this in a typical syntax tree as follows:

```
drink
 / \
drink water
```

or, with more technical terms, as:

```
VP
 / \
drink water
```

Merge can also operate on structures already built. If it couldn't, then such a system would predict only two-word utterances to be grammatical. Say we Merge a new head with a previously formed object (a phrase).

$\text{Merge}(\gamma, \{\alpha, \{\alpha, \beta\}\}) \rightarrow \{\gamma, \{\gamma, \{\alpha, \{\alpha, \beta\}\}\}\}$

Here, γ is the label, so we say that γ 'projects' from the label of the head. This corresponds to the following tree structure:

```
Y
 / \
Y   α
 / \
α   β
```

Note crucially that Merge operates blindly, projecting labels in all possible combinations. The subcategorization features of the head then license certain label projections and eliminate all derivations with alternate projections.

Phases

A **phase** is a syntactic domain first hypothesized by Noam Chomsky in 1998.^[11] A simple sentence is often decomposed into two phases, CP and vP (see X-bar theory). Movement of a constituent out of a phase is (in the general case) only permitted if the constituent has first moved to the left edge of the phase. This condition is described in the *Phase Impenetrability Condition*, which has been variously formulated within the literature. In its original conception, only the vP in transitive and unergative verbs constitute phases. The vP in passives and unaccusative verbs (if even present) are not phases. This topic is, however, currently under debate in the literature.^[12]

Criticisms

In the late 1990s, David E. Johnson and Shalom Lappin published the first detailed critiques of Chomsky's minimalist program.^[13] This technical work was followed by a lively debate with proponents of minimalism on the scientific status of the program.^{[14][15][16]} The original article provoked several replies^{[17][18][19][20][21]} and two further rounds of replies and counter-replies in subsequent issues of the same journal.

Lappin et al. argue that the Minimalist Program is a radical departure from earlier Chomskyan linguistic practice that is not motivated by any new empirical discoveries, but rather by a general appeal to "perfection", which is both empirically unmotivated and so vague as to be unfalsifiable. They compare the adoption of this paradigm by linguistic researchers to other historical paradigm shifts in natural sciences and conclude that the adoption of the Minimalist Program has been an "unscientific revolution", driven primarily by Chomsky's authority in linguistics. The several replies to the article in *Natural Language and Linguistic Theory* Volume 18 number 4 (2000) make a number of different defenses of the Minimalist Program. Some claim that it is not in fact revolutionary or not in fact widely adopted, while others agree with Levine and Johnson on these points, but defend the vagueness of its

formulation as not problematic in light of its status as a research program rather than a theory (see above).

Further Readings on the Minimalist Program

Much research has been devoted to the study of the consequences that arise when minimalist questions are formulated. This list is not exhaustive.

Works by Noam Chomsky

- Chomsky, Noam. 1993. "A minimalist program for linguistic theory". In Hale, Kenneth L. and S. Jay Keyser, eds. *The view from Building 20: Essays in linguistics in honor of Sylvain Bromberger*. Cambridge, MA: MIT Press. 1–52
- Chomsky, Noam. 1995. *The Minimalist Program*. Cambridge, Mass.: The MIT Press.
- Chomsky, Noam. 2000. Minimalist inquiries: the framework. In *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*, eds. Roger Martin, David Michaels and Juan Uriagereka, 89–155. Cambridge, Mass: MIT Press.
- Chomsky, Noam. 2000. New horizons in the study of language and mind. Cambridge, UK ; New York: Cambridge University Press.
- Chomsky, Noam. 2001. Derivation by Phase. In *Ken Hale: A Life in Language*, ed. Michael Kenstowicz, 1–52. Cambridge, Mass.: MIT Press.
- Chomsky, Noam. 2004. Beyond Explanatory Adequacy. In *Structures and Beyond. The Cartography of Syntactic Structures*, ed. Adriana Belletti, 104–131. Oxford: Oxford University Press.
- Chomsky, Noam. 2005. Three Factors in Language Design. *Linguistic Inquiry* 36: 1–22.
- Chomsky, Noam. 2007. Approaching UG From Below. In *Interfaces + Recursion = Language?*, eds. Uli Sauerland and Hans Martin Gärtner, 1–29. New York: Mouton de Gruyter.
- Chomsky, Noam. 2008. On Phases. In *Foundational Issues in Linguistic Theory. Essays in Honor of Jean-Roger Vergnaud*, eds. Robert Freidin, Carlos Peregrín Otero and María Luisa Zubizarreta, 133–166. Cambridge, MA: MIT Press.

Linguistic Textbooks on Minimalism

- Adger, David. 2003. *Core Syntax. A Minimalist Approach*. Oxford: Oxford University Press
- Boeckx, Cedric. 2006. *Linguistic Minimalism. Origins, Concepts, Methods and Aims*. Oxford: Oxford University Press.
- Bošković, Željko and Howard Lasnik (eds). 2006. *Minimalist Syntax: The Essential Readings*. Malden, MA: Blackwell.
- Cook, Vivian J. and Newson, Mark. 2007. *Chomsky's Universal Grammar: An Introduction*. Third Edition. Malden, MA: Blackwell.
- Hornstein, Norbert, Jairo Nunes and Kleanthes K. Grohmann. 2005. *Understanding Minimalism*. Cambridge: Cambridge University Press
- Lasnik, Howard, Juan Uriagereka, Cedric Boeckx. 2005. *A Course in Minimalist Syntax*. Malden, MA: Blackwell
- Radford, Andrew. 2004. *Minimalist Syntax: Exploring the Structure of English*. Cambridge, UK:Cambridge University Press.
- Uriagereka, Juan. 1998. *Rhyme and Reason. An Introduction to Minimalist Syntax*. Cambridge, Mass: MIT Press.
- Webelhuth, Gert (ed.). 1995. *Government and Binding Theory and the Minimalist Program: Principles and Parameters in Syntactic Theory*. Wiley-Blackwell

Works on the main theoretical notions and their applications

- Boeckx, Cedric (ed). 2006. *Minimalist Essays*. Amsterdam: John Benjamins.
- Bošković, Željko. 1997. *The Syntax of Nonfinite Complementation. An Economy Approach*. Cambridge, Mass: MIT Press.
- Brody, Michael. 1995. *Lexico-Logical Form: a Radically Minimalist Theory*. Cambridge, MA: MIT Press.
- Collins, Chris. 1997. *Local Economy*. Cambridge, MA: MIT Press.
- Epstein, Samuel David, and Hornstein, Norbert (eds). 1999. *Working Minimalism*. Cambridge, MA: MIT Press.
- Epstein, Samuel David, and Seely, T. Daniel (eds). 2002. *Derivation and Explanation in the Minimalist Program*. Malden, MA: Blackwell.
- Fox, Danny. 1999. *Economy and Semantic Interpretation*. Cambridge, MA: MIT Press.
- Martin, Roger, David Michaels and Juan Uriagereka (eds). 2000. *Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik*. Cambridge, MA: MIT Press.
- Pesetsky, David. 2001. *Phrasal Movement and its Kin*. Cambridge, MA: MIT Press.
- Richards, Norvin. 2001. *Movement in Language*. Oxford: Oxford University Press.

References

- [1] Chomsky, Noam. 1993. *A minimalist program for linguistic theory*. MIT occasional papers in linguistics no. 1. Cambridge, MA: Distributed by MIT Working Papers in Linguistics.
- [2] For a thorough discussion of this distinction in the context of linguistics, see Boeckx, Cedric. 2006. *Linguistic Minimalism: Origins, Concepts, Methods, and Aims*. Oxford: Oxford University Press.
- [3] Boeckx, Cedric *Linguistic Minimalism. Origins, Concepts, Methods and Aims*, pp. 84 and 115.
- [4] Boeckx, Cedric. 2006. *Linguistic Minimalism. Origins, Concepts, Methods and Aims*. Oxford: Oxford University Press.
- [5] There are many introductions to Principle and Parameters. Two that align PP in such a way that make the transition to MP smooth are Carnie, Andrew. 2006. *Syntax: A Generative Introduction*, 2nd Edition. Malden, MA: Blackwell, and Cook, Vivian J. and Newson, Mark. 2007. *Chomsky's Universal Grammar: An Introduction*. Third Edition. Malden, MA: Blackwell.
- [6] For a detailed introductory discussion between the transition of the technicalities from PP to MP see, among others, Gert Webelhuth. 1995. *Government and Binding Theory and the Minimalist Program: Principles and Parameters in Syntactic Theory*. Wiley-Blackwell; Uriagereka, Juan. 1998. *Rhyme and Reason. An Introduction to Minimalist Syntax*. Cambridge, Mass: MIT Press; Hornstein, Norbert, Jairo Nunes and Kleantzes K. Grohmann. 2005. *Understanding Minimalism*. Cambridge: Cambridge University Press; and Boeckx, Cedric. 2006. *Linguistic Minimalism. Origins, Concepts, Methods and Aims*. Oxford: Oxford University Press.
- [7] For a full description of the checking mechanism see Adger, David. 2003. *Core Syntax. A Minimalist Approach*. Oxford: Oxford University Press; and also Carnie, Andrew. 2006. *Syntax: A Generative Introduction*, 2nd Edition. Blackwell Publishers
- [8] For some conceptual and empirical advantages of the MP over the traditional view see: Bošković, Željko. 1994. D-Structure, Θ-Criterion, and Movement into Θ-Positions. *Linguistic Analysis* 24: 247–286, and for more detailed discussions Bošković, Željko and Howard Lasnik (eds). 2006. *Minimalist Syntax: The Essential Readings*. Malden, MA: Blackwell.
- [9] See Chomsky, Noam. 1995. Bare Phrase Structure. In *Evolution and Revolution in Linguistic Theory. Essays in honor of Carlos Otero*, eds. Hector Campos and Paula Kempchinsky, 51–109.
- [10] Osborne, Timothy, Michael Putnam, and Thomas Gross 2011. Bare phrase structure, label-less structures, and specifier-less syntax: Is Minimalism becoming a dependency grammar? *The Linguistic Review* 28: 315–364
- [11] Chomsky, Noam (1998). "Minimalist Inquiries: The Framework" MIT Occasional Papers in Linguistics 15. Republished in 2000 in R. Martin, D. Michaels, & J. Uriagereka (eds.). *Step By Step: Essays In Syntax in Honor of Howard Lasnik*. 89–155. MIT Press.
- [12] See, among others, Legate, Julie Anne. 2003. Some Interface Properties of the phrase. *Linguistic Inquiry* 34: 506–516 and Chomsky, Noam. 2008. On Phases. In *Foundational Issues in Linguistic Theory. Essays in Honor of Jean-Roger Vergnaud*. eds. Robert Freidin, Carlos Peregrín Otero and María Luisa Zubizarreta, 133–166. Cambridge, MA: MIT Press
- [13] Johnson, David E. and Shalom Lappin (1997), "A Critique of the Minimalist Program" in *Linguistics and Philosophy* 20, 273–333, and Johnson, David E. and Shalom Lappin (1999). *Local Constraints vs Economy*. Stanford: CSLI
- [14] *Lappin, Shalom, Robert Levine and David E. Johnson (2000a). "The Structure of Unscientific Revolutions." *Natural Language and Linguistic Theory* 18, 665–771
- [15] Lappin, Shalom, Robert Levine and David E. Johnson (2000b). "The Revolution Confused: A Reply to our Critics." *Natural Language and Linguistic Theory* 18, 873–890
- [16] Lappin, Shalom, Robert Levine and David E. Johnson (2001). "The Revolution Maximally Confused." *Natural Language and Linguistic Theory* 19, 901–919
- [17] Holmberg, Anders (2000). "Am I Unscientific? A Reply to Lappin, Levine, and Johnson". *Natural Language & Linguistic Theory* 18: 837–842. doi:10.1023/A:1006425604798.

- [18] Reuland, Eric (2000). "Revolution, Discovery, and an Elementary Principle of Logic". *Natural Language & Linguistic Theory* **18**: 843–848. doi:10.1023/A:1006404305706.
- [19] Roberts, Ian (2000). "Caricaturing Dissent". *Natural Language & Linguistic Theory* **18**: 849–857. doi:10.1023/A:1006408422545.
- [20] >Piatelli-Palmarini, Massimo (2000). "The Metric of Open-Mindedness". *Natural Language & Linguistic Theory* **18**: 859–862. doi:10.1023/A:1006460406615.
- [21] >Uriagereka, Juan (2000). "On the Emptiness of 'Design' Polemics". *Natural Language & Linguistic Theory* **18**: 863–871. doi:10.1023/A:1006412507524.

Relational grammar

In linguistics, **Relational Grammar** (RG) is a syntactic theory which argues that primitive grammatical relations provide the ideal means to state syntactic rules in universal terms. Relational grammar began as an alternative to transformational grammar.

Term Relations

In Relational Grammar, constituents that serve as the arguments to predicates are numbered. This numbering system corresponds loosely to the notions of subject, direct object and indirect object. The numbering scheme is subject → (1), direct object → (2) and indirect object → (3). A schematic representation of a clause in this formalism might look like:

1	P	3	2
John	gave	Mary	a kiss

Other Features

- Strata
- Chomage
- Predicate valence

Universals

One of the components of RG theory is a set of linguistic universals stated in terms of the numbered roles presented above. Such a universal is the **Stratal Uniqueness Law** which states that there can be "at most one 1, 2, and 3 per stratum.

Further reading

- Blake, Barry J. (1990). *Relational grammar*. London: Routledge.
- Johnson, David; Adam Meyers and Lawrence S. Moss (1993). "A Unification-Based Parser for Relational Grammar" [1]. *Meeting of the Association for Computational Linguistics*. pp. 97–104.
- Perlmutter, David M. (1980). Relational grammar. In E. A. Moravcsik & J. R. Wirth (Eds.), *Syntax and semantics: Current approaches to syntax* (Vol. 13, pp. 195–229). New York: Academic Press.
- Perlmutter, David M. (Ed.). (1983). *Studies in relational grammar 1*. Chicago: Chicago University Press.
- Perlmutter, David M.; & Rosen, Carol G. (Eds.). (1984). *Studies in relational grammar 2*. Chicago: Chicago University Press.
- Postal, Paul M.; & Joseph, Brian D. (Eds.). (1990). *Studies in relational grammar 3*. Chicago: Chicago University Press.

Sources

- Johnson, David E. (1974/1979). *Toward a Theory of Relationally-based Grammar*. Outstanding Dissertations in Linguistics Series, ed. Jorge Hankamer. NY: Garland Publishing, Inc. ISBN 978-0-8240-9682-3
- Johnson, David E. and Paul M. Postal (1980). *Arc Pair Grammar*. Princeton: PUP. ISBN 0-691-08270-7
- Newmeyer, Frederick (1980). *Linguistics in America*. New York: Academic Press. ISBN 978-90-277-1290-5
- Postal, Paul M. (1974). *On Raising - An Inquiry into One Rule of English Grammar and Its Theoretical Implications*. Mass.: MIT Press. ISBN 978-0-262-66041-9

External links

- RG SIL bibliography [2]

References

- [1] <http://citeseer.ist.psu.edu/johnson93unificationbased.html>
[2] http://www.ethnologue.com/show_subject.asp?code=rlg

Lexical functional grammar

Lexical functional grammar (LFG) is a grammar framework in theoretical linguistics, a variety of generative grammar. It is a type of phrase structure grammar, as opposed to a dependency grammar. The development of the theory was initiated by Joan Bresnan and Ronald Kaplan in the 1970s, in reaction to the direction research in the area of transformational grammar had begun to take. It mainly focuses on syntax, including its relation with morphology and semantics. There has been little LFG work on phonology (although ideas from optimality theory have recently been popular in LFG research).

LFG views language as being made up of multiple dimensions of structure. Each of these dimensions is represented as a distinct structure with its own rules, concepts, and form. The primary structures that have figured in LFG research are:

- the representation of grammatical functions (**f-structure**). See feature structure.
- the structure of syntactic constituents (**c-structure**). See phrase structure rules, ID/LP grammar.

For example, in the sentence *The old woman eats the falafel*, the c-structure analysis is that this is a sentence which is made up of two pieces, a noun phrase (NP) and a verb phrase (VP). The VP is itself made up of two pieces, a verb (V) and another NP. The NPs are also analyzed into their parts. Finally, the bottom of the structure is composed of the words out of which the sentence is constructed. The f-structure analysis, on the other hand, treats the sentence as being composed of attributes, which include features such as number and tense or functional units such as subject, predicate, or object.

There are other structures which are hypothesized in LFG work:

- argument structure (a-structure), a level which represents the number of arguments for a predicate and some aspects of the lexical semantics of these arguments. See theta-role.
- semantic structure (s-structure), a level which represents the meaning of phrases and sentences. See Glue Semantics.
- information structure (i-structure)
- morphological structure (m-structure)
- phonological structure (p-structure)

The various structures can be said to be **mutually constraining**.

The LFG conception of language differs from Chomskyan theories, which have always involved separate levels of constituent structure representation being mapped onto each other sequentially, via transformations. The LFG approach has had particular success with nonconfigurational languages, languages in which the relation between structure and function is less direct than it is in languages like English; for this reason LFG's adherents consider it a more plausible universal model of language.

Another feature of LFG is that grammatical-function changing operations like passivization are said to be lexical. This means that the active-passive relation, for example, is a relation between two types of verb rather than two trees. Active and passive verbs are both listed in the lexicon, and involve alternative mapping of the participants to grammatical functions.

Through the positing of productive processes in the lexicon and the separation of structure and function, LFG is able to account for syntactic patterns without the use of transformations defined over syntactic structure. For example, in a sentence like *What did you see?*, where *what* is understood as the object of *see*, transformational grammar puts *what* after *see* (the usual position for objects) in "deep structure", and then moves it. LFG analyzes *what* as having two functions: question-focus and object. It occupies the position associated in English with the question-focus function, and the constraints of the language allow it to take on the object function as well.

A central goal in LFG research is to create a model of grammar with a depth which appeals to linguists while at the same time being efficiently parsable and having the rigidity of formalism which computational linguists require. Because of this, LFG has been used as the theoretical basis of various machine translation tools, such as AppTek's TranSphere, and the Julietta Research Group's Lekta.

References

- Bresnan, Joan (2001). *Lexical Functional Syntax*. Blackwell. ISBN 0-631-20973-5
- Dalrymple, Mary (2001). *Lexical Functional Grammar*. No. 42 in *Syntax and Semantics Series*. New York: Academic Press. ISBN 0-12-613534-7
- Falk, Yehuda N. (2001). *Lexical-Functional Grammar: An Introduction to Parallel Constraint-Based Syntax*. CSLI. ISBN 1-57586-341-3

External links

- Lexical Functional Grammar Home Page ^[1]
- Proceedings of the annual LFG conference ^[2]
- Julietta Research Group, Lekta Manual ^[3]

References

- [1] <http://www.essex.ac.uk/linguistics/external/LFG/>
[2] <http://cslipublications.stanford.edu/LFG/>
[3] http://grupo.us.es/julietta/lekta/lekta_manual.html

Generalized phrase structure grammar

Generalized phrase structure grammar (GPSG) is a framework for describing the syntax and semantics of natural languages. It is a type of phrase structure grammar, as opposed to a dependency grammar. GPSG was initially developed in the late 1970s by Gerald Gazdar. Other contributors include Ewan Klein, Ivan Sag, and Geoffrey Pullum. Their book *Generalized Phrase Structure Grammar*, published in 1985, is the main monograph on GPSG, especially as it applies to English syntax.

One of the chief goals of GPSG is to show that the syntax of natural languages can be described by context-free grammars (written as ID/LP grammars), with some suitable conventions intended to make writing such grammars easier for syntacticians. Among these conventions are a sophisticated feature structure system and so-called "meta-rules", which are rules generating the productions of a context-free grammar. GPSG further augments syntactic descriptions with semantic annotations that can be used to compute the compositional meaning of a sentence from its syntactic derivation tree. However, it has been argued (for example by Robert Berwick) that these extensions require parsing algorithms of a higher order of computational complexity than those used for basic CFGs.

Gerald Gazdar, and many other syntacticians, have since argued that natural languages cannot in fact be adequately described by CFGs [1].

GPSG is in part a reaction against transformational theories of syntax. In fact, the notational extensions to context-free grammars developed in GPSG are claimed to make transformations redundant. Most of the syntactic innovations of GPSG were subsequently incorporated into head-driven phrase structure grammar.

References

- Gazdar, Gerald; Ewan H. Klein, Geoffrey K. Pullum, Ivan A. Sag (1985). *Generalized Phrase Structure Grammar*. Oxford: Blackwell, and Cambridge, MA: Harvard University Press. ISBN 0-674-34455-3.

External links

- Gerald Gazdar's homepage ^[2], containing links to publications on GPSG.
- Generalised Phrase Structure Grammar page at University of Sussex ^[3]

References

- [1] <http://www.informatics.susx.ac.uk/research/nlp/gazdar/briscoe/gpsg.html>
- [2] <http://www.cogs.susx.ac.uk/lab/nlp/gazdar/gazdar.html>
- [3] <http://www.informatics.sussex.ac.uk/research/groups/nlp/gazdar/briscoe/gpsg.html>

Head-driven phrase structure grammar

Head-driven phrase structure grammar (HPSG) is a highly lexicalized, non-derivational generative grammar theory developed by Carl Pollard and Ivan Sag.^{[1][2]} It is a type of phrase structure grammar, as opposed to a dependency grammar, and it is the immediate successor to generalized phrase structure grammar. HPSG draws from other fields such as computer science (data type theory and knowledge representation) and uses Ferdinand de Saussure's notion of the sign. It uses a uniform formalism and is organized in a modular way which makes it attractive for natural language processing.

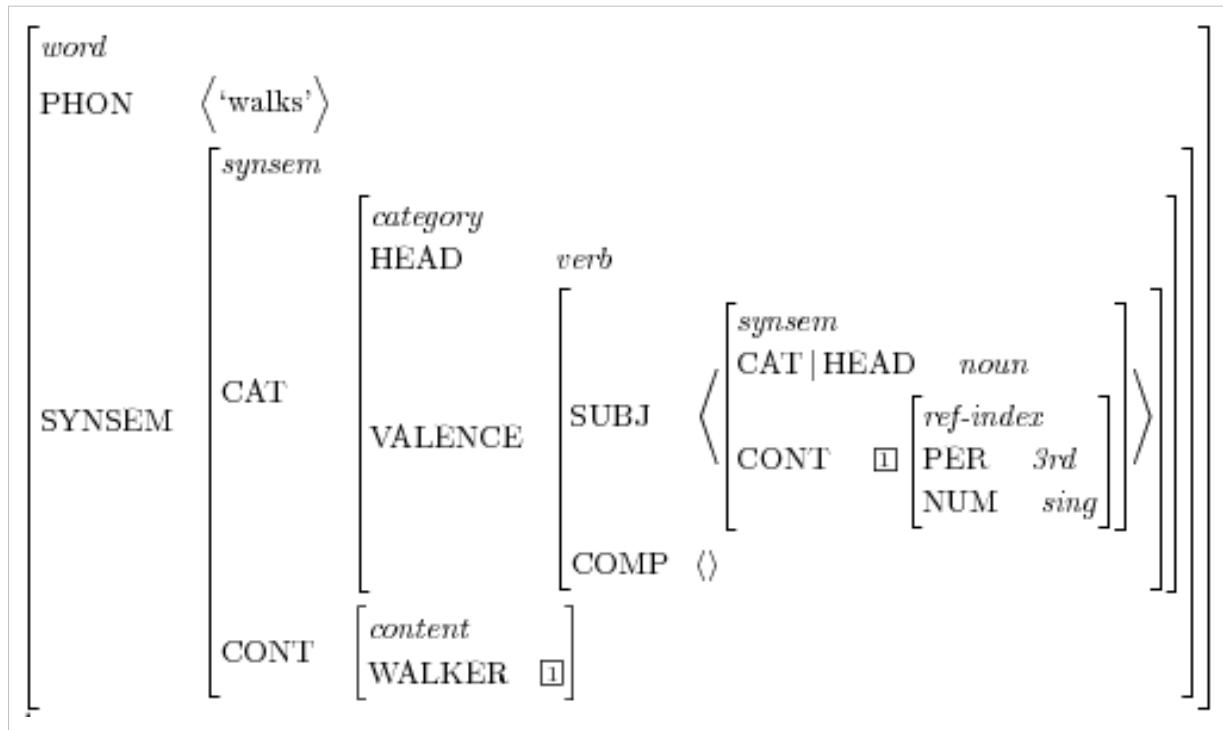
An HPSG grammar includes principles and grammar rules and lexicon entries which are normally not considered to belong to a grammar. The formalism is based on lexicalism. This means that the lexicon is more than just a list of entries; it is in itself richly structured. Individual entries are marked with types. Types form a hierarchy. Early versions of the grammar were very lexicalized with few grammatical rules (schema). More recent research has tended to add more and richer rules, becoming more like Construction Grammar.^[3]

The basic type HPSG deals with is the sign. Words and phrases are two different subtypes of sign. A word has two features: *[PHON]* (the sound, the phonetic form) and *[SYNSEM]* (the syntactic and semantic information), both of which are split into subfeatures. Signs and rules are formalized as typed feature structures.

A Sample Grammar

HPSG generates strings by combining signs, which are defined by their location within a type hierarchy and by their internal feature structure, represented by attribute value matrices (AVMs).^{[4][5]} Features take types or lists of types as their values, and these values may in turn have their own feature structure. Grammatical rules are largely expressed through the constraints signs place on one another. A sign's feature structure describes its phonological, syntactic, and semantic properties. In common notation, AVMs are written with features in upper case and types in italicized lower case. Numbered indices in an AVM represent token identical values.

In the simplified AVM for the word "walks" below, the verb's categorical information is divided into features that describe it (HEAD) and features that describe its arguments (VALENCE).



"Walks" is a sign of type *word* with a head of type *verb*. As an intransitive verb, "walks" has no complement but requires a subject that is a third person singular noun. The semantic value of the subject (CONTENT) is co-indexed with the verb's only argument (the individual doing the walking). The following AVM for "she" represents a sign with a SYNSEM value that could fulfill those requirements.

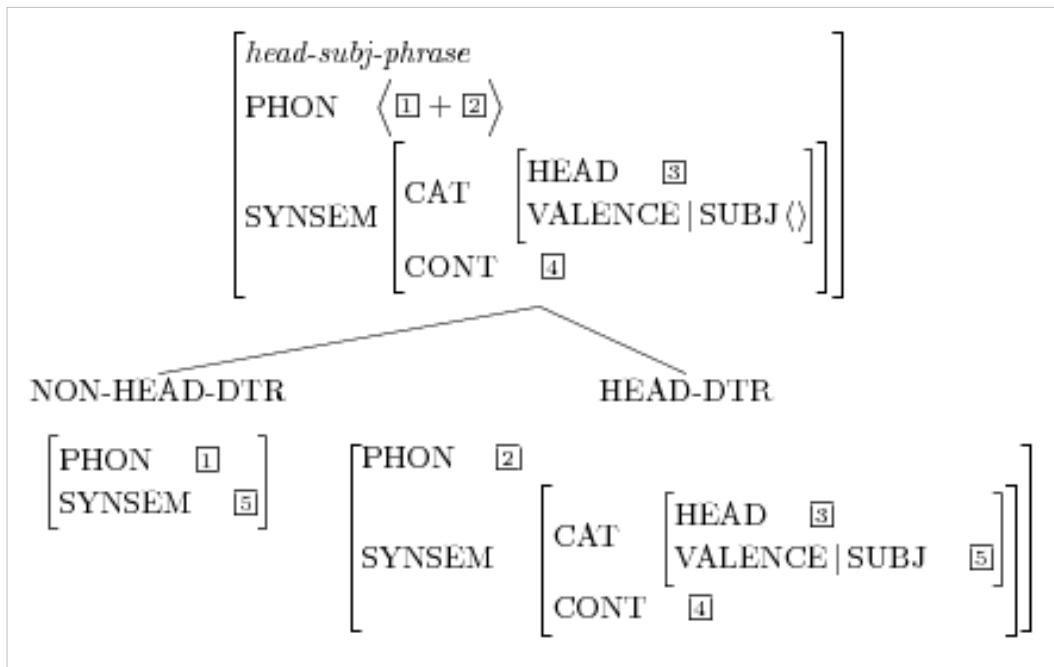
<i>word</i>													
PHON	$\langle \text{'she}' \rangle$												
SYNSEM	<table border="1"> <tr> <td><i>synsem</i></td> <td></td> </tr> <tr> <td>CAT</td><td>[HEAD noun]</td> </tr> <tr> <td>CONT</td><td> <table border="1"> <tr> <td><i>ref-index</i></td> <td></td> </tr> <tr> <td>PER</td><td>3rd</td> </tr> <tr> <td>NUM</td><td>sing</td> </tr> </table> </td> </tr> </table>	<i>synsem</i>		CAT	[HEAD noun]	CONT	<table border="1"> <tr> <td><i>ref-index</i></td> <td></td> </tr> <tr> <td>PER</td><td>3rd</td> </tr> <tr> <td>NUM</td><td>sing</td> </tr> </table>	<i>ref-index</i>		PER	3rd	NUM	sing
<i>synsem</i>													
CAT	[HEAD noun]												
CONT	<table border="1"> <tr> <td><i>ref-index</i></td> <td></td> </tr> <tr> <td>PER</td><td>3rd</td> </tr> <tr> <td>NUM</td><td>sing</td> </tr> </table>	<i>ref-index</i>		PER	3rd	NUM	sing						
<i>ref-index</i>													
PER	3rd												
NUM	sing												

Signs of type *phrase* unify with one or more children and propagate information upward. The following AVM encodes the immediate dominance rule for a *head-subj-phrase*, which requires two children: the head child (a verb) and a non-head child that fulfills the verb's SUBJ constraints.

<i>head-subj-phrase</i>									
PHON	$\langle \boxed{1} + \boxed{2} \rangle$								
SYNSEM	<table border="1"> <tr> <td>CAT</td><td>[HEAD VALENCE SUBJ $\boxed{3}$]</td> </tr> <tr> <td>CONT</td><td>$\boxed{4}$</td> </tr> </table>	CAT	[HEAD VALENCE SUBJ $\boxed{3}$]	CONT	$\boxed{4}$				
CAT	[HEAD VALENCE SUBJ $\boxed{3}$]								
CONT	$\boxed{4}$								
HEAD-DTR	<table border="1"> <tr> <td>PHON</td><td>$\boxed{2}$</td> </tr> <tr> <td>SS</td><td> <table border="1"> <tr> <td>CAT</td><td>[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]</td> </tr> <tr> <td>CONT</td><td>$\boxed{4}$</td> </tr> </table> </td> </tr> </table>	PHON	$\boxed{2}$	SS	<table border="1"> <tr> <td>CAT</td><td>[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]</td> </tr> <tr> <td>CONT</td><td>$\boxed{4}$</td> </tr> </table>	CAT	[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]	CONT	$\boxed{4}$
PHON	$\boxed{2}$								
SS	<table border="1"> <tr> <td>CAT</td><td>[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]</td> </tr> <tr> <td>CONT</td><td>$\boxed{4}$</td> </tr> </table>	CAT	[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]	CONT	$\boxed{4}$				
CAT	[HEAD $\boxed{3}$ VALENCE SUBJ $\boxed{5}$]								
CONT	$\boxed{4}$								
NON-HEAD-DTR	<table border="1"> <tr> <td>PHON</td><td>$\boxed{1}$</td> </tr> <tr> <td>SS</td><td>$\boxed{5}$</td> </tr> </table>	PHON	$\boxed{1}$	SS	$\boxed{5}$				
PHON	$\boxed{1}$								
SS	$\boxed{5}$								

The end result is a sign with a verb head, empty subcategorization features, and a phonological value that orders the two children.

Although the actual grammar of HPSG is composed entirely of feature structures, linguists often use trees to represent the unification of signs where the equivalent AVM would be unwieldy.



Implementations

Various parsers based on the HPSG formalism have been written and optimizations are currently being investigated. An example of a system analyzing German sentences is provided by the Freie Universität Berlin.^[6] In addition the Grammar Group of the Freie Universität Berlin provides open source grammars that were implemented in the **TRALE** system. Currently there are grammars for German,^[7] Mandarin Chinese,^[8] Maltese,^[9] and Persian^[10] that share a common core and are publicly available.

Large HPSG grammars of various languages are being developed in the Deep Linguistic Processing with HPSG Initiative (DELPH-IN).^[11] Wide-coverage grammars of German,^[12] English^[13] and Japanese^[14] are available under an open-source license. These grammars can be used with the open-source HPSG systems **LKB** and **PET**. DELPH-IN grammars can typically be used for both parsing and generation. Treebanks also distributed by DELPH-IN are being used to develop and test the grammars, as well as train ranking models to decide on plausible interpretations when parsing (or realizations when generating).

Enju is a freely available wide-coverage probabilistic HPSG parser for English developed by the Tsujii Laboratory at The University of Tokyo in Japan. Its robustness sets it apart from most other HPSG parsers.^[15]

References

- [1] Pollard, Carl, and Ivan A. Sag. 1987. Information-based syntax and semantics. Volume 1. Fundamentals. CLSI Lecture Notes 13.
- [2] Pollard, Carl; Ivan A. Sag. (1994). *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.
- [3] Sag, Ivan A. 1997. English Relative Clause Constructions. *Journal of Linguistics* . 33.2: 431-484
- [4] Pollard, Carl; Ivan A. Sag. (1994). *Head-driven phrase structure grammar*. Chicago: University of Chicago Press.
- [5] Sag, Ivan A.; Thomas Wasow; & Emily Bender. (2003). *Syntactic theory: a formal introduction*. 2nd ed. Chicago: University of Chicago Press.
- [6] The Babel-System: HPSG Interactive (<http://hpsg.fu-berlin.de/~stefan/Babel/Interaktiv/>)
- [7] Berligram (<http://hpsg.fu-berlin.de/Fragments/Berligram/>)
- [8] Chinese (<http://hpsg.fu-berlin.de/Fragments/Chinese/>)
- [9] Maltese (<http://hpsg.fu-berlin.de/Fragments/Maltese/>)
- [10] Persian (<http://hpsg.fu-berlin.de/Fragments/Persian/>)
- [11] DELPH-IN: Open-Source Deep Processing (<http://www.delph-in.net/>)
- [12] Berthold Crysmann (<http://gg.dfki.de/>)
- [13] English Resource Grammar and Lexicon (<http://www.delph-in.net/erg/>)

- [14] JacyTop - Deep Linguistic Processing with HPSG (DELPH-IN) (<http://www.delph-in.net/jacy>)
- [15] Tsuji Lab: Enju parser home page (<http://www-tsujii.is.s.u-tokyo.ac.jp/enju/>) (retrieved Nov 24, 2009)

Further reading

- Carl Pollard, Ivan A. Sag (1987): *Information-based Syntax and Semantics. Volume 1: Fundamentals*. Stanford: CSLI Publications.
- Carl Pollard, Ivan A. Sag (1994): *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press. ((<http://cslipublications.stanford.edu/site/0226674479.html>))
- Ivan A. Sag, Thomas Wasow, Emily Bender (2003): *Syntactic Theory: a formal introduction, Second Edition*. Chicago: University of Chicago Press. ((<http://cslipublications.stanford.edu/site/1575864002.html>))
- Levine, Robert D.; W. Detmar Meurers (2006). "Head-Driven Phrase Structure Grammar: Linguistic Approach, Formal Foundations, and Computational Realization" (<http://www.ling.ohio-state.edu/~dm/papers/ell2-hpsg.pdf>). In Keith Brown. *Encyclopedia of Language and Linguistics* (second edition ed.). Oxford: Elsevier.

External links

- Stanford HPSG homepage (<http://hpsg.stanford.edu>) – includes on-line proceedings of an annual HPSG conference
- Ohio State HPSG homepage (<http://linguistics.osu.edu/research/groupsResources/hpsg/>)
- DELPH-IN network for HPSG grammar development (<http://www.delph-in.net/>)
- Basic Overview of HPSG (http://emsah.uq.edu.au/linguistics/Working%20Papers/ananda_ling/HPSG_Summary.htm)
- Comparison of HPSG with alternatives, and a historical perspective (<http://stellar.mit.edu/S/course/24/fa03/24.960/index.html>)
- Bibliography of HPSG Publications (<http://hpsg.fu-berlin.de/HPSG-Bib/>)
- LaTeX package for drawing AVMs (<http://nlp.stanford.edu/~manning/tex/>) – includes documentation

Categorial grammar

Categorial grammar (also **categorical grammar**) is a term used for a family of formalisms in natural language syntax motivated by the principle of compositionality and organized according to the view that syntactic constituents should generally combine as functions or according to a function-argument relationship. Most versions of Categorial Grammar analyze sentence structure in terms of constituencies (as opposed to dependencies) and are therefore phrase structure grammars (as opposed to dependency grammars).

Basics of categorial grammar

A **categorial grammar** consists of two parts, a lexicon, which assigns a set of types (also called categories) to each basic symbol, and some type inference rules, which determine how the type of a string of symbols follows from the types of the constituent symbols. It has the advantage that the type inference rules can be fixed once and for all, so that the specification of a particular language grammar is entirely determined by the lexicon.

A **categorial grammar** shares some features with the simply typed lambda calculus. Whereas the lambda calculus has only one function type $A \rightarrow B$, a categorial grammar typically has two function types, one type which is applied on the left, one on the right. For example, a simple categorial grammar might have two function types B/A and $A\backslash B$. The first, B/A , is the type of a phrase that results in a phrase of type B when followed (on the right) by a phrase of type A . The second, $A\backslash B$, is the type of a phrase that results in a phrase of type B when preceded (on the left) by a phrase of type A .

As Lambek explains, the notation is based upon algebra. A fraction when multiplied by (i.e. concatenated with) its denominator yields its numerator. Since concatenation is not commutative, it makes difference whether the denominator occurs to the left or right. The concatenation must be on the same side as the denominator for it to cancel out.

The first and simplest kind of categorial grammar is called a basic categorial grammar, or sometimes an AB-grammar (after Ajdukiewicz and Bar-Hillel). Given a set of primitive types Prim , let $\text{Tp}(\text{Prim})$ be the set of types constructed from primitive types. In the basic case, this is the least set such that $\text{Prim} \subseteq \text{Tp}(\text{Prim})$ and if $X, Y \in \text{Tp}(\text{Prim})$ then $(X/Y), (Y\backslash X) \in \text{Tp}(\text{Prim})$. Think of these as purely formal expressions freely generated from the primitive types; any semantics will be added later. Some authors assume a fixed infinite set of primitive types used by all grammars, but by making the primitive types part of the grammar, the whole construction is kept finite.

A basic categorial grammar is a tuple $(\Sigma, \text{Prim}, S, \triangleleft)$ where Σ is a finite set of symbols, Prim is a finite set of primitive types, and $S \in \text{Tp}(\text{Prim})$.

The relation \triangleleft is the lexicon, which relates types to symbols $(\triangleleft) \subseteq \text{Tp}(\text{Prim}) \times \Sigma$. Since the lexicon is finite, it can be specified by listing a set of pairs like $\text{TYPE} \triangleleft \text{symbol}$.

Such a grammar for English might have three basic types (N , NP , and S), assigning count nouns the type N , complete noun phrases the type NP , and sentences the type S . Then an adjective could have the type N/N , because if it is followed by a noun then the whole phrase is a noun. Similarly, a determiner has the type NP/N , because it forms a complete noun phrase when followed by a noun. Intransitive verbs have the type $NP\backslash S$, and transitive verbs the type $(NP\backslash S)/NP$. Then a string of words is a sentence if it has overall type S . For example, take the string "the bad boy made that mess". Now "the" and "that" are determiners, "boy" and "mess" are nouns, "bad" is an adjective, and "made" is a transitive verb, so the lexicon is $\{ NP/N \triangleleft \text{the}, NP/N \triangleleft \text{that}, N \triangleleft \text{boy}, N \triangleleft \text{mess}, N/N \triangleleft \text{bad}, (NP\backslash S)/NP \triangleleft \text{made} \}$. and the sequence of types in the string is

the bad boy made that mess
 $NP/N, N/N, N, (NP \setminus S)/NP, NP/N, N$

now find functions and appropriate arguments and reduce them according to the two inference rules
 $X \leftarrow X/Y, Y$ and $X \leftarrow Y, Y \setminus X$:

- . $NP/N, N/N, N, (NP \setminus S)/NP, \overbrace{NP/N, N}$
- . $NP/N, N/N, N, \overbrace{(NP \setminus S)/NP, NP}$
- . $NP/N, \overbrace{N/N, N}, \overbrace{(NP \setminus S)}$
- . $\overbrace{NP/N, N}, \overbrace{(NP \setminus S)}$
- . $\overbrace{NP, (NP \setminus S)}$

The fact that the result is S means that the string is a sentence, while the sequence of reductions shows that it must be parsed as ((the (bad boy)) (made (that mess))).

Categorial grammars of this form (having only function application rules) are equivalent in generative capacity to context-free grammars and are thus often considered inadequate for theories of natural language syntax. Unlike CFGs, categorial grammars are lexicalized, meaning that only a small number of (mostly language-independent) rules are employed, and all other syntactic phenomena derive from the lexical entries of specific words.

Another appealing aspect of categorial grammars is that it is often easy to assign them a compositional semantics, by first assigning interpretation types to all the basic categories, and then associating all the derived categories with appropriate function types. The interpretation of any constituent is then simply the value of a function at an argument. With some modifications to handle intensionality and quantification, this approach can be used to cover a wide variety of semantic phenomena.

Lambek Calculus

A Lambek grammar is an elaboration of this idea which has a concatenation operator for types, and several other inference rules. Pentus has shown that these still have the generative capacity of context-free grammars.

For the Lambek calculus, there is a type concatenation operator \star , so that $\text{Prim} \subseteq \text{Tp}(\text{Prim})$ and if $X, Y \in \text{Tp}(\text{Prim})$ then $(X/Y), (X \setminus Y), (X \star Y) \in \text{Tp}(\text{Prim})$.

The Lambek calculus consists of several deduction rules which specify how type inclusion assertions can be derived. In the following rules, upper case roman letters stand for types, upper case Greek letters stand for sequences of types. A sequent of the form $X \leftarrow \Gamma$ can be read: a string is of type X if it consists of the concatenation of strings of each of the types in Γ . If a type is interpreted as a set of strings, then the \leftarrow may be interpreted as \supseteq , that is, "includes as a subset". A horizontal line means that the inclusion above the line implies the one below the line.

The process is begun by the Axiom rule, which as no antecedents and just says that any type includes itself.

$$(Axiom) \quad \overline{X \leftarrow X}$$

The Cut rule says that inclusions can be composed.

$$(Cut) \quad \frac{Z \leftarrow \Delta X \Delta' \quad X \leftarrow \Gamma}{Z \leftarrow \Delta \Gamma \Delta'}$$

The other rules come in pairs, one pair for each type construction operator, each pair consisting of one rule for the operator in the target, one in the source, of the arrow. The name of a rule consists of the operator and an arrow, with the operator on the side of the arrow on which it occurs in the conclusion.

Target	Source
$(\backslash \leftarrow) \frac{Y \leftarrow X\Gamma}{X \setminus Y \leftarrow \Gamma}$	$(\leftarrow \backslash) \frac{Z \leftarrow \Delta Y \Delta' \quad X \leftarrow \Gamma}{Z \leftarrow \Delta \Gamma(X \setminus Y) \Delta'}$
$(/ \leftarrow) \frac{Y \leftarrow \Gamma X}{Y/X \leftarrow \Gamma}$	$(\leftarrow /) \frac{Z \leftarrow \Delta Y \Delta' \quad X \leftarrow \Gamma}{Z \leftarrow \Delta(Y/X) \Gamma \Delta'}$
$(\star \leftarrow) \frac{X \leftarrow \Gamma \quad Y \leftarrow \Gamma'}{X \star Y \leftarrow \Gamma \Gamma'}$	$(\leftarrow \star) \frac{Z \leftarrow \Delta X Y \Delta'}{Z \leftarrow \Delta(X \star Y) \Delta'}$

For an example, here is a derivation of "type raising", which says that $(B/A) \setminus B \leftarrow A$. The names of rules and the substitutions used are to the right.

$$\frac{\overline{B \leftarrow B} \quad \overline{A \leftarrow A}}{\overline{B \leftarrow (B/A), \quad A} \quad \begin{array}{l} (\text{Axioms}) \\ (\leftarrow /) [Z = Y = B, X = A, \Gamma = (A), \Delta = \Delta' = ()] \\ (\backslash \leftarrow) [Y = B, X = (B/A), \Gamma = (A)] \end{array}} \quad \frac{}{(B/A) \setminus B \leftarrow A}$$

Relation to Context Free Grammars

Recall that a context-free grammar is a 4-tuple:

$$G = (V, \Sigma, ::=, S) \text{ where}$$

1. V is a finite set of *non-terminals* or *variables*.
2. Σ is a finite set of *terminal symbols*.
3. $::=$ is a finite set of production rules, that is, a finite relation $(::=) \subseteq V \times (V \cup \Sigma)^*$.
4. S is the start variable.

From the point of view of categorial grammars, a context-free grammar can be seen as a calculus with a set of special purpose axioms for each language, but with no type construction operators and no inference rules except Cut.

Specifically, given a context-free grammar as above, define a categorial grammar $(\text{Prim}, \Sigma, \triangleleft, S)$ where $\text{Prim} = V \cup \Sigma$, and $\text{Tp}(\text{Prim}) = \text{Prim}$. Let there be an axiom $x \leftarrow x$ for every symbol $x \in V \cup \Sigma$, an axiom $X \leftarrow \Gamma$ for every production rule $X ::= \Gamma$, a lexicon entry $s \triangleleft s$ for every terminal symbol $s \in \Sigma$, and Cut for the only rule. This categorial grammar generates the same language as the given CFG.

Of course, this is not a basic categorial grammar, since it has special axioms that depend upon the language; i.e. it is not lexicalized. Also, it makes no use at all of non-primitive types.

To show that any context-free language can be generated by a basic categorial grammar, recall that any context-free language can be generated by a context-free grammar in Greibach normal form.

The grammar is in Greibach normal form if every production rule is of the form $A ::= s A_0 \dots A_{N-1}$, where capital letters are variables, $s \in \Sigma$, and $N \geq 0$, that is, the right side of the production is a single terminal symbol followed by zero or more (non-terminal) variables.

Now given a CFG in Greibach normal form, define a basic categorial grammar with a primitive type for each non-terminal variable $\text{Prim} = V$, and with an entry in the lexicon $A/A_{N-1}/\dots/A_0 \triangleleft s$, for each production rule $A ::= s A_0 \dots A_{N-1}$. It is fairly easy to see that this basic categorial grammar generates the same language as the original CFG. Note that the lexicon of this grammar will generally assign multiple types to each symbol.

The same construction works for Lambek grammars, since they are an extension of basic categorial grammars. It is necessary to verify that the extra inference rules do not change the generated language. This can be done and shows that every context-free language is generated by some Lambek grammar.

To show the converse, that every language generated by a Lambek grammar is context-free, is much more difficult. It was an open problem for nearly thirty years, from the early 1960s until about 1991 when it was proven by Pentus.

The basic idea is, given a Lambek grammar, $(\text{Prim}, \Sigma, \triangleleft, S)$ construct a context-free grammar $(V, \Sigma, ::=, S)$ with the same set of terminal symbols, the same start symbol, with variables some (not all) types $V \subset \text{Tp}(\text{Prim})$, and with a production rule $T ::= s$ for each entry $T \triangleleft s$ in the lexicon, and production rules $T ::= \Gamma$ for certain sequents $T \leftarrow \Gamma$ which are derivable in the Lambek calculus. Of course, there are infinitely many types and infinitely many derivable sequents, so in order to make a finite grammar it is necessary put a bound on the size of the types and sequents that are needed. The heart of Pentus's proof is to show that there is such a finite bound.

Notation

The notation in this field is not standardized. The notations used in formal language theory, logic, category theory, and linguistics, conflict with each other. In logic, arrows point to the more general from the more particular, that is, to the conclusion from the hypotheses. In this article, this convention is followed, i.e. the target of the arrow is the more general (inclusive) type.

In logic, arrows usually point left to right. In this article this convention is reversed for consistency with the notation of context-free grammars, where the single non-terminal symbol is always on the left. We use the symbol $::=$ in a production rule as in Backus-Naur form. Some authors use an arrow, which unfortunately may point in either direction, depending on whether the grammar is thought of as generating or recognizing the language.

Some authors on categorial grammars write $B \setminus A$ instead of $A \setminus B$. The convention used here follows Lambek and algebra.

Historical notes

The basic ideas of categorial grammar date from work by Kazimierz Ajdukiewicz (in 1935) and Yehoshua Bar-Hillel (in 1953). In 1958, Joachim Lambek introduced a syntactic calculus that formalized the function type constructors along with various rules for the combination of functions. This calculus is a forerunner of linear logic in that it is a substructural logic. Montague grammar uses an ad hoc syntactic system for English that is based on the principles of categorial grammar. Although Montague's work is sometimes regarded as syntactically uninteresting, it helped to bolster interest in categorial grammar by associating it with a highly successful formal treatment of natural language semantics. More recent work in categorial grammar has focused on the improvement of syntactic coverage. One formalism which has received considerable attention in recent years is Steedman and Szabolcsi's combinatory categorial grammar which builds on combinatory logic invented by Moses Schönfinkel and Haskell Curry.

There are a number of related formalisms of this kind in linguistics, such as type logical grammar and abstract categorial grammar.

Some definitions

- **Derivation**

A derivation is a binary tree that encodes a proof.

- **Parse tree**

A derivation can be displayed as a parse tree, showing the syntactic structure of a sentence.

- **Functor and Argument**

In a right (left) function application, the node of the type $A \setminus B$ (B/A) is called the functor, and the node of the type A is called an argument.

- **Functor-argument structure**

Refinements of categorial grammar

A variety of changes to categorial grammar have been proposed to improve syntactic coverage. Some of the most common ones are listed below.

Features and subcategories

Most systems of categorial grammar subdivide categories. The most common way to do this is by tagging them with features, such as person, gender, number, and tense. Sometimes only atomic categories are tagged in this way. In Montague grammar, it is traditional to subdivide function categories using a multiple slash convention, so A/B and $A//B$ would be two distinct categories of left-applying functions, that took the same arguments but could be distinguished between by other functions taking them as arguments.

Function composition

Rules of function composition are included in many categorial grammars. An example of such a rule would be one that allowed the concatenation of a constituent of type A/B with one of type B/C to produce a new constituent of type A/C . The semantics of such a rule would simply involve the composition of the functions involved. Function composition is important in categorial accounts of conjunction and extraction, especially as they relate to phenomena like right node raising. The introduction of function composition into a categorial grammar leads to many kinds of derivational ambiguity that are vacuous in the sense that they do not correspond to semantic ambiguities.

Conjunction

Many categorial grammars include a typical conjunction rule, of the general form $X \text{ CONJ } X \rightarrow X$, where X is a category. Conjunction can generally be applied to nonstandard constituents resulting from type raising or function composition..

Discontinuity

The grammar is extended to handle linguistic phenomena such as discontinuous idioms, gapping and extraction.

References

- Curry, Haskell B. and Richard Feys (1958), Combinatory Logic, Vol. 1. North-Holland.
- Jacobson, Pauline (1999), "Towards a variable-free semantics." *Linguistics and Philosophy* 22, 1999. pp. 117–184
- Lambek, J. (1958) "The mathematics of sentence structure", *Amer. Math. Monthly* 65, 3 pp 154–170
- Pentus, Mati (1997) "Lambek Calculus and Formal Grammars", *Amer. Math. Soc. Transl.*
- Steedman, Mark (1987), "Combinatory grammars and parasitic gaps". *Natural Language and Linguistic Theory* 5, 403–439.
- Steedman, Mark (1996), *Surface Structure and Interpretation*. The MIT Press.
- Steedman, Mark (2000), *The Syntactic Process*. The MIT Press.
- Szabolcsi, Anna (1989), "Bound variables in syntax (are there any?)." *Semantics and Contextual Expression*, ed. by Bartsch, van Benthem, and van Emde Boas. Foris, 294–318.
- Szabolcsi, Anna (1992), "Combinatory grammar and projection from the lexicon." *Lexical Matters*. CSLI Lecture Notes 24, ed. by Sag and Szabolcsi. Stanford, CSLI Publications. 241–269.
- Szabolcsi, Anna (2003), "Binding on the fly: Cross-sentential anaphora in variable-free semantics". *Resource Sensitivity in Binding and Anaphora*, ed. by Kruijff and Oehrle. Kluwer, 215–229.
- Morril, Glynn (1995), "Discontinuity in categorial grammar" *Linguistics and Philosophy*. Springer, 175-219.

Further reading

- Michael Moortgat, *Categorial Type Logics*, Chapter 2 in J. van Benthem and A. ter Meulen (eds.) *Handbook of Logic and Language*. Elsevier, 1997, ISBN 0-262-22053-9
- Wojciech Buszkowski, *Mathematical linguistics and proof theory*, Chapter 12 in J. van Benthem and A. ter Meulen (eds.) *Handbook of Logic and Language*. Elsevier, 1997, ISBN 0-262-22053-9
- Gerhard Jäger (2005). *Anaphora and Type Logical Grammar*. Springer. ISBN 978-1-4020-3904-1.
- Glyn Morrill (2010). *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford University Press. ISBN 978-0-19-958986-9.
- Richard Moot; Christian Retore (2012). *The Logic of Categorial Grammars: A Deductive Account of Natural Language Syntax and Semantics*. Springer Verlag. ISBN 978-3-642-31554-1.

External links

- Grammar, categorial ^[1] at Springer Encyclopaedia of Mathematics
- <http://plato.stanford.edu/entries/typological-grammar/>

References

[1] <http://eom.springer.de/g/g044770.htm>

Tree-adjoining grammar

Tree-adjoining grammar (TAG) is a grammar formalism defined by Aravind Joshi. Tree-adjoining grammars are somewhat similar to context-free grammars, but the elementary unit of rewriting is the tree rather than the symbol. Whereas context-free grammars have rules for rewriting symbols as strings of other symbols, tree-adjoining grammars have rules for rewriting the nodes of trees as other trees (see tree (graph theory) and tree (data structure)).

History

TAG originated in investigations by Joshi and his students into the family of adjunction grammars (AG),^[1] the "string grammar" of Zellig Harris. AGs handle endocentric properties of language in a natural and effective way, but do not have a good characterization of exocentric constructions; the converse is true of rewrite grammars, or phrase-structure grammar (PSG). In 1969, Joshi introduced a family of grammars that exploits this complementarity by mixing the two types of rules. A few very simple rewrite rules suffice to generate the vocabulary of strings for adjunction rules. This family is distinct from the Chomsky hierarchy but intersects it in interesting and linguistically relevant ways.^[2]

Description

The rules in a TAG are trees with a special leaf node known as the *foot node*, which is anchored to a word. There are two types of basic trees in TAG: *initial* trees (often represented as ' α ') and *auxiliary* trees (' β '). Initial trees represent basic valency relations, while auxiliary trees allow for recursion.^[3] Auxiliary trees have the root (top) node and foot node labeled with the same symbol. A derivation starts with an initial tree, combining via either *substitution* or *adjunction*. Substitution replaces a frontier node with another tree whose top node has the same label. Adjunction inserts an auxiliary tree into the center of another tree.^[4] The root/foot label of the auxiliary tree must match the label of the node at which it adjoins.

Other variants of TAG allow multi-component trees, trees with multiple foot nodes, and other extensions.

Complexity and application

Tree-adjoining grammars are often described as mildly context-sensitive, meaning that they possess certain properties that make them more powerful (in terms of weak generative capacity) than context-free grammars, but less powerful than indexed or context-sensitive grammars. Mildly context-sensitive grammars are conjectured to be powerful enough to model natural languages while remaining efficiently parsable in the general case.^[5]

A TAG can describe the language of squares (in which some arbitrary string is repeated), and the language $\{a^n b^n c^n d^n | 1 \leq n\}$. This type of processing can be represented by an embedded pushdown automaton.

Languages with cubes (i.e. triplicated strings) or with more than four distinct character strings of equal length cannot be generated by tree-adjoining grammars.

For these reasons, languages generated by tree-adjoining grammars are referred to as *mildly context-sensitive languages*.

Equivalencies

Vijay-Shanker and Weir (1994)^[6] demonstrates that Linear Indexed Grammars, Combinatory Categorial Grammars, Tree-adjoining Grammars, and Head Grammars are weakly equivalent formalisms, in that they all define the same string languages.

References

- [1] Joshi, Aravind; S. R. Kosaraju, H. Yamada (1969). *String Adjunct Grammars*. Proceedings Tenth Annual Symposium on Automata Theory, Waterloo, Canada.
- [2] Joshi, Aravind (1969). *Properties of Formal Grammars with Mixed Types of Rules and Their Linguistic Relevance*. Proceedings Third International Symposium on Computational Linguistics, Stockholm, Sweden.
- [3] Jurafsky, Daniel; James H. Martin (2000). *Speech and Language Processing*. Upper Saddle River, NJ: Prentice Hall. pp. 354.
- [4] Joshi, Aravind; Owen Rambow (2003). "A Formalism for Dependency Grammar Based on Tree Adjoining Grammar" (<http://www1.cs.columbia.edu/~rambow/papers/joshi-rambow-2003.pdf>). *Proceedings of the Conference on Meaning-Text Theory*..
- [5] Joshi, Aravind (1985). "How much context-sensitivity is necessary for characterizing structural descriptions". In D. Dowty, L. Karttunen, and A. Zwicky, (eds.). *Natural Language Processing: Theoretical, Computational, and Psychological Perspectives*. New York, NY: Cambridge University Press. pp. 206–250.
- [6] Vijay-Shanker, K. and Weir, David J. 1994. *The Equivalence of Four Extensions of Context-Free Grammars*. Mathematical Systems Theory 27(6): 511-546.

External links

- The XTAG project (<http://www.cis.upenn.edu/~xtag/>), which uses a TAG for natural language processing.
- A tutorial on TAG (<http://www.let.rug.nl/~vannoord/papers/diss/diss/node59.html>)
- Another tutorial (<http://www.computing.dcu.ie/~yguo/doc/talk/TAG.pdf>) with focus on comparison with Lexical Functional Grammar and grammars extraction from Treebank
- SemConst Documentation (<http://wiki.loria.fr/wiki/SemConst/Documentation#Background>) A quick survey on Syntax and Semantic Interface problematic within the TAG framework.
- The TuLiPa project (<http://sourcesup.cru.fr/tulipa/>) The Tübingen Linguistic Parsing Architecture (TuLiPA) is a multi-formalism syntactic (and semantic) parsing environment, designed mainly for Multi-Component Tree Adjoining Grammars with Tree Tuples
- The Metagrammar Toolkit (<http://mgkit.gforge.inria.fr/>) which provides several tools to edit and compile MetaGrammars into TAGs. It also include a wide coverage French Metagrammars.
- LLP2 (<http://www.loria.fr/~azim/LLP2/help/fr/index.html>) A Lexicalized Tree Adjoining Grammar parser which provides an easy to use graphical environment (page in French)

Nanosyntax

Nanosyntax is an approach to syntax in which syntactic parse trees are built from a large number of syntactic constituents. Each morpheme may correspond to several such elements, which do not have to form a "subtree".

Some recent work in theoretical linguistics suggests that the "atoms" of syntax are much smaller than words or morphemes. From that it immediately follows that the responsibility of syntax is not limited to ordering "preconstructed" words. Instead, within the framework of nanosyntax,^[1] the words are derived entities built in syntax, rather than primitive elements supplied by a lexicon.

The beginnings of nanosyntax can be traced to a 1993 article by Kenneth Hale and S. Jay Keyser titled 'On Argument Structure and the Lexical Representation of Syntactic Relations'^[2], which first introduced the concept of l-syntax.

References

- [1] Starke, Michal. 2011. Towards an elegant solution to language variation: Variation reduces to the size of lexically stored trees. MS. Barcelona, Spain
- [2] Hale, Kenneth and S. Jay Keyser. 1993. On Argument Structure and the Lexical Representation of Syntactic Relations. In *The View from Building 20*, edited by Kenneth Hale and S. Jay Keyser, pp. 53-109. Cambridge, MA: The MIT Press. ISBN 978-0-262-58124-0

External links

- Nanosyntax project at Center for Advanced Study in Theoretical Linguistics (<http://nanosyntax.auf.net>)

Arc pair grammar

In linguistics, **Arc Pair grammar** is a syntactic theory developed by David E. Johnson and Paul Postal which is a formalized continuation of relational grammar developed by David M. Perlmutter and Paul M. Postal.

Like relational grammar, arc pair grammar is greatly concerned with grammatical relations (as opposed to the constituent structure focus of other generative theories like versions of Chomskyan transformational grammar). In contrast to the generative-enumerative (proof-theoretic) approach to syntax assumed by transformational grammar, arc pair grammar takes a model-theoretic approach. In arc pair grammar, linguistic laws and language-specific rules of grammar are formalized in the same manner, namely, as logical statements in an axiomatic theory. Further, sentences of a language, understood as structures of a certain type, are the models of the set of linguistic laws and language-specific statements, thereby reducing the notion of grammaticality to the logical notion of model-theoretic satisfaction.

For a brief history of early work on relational grammar and arc pair grammar, see Newmeyer, 1980. For a more detailed history of model-theoretic approaches in linguistics, see Pullum and Scholz, 2005 and Pullum, 2007.

Bibliography

- Johnson, David E.; & Postal, Paul M. (1980). *Arc pair grammar*. Princeton: Princeton University Press. ISBN 0-691-08270-7
- Postal, Paul M. (1982). "Some arc pair grammar descriptions". In P. Jacobson & G. K. Pullum (Eds.), *The nature of syntactic representation* (pp. 341-425). Dordrecht: D. Reidel. ISBN 978-90-277-1290-5
- Newmeyer, Frederick (1980). *Linguistics in America*. New York: Academic Press. ISBN 978-90-277-1290-5
- Pullum, Geoffrey K. and Barbara C. Scholz. (2005). "Contrasting applications of logic in natural language syntactic description." In Petr Hájek, Luis Valdés-Villanueva, and Dag Westerståhl (eds.), *Logic, Methodology and Philosophy of Science: Proceedings of the Twelfth International Congress*, 481-503. ISBN 978-1-904987-21-5
- Pullum, Geoffrey K. (2007) "The evolution of model-theoretic frameworks in linguistics." In the proceedings of the Model-Theoretic Syntax at 10 workshop at ESSLLI 2007, Trinity College, Dublin.

Generative semantics

Generative semantics is the name of a research program within linguistics, initiated by the work of various early students of Noam Chomsky: John R. Ross, Paul Postal, and later James McCawley. George Lakoff was also instrumental in developing and advocating the theory.¹

The approach developed out of transformational generative grammar in the mid 1960s, but stood largely apart from, and in opposition to, work by Noam Chomsky and his later students. This move led to a more abstract framework and lately to the abandonment of the notion of deep structure.

A number of ideas from later work in generative semantics have been incorporated into cognitive linguistics, Head-Driven Phrase Structure Grammar (HPSG), Construction Grammar, and into mainstream Chomskyan linguistics.^[1]

History

The nature and genesis of the program are a matter of some controversy and have been extensively debated. Generative semanticists took Chomsky's concept of Deep Structure and ran with it, assuming (contrary to later work by Chomsky and Ray Jackendoff) that deep structures were the sole input to semantic interpretation. This assumption, combined with a tendency to consider a wider range of empirical evidence than Chomskyan linguists, led generative semanticists to develop considerably more abstract and complex theories of deep structure than those advocated by Chomsky and his students—and indeed to abandon altogether the notion of "deep structure" as a locus of lexical insertion.

Throughout the late 1960s and 1970s, there were heated debates between generative semanticists and more orthodox Chomskyans. The generative semanticists lost the debate, insofar as their research program ground to a halt by the 1980s. However, this was in part because the interests of key generative semanticists such as George Lakoff had gradually shifted away from the narrow study of syntax and semantics.

"Interpretive" vs. "generative" semantics

The controversy surrounding generative semantics stemmed in part from the competition between two fundamentally different approaches to semantics within transformational generative syntax. The first semantic theories designed to be compatible with transformational syntax were *interpretive*. Syntactic rules enumerated a set of well-formed sentences paired with syntactic structures, each of which was assigned an *interpretation* by the rules of a separate semantic theory. This left syntax relatively (though by no means entirely) "autonomous" with respect to semantics,

and was the approach preferred by Chomsky.

In contrast, generative semanticists argued that interpretations were generated directly by the grammar as deep structures, and were subsequently transformed into recognizable sentences by transformations. This approach necessitated more complex underlying structures than those proposed by Chomsky, and more complex transformations as a consequence. Despite this additional complexity, the approach was appealing in several respects. First, it offered a powerful mechanism for explaining synonymy. In his initial work in generative syntax, Chomsky motivated transformations using active/passive pairs such as "I hit John" and "John was hit by me", which despite their identical meanings have quite different surface forms.² Generative semanticists wanted to account for *all* cases of synonymy in a similar fashion—an impressively ambitious goal before the advent of more sophisticated interpretive theories in the 1970s. Second, the theory had a pleasingly intuitive structure: the form of a sentence was quite literally *derived* from its meaning via transformations. To some, interpretive semantics seemed rather "clunky" and ad-hoc in comparison. This was especially so before the development of trace theory.

Notes

There is little agreement concerning the question of whose idea generative semantics was. All of the people mentioned here have been credited with its invention (often by each other).

Strictly speaking, it was not the fact that active/passive pairs are *synonymous* that motivated the passive transformation, but the fact that active and passive verb forms have the same *selectional requirements*. For example, the agent of the verb *kick* (i.e. the thing that's doing the kicking) must be animate whether it is the subject of the active verb (as in "**John** kicked the ball") or appears in a *by* phrase after the passive verb ("The ball was kicked by **John**").

References

[1] Newmeyer, Frederick J. (1986). *Linguistic Theory in America (Second Edition)*. Academic Press. See p. 138.

Bibliography

- Brame, Michael K.. (1976). *Conjectures and refutations in syntax and semantics*. New York: North-Holland Pub. Co. ISBN 0-7204-8604-1.
- Chomsky (1957). *Syntactic structures*. The Hague: Mouton.
- Chomsky (1965). *Aspects of the theory of syntax*. Cambridge: The MIT Press.
- Chomsky (1965). *Cartesian linguistics*. New York: Harper and Row.
- Dougherty, Ray C. (1974). Generative semantics methods: A Bloomfieldian counterrevolution. *International Journal of Dravidian Linguistics*, 3, 255-286.
- Dougherty, Ray C. (1975). Reply to the critics on the Bloomfieldian counterrevolution. *International Journal of Dravidian Linguistics*, 4, 249-271.
- Fodor, Jerry A.; & Katz, Jerrold J. (Eds.). (1964). *The structure of language*. Englewood Cliffs, NJ: Prentice-Hall.
- Harris, Randy Allen. (1995). *The linguistics wars*. Oxford University Press. ISBN 0-19-509834-X.
- Huck, Geoffrey J.; & Goldsmith, John A.. (1995). *Ideology and Linguistic Theory: Noam Chomsky and the deep structure debates*. New York: Routledge.
- Katz, Jerrold J.; & Fodor, Jerry A. (1964). The structure of a semantic theory. In J. A. Fodor & J. J. Katz (Eds.) (pp. 479-518).
- Katz, Jerrold J.; & Postal, Paul M. (1964). *An integrated theory of linguistic descriptions*. Cambridge, MA: MIT Press.
- Lakoff, George. (1971). On generative semantics. In D. D. Steinberg & L. A. Jakobovits (Eds.), *Semantics: An interdisciplinary reader in philosophy, linguistics and psychology* (pp. 232-296). Cambridge: Cambridge

- University Press.
- Lakoff, George. (1976 [1963]). Toward generative semantics. In J. D. McCawley (Ed.) (pp. 43-61).
 - Lakoff, George; & Ross, John R. [Háj]. (1976). Is deep structure necessary?. In J. D. McCawley (Ed.), *Syntax and semantics 7* (pp. 159-164).
 - McCawley, James D. (1975). Discussion of Ray C. Dougherty's "Generative semantics methods: A Bloomfieldian counterrevolution". *International Journal of Dravidian Linguistics*, 4, 151-158.
 - McCawley, James D. (Ed.). (1976a). *Syntax and semantics 7: Notes from the linguistic underground*. New York: Academic Press.
 - McCawley, James D. (1976b). *Grammar and meaning*. New York: Academic Press.
 - McCawley, James D. (1979). *Adverbs, vowels, and other objects of wonder*. Chicago: University of Chicago Press.
 - Postal, Paul M. (1972). The best theory. In S. Peters (Ed.), *Goals of linguistic theory*. Englewood Cliffs, NJ: Prentice-Hall.
 - Ross, John R. (1967). Constraints on variables in syntax. (Doctoral dissertation, Massachusetts Institute of Technology). Free copy available at <http://hdl.handle.net/1721.1/15166>. (Published as Ross 1986).
 - Ross, John R. (1986). *Infinite syntax!*. Norwood, NJ: ABLEX, ISBN 0-89391-042-2.
 - Ross, John R. [Háj]. (1970). On declarative sentences. In R. A. Jacobs & P. S. Rosenbaum (Eds.), *Readings in English transformational grammar* (pp. 222-272). Washington: Georgetown University Press.
 - Ross, John R. [Háj]. (1972). Doubl-ing. In J. Kimball (Ed.), *Syntax and semantics* (Vol. 1, pp. 157-186). New York: Seminar Press.
 - Seuren, Pieter A. M. (1974). *Semantic syntax*. Oxford: Oxford University Press. ISBN 0-19-875028-5.

Dependency grammar

Dependency grammar

Dependency grammar (DG) is a class of modern syntactic theories that are all based on the dependency relation and that can be traced back primarily to the work of Lucien Tesnière. The dependency relation views the (finite) verb as the structural center of all clause structure. All other syntactic units (e.g. words) are either directly or indirectly dependent on the verb. DGs are distinct from phrase structure grammars (= constituency grammars), since DGs lack phrasal nodes. Structure is determined by the relation between a word (a head) and its dependents. Dependency structures are flatter than constituency structures in part because they lack a finite verb phrase constituent, and they are thus well suited for the analysis of languages with free word order, such as Czech and Turkish.

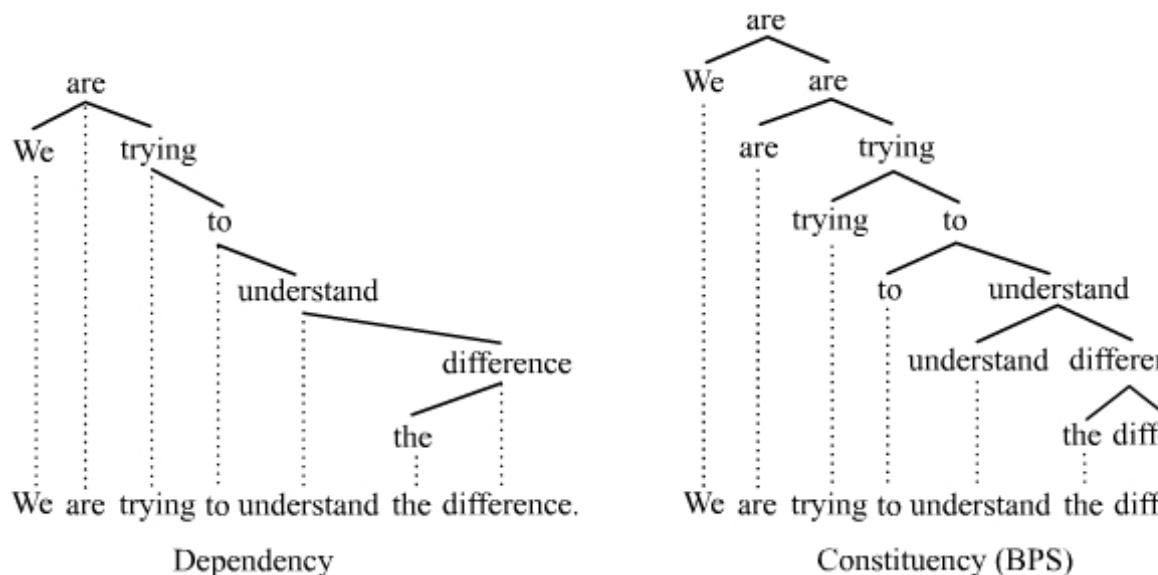
History

The notion of dependencies between grammatical units has existed since the earliest recorded grammars, e.g. Pāṇini, and the dependency concept therefore arguably predates the constituency notion by many centuries.^[1] Ibn Mada, who lived in Cordoba and Sevilla in the 12th century, may have been the first grammarian to use the term *dependency* in the grammatical sense that we use it today. In early modern times, the dependency concept seems to have coexisted side by side with the constituency concept, the latter having entered Latin, French, English and other grammars from the widespread study of term logic of antiquity.^[2]

Modern dependency grammars, however, begin primarily with the work of Lucien Tesnière. Tesnière was a Frenchman, a polyglot, and a professor of linguistics at the universities in Strasbourg and Montpellier. His major work *Éléments de syntaxe structurale* was published posthumously in 1959 – he died in 1954. The basic approach to syntax he developed seems to have been seized upon independently by others in the 1960s^[3] and a number of other dependency-based grammars have gained prominence since those early works.^[4] DG has generated a lot of interest in Germany^[5] in both theoretical syntax and language pedagogy. In recent years, the great development surrounding dependency-based theories has come from computational linguistics. Dependency-based systems are increasingly being used to parse natural language and generate tree banks. Interest in dependency grammar is growing at present, the first international conference on dependency linguistics having taken place recently (Depling 2011^[6]).

Dependency vs. constituency

Dependency is a one-to-one correspondence: for every element (e.g. word or morph) in the sentence, there is exactly one node in the structure of that sentence that corresponds to that element. The result of this one-to-one correspondence is that dependency grammars are word (or morph) grammars. All that exist are the elements and the dependencies that connect the elements into a structure. This situation should be compared with the constituency relation of phrase structure grammars. Constituency is a one-to-one-or-more correspondence, which means that given a sentence, for every element in that sentence, there are one or more nodes in the structure that correspond to that element. The result of this difference is that dependency structures are minimal^[7] compared to their constituency structure counterparts, since they tend to contain many fewer nodes.



These two trees illustrate just two possible ways to render the dependency and constituency relations (see below).

The dependency tree is an "ordered" tree, i.e. it reflects actual word order. Many dependency trees abstract away from linear order and focus just on hierarchical order, which means they do not show actual word order. The constituency tree follows the conventions of bare phrase structure (BPS), whereby the words themselves are employed as the node labels.

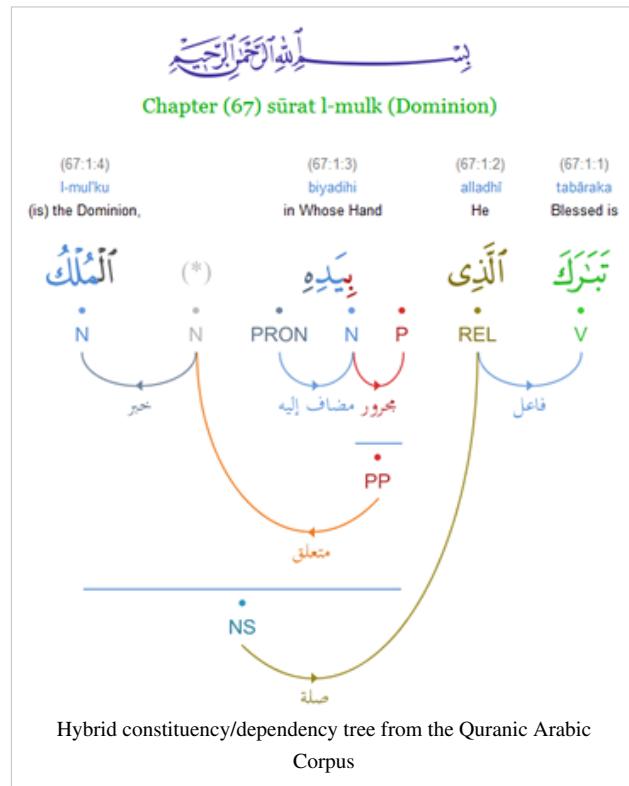
The distinction between dependency- and constituency-based grammars derives in a large part from the initial division of the clause. The constituency relation derives from an initial binary division, whereby the clause is split into a subject noun phrase (NP) and a predicate verb phrase (VP). This division is certainly present in the basic analysis of the clause that we find in the works of, for instance, Leonard Bloomfield and Noam Chomsky. Tesnière, however, argued vehemently against this binary division, preferring instead to position the verb as the root of all clause structure. Tesnière's stance was that the subject-predicate division stems from term logic and has no place in linguistics.^[8] The importance of this distinction is that if one acknowledges the initial subject-predicate division in syntax as something real, then one is likely to go down the path of constituency grammar, whereas if one rejects this division, then the only alternative is to position the verb as the root of all structure, which means one has chosen the path of dependency grammar.

Dependency grammars

The following frameworks are dependency-based:

- Algebraic syntax
- Operator grammar
- Functional generative description
- Lexicase grammar
- Meaning–text theory
- Word grammar
- Extensible dependency grammar^[9]

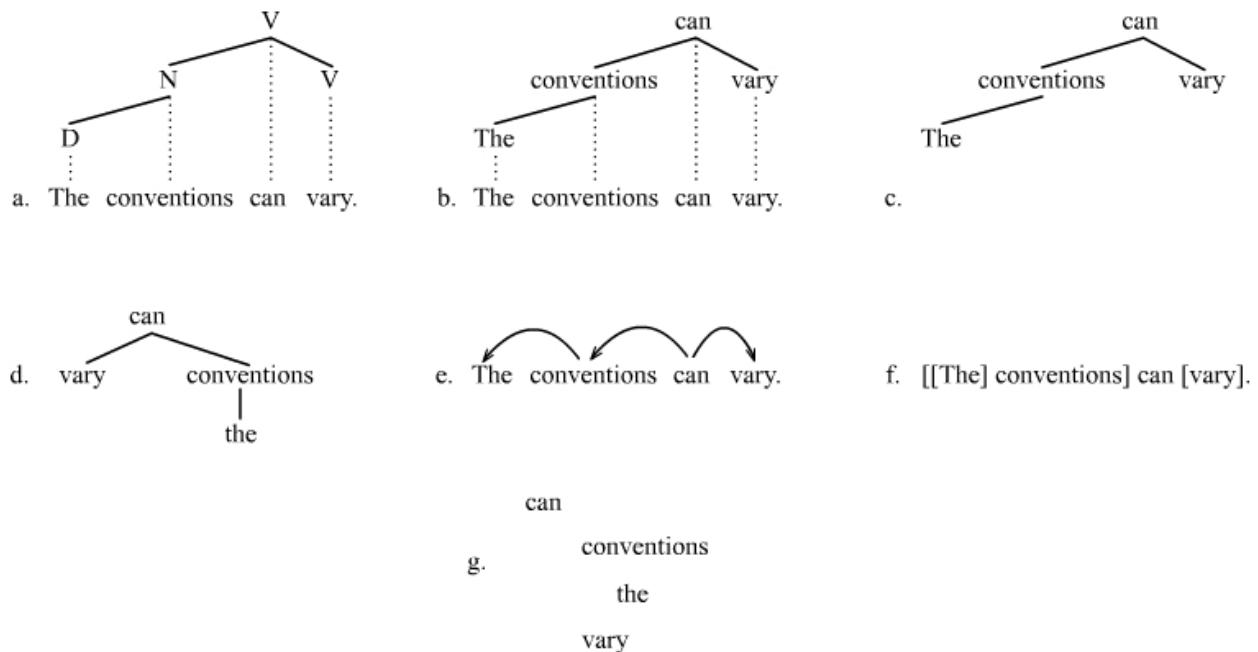
Link grammar is also based on the dependency relation, but link grammar does not include directionality in the dependencies between words, and thus does not describe head-dependent relationships. Hybrid dependency/constituency grammar uses dependencies between words, but also includes dependencies between phrasal nodes – see for example the Quranic Arabic Dependency Treebank. The derivation trees of Tree-adjoining grammar are dependency-based, although the full trees of TAG are constituency-based, so in this regard, it is not clear whether TAG should be viewed more as a dependency or constituency grammar.



There are major differences between the grammars just listed. In this regard, the dependency relation is compatible with other major tenets of theories of grammar. Thus like constituency grammars, dependency grammars can be mono- or multistratal, representational or derivational, construction- or rule-based.

Representing dependencies

There are various conventions that DGs employ to represent dependencies. The following schemata (in addition to the tree above and the trees further below) illustrate some of these conventions:



The representations in (a-d) are trees, whereby the specific conventions employed in each tree vary. Solid lines are *dependency edges* and lightly dotted lines are *projection lines*. The only difference between tree (a) and tree (b) is that tree (a) employs the category class to label the nodes whereas tree (b) employs the words themselves as the node labels.^[10] Tree (c) is a reduced tree insofar as the string of words below and projection lines are deemed unnecessary and are hence omitted. Tree (d) abstracts away from linear order and reflects just hierarchical order.^[11] The arrow arcs in (e) are an alternative convention used to show dependencies and are favored by Word Grammar^[12] The brackets in (f) are seldom used, but are nevertheless quite capable of reflecting the dependency hierarchy; dependents appear enclosed in more brackets than their heads. And finally, the indentations like those in (g) are another convention that is sometimes employed to indicate the hierarchy of words.^[13] Dependents are placed underneath their heads and indented. Like tree (d), the indentations in (g) abstract away from linear order.

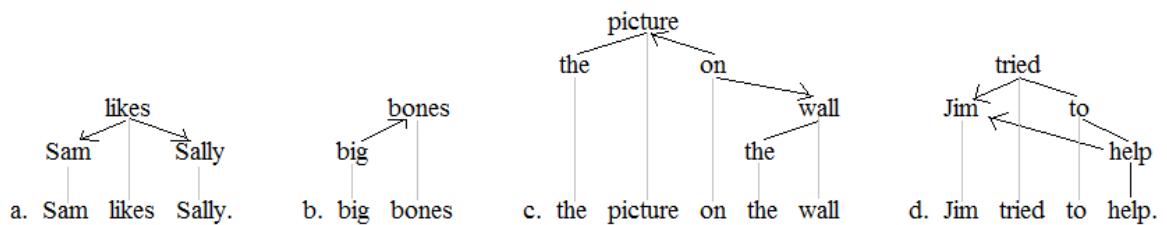
The point to these conventions is that they are just that, namely conventions. They do not influence the basic commitment to dependency as the relation that is grouping syntactic units.

Types of dependencies

The dependency representations above (and further below) show syntactic dependencies. Indeed, most work in dependency grammar focuses on syntactic dependencies. Syntactic dependencies are, however, just one of three or four types of dependencies. Meaning-Text Theory, for instance, emphasizes the role of semantic and morphological dependencies in addition to syntactic dependencies.^[14] A fourth type, prosodic dependencies, can also be acknowledged. Distinguishing between these types of dependencies can be important, in part because if one fails to do so, the likelihood that semantic, morphological, and/or prosodic dependencies will be mistaken for syntactic dependencies is great. The following four subsections briefly sketch each of these dependency types. During the discussion, the existence of syntactic dependencies is taken for granted and used as an orientation point for establishing the nature of the other three dependency types.

Semantic dependencies

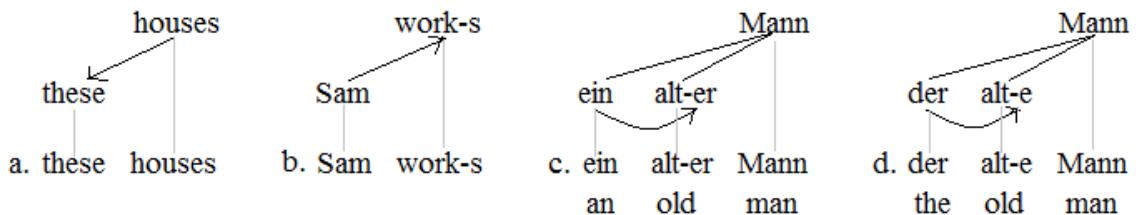
Semantic dependencies are understood in terms of predicates and their arguments.^[15] The arguments of a predicate are semantically dependent on that predicate. Often, semantic dependencies overlap with and point in the same direction as syntactic dependencies. At times, however, semantic dependencies can point in the opposite direction of syntactic dependencies, or they can be entirely independent of syntactic dependencies. The hierarchy of words in the following examples show standard syntactic dependencies, whereas the arrows indicate semantic dependencies:



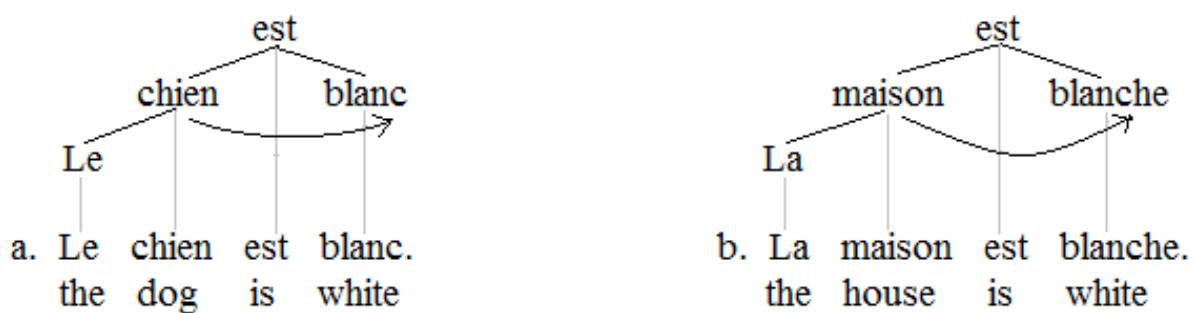
The two arguments *Sam* and *Sally* in tree (a) are dependent on the predicate *likes*, whereby these arguments are also syntactically dependent on *likes*. What this means is that the semantic and syntactic dependencies overlap and point in the same direction (down the tree). Attributive adjectives, however, are predicates that take their head noun as their argument, hence *big* is a predicate in tree (b) that takes *bones* as its one argument; the semantic dependency points up the tree and therefore runs counter to the syntactic dependency. A similar situation obtains in (c), where the preposition predicate *on* takes the two arguments *the picture* and *the wall*; one of these semantic dependencies points up the syntactic hierarchy, whereas the other points down it. Finally, the predicate *to help* in (d) takes the one argument *Jim* but is not directly connected to *Jim* in the syntactic hierarchy, which means that that semantic dependency is entirely independent of the syntactic dependencies.

Morphological dependencies

Morphological dependencies obtain between words or parts of words.^[16] When a given word or part of a word influences the form of another word, then the latter is morphologically dependent on the former. Agreement and concord are therefore manifestations of morphological dependencies. Like semantic dependencies, morphological dependencies can overlap with and point in the same direction as syntactic dependencies, overlap with and point in the opposite direction of syntactic dependencies, or be entirely independent of syntactic dependencies. The arrows are now used to indicate morphological dependencies.



The plural *houses* in (a) demands the plural of the demonstrative determiner, hence *these* appears, not *this*, which means there is a morphological dependency that points down the hierarchy from *houses* to *these*. The situation is reversed in (b), where the singular subject *Sam* demands the appearance of the agreement suffix *-s* on the finite verb *works*, which means there is a morphological dependency pointing up the hierarchy from *Sam* to *works*. The type of determiner in the German examples (c) and (d) influences the inflectional suffix that appears on the adjective *alt*. When the indefinite article *ein* appears, it lacks gender, so the strong masculine ending *-er* appears on the adjective. When the definite article *der* appears, in contrast, it shows masculine gender, which means the weak ending *-e* appears on the adjective. Thus since the choice of determiner impacts the morphological form of the adjective, there is a morphological dependency pointing from the determiner to the adjective, whereby this morphological dependency is entirely independent of the syntactic dependencies. Consider further the following French sentences:



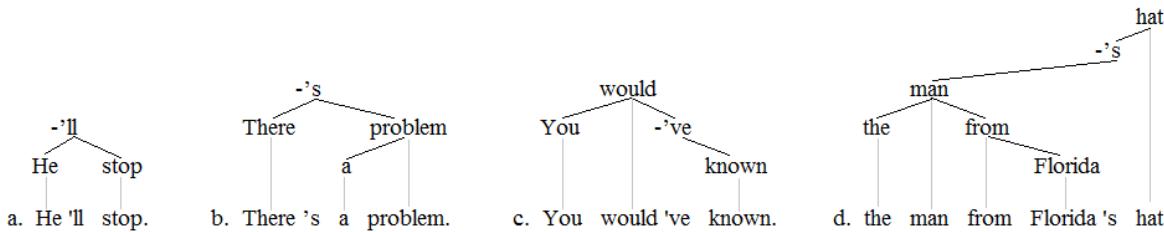
The masculine subject *le chien* in (a) demands the masculine form of the predicative adjective *blanc*, whereas the feminine subject *la maison* demands the feminine form of this adjective. A morphological dependency that is entirely independent of the syntactic dependencies therefore points again across the syntactic hierarchy.

Morphological dependencies play an important role in typological studies. Languages are classified as mostly head-marking (*Sam work-s*) or mostly dependent-marking (*these houses*), whereby most if not all languages contain at least some minor measure of both head and dependent marking.^[17]

Prosodic dependencies

Prosodic dependencies are acknowledged in order to accommodate the behavior of clitics.^[18] A clitic is a syntactically autonomous element that is prosodically dependent on a host. A clitic is therefore integrated into the prosody of its host, meaning that it forms a single word with its host. Prosodic dependencies exist entirely in the linear dimension (horizontal dimension), whereas standard syntactic dependencies exist in the hierarchical dimension (vertical dimension). Classic examples of clitics in English are reduced auxiliaries (e.g. *-ll*, *-s*, *-ve*) and the possessive marker *-s*. The prosodic dependencies in the following examples are indicated with the hyphen and the lack of a

vertical projection line:



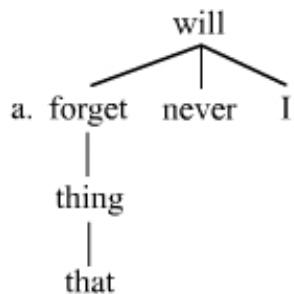
The hyphens and lack of projection lines indicate prosodic dependencies. A hyphen that appears on the left of the clitic indicates that the clitic is prosodically dependent on the word immediately to its left (*He'll*, *There's*), whereas a hyphen that appears on the right side of the clitic (not shown here) indicates that the clitic is prosodically dependent on the word that appears immediately to its right. A given clitic is often prosodically dependent on its syntactic dependent (*He'll*, *There's*) or on its head (*would've*). At other times, it can depend prosodically on a word that is neither its head nor its immediate dependent (*Florida's*).

Syntactic dependencies

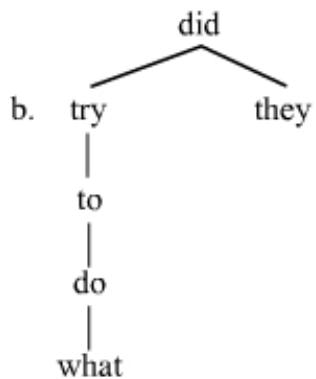
Syntactic dependencies are the focus of most work in dependency grammar, as stated above. How the presence and the direction of syntactic dependencies are determined is of course often open to debate. In this regard, it must be acknowledged that the validity of syntactic dependencies in the trees throughout this article is being taken for granted. However, these hierarchies are such that many dependency grammars can largely support them, although there will certainly be points of disagreement. The basic question about how syntactic dependencies are discerned has proven difficult to answer definitively. One should acknowledge in this area, however, that the basic task of identifying and discerning the presence and direction of the syntactic dependencies of dependency grammars is no easier or harder than determining the constituent groupings of constituency grammars. A variety of heuristics are employed to this end, basic constituency tests being useful tools; the syntactic dependencies assumed in the trees in this article are grouping words together in a manner that most closely matches the results of standard permutation, substitution, and ellipsis constituency tests. Etymological considerations also provide helpful clues about the direction of dependencies. A promising principle upon which to base the existence of syntactic dependencies is distribution.^[19] When one is striving to identify the root of a given phrase, the word that is most responsible for determining the distribution of that phrase as a whole is the root of that phrase.

Linear order and discontinuities

Traditionally, DGs have had a different approach to linear order (word order) than constituency grammars. Dependency-based structures are minimal compared to their constituency-based counterparts, and these minimal structures allow one to focus intently on the two ordering dimensions.^[20] Separating the vertical dimension (hierarchical order) from the horizontal dimension (linear order) is easily accomplished. This aspect of dependency-based structures has allowed DGs, starting with Tesnière (1959), to focus on hierarchical order in a manner that is hardly possible for constituency grammars. For Tesnière, linear order was secondary to hierarchical order insofar as hierarchical order preceded linear order in the mind of a speaker. The stemmas (trees) that Tesnière produced reflected this view; they abstracted away from linear order to focus almost entirely on hierarchical order. Many DGs that followed Tesnière adopted this practice, that is, they produced tree structures that reflect hierarchical order alone, e.g.



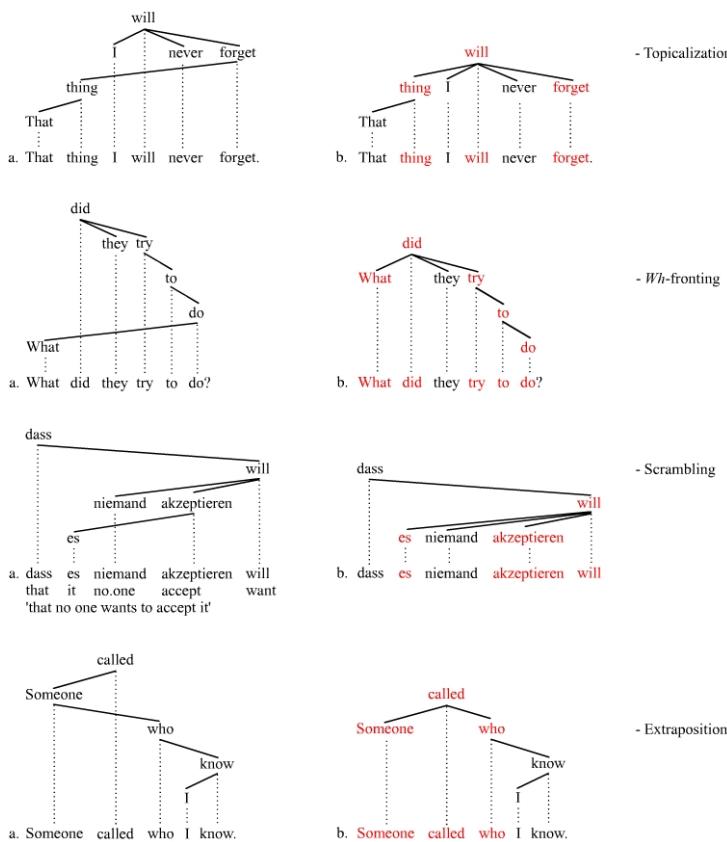
- Unordered tree of the sentence *That thing I will never forget.*



- Unordered tree of the sentence *What did they try to do?.*

The traditional focus on hierarchical order generated the impression that DGs have little to say about linear order, and it has contributed to the view that DGs are particularly well-suited to examine languages with free word order. A negative result of this focus on hierarchical order, however, is that there is a dearth of dependency-based explorations of particular word order phenomena, such as of standard discontinuities. Comprehensive dependency grammar accounts of topicalization, *wh*-fronting, scrambling, and extraposition are mostly absent from many established dependency-based frameworks. This situation can be contrasted with constituency grammars, which have devoted tremendous effort to exploring these phenomena.

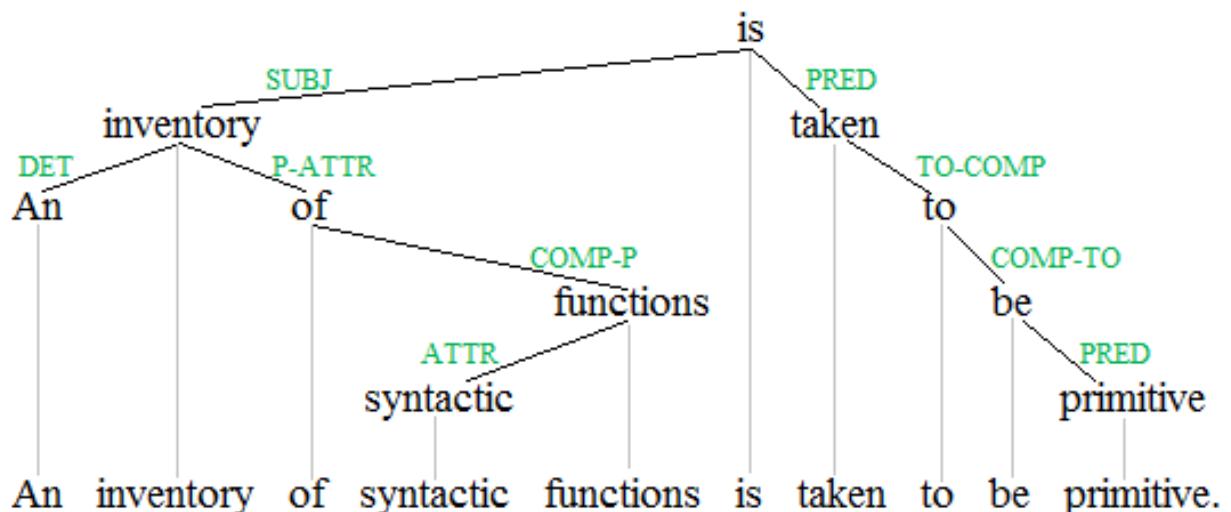
The nature of the dependency relation does not, however, prevent one from focusing on linear order. Dependency-based structures are as capable of exploring word order phenomena as constituency-based structures. The following trees illustrate this point; they represent one way of exploring discontinuities using dependency-based structures. The trees suggest the manner in which common discontinuities can be addressed. An example from German is used to illustrate a scrambling discontinuity:



The a-trees on the left show projectivity violations (= crossing lines), and the b-trees on the right demonstrate one means of addressing these violations. The displaced constituent takes on a word as its head that is not its governor. The words in red mark the catena (=chain) of words that extends from the root of the displaced constituent to the governor of that constituent.^[21] Discontinuities are then explored in terms of these catenae. The limitations on topicalization, wh-fronting, scrambling, and extraposition can be explored and identified by examining the nature of the catenae involved.

Syntactic functions

Traditionally, DGs have treated the syntactic functions (= grammatical functions, grammatical relations) as primitive. They posit an inventory of functions (e.g. subject, object, oblique, determiner, attribute, predicative, etc.). These functions can appear as labels on the dependencies in the tree structures, e.g.^[22]



The syntactic functions in this tree are shown in green: ATTR (attribute), COMP-P (complement of preposition), COMP-TO (complement of to), DET (determiner), P-ATTR (prepositional attribute), PRED (predicative), SUBJ (subject), TO-COMP (to complement). The functions chosen and abbreviations used in the tree here are merely representative of the general stance of DGs toward the syntactic functions. The actual inventory of functions and designations employed vary from DG to DG.

As a primitive of the theory, the status of these functions is much different than in some constituency grammars. Traditionally, constituency grammars derive the syntactic functions from the constellation. For instance, the object is identified as the NP appearing inside finite VP, and the subject as the NP appearing outside of finite VP. Since DGs reject the existence of a finite VP constituent, they were never presented with the option to view the syntactic functions in this manner. The issue is a question of what comes first: traditionally, DGs take the syntactic functions to be primitive and they then derive the constellation from these functions, whereas constituency grammars traditionally take the constellation to be primitive and they then derive the syntactic functions from the constellation.

This question about what comes first (the functions or the constellation) is not an inflexible matter. The stances of both grammar types (dependency and constituency grammars) is not narrowly limited to the traditional views. Dependency and constituency are both fully compatible with both approaches to the syntactic functions. Indeed, monostratal systems, be they dependency- or constituency-based, will likely reject the notion that the functions are derived from the constellation or that the constellation is derived from the functions. They will take both to be primitive, which means neither can be derived from the other.

Notes

- [1] Concerning the history of the dependency concept, see Percival (1990).
- [2] Concerning the influence of term logic on the theory of grammar, see Percival (1976).
- [3] Concerning early dependency grammars that may have developed independently of Tesnière's work, see for instance Hays (1960), Gaifman (1965), and Robinson (1970).
- [4] Some prominent dependency grammars that were well established by the 1980s are from Hudson (1984), Sgall, Hajičová et Paněvová (1986), Mel'čuk (1988), and Starosta (1988).
- [5] Some prominent dependency grammars from the German schools are from Heringer (1996), Engel (1994), Eroms (2000), and Ágel et al. (2003/6) is a massive two volume collection of essays on dependency and valence grammars from more than 100 authors.
- [6] <http://depling.org>
- [7] The minimality of dependency structures is emphasized, for instance, by Osborne et al. (2011).
- [8] Concerning Tesnière's rejection of the subject-predicate division of the clause, see Tesnière (1959:103–105), and for discussion of empirical considerations that support Tesnière's point, see Matthews (2007:17ff.), Miller (2011:54ff.), and Osborne et al. (2011:323f.).
- [9] <http://www.ps.uni-saarland.de/~rade/xdg.html>
- [10] The conventions illustrated with trees (a) and (b) are preferred by Osborne et al. (2011, 2013).
- [11] Unordered trees like (d) are associated above all with Tesnière's stemmas and with the syntactic strata of Mel'čuk's Meaning-Text Theory.
- [12] Three major works on Word Grammar are Hudson (1984, 1990, 2007).
- [13] Lobin (2003) makes heavy use of he indentations.
- [14] For a discussion of semantic, morphological, and syntactic dependencies in Meaning-Text Theory, see Mel'čuk (2003:191ff.).
- [15] Concerning semantic dependencies, see Mel'čuk (2003:192f.).
- [16] Concerning morphological dependencies, see Mel'čuk (2003:193ff.).
- [17] The distinction between head- and dependent-marking was established by Nichols (1986). Nichols was using a dependency-based understanding of these distinctions.
- [18] Concerning prosodic dependencies and the analysis of clitics, see Groß (2011).
- [19] Distribution is primary principle used by Owens (1984:36), Schubert (1988:40), and Mel'čuk (2003:200) for discerning syntactic dependencies.
- [20] Concerning the importance of the two ordering dimensions, see Tesnière (1959:16ff.).
- [21] See Osborne et al. (2013) concerning catenae.
- [22] For discussion and examples of the labels for syntactic functions that are attached to dependency edges and arcs, see for instance Mel'čuk (1988:22, 69) and van Valin (2001:102ff.).

References

- Ágel, Vilmos; Eichinger, Ludwig M.; Eroms, Hans Werner et al., eds. (2003) (in German). *Dependenz und Valenz: Ein internationales Handbuch der zeitgenössischen Forschung [Dependency and Valency: An International Handbook of Contemporary Research]* (<http://www.degruyter.com/view/serial/18595>). Berlin: de Gruyter. ISBN 978-3110141900. Retrieved 24 August 2012.
- Engel, U. 1994. Syntax der deutschen Sprache, 3rd edition. Berlin: Erich Schmidt Verlag.
- Eroms, Hans-Werner (2000). *Syntax der deutschen Sprache* (<http://www.degruyter.com/view/product/5087?format=G>). Berlin [u.a.]: de Gruyter. ISBN 978-3110156669. Retrieved 24 August 2012.
- Groß, T. 2011. Clitics in dependency morphology. Depling 2011 Proceedings, 58-68.
- Helbig, Gerhard; Buscha, Joachim (2007). *Deutsche Grammatik: ein Handbuch für den Ausländerunterricht [German Grammar: A Guide for Foreigners Teaching]* (http://www.langenscheidt.de/produkt/2881_8731/Deutsche_Grammatik-Buch/978-3-468-49493-2) (6. [Dr.]. ed.). Berlin: Langenscheidt. ISBN 978-3-468-49493-2. Retrieved 24 August 2012.
- Heringer, H. 1996. Deutsche Syntax dependentiell. Tübingen: Stauffenburg.
- Hays, D. 1960. Grouping and dependency theories. P-1910, RAND Corporation.
- Hudson, Richard (1984). *Word grammar* (1. publ. ed.). Oxford, OX, England: B. Blackwell. ISBN 978-0631131861.
- Hudson, R. 1990. An English Word Grammar. Oxford: Basil Blackwell.
- Hudson, R. 2007. Language Networks: The New Word Grammar. Oxford University Press.
- Liu, H. 2009. Dependency Grammar: from Theory to Practice. Beijing: Science Press.
- Lobin, H. 2003. Koordinationssyntax als prozedurales Phänomen. Tübingen: Gunter Narr-Verlag.
- Matthews, P. H. (2007). *Syntactic Relations: a critical survey* (<http://www.cambridge.org/us/knowledge/isbn/item1157046>) (1. publ. ed.). Cambridge: Cambridge University Press. ISBN 9780521608299. Retrieved 24 August 2012.
- Mel'čuk, Igor A. (1987). *Dependency syntax : theory and practice* (<http://www.sunypress.edu/p-164-dependency-syntax.aspx>). Albany: State University Press of New York. ISBN 978-0-88706-450-0. Retrieved 24 August 2012.
- Mel'čuk, I. 2003. Levels of dependency in linguistic description: Concepts and problems. In Ágel et al., 170-187.
- Miller, J. 2011. A critical introduction to syntax. London: continuum.
- Nichols, J. 1986. Head-marking and dependent-marking languages. *Language* 62, 56-119.
- Osborne, T., M. Putnam, and T. Groß 2011. Bare phrase structure, label-less trees, and specifier-less syntax: Is Minimalism becoming a dependency grammar? *The Linguistic Review* 28, 315–364.
- Osborne, T., M. Putnam, and T. Groß 2013. Catenae: Introducing a novel unit of syntactic analysis. *Syntax* 16, in press.
- Owens, J. 1984. On getting a head: A problem in dependency grammar. *Lingua* 66, 25-42.
- Percival, K. 1976. On the historical source of immediate-constituent analysis. In: Notes from the linguistic underground, James McCawley (ed.), *Syntax and Semantics* 7, 229–242. New York: Academic Press.
- Percival, K. 1990. Reflections on the history of dependency notions in linguistics. *Historiographia Linguistica*, 17, 29–47.
- Robinson, J. 1970. Dependency structures and transformational rules. *Language* 46, 259–285.
- Schubert, K. 1988. Metataxis: Contrastive dependency syntax for machine translation. Dordrecht: Foris.
- Sgall, P., E. Hajíčová, and J. Panevová 1986. The meaning of the sentence in its semantic and pragmatic aspects. Dordrecht: D. Reidel Publishing Company.
- Starosta, S. 1988. The case for lexicase. London: Pinter Publishers.
- Tesnière, L. 1959. *Éléments de syntaxe structurale*. Paris: Klincksieck.
- Tesnière, L. 1969. *Éléments de syntaxe structurale*, 2nd edition. Paris: Klincksieck.
- van Valin, R. 2001. An introduction to syntax. Cambridge, UK: Cambridge University Press.

Implementations

- DeSR (<http://sites.google.com/site/desrparser/>) A statistical shift/reduce dependency parser
- MaltParser (<http://maltparser.org/>) A system for data-driven dependency parsing
- MST Parser (<http://sourceforge.net/projects/mstparser/>) A non-projective dependency parser that searches for maximum spanning trees over directed graphs
- MST Parser (C#) (<https://github.com/rasoolims/MSTParserCSharp/>) A non-projective dependency parser that searches for maximum spanning trees over directed graphs (C# conversion of the Java code)
- RelEx (<http://wiki.opencog.org/w/RelEx>) A parser that generates a dependency parse for the English language, by applying graph rewriting to the output of the link grammar parser. Open source license
- ClearParser (<http://code.google.com/p/clearparser/>) A statistical, transition-based dependency parser.
- Stanford parser (<http://nlp.stanford.edu/software/lex-parser.shtml>) A statistical phrase-structure parser which provides a tool to convert the output into a form of dependency graph called "Stanford Dependencies"
- TULE (<http://www.parsit.it/tule.php>) A linguistic framework that takes a natural language sentence in input (Italian) and returns a full dependency tree describing its syntactic structure
- XDG Development Kit (<http://www.ps.uni-saarland.de/~rade/mogul/publish/doc/debusmann-xdk/>) An integrated development environment for Extensible Dependency Grammar (XDG)

External links

- Link Grammar online demonstration (<http://www.link.cs.cmu.edu/link/submit-sentence-4.html>)
- Extensible Dependency Grammar articles and grammar development kit (<http://www.ps.uni-sb.de/~rade/xdg.html>)
- Prague Dependency Treebank (<http://ufal.mff.cuni.cz/pdt2.0/>)
- Persian Dependency Treebank (<http://dadegan.ir/en>)
- Quranic Arabic Dependency Treebank (<http://corpus.quran.com>)
- Word Grammar (<http://www.phon.ucl.ac.uk/home/dick/wg.htm>)
- Depling 2011: The first International Conference on Dependency Linguistics (<http://depling.org>)

Recursive categorical syntax

Recursive categorical syntax, also sometimes called **algebraic syntax**, is an algebraic theory of syntax developed by Michael Brame as an alternative to transformational-generative grammar.

References

- Brame, Michael. "Universal Word Induction vs Move α " in *Linguistic Analysis*, Vol. 14, No. 4, 1984.
- Brame, Michael. "Recursive Categorical Syntax I: Semigroups, Monoids, Lattices, and Categories" in *Linguistic Analysis*, Vol. 14, No. 1.
- Brame, Michael. "Recursive Categorical Syntax II: n -arity and Variable Continuation" in *Linguistic Analysis*, Vol. 15, No. 2-3, 1985.
- Brame, Michael. "Recursive Categorical Syntax III: dl-Induction" in *Linguistic Analysis*.

Operator grammar

Operator grammar is a mathematical theory of human language that explains how language carries information. This theory is the culmination of the life work of Zellig Harris, with major publications toward the end of the last century. Operator Grammar proposes that each human language is a self-organizing system in which both the syntactic and semantic properties of a word are established purely in relation to other words. Thus, no external system (metalanguage) is required to define the rules of a language. Instead, these rules are learned through exposure to usage and through participation, as is the case with most social behavior. The theory is consistent with the idea that language evolved gradually, with each successive generation introducing new complexity and variation.

Operator Grammar posits three universal constraints: dependency (certain words depend on the presence of other words to form an utterance), likelihood (some combinations of words and their dependents are more likely than others) and reduction (words in high likelihood combinations can be reduced to shorter forms, and sometimes omitted completely). Together these provide a theory of language information: dependency builds a predicate–argument structure; likelihood creates distinct meanings; reduction allows compact forms for communication.

Dependency

The fundamental mechanism of operator grammar is the dependency constraint: certain words (operators) require that one or more words (arguments) be present in an utterance. In the sentence *John wears boots*, the operator *wears* requires the presence of two arguments, such as *John* and *boots*. (This definition of dependency differs from other dependency grammars in which the arguments are said to depend on the operators.)

In each language the dependency relation among words gives rise to syntactic categories in which the allowable arguments of an operator are defined in terms of their dependency requirements. Class N contains words (e.g. *John*, *boots*) that do not require the presence of other words. Class O_N contains the words (e.g. *sleeps*) that require exactly one word of type N. Class O_{NN} contains the words (e.g. *wears*) that require two words of type N. Class O_{OO} contains the words (e.g. *because*) that require two words of type O, as in *John stumbles because John wears boots*. Other classes include O_O (*is possible*), O_{NNN} (*put*), O_{ON} (*with, surprise*), O_{NO} (*know*), O_{NNO} (*ask*) and O_{NOO} (*attribute*).

The categories in operator grammar are universal and are defined purely in terms of how words relate to other words, and do not rely on an external set of categories such as noun, verb, adjective, adverb, preposition, conjunction, etc. The dependency properties of each word are observable through usage and therefore learnable.

Likelihood

The dependency constraint creates a structure (syntax) in which any word of the appropriate class can be an argument for a given operator. The likelihood constraint places additional restrictions on this structure by making some operator/argument combinations more likely than others. Thus, *John wears hats* is more likely than *John wears snow* which in turn is more likely than *John wears vacation*. The likelihood constraint creates meaning (semantics) by defining each word in terms of the words it can take as arguments, or of which it can be an argument.

Each word has a unique set of words with which it has been observed to occur called its selection. The **coherent selection** of a word is the set of words for which the dependency relation has above average likelihood. Words that are similar in meaning have similar coherent selection. This approach to meaning is self-organizing in that no external system is necessary to define what words mean. Instead, the meaning of the word is determined by its usage within a population of speakers. Patterns of frequent use are observable and therefore learnable. New words can be introduced at any time and defined through usage.

Reduction

The reduction constraint acts on high likelihood combinations of operators and arguments and makes more compact forms. Certain reductions allow words to be omitted completely from an utterance. For example, *I expect John to come* is reducible to *I expect John*, because *to come* is highly likely under *expect*. The sentence *John wears boots and John wears hats* can be reduced to *John wears boots and hats* because repetition of the first argument *John* under the operator *and* is highly likely. *John reads things* can be reduced to *John reads*, because the argument *things* has high likelihood of occurring under any operator.

Certain reductions reduce words to shorter forms, creating pronouns, suffixes and prefixes (morphology). *John wears boots and John wears hats* can be reduced to *John wears boots and he wears hats*, where the pronoun *he* is a reduced form of *John*. Suffixes and prefixes can be obtained by appending other freely occurring words, or variants of these. *John is able to be liked* can be reduced to *John is likeable*. *John is thoughtful* is reduced from *John is full of thought*, and *John is anti-war* from *John is against war*.

Modifiers are the result of several of these kinds of reductions, which give rise to adjectives, adverbs, prepositional phrases, subordinate clauses, etc.

1. *John wears boots; the boots are of leather* (two sentences joined by semicolon operator) →
2. *John wears boots which are of leather* (reduction of repeated noun to relative pronoun) →
3. *John wears boots of leather* (omission of high likelihood phrase *which are*) →
4. *John wears leather boots* (omission of high likelihood operator *of*, transposition of short modifier to left of noun)

Each language has a unique set of reductions. For example, some languages have morphology and some don't; some transpose short modifiers and some do not. Each word in a language participates only in certain kinds of reductions. However, in each case, the reduced material can be reconstructed from knowledge of what is likely in the given operator/argument combination. The reductions in which each word participates are observable and therefore learnable, just as one learns a word's dependency and likelihood properties.

Information

The importance of reductions in operator grammar is that they separate sentences that contain reduced forms from those that don't (base sentences). All reductions are paraphrases, since they do not remove any information, just make sentences more compact. Thus, the base sentences contain all the information of the language and the reduced sentences are variants of these. Base sentences are made up of simple words without modifiers and largely without affixes, e.g. *snow falls*, *sheep eat grass*, *John knows sheep eat grass*, *that sheep eat snow surprises John*.

Each operator in a sentence makes a contribution in information according to its likelihood of occurrence with its arguments. Highly expected combinations have low information; rare combinations have high information. The

precise contribution of an operator is determined by its selection, the set of words with which it occurs with high frequency. The arguments *boots*, *hats*, *sheep*, *grass* and *snow* differ in meaning according to the operators for which they can appear with high likelihood in first or second argument position. For example, *snow* is expected as first argument of *fall* but not of *eat*, while the reverse is true of *sheep*. Similarly, the operators *eat*, *devour*, *chew* and *swallow* differ in meaning to the extent that the arguments they select and the operators that select them differ.

Operator grammar predicts that the information carried by a sentence is the accumulation of contributions of each argument and operator. The increment of information that a given word adds to a new sentence is determined by how it was used before. In turn, new usages stretch or even alter the information content associated with a word. Because this process is based on high frequency usage, the meanings of words are relatively stable over time, but can change in accordance with the needs of a linguistic community.

Bibliography

- Harris, Zellig (1982), *A Grammar of English on Mathematical Principles*, New York: John Wiley and Sons, ISBN 0-471-02958-0
- Harris, Zellig (1988), *Language and Information*, New York: Columbia University Press, ISBN 0-231-06662-7
- Harris, Zellig (1989), *The Form of Information in Science: Analysis of an immunology sublanguage*, Springer, ISBN 90-277-2516-0
- Harris, Zellig (1991), *A Theory of Language and Information: A Mathematical Approach*, Oxford University Press, USA, ISBN 0-19-824224-7

Functional generative description

Functional generative description (FGD) is a linguistic framework developed at Charles University in Prague since the 1960s by a team led by Petr Sgall. Based on the dependency grammar formalism, it is a stratificational grammar formalism that treats the sentence as a system of interlinked layers: phonological, morphematical, morphonological, analytical (surface syntax) and tectogrammatical (deep syntax). Continuing the tradition of Prague School, special attention is paid to the phenomenon of topic–focus articulation.

The Prague Dependency Treebank (PDT) is a treebank consisting of a subset of the Czech National Corpus annotated along the lines of FGD.

External links

- Prague Dependency Treebank 2.0^[1]

References

- Sgall, P., Hajičová, E., Panevová, J. (1986). *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects*. Dordrecht: D. Reidel Publishing Company. ISBN 90-277-1838-5.

References

[1] <http://ufal.mff.cuni.cz/pdt2.0/>

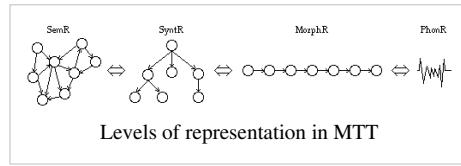
Meaning–text theory

Meaning–text theory (MTT) is a theoretical linguistic framework, first put forward in Moscow by Aleksandr Žolkovskij and Igor Mel'čuk,^[1] for the construction of models of natural language. The theory provides a large and elaborate basis for linguistic description and, due to its formal character, lends itself particularly well to computer applications, including machine translation, phraseology, and lexicography. General overviews of the theory can be found in Mel'čuk (1981)^[2] and (1988).^[3]

Levels of representation

Linguistic models in MTT operate on the principle that language consists in a mapping from the content or meaning (semantics) of an utterance to its form or text (phonetics). Intermediate between these poles are additional levels of representation at the syntactic and morphological levels.

Representations at the different levels are mapped, in sequence, from the unordered network of the semantic representation (SemR) through the dependency tree-structures of the Syntactic Representation (SyntR) to a linearized chain of morphemes of the Morphological Representation (MorphR) and, ultimately, the temporally-ordered string of phones of the Phonetic Representation (PhonR) (not generally addressed in work in this theory). The relationships between representations on the different levels are considered to be translations or mappings, rather than transformations, and are mediated by sets of rules, called "components", which ensure the appropriate, language-specific transitions between levels.



Semantic representation

Semantic representations (SemR) in meaning–text theory consist primarily of a web-like semantic structure (SemS) which combines with other semantic-level structures (most notably the Semantic-Communicative Structure [SemCommS],^[4] which represents what is commonly referred to as "Information Structure" in other frameworks). The SemS itself consists of a network of predication, represented as nodes with arrows running from predicate nodes to argument node(s). Arguments can be shared by multiple predicates, and predicates can themselves be arguments of other predicates. Nodes generally correspond to lexical and grammatical meanings as these are directly expressed by items in the lexicon or by inflectional means, but the theory allows the option of decomposing meanings into more fine-grained representation via processes of semantic paraphrasing,^[5] which are also key to dealing with synonymy and translation-equivalencies between languages. SemRs are mapped onto the next level of representation, the Deep-Syntactic Representation, by the rules of the Semantic Component, which allow for a one to many relationship between levels (that is, one SemR can potentially be expressed by a variety of syntactic structures, depending on lexical choice, the complexity of the SemR, etc.).

Syntactic representation

Syntactic representations in MTT are implemented using dependency trees, which constitute the Syntactic Structure (SyntS). SyntS is accompanied by various other types of structure, most notably the syntactic communicative structure and the anaphoric structure. There are two levels of syntax in MTT, the deep syntactic representation (DSyntR) and the surface syntactic representation (SSyntR). A good overview of MTT syntax, including its descriptive application, can be found in Mel'čuk (1988).^[6] A comprehensive model of English surface syntax is presented in Mel'čuk & Pertsov (1987).^[7]

The deep syntactic representation (DSyntR) is related directly to SemS and seeks to capture the "universal" aspects of the syntactic structure. Trees at this level represent dependency relations between lexemes (or between lexemes

and a limited inventory of abstract entities such as lexical functions). Deep syntactic relations between lexemes at DSyntR are restricted to a universal inventory of a dozen or syntactic relations including seven ranked actantial (argument) relations, the modificative relation, and the coordinative relation. Lexemes with purely grammatical function such as lexically-governed prepositions are not included at this level of representation; values of inflectional categories that are derived from SemR but implemented by the morphology are represented as subscripts on the relevant lexical nodes that they bear on. DSyntR is mapped onto the next level of representation by rules of the deep-syntactic component.

The surface-syntactic representation (SSyntR) represents the language-specific syntactic structure of an utterance and includes nodes for all the lexical items (including those with purely grammatical function) in the sentence. Syntactic relations between lexical items at this level are not restricted and are considered to be completely language-specific, although many are believed to be similar (or at least isomorphic) across languages. SSyntR is mapped onto the next level of representation by rules of the surface-syntactic component.

Morphological representation

Morphological Representations in MTT are implemented as strings of morphemes arranged in a fixed linear order reflecting the ordering of elements in the actual utterance. It should be noted that this is the first representational level at which linear precedence is considered to be linguistically significant, effectively grouping word-order together with morphological processes and prosody, as one of the three non-lexical means with which languages can encode syntactic structure. As with Syntactic Representation, there are two levels of Morphological Representation—Deep and Surface Morphological Representation. Detailed descriptions of MTT Morphological Representations are found in Mel'čuk (1993–2000)^[8] and Mel'čuk (2006).^[9]

Deep Morphological Representation (DMorphR) consists of strings of lexemes and morphemes—e.g., THE SHOE+{PL} ON BILL+{POSS} FOOT+{PL}. The deep morphological component of rules maps this string onto the Surface Morphological Representation (SMorphR), converting morphemes into the appropriate morphs and performing morphological operations implementing non-concatenative morphological processes—in the case of our example above, giving us /the shoe+s on Bill+s feet/. Rules of the surface morphological component, a subset of which include morphophonemic rules, map the SMorphR onto a phonetic representation [ðə Juz an bɪlz fit].

The lexicon

A crucial aspect of MTT is the lexicon, considered to be a comprehensive catalogue of the lexical units (LUs) of a language, these units being the lexemes, collocations and other phrasemes, constructions, and other configurations of linguistic elements that are learned and implemented in speech by users of language. The lexicon in MTT is represented by the Explanatory Combinatorial Dictionary (ECD)^{[10][11]} which includes entries for all of the LUs of a language along with information speakers must know regarding their syntaxics (the LU-specific rules and conditions on their combinatorics). An ECD for Russian was produced by Mel'čuk et al. (1984),^[12] and ECDs for French were published as Mel'čuk et al. (1999)^[13] and Mel'čuk & Polguère (2007).^[14]

Lexical functions

One important discovery of meaning–text linguistics was the recognition that LUs in a language can be related to one another in an abstract semantic sense and that this same relation also holds across many lexically-unrelated pairs or sets of LUs. These relations are represented in MTT as lexical functions (LF).^[15] An example of a simple LF is Magn(L), which represents collocations used in intensification such as *heavy rain*, *strong wind*, or *intense bombardment*. A speaker of English knows that for a given lexeme L such as RAIN the value of Magn(RAIN) = HEAVY, whereas Magn(WIND) = STRONG, and so on. MTT currently recognizes several dozen standard LFs that are known to recur across languages.

References

- [1] Žolkovskij, Aleksandr K.; Igor A. Mel'čuk (1965). "O vozmožnom metode i instrumentax semantičeskogo sinteza (On a possible method and instruments for semantic synthesis)". *Naučno-texničeskaja informacija* 5: 23–28.
- [2] Mel'čuk, Igor A. (1981). "Meaning-Text Models: A recent trend in Soviet linguistics". *Annual Review of Anthropology* 10: 27–62.
- [3] Mel'čuk, Igor A. (1988). *Dependency syntax: Theory and practice*. Albany, NY: SUNY Press.
- [4] Mel'čuk, Igor A. (2001). *Communicative organization in natural language: The semantic-communicative structure of sentences*. Amsterdam: John Benjamins.
- [5] Milićević, Jasmina (2007). *La paraphrase. Modélisation de la paraphrase langagièrre*. Bern: Peter Lang.
- [6] Mel'čuk, Igor A. (1988). *Dependency syntax: Theory and practice*. Albany, NY: SUNY Press.
- [7] Mel'čuk, Igor A.; Nikolai V. Pertsov (1987). *Surface syntax of English: A formal model within the Meaning-Text framework*. Amsterdam: John Benjamins.
- [8] Mel'čuk, Igor A. (1993–2000). *Cours de morphologie générale*. Montréal: Les Presses de l'Université de Montréal.
- [9] Mel'čuk, Igor A. (2006). *Aspects of the Theory of Morphology*. Berlin: Mouton de Gruyter.
- [10] Mel'čuk, Igor A.; Andre Clas, Alain Polguère (1995). *Introduction à la lexicologie explicative et combinatoire*. Paris: Duculot.
- [11] Mel'čuk, Igor A. (2006). Sica, G. ed. "Explanatory combinatorial dictionary". *Open Problems in Linguistics and Lexicography* (Monza: Polimetrica): 222–355.
- [12] Mel'čuk, Igor A.; Aleksandr K. Žolkovsky, Juri Apresjan (1984). *Толково-комбинаторный словарь современного русского языка: Опыты семантико-синтаксического описания русской лексики*. [Explanatory Combinatorial Dictionary of Modern Russian: Semantico-Syntactic Studies of Russian Vocabulary]. Wiener Slawistischer Almanach: Vienna.
- [13] Mel'čuk, Igor A.; N. Arbatchewsky-Jumarie, Lida Iordanskaja, S. Mantha & Alain Polguère (1999). *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques IV*. Montréal: Les Presses de l'Université de Montréal.
- [14] Mel'čuk, Igor A.; Alain Polguère (2007). *Lexique actif du français : L'apprentissage du vocabulaire fondé sur 20000 dérivations sémantiques et collocations du français*. Paris: Duculot.
- [15] Mel'čuk, Igor A. (1996). Wanner, Leo. ed. "Lexical functions: a tool for the description of lexical relations in a lexicon". *Lexical Functions in Lexicography and Natural Language Processing*: 37–102.

Further readings

General

- Žolkovskij, A.K. and Mel'čuk, Igor A. (1965). O vozmožnom metode i instrumentax semantičeskogo sinteza (On a possible method and instruments for semantic synthesis). *Naučno-texničeskaja informacija* 5, 23–28.
- И. А. Мельчук. Опыт теории лингвистических моделей «Смысл ↔ Текст». М., 1974 (2nd ed., 1999).
- И. А. Мельчук. Русский язык в модели «Смысл ↔ Текст». Москва-Вена, 1995.
- I. A. Mel'čuk. Vers une linguistique Sens-Texte. Leçon inaugurale. P.: Collège de France, Chaire internationale, 1997.
- Leo Wanner (ed.), *Recent Trends in Meaning-Text Theory*. Amsterdam, Philadelphia: J. Benjamins Pub., 1997. ISBN 1-55619-925-2, ISBN 90-272-3042-0
- I.A. Bolshakov, A.F. Gelbukh. The Meaning-Text Model: Thirty Years After (<http://gelbukh.com/CV/Publications/2000/Forum-MTM-eng.htm>) J. International Forum on Information and Documentation, FID 519, ISSN 0304-9701, N 1, 2000.

Syntax

- И. А. Мельчук. Поверхностный синтаксис русских числовых выражений. Wien: Wiener Slawistischer Almanach, 1985.
- I. A. Mel'čuk & N. V. Pertsov. Surface syntax of English: A formal model within the Meaning-Text framework. Amsterdam, Philadelphia: Benjamins, 1987. ISBN 90-272-1515-4
- I. A. Mel'čuk. Dependency syntax: Theory and practice. Albany, NY: SUNY, 1988. ISBN 0-88706-450-7, ISBN 0-88706-451-5
- I. A. Mel'čuk. Actants in Semantics and Syntax. I,II, *Linguistics*, 2004, 42:1, 1–66; 42:2, 247—291.

Morphology

- I. A. Mel'čuk. Cours de morphologie générale, vol. 1–5. Montréal: Les Presses de l'Université de Montréal/Paris: CNRS Éditions, 1993—2000
- I. A. Mel'čuk. Aspects of the Theory of Morphology. Berlin; New York: Mouton de Gruyter, 2006. ISBN 3-11-017711-0

Lexicography

- I. A. Mel'čuk, A. K. Zholkovsky, Ju. D. Apresjan et al. Explanatory Combinatorial Dictionary of Modern Russian: Semantico-Syntactic Studies of Russian Vocabulary / Толково-комбинаторный словарь современного русского языка: Опыты семантико-синтаксического описания русской лексики. Wien: Wiener Slawistischer Almanach, 1984.
- I. A. Mel'čuk, A. Clas & A. Polguère. Introduction à la lexicologie explicative et combinatoire. P.: Duculot, 1995. — ISBN 2-8011-1106-6
- I. A. Mel'čuk et al. Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques IV, Montréal: Les Presses de l'Université de Montréal, 1999. — ISBN 2-7606-1738-6

External links

- The Meaning-Text Theory web site (<http://meaningtext.net/>), hosts the proceedings of the biannual MTT-conference
- Observatoire de linguistique Sens-Texte (OLST) (<http://www.olst.umontreal.ca/>)
- Meaning-Text @ neuvel.net (<http://www.neuvel.net/meaningtext.htm>), an excellent introduction to the theory
- Meaning-Text on-line library (<http://www.zotero.org/groups/mtt/items>)

Meaning-text software

- Semantic search engine (<http://www.inbenta.com/index.php/en/meaning-text-theory.html>) based on Meaning-Text theory provided by Inbenta (<http://www.inbenta.com>)
- Carabao Language Kit (<http://www.digitalsongata.com/faq.aspx>) provided by Digital Sonata (<http://www.digitalsongata.com>)
- ETAP-3 Linguistic Processing System, described as 'a Full-Fledged NLP Implementation of the Meaning-Text Theory' (official site) (<http://cl.iitp.ru/etap>)

Word grammar

Word grammar has been developed by Richard Hudson since the 1980s. It started as a model of syntax, whose most distinctive characteristic is its use of dependency grammar, an approach to syntax in which the sentence's structure is almost entirely contained in the information about individual words, and syntax is seen as consisting primarily of principles for combining words. The central syntactic relation is that of dependency between words; constituent structure is not recognized except in the special case of coordinate structures.

However an even more important claim of Word Grammar is that statements about words and their properties form a complex network of propositions. More recent work on Word Grammar cites neurocognitive linguistics as a source of inspiration for the idea that language is nothing but a network. One of the attractions of the network view is the possibility of analysing language in the same way as other kinds of knowledge, given that knowledge, or long-term memory, is widely considered to be a network.

Word grammar is an example of cognitive linguistics, which models language as part of general knowledge and not as a specialised mental faculty.^[1] This is in contrast to the nativism of Noam Chomsky and his students.

Notes

[1] *Word Grammar: New Perspectives on a Theory of Language Structure*, Kensei Sugayama, et al., 2006, p.xv. ISBN 0-8264-8645-2

Bibliography

- *Language Networks: The New Word Grammar*, Richard Hudson, 2007 ISBN 0-19-926730-8

External links

- General introduction (<http://www.phon.ucl.ac.uk/home/dick/wg.htm>)
- Word grammar presented in an encyclopedic style (<http://www.phon.ucl.ac.uk/home/dick/enc-gen.htm>)
- Jasper, Holmes (2005) (PDF (341 pages)). *Lexical properties of English verbs (Phd thesis)* (<http://www.coventry.ac.uk/cu/external/content/1/c4/71/73/v1211898394/user/lexpro.pdf>). University of London.
Retrieved 2008-06-07.

Link grammar

Link grammar (LG) is a theory of syntax by Davy Temperley and Daniel Sleator which builds relations between pairs of words, rather than constructing constituents in a tree-like hierarchy. There are two basic parameters: directionality and distance. Link grammar is similar to dependency grammar, but dependency grammar includes a head-dependent relationship, as well as lacking directionality in the relations between words. Colored Multiplanar Link Grammar (CMLG) is an extension of LG allowing crossing relations between pairs of words^[1]

For example, in a subject–verb–object language like English, the verb would look left to form a subject link, and right to form an object link. Nouns would look right to complete the subject link, or left to complete the object link.

In a subject–object–verb language like Persian, the verb would look left to form an object link, and a more distant left to form a subject link. Nouns would look to the right for both subject and object links.

Syntax

Rightward links are represented as a +, and leftward links with a -. Optional links are contained in curly brackets {...}. Undesirable links are contained in any number of square brackets [...]. Multiple links are joined either by a conjunction & or a disjunction or. Each rule ends with a semicolon ;.

Examples

Example 1

A basic rule file for an SVO language might look like:

```
<determiner>: D+;
<noun-subject>: {D-} & S+;
<noun-object>: {D-} & O-;
<verb>: S- & {O+};
```

Thus the English sentence, “The boy painted a picture” would appear as:

+-----O-----+
+-D-+-S---+ +-D---+
The boy painted a picture

Example 2

Conversely, a rule file for a null subject SOV language might consist of the following links:

```
<noun-subject>: S+;
<noun-object>: O+;
<verb>: {O-} & {S-};
```

And a simple Persian sentence, *man nAn xordam* (من نان خوردم) 'I ate bread' would look like:

+-----S-----+
+-+O--+
man nAn xordam

Implementations

The link grammar syntax parser is a library for natural language processing written in C. It is available under the BSD license, which is compatible with the GNU General Public License. The parser is an ongoing project, located here [2]. Recent versions include improved sentence coverage, various bug and security fixes, and Java language bindings.

There are also Perl, Python, Ruby, Java, OCaml and .NET bindings available.^[3]

The *link-grammar* program along with rules and word lists for English may be found in standard Linux distributions, e.g., as a Debian package.^[4]

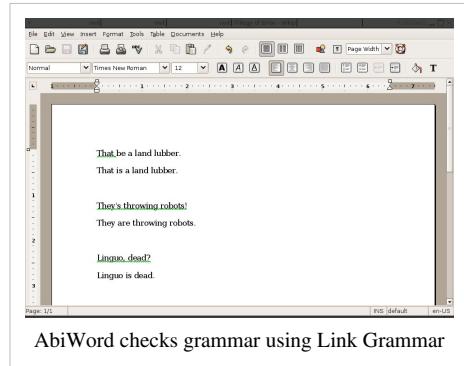
Applications

AbiWord, a free word processor, uses Link Grammar for on-the-fly grammar checking.^[2] Words that cannot be linked anywhere are underlined in green.

The RelEx semantic relationship extractor^[5], layered on top of the Link Grammar library, generates a dependency grammar output by making explicit the semantic relationships between words in a sentence. Its output can be classified as being at a level between that of SSyntR and DSyntR of Meaning-Text Theory. It also provides framing/grounding, anaphora resolution, head-word identification, lexical chunking, part-of-speech identification, and tagging, including entity, date, money, gender, etc. tagging. It includes a compatibility mode to generate dependency output compatible with the Stanford parser^[6], and Penn TreeBank-compatible POS tagging.

Link Grammar has also been employed for information extraction of biomedical texts^{[7][8]} and events described in news articles,^[9] as well as experimental machine translation systems from English to German and Turkish.

The Link Grammar link dictionary is used to generate and verify the syntactic correctness of two different natural language generation systems: NLGen^[10] and NLGen2.^[11] It is also used as a part of the NLP pipeline in the OpenCog AI project.



AbiWord checks grammar using Link Grammar

Notes

- [1] Anssi Yli-Jyrä and Matti Nykänen (2004). "A Hierarchy of Mildly Context-Sensitive Dependency Grammars" (<http://www.ling.helsinki.fi/~aylijyra/dissertation/5.pdf>). In G. P. Gerhard Jäger, Paola Monachesi and S. Wintner. *Proceedings of the 9th conference on Formal Grammar 2004 "FGNancy". Pre-Proceedings.* pp. 151–165. .
- [2] <http://www.abisource.com/projects/link-grammar/>
- [3] (Perl) (<http://search.cpan.org/~dbrian/Lingua-LinkParser/>) (Python) (<https://launchpad.net/pylinkgrammar/>) (Ruby) (<http://www.deveiate.org/projects/Ruby-LinkParser>) (OCaml) (<http://ramamurthy.ramu.googlepages.com/ocamllinkgrammar>) (.NET) (<http://proai.com/cs/files/folders/linkgrammar/default.aspx>)
- [4] Debian - Package Search Results - link-grammar (<http://packages.debian.org/link-grammar>)
- [5] <http://opencog.org/wiki/RelEx>
- [6] <http://nlp.stanford.edu/software/lex-parser.shtml>
- [7] Jing Ding, Daniel Berleant, Jun Xu, Andy W. Fulmer (November 2003). "Extracting biochemical interactions from MEDLINE using a link grammar parser" (<http://ifsc.ualr.edu/jdberleant/papers/LGPmanuscript8-8-03a.pdf>). *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on.* pp. 467–471. ISBN 0-7695-2038-3. .
- [8] Sampo Pyysalo, Tapio Salakoski, Sophie Aubin and Adeline Nazarenko, " Lexical Adaptation of Link Grammar to the Biomedical Sublanguage: a Comparative Evaluation of Three Approaches (<http://www.biomedcentral.com/1471-2105/7/S3/S2>", *BMC Bioinformatics* 7(Suppl 3):S2 (2006).
- [9] Harsha V. Madhyastha, N. Balakrishnan, K. R. Ramakrishnan (2003). "Event Information Extraction Using Link Grammar". *13th International WorkShop on Research Issues in Data Engineering: Multi-lingual Information Management (RIDE'03)*. pp. 16. doi:10.1109/RIDE.2003.1249841.

- [10] Ruiting Lian, *et al*, "Sentence generation for artificial brains: a glocal similarity matching approach", Neurocomputing (Elsevier) (2009, submitted for publication).
- [11] Blake Lemoine, NLGen2: A Linguistically Plausible, General Purpose Natural Language Generation System (<http://www.louisiana.edu/~bal2277/NLGen2.doc>) (2009)

Further reading

- Schneider, Gerold (1998). *A Linguistic Comparison Constituency, Dependency, and Link Grammar* (<http://www.ifi.unizh.ch/cl/study/lizarbeiten/lizgerold.pdf>). Masters Thesis, University of Zurich. Retrieved 2007-12-26.
- Daniel Sleator & Davy Temperly (1993). "Parsing English with a Link Grammar" (<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/LG-IWPT93.pdf>). *Third International Workshop on Parsing Technologies*.
- "A robust parsing algorithm for link grammars" (<http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/tr95-125.pdf>). *Proceedings of the Fourth International Workshop on Parsing Technologies* (Prague). September 1995.

External links

- The original Link Grammar homepage (<http://www.link.cs.cmu.edu/link/>) (which has been replaced by the current project (<http://www.abisource.com/projects/link-grammar/>).)
 - Online English demonstration (<http://www.link.cs.cmu.edu/link/submit-sentence-4.html>) (for an older, out-of-date version; many bugs have been fixed since this version.)
- LinkGrammar-WN (<http://www.eturner.net/linkgrammar-wn/>), lexicon expansion for the Link Grammar Parser (out of date, superseded by recent work that has been incorporated into the link-grammar parser.)
- BioLG (<http://mars.cs.utu.fi/biolg/>), a modification of the Link Grammar Parser adapted for the biomedical domain (many, but not all, BioLG enhancements have been folded back into the main link-grammar distribution).
- Parsing sentences with Link Grammar and Python (<http://www.youtube.com/Wk8zAr0R9zQ>) by Jeff Elmore (<http://jeffelmore.org/2012/03/23/parsing-sentences-with-link-grammar-and-python/>) at PyCon 2012 (<https://us.pycon.org/2012/>)

Language extensions

- Arabic Link Grammar extension (<http://www.ling.ohio-state.edu/~jonsafari/arabiclg/bin/arabiclg.html>) (Source package (<http://www.ling.ohio-state.edu/~jonsafari/arabiclg/arabiclg.20060829.tar.bz2>))
- Persian Link Grammar extension (<http://www.ling.ohio-state.edu/~jonsafari/persianlg/persianlg-0.8.5.tar.gz>)
- Online Persian demonstration (<http://students.cs.byu.edu/~jonsafar/persianlg.html>)
- Russian Link Grammar demonstration (<http://sz.ru/parser/>)
- Turkish Link Grammar extension developed as Masters degree thesis (http://www.cs.bilkent.edu.tr/~ilyas/PDF/THESES/ozlem_istek_thesis.pdf)

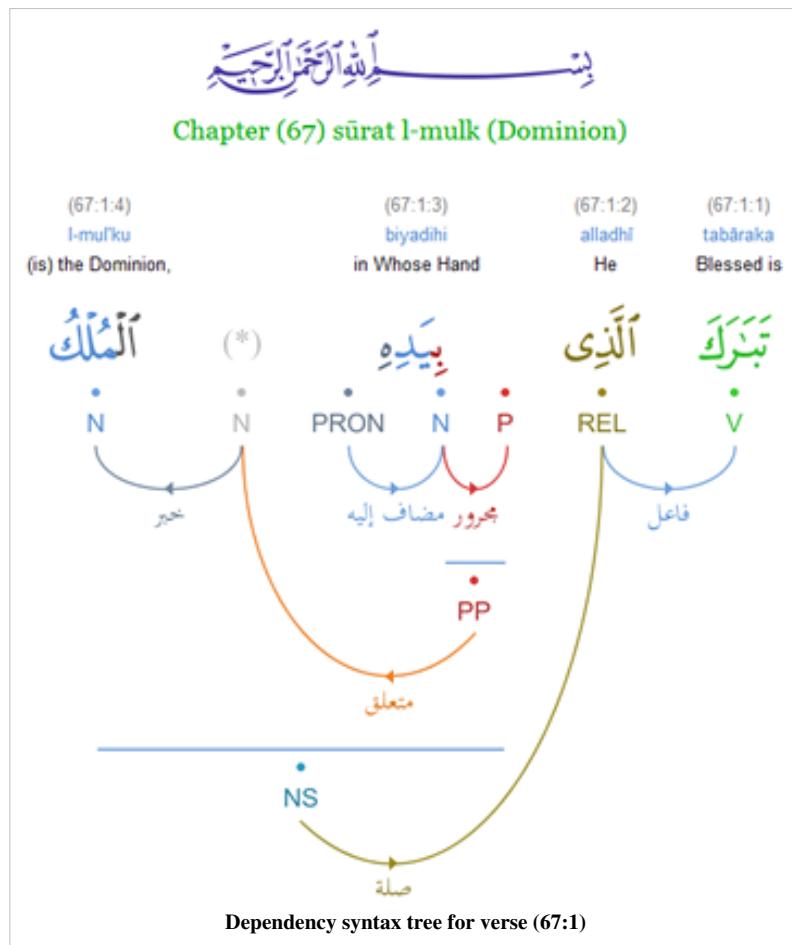
Quranic Arabic Corpus

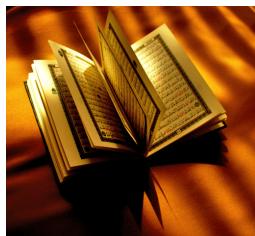
Quranic Arabic Corpus	
Research center:	University of Leeds
Initial release:	November 2009
Language:	Quranic Arabic, English
Annotation:	Syntax, morphology
Framework:	Dependency grammar
License:	GNU General Public License
Website:	http://corpus.quran.com/

The **Quranic Arabic Corpus** is an annotated linguistic resource consisting of 77,430 words of Quranic Arabic. The project aims to provide morphological and syntactic annotations for researchers wanting to study the language of the Quran [1] [2]. The grammatical analysis helps readers further in uncovering the detailed intended meanings of each verse and sentence. Each word of the Quran is tagged with its part-of-speech as well as multiple morphological features. Unlike other annotated Arabic corpora, the grammar framework adopted by the Quranic Corpus is the traditional Arabic grammar of i'rāb (اعراب). The research project is led by Kais Dukes [3] at the University of Leeds, and is part of the Arabic language computing research group within the School of Computing, supervised by Eric Atwell [4].

The annotated corpus includes [1] [5]:

- A manually verified part-of-speech tagged Quranic Arabic corpus.
- An annotated treebank of Quranic Arabic.
- A novel visualization of traditional Arabic grammar through dependency graphs.
- Morphological search for the Quran.
- A machine-readable morphological lexicon of Quranic words into English.
- A part-of-speech concordance for Quranic Arabic organized by lemma.
- An online message board for community volunteer annotation.





Corpus annotation assigns a part-of-speech tag and morphological features to each word. For example, annotation involves deciding whether a word is a noun or a verb, and if it is inflected for masculine or feminine. The first stage of the project involved automatic part-of-speech tagging by applying Arabic language computing technology to the text. The annotation for each of the 77,430 words in the Quran was then reviewed in stages by two annotators, and improvements are still ongoing to further improve accuracy.

Linguistic research for the Quran that uses the annotated corpus includes training Hidden Markov model part-of-speech taggers for Arabic^[6], automatic categorization of Quranic chapters^[7], and prosodic analysis of the text^[8].

References

- [1] K. Dukes, E. Atwell and N. Habash (2011). *Supervised Collaboration for Syntactic Annotation of Quranic Arabic*. (<http://www.kaisdukes.com/papers/qcorpus-lrej2011.pdf>) Language Resources and Evaluation Journal (LREJ). Special Issue on Collaboratively Constructed Language Resources.
- [2] K. Dukes and T. Buckwalter (2010). *A Dependency Treebank of the Quran using Traditional Arabic Grammar*. (<http://www.kaisdukes.com/papers/qadt-infos2010.pdf>) In Proceedings of the 7th International Conference on Informatics and Systems (INFOS). Cairo, Egypt.
- [3] <http://www.kaisdukes.com>
- [4] <http://www.comp.leeds.ac.uk/eric>
- [5] K. Dukes and N. Habash (2011). *One-step Statistical Parsing of Hybrid Dependency-Constituency Syntactic Representations*. (<http://www.kaisdukes.com/papers/qparse-iwpt2011.pdf>) International Conference on Parsing Technologies (IWPT). Dublin, Ireland.
- [6] M. Albared, N. Omar and M. Ab Aziz (2011). *Developing a Competitive HMM Arabic POS Tagger using Small Training Corpora*. (<http://www.springerlink.com/content/028kv16361qj1436>) Intelligent Information and Database Systems. Springer Berlin, Heidelberg.
- [7] A. M. Sharaf and E. Atwell (2011). *Automatic Categorization of the Quranic Chapters*. (http://www.comp.leeds.ac.uk/scsams/papers/ICCA2011_proceedings_paper26.pdf) 7th International Computing Conference in Arabic (ICCA11). Riyadh, Saudi Arabia.
- [8] C. Brierley, M. Sawalha and E. Atwell (2012). *Boundary Annotated Qur'an Corpus for Arabic Phrase Break Prediction*. (http://ivacs.info/download/i/mark_dl/u/4009575456/4560843681/IVACSprogramme2012.pdf) IVACS Annual Symposium. Cambridge.

External links

- Quranic Arabic Corpus (<http://corpus.quran.com>)

Functional theories of grammar

Functional discourse grammar

Functional grammar (FG) and **functional discourse grammar (FDG)** are grammar models and theories motivated by functional theories of grammar. These theories explain how linguistic utterances are shaped, based on the goals and knowledge of natural language users. In doing so, it contrasts with Chomskyan transformational grammar. Functional discourse grammar has been developed as a successor to functional grammar, attempting to be more psychologically and pragmatically adequate than functional grammar.^{[1][2]}

The top-level unit of analysis in functional discourse grammar is the discourse move, not the sentence or the clause. This is a principle that sets functional discourse grammar apart from many other linguistic theories, including its predecessor functional grammar.

History

Functional grammar (FG) is a model of grammar motivated by functions.^[3] The model was originally developed by Simon C. Dik at the University of Amsterdam in the 1970s,^[4] and has undergone several revisions since then. The latest standard version under the original name is laid out in the 1997 edition,^[5] published shortly after Dik's death. The latest version features the expansion of the model with a pragmatic/interpersonal module by Kees Hengeveld and Lachlan Mackenzie.^[1] This has led to a renaming of the theory to functional discourse grammar. This type of grammar is quite distinct from systemic functional grammar as developed by Michael Halliday and many other linguists since the 1970s.

The notion of "function" in FG generalizes the standard distinction of grammatical functions such as subject and object. Constituents (parts of speech) of a linguistic utterance are assigned three types or levels of functions:

1. Semantic function (Agent, Patient, Recipient, etc.), describing the role of participants in states of affairs or actions expressed
2. Syntactic functions (Subject and Object), defining different perspectives in the presentation of a linguistic expression
3. Pragmatic functions (Theme and Tail, Topic and Focus), defining the informational status of constituents, determined by the pragmatic context of the verbal interaction

Principles of functional discourse grammar

There are a number of principles that guide the analysis of natural language utterances according to functional discourse grammar.

Functional discourse grammar explains the phonology, morphosyntax, pragmatics and semantics in one linguistic theory. According to functional discourse grammar, linguistic utterances are built top-down in this order by deciding upon:

1. The pragmatic aspects of the utterance
2. The semantic aspects of the utterance
3. The morphosyntactic aspects of the utterance
4. The phonological aspects of the utterance

According to functional discourse grammar, four components are involved in building up an utterance:

- The conceptual component, which is where the communicative intention that drives the utterance construction arises
- The grammatical component, where the utterance is formulated and encoded according to the communicative intention
- The contextual component, which contains all elements that can be referred to in the history of the discourse or in the environment
- The output component, which realizes the utterance as sound, writing, or signing

The grammatical component consists of four levels:

- The interpersonal level, which accounts for the pragmatics
- The representational level, which accounts for the semantics
- The morphosyntactic level, which accounts for the syntax and morphology
- The phonological level, which accounts for the phonology of the utterance

Example

This example analyzes the utterance "I can't find the red pan. It is not in its usual place." according to functional discourse grammar at the interpersonal level.

At the interpersonal level, this utterance is one discourse move, which consists of two discourse acts, one corresponding to "I can't find the red pan." and another corresponding to "It is not in its usual place."

- The first discourse act consists of:
 - A declarative illocutionary force
 - A speaker, denoted by the word "I"
 - An addressee
 - A communicated content, which consists of:
 - A referential subact corresponding to "I"
 - An ascriptive subact corresponding to "find", which has the function Focus
 - A referential subact corresponding to "the red pan", which contains two ascriptive subacts corresponding to "red" and "pan", and which has the function Topic
- The second discourse act consists of:
 - A declarative illocutionary force
 - A speaker
 - An addressee
 - A communicated content, which consists of:
 - A referential subact corresponding to "it", which has the function Topic
 - An ascriptive subact corresponding to "in its usual place", which has the function Focus
 - Within this subact there is a referential subact corresponding to "its usual place", which consists of:
 - A referential subact corresponding to "its"
 - An ascriptive subact corresponding to "usual"
 - An ascriptive subact corresponding to "place"

Similar analysis, decomposing the utterance into progressively smaller units, is possible at the other levels of the grammatical component.

References

- [1] Hengeveld, Kees; Mackenzie, J. Lachlan (August, 2008). *Functional Discourse Grammar: A Typologically-Based Theory of Language Structure* (<http://www.oup.com/uk/catalogue/?ci=9780199278114>). Oxford: Oxford University Press. ISBN 978-0-19-927811-4. .
- [2] Mackenzie, J. Lachlan / Gómez-González, María de los Ángeles, ed. (2005). *Studies in Functional Discourse Grammar* (<http://www.peterlang.com/Index.cfm?vID=10696&vLang=E>). Linguistic Insights, Studies in Language and Communication. **26**. Peter Lang Publishing Group. ISBN 978-3-03910-696-7. .
- [3] Hurford, J (1990). Roca, I. M. ed. "Nativist and functional explanations in language acquisition" (<http://www.isrl.uiuc.edu/~amag/langev/paper/hurford90nativist.html>). *Logical Issues in Language Acquisition* (Foris, Dordrecht): 85–136. .
- [4] Dik, Simon C. (1989). *The Theory of functional grammar, Parts 1 & 2* (1 ed.).
- [5] Dik, Simon C. (1997). *The Theory of Functional Grammar, Part 1: The Structure of the Clause* (http://books.google.com/books?id=qeMLE_5uvHcC&dq=theory+of+functional+grammar&lr=&source=gbs_summary_s&cad=0) (2 ed.). Berlin: Mouton de Gruyter. ISBN 3-11-015539-7. ISBN 3-11-015404-8. .

External links

- Functional Grammar home page (<http://www.functionalgrammar.com/>)
- Functional Discourse Grammar homepage (<http://www.functionaldiscoursegrammar.info/>)

Prague school

The **Prague school**, or **Prague linguistic circle**,^[1] was an influential^[2] group of literary critics and linguists in Prague. Its proponents developed methods of structuralist literary analysis during the years 1928–1939.^[3] It has had significant continuing influence on linguistics and semiotics. Following the Czechoslovak coup d'état of 1948, the circle was disbanded in 1952, but the Prague School continued as a major force in linguistic functionalism (distinct from the Copenhagen school or English Firthian — later Hallidean — linguistics). American scholar Dell Hymes cites his 1962 paper, "The Ethnography of Speaking," as the formal introduction of Prague functionalism to American linguistic anthropology.^[4]

The Prague linguistic circle included the Russian émigrés Roman Jakobson, Nikolai Trubetzkoy, and Sergei Karcevskiy, as well as the famous Czech literary scholars René Wellek and Jan Mukařovský. The instigator of the circle and its first president was the Czech linguist Vilém Mathesius (President of PLC until his death in 1945).

In 1929 the Circle promulgated its theses in a paper submitted to the First Congress of Slavists. "The programmatic 1929 Prague *Theses*, surely one of the most imposing linguistic edifices of the 20th century, encapsulated [sic] the functionalist credo."^[5] In the late 20th century, English translations of the Circle's seminal works were published by the Czech linguist Josef Vachek in several collections.

Also in 1929, the group launched a journal, *Travaux du Cercle Linguistique de Prague*. World War II brought an end to it. The *Travaux* was briefly resurrected from 1966–1971. The inaugural issue was devoted to the political science concept of center and periphery. It was resurrected yet again in 1995. The group's Czech language work is published in *Slovo a slovesnost* (Word and Literature).

Members

- Petr Bogatyrev (cs) (ru)
- František Čermák (cs)
- Miroslav Červenka (cs) (de)
- Bohuslav Havránek (cs) (ru)
- Tomáš Hoskovec (cs)
- Josef Hrabák (cs) (ru)
- Roman Jakobson
- Sergej Karcevskij, i.e. Sergej Josifovič Karcevskij (cs) (ru)
- Oldřich Leška (cs)
- Alena Macurová (cs)
- Vilém Mathesius
- Jan Mukářovský
- Karel Oliva (cs)
- Vladimír Skalička (cs) (ru)
- Bohumil Trnka (cs)
- Pavel Trost (cs) (de)
- Nikolai Trubetzkoy
- Josef Vachek (cs)
- Jiří Veltruský (cs)
- Miloš Weingart (1980-1939) (cs)
- René Wellek

Contributors

- Aleksandar Belić, president of the Serbian Academy of Sciences and Arts (sr), (it)
- Émile Benveniste
- Karl Bühler
- Albert Willem de Groot
- Daniel Jones
- André Martinet
- Ladislav Matejka
- Lucien Tesnière
- Valentin Voloshinov

Influences

- Baudouin de Courtenay
- Filipp Fedorovich Fortunatov, the founder of the Moscow linguistic circle
- Ferdinand de Saussure^[6]

Influenced

- Joseph Greenberg
- Jiří Levý
- Dell Hymes
- Alf Sommerfelt
- Jože Toporišič
- Michael Halliday
- Viktor Shklovsky
- Michael Silverstein
- Jan Firbas

- Lubomír Doležel
- Austin Warren
- Jan Niecisław Baudouin de Courtenay
- Louis Hjelmslev
- Jaroslav Vacek
- Jaroslav Peregrin
- Tartu–Moscow Semiotic School

References

- [1] Czech: *Pražský lingvistický kroužek*, Russian: Пражский лингвистический кружок *Pražskij lingvističeskij kružek*, French: *Cercle linguistique de Prague*.
- [2] George Steiner. Linguistics and Poetics. In Extraterritorial. 1972. 137ff.
- [3] "Semiotic poetics of the Prague Scholl (Prague School)" (http://books.google.gr/books?id=CTJCIg9AeoC&pg=PA179&dq=%22semiotic+structuralism%22&hl=en&sa=X&ei=ulKbUNDPA8rOsga384GgDA&redir_esc=y#v=onepage&q=%22semiotic+structuralism%22&f=false): entry in the *Encyclopedia Of Contemporary Literary Theory: Approaches, Scholars, Terms*, University of Toronto Press, 1993.
- [4] Hymes, Dell (1982). "Prague Functionalism". *American Anthropologist* 84 (2): 398–399. doi:10.1525/aa.1982.84.2.02a00130.
- [5] Luelsdorf 1983, p. xvi
- [6] Linguistics. Volume 7, Issue 53, pages 100–127 ([http://www.degruyter.com/dg/viewarticle/j\\$002fling.1969.7.issue-53\\$002fling.1969.7.53.100.xml](http://www.degruyter.com/dg/viewarticle/j$002fling.1969.7.issue-53$002fling.1969.7.53.100.xml)).

Bibliography

- Luelsdorf, Philip A. 1983. On Pragian functionalism and some extensions. In Josef Vachek, Libuše Dušková, (eds.). *Praguiana: Some Basic and Less Known Aspects of The Prague Linguistic School*. John Benjamins. Linguistic and literary studies in Eastern Europe; 12. pp. xi-xxx.

External links

- The Prague Linguistic Circle homepage (<http://www.praguelinguistics.org>) (includes a list of publications about the Circle)
- (French) Dynamique du système linguistique - numéro thématique de *l'Echo des études romanes* consacré à la théorie de *potentialité* (Vilém Mathesius) (http://www.eer.cz/?s=2007_1-2)
- (French) Centre-périmétrie dans le système linguistique - numéro thématique de *l'Echo des études romanes* consacré à l'optique centro-périmétrique, une théorie élaborée au sein du Cercle linguistique de Prague (http://www.eer.cz/?s=2010_1-2)
- Perspective fonctionnelle de la phrase - l'apport du Cercle de Prague - numéro thématique de *l'Echo des études romanes* (http://www.eer.cz/?s=2012_1)

Systemic functional grammar

Systemic functional grammar (SFG) is a form of grammatical description originally developed by Michael Halliday in a career spanning more than 50 years.^[1] It is part of a social semiotic approach to language called systemic functional linguistics. The term *systemic* refers to the view of language as "a network of systems, or interrelated sets of options for making meaning".^[2] The term *functional* refers to Halliday's view that language is as it is because of what it has evolved to do (see metafunction). Thus, what he refers to as the *multidimensional architecture of language* "reflects the multidimensional nature of human experience and interpersonal relations."^[3]

Influences

Halliday describes his grammar as built on the work of Saussure, Louis Hjelmslev,^[4] Malinowski, J.R. Firth, and the Prague school linguists. In addition, he drew on the work of the American anthropological linguists Boas, Sapir and Whorf. His "main inspiration" was Firth, to whom he owes, among other things, the notion of language as system.^[5] Among American linguists, Benjamin Lee Whorf had "the most profound effect on my own thinking". Whorf "showed how it is that human beings do not all mean alike, and how their unconscious ways of meaning are among the most significant manifestations of their culture"^[6]

From his studies in China, he lists Luo Changpei and Wang Li as two scholars from whom he gained "new and exciting insights into language". He credits Luo for giving him a diachronic perspective and insights into a non-Indo-European language family. From Wang Li he learnt "many things, including research methods in dialectology, the semantic basis of grammar, and the history of linguistics in China".^[6]

Basic tenets

Some interrelated key terms underpin Halliday's approach to grammar, which forms part of his account of how language works. These concepts are: system, (meta)function, and rank. Another key term is lexicogrammar. In this view, grammar and lexis are two ends of the same continuum.

Analysis of the grammar is taken from a trinocular perspective, meaning from three different levels. So to look at lexicogrammar, we can analyze it from two more levels, 'above'(semantic) and 'below' (phonology). This grammar gives emphasis to the view from above.

For Halliday, grammar is described as systems not as rules, on the basis that every grammatical structure involves a choice from a describable set of options. Language is thus a *meaning potential*. Grammarians in SF tradition use system networks to map the available options in a language. In relation to English, for instance, Halliday has described systems such as *mood*, *agency*, *theme*, etc. Halliday describes grammatical systems as closed, i.e. as having a finite set of options. By contrast, lexical sets are open systems, since new words come into a language all the time.^{[7][8]}

These grammatical systems play a role in the construal of meanings of different kinds. This is the basis of Halliday's claim that language is *metafunctionally* organised. He argues that the *raison d'être* of language is meaning in social life, and for this reason all languages have three kinds of semantic components. All languages have resources for construing experience (the *ideational* component), resources for enacting humans' diverse and complex social relations (the *interpersonal* component), and resources for enabling these two kinds of meanings to come together in coherent text (the *textual* function).^{[9][10]} Each of the grammatical systems proposed by Halliday are related to these metafunctions. For instance, the grammatical system of 'mood' is considered to be centrally related to the expression of interpersonal meanings, 'process type' to the expression of experiential meanings, and 'theme' to the expression of textual meanings.

Traditionally the "choices" are viewed in terms of either the content or the structure of the language used. In SFG, language is analysed in three ways (strata): semantics, phonology, and lexicogrammar.^[11] SFG presents a view of

language in terms of both structure (grammar) and words (lexis). The term "lexicogrammar" describes this combined approach.

Metafunctions

From early on in his account of language, Halliday has argued that it is inherently functional. His early papers on the grammar of English make reference to the "functional components" of language, as "generalized uses of language, which, since they seem to determine the nature of the language system, require to be incorporated into our account of that system."^[12] Halliday argues that this functional organization of language "determines the form taken by grammatical structure".^[13]

Halliday refers to his functions of language as metafunctions. He proposes three general functions: the *ideational*, the *interpersonal* and the *textual*.

Ideational metafunction

The ideational metafunction is the function for construing human experience. It is the means by which we make sense of "reality".^[14] Halliday divides the ideational into the logical and the experiential metafunctions. The logical metafunction refers to the grammatical resources for building up grammatical units into complexes, for instance, for combining two or more clauses into a clause complex. The experiential function refers to the grammatical resources involved in construing the flux of experience through the unit of the clause.

The ideational metafunction reflects the contextual value of "field", that is, the nature of the social process in which the language is implicated.^[10] An analysis of a text from the perspective of the ideational function involves inquiring into the choices in the grammatical system of "transitivity": that is, process types, participant types, circumstance types, combined with an analysis of the resources through which clauses are combined. Halliday's *An Introduction to Functional Grammar* (in the third edition, with revisions by Christian Matthiessen)^[15] sets out the description of these grammatical systems.

Interpersonal metafunction

The interpersonal metafunction relates to a text's aspects of *tenor* or interactivity.^[16] Like field, tenor comprises three component areas: the speaker/writer persona, social distance, and relative social status.^[17] Social distance and relative social status are applicable only to spoken texts, although a case has been made that these two factors can also apply to written text.^[18]

The speaker/writer persona concerns the stance, personalisation and standing of the speaker or writer. This involves looking at whether the writer or speaker has a neutral attitude, which can be seen through the use of positive or negative language. Social distance means how close the speakers are, e.g. how the use of nicknames shows the degree to which they are intimate. Relative social status asks whether they are equal in terms of power and knowledge on a subject, for example, the relationship between a mother and child would be considered unequal. Focuses here are on speech acts (e.g. whether one person tends to ask questions and the other speaker tends to answer), who chooses the topic, turn management, and how capable both speakers are of evaluating the subject.^[19]

Textual metafunction

The textual metafunction relates to *mode*; the internal organisation and communicative nature of a text.^[20] This comprises textual interactivity, spontaneity and communicative distance.^[21]

Textual interactivity is examined with reference to disfluencies such as hesitations, pauses and repetitions.

Spontaneity is determined through a focus on lexical density, grammatical complexity, coordination (how clauses are linked together) and the use of nominal groups. The study of communicative distance involves looking at a text's cohesion—that is, how it hangs together, as well as any abstract language it uses.

Cohesion is analysed in the context of both lexical and grammatical as well as intonational aspects^[22] with reference to lexical chains^[23] and, in the speech register, tonality, tonicity, and tone.^[24] The lexical aspect focuses on sense relations and lexical repetitions, while the grammatical aspect looks at repetition of meaning shown through reference, substitution and ellipsis, as well as the role of linking adverbials.

Systemic functional grammar deals with all of these areas of meaning equally within the grammatical system itself.

Children's grammar

Michael Halliday (1973) outlined seven functions of language with regard to the grammar used by children:^[25]

- the instrumental function serves to manipulate the environment, to cause certain events to happen;
- the regulatory function of language is the control of events;
- the representational function is the use of language to make statements, convey facts and knowledge, explain, or report to represent reality as the speaker/writer sees it;
- the interactional function of language serves to ensure social maintenance;
- the personal function is to express emotions, personality, and "gut-level" reactions;
- the heuristic function used to acquire knowledge, to learn about the environment;
- the imaginative function serves to create imaginary systems or ideas.

Relation to other branches of grammar

Halliday's theory sets out to explain how spoken and written texts construe meanings and how the resources of language are organised in open systems and functionally bound to meanings. It is a theory of language in use, creating systematic relations between choices and forms within the less abstract strata of grammar and phonology, on the one hand, and more abstract strata such as context of situation and context of culture on the other. It is a radically different theory of language from others which explore less abstract strata as autonomous systems, the most notable being Noam Chomsky's. Since the principal aim of systemic functional grammar is to represent the grammatical system as a resource for making meaning, it addresses different concerns. For example, it does not try to address Chomsky's thesis that there is a "finite rule system which generates all and only the grammatical sentences in a language". Halliday's theory encourages a more open approach to the definition of language as a resource; rather than focus on grammaticality as such, a systemic functional grammatical treatment focuses instead on the relative frequencies of choices made in uses of language and assumes that these relative frequencies reflect the probability that particular paths through the available resources will be chosen rather than others. Thus, SFG does not describe language as a finite rule system, but rather as a system, realised by instantiations, that is continuously expanded by the very instantiations that realise it and that is continuously reproduced and recreated with use.

Another way to understand the difference in concerns between systemic functional grammar and most variants of generative grammar is through Chomsky's claim that "linguistics is a sub-branch of psychology". Halliday investigates linguistics more as a sub-branch of *sociology*. SFG therefore pays much more attention to pragmatics and discourse semantics than is traditionally the case in formalism.

The orientation of systemic functional grammar has served to encourage several further grammatical accounts that deal with some perceived weaknesses of the theory and similarly orient to issues not seen to be addressed in more structural accounts. Examples include the model of Richard Hudson called *word grammar*.

References

- [1] <http://www.isfla.org/Systemics/definition.html>. Retrieved 1 July 2011
- [2] Halliday, M.A.K. *Introduction to functional grammar*, 2nd ed. (1994) London: Edward Arnold., p. 15
- [3] Halliday, M.A.K. 2003. On the "Architecture" of Human Language. In *On Language and Linguistics*. Volume 3 in the Collected Works of M.A.K. Halliday. Edited by Jonathan Webster. p29.
- [4] (Halliday, 1994:xxvi):
- [5] Halliday, M.A.K. 1985. Dimensions of Discourse Analysis: Grammar. In *The Handbook of Discourse Analysis*, Vol 2: Dimensions of Discourse. London: Academic Press. Reprinted in full in *On Grammar*, Volume 1 in the Collected Works of M.A.K. Halliday. London and New York: Continuum. p262.
- [6] Halliday, M.A.K. 1985. Systemic Background. In "Systemic Perspectives on Discourse, Vol. 1: Selected Theoretical Papers" from the *Ninth International Systemic Workshop*, James D. Benson and William S. Greaves (eds). Ablex. Reprinted in Full in Volume 3 in *The Collected Works of M.A.K. Halliday*. London: Continuum. p. 188.
- [7] Halliday, M.A.K. 1961. Categories of the Theory of Grammar. *Word*, 1961, 17(3), pp241–92. Reprinted in full in Halliday, M.A.K. *On Grammar*. Volume 1 in the Collected Works of M.A.K. Halliday. Edited by J.J. Webster. London and New York: Continuum. pp40–41.
- [8] Halliday, M.A.K. and Matthiessen, C.M.I.M. 2004. An Introduction to Functional Grammar. Arnold. p37ff.
- [9] Halliday, M.A.K. 1977. Text as Semantic Choice in Social Context. In Teun A. van Dijk and János S. Petofi. Berlin: Walter de Gruyter, Grammars and Descriptions. Reprinted in full in M.A.K. Halliday, 2002. *Linguistic Studies of Text and Discourse*. Edited by J.J. Webster. London: Continuum.,
- [10] Halliday, M.A.K. and Hasan, R. 1985. *Language, context and text: Aspects of language in a social semiotic perspective*. Geelong: Deakin University Press.
- [11] <http://www.isfla.org/Systemics/Definition/chapelle.html>. Retrieved 30 July 2008
- [12] Halliday, M.A.K. 1970. Functional Diversity in Language as seem from a Consideration of Modality and Mood in English. *Foundations of Language: International Journal of Language and Philosophy*, 6. Reprinted in full in *Studies in English Language*, Volume 7 in the Collected Works of M.A.K. Halliday. Edited by J. J Webster. London and New York: Continuum. p167.
- [13] Halliday, M.A.K. 1970. Functional Diversity in Language as seem from a Consideration of Modality and Mood in English. *Foundations of Language: International Journal of Language and Philosophy*, 6. Reprinted in full in *Studies in English Language*, Volume 7 in the Collected Works of M.A.K. Halliday. Edited by J. J Webster. London and New York: Continuum. p166.
- [14] Halliday, M.A.K. *The Essential Halliday*. London and New York: Continuum. Chapter 12: Metafunctions.
- [15] Halliday, M.A.K. and Matthiessen, C.M.I.M. 2004. An Introduction to Functional Grammar. Arnold.
- [16] O'Halloran, K.A. (ed.) *English grammar in context, Book 2: Getting inside English* (2006), The Open University, p. 15.
- [17] Coffin, C (ed.) *English grammar in context, Book 3: Getting practical* (2006), The Open University, p. 11
- [18] O'Halloran, K.A. (ed.) *English grammar in context, Book 2: Getting inside English* (2006), The Open University, p. 22.
- [19] Coffin, C (ed.) *English grammar in context, Book 3: Getting practical* (2006), The Open University, pp. 22–23
- [20] O'Halloran, K.A. (ed.) *English grammar in context, Book 2: Getting inside English* (2006), The Open University, p. 36.
- [21] Coffin, C (ed.) *English grammar in context, Book 3: Getting practical* (2006), The Open University, p. 245
- [22] Coffin, C (ed.) *English grammar in context, Book 3: Getting practical* (2006), The Open University, p. 158
- [23] Coffin, C (ed.) *English Grammar in Context, Book 3, Getting Practical* (2006) The Open University, p.158
- [24] Coffin, C (ed.) *English grammar in context, Book 3: Getting practical* (2006), The Open University, p. 184
- [25] Butler, C.S., *Structure and function* (2003), John Benjamins, p. 415

External links

- For more information, see the SFG web site at: Systemic functional grammar (<http://www.isfla.org/Systemics/>)
- For a large bibliography containing the vast majority of systemic functional writings, see the bibliography site at: (<http://www.fb10.uni-bremen.de/anglistik/langpro/bibliographies/index.htm>)
- Word grammar (<http://www.phon.ucl.ac.uk/home/dick/wg.htm>)
- Layman's introduction: A simple description of using SFG techniques in language and literacy tuition (<http://manxman.ch/moodle2/course/view.php?id=4>)

Cognitive grammar

Cognitive grammar is a cognitive approach to language developed by Ronald Langacker, which considers the basic units of language to be symbols or conventional pairings of a semantic structure with a phonological label. Grammar consists of constraints on how these units can be combined to generate larger phrases which are also a pairing of semantics and phonology. The semantic aspects are modeled as image schemas rather than propositions, and because of the tight binding with the label, each can invoke the other.

Langacker develops the central ideas of cognitive grammar in his seminal, two-volume *Foundations of cognitive grammar*, which became a major departure point for the emerging field of Cognitive Linguistics.

Like construction grammar (developed by Langacker's student Adele Goldberg), and unlike many mainstream linguistic theories, cognitive grammar extends the notion of symbolic units to the grammar of languages. Langacker further assumes that linguistic structures are motivated by general cognitive processes. In formulating his theory, he makes extensive use of principles of gestalt psychology and draws analogies between linguistic structure and aspects of visual perception.

References

- Langacker, Ronald W. (1982) 'Space Grammar, Analysability, and the English Passive', *Language*, 58, 1, 22-80.
- Langacker, Ronald W. (1987) *Foundations of Cognitive Grammar*, Volume 1, *Theoretical Prerequisites*. Stanford: Stanford University Press.
- Langacker, Ronald W. (1990) *Concept, Image, and Symbol: The Cognitive Basis of Grammar*. (Cognitive Linguistics Research 1.) Berlin/New York: Mouton de Gruyter. [paperback edition 1991]
- Langacker, Ronald W. (1991) *Foundations of Cognitive Grammar*, Volume 2, *Descriptive Application*. Stanford: Stanford University Press.
- Langacker, Ronald W. (2008) *Cognitive Grammar: A Basic Introduction*. New York: Oxford University Press.
- Taylor, John R. (2002) *Cognitive Grammar*. Oxford Textbooks in Linguistics. Oxford: Oxford University Press.

Construction grammar

The term **construction grammar (CxG)** covers a family of theories, or models, of grammar that are based on the idea that the primary unit of grammar is the grammatical construction rather than the atomic syntactic unit and the rule that combines atomic units, and that the grammar of a language is made up of taxonomies of families of constructions.

CxG is typically associated with cognitive linguistics, partly because many of the linguists that are involved in CxG are also involved in cognitive linguistics, and partly because CxG and cognitive linguistics share many theoretical and philosophical foundations.

Some history

Historically, the notion of construction grammar developed out of the ideas of "global rules" and "transderivational rules" in generative semantics, together with the generative semantic idea of a grammar as a constraint satisfaction system. George Lakoff's "Syntactic Amalgams" paper in 1974 (Chicago Linguistics Society, 1974) posed a challenge for the idea of transformational derivation.

CxG was spurred on by the development of Cognitive Semantics, beginning in 1975 and extending through the 1980s. Lakoff's 1977 paper, *Linguistic Gestalts* (Chicago Linguistic Society, 1977) was an early version of CxG, arguing that the meaning of the whole was not a compositional function of the meaning of the parts put together locally. Instead, he suggested, constructions themselves must have meanings.

CxG was developed in the 1980s by linguists such as Charles Fillmore, Paul Kay, and George Lakoff. CxG was developed in order to handle cases that intrinsically went beyond the capacity of generative grammar.

The earliest study was "There-Constructions," which appeared as Case Study 3 in George Lakoff's *Women, Fire, and Dangerous Things*.^[1] It argued that the meaning of the whole was not a function of the meanings of the parts, that odd grammatical properties of Deictic There-constructions followed from the pragmatic meaning of the construction, and that variations on the central construction could be seen as simple extensions using form-meaning pairs of the central construction.

Fillmore et al.'s (1988) paper on the English *let alone* construction was a second classic. These two papers propelled cognitive linguists into the study of CxG.

The grammatical construction in CxG

In CxG, like in general semiotics, the grammatical construction is a pairing of form and content. The formal aspect of a construction is typically described as a syntactic template, but the form covers more than just syntax, as it also involves phonological aspects, such as prosody and intonation. The content covers semantic as well as pragmatic meaning.

The semantic meaning of a grammatical construction is made up of conceptual structures postulated in cognitive semantics: Image-schemas, frames, conceptual metaphors, conceptual metonymies, prototypes of various kinds, mental spaces, and bindings across these (called "blends"). Pragmatics just becomes the cognitive semantics of communication — the modern version of the old Ross-Lakoff performative hypothesis from the 1960s.

The form and content are symbolically linked in the sense advocated by Langacker.

Thus a construction is treated like a sign in which all structural aspects are integrated parts and not distributed over different modules as they are in the componential model. Consequentially, not only constructions that are lexically fixed, like many idioms, but also more abstract ones like argument structure schemata, are pairings of form and conventionalized meaning. For instance, the ditransitive schema [S V IO DO] is said to express semantic content X CAUSES Y TO RECEIVE Z, just like *kill* means X CAUSES Y TO DIE.

In CxG, a grammatical construction, regardless of its formal or semantic complexity and make up, is a pairing of form and meaning. Thus words are instances of constructions. Indeed, construction grammarians argue that all pairings of form and meaning are constructions including phrase structures, idioms, words and even morphemes.

Syntax-lexicon continuum

Unlike the componential model, CxG denies any strict distinction between the two and proposes a *syntax-lexicon continuum*. The argument goes that words and complex constructions are both pairs of form and meaning and differ only in internal symbolic complexity. Instead of being discrete modules and thus subject to very different processes they form the extremes of a continuum (from regular to idiosyncratic): syntax>subcategorization frame>idiom>morphology>syntactic category>word/lexicon (these are the traditional terms; construction grammars use a different terminology).

Grammar as an inventory of constructions

In CxG the grammar of a language is made up of taxonomic networks of families of constructions, which are based on the same principles as those of the conceptual categories known from cognitive linguistics, such as inheritance, prototypicality, extensions, and multiple parenting.

Four different models are proposed in relation to how information is stored in the taxonomies:

1. Full-entry model

In the full-entry model information is stored redundantly at all relevant levels in the taxonomy, which means that it operates, if at all, with minimal generalization.

2. Usage-based model

The usage-based model is based on inductive learning, meaning that linguistic knowledge is acquired in a bottom-up manner through use. It allows for redundancy and generalizations, because the language user generalizes over recurring experiences of use.

3. Default inheritance model

According to the default inheritance model, each network has a default central form-meaning pairing from which all instances inherit their features. It thus operates with a fairly high level of generalization, but does also allow for some redundancy in that it recognizes extensions of different types.

4. Complete inheritance model

In the complete inheritance model, information is stored only once at the most superordinate level of the network. Instances at all other levels inherit features from the superordinate item. The complete inheritance does not allow for redundancy in the networks.

Shift towards usage-based model

All four models are advocated by different construction grammarians, but since the late 1990s there has been a shift towards a general preference for the usage-based model. The shift towards the usage-based approach in CxG has inspired the development of several corpus-based methodologies of constructional analysis (for example, collostructional analysis).

Synonymy and monotony

As CxG is based on schemas and taxonomies, it does not operate with dynamic rules of derivation. Rather, it is monotonic.

Since CxG does not operate with surface derivations from underlying structures, it rejects constructional polysemy and adheres to functionalist linguist Dwight Bolinger's *principle of no synonymy*, on which Adele Goldberg

elaborates in her book.

This means that construction grammarians argue, for instance, that active and passive versions of the same proposition are not derived from an underlying structure, but are instances of two different constructions. As constructions are pairings of form and meaning, active and passive versions of the same proposition are not synonymous, but display differences in content: in this case the pragmatic content.

Some construction grammars

As mentioned above, CxG is a "family" of theories rather than one unified theory. There are a number of formalized CxG frameworks. Some of these are:

Berkeley Construction Grammar

Berkeley Construction Grammar (formerly also simply called Construction Grammar in upper case) focuses on the formal aspects of constructions and makes use of a unification-based framework for description of syntax, not unlike Head-Driven Phrase Structure Grammar. Some of its proponents/developers are Charles Fillmore, Paul Kay, Laura Michaelis, and to a certain extent Ivan Sag.

Goldbergian/Lakovian construction grammar

The type of construction grammar associated with linguists like Goldberg and Lakoff looks mainly at the external relations of constructions and the structure of constructional networks. In terms of form and function, this type of construction grammar puts psychological plausibility as its highest desideratum. It emphasizes experimental results and parallels with general cognitive psychology. It also draws on certain principles of cognitive linguistics.

Cognitive Grammar

Sometimes, Ronald Langacker's Cognitive grammar framework is described as a type of construction grammar. Cognitive grammar deals mainly with the semantic content of constructions, and its central argument is that conceptual semantics is primary to the degree that form mirrors, or is motivated by, content. Langacker argues that even abstract grammatical units like part-of-speech classes are semantically motivated and involve certain conceptualizations.

Radical construction grammar

William A. Croft's radical construction grammar is designed for typological purposes and takes into account cross-linguistic factors. It deals mainly with the internal structure of constructions. Radical Construction Grammar is totally non-reductionist, and Croft argues that constructions are not derived from their parts, but that the parts are derived from the constructions they appear in. Thus, in Radical Construction Grammar, constructions are likened to Gestalts. Radical Construction Grammar rejects the idea that syntactic categories, roles, and relations are universal and argues that they are not only language-specific, but also construction specific. Thus, there are no universals that make reference to formal categories, since formal categories are language- and construction-specific. The only universals are to be found in the patterns concerning the mapping of meaning onto form. Radical Construction Grammar rejects the notion of syntactic relations altogether and replaces them with semantic relations. Like Goldbergian/Lakovian construction grammar and Cognitive Grammar, Radical Construction Grammar is closely related to cognitive linguistics, and like Cognitive Grammar, Radical Construction Grammar appears to be based on the idea that form is semantically motivated.

Embodied construction grammar

Embodied construction grammar (ECG), which is being developed by the Neural Theory of Language (NTL) [2] group at ICSI, UC Berkeley, and the University of Hawai‘i, particularly including Benjamin Bergen and Nancy Chang, adopts the basic constructionist definition of a grammatical construction, but emphasizes the relation of constructional semantic content to embodiment and sensorimotor experiences. A central claim is that the content of all linguistic signs involve mental simulations and are ultimately dependent on basic image schemas of the kind advocated by Mark Johnson and George Lakoff and so ECG aligns itself with cognitive linguistics. Like Construction Grammar, Embodied Construction Grammar makes use of a unification-based model of representation. A non-technical introduction to the NTL theory behind Embodied Construction Grammar as well as the theory itself and a variety of applications can be found in Jerome Feldman's *From Molecule to Metaphor: A Neural Theory of Language* [3].

Fluid construction grammar

Fluid construction grammar (FCG) was designed by Luc Steels for doing experiments on the origins and development of language [4][5]. FCG is a fully operational and computationally implemented formalism for construction grammars and proposes a uniform mechanism for parsing and production. The Grammar integrates many notions from contemporary computational linguistics such as feature structures and unification-based language processing. Constructions are considered bidirectional and hence usable both for parsing and production. Processing is flexible in the sense that it can even cope with partially ungrammatical or incomplete sentences. FCG is called 'fluid' because it acknowledges the premise that language users constantly change and update their grammars. The research on FCG is conducted at Sony CSL Paris [6] and the AI Lab at the Vrije Universiteit Brussel.

Others

In addition there are several construction grammarians who operate within the general framework of CxG without affiliating themselves with any specific CxG program. There is a growing interest in the diachronic aspect of grammatical constructions and thus in the importation of methods and ideas from grammaticalization studies. Another area of growing interest is the pragmatics of pragmatic constructions. This is probably one of the reasons why the usage-based model is gaining popularity among construction grammarians. Another area of increasing interest among construction grammarians is that of language acquisition which is mainly due to Michael Tomasello's work.

References

- [1] Lakoff, George (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: University of Chicago Press.
 - [2] <http://www.icsi.berkeley.edu/NTL/>
 - [3] <http://www.m2mbook.org>
 - [4] <http://www.emergent-languages.org>
 - [5] Steels, Luc, ed. (2011). *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins.
 - [6] <http://csl.sony.fr>
- Bergen, Benjamin and Nancy Chang. Embodied Construction Grammar in Simulation-Based Language Understanding. In press. J.-O. Östman and M. Fried (eds.). *Construction Grammar(s): Cognitive and Cross-Language Dimensions*. Johns Benjamins.
 - Croft, William A. (2001). *Radical Construction Grammar: Syntactic Theory in Typological Perspective*. Oxford: Oxford University Press.
 - Croft, William A. and D. Alan Cruse (2004). *Cognitive Linguistics*. Cambridge: Cambridge University Press.
 - Feldman, Jerome A. (2006). *From Molecule to Metaphor: A Neural Theory of Language*. Cambridge : MIT Press.

- Fillmore, Charles, Paul Kay and Catherine O'Connor (1988). Regularity and Idiomaticity in Grammatical Constructions: The Case of *let alone*. *Language* 64: 501–38.
- Goldberg, Adele. (1995) *Constructions: A Construction Grammar Approach to Argument Structure*. Chicago: University of Chicago Press.
- Goldberg, Adele (2006). *Constructions at Work: The Nature of Generalization in Language*. Oxford: Oxford University Press.
- Lakoff, George (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: CSLI.
- Langacker, Ronald (1987, 1991). *Foundations of Cognitive Grammar*. 2 vols. Stanford: Stanford University Press.
- Michaelis, Laura A. and Knud Lambrecht (1996). Toward a Construction-Based Model of Language Function: The Case of Nominal Extrapolation. *Language* 72: 215–247.
- Michaelis, Laura A. and Josef Ruppenhofer (2001). *Beyond Alternations: A Construction-Based Account of the Applicative Construction in German*. Stanford: CSLI Publications.
- Michaelis, Laura A. (2004). Type Shifting in Construction Grammar: An Integrated Approach to Aspectual Coercion. *Cognitive Linguistics* 15: 1–67.
- De Beule Joachim and Steels Luc (2005). Hierarchy in Fluid Construction Grammar. Lecture Notes in Artificial Intelligence (LNCS/LNAI) 3698 (2005) pages 1–15. Berlin: Springer.
- Steels, Luc and De Beule, Joachim (2006). Unify and merge in fluid construction grammars. In: Vogt, P., Sugita, Y., Tuci, E. and Nehaniv, C., editors, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication, EELC 2006, Rome, Italy, September 30–October 1, 2006*, Lecture Notes in Computer Science (LNCS/LNAI) Vol. 4211, Berlin. Springer-Verlag. pp. 197–223.

External links

- Construction Grammar (<http://www.constructiongrammar.org>)
- Fluid Construction Grammar (<http://www.fcg-net.org/>)
- AI Lab Brussels (<http://arti.vub.ac.be>)
- Sony CSL Paris (<http://www.csl.sony.fr>)
- NTL Project (<http://www.icsi.berkeley.edu/NTL>)

Role and reference grammar

Role and Reference Grammar (RRG) is a model of grammar developed by William Foley and Robert Van Valin, Jr. in the 1980s, which incorporates many of the points of view of current functional grammar theories.

In RRG, the description of a sentence in a particular language is formulated in terms of (a) its logical (semantic) structure and communicative functions, and (b) the grammatical procedures that are available in the language for the expression of these meanings.

Among the main features of RRG are the use of lexical decomposition, based upon the predicate semantics of David Dowty (1979), an analysis of clause structure, and the use of a set of thematic roles organized into a hierarchy in which the highest-ranking roles are 'Actor' (for the most active participant) and 'Undergoer'.

Bibliography

- Foley, William A.; & Robert D. Van Valin, Jr (1984). *Functional syntax and universal grammar*. Cambridge: Cambridge University Press.
- Van Valin, Robert D., Jr. (Ed.). (1993). *Advances in Role and Reference Grammar*. Amsterdam: Benjamins.
- Van Valin, Robert D., Jr. (1993). A synopsis of Role and Reference Grammar. In R. D. Van Valin Jr. (Ed.), *Advances in Role and Reference Grammar* (pp. 1-164). Amsterdam: Benjamins.
- Van Valin, Robert D., Jr.; & William A. Foley (1980). Role and Reference Grammar. In: E. A. Moravcsik & J. R. Wirth (Eds.), *Current approaches to syntax* (pp. 329-352). Syntax and semantics (Vol. 13). New York: Academic Press.
- Van Valin, Robert D., Jr.; & Randy LaPolla (1997). *Syntax: Structure, meaning and function*. Cambridge: Cambridge University Press.
- Van Valin, Robert D., Jr. (2003). *Exploring the Syntax-Semantics Interface*. Cambridge: Cambridge University Press.

External links

- Introduction to RRG ("A Summary of Role and Reference Grammar" ^[1] PDF
- Role and Reference Grammar ^[2] (Van Valin's site)

References

- [1] <http://linguistics.buffalo.edu/people/faculty/vanvalin/rrg/RRGsummary.pdf>
[2] <http://wings.buffalo.edu/linguistics//people/faculty/vanvalin/rrg.html>

Emergent grammar

Emergent grammar is an approach to the study of syntax, originally proposed by Paul Hopper, which postulates that rules for grammar and syntactic structure emerge as language is used. It is distinguished from what Hopper calls the *A Priori Grammar Postulate*,^[1] which posits that grammar is a set of rules existing in the mind before anything else, and is exemplified by the school of generative grammar and the concept of Universal Grammar. Whereas Universal Grammar claims that features of grammar are innate,^[2] emergent grammar claims that the human language faculty has no innate grammar and that features of grammar are learned through experience; the term "emergent" in this context is synonymous with "non-innate."^[3]

Emergent grammar was formally proposed in 1987.^[4] Since then, it has been an approach common in linguistic discourse analysis and conversation analysis and has been used to investigate the relationship between grammatical structure and real-time interaction and language use.^[5]

References

- [1] Hopper, Paul (1988). "Emergent Grammar and the A Priori Grammar Postulate." In *Linguistics in Contact*, ed. Deborah Tannen.
- [2] Hornstein, Norbert, Jairo Nunes, and Kleanthes K. Grohmann (2005). *Understanding Minimalism*. New York: Cambridge University Press. pp. 99–100.
- [3] Mielke, Jeff (2005). "Ambivalence and ambiguity in laterals and nasals". *Phonology* 22 (2): 193. doi:10.1017/S0952675705000539.
- [4] Hopper, Paul. "Emergent Grammar" (<http://web.archive.org/web/20080918233214/http://home.eserver.org/hopper/emergence.html>). *Berkeley Linguistics Society* 13: 139–157. . Retrieved 2008-09-18.
- [5] Fox, Barbara (2007). "Principles shaping grammatical practices: an exploration" (<http://dis.sagepub.com/cgi/content/abstract/9/3/299>). *Discourse Studies* 9 (3): 299. doi:10.1177/1461445607076201. . Retrieved 2008-10-16.

Article Sources and Contributors

Linguistics Source: <http://en.wikipedia.org/w/index.php?oldid=526114353> Contributors: 12.88.120.xxx, 130.225.51.xxx, 209.240.222.xxx, 2bornot2b, 64.26.98.xxx, AK IM OP, Aalahazrat, Aaron Schulz, Aaron north, AdamRaizen, AdilJapan, Ahoerstemeier, Aintsemic, Akanemoto, Akerbeltz, AkselGerner, AlanBarnet, Alansohn, Alex S, Alfio, Alinovic, Allformweek, Almkglor, Alt-o, Altenmann, Alton, Alvinpoe, Amirk6585, Andjor, Andrea moro, Andycip, Angr, Antandrus, Ante Aikio, Antonio Lopez, Ap, Aphaiia, Apmbal1, Aranea Mortem, Artegis, Asenie, Ashmoo, Ashot Gabrielyan, Askari Mark, Astuishiin, Avitohol, AxelBoldt, BRG, Barbara Shack, Bastique, Benc, Bentong Isles, Bluemask, Bobelvis, Bobo192, Bogdangiusca, Boleslav Bobcik, Bomac, Bongwarrior, Bookuser, Borislav, Brad7777, Brandoobbe, Brandon.macuser, Brion VIBBER, Buridan, Byelf2007, CRGreathouse, Cadr, Caiomarinho, Caledones, Capricorn42, Chdorsett, Charles Matthew, CharlesGillingham, Chris the speller, Christopher Kraus, Chun-hian, Closedmouth, Cnilep, Colonies Chris, Combreir, Connormah, Conversion script, Cooper-42, Coren, CorpX, Crazycrazycrazy, Cromwell, Crusadeonilliteracy, Css, Cybercobra, DDD DDD, DJ Clayworth, DMacks, DVD R W, DanKeshet, Darigan, Davidjobson, Daykart, Dbachmann, Debresser, Deeptrivia, Deleet, Dennis Brown, DennisDaniels, Desiphral, Devotkha, Dezidor, DionysosProteus, Discospinster, Dlhcierekim, Docu, Dolmagray, Donald Albury, Doric Loon, Drphilharmonic, E104421, Ecksemess, Efe, Ehrbar, Ej, El C, ElBenevolente, Elatb, ElbowingYouOut, Ellmist, Ematch1, Emil Perder, Emufreak2, Emw, Eob, Epbr123, Ergative rlt, Esanchez7587, Eshley, Eugene-elgato, EugeneZelenko, Ewan dunbar, FaerielnGrey, FatsOGSD, Feedmymind, Fellowscientist, Ferhengyan, FilipeS, Filll, Firstwingman, Flatterworld, Fplay, Fram, Francis Tyers, FrancisTyers, Freakofniture, Func, Furykef, Fuzheado, Fwc, Fæ, Gaius Cornelius, Galoubet, Gansam12, Garik, Gem131, Giftlite, Gigacannon, Gilliam, Gioto, Glane23, Glenn, Glossologist, Gogo Dodo, Googl, GordonUS, Grafitus, Greg-si, GregLee, Gregbab, Grick, Gronky, Grstain, Guaka, Gum375, Gun Powder Ma, Gurch, Hadal, HamYoyo, Hannez Hirzel, Hans Adler, Hazchem, Headbomb, Headfacemouth, Hectorthebat, Hephaestos, Hermitsstudy, Himatsu Bushi, Hires an editor, Hirzel, Hm423, Hoo man, Hoss, HusseiniJacob75, Icarins, Ike9898,imiraven, Indeterminate, Infovoria, Innoak, Iph, Ish ishwar, Island, J Crow, J Spencer, JALockhart, JMStewart, JNF TweiT, Jacquerie27, Jadoogiri, Jagged 85, Jalwikip, James Crippen, Jane Santos, Jeff G, Jelly Roal, Jkominek, Jlitlett, Jmundo, Jo3sampl, Jobber, John RV, Joelemtalais, JoergenB, John, JohnOwens, Jonathan.s.kt, Jones Malcolm, Jorge Stolfi, Jose77, Jossi, Joycloete, Jrw7, Jsteph, Jus plain Bill, KBYU, Kagedron, Kajerov, Kalogeropoulos, Kalomfa, Kaobear, Karmosin, Khb3rd, Khokhoi, Kinema, KissitheGEEK, Knightingtail, Knutux, Kompar, Kowey, Krawi, Kwamikagami, Kylu, LA2, LMBM2012, Lacrimosus, Lam Kin Keung, Langageleon, LeadSongDog, Lecorbeaus, Lee Daniel Crocker, Lexor, Liftarn, Liifireball05, Ling.Nut, Linguaaua, LittleDan, LittleOldMe, Livajo, Looxix, Lova Falk, MIRACLE ONOCHE, MIT Trekkie, Maarten van Vliet, Macbookair3140, Macedonian, Macrakis, Mahaabaala, Majorly, May yi, Mandyvigilante, Mani1, Marek69, Mark Dingemanse, Marnanel, MarsRover, MastCell, Mathematicmajic, Mattisse, Matve, Maunus, Maurice Carbonaro, Mav, Mayumashu, Mdd, Menchi, Metropolitan90, Michael Hardy, Michaelrayw2, Mike Dillon, Mike2000, Milesbarger, Mindspillage, Mindstore, Mitsuki152, Mogism, Monty845, Moxfyre, Mr. Stradivarius, Mr. Stradivarius on tour, MrsCaptcha, Mundart, Musical Linguist, Muspilli, Mxn, NTox, Nanshu, NaomiRG, Nasz, Natalie Erin, Nawafpower, NawlinWiki, Ncapriola, Neelix, Netoholic, Newbyguesses, Newnoise, Nickshanks, Nigholith, Nihil novi, Nikai, Ninabeck, Nine Tail Fox, Ninly, Nohat, Norm mit, Noticket, Notyourbroom, Nposs, Objectivesea, Oda Mari, OlEnglish, Ophion, Oskar.shura.too.cool.101, Osmotic, Pakaran, PasswOrd, Paul A, Paul500, Pb30, Pedrodius, Pepper, Pete.Hurd, Pfold, Phantomtiger, Picus viridis, Pinethicket, Poetaris, Poitypoity, Polocrunch, Poor Yorick, Powered, Prpbhakarrao, Prof. Sergei Grinev-Griniewicz, Puceron, Puellianus, Purple Sheep, Python eggs, Quaddel, QuartierLatin1968, Quebec99, Quiddity, Quintessent, R'n'B, RA0808, Ramendra, Ram, Random account 47, Raymond Feinler, Reinhard Hartmann, ResidentAnthropologist, RexNL, Reynoldst, Rich Farmbrough, Richaraj, Richard Brooks MK, Rick richards, Rjanag, Rje, Robbiemuffin, Rogerb67, Ronjhones, RoseParks, Rosiestep, Rotem Dan, Rulevoider, Rumiton, Ryguasu, Sara Parks Ricker, Scarlet Lioness, Scholar1982, Sebesta, Shermanmonroe, Sietse Snel, Sintonak.X, Sinuhe, Sj, Skifunkster2011, Skizzik, Slrubenstein, Snyoes, Sobreira, SolKarma, Spifire19, Squarrels, Squideshi, Squids and Chips, Steven Zhang, Stevenmitchell, Stevertigo, Stinguist, SunDragon34, Supadawg, Supriya, Suruena, Susan118, Svenonius, Synchronism, Szqurrel, Szyslak, Tabitha2000, Taivo, Tappel, Tartarus21, Tassedethe, Taw, Tayste, Tbackson, Tbhotch, Tcampas42, Tech408, Technopat, Template namespace initialisation script, The Anome, The Nerd from Earth, The Transhumanist, The Transhumanist (AWB), TheAMmollusc, Thebeginning, Theoretick, Thewayforward, ThinkBlue, Tide rolls, Tijfo098, Tim Starling, TimBentley, Tobby72, Topbanana, Torgo, Touch Of Light, Triggerzeal13, Trilobitealive, TroubledTraveler, Trusilver, Trwier, Two halves, Ufotrain, UkrPaolo, Universal Life, Uriyan, Vaganyik, Vaiyach, Valmi, Vanberg, VegKilla, Velho, Venu62, VinTing, WLW, Wapcaplet, Wavelength, Wdkaye, Weaponb7, Wik, Wikipelli, Wikisan, WinedAndDined, Wknight94, Wleman, WmGB, Woohooikit, XJamRastafire, Xemoi, Yidisherryid, Yury Tarasievich, Zaheen, Zalmoxe, Zeke (surped), Ziggytar, 675 anonymous edits

Language Source: <http://en.wikipedia.org/w/index.php?oldid=526460453> Contributors: 01011000, 123abc432def, 151.24.190.xxx, 152.163.197.xxx, 168.209.98.xxx, 16@r, 172.170.236.xxx, 203.170.3.xxx, 212.34.204.xxx, 213.253.39.xxx, 28421u223nfenfcenc, 334a, 63.197.1.xxx, 7, 7kingis, A D 13, A. B., A193280, A314268, A520, AVand, Aadal, Abdurahman49, Abusharjeel, Achilleshart, Adam Bishop, Adammathias, Adashiell, Adjusting, Aeortiz, Aezirom, Ahoerstemeier, Akanemoto, Alansohn, Albrozdude, Ale jrb, Alessgrimal, Alexius08, Alexrexpt, Ali1986, Alison22, Alksentr, Allformweek, AlphaGamma1991, Altemann, Altg20April2nd, Amcguinn, Amit6, Ancheta Wis, Andonic, Andrea moro, AndreasJS, Andres, Andrew Lancaster, Andrew41052, Andrew, Andrewpmk, Andy M. Wang, Andy Marchbanks, Andycjp, Andysmith, Anetode, Ange, Angr, Animum, AnonGuy, AnonMoos, Antandrus, Ante Aikio, Antonielly, Antonio Lopez, Ap, Aphito, Apolotiger, Arakunem, Archer3, Aremith, ArglebargleIV, Aristotle1990, Arj, Art LaPella, Arwas, AshLin, Astropicithus, Attoal, Autoterm, Averaver, B J Bradford, BDFF2, BMF81, Babelfisch, Badly Bradley, Baronett, Bart133, Basharh, Bastianperrot, Batistontain, Bc2d00, Bdubay, Beetstra, Before My Ken, Beginning, Ben Standeven, Bentong Isles, Bernard Vanderydt, BillGates1234, BillGates12345, Bismaydash, Bkell, BlackAndy, Blue.man.blue, Bobo192, Bobster1961, Bongbang, Bongwarrior, BorgQueen, Borislav, Boznia, BrainyBabe, Bretz9, Brian0918, Brick Thrower, Brightkingdom, Brion VIBBER, Brunnock, Burnt, Burschik, Bwhacke, Bücherwürmlein, CJLL Wright, Cadr, Caf288, Caiverzi, Cal Evans, Caleb99999999, Caltas, Calypso, Cameron Nedland, Camme007, Can't sleep, clown will eat me, Captain-tucker, CardinalDan, Catgut, Cazwilson, Ccaasmss, CdaMVvWgS, Chaojoker, CharlesDexterWard, Chavrush, Chicovenancio, Chinese language is the bubonic plague, Chiswick Chap, Chris k, ChristopherWillis, Chuck SMITH, Chunky7, CivilCasualty, Cj.currie007, Cjppuffin, Ckatz, Cla68, Closedmouth, Cnilep, Codex Sinaiticus, Coelacan, Colorsontrial, Cometstyles, CommonsDelinker, Conscious, Conversion script, Cookie90, Cool Blue, Cquan, Crackaonrice, Crazy monkey84, Crculver, Crusadeonilliteracy, Css, Cybercobra, DJ Clawayorth, DVD R W, DahTroll123432213134597657, Damicatz, Dandellion, Danger, Danielstaithnall12, Darth Galper, DarylNickerons, Daugavpils, DaveGorman, David Pierce, David Shankbone, Davpar1, Dawn Bard, Dbachmann, Deljr, Dcoetze, Delloft, Denisarona, Denny, Denovoid, Denstet, DerHexer, Derekorah, Desiphral, Deus Ex, Deviathan, Dewritch, Dfrg.msc, Dhawali1, Dina, Dinesh Chander Kapoor, Dionsyous, Discospinster, Dlhcierekim, Dmacawb, Dmitri Lytov, Docu, Dolfrog, DonPaolo, Donarreiskoffer, Donnie Love, Dputig07, Dr Oldekop, Dr bab, Dr drsh, Dragon Slayer c.aily, Dummyface, Dwendedwende, Dwyermas214, Dylan Lake, Dysepsons, Dysmodrepanis, Dunadan, EPM, ESkog, Easton4516, Ebones, Edgar181, Edison, Eequor, Ehrenkater, Eia95, Ejosse1, Ejrijis, El Chemaniaco, Eleassar, Eleassar777, ElinorD, Eliyaho, Ellmist, Ellywa, Elusker, Ely el82, Emaregtreable, Ematch1, Emirjp, Emperorbma, EncycloPetey, Endrif, EngineerScotty, Enira Sum, Entropy, Enviroboy, Epbr123, Espoo, Eu.stefan, Eugene-elgato, EugenzeZelenko, Euryalus, Everyking, Eylekis, FCSundae, Feedmecereal, Felis, Dance, Felizdenovo, Feydey, Figs, Filelakeshoe, FilipeS, Finngall, Firewallsworks, Firespeaker, Fliffluff, Floorwalker, Fotisaros, Fplay, Fraggie81, FrancisTyers, Fredbauder, Free Bear, Freshenees, Frozenevolution, FruitMonkey, Fufthmin, Fullmon7722, Funkimon, Furykef, Fuzheado, Fæ, G Purevdorj, GHcool, Gabriel vii, GabrielAPetrie, Gaia Octavia Agrippa, Galoubet, Gandalf61, Gansam12, Gareth Griffith-Jones, Garik, Garion96, Garzo, Ghaly, Gilliam, Ginkgo100, Gioto, Gizmo II, Glane23, GnuDoyng, Goethean, Gogo Dodo, GoingBatty, Gomada, Googl, GraemeL, Grafen, Graham87, Gregbard, GregorB, Grrrubber, Grstain, Gcseshoyru, Gsmgmn, Guaka, Gulercan, Gzx0000, Hadal, Haaelth, Halaqah, Halmstad, Ham sandwich, Hanasato, Hannes Hirzel, Harac, Hardys, Harro5, Hasatalabunum, Haydenrules, Head, Headbomh, HeikoEvermann, Heironymous Rowe, Helikophis, Henry Gray, Heorot, Hephaestos, Hetar, HexaChord, Hippittrail, Hirzel, Hli11, Hmisland, HobbyManiac, Hoss, Hottentot, Hunaphu, Hyacinth, I dream of horses, IRP, Iamnotorange, Ian Maxwell, Ian.thomson, Iancarter, Icarins, Icarus of old, Ike9898, Imaglang, Imroy, Imsosex, Imz, Inanity, Indududududunesia, Instantramen92, Ioscius, Iridescent, Ish ishwar, Iswm, Itai, J. Martin, J.delanoy, J3ff, JCarriker, JForget, JNW, JS950910, JWb, Jabir99, Jackthepriest, Jade Knight, Jagged 85, Jahangard, Jakedasnakerules, James C. March, JamesBurns, JamesSooders, Janko, Jarble, Jarroldting, Jauerber, JayvierMC, Jayantyan, Jbinder, Jde Jeff G, Jeffrey Mall, JeremyA, Jesanj, Jfdwolff, Jgships, Jhn735, Jiang, Jiddisch, Jimandersson123, Jira123, Jjaquinta, Jkominek, Jmh649, Jmoran, Jmundo, JoanneB, John, John Lake, John of Reading, John254, JohnLai, Johnstar3345, Jonadin93, Jonah.ru, Jondel, Jordans92, Jose77, Joseph Solis in Australia, JoshuaZ, Jotomicron, Joymmart, Jpbown, Jrfartfanatik, Jrockley, Julian Grybowski, Jusdafax, Jusjih, Just plain Bill, Jwbilharvey, K.Nevelsteen, KF, KGasso, Kaiths, Kamicase, Kanags, Kang1164, Kaorhusu21, Karafias, Keenan Pepper, Keeno, Kelsie006, Keraka, Kharhaz, Khoikhoi, Killham, Kingturl, Kku, Koakhtzivid, Koavf, Kowey, Koyaanis Qatsi, Krawi, Kwamikagami, Kwertii, Kyshakk, LC, Lacrimosus, Lagalag, Lambiam, Landroving Linguist, LeaveSleaves, Lemirobi, Lenoxus, Leon7, Lexi Marie, Lexpression, Liftarn, Lightdarkness, Ling.Nut3, LinguistAtLarge, Ljl, Llakais, Locos epraxia, LogisticsMarmose1729, Lonewarrior12, Lova Falz, LoveYourLanguage, Lowellian, Lucole, Lupin, Lupo, M4390116, MARussellPESE, MER-C, Mac, Macmanui, Maddie!, Magioladitis, Magister Mathematicae, MagneticFlux, Mahmoud Youssef, Mairi, Makeemlighter, Malcs64, Malecasta, Malerin, Man yvi, Mani1, Marck, Marek69, Mark J, Markhurd, Marnanel, MartnetteD, Martin-C, Martinkunev, Mary96, Master Jay, Mathatalist, Mathieupp, Mathushan09, Matthew Stannard, Matthew Yeager, MattieTK, Maunus, Maureen, Max, Maximus Rex, Mayumashu, Mcarling, Mdd4696, Mel Etitis, Mensfortis, Mhkng, Michael Hardy, MichaelMcGuffin, Mike Linksvayer, Mike2000, Mikeblas, Mikkel R, Jakobsen, Mild Bill Hiccup, Mind Surfer, Minghong, Minorcorrections, MistyMorn, Mithridates, Mjb, Modernist, Monedula, Montgolfiere, Morwen, Mpmlan, Mr. Billion, Mr. Stradivarius, MrRandomPerson, MrTree, MrsCaptcha, Muchosucko, Mulder416, Muntuwandi, Musical Linguist, Mxn, N0thingness, N5iln, Nagy, Nakon, Nanshu, Natalie Erin, NawlinWiki, Nonayik, Neko-chan, Nellis, Neutrality, Neverquicks, Newbyguesses, Nicgeyuedee, NICK Number, NickRinger, Nitgv, Nihil novi, Nitricoxide, Nneoneo, No-Name, Noel 105, Non-droframe, Nono64, Notheruser, Novem Linguae, Numbersinstitute, Nylaus, Oblomov2, Oddtruth, Odwalla drinker, Olaf Davis, Olewgaki, Olivier, Olivia Curtis, Olivier, Oodelsofned, Osx85, Oets, OwenBlacker, Ownervtal, P0lyglut, PCode, PS2pcGAMER, PTSE, Pak21, Pan nelson, Parrama, Pasqual, Pass a Method, Pathwrote, Patstuas, Pau Erik, Paul Stanisfer, Paula Clare, Pb30, Pcap, Pedro, PedroPVZ, Pekayer11, Peter Horn, Peter Isotalo, Petr.adamek, Petralpa, Peza92, Pgk, Phgao, Philip Trueman, PhnomPencil, Physis, PierreAbbat, Pigman, Pigs677, Pinethicket, Pmjji, Pokemon1234321, Polaron Language Services, Poor Yorick, Pooryoric, Praveenp, Quadell, Quota, Qurgh, Qwenty345, Qwert, R'n'B, RJHall, RPlunk2853, Ramesses191, Ran, Rand, Ranveig, Raphink, Ratemonth, Rdsmith4, Reach Out to the Truth, Rebent, Recurring dreams, RedHillian, Reformationisnotenough, Regibox, Retostamm, Reubenator18, Rex Germanus, RexNL, Rhombus, Rich Farmbrough, RichardF, Rjanag, Rjwilmisi, Robbiemuffin, RobertG, RobinDuff, Rogerb67, Romanrn, Ronabop, Ronenkator, Rozzami, RoughOutline, Rowdymiriam, RoyBoy, Royalmate1, Rparle, RSpeer, Ruakh, Rursus, Rv77ax, Ryguasu, Ryulong, S, SLFAW, SMC, SS2005, Saforrest, Saintswithin, Salie34, Saluyot, Salvio giuliano, Sam Hocevar, Sango123, Sannab, Sarafemme, Sceptre, SchifftyThree, Scott Delaney, Scowlong, Sdlat, SecretLondon, Selket, Semmelweiss, Sfmannamia, ShadowRangerRIT, Shadowjams, Shanamifsdzelbst, Shanes, Shawnc, Shillog, Shoecream, Sisalot, Skizzik, Slave of Truth, Slrubenstein, Snowmonster, Snowlf, Snoyes, Soccerguy7735, Son of Paddy's Ego, Sonjaaa, Sparkspill, Spartaz, SpookyMulder, Spundun, Srini au eee, Srkris, Stagyar Zil Doggo, Stalve, Stancy, abraham, Staufisr, Stay cool, Stbalbach, Stellis, Stephen G. Brown, Stephenb, Stevertigo, Stopherchris, Storm Rider, Substitutesc, Sudachi, Suffusion of Yellow, Suidafrikaan, Sunnahh, SuperHamster, Supernaturalist, Supriya, Suruena, Swoomingdisaster, Sxbrown, Symane, TEB728, THEN WHO WAS PHONE?, Tabletop, Taeshadow, Tagishsimon, Tahmasp, Taivo, Tangotango, Taw, Tbook, TeaDrinker, Tedder, Teles, Terse, TexasAndroid, Tgeairn, Tgpqaz, The Anome, The Beagle, The Savagedry, The Thing That Should Not Be, The User 567, The Wiki ghost, TheFlyingApe, TheNewPhobia, TheTrainEnthusiast, Theda, Theopapada, Theroadislong, Tide rolls, Tim PF, Tim dorf, TimVickers, Timwi,

Tiop, Titoxd, Trxman307, Tobby72, Tobias Hoevekamp, Tomaxer, Torgo, Touch, Peel and Stand, Tox, Tpbradbury, Trachys, Traroth, Tresiden, Trevor MacInnis, Tslocum, Tumulus, Tunisia360tunisia, Twri, UberScienceNerd, Ulric1313, Uncle Dick, Uncle Milt, Universal Life, Unlocked, Usaartstore, Vacation9, Vaniba12, Vanwhistler, Ved036, Velella, Velho, Venu62, Versus22, VeryVerily, Vexperiential, Vicki Rosenzweig, Vidshow, VikSol, Vildricianus, Vivin, WLU, WMCEREBELLUM, Wafulz, Walker55, Wasell, Wavelength, Wayland, Weltall99, WiXWiXTM, Widr, Wik, Wiki alf, WikiLeon, Wikiloop, WikipedianMarlith, Willow1718, Woland37, Woodrrg, Woodstone, Woohookitty, Wragge, WwMalibu, Www.wikinerds.org, XLerate, Xander990, Xemoi, Xiahou, Yerpo, Yorubatashi, Yuckfoo, Yupik, Yvesimmo, Zaldax, Zeno Gantner, Zerida, Zero g, Zginder, Ziko, Zntrip, Zuuuzz, Ævar Arnfjörð Bjarmason, Σ, Александър, அவையுடன்னார், 大西洋鮀, 1440 anonymous edits

Natural language Source: <http://en.wikipedia.org/w/index.php?oldid=522323393> Contributors: 16@r, 1exec1, A. di M., Accents, AdeMiami, AgarwalSumeet, Akamad, Alphaios, Alvatov, Andycjp, Angr, Aphaiia, Armande, Autoterm, B J Bradford, Beland, BendersGame, BioPupil, Bonewah, Bongwarrior, BrainyBabe, Burschik, Cal Evans, Ccalvin, Cenarium, Chuck Smith, Conversion script, Cookie90, DVassallo, Damian Yerrick, Danebell, Davies, Dbachmann, Deivid, DewBmtn, Dmowers, Dolfrog, Dpr, Drork, Edcolins, Edward321, El C, ElinorD, Furykef, Gadykozma, Garik, Gregbard, Grin, Guyjohnston, Headbomb, Hibernal, Icze, Ivan Štambuk, IvanLanin, JMK, JasonAQuest, Jessicann, Jonathan.Wickens, Jordan123, JorisvS, Joseph Solis in Australia, Kauczuk, Kku, Krishnavedala, Kwamikagami, Kyoakoa, Lee Daniel Crocker, Leibniz, Linforest, LittleHow, Lova Falk, Luna Santin, MXVN, Mabisa, Magnus, MagnaMopus, Mange01, Marcos, Mattisse, Maunus, Maxsonbd, Michael Hardy, Micru, Mike.lifeeguard, Mild Bill Hiccup, Minur, Mjs072, Mr. Credible, Mr. Stradiarius, Mrevan, Nanshu, Netrapit, Nirmos, Nono64, Nricardo, Ntennis, Odonacao, Paperflowergirl, Pcap, Peter Horn, Peteri, Pi zero, PiKeeper, Plasticlar, Poccil, Pompeufab, Poryorick, Prosfilaes, Quiddity, Ray Foster, Red Slash, Ruakh, Ruud Koot, Ryan Vesey, Ryan524, Saeed, Veradi, Smeg8374, Smith felix, Sodmy, Srikipedia, Stephanos21, Stevertigo, Szyslak, Tanvir Ahmmmed, Taxman, The Wiki ghost, Thingg, Timwi, Tox, Trident13, Triplejumper, Tristatestar, UnitedStatesian, Vina, WikiRuler03, Winchaser, Yolgnu, Феникс, 112 anonymous edits

Theoretical linguistics Source: <http://en.wikipedia.org/w/index.php?oldid=525888504> Contributors: Ajd, Akhil999in, Altg20April2nd, Angr, Augustus Leonhardus Cartesius, BD2412, Cadr, Calumm, Dbachmann, Dinglydell, Edward, Edward321, Epbr123, FayssalF, Firstwingman, GlasGhost, GoodSirJava, Grendelkhan, Hippietrail, Hu12, Imiraven, Imran, IvanLanin, J Crow, Jasonbook99, Jasy jatere, Jerry, Jlittlet, Jobber, Jodie2003, Jonsafari, Jsteph, KBYU, Kwamikagami, Lizmarie, MacTire02, MaximvsDecimvs, Mxn, Nema Fakei, Noctibus, Nono64, Nschefey, Nurg, Rjwilmsi, Rmashadi, Schnitzi, Semmelweiss, Signalhead, SolKarma, Suraduttashandilya, Suruena, Szyslak, Template namespace initialisation script, The Wiki ghost, Trondtr, WereSpielChequers, Wiseskier, Yuribryk, Zhen Lin, Zhozefine, 創, 62 anonymous edits

Psycholinguistics Source: <http://en.wikipedia.org/w/index.php?oldid=52555213> Contributors: *drew, Aaron Kauppi, Action potential, Als022, Altenmann, Analogisub, Andre Engels, Anticipator, Barfooz, Bclbloutreach, Bit, Burschik, Chris the speller, Chuckiesdad, Ckatz, CliffC, D Monack, David Shay, Deaconse, Dmitri Lytov, Doczilla, Dpr, Duckbill, Dzou, ELApro, Eaeefremov, Fnienels, Gabnh, Garik, Gerteger, GlennLThompson, Golshaie, Heyitspeter, Hitoshi Inoue, II MusLiM HyBRID II, Icairns, Ioverka, Ish ishwar, JMW64, Jaihanuman, Jauhienij, JeLuF, Jeffmatt, Jgull, John Vandenberg, Johnkarp, Jonas.kluk, Jph89, Jroberts548, Junes, Karol Langner, Kimkenkat, Kingturtle, Kirill Lokshin, Koavf, Kora09, Lazcorp, Lerdswa, Lino08, Lova Falk, MarkkuP, Matlee, Mcbradaigh, Mcr hxc, Mellery, Michaelwilson, Michelle.chui, Millahhma, Misterx2000, Moncrief, Moyogo, Mr. Stradiarius, Msasscts, Neparis, NorVegan, Nposs, Numbo3, Pete142, Postman, Psykhosis, Psylng, Quibbles, Quiddity, Rdsmith4, RedWolf, Reinhard Hartmann, Rich Farmbrough, RichardF, Rjanag, Rjwilmsi, Rror, Rurus, RyanCross, SU26lham, Salthizar, Sandman2007, Sardanaphalus, Scope creep, Scottralter, SimonP, Smartse, Stemonitis, Swerdenabe, Taharley, Tanaata, Tassedethe, Technopat, Ted.strauss, Thakorn, The Rambling Man, Thedude22, Thüringer, Tlnqxn, Tom.k, Tremilux, Varti, Walej, Whatamidoing, Yahya Abdal-Aziz, Zoe, Александър, 114 anonymous edits

Sociolinguistics Source: <http://en.wikipedia.org/w/index.php?oldid=526253332> Contributors: 16@r, ABF, Aaron north, Access Denied, Aleksd, Alexkon, AnOrdinaryGuy, Andre Engels, AndrewHowse, Andycjp, Angr, Anomalocaris, Anticipation of a New Lover's Arrival, The, Antonielly, Ardonik, Argylemoose, Aubadaurada, Benne, Bgwhite, Brendan.wolfe, Bubuka, Byalinguist, CALR, CTangy, Canto2009, CapitalR, Charbe, Charles Matthews, Chasingsol, Chris the speller, Christian List, Chzz, Ckatz, Closedmouth, Cnilep, Cometstyles, Connor4355, CreoleFungus, Crónica, DASonnenfeld, Danji, Darigon Jr., Daveyboysimmo, DennisDaniels, Diana LeCrois, Diego, Djnjwd, Dlhochierekima, Dogah, EJPyat, Eaeefremov, Eburaconos, Ejrrjs, Entheta, Fastlysock, Fayanetic london, Filemon, FrancisTyers, Frankey silington, Frankenpuppy, Fratrep, Freddbaer, Fwappler, Gabnh, Gaius Cornelius, Gerteger, Grape1, Halaqaq, Inogad, J Di, J.delanoy, JaGa, Jbur1985, Jef-Infojef, Jeroen Claes, Jfpierce, JohnCD, JorisvS, Joy, Jugger90, Kaffeeringe.de, Karl-Henner, Kevlar67, Kh7, Ko'oy, Kzzl, LaggedOnUser, Lanov, Libcub, Liifireball05, Limegreen, Lucidish, MC10, Mack2, Matlee, Maxcap, Mdoff, Meeple, Meeples, Metonym, Michael Hardy, Michielob, Michkalas, Mirrorblade, Misarxit, Miskwito, Mnewmanqc, Modulatum, Montrealais, Moose91894, Mr. Stradiarius, MuffledThud, Musicalantonio, NERIUM, Neilc, Neongrey, Nicke L, NikoSilver, Notinasnid, Noula69, Nysin, Paperflowergirl, Pete unsth, Peter Isotalo, Peter Ladage, Petiphoque, PhilKnight, PiMaster3, Picabeat, Pintuful, Popwriter, Prmdunge, RandomP, Redlurses, Rich Farmbrough, Rjanag, Rjwilmsi, Rolyateleahicim, Sarichkaa, Sgsilver, Shizhao, Simetrical, Sinatra, Sluzelin, Smgc8374, SorenLinden, Spbm, SpeedyGonsales, Szyslak, Taral, Tarquin, Technopat, The Phoenix, The Sage of Stamford, TheAMmollusc, Theserialcomma, Tomseg, TreasuryTag, Triddle, Ufotrain, Unmerrlich, Verbum Veritas, Vrenator, Wavelength, WikHead, Wiki Wikardo, Windharp, Youshouldask, Zantedeschia, Zohab, Zuuuzz, Александър, سعی, 193 anonymous edits

Linguistic universal Source: <http://en.wikipedia.org/w/index.php?oldid=521835933> Contributors: Annabelle Lukin, Babbage, Brettz9, Cadr, Closedmouth, Commander Keane, Croquant, Cybercobra, Dbachmann, Dragon guy, Edriscon, Ergative rlt, Felix ahmler, FrancisTyers, Fratrep, Furykef, Grover cleveland, IJzeren Jan, Ish ishwar, JorisvS, Kolinuts68, Kwamikagami, Lambiam, Ligulem, Livajlo, Mark Dingemanse, McCajor, Mlewan, Mr. Stradiarius, Mustafaa, Ntennis, Palpalpal, RJCraig, Rama, Rspeer, Sburke, Solomonfromfinland, Stevertigo, Sundar, TAKASUGI Shinji, The Wiki ghost, Timwi, Trickstar, Tropylum, VikSol, WRK, 25 anonymous edits

Grammar Source: <http://en.wikipedia.org/w/index.php?oldid=526114257> Contributors: 15dancing, 16@r, 193.193.195.xxx, 1seeker, 2004-12-29T22:45Z, 2206, 28421u2232nfencenc, Adimovk5, Adwiii, AgentCDE, Ahoerstemeier, Airborne84, Akroch, Al E, Alansohn, Aliotra, Amaz777, Amire80, AnakngAraw, Andocomando, Andrea moro, Andres, AndrewWTaylor, Andycjp, Andyjsmith, Angela, Angr, Anomimski, Anthony Appleyard, Antonio Lopez, Apoyon, Arctic.gnome, ArglebargleIV, Arraes.daniel, Asav, Ashdog137, Aspire3623WXCI, Atomician, Azkar, B1narymaster, BLueFiSH.as, Babajobu, Badanedwa, Baklap123, Bananastalktome, Baronnet, Beetstra, Beland, Benc, Benjamin9832, Bernie Wadelheim, Bertman2, BiT, Bilbo1507, BiohazardSteven, BlanchardBlast, Blapstool, Blether, Blu Aardvark, Bluefalcon07, Boing! said Zebedee, BokikaC, Bongwarrior, Bornhuettner, Brambleclawx, Brian Sayrs, Brion VIBBER, Brock, Brocket, Burgsbee, Butko, Byef2007, CJGB, CZmarlin, Cadr, Caknuck, Calcyman, Calumn, CapnPrep, Ccotts, Cervello84, Chaos, Charles Sturm, CharlieGrammar, Chase13579, Cheesemonron666, Christian List, Christopher Kraus, Chuck Smith, ChuckSmith, Chupon, Cillerkrol, Cloudedy, Cntras, Colonies Chris, Cometstyles, Connel MacKenzie, Content unknown, Conversion script, Cool Blue, Cyberstrike2000x, D6, DARTH SIDIOUS 2, DMacks, DSRH, Daa89563, DancingPhilosopher, Darrenhusted, Davejohnsan, David Kernow, David Pierce, Dawn Bard, Dbachmann, Dbfers, Dcuber1, Ddar, Dduck, DeadEyeArrow, Deb, Deeptrivia, Dekisugi, Demi, Dennis Brown, DennisDaniels, Derflexer, Derek Ross, Dethme0w, Dgw, Dieter Simon, Dimimimon7, Dini05, DirectEdge, Discospinster, Dmerrill, DocWatson42, Dfog, Dominictheodonkey, DoubleBlue, Dreamata, Drumbrum18, Dudeiamwangin, Dwheeler, Dws, Dylansmrjones, Ed g2s, Eddie.willers, Edward321, Elvispress5, Emily912, Enchanter, Englishmambo, Eprb123, Esanchez7587, Esprit15d, Euthynon, Evil saltine, Excirial, FastLizard4, Favonian, Fennessy, Fergieruestheworld, FerralMoonrender, Flemirna, Flyguy649, Fred Bauder, Freond, Friedrich von Königsberg, Friviere, Fyyer, GDonato, GKaczinsky, GRAHAMUK, Gailib, Gandyloo2, Garik, Gelingvistoj, George Ho, Geschichte, Gfoley4, Gilgamesh he, Gioto, Glane23, Glen, Glenn, Gogo Dodo, Gourisgupta, Graham87, Grammar Cindy, Grammar nazi77, Grammaridae, Gregbard, Gregiscool14, Guaka, Gunsnroses15, Gwalla, Haemo, Hall Monitor, Hailo mlg, Hannes Hirzel, Hard Sin, Hayford Peirce, Hejimony, Hespccamp689, Hetar, Hghyx, Hike395, Hirzel, Hobartimus, Huwbwici, I dream of horses, IRP, IW.HG, ICseaturlies, Idksunitmn, Ilya Voyager, Improve, Imroy, Ioscius, IronGargoyle, Ish ishwar, Itsmejudith, Ifxfd64, J.delanoy, JB Duece, Jacquerie27, James Haughton, James086, JamesBWatson, Jamesontai, Jamie Shaw, Jayden54, Jayhawk of Justice, Jbeans, Jed 20012, Jeff G, Jeffrey Mall, Jehnidiah, Jerith, JerryFriedman, Jfingers88, Jiang, JinJian, Jnestorius, JoeSmack, JohnnyB256, Jon Awbrey, Josesnai, Joseph Solis in Australia, Josh Cherry, Jossi, Joy80, Joyous!, JulieADriver, Justin Eiler, Karl Dickman, Karl E. V. Palmen, Karl Palmen, Kasper17, Katalevano, Kathy T, Katiegiggles95, Keggrf, Kelsid, Kevinewolf, Kimmo Alm, Kindling, King of Hearts, Kingturtle, Kjoonlee, KnowledgeOfSelf, Krsont, Kuru, Kwamikagami, Kwertii, Kyakoa, L Kensington, LC, LaXian, LarRan, Laurens-af, Laxlacrossplayer, LeaveSleaves, Leeborkman, Leibniz, Levineps, Likhit, LilHelpa, LinasLit, Lisch1314, LordRobert, Luna Santin, Lupo, MER-C, MKoltnow, Mac, Macedonian, Magioladitis, Mani1, Maralia, Mareino, Mark102030, MarsRover, MattieTK, Maunus, Mav, Mayooranathan, Mayumashu, Mbell, MeStovo, Mellowyellowdog, Meno25, Meske, Michael Hardy, Miguelrmr, Miguels, Mike Klaassen, Milkmanjh, Mindmatrix, Minimac, Mista Bob Dobbsalina, Mjw212, Mogism, Montrealais, Mosionzd, Motorider57, Mr. Stradiarius, Mr. Stradiarius on tour, Mrwo, Mukadderat, Mundat, NHRHS2010, Naqebullah12, NawlinWiki, Never give in, Neverquick, Newbeard, Nick UA, Nicolas Eynaud, Noel Streetfield, Northamerica1000, Nposs, Numbo3, Odonacao, Ohnoitsjamie, Ontoraul, Orion1IM87, Ortolan88, Oxymoron83, PBS-AWB, Pat8722, Pax:Vobiscum, Penwhale, PeterBowing, Philip Trueman, PhnomPencil, Physicist, Pne, Postmodern Beatnik, Prabhakar P Rao, Pretzelpaws, Proofreader77, Proveft, PwnFlakes, Qwertyu, Qwfp, R'n'B, RHaworth, Randwicked, Razimantv, RazorICE, Rdsmith4, RedWolf, RexNL, Rijkbenik, Rivertorch, Rob Hoot, Robert Foley, Robertaji, RockMFR, Ronjhones, Rootbeer, Rtdixon86, Ruakh, S t 2007, SDLarsen, SMC, ST47, Sacundim, Saforrest, Sagaciousuk, Sanbeg, Saqib, Satanael, SatiricalTruth, Scwlong, Sean D Martin, Seba, Srgeddin, Shanel, Shantavira, Shirham, Shunpiker, SilkTork, Silvergooseman, Smcmanus, Smedwards95, Snowfl, Some Australian, Somno, SpaceFlight89, Spalding, SpeedyGonsales, SpookyMulder, Srkris, St1865, Standonbible, Stemonitis, SteveFredBob, Stevertigo, Storm Rider, Stroppolo, Subdulous, SuperHamster, Supernaturalist, Sverdrup, Symane, TAKASUGI Shinji, THEN WHO WAS PHONE?, TYelliott, Tanweyan, Tbhotch, Tdehnel, Tenguopr, Tentacles, The Anome, The Random Editor, The Rationalist, The Thing That Should Not Be, The undertow, Themfromspace, Tide rolls, Tinkuxln, Tolstoyevsk, Tommy2010, Torgo, Tsgreenhill, Tsm1128, Turk oğlan, Tyler, Unnecessarycommas, Vaganyik, Vague Rant, Vary, Viskonsas, Vrenator, Wahrmund, Wapcaplet, Wavelength, Wayiray, Wayne Miller, West Brom 4ever, Wickecho, Wikipeli, Wikiwhat?, Will Hen, Willaq, Willy132, Wireless Keyboard, Wjejskenewr, Woodstone, Woohookitty, Ww2censor, XYAaSehShalomX, Xanzzibar, Xtina'S, Yahya Abdal-Aziz, Yellow0924, YellowLeftHand, Zack wdaghiri, Zoicon5, Zomglolwtzor, Иле флоттант, Александър, Саша Стефанови, Կապոբե, יניבי ר' נאשְׁרָם, 808, 632 anonymous edits

English grammar Source: <http://en.wikipedia.org/w/index.php?oldid=525684822> Contributors: 1ForTheMoney, 28421u2232nfencenc, 2help, 80.255, 8ung3st, AK IM OP, AaronW, ActivExpression, Adam78, Adamw1993, Addshore, Afterwriting, Alexandre8, AliSmith7, Altermike, Andre Engels, AndreSt, Andreasge, Angr, ArchetypeRyan, Asvaghosa, Augur, Ayrtion Prost, BD2412, Babajobu, Badgermet, Badman89, Bamkin, Bedelato, Beetstra, Beland, Benc, Benjaburns, Bigtoe, Bináris, Blanchard, Bluenoose, Bongwarrior, BradBeattie, Brando130, Brett, Bud-E, Bugloaf, Cameron Nedland, Canthusus, Capnpre, Capricorn42, Carcharoth, Catgut, Ccady, Chumpdog, Charles Matthews, Chicksplatt, Chintu1992, Ckatz, ClamDip, ClickRick, Closedmouth, Cmyaseen, Cnilep, Cockneycapybara, Cornelfrockey, Count Truthstein, Courcelles, Cristianomacaluso, Cromulant, CyrilThePig4, Damezi, Damian Yerrick, DarthShrine, DavidHOzAu, Dawn Bard, Deljr, Dduck, Denisarona, DennisDaniels, Der Sporkmeister, Deville, Dieghosttown, Dinoftreak247, Discospinster, Doric Loon, DoubleBlue, Dr. Klim, Drew.ward, Duoduoduo, Dylan Lake, Dlugosz, Eivindsol, EmersonLowry, English Subtitle, English-checker, English-coach, English-helper, Eric Herboso, Esperant, Espoo, Estoy Aquí, Ezzett,

Evertype, Exeunt, FactoidCow, Facts707, Fama Clamosa, FilipeS, FisherQueen, Fishwristwatch, Flamingspinach, Flateric, Florian Blaschke, ForDorothy, Fortinbras, Fowler&fowler, Fwed66, G.broadwell, Gailth, Gaius Cornelius, Gary King, Garzo, Gogo Dodo, Goingin, Gordie, Graham87, GrammarGuide, Grover cleveland, Haipa Doragon, Hajhouse, Halverso, Hanxu9, Hayden120, Heatherceana, Heilongjiang2011, Henrygb, Hike395, Hohohob, Human anatomy, IMSOp, Iggy Koopa, Ihcocy, Ikar.us, Illnab1024, Ilvcapra, Indeterminate, Infectiba, Intershark, Jonesco, Irbisgreif, Irpond, Ish ishwar, Ixfd64, J.delanoy, Jacquerie27, Jakohn, JeremyStein, Jerome Charles Potts, Jesse Viviano, Jim1138, Jimp, Jobber, John Bentley, John2007, John254, Jon Awbrey, Jonadab, Jonathan de Boyne Pollard, Joseph Solis in Australia, Jusjih, Kaiwhakahaere, Kallerdis, Kamags, Katalevano, KathrynLybarger, Keilana, Keith Cascio, Kennn894, Kesuari, Kevinbromberg, Keycard, Khazar, Kelmer, Kingal86, Kiren07, KnowledgeOfSelf, Kornbelt888, Krutika, Ksanyi, Kungfuadam, Kwamikagami, Kweeket, LaXian, Lanthanum-138, Lars Washington, Lethe, Liao, LibLord, Lightmouse, LittleOldMe, Lofty, Loftypex, Logicaempiricist, Loje, Manfried, Manjusha Santosh, Manning Bartlett, Mark Dingemanse, Mark91, MartinHarper, Massimiliano Lincetto, Materialscientist, Mattdiaz, Mav, Mazeface, Mbell, Medtopic, Menchi, Michael Devore, Michael Sloane, Midnightdreary, MikeRic, Mindmatrix, Minot23, MisterSheik, Mogism, Monzac, Moose10000, Mwazzap, N-edits, N5iln, Nakon, NamelsRon, NawlinWiki, Neparis, Netspy, Neutrality, Ningauble, NixonB, Noetica, Npd2983, Nricardo, Ocrasaroon, Ole meram, Olivia Curtis, Ortolan88, Oth, Otheus, PaulGS, Paxxe, Pearl, Peti Kulatty, Pgduddha, Philip Trueman, Piast93, Pinethicket, Pithecanthropus, Pne, Poccil, Pokegdonavin, Prokopenya Viktor, Prunesquar, Quarl, QuiteUnusual, RA0808, RPlunk2853, RaiderRobert, Rajusutar999, Recognition, Red Slash, RedWolf, Redvers, Rex Germanus, Rfc1394, Rhymeword, Ricardo Carneiro Pires, Rich Farmbrough, RickGriffin, Ricky81682, Rjanag, Rjwilmsi, Rmhermen, Rms125a@hotmail.com, Robi-jam, Rocketrod1960, RuM, Ruakh, Sadhu44, Samwaltz, Saruman-the-white, Sebesta, Seonaibeachwith, Serendipodous, Shanes, Shreevatsa, SigPig, SiltTork, Simetrical, Siroxo, Sladeatul, SlimVirgin, Smallwhitelight, Smaug123, Soliloquial, Sotakeit, Spalding, Stegop, Sten for the win, Steverapaport, StradivariusTV, Struway, Sunray, Supergrunch, Supreme geek overlord, Supuhstar, Susan118, Sut84, Systems in Learning, TAKASUGI Shinji, TUF-KAT, Taikochediyoshi, Taowind, Taxman, Tearlach, Technopat, Testabright, TheMidnighters, Thegreen9, Thingy, ThinkEnemies, Tiddly Tom, Tiggerjay, Time, Timwi, Timoo58, Tomethy2, Tony1, Toodelemom, TopAce, Tow, UBeR, Ucucha, Unfree, Unoriginalname38, VKokielov, Victor Yus, Violetriga, Vrenator, Wa2ise, Wanion, Wavelength, Welsh, West.andrew.g, Wgoetsch, Whimemsz, White Trillium, Whitepaw, Wik, Wimt, Witan, Wknight94, Wohlhamhay, Wood Thrush, Woohookitty, Xaosflux, Yitscar, Yoko bono, Zeimus, Zorro CX, Zuuuzz, Èle flottante, احمد مصطفى السعد, 615 anonymous edits

Clause Source: <http://en.wikipedia.org/w/index.php?oldid=525494073> Contributors: 123x123x, 2602:306:2586:4A49:692B:3935:3539:751A, 28bytes, Adam Kern, Alansohn, Alexf, Anastius, Andrea moro, Antandrus, Arcan, Aulis Eskola, AvicAWB, Bartledan, Beland, Bevo, Bgs022, Bocobrock, BokiacK, Burschik, COMPFUNK2, CapnPrep, Chibigurl, Chirpish, Chrajohn, Chris.let, ChriGualtieri, DARTH SIDIOUS 2, DSRH, DVD R W, Dac04, DennisDaniels, Discospinster, Djinn112, Dlohcierekim, Donner60, Duoduo, Dysprosia, EEMIV, Eleassar, Elih Odgers, Epbr123, EscapingLife, Fratrep, GPHeemsley, Gaith, Gibues, Gilliam, Gobonobo, Gzkn, HV, Hariva, Hesperian, Idunn0271828, Indefatigable, Infovarius, InverseHypercube, Iridescent, J.delanoy, Jamesontai, Jdunck, Jesboat, John of Reading, Jonathan Webley, JorisvS, Jossi, Jusdafax, Keris Rain, Kingpin13, Kubigula, Largoplazo, Learningnerd, Levineps, Lordcrazy59, Luokehao, MacedonianBoy, Materialscientist, Mav, Mavigogun, Mazeface, Michael Hardy, Mschel, NawlinWiki, Netrapt, Nugzor, Octahedron80, One.guardian.angel, OneWeirdDude, Pepper, Pequod76, Pichu826, Pinethicket, Pol098, Ragesoss, Reach Out to the Truth, RedBLACKAndRun, Regenlied, Renaissance, RexNL, Ripohopeteq, Rjanag, Rror, Ruakh, SM, Sannab, SchreiberBike, Screensaver7, Screwpassenger, SgtThroat, StaticGull, Stephan Schulz, Sumone10154, Symane, The Thing That Should Not Be, ThePlaz, Theopolisme, Tizio, Tjo3ya, Tony1, Udkokerjapan, Uni4dx4, Velho, Veritas Aeterna, Victor Yus, WikHead, WikiHannibal, Xp54321, Zaharous, Zerida, Zien3, ۲۳۰ anonymous edits

Phrase Source: <http://en.wikipedia.org/w/index.php?oldid=523735771> Contributors: (ef), AbdulRaheemChanna, AdamRaizen, Ahoerstemeier, Alarm, Alksentr, Aly55655, Ammodramus, AnakngAraw, Andres, Andycjp, Angela, Anypodetos, Appleortiz, Araiza, Arcan, Ascidian, Bagatelle, Bassbonerocks, Beland, Bob743, Bobo192, Brett, Burschik, CJLL Wright, CLW, COMPFUNK2, Cadr, CambridgeBayWeather, Cameron Dewe, Caspian, Catgut, ChrisGualtieri, Chriswiki, Cnzs wikipedia, Confile, DanielCD, Darolew, Deor, Devilbreaker, Dlohcierekim, DonPaolo, Ekmr, Epbr123, Eumolpo, Euryalus, Excial, Filipes, Frosted14, GARYQUEZADA, Gedaechnis, Glenn, Gpvs, Gunkarta, Gwalla, HangingCurve, Hariva, Hippietrail, Hyacinth, I dream of horses, IZAK, Ilyaroz, ImacollegeprofessHer, Imz, Inferno, Lord of Penguins, Interactor, Isidore, J.delanoy, JForget, Jay Litman, Jncraton, Joanjoc, Jogloran, Jyusin, KEJ, Karl E. V. Palmen, Karl Palmen, Knubbi, Levines, Magioladitis, Malhonen, Mav, Mazeface, McSly, MelbourneStar, Menchi, Mets501, Meursault2004, Michael Hardy, Mikaey, Mr3366, Naddy, Nebulousity, NewEnglandYankee, Nickynick298, Nihil novi, Nimic86, Nk, Nohat, Noobvlking, NotAnonymous0, Octahedron80, One.guardian.angel, OwenX, Patrick, PierreAbbat, Pillsberry, Pinethicket, PleaseStand, Pne, Prashanthus, PvM Iain, Quiddity, R. S. Shaw, RJRCraig, RUL3R, Rell Canis, Rianadanielle 07, Rmashadi, Roastytoast, Robertlovescss, Ronz, Rrburke, Ruakh, SchfiftyThree, Schniggendiller, Seefan, Shadowjams, Shimmaua, Sintonak.X, Sir Tanx, Skyezx, Steve wogstyle tpain, Sxim, Taka, Teeyess, Thovt, Tjo3ya, TomTheHand, Tony Sidaway, Trusilver, Vaganyik, Violadave, Walter9it, Wideangle, Wikipelli, Wulfmgmorthaler, Yidisherdy, Ykhwong, Yingadottir, Zien3, ۲۲۵ anonymous edits

Word Source: <http://en.wikipedia.org/w/index.php?oldid=526217191> Contributors: SunShine\$, (jarbarf), 10, 100110100, 123456ty, Iexec1, 2D, 4k7a, 4k7z2, 5 albert square, 890F, ABF, AJR, Abhi44549, Abrecht, Ace ETP, Acroterion, Addshore, AdilJapan, Aggieguy, Ahoerstemeier, Aitias, Alansohn, Ale_jrb, Alec112358, Alex.muller, Alex11164, Alexbrewer, Alfio, Allmightyduck, AllyUnion, Altair, Anas Salloum, Andjor, Andy85719, Andycjp, Angr, Anonymous Dissident, Antandrus, Anubis Godfather, Aquatics, Aquinas jr, Arbitrarily0, Arbitrarily00, Arcandam, Asoer, Avenue, Avicennasis, AzatToth, Baa, Badgernet, Basawala, Baseballfan614, Batchee, Beland, Bellenion, Benjiboi, Berean Hunter, Bewboe, Bfnn, Bgtu, Bhazad, Bigloser, Blake-, Blanchardb, Blimjim, Bloo1998, Boby192, Bomac, Bongwarrior, Bookandcoffee, Brandon, Braydonf, Brodyisnumber1, C2kho, Calabre1992, Calimo, Callipides, Calmer Waters, Caltas, Can't sleep, clown will eat me, Candis Rochelle, Canthusus, CapnPrep, Capricorn42, ChaosPrince, ChemGardener, Clark89, Cleared as filed, Cleverline, Closedmouth, Cloudyed, Cmrnmrs, Cnzs wikipedia, Cnilep, CoJaBo, Cobi, Colfer2, Colonies Chris, Connorama12, Coolstarkman11, Cornellrockey, Courcelles, Crazyquesadilla, Crohnie, Cubiq, Cureden, Cyclonenim, Cyfal, Cynical, D, DVD R W, DVdm, Dan653, Daniel Olsen, DanielDemaret, Daonguyen95, Darwinek, DavidLittleMagic, Davidprior, Dbachmann, Dekisugi, Demize, DerHexer, Dermann69, Dez6 László, Dfeuer, Dgw, Diannas, Dis-ian, Discospinster, Dissident, Dlabet, Dlohcierekim, DoNotAnnoyMe, Docboat, Dojah, Doulous Christos, Dranorter, Drewf297, Dysepssion, E Wing, ESkog, EagleEyeEric, Ebones, Eddy117588, Edison, Erehenkater, Elaragir, Eliz81, Elusker, Elven2006, Epbr123, Esanchez7587, EscapingLife, Ezzett, Everything, Excirial, Facesmelt, Faigl, ladislav, Fate012, Feinoha, Fences and windows, Fetchcomms, Fieldday-sunday, Fobrox97, FoodRockss, Fordan, Forsakensundown, Fredrik, FreplySpang, Friedy-peach, Funandtrvl, Furykef, Fa, GB fan, GLaDOS, GSlicer, Galactor213, Galoubet, Gareth Griffith-Jones, Garzo, Gerbrant, Ghaly, Ghostshock, Ghét mâu dô, Git2010, GlobalCore, God1024, Gogo Dodo, Gourisgupta, Grafen, Grayish, Gregbard, Gcschoyr, Gurch, HalfShadow, HappySailor, HateMeMore, Hawkes, Helix84, HenryLi, HexaChord, Hman123232, Hyacinth, Hydrogen Iodide, I dream of horses, II MusLiM HyBRID II, IRP, IcedXxkillerXx, Iluvteletibus, Iridescent, Irishdnqrq, Isaiyah656, Ish ishwar, IslaySolomon, IvanLanin, Ixfd64, J. Spencer, J.delanoy, JForget, JaGa, Jacob.jose, Jacotto, Jahsonic, Jaketherum, James.monta.homie, Jammy0002, Javix98, Jeff G., Jeltz, JeremyA, Jersey Devil, Jgprovencio, Jikybwea, Jim1138, JinJian, Jiy, JJ137, JK2q3jrkse, Jkartsy, Jmlk17, Jmundo, Jncraton, John254, JohnCD, JohnCub, Johnnybriar, Jojit fb, Joshinschool, Joy, Jpgordon, Jpl1994, Juliancolton, JuneGloom07, Junglecat, Justzisgu, KFP, Kamakaze4, Karl-Henner, Karlhahn, Katalevano, Katieh584, Keraton, Kharissa, King Bee, Kingpin13, Kipala, Kiratani, Kittenzcp, KnowledgeOfSelf, Koafv, Koraiem, KoshVorlon, Kravi, Kubigula, Kusma, Kwamikagami, Kyky8952, Lam055839, Landy nuzun, Latinoadam, Latka, Lavintzin, Law, Lawliman15, Lazylaces, Learning120, LeaveSleaves, LedgendGamer, Leeannedy, Lemonhead3131, Lighterside, Lights, LikeLakers2, Lilodeh, LinguistAtLarge, Lmhabs, Logophile, Lotsofhugs999, Lucyin, Luna Santin, LuoShengli, MBisan, MD87, MER-C, MJ94, MKar, MR.WANNATOUCHA, MSGJ, Maciej Adwent, Magnet For Knowledge, Majorly, Mandarax, Mapetite526, Marek69, Mark Dingemanse, Mark91, MattieTK, Mattmils03, Matttoothman, Maunus, Maurice Carbonaro, Maurog, Maxiai (admin), Mayooranathan, Mazca, McLay1, Meaghan, Mean as custard, Member of the Brute Squad Faction of the EPY Foundation, Meno25, Mentifisto, Mephistophelian, Meske, Metropolis, Michael Angelkovich, Michael fyodorov, Michael93555, Miguelmm, Mike Rosoft, Mindmatrix, Miquonranger03, MitchLay, Mjr162006, Mnewmanqc, Mo-Al, Monkeybreth254, Mschel, Munici, Murtifton, Mxn, My76Strat, Mzajac, N2lean, N5iln, Nakon, Nasajin, Natalie Erin, Navnir123, NawlinWiki, Neo-Jay, Neurolysis, Nfwu, Nick, Nikthestrung, Nishkid64, Noctibus, Nokkosukko, Nsaa, Ntensis, Nubiotech, NuclearWarfare, Nunquam Dormio, NurseryRhyme, Nutiketai, Nymf, Oasjj, Oda Mari, Odie5533, Ohannaiw, Ohnoitsjamie, Omicronperseis8, Otend, OverlordQ, Pandacomics, Paronomia, ParisianBlade, ParkingBack69, Pathwrote, Paulii48, Pepper, Persian Poet Gal, Peter3838, Petrb, Pharaoh of the Wizards, Phil5798, PhilCAD, Philip Trueman, Piano non troppo, Piast93, Pinethicket, Pingveno, Pinkyhoghan, Pockets117, Poopman45, Possum, Preety101, Prestonmag, Ptcamn, Pulameingurata, Purpley, Pyfan, Qaddosh, Quadrell, Quintote, QwerpQwertus, Qxz, R'n'B, R. S. Shaw, REGGIE49, RHaworth, RJaguar3, Radon210, Raganaut, Ran, Ranveig, RazorICE, Reach Out to the Truth, ReinforcedReinforcements, Renaissance, Retired user 0001, Riana, Rich Farmbrough, Richard001, Rivertorch, Rjanag, Rnb, RobertG, Robert Edberg, Ronjhones, Rostz, Rrburke, S19991002, SJID914, ST47, Salmar, Sarah, Sarjan100, Schac5, SchfiftyThree, Scientizz, Setthon, Shadowjams, Shaficlus, Shanes, Shapejsr, Shuckless, Simba Jones, Simetrical, Sjakkale, Sk 566, Skunkboy74, Skyy Train, SlimVirgin, Sluzzelin, Smart or not, Smooth O, Snowolf, SoCalSuperEagle, SofieElisBexter, Softballk001, Some jerk on the Internet, Sopher99, Space rock 500, Special-T, Stefanoime, Stephenb, Stevertigo, Stickynips, Strait, Subdolous, Sue Rangell, Sunny910910, Sunnybayz, Sunshine222, Swaq, Syllabology, Symane, Synchronism, TAKASUGI Shinji, THEN WHO WAS PHONE?, Tacolilygirl, Tbhotch, The Illusiva Man, The Rhymesmith, The Thing That Should Not Be, The undertow, TheEnderDragon, TheLedBalloon, TheRedPenOfDoom, Theasad.ali, Theeditoroflace, Thefactmasterofchucknorris, Thinboy00P, Thinggg, Think outside the box, Thomas Skogestad, Throwaway85, Tiddly Tom, Tide rolls, TigerShark, Tigershrike, Timothybeasley, Tinkuxln, Titoxd, Tommy2010, ToomanyTresses, Treisjs, Trevor MacInnis, Triona, Tulsoeh, Twin Bird, Ujean2000, Umofomia, Unapiedra, User name 12, Van helsing, Vanished user 5zariu3jisj04irj, Velella, Velho, Versus22, Vickwick2, Vickihibbard, Vishnava, Viskonsla, Vlma11, Vpiercy, Vrenator, Wagggers, Wakebrdkid, Wakuran, Warhorus, Wayne Slam, Wayward, WhatamIdoing, Whispering, Whoffmann, Whym, Wickey-nl, Wikidudem, Wikipelli, Wikiscar, William Avery, Wint, Wine Guy, Woohookitty, Wtmitchell, X-ectioner, XStatizX, Xionbox, Xmsdragonovfire, Xunex, Xyzzyplugh, YellowMonkey, Yes, not yes, Yesisgood, Zad68, Zaxter321, Zerida, Zhou yi777, Zirland, Zondor, كاشف غليل, 993 anonymous edits

Morphology (linguistics) Source: <http://en.wikipedia.org/w/index.php?oldid=525107022> Contributors: AK IM OP, Abrasha, Ahoerstemeier, Alansohn, AnatNinio, Anclation, Andre Engels, Antonio Lopez, ArnoldReinholt, Arthena, Astronautics, Atatnenu, Auf, Azalea pomp, Babbage, BabelStone, Baritic88, Bd848, Beland, Binadot, Bookgrrl, BrainMagMo, Brunton, Bryan Derksen, Can't sleep, clown will eat me, CanisRufus, Caoimhin, Cesine, Chrislk02, Cnilep, Conversion script, Corvun, Cwkmail, Dale Germann, David.Mestel, Dbachmann, Deagle AP, Democracy to information, Electicos, Ed menendez, Eequor, Ekotkie, El C, Elduderino, Enmajybee, Emw, Estevoaei, FelipeS, Folajimi, Francopoulou, Furykef, Gabnh, Galaxiaad, Garik, Gilliam, Glen, Gonzonoir, Greysonyakabooty, Grritchka, Grstain, Gurch, Gutsul, Hannes Hirzel, Henrik.petaisto, Hirzel, Hoary, Hurmata, Hvn0413, Ish ishwar, J. Spencer, JLC2, Jaberwocky6669, Jagged 85, Jamacfarlane, JamesBWatson, Jburr1985, JesseRafe, Jmabel, Jo3sampl, Jobber, Joehall45, Joelmills, Joren, Jorge Stolfi, Joriki, Jpatokal, KF, Kekoconnor, KingTT, Ko'oy, Koty1905, Kusunose, Kwamikagami, L Kensington, La goutte de pluie, Lam Kin Keung, Laudaka, Laurens-af, LinguitAtLarge, Linguizic, LittleDan, Lophotrochozoa, Loudenvier, Lukedpotter, MC MasterChef, MIT Trekkie, MPPerel, MacedonianBoy, Magnus Bakken, Marnalen, Mdoff, Megalophias, Mike Dillon, Mindmatrix, Montrealais, Morta0918, Mrg3105, Nadia007ss, Nickshanks, Nohat, Nyttend, Pablo-flores, Pandamonium, Pascal.Tesson, Patche99z, Pedant17, Pi zero, Poppy, Pádraic MacUidhir, Quntilian, R'n'B, Rajiv.linguistics, RavShimon, Rbeard, Rcj007, Rik G., Rjanag, Rorrenigol, Ross Burgess, RoyBoy, Rror, Ruakh, Ryguasu, SRAnderson, Sacundim, Scalise, Shadowjams, Shinju, Silverhand, Sirkickdon, Skittleys, Spellbinder, Spellcast, Stevertigo, Styfynsemsons, Suisui, Sundar, Sverdrup, Svick, Sólyomszem, TFCforever, Tamfang, Tbhotch, Template namespace initialisation script, Tide rolls, Tkeu, Toon05,

Tpbradbury, Tregoweth, Trickstar, Trio, Trondtr, Tvarnoe, Twenex, Ulric1313, Universityuser, Versus22, Victor Yus, Violetriga, Vivacissamamente, Waverley2000, Whimemsz, Wiggin15, WikHead, Wiki alf, Wleman, Wragge, Yahya Abdal-Aziz, Zeman, Zerunagerous, 217 anonymous edits

Syntax Source: <http://en.wikipedia.org/w/index.php?oldid=525506745> Contributors: 16@r, 194.230.160.xxx, 195.186.148.xxx, 2bluey, 62.202.117.xxx, AK IM OP, Acb4341, Action potential, Adw2000, Ahoerstemeier, Airborne84, Aitias, Allan McInnes, Altenmann, Andre Engels, AndyCjp, Angr, Angryafghan, Anomalocaris, Anthony Appleyard, Atlant, Aurista25, Axeman89, Beige Tangerine, Beland, Bellonian, Ben-Zin, BiT, BigBangTheoryLad, Bobo192, Bohey97, Bogger, Brian the Editor, Byelf2007, C.Fred, CJLL Wright, CR7, CWenger, Cadr, Can't sleep, clown will eat me, CapnPrep, Cc116, Charles Matthews, Cherry blossom tree, ChrisGualtieri, Chromotox, Ciceronl, CityOfSilver, ClosedEyesSeeing, Cnilep, Coffeassured, Comhreir, Conversion script, Cquan, Cromwellt, CyriThePig4, D6, DARTH SIDIOUS 2, David Kerno, David.Mestel, DerHexer, Derek farn, Discospinster, Dittaeva, Divespluto, Dmerrill, Donald Albury, Dude1818, Dungodung, Dwspig2, Ed g2s, Eilthreach, El C, Eprb123, Erutuon, Fattyjwoods, FilipeS, FlyHigh, Freedominlux, FreplySpang, FromFoamsToWaves, Fuzzypieg, Gbouzon, Ghiraddje, Giftlite, Gilliam, Glen, Glenn, Goudron, Graham87, Grahas02, Gregbard, Guillaume2303, Hadal, Haggeng Kennedy, Hannes Hirzel, Haukurt, Hectorthebat, Hexham, Hippasus, Hippietrail, Hirzel, Hulten, Hyacinth, Hyad, IRP, Icke, Illovereandrea, Ish ishwar, J Crow, J Spencer, Jasy jatere, Jauhieni, Jbeans, Jcarroll, Jdavid, Jeff G., JeremyA, Jerry, Jimbo2222, Jmonk95, Joanjoc, JoanneB, Jobber, Joe Jarvis, Johan1298, John254, Jon Awbrey, JonnybrotherJr, Jonsafari, Joseph Solis in Australia, Joshuagross, Jwestbrook, K.lee, KEJ, Kajervi, Kallerdis, Kappa, Karl E. V. Palmen, Karl-Henner, Keta, Kjkollb, KnowledgeOfSelf, Kokoriko, Konfan71, KrakatoaKatie, Krash, Kwertii, Leuko, Linguizic, Looxix, Lpjurca, Luna Santin, MER-C, Mactet, Mainstreet27, Malhonen, Mark Dingemanse, MarkSweep, MarkkuP, Marnanel, Martin Hinks, Maunu, Mav, Mazeface, McSly, Mggreenbe, Michael Hardy, Miguelmrr, Mike Klaassen, Mikeo, Mindmatrix, Minna Sora no Shita, Morganaci, Mr.Z-man, Mrcool1122, Mukerjee, Mundart, Mycomp, N-k, NatusRoma, Neile, Neutral-en, Nguyen Thanh Quang, Nick Number, Nihil novi, Niro5, Novangelis, Numb03, Nuno Tavares, Nyoro n, Ojigiri, OlEnglish, Omnipaediast, Oneiros, OverlordQ, Oxymoron83, Palica, Pasta Salad, PaulTanenbaum, Pdblues, Physicist, Pi zero, Pigman, Pleckaitis, Poccil, Polyvios, Pprabhakarao, Prabhakar P Rao, Qetutu, Quangba, R'n'B, RJFJR, RandomP, Raymond Meredith, Reaper Eternal, RedWolf, Rich Farmbrough, Rijkbenik, Rik G., Rivertorch, Rjanag, Rlevse, Rmosler2100, Robin klein, Robzy123, Rollred15, RoseParks, Rurusu, Rwessel, SJP, Semmelweiss, Simetrical, Skal, Skyler, Smokizzy, Sonjaaa, Sopher99, Space Dracula, Spellbinder, Springindd, Starfire777, Stemonitis, Stevemitchell, Stevertigo, Stifynsemons, Sverdrup, Tautologist, Tbhotch, Template namespace initialisation script, TheRingess, Thom2002, Thomas Blomberg, Thomasmeeks, Tide rolls, Tijfo098, Tim Q. Wells, Tjo3ya, Tooki, Torgo, Triplejo2, TuukkaH, TwigsCogito, Vaganyik, Versus22, Viriditas, Vishnava, Wavelength, Wayiran, Whimemsz, Will Hen, William Avery, Willking1979, Wiredrabbit, Wknight94, Wooddoo-eng, Woohookitty, Wutsje, Xpus4321, Yaxu, Yerpo, Youssefsan, Zack wadghiri, Zarel, Zerida, ÅEk, 379 anonymous edits

Phonology Source: <http://en.wikipedia.org/w/index.php?oldid=525676805> Contributors: 13alexander, 65.68.87.xxx, 96.186, Achowat, Aeuso1s1, Alinguist, Alipir, Altenmann, Anarkisto, Andre Engels, Andrewpmk, Andycjp, Angr, Atkinson 291, Avjoska, Benjamin9832, Benwing, BiggerAristotle, Bissinger, Bobby D. Bryant, Brad7777, Brevani, Burzmal, CCLS, Cadr, Calliopejen1, CapnPrep, Cassowary, Cervello84, Charles Matthews, Cntras, Conversion script, CyborgTosser, D6, Damian Yerrick, DanielCD, Denihilonhil, Denisarona, Diego UFCG, Djimn112, Dopependek, DuoduoDuo, EagleFan, Edward, El C, Eleassar, Erutuon, Felix Folio Secundus, Flamingspinach, Fon, Freelance Intellectual, Freethinker77, Friviere, Furykef, Gaiitb, Galoubet, Gandalf1491, Geoffrey S. Nathan, Ghirlandoj, Giftlite, Gjm, Godfrey Daniel, Gogo Dodo, Grblomter, GregLee, Gum375, Haham hanuka, Hannes Hirzel, Hirzel, Honeybop, Hvn0413, Indeterminate, InfoCan, Ioscius, Ish ishwar, IstvanWolf, J Crow, J. 'mach' wust, J.delanoy, Jagged 85, Jaicheter, JdeJ, Jlittlet, John Riemann Soong, Jonik, Jonnabuz, Joost.b, Jrdioko, Jsteph, Junglecat, KBYU, KakistocraticLaw, Kaobear, Karmosin, Kevin Steinhardt, Khoikhoi, Kikos, Kku, Kronecker, Ks 7508, Kwamikagami, Kyaoaka, LaggedOnUser, Lee Daniel Crocker, Lightmouse, LittleDan LittleWink, Livajo, LokiClock, MER-C, Mack2, MagneticFlux, Makrai, Mayumashu, Merfster, Mindmatrix, Mingwangx, Miskwito, Mogism, Mrg3105, Msanford, Mxn, Mzajac, N-true, Nikai, Ninly, Nstringer, Obradovic Goran, Oghmoir, Onco p53, Pauleldacy, Paxse, PeepP, Pengkeu, Phil webster, Pinethicket, Pirkshala, Poccil, R'n'B, Rich Farmbrough, Richwales, Rik G., Rjanag, Rjwilmsi, Rosanne, Ryguasu, Sabraw1, Saterry, Scipius, SebastianHelm, Setok, Shlomital, Sirmylesnagopaleentheda, Spellbinder, SpuriousQ, Squidley, Stevertigo, Stifynsemons, Sunvgirl, SuperativeHors, Swooningdisaster, Syllabology, Tassedep, Template namespace initialisation script, The vili ghost, Thincat, Thnidu, Tim Q. Wells, Tobby72, Tom Hope, Trusilver, Veella, Velho, Victor Yus, Wavelength, WikHead, Wikipedior40, Wikiwow, WmGB, Wooddoo-eng, Wotnow, Xact, Zack wadghiri, Zanimum, Zerida, KEKPQΩ², 200 anonymous edits

Morphophonology Source: <http://en.wikipedia.org/w/index.php?oldid=523329421> Contributors: Aeuso1s1, Andycjp, Angr, AuburnPilot, Auntnof, Caeruleantaur, Carlog3, Cathfolant, Conversion script, Deflective, Ergative rlt, FilipeS, Gap9551, Hvn0413, Imz, Inkstersco, Ish ishwar, IvanLanin, J. Spencer, Jwestbrook, Kwamikagami, LilHelpa, Lockesdonkey, Mairi, MattPres, Nbarth, R'n'B, Ram-Man, Robofish, SimonP, Tjdw, True, Victor Yus, Vu, Welsh, Woohookitty, 26 anonymous edits

Phonetics Source: <http://en.wikipedia.org/w/index.php?oldid=525119472> Contributors: :.Avvol:, 13alexander, 2D, 65.68.87.xxx, Abc518, Aeuso1s1, Alx3762, Andre Engels, Andres, Andycjp, Angr, Antandrus, Aref Saki, Avb, AxSkov, BRG, Bbtself, Belovedfreak, Bjankuloski06en, Borgx, BrainMarble, Branddobb, COMPATT, Capricorn42, Carey Evans, Chaojoker, Chinasa, Colin Keigher, Conversion script, Crazycomputers, Cult-p, D, Daf, Dairyqueen8, Damian Yerrick, Darrt Panda, DePiep, DennisDaniels, Dhirsbrunner, Dinesh smita, DionysiusThrax, Discospinster, Dissident, Djj6058, Dominic.sedghi, Dylansmjones, Dynablastter, Eleassar, Emperorbma, Eniw, EncycloPetey, EugeneZelenko, Femto, Fgdfgdsfghrrfro7g756, FilipeS, Flyingidiot, Friviere, Fryed-peach, Future Perfect at Sunrise, Ganymead, Garant^A, Garik, Gil mo, Gpkh, Graham87, Grahamgraeme, Grebaldar, Guaka, Gum375, Gurt Posh, Hairy Dude, Hannes Hirzel, Hari, Hearnoseenospakno, Heron, Hirzel, HistoryBA, Hm423, Hmain, Homerjay, Hydrogen Iodide, Igor, Imiraven, Inbetweener, Iohannes Animosus, IronGargoyle, Ish ishwar, Ishkeno, Italianoinspirti, J-Wiki, J.delanoy, JH-man, Jagged 85, Jammie2, Jimp, Jonik, Jph, Jrtayloriv, Kaobear, Karl Palmen, Karmosin, Kenny sh, Kesla, Kevin Russell, KevinMcGowan, Khoikhoi, Khym Chanur, Kikos, Kimmelin, Klow, Kokoriko, Ks 7508, Kwamikagami, LOL, Lalala171717, Lament, Lingboy, LittleDan, Logixoul, Lukasstad, Luna Santin, LunaticFringe, Lupin, MER-C, Majmun, Man1, Mark Dingemanse, Markmark12, Marnanel, Masatran, Massimo Macconi, Matusz, McGeddon, Mcld, Menchi, Mindmatrix, Mizike, Monstre, MrOllie, N-true, NHRHS2010, Nasnema, Natzi Boy, Nilfanion, Niteowlneils, Noahpoah, Nohat, Nono64, Nppos, Oda Mari, Ojigiri, Olivier, Ortonmc, Oscarthecat, Paddy O'Scallion, Park jae-hoon, Patrick, Perezkelly, Phil wink, PierreAbbat, Pigman, Piotr Gasiorowski, Pjacobi, Portalian, Postmodern Beatnik, Quangbao, R'n'B, Ragib, Rajnisshkmr083, Raymond Meredith, Rjwilmsi, Romann, Roylee, Royote, RucasHost, Rygelski, Ryguasu, Secondwatch, Simeondahl, SintonX, SiobhanHansa, Sivara, Slomo, Sonjaaa, Sparky147, Spellbinder, Smec, Stephen, Stevertigo, Sunborn, THEN WHO WAS PHONE?, Tbackstr, Template namespace initialisation script, TenPoundHammer, The Anome, TheJrlinguist, TheOtherBob, Timschmi, Tobyy72, Tom.Reding, Topace, Twsx, Umofovnia, Vaganyik, Vahagn Petrosyan, Venu62, Vssun, Wasbeer, Wikikrlsc, Wikiwow, Ygwnkm, Yidisheryid, Zerida, Zhangshuo517, KEKPQΩ², مسندی, 276 anonymous edits

Semantics Source: <http://en.wikipedia.org/w/index.php?oldid=525570655> Contributors: 16@r, 2bluey, 8ung3st, A. B., AGTOTH, AKA MBG, AVand, Aaeamdar, Aboctok, Active Banana, Aenioc, Aherandez33, Alanthehat, Aleksd, Amos Han, AnakngAraw, Analogisub, Andres, AndriesVanRenssen, Aniketaladal, Anomalocaris, Anthony Appleyard, Asderff, Augur, Awi007, Ayatiniazi, Azymuthca, Battamer, Ben Babcock, Ben-Zin, BenSmak, Benjamin Dominic, BGuest, Boblibr, Bornslippy, Brightorange, Byelf2007, Cacycle, CanadianLinuxUser, Cdc, Cenarium, Chalst, CharlesGillingham, Chris the speller, Cnilep, Comicussons, Conversion script, Cybercobra, Cyborg Ninja, DEddy, Dalahast, Danny lost, Dave5702, Daveh1, Dbachmann, Dbiel, Dcooper, Dmccreary, Doc honcho, Donald Albany, Droll, Dylan Lake, Edunoramus, Eequor, Eklir, Ektokie, El C, Emptymountains, Enric Naval, Eric Biggs, Erick.Antezana, Erutuon, EvenT, Everyking, Evice, Förbidik, FilipeS, Frap, Friginator, Fvw, G. Lakoff, Gabnh, Galaxiaad, Gandalf1491, Gdarin, George100, Gimmetrow, Gioto, Glenn, Goberiko, Govindmaheswaran, Graham87, Gregbard, Guoguo12, Gwalla, HHirzel, Hairy Dude, Hannes Hirzel, Harold Philby, Haukurth, Hectorthebat, Helix84, Heron, HillarySco, Hpvp, Hu12, Hyggelig, Iblardi, Immersion, Ioscius, Ish ishwar, Isotope23, J Di, J.delanoy, JAHandler, JC Chu, JECompton, Jaberwocky6669, Jackol, Jakohn, Jason L. Gohlike, Javier Carro, Jeffi, Jens Meiert, Jerryobject, Jim1138, Jimbo2222, Jsite Niesen, Jivecat, Jmcchangers90, Joan, Joanjoc, Jon Awbrey, Jon Roland, Jonkerz, Jonsafari, Jost.b, Jooyoonchung, Jpta, Jtauber, JustinCOPE82, K1US, Kensall, Kevin12xd, Kgoarany, Khalid hassani, Khoikhoi, Kiwibird, Kku, Kntg, Koaf, Kokoriko, Krixou, Kzzl, Lambda, Lambyuk, Landon1980, Lawandeconomics1, Levelay, Light current, LiHelpa, Lindosland, Linkoman, Llamas4drama'10, Logan, LokiClock, Looxix, Lotje, Lova Falk, Lrunge, Lucidish, Lygophile, Lynch9000s, MER-C, MK8, Macedonian, MacedonianBoy, Mahitgar, Mandarax, Marc Venot, Mark Dingemanse, MarkBrooks, Markus Krötzsch, Martarius, MasterofUnvrs314, Mbell, Mchcopl, Mechanistic, Merijn2, MiNombreDeGuerra, Michael Hardy, Micmacete, Mindmatrix, Mjklin, Mlinar, Moink, Molewood6, Mukerjee, Mundart, Nabeth, NawlinWiki, NeoAmsterdam, Netesq, Nick Number, NickelShoe, Nickleus, Nicodemus13, Nora lives, Nortexoid, Nubiatech, Omnipaediast, Ortolan88, Ott, PaulColby, Philip Trueman, Piercertheorganis, Pisongthewing, Pinethicket, Pirkshala, Pit, Pred, Purpose Observatory, Quibile, Rama, RasputinJSvengali, Rdanneskjold, Ready, Redvers, Retodon8, Revery, Rich Farmbrough, Rjanag, Rjwilmsi, Rod57, Rossami, Rp, Runch, Rurusu, Ryguasu, SDC, ST47, Sabrebattletank, ScAvenger lv, Scwlong, Semantia, Seriv, Shamalyug, Shaw wiki, Shermanmonroe, Shizhava, Simlorie, Sinatra, Sir Vicious, Sliogocki, Slipstream, Smithonian, Sonic Mew, Spellbinder, Squarrels, Stephen, Stevertigo, Stifynsemons, Strife911, Suradatashandilya, Sverdrup, Template namespace initialisation script, The Anome, The singapore ministry of education sucks, Theyetiman12345, Tide rolls, Tijfo098, TimVickers, Tmusgrove, Tom Pippens, Tycho, UnconsciousInferno, Urhixidur, Vaganyik, Vanisheduser12345, Velho, Vonkje, WAS 4.250, Warhorus, Wavelength, Wikid77, Wolfdog, Woohookitty, X201, Yerpo, Youssefsan, Yuriz, Zegarad, Zerida, Zhen Lin, Ziusudra, Zsinj, گانش غل، 356 anonymous edits

Pragmatics Source: <http://en.wikipedia.org/w/index.php?oldid=525155338> Contributors: 1ForTheMoney, 2bluey, Adam78, Ahoerstemeier, Alexkon, Allens, Anaphalis, Andre Engels, Andreastokke, Angr, Antonielly, Ap, Ashfan83, AustralianMelodrama, Bdesham, Belovedeagle, Byelf2007, C. A. Russell, Camw, Chris the speller, ChrisGualtieri, Cnilep, Cquan, Crónica, DASonnenfeld, DCDuring, Dasternberg, Davidjobson, Dduck, Dear ODear ODear, Defyn, Denisarona, Leonord, Dmacw6, Dorvaq, Dúnadan, Eaeftremov, Easterboxx, Edhubbard, Edward, ElKevbo, Eniarrol, EnisSoz, Erud, Facopad, Favanian, Flyte25, Fowler&fowler, GTBacchus, Garik, Giraffedata, Granitethighs, Grape1, Gregbard, Hakeem,gadi, Hamsterlopithecus, Hong12kong, Hu12, Islievelve, Inwid, Ish ishwar, Itoula, IvanLanin, Jacksonflint, Jdenen, JimD, Jon Awbrey, Jerv, KC 12321, Kennetha, Khoikhoi, Kku, Knotwork, Ko'oy, Kyaoaka, LKNUTZ, Lindsay658, Lingrace us, Livajo, Lucidish, Luna Santin, MER-C, Maashatra11, Macedonian, Marek69, Matturn, Mekanik78, Michael Hardy, Michiklas, Micmacete, Mike hayes, Milais, Mindmatrix, MissFrappell, Mr. Stradivarius on tour, Msasscts, Mu, Musiphil, Ngio, Notinasaid, Ntensis, Nurg, Omnipaediast, Passing mouse, Peachlette, Philip Trueman, Pitan, Pupster21, Rdsmith4, RedHouse18, Reedy, Reinhard Hartmann, RekishiEJ, Reneeholle, Rjanag, Rsrikanth05, Rurusu, Samsara, Sethmahoney, Sinatra, Smartre, Smokesomedillweed, So-called Genius, Solute, Spellbinder, Stevertigo, Stifynsemons, SummerWithMorons, Superabo, Supriya, Szyslak, Taak, Teamprag, Template namespace initialisation script, The Anome, The singapore ministry of education sucks, Theyetiman12345, Tide rolls, Tijfo098, TimVickers, Tmusgrove, Tom Pippens, Tycho, UnconsciousInferno, Urhixidur, Vaganyik, Vanisheduser12345, Velho, Vonkje, WAS 4.250, Warhorus, Wavelength, Wikid77, Wolfdog, Woohookitty, X201, Yerpo, Youssefsan, Yuriz, Zegarad, Zerida, Zhen Lin, Ziusudra, Zsinj, 177 anonymous edits

Formal grammar Source: <http://en.wikipedia.org/w/index.php?oldid=526493731> Contributors: ABCD, AHMartin, Accounting4Taste, Adriatikus, Agnus, Ahmedmasud, Andre Engels, Ankog, Anonymous Dissident, Archelon, Arman Cagle, Arthur Rubin, Babajobu, Beland, Bernhard Bauer, Bilbo1507, Bonadea, Brynosaurus, CBM, Canadian-Bacon, Captain-n00dle, Chemica, Chopchopwhitey, Conversion script, Damian Yerrick, Daniel Mahu, David Latapie, Dbachmann, Derek Ross, Djordjes, DopefishJustin, Dr.Halfbaked, Dreadstar, Emet truth, ErikErnst, Excirial, Ezrakilty, False vacuum, Florian D., Fniesen, Fyrael, Gandalf61, Gantlord, General Wesc, Giftlite, Gmlk, Gregbard, Gutsul, Ilya Voyager, Isilando, Jafet, Jan Hidders, Jaredwf, Jellystones, Jiri

1984, Jonsafari, JulesH, Kaihsu, LC, LKenzo, LeonardoGregorian, LiDaobing, Linas, Lmatt, Lokentaren, Mean as custard, Methossant, Mhss, Michael Hardy, Mitrius, Mpah, Mr.internete, Myria, Nbarth, Nick Number, Nicke L, Nxavar, Patrick, Pcap, QTJ, R.e.s., Reedy, Rgamble, Rjwilmsi, Rp, Ruakh, Ryan256, Savantas83, Schneelocke, Schweiwikist, Shahbaz Youssefi, Shenme, Simeon, Smalljim, Splrahul, StephenFerg, Stevertigo, Tamarkot, The Anome, Tyler McHenry, UKoch, Unyoyega, Upedia, Vantelimus, Vjswarna, Wlievens, Zearin, 79 anonymous edits

Formal language *Source:* <http://en.wikipedia.org/w/index.php?oldid=525114401> *Contributors:* 24.60.209.xxx, Aaron Schulz, ActivExpression, Adriatikus, Ahoerstemeier, Akerbos, Alpha Beta Epsilon, Andre Engels, AndriesVanRenssen, Ankog, Archelon, Architectchao, Arthur Rubin, Astuhsin, Bemocia, Bkil, Bonadea, Brest, CBM, CBM2, CRGreathouse, Cerebralpayne, Chalst, Charles Matthews, Charvest, Cic, Classicaelecon, Cleduc, Clickey, Connelly, ConradPino, Conversion script, Cuaxdon, DBooth, Daniele.tampieri, Danny lost, David Eppstein, Dbachmann, Dmuniene, Dreftymac, Dzidiagba, Eequor, Excirial, Ezrakily, FelisLeo, Fikus, Finell, Flamingospinach, Frap, Fratrep, Gards, Giftlite, Gregbard, Gzorg, Hahahaha4, Halukakin, Hans Adler, Hebe2000, Helix84, Hermet, Incnis Mrsi, Iridescent, Isthatme, J Spencer, J.delanoy, JNW, JakobVoss, Jan Hidders, Jaredwf, Jim1138, Jokers, Jon Awbrey, Jonsafari, Jpvinall, Kim Bruning, LC, LKenzo, Linas, LoStrangolatore, Lokentaren, MarkSweep, Mdd, Mean as custard, MiNombreDeGuerra, Mike Fikes, Milloss, MithrandirMage, MrTree, Mukerjee, Muu-karhu, NERIUM, Nanshu, Newbyguesses, Nick Number, Nk, Ntensis, NuclearWarfare, Nuno Tavares, Obradovic Goran, Od Mishehu, OIEnglish, Pcap, Pexatus, Philogo, Philomathoholic, PlatypianArchcow, Quercus basaseachicensis, Rich Farmbrough, Rick Norwood, Ripper234, RobinK, Rp, Saric, Schneelocke, Sdorman, Serberimor, Sff9, Spikey, Stephan Leeds, Tedickey, ThomasOwens, Tijfo098, Tobias Bergemann, Trusilver, TuukkaH, Tyler McHenry, Unyoyega, Vantelimus, VictorAnyakin, Waku, Wavelength, Wei.cs, Wic2020, Youandme, 97 anonymous edits

Formation rule *Source:* <http://en.wikipedia.org/w/index.php?oldid=496467650> *Contributors:* Arthur Rubin, EmilJ, Giftlite, Gregbard, Hans Adler, Kbdkn71, Michael Hardy, Tahu88810, The Wiki ghost, Timrollpicking, 1 anonymous edits

String (computer science) *Source:* <http://en.wikipedia.org/w/index.php?oldid=526398628> *Contributors:* 216.60.221.xxx, A4b3c2d1e0f, Ahoerstemeier, Alai, Alan Millar, Alfabalon, Andreas Kaufmann, Andres, Andrew Helwer, Andy Dingley, Anphanax, Anthony Borla, Arthur Frayn, AxelBoldt, B4hand, BIL, Beland, Bevo, Bkkbrad, Black Falcon, Bogdangiusca, Boleslav Bobcik, Borgx, Bryan Derksen, BurnSky, C45207, CanadianMaritime, CanisRufus, Castaa, Cedders, Charles Matthews, Charvest, Chris the speller, Christopherlin, Conversion script, Courcelles, Cybercobra, Dainomite, Damian Yerrick, Dcoetze, Derek farn, Doctorfluffy, Doug Bell, Dreadstar, Dreftymac, Drj, Drphilharmonic, Dysprosia, Elassint, Eloquence, Error, Fabartus, Fropuff, Fudo, Furrykef, GNRY09, Gaiaacara, Garyzx, Georg Peter, Ghewill, Giftlite, GoingBatty, Gparker, Gregbard, Gurch, Gwyl, Gyro Copter, Hairy Dude, Hippophaë, Hornlo, Howcheng, IOLJeff, Ian Pitchford, Jcep, JD90, JLATondre, Jay-Sebastos, Jc3s5h, Jeremysr, Jiri 1984, Jncraton, John254, Jonnabuz, Jordandanford, Kbdkn71, Koyaanis Qatsi, Kubigula, Kusunose, Kyng, Lambiam, Linas, Loadmaster, Local.electric, Luke Igoe, Mad Tinman, Marc van Leeuwen, MattGiuciua, Maximimax, Michael Hardy, Mikeblas, Minghong, MisterSheik, Mjb, Mkweise, Mpkr, Murray Langton, Murtasa, Nasnema, Nbarth, Neelix, Nevyn, Obradovic Goran, Oleg Alexandrov, OpinionPerson, Orderud, Pantser, Patrick, Pengo, Perique des Palottes, Peterdjones, Pexatus, Philg88, Pimlotte, Pinguin.tk, Plugwash, Prm, Ptarjan, R. S. Shaw, RTC, Richard W.M. Jones, RogerofRomsey, Rory O'Kane, Ruud Koot, S.Örvarr.S, Scarboy, Sebbe, Seec77, Sewing, Shahab, Shirifan, Sietse Snel, Slady, Spearhead, Spitzak, Stephen Gilbert, StuartBrady, TBloemink, Taemyr, TakuyaMurata, Teles, The Anome, The Thing That Should Not Be, TheIncredibleEdibleOompaLoompa, Thumperward, Tigrisek, Tompsc, Tortoise3, Treekids, Ubermonkey, Underrated! 17, Urod, Utif6429, Vadmium, Wayfarer, WinterSpw, Witharebelyell, Zundark, Александър, 133 anonymous edits

Formal semantics (logic) *Source:* <http://en.wikipedia.org/w/index.php?oldid=522559841> *Contributors:* Accounting4Taste, AnAj, Arthur Rubin, BD2412, Brad7777, Byelf2007, Chalst, Cybercobra, Doctorambient, Firstwingman, Gamewizard71, Gf uiip, Giftlite, Gregbard, Hans Adler, Jpbown, Kumioko (renamed), Lambiam, Mhss, Nortexoid, Philogo, Porcher, Robin klein, Rp, StefanMangold, Stevertigo, Tabor, Thv, Thüringer, Tijfo098, Wareh, Woohookky, 7 anonymous edits

Finite-state machine *Source:* <http://en.wikipedia.org/w/index.php?oldid=526485932> *Contributors:* 0x6adb015, 1exec1, A3 nm, A5b, Adam majewski, Adamd1008, Ademkader, Adrianwn, Aimaz, Alotau, Am117, Amalas, Amux, AndersL, Andrei Stroe, AnnaFrance, Anrie Nord, Antonioli, ArmandinGeneral, Armandeh, Art LaPella, Arvindn, Ashishpurna, Ashutosh y0078, Atlys, Avengerx, AxelBoldt, Bensin, Bergsten, BIT, Bkkbrad, BlueAmethyst, BlueNovember, Bmiconp, Bnognent, Borislav, Brech, Bruyninc, CAKira, CBM, Calrefa Wéna, Charles Matthews, Chetvorno, Chrishmt0423, Colin Marquardt, CommonsDelinker, Confliqk, Conversion script, Courcelles, D6, DFRussia, Daghall, Damian Yerrick, Darkmeerkat, David Eppstein, DavidCary, Davidbspalding, Dcoetze, Deleron, DerBorg, Dh1509, Dinesh.lei, Discospinter, DomPrice, Don4of4, Dough, Dr. Bruce W. Watson, Drebs, Dysprosia, Dzordzm, Easwarno1, Ec79, EnDunEm, Enochhwang, Eubulides, EvanED, Evercat, Evgeny.berkovich, Fake0Name, Frap, Fred Bradstadt, Fredrik, Fresheenesz, Furrykef, Fvw, GRAHAMUK, Gaius Cornelius, Gargan26, Gedeon, Geni, Giftlite, Glrx, GoodStuff, Gpvos, Graham87, GregHolmberg, Gwyl, Hairy Dude, HalHal, Hamaryns, Hermel, Heron, Hervegirod, Hirzel, Hotfeba, ILike2BeAnonymous, Ianalis, Iancine0, Intelliw, J.delanoy, JamesBWatson, Jan Hidders, Jaredwf, Javalenok, Jaxl, Jazccello, Jeepday, Jerome Charles Potts, Jeronimo, JesseW, Johann Peter Dirichlet, Jonik, Jonsafari, Joris Gillis, Joseph Dwayne, Jpbown, Justin W Smith, Kakesson, KamasaamK, Khendon, Kimiko, Kishan247, Knowledge Seeker, LOL, Linas, Linus M., LionKimbro, Looxix, LordArtemis, Lukman Sasmita, M'luqomzXb, Mac c, MagiMaster, Male1979, Manu31415, Marek69, MarkSweep, Martarius, Martynas Patasius, MattBan, Mattoothman, Maurice Carbonaro, Mav, Maximimax, Mcarone, Mcaroni1, Mdd, Mesepitiamus, Mezmeriseu, Mhwang2002, Michael Devore, Michael Hardy, Mirosamek, Mormegil, MrOllie, Mynameisnotipj, Nakon, Nateman1352, Neilc, Nick Pisarro, Jr., Nosebud, Ollydbg, Oravec, Paulbmann, Peets, Petr Kopač, Pgallert, Phil Boswell, Pj 1974, Pnn, Poccil, Prmacn, Qwertys, RDBury, RaCha'ar, Radiojon, Raul654, Ray Van De Walker, Rdancer, Reedy, Rememberfun, RevRagnarok, Rich Farmbrough, Richard.astbury, Riki, Robert K S, Rrohbeck, Ruud Koot, S.K., Sakurambo, Salsa Shark, Sam Staton, Sansepur, Saric, Schultkl, SheldonN, Simonneur, Sir Nicholas de Mimsy-Porpington, Skinkie, Skittles, Snafflekid, Snoyes, Sperxios, Stannered, Stepa, Stewartadcock, Stormy56, SvendTofte, TakuyaMurata, TheArguer, Thing, Thowa, Thsgmn, Thumperward, Tijfo098, Timwi, Tobias Bergemann, Urhixdur, Usien6, Vantelimus, Vdm, Vector Potential, Villarinho, Wiggers, Wtshymanski, Wvbailey, Xeno8, Ybungabill, Zahradzacken, Zarboublian, Zeimus, ZeroOne, ۷۰۵, 454 anonymous edits

Automata theory *Source:* <http://en.wikipedia.org/w/index.php?oldid=525412759> *Contributors:* A Generic Reality, Aaron Schulz, Ahmad.shahwan, Ahoerstemeier, Alansohn, Allan McInnes, Andrew Eisenberg, Ankog, Arslan asghar, Ashutosh y0078, AxelBoldt, Bci2, Begoon, Bethnim, Bjankuloski06en, Bookandcoffee, Bzier, CRGreathouse, ChuckHG, Cjlim, Coming, Crystallina, Dcoetze, Deldotvee, Deleet, Deldotved, DerHexed, Dmcq, Donner60, Dudesleeper, Déjà Vu, Ehn, Ericzsiao1, Eslip17, Fluffernutter, FoxLogick, Fudo, Gaius Cornelius, Gareth Griffith-Jones, Giftlite, GlasGhost, Gmb7-NJT WILL, Gonzozon, HRV, Helder.wiki, Helix84, Helptry, Hermel, Hirfrey, Ilyaroz, IvanAndreevich, JHolman, JK the unwise, Jackson, Jagged 85, Jasamayoa, Jcarroll, Jdoe87, Jeff G, Jeffrey Mall, Jibarra, Jim1138, Jimbyrho, Jiri 1984, Jitse Niesen, Jjovanw, Johnbdritten, Jokes Free4Me, Joost, Joseph Solis in Australia, Jpbown, Jpcayene, Jpvinall, Juniuswikia, KSlayer, KSmrq, Kmarinas86, Knverma, Konradek, Linas, Ling.Nut, Lobner, MONGO, Machine Elf 1735, Magnni, Mangledorf, Maple.writes, Mark lee stillwell, MarkSweep, Marudubshinki, MathMartin, MatthewUND, Met mht, Mean as custard, Melidee, Mhwang2002, Michael Devore, Mirshadk, Msoos, Musiphil, Nunquam Dormio, Nuttycoconut, Nxavar, Oddity, Oleg Alexandrov, Omnipaedista, Pinar, Quoth, Qwertys, Rjwilmsi, Ruud Koot, Ryanli (usurped), S h i v a (Visnu), SOMNIVM, Saber girl08, Saforest, Salix alba, Schwarzbichler, Skgak, Sharaar22, Sietse Snel, Silvonen, Snowolf, Spoon!, Spot, Stephen Shaw, Suruena, The Thing That Should Not Be, Thunderboltz, TimBentley, Tobias Bergemann, Toolnut, Ubermonkey, Unbitwise, Valodzka, Vegpuff, Vento, Vojta, Wasbeer, Wdchk, Wjmallard, Yosef Berman, Zahradzacken, Ze Miguel, Zero sharp, Zorakoid, Zundark, ۲۴۱ anonymous edits

Principle of compositionality *Source:* <http://en.wikipedia.org/w/index.php?oldid=522605839> *Contributors:* Aldux, Andycjp, Axt, Btyner, CBM, CarlHewitt, Chalst, Chikuku, Cpiral, Defleck, Deleet, Dicklyon, Donhalcon, Fawcett5, Foobarnix, Gregbard, Gulsacerda, Imz, Kallerdis, Kzollman, Leibniz, Lucidish, Mets501, Nemesis of Reason, Newbyguesses, Nick Number, Nrlsouza, Payrard, Peter Isotalo, Q10, RJCraig, Sinatra, TakuyaMurata, 21 anonymous edits

Parse tree *Source:* <http://en.wikipedia.org/w/index.php?oldid=511060578> *Contributors:* AHMartin, Alan U. Kennington, Alansohn, Angr, Arabismo, Arjun G. Menon, Beland, BenFrantzDale, BrainMagMo, Bryan Derksen, Cadr, Chris857, ChrisGualtieri, David Pierce, Donhalcon, Dono, Dysprosia, Egrifsin, Emperorbma, Extra999, Falstaft, FatalError, Fetchmaster, Frap, Fredrik, Frigoris, Ganymead, Giftlite, History2007, Ioscius, Jafet, JakobVoss, JoeFromrandb, Jonsafari, LOL, Modify, Pps, Qwertys, RJFJR, RMFan1, Ruakh, Ryan Postlethwaite, Smelialichu, Spyard, Stannered, Tamur, The Anome, The High Fin Sperm Whale, Tjo3ya, TuukkaH, Wavelength, Zscout370, 59 anonymous edits

Deep structure *Source:* <http://en.wikipedia.org/w/index.php?oldid=524211946> *Contributors:* Beland, Bertrand Bellet, Brainecology, Burschik, Byelf2007, Cadr, Charles Matthews, Comhreir, Cuaxdon, Dduck, Dysprosia, FrancisTyers, Georgius, GrammarGorilla, Gregorysutherland1, Hephaestos, Hyacinth, Ish ishwar, Jujuatular, Lambiam, Llywrch, Marksweep, Michael Hardy, Michael Sloane, Mr. Stradivarius, Musicalaps, Ninly, Peter Isotalo, RichardMills65, Rp, Ruakh, Scorn, Stevertigo, Supernerd707, The Anome, The Thing That Should Not Be, Torgo, Umofomia, 24 anonymous edits

Ambiguous grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=526538395> *Contributors:* AySz88, Ben Standeven, Darklock, Derek farn, Eric119, Framhein, Furrykef, Gman124, Hermel, Jamelan, Jaredwf, Jdoe87, Joris.deguguet, LOL, Lupin, Nick Number, Nxavar, Phil Sandifer, RJFJR, Ruakh, Ruud Koot, Ryan256, Saxbryn, SimonP, Svick, Tremilux, Trunks ishida, Woohookity, 31 anonymous edits

Phrase structure grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=519964686> *Contributors:* AlexandrDmitri, Burschik, Cadr, ChrisGualtieri, Dominus, Dr Zen, Ganna24, Gregbard, Kyokoao, Linas, Nxavar, Oliverbeatson, Pnm, Rp, Saccade, Tjo3ya, Vantelimus, 6 anonymous edits

Chomsky hierarchy *Source:* <http://en.wikipedia.org/w/index.php?oldid=524911666> *Contributors:* 62.254.0.xxx, Alanlastufka, Aledeniz, Androby, Ankog, Anonymous101, AxelBoldt, BACbKA, Babajobu, Blaisorblade, Bluear, Bn, Bonadea, Borislav, Chris Pressey, Chris Q, Conversion script, Diego Moya, Dominus, EON, Evercat, FHen, FrancisTyers, Fred Bradstadt, GRuban, Garik, Giftlite, Graham87, Gwpl, Hannes Hirzel, InverseHypercube, J. Finkelstein, JakobVoss, Jan Hidders, Jaredwf, Jason Quinn, JeromeKelly, Jeronimo, Jessepeq, Jiang, Jim1138, Jitse Niesen, Johnlogic, Jon Awbrey, Jonsafari, Josh Cherry, JulesH, Kaihsu, Kpmiyapuram, LKenzo, Lemontea, Likeminas, Macaddct1984, Mathias.Peron, Mcewan, Mean as custard, Mhyim, Mikko Paaninen, Morton Shumway, Mountain, Muu-karhu, Nicapicella, Nicuguyedc, Nitin420, Oerjan, Ohms law, Oleg Alexandrov, Olivier, Omnipaedita, Optikos, Paul Magnussen, Prufer, QTJ, Qwertys, Reedy, Rjwilmsi, Robertgreer, Ruud Koot, Saccade, Saravask, Schneelocke, Splrahul, Stevertigo, Tarotcards, TechnoFaye, TheAntiZealot, Timwi, Tobias Bergemann, Torreslfchero, Trondtr, Tuetschek, Universityuser, Xyzzy n, Zahnradzacken, 122 anonymous edits

Unrestricted grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=524886880> *Contributors:* Ahmedmasud, Amalas, Chris Pressey, Cuaxdon, Jonsafari, Nxavar, Obradovic Goran, Pcap, Ruakh, Stormwyrn, Thecheesykid, Woohookitty, 15 anonymous edits

Turing machine *Source:* <http://en.wikipedia.org/w/index.php?oldid=523854074> *Contributors:* -Ril-, Abovechief, Abune, Accelerometer, Ad88110, AdamPeterman, Ahoerstemeier, Ahpook, Alain Vey, Alaminio, Alaniaris, Alejo2083, Alex Vinokur, Aliazimi, Allan McInnes, Altenmann, Altg20April2nd, Andre Engels, Anonymous Dissident, Antandrus, Anton203, Apocalyps956, ArmadniGeneral, Artur adib, Arvinda, Ashutosh y0078, AskI23, Asmeurer, AxelBoldt, B.d.mills, Barak, BearMachine, Ben.croberts, BenRG, Bensin, Bigmantonyd, Blahma, Blaxthos, Blehfu, BorgHunter, Brest, Brion VIBBER, Brouaha, Bryan Derksen, Byrial, CBM, CRGreathouse, Cal 1234, Calibwam, Calliopejen1, Can't sleep, clown will eat me, Carleas, Centrx, Cgay88, Chas zzz brown, Cheran, Cholnes75, Chridd, Chris Pressey, Claygate, Cloversmate, Cocteau834, Cole Kitchen, Conversion script, Creidieki, CryptoDerk, D6, DARTH SIDIOUS 2, DKqwerty, DYLAN LENNON, Dao44, Damian Yerrick, DanielCristofani, Davehwo2, David H Braun (1964), David Koller, DavisSta, Dcoetze, Derek Ross, Dicklyon, Diego Queiroz, Dominus, Doradus, Dratman, DroEsperanto, Drott, Drunkasian, Duncan.Hull, Dzonatas, Ec5618, Edetic, Edward, Ehn, Ellywa, Ender101, ErikTheBikeMan, Ezzett, Ebulides, Eus Kevin, Ewakened, False vacuum, Ferkel, Fleuryeric, Frap, Ftiercel, Fuzheado, Gachet, Gaius Cornelius, Gavia immer, Gdr, Gene Nygaard, GermanX, Giftlite, Gioto, Glacialfox, Glome83, Golbez, GrafZahl, Graham87, GrahamDavies, Greenmatter, Grover cleveland, Gtxfrance, Gubbubu, Gwern, Gwythoff, Hairy Dude, HamburgerRadio, HarisM, Harmil, Head, HenryCorp, Heooo, Heron, HeyStopThat, Howard McCay, Hu12, Hydrogen Iodide, Hzenilc, IMSoP, IShadowed, Iamsckeed, Ieee8023, Ilia Kr., Iridescent, Isaak Rabinovich, J. Spencer, JForget, JMK, JRSpriggs, JSimmonz, JarlHidders, Jaredwf, Jauhienij, Jaye, Jeph paul, Jheiv, Jidan, Jinwick, Johnnuni, Jon Awbrey, Jpbrown, Jpmelos, Jstor, Jurvetson2, K.Nevelsteen, Kadin2048, Kaisershatner, Karl Dickman, Kevin143, Khym Chanur, Kieff, King mike, Kisted, Kku, Klickagente, Kne1p, Kntg, Knutux, Krauss, Kris Schnee, L Kensington, LC, Laminatrix, LarryLACa, Ld100, Liftarn, Lingwitt, LittleDan, Loisel, Lokentaren, LoopZilla, Lotje, Lousyd, MDoggNoGFresh, MSGJ, Machine Elf 1735, Maester mensch, Malleus Fatuorum, Marco.caminati, Martynas Patasius, Materialscientist, MathMartin, Matman132, MattGiua, Merzbow, Metaeducation, Meursault2004, Michael Hardy, Mitch Ames, Miym, Minnex, MoraSique, Mormegil, Mousonner, Mrengy, Mvanveen, Nagato, Nanshu, Napoleon Dynamite42, NapoliRoma, Nikitadamilov, Nitishkorula, Nuno Tavares, Nynexman4464, Obradovic Goran, Oleg Alexandrov, Olivier, Oneiros, Opticon, OrgasGirl, Ott2, P.L.A.R., Pagw, Parhamr, Pascal.Tesson, Patrick, Paul Stansifer, Pcap, Penumbra2000, Pet5, Pexus, Phfyde, Phil Boswell, Philip Trueman, Pleasantville, Plumpy, Polymath69, Populus, Pratimatum, Prolog, Psinu, Punctilius, Q17, QuiteUnusual, R.e.s., RCX, Ramesh Chandra, Raul654, Rbkillea, Readams, Reedy, Reinderien, Rjpryan, Rjwilmsi, Roadrunner, Rob-nick, Robert Merkel, RobertG, RossPatterson, Rp, Ruud Koot, Saforrest, Saty9, Schadel, ScottSteiner, Sheerfirepower, Shell Kinney, Shizhao, Shreevatsa, SimonP, Sligocki, Slike2, Slowking Man, Smimram, Smmurphy, Snoyes, SpNeo, Spikey, Stevertigo, Strangethingsintheland, Steakofhope, Sun Creator, Sundar, Supertouch, Svick, Sviemeister, Syko, TSihlo12, TakuyaMurata, Tarcieri, Tarotcards, TedColes, That Guy, From That Show!, The Anome, Thecheesykid, Themfromspace, Therebelcountry, Theroadislong, Three887, Tide rolls, Tillwe, Timwi, TobiasKlaus, Tom-, TomT0m, Topbanana, Tristamb, Trovatore, UberScienceNerd, Vasyaivanov, Ventolin, Verne Equinox, VictorAnyakin, Vineetgupta, Vrenator, Wavelength, Wednesday Next, WikiTony999, Wikiwikifast, Wolfrock, Wshun, Wuffe, Wvbailey, XJamRastafire, Yan Kuligin, Yipdw, Yourmomblah, Zbxscqf, Zeno Gantner, Александър, 496 anonymous edits

Recursively enumerable language *Source:* <http://en.wikipedia.org/w/index.php?oldid=526538486> *Contributors:* 202.156.2.xxx, Arthur MILCHIOR, AxelBoldt, B6s, CBM, Dcoetze, Giftlite, Gregbard, Hairy Dude, Hans Adler, Jamelan, Jaredwf, Joriki, Kku, LC, Lars Trebing, LodeRunner, MathMartin, Mhss, Michael Hardy, NekoDaemon, Obradovic Goran, Pkirlin, R'n'B, RTC, Salgueiro, StephanWehner, The Anome, Tobias Bergemann, Trovatore, Twri, Tyler McHenry, Wmainer, Zashaw, Zero sharp, 43 anonymous edits

Recursive language *Source:* <http://en.wikipedia.org/w/index.php?oldid=516999760> *Contributors:* Ahoerstemeier, Arthur Frayn, B6s, Big Bob the Finder, Bryan Derksen, CBM, Charles Matthews, Chris Pressey, Conversion script, Dcoetze, Drafflow, Graham87, Gregbard, Hadal, Hairy Dude, Hans Adler, Hermel, Hugo.cs, Jamelan, John of Reading, Jonsafari, LOL, MarkSweep, MathMartin, Max Longint, Michael Hardy, Misof, NekoDaemon, Obradovic Goran, Pcap, Pkirlin, RedDwarf, Saforrest, Salgueiro, Seb, Spayard, Svick, That Guy, From That Show!, Theda, Tobias Bergemann, Trovatore, Tyler McHenry, Vanoco, Zahrradzacken, 31 anonymous edits

Machine that always halts *Source:* <http://en.wikipedia.org/w/index.php?oldid=512344819> *Contributors:* Altemann, Artur adib, CBM, CRGreathouse, Chris Pressey, Cicero, Damian Yerrick, Dcoetze, DeadEyeArrow, Dyukanon, GregorB, Hairy Dude, Isaac Dupree, JRSpriggs, Krauss, L33tmision, Mr Beale, NekoDaemon, Obradovic Goran, Qc, Saforrest, SchreyP, Suruena, Tobias Bergemann, Woohookitty, Yworo, 18 anonymous edits

Context-sensitive grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=524471621> *Contributors:* 1&only, 195.92.67.xxx, AdamDi, Babajobu, Ben Standeven, Bernhard Bauer, BigFatBuddha, Bkkbrad, Bryan Derksen, CRGreathouse, Chris Pressey, Conversion script, Datsun80, Dcoetze, Don4of4, Eubulide, Helios369, Helix84, History2007, Intangir, Jaredwf, Jogloran, Jwalden, Kaihsu, LC, LeonardoGregorian, Mattghg, Matty3269, Mav, Michael Hardy, Nikdo, Obradovic Goran, Qwertyus, RJFJR, Rich Farmbrough, Rp, Sietse, Simon04, Stormwyrn, Svabhishek, The Anome, Themusicgod1, Tobias Bergemann, Tyler McHenry, Woohookitty, 43 anonymous edits

Context-sensitive language *Source:* <http://en.wikipedia.org/w/index.php?oldid=525388684> *Contributors:* Arthur Rubin, AxelBoldt, Ben Standeven, Btwied, Charles Matthews, Conversion script, DanielCohen, Dbachmann, Dratman, Four Dog Night, Intangir, Jonsafari, Michael Hardy, Obradovic Goran, PierreSenellart, Qwertyus, Rich Farmbrough, Ricky81682, Rjwilmsi, Rp, Ruakh, SamuelRiv, The Thing That Should Not Be, Timwi, Tyler McHenry, Vantelimus, 28 anonymous edits

Context-free grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=522135831> *Contributors:* :.Ajvol., 1&only, 2001:660:330F:CC:226:B9FF:FE77:C243, 62.31.64.xxx, A3 nm, Adrian.benko, AlanUS, Alaudo, Algebraan, AnAj, Arialblack, Arnavchaudhary, Arthena, Arvinda, Augur, AxelBoldt, Azndragon1987, BAxelrod, Babajobu, Bedelato, Ben Standeven, Boleslav Bobcik, Brion VIBBER, Brynosaurus, CBKAT Topsails, Cadr, Caesura, Camr86, Chip Zero, ConradPino, Conversion script, Crasshopper, DARTH SIDIOUS 2, Daniel.Cardenas, DavePeixotto, David Eppstein, David Latapie, Dcoetze, Delester, Dominique.devries, Doradas, Dreftymac, Ebraminio, Ehamberg, Erwinrat, Eugenwpg, Flamingospinach, Florian D., FrancisTyers, FreplySpang, Future Perfect at Sunrise, Fuzheado, Gelingvistoj, Giftlite, Givicencio, Gogo Dodo, Grafen, Gran, HenkeB, Hephaestos, Hermel, Iltsgeng, Inimimo, J.H, Jafet, Jamesdady, Jan Hidders, Jaredwf, Jayc, Jni, JoanneB, Joha Fader, JohnOwens, Johnflan, Jokl, Jonsafari, JordanDeLong, Josh Triplett, Klee, Kaihsu, Karl Strootmann, Kenyon, Khazar, Kishamy444, Kmetla, KoenDelaere, Koton, Kronoss, Kwamikagami, Kwertii, LC, Lejatorn, LeonardoGregorian, Likebox, Luquii, Mararo, Marcekes, Marchash, Mastermind3k, Mcld, Merutak, Miaow, Miaow, Minesweeper, Mouhaned99, Mountain, Muu-karhu, N3bulous, Naddy, Nanshu, Nxavar, Obradovic Goran, Oliphant, Polylglut, Paul Foxworthy, Peak, Pete142, Phil Boswell, Phylake, QTJ, Qsebas, Quixpluseone, RaulMiller, Rbrewer42, Ringesherre, Rizome, Rotem Dan, Rp, Ruakh, Ruds, Ruud Koot, Saccade, Sae1962, Saforrest, Saratchandraprasad, Sdornan, Shikhil pict, Sibi antony, Simeon, SixWingedSeraph, Smmurphy, Spencer.mathew, SpikeToronto, Sundar, Susfele, Svick, TakuyaMurata, The Stingray, TheLibrarian, Tim Starling, Timwi, Tjo3ya, TreasuryTag, Tristanreid, Tyler McHenry, UKoch, Universityuser, Urhixidur, VegKilla, Victor Yus, Vsion, Weixifan, William Avery, Wwwwolf, Yoderj, Yoshigev, Zahnradzacken, Zero sharp, ZeroOne, Zeus-chu, Zundark, 302 anonymous edits

Context-free language *Source:* <http://en.wikipedia.org/w/index.php?oldid=526031522> *Contributors:* AgarwalSumeet, Akerbos, Alayth, Alem Dain, AxelBoldt, Ben Standeven, Boris Alexeev, Byel2007, CBM, Ceroklis, ChKa, Charles Matthews, Conversion script, Creidieki, Dcoetze, Euchiasmus, Fffrv, Frungi, Gogo Dodo, Hermel, Iridescent, Jamelan, Jbalint, LungZero, Mark Dingemanse, Miaow Miaow, Misof, Mouhaned99, Muu-karhu, Neeleshmal123, Neilc, Neworder1, Nxavar, Obradovic Goran, PasabaPorAqui, Phil Boswell, Quadrescence, Rp, Schiffli, Stelio, Thumperward, Tobias Bergemann, Tyler McHenry, UKoch, Vantelimus, Zearin, 53 anonymous edits

Pushdown automaton *Source:* <http://en.wikipedia.org/w/index.php?oldid=523179463> *Contributors:* 193.188.1.1xx, Alansohn, Andris, Artem M. Pelenitsyn, Arthur MILCHIOR, Arvindn, Ashutosh y0078, Bjankuloski06en, Bk314159, BlueNovember, Brest, CBKAT Topsails, Cadr, Caesura, Calculuslover, Cerby87, Conversion script, Cyfal, DBSand, Damian Yerrick, Dcoetze, Diego Queiroz, Doradus, Dysprosia, Elektron, Emperormba, FrenchIsAwesome, Germet, Ghettoblaster, Hairy Dude, Henrik, Hermel, Ilya Voyager, Jaredwf, Jochgem, Jonsafari, Josh Cherry, Karl Strootmann, Kku, LKenzo, Lim Wei Quan, Linas, LionKimbro, Livingthingdan, LungZero, Mage327, MarXidad, MathMartin, McGeddon, Mihai Damian, Mike-de-S, Mormegil, Nasra111111, Nxavar, Obradovic Goran, PasabaPorAqui, Phil Boswell, Quadrescence, Rp, Schiffli, Stelio, TimBentley, Urhixidur, Utkwes, Vincent Balat, Xjirak03, 92 anonymous edits

Regular grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=502166579> *Contributors:* Arminahmad, Ashutosh y0078, Babajobu, Ben Standeven, Bkkbrad, ChKa, Conversion script, Cquimper, Damian Yerrick, Dzordzm, Estemle, Ilia Kr., Jaredwf, Kaihsu, LeonardoGregorian, Linas, MadLex, Matt4077, Morton Shumway, Philip Trueman, Qsebas, Rp, Sebesta, Simeon, Slaniel, Suffusion of Yellow, TWiStErRob, The Thing That Should Not Be, Tyler McHenry, Vopros, 35 anonymous edits

Regular language *Source:* <http://en.wikipedia.org/w/index.php?oldid=525211190> *Contributors:* 142.166.108.xxx, Adam McMaster, Ahoerstemeier, Akerbos, Alex Dainiak, Arjayay, Ashutosh y0078, AxelBoldt, Beefman, BenFrantzDale, Bender2k14, Bryan Derksen, ChKa, ChrisGualtieri, ClaesWallin, Cokaban, ConradPino, Conversion script, DFRussia, DanielKO, David Eppstein, Dcoetze, Delirium, Deltahedron, DemonThing, Diegodim, DopefishJustin, Dratman, EngineerScotty, Eric119, Ericboden, Erudecorp, Flyhighplato, Fropuff, Fudo, Gachet, GroveGuy, Haham hanuka, Hannes Hirzel, Harp, Helder.wiki, Hermel, Isometric, Jaredwf, Jellystones, Jerryseb, Jochgem, Jsnx, Karl Dickman, King of Hearts, LC, Linas, Lord Roem, Luqui, Lzur, Mani1, Michael Hardy, Muu-karhu, Mzharison, NTF, Nitishkorula, Olivier, Pako, Pichpich, Proleph, R'n'B, RPHV, Rich Farmbrough, Ripper234, Rjwilmsi, Runttime, Schneloelcke, Shouvik, Simeon, Stevertigo, Tannin, That Guy, From That Show!, Tjifo098, Timwi, Tlev04, Tobias Bergemann, Twri, Tyler McHenry, UKoch, WayneMokane, Zahnradzacken, 103 anonymous edits

Regular expression *Source:* <http://en.wikipedia.org/w/index.php?oldid=525511640> *Contributors:* 142.177.80.xxx, 2602:304:AEB9:5AB9:B8C9:FB1A:4489:3BD, 336, @modi, Aaron Schulz, Adxd, Adarw, Ahy1, AlanUS, Alansohn, Alexander256, Alexibu, Aligma, Altales Teriadem, Altenmann, Amire80, AnK, Anabus, Andre Engels, Andreas Kaufmann, AndreasSchreiber, Angrydwarf, Anilashanbhag, AnnaFinotera, AnnaFrance, Antimatter15, Apokrif, Appz, Arbor, Ardonik, Arthur Smart, Arvindn, Ashutosh y0078, AstroNox, Asuffield, Athenae, Audacity, AxelBoldt, Baa, Balabiot, Bananabruno, Barinjato, Barrien, Bdsham, Bdq, Ben Standeven, BenForta, Benrick, Betterusername, Bevo, Bigdottawa, BillFlis, Billposer, Bisqwit, Blenda, BluePlateSpecial, Bluegrass, Bluemoose, Booyabazooka, BradBeattie, Brendta, Brest, Brian Geppert, Bryan Derksen, Btx40, BudVeezer, Bwoodacre, C.Fred, CALR, CBM, CRGreathouse, Cacophony, Calvinscorner, Captmj, Carl.antuar, Catamorphism, Celada, Cgtdk, Chapman, Charbelgerge, Charles Matthews, Chenzw, Chetan.akarte, Chowbok, ChrisHodgesUK, ClaretAsh, Cntras, Coffee2theorems, Colonies Chris, Cometstyles, Conversion script, Corti, Cowplomporris, Cpacifico, Cremepuff222, CyberSkull, Cygnus78, Cymru.lass, Daverocks, David Eppstein, Dbenbenn, Dcoetze, Demerphq, DerHexer, Derek farm, Diavel, Diderot, Dinosaurdarrell, Dionyziz, Diz, Djmitche, Dmccreary, DocWatson42, Dogsgomoo, Dominixis, Donarreiskoffer, Donga

alpesh, Dongre.avinash, Downtown dan seattle, Dreftymac, Drrngrvy, EagleOne, Edggar, Editor117, Editor2, Edward, Edwinstearns, Eeve, Eharney, El C, Eleassar, Eliashedberg, Eloquence, EncMstr, Erel Segal, ErikStewart, Errendir, Euphrosyne, FERsI, FayssalF, FeyFre, Fibonacci, FinsJ, Lewis, Fredrik, FrenchlsAwesome, Frosty34, Fsiler, Fullstop, Furykef, Fyyer, Gaiacarra, Gbdarren, GeeksHaveFeelings, Gerbrant, Ghettoblaster, Ghiaurul, Gibber blot, Gifflite, Gioto, Glenn, Gogo Dodo, Gojomo, Gokusandwich, Goodmedalist, GordonMcKinney, Gpvos, Gracenotes, Grawity, Gregben, Grim Revenant, Gunarsekar, H2g2bob, Hajatvrc, Hardeeps, Harmil, Haruth, Hazelorb, Helder.wiki, HelgeHan, Hermel, Hervegirod, Hirzel, Hurmata, Husky, Hyperyl, I already forgot, Isceaturlles, Igna, Immunize, Intelliproject, Interiot, Isaac, IsotopeOrange, Ivan Pozdeev, Ixfd64, J Di, JLaTondre, JY Ouyang, Jacj, Jacob Finn, JadeNB, Jalwikip, James Foster, Jan Hidders, Jarble, Jaxl, JennyRad, Jesin, Jezmck, Jgeer, Jiri 1984, Jk2q3jrkse, Jleedev, Jolgoran, Johnhbibby, Johnh, Johnmarkh, Johnuniq, Jon Awbrey, Jonik, Jphofmann, Jpk, Jrtayloriv, JuanpdP, klee, KNHaw, Kameraad Pjotr, Karl Dickman, Kellen', Khendon, Kilo-Lima, Kingturtle, Knotaholic, Korg, Krauss, Krymnzon, Kwamikagami, L Kensington, LLarson, LOL, La goutte de pluie, Learsdata, LeadSongDog, Leotohill, Libroescondido, Lidden, Ligulem, Lobner, Logan, Lost.goblin, LoveFest, Lowellian, Lucyin, MC10, MER-C, MPerel, Mac, MacGyver07, Macrakis, Magore, Mahanga, Malbrain, Maniac18, ManuelGR, Mark R Johnson, MarkJoseph sc, Marudubshinki, Materialscientist, MathMartin, Matt bucknall, Mbumber, Mdialogo, Mecanismo, MeekMark, Memiux, Mfv2, Michael Hardy, Michael Shields, Mike Rosol, Miles, Mindtrix, Minghong, Minou30, Miskaton, Mjarr, Mondebleu, Monger, Mormegil, Mortense, MsZiger, Muammer313, Mxn, Mysterytrey, Mysticprg, Mythago, Mythpage88, Mzyxplk, NTF, Nakon, Nanshu, Nb93, Ncmathsadist, Nealmcb, Neier, NeilN, Neilc, Nicieguyedc, Nikola Smolenski, Nmagedman, Noldoaran, Nooxoo, Octahedron80, Officiallyover, OIEnglish, Olathe, Omegatron, Omphaloscope, Oniscoid, Onlynone, Optikos, PGSONIC, PSheratt, Palapala, Pankaj, dadure, Paranoia, Pauljerry, Peng, Pennmachine, Penno, Percy Snoodle, Peterl, Pexusat, Phansen, Pinethicket, Pistol651, Pne, Pol098, Popracer, Porges, Psd, Ptmc2112, Pyrog33k, QTJ, Raistlin1325, Ralmin, RandalSchwartz, RandomP, RevRagnarok, Rfschmid, Rich Farmbrough, Rikkus, Ringbang, Rjanag, Robinjam, Rodion Gork, Roehsler, Romanim, Roro, Royote, Rrjanbiah, Rsteif, Rudd-O, Rudinsky, Ruud Koot, SRaffman123, ST47, Sadib en, Saforrest, Sairahulreddy, SaltyPig, SamB, SarekOfVulcan, Savinovboris, Sboses7990, Scandum, Schneelocke, Scientius, SeanDuggan, Sekelsenmat, Semnard, SergeantPepper, Shadowjams, Shamatt, Shipssales, Shrish, Sigurdhsson, SikOfewl, SimBeSim, SimonP, Slawekb, Sleske, Slicing, Smyth, Snowolf, Softsmith, Spearhead, Spedudmak, Sping, Splintax, SqueakBox, Stephan Leeds, Steven Hepting, SudoGhost, SueFollowes, Sun Creator, Supersteve1440, Svick, Szaboles Nagy, Takanoha, Tarquin, Tasha777, Tchalvakspm, Tedickey, Teemli Leisti, Tekinico, Terrrminator, Tetracube, That Guy, From That Show!, The Anome, The Transhumanist, The-stickman, TheOnlyMike, Thomas Larsen, Thore, Thruston, Timwi, TitanX, Txnman307, Tobias Bergemann, Tpbtradbury, Turmoi, TutterMouse, Tuukkah, Twarther, USConsLib, UTF-8, Ueucha, Ultimad, Underdog, UnicornPresty, Unitman00, Urhixidur, Vanoogle, VictorAnyakin, Vijayviji, Vikreykja, ViperSnake151, Voidor, Volkant, W.andreas, WODUP, Webaware, Widefox, Wikid77, Winston365, Winterst, Witharebelyell, Xongnopp, Xp54321, Xynariz, Yogendra.joshi, ZZyXx, Zahical, Zahnradzacken, Zap Rowsdower, Zarboublian, Zenohockey, Zhen-Xjell, Zondor, Ævar Arnfjörð Bjarmason, 878 anonymous edits

Generative grammar Source: <http://en.wikipedia.org/w/index.php?oldid=526044685> Contributors: ACSE, Action potential, Adam78, Adoniscik, Aeusoies1, AkselGerner, Alammaher, Aleksd, Alinguist, Alipir, AllanBz, Anomalocaris, Babajobu, Beland, Blacksamourai, Bratschespieler, Byelf2007, Cadr, Caesura, Charles Matthews, Colonies Chris, Combreir, Cuaxdon, Dale Chock, Damian Yerrick, Dbachmann, Dolfrg, Drpixie, Eequor, Elmer Clark, Eransran, Erebus555, Euthynon, Fbkintanar, Filiocht, FrancisTyers, G.broadwell, Gwestphal, Gracewhizz, Greg Hullender, Grunge6910, Hstel, Hans castorp81, Hornlitz, Hstsr8, Hyacinth, Ioscius, Jbergquist, Jhessela, Johnchacks, Jonsafari, Joseph Solis in Australia, Kku, Klauys, Kobokai, Kwerti, Le vin blanc, Linas, Linguizic, MacGyverMagic, Mandarax, Mardus, MarkSweep, Mild Bill Hiccup, Miru51, Mundart, QzDaddy, RCSB, Rainwarrior, RedWolf, RekishiEJ, Rhobite, Rjanag, Rjwilmsi, Sean.hoyland, Seglea, Semmelweiss, Sverdrup, Syndicate, Technopat, Teeteto, Theshibboleth, Thüringer, TobyJ, TopAce, Trigaranus, Trondtr, Tulpan, Usernodunno, Woohookitty, ZooFari, 83 anonymous edits

Transformational grammar Source: <http://en.wikipedia.org/w/index.php?oldid=522772881> Contributors: Action potential, Aidan Elliott-McCrea, Alipir, Altenmann, Angr, Ansa211, Ap, Babajobu, Beland, Bit, Bovlb, Byelf2007, Cadr, ChrisGaultieri, Chitto, Combreir, Cpaleos, Cuaxdon, Damian Yerrick, DanKeshet, Dbachmann, Dduck, Dissident, Dodger, Donald Albury, Dzzl, Ealdent, Enchanter, FrancisTyers, Frankieroberto, Furykef, Gary King, Gjako^it, Gonzalo Diethelm, Greyskinnedboy, Gwil, Iridium77, Irpond, Jim1138, Jokestress, Jonsafari, JorisvS, Kku, Kunal Sharma, LaggedOnUser, Lightmouse, Llywrch, Lynneguistics, Mani1, MarkSweep, Mattcolet, Michael Hardy, Miguelmrm, Monikers, Mundart, Myshkin, Nora lives, Pax:Vobiscum, Peter Isotalo, Pjrm, Rama, Redmblu, Rehoot, RekishiEJ, Rjanag, Roehl Sybing, Ruakh, Russky1802, Ryguasu, Sburke, Seth Ilys, Shizhao, Stevertigo, Stpidhead, Sundar, Superfascist, Sverdrup, The Wiki ghost, Tjo3ya, Trevor MacInnis, Tristatestar, Uanfala, Unyoyega, Viriditas, Vox populi 2008, Wayward, William M. Connolley, Woohookitty, Zack wadghiri, 108 anonymous edits

Principles and parameters Source: <http://en.wikipedia.org/w/index.php?oldid=517987961> Contributors: Andrea moro, Asfarer, BDFD2, Beland, Belfry, BovineBeast, Briangellis, Byelf2007, ChrisCork, ChrisGaultieri, DMacks, Damian Yerrick, Darkllae, Dbachmann, Dewrad, Endangeredlinguist, Er Komandante, Francis Tyers, FrancisTyers, Grumpygiraffe, Hejdoa, Ioscius, Iridescent, Ish ishwar, Jimw338, John, Jon Awbrey, Matve, Mcswell, Michael Hardy, Omnipaedista, R'h'B, Rl, Space Dracula, Stevertigo, Synchronism, Tjo3ya, Tyrell turing, 45 anonymous edits

Government and binding theory Source: <http://en.wikipedia.org/w/index.php?oldid=471034465> Contributors: Altenmann, Angr, Beland, Burschik, Cadr, Crossmr, Dduck, Eequor, Fnorp, Furykef, Gregbard, JeffyP, Jonsafari, Kahn, Lambiam, Marcus22, Michael Hardy, Mitchoyoshitaka, NidFlocken, Pearle, Peter Isotalo, Rjwilmsi, Secretlondon, Seth Ilys, Shevaun384, Sillyfolkboy, Stevertigo, Tijfo098, Tjo3ya, Torzsmokus, Watasenia, Zaheen, 20 anonymous edits

Minimalist program Source: <http://en.wikipedia.org/w/index.php?oldid=515246351> Contributors: Alexhsu, Alipir, Anomalocaris, Asfarer, Byelf2007, Cadr, Catawampus122, Chris the speller, Combreir, David van bruwae, Dbachmann, DocteurCosmos, Dougher, Download, InverseHypercube, Kepwick, Michael A. White, Miguelmrm, Rjwilmsi, RobertBaruch, Stegop, Tjo3ya, TwigsCogito, Uanfala, Zundark, 22 anonymous edits

Relational grammar Source: <http://en.wikipedia.org/w/index.php?oldid=491442791> Contributors: AnonMoos, CXCV, Dbachmann, Edward, Gengogakusha, Ish ishwar, Jonsafari, LeoNomis, Masonhickman, Mundart, Nihongogakusha, The Wiki ghost, 3 anonymous edits

Lexical functional grammar Source: <http://en.wikipedia.org/w/index.php?oldid=509170229> Contributors: Anomalocaris, Beland, Benoît Sagot, Burschik, Byelf2007, Cadr, CapnPrep, Charles Matthews, Cuaxdon, Dbachmann, G.broadwell, Hellth, Hirzel, Innoak, Jacobko, Jamelan, Jonsafari, KEJ, Kiwibird, MT301, Mary.dalrymple, Mike Dillon, Neutral-en, Nihongogakusha, Paul D. Anderson, Peter Isotalo, Smartaalec, The Anome, The Wiki ghost, Tjo3ya, UKoch, Uanfala, Woohookitty, Yehuda Falk, 20 anonymous edits

Generalized phrase structure grammar Source: <http://en.wikipedia.org/w/index.php?oldid=494190208> Contributors: Beland, Burschik, Cadr, CapnPrep, FrancisTyers, Jamelan, Kku, MarkSweep, Netesq, P Ingerson, Sillyfolkboy, Stevenmitchell, Szyslak, The Anome, Tjo3ya, UKoch, 6 anonymous edits

Head-driven phrase structure grammar Source: <http://en.wikipedia.org/w/index.php?oldid=518463202> Contributors: Alvations, BAxelrod, Babajobu, Beland, Belladon, Burschik, Cadr, CapnPrep, Chriki, Chris the speller, Colonies Chris, Dduck, Dront, Ffbond, HeartofaDog, Hirzel, Intelligentium, Jafet, Jamelan, James Crippen, Jobber, Johnor, Jonsafari, Kahn, Kallerdis, Kku, Lizmarie, Mariuslj, Netesq, Nieske, Nihongogakusha, Siegel, Stemonitis, The Anome, Thüringer, Tjo3ya, TonyW, UKoch, 38 anonymous edits

Categorial grammar Source: <http://en.wikipedia.org/w/index.php?oldid=525085999> Contributors: ABCD, Angela, Babajobu, Bloodshedder, Bobblehead, CBM, Chrisblom86, Cryptic, Fratrep, FreeChief, Gökhan, Jnothman, Jonsafari, Leibniz, Luna Santin, MarkSweep, Matve, Mets501, Omnipaedista, Pcap, Q10, Serapio, Synsem, The.cunning.ham, Tijfo098, Tjo3ya, UKoch, Votlook, Will Beback Auto, Woohookitty, Zaheen, 22 anonymous edits

Tree-adjoining grammar Source: <http://en.wikipedia.org/w/index.php?oldid=493068580> Contributors: Augur, Beland, Ben Standeven, Bkkbrad, Bn, Brutaldeluxe, Burschik, Cadr, Charles Matthews, David Chiang, Djame, Dsdedah, Ghettoblaster, Gurch, Jamelan, Jolgoran, Jonsafari, Michael Hardy, MikeGasser, Mutt Lunker, Orballo, Phil Boswell, Piedmontchris, SamuelRiv, Stephen Shaw, UKoch, 14 anonymous edits

Nanosyntax Source: <http://en.wikipedia.org/w/index.php?oldid=490886354> Contributors: 28bytes, AvicAWB, Fabrictramp, Gene Nygaard, Hirsutism, Infrangible, Jasonbook99, M656, Markiewp, Nickyus, Sadads, 10 anonymous edits

Arc pair grammar Source: <http://en.wikipedia.org/w/index.php?oldid=491297096> Contributors: Bhadani, Gengogakusha, Ish ishwar, Nihongogakusha, Omnipaedista, Twp, Yakushima, 1 anonymous edits

Generative semantics Source: <http://en.wikipedia.org/w/index.php?oldid=512723292> Contributors: Anomalocaris, AnonMoos, Byelf2007, Cadr, Denisp, Ffbond, Ian Pitchford, InverseHypercube, Ish ishwar, Jonsafari, Karada, Kevin.cohen, Ludling, Omphaloscope, PaperCutlet, Pranesh Bhargava, Rdsmith4, Stefano85, The Wiki ghost, Yug, 21 anonymous edits

Dependency grammar Source: <http://en.wikipedia.org/w/index.php?oldid=525450634> Contributors: Arabismo, Attardi, Byelf2007, Chenli, Ddxc, Dnmaxwell, Informatician, JamesAM, Jason Quinn, Jonsafari, Linas, Linguistlist, Metavivo, Pcap, Peak, Pitam, Qsdqsf, RichardHudson, The Wiki ghost, Tjo3ya, Tony1, Trickstar, U-571, UKoch, Uncle G, Waltpohl, Whym, Zuky79, 47 anonymous edits

Recursive categorical syntax Source: <http://en.wikipedia.org/w/index.php?oldid=431697233> Contributors: Augur, Beland, BradBeattie, Burschik, Chzz, Gene Nygaard, Guy Macon, Haza-w, IPSOS, Jerzy, Linas, MCiura, Michael Slone, Pearle, Rejnal, Rich Farmbrough, Scoty6776, Semmelweiss, Unlambda, 1 anonymous edits

Operator grammar Source: <http://en.wikipedia.org/w/index.php?oldid=494751085> Contributors: Informatician, Jonsafari, Koavf, Kogorman, R'n'B, Tony1, 9 anonymous edits

Functional generative description Source: <http://en.wikipedia.org/w/index.php?oldid=491585974> Contributors: Bearcat, Kwamikagami, Kyoakoa, Malcolma, Sandius, Tony1

Meaning–text theory *Source:* <http://en.wikipedia.org/w/index.php?oldid=500899594> *Contributors:* 4th-otaku, Babbage, Black Falcon, D6, Davidjamesbeck, Galilite, Giraffedata, GregorB, Kyoakoa, Malcolma, Menelik3, Nick Number, Qsdfqsdf, SacredCheese, Soshial, Tony1, Tuetschek, 39 anonymous edits

Word grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=519230583> *Contributors:* 0, Beland, Burschik, Cadr, Closeapple, Css, Fredrik, Hannes Hirzel, Jmendez, Mk380, Mrwojo, Nfwu, Poccil, RichardHudson, The Wiki ghost, Yakushima, 6 anonymous edits

Link grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=495511092> *Contributors:* AxelBoldt, Babajobu, Beland, Cadr, Cybercobra, Ddxc, Delirium, Derek Ross, Doradus, Dranorter, Eequor, Fnielsen, Garronb, InverseHypercube, Jafeluv, Jogloran, Jonsafari, Kwamikagami, Liao, Linas, Madanpiyush, Marudubshinki, Onyee, PTSE, Richard W.M. Jones, Srd2005, Yurivict, Yworo, 18 anonymous edits

Quranic Arabic Corpus *Source:* <http://en.wikipedia.org/w/index.php?oldid=524849658> *Contributors:* Arabismo, Fi11222, Frap, Imdkzmaa, Jaksmata, Koavf, Matt57, Oneeyedboxer, SBaker43, Salix alba, 9 anonymous edits

Functional discourse grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=526556554> *Contributors:* Anypodetos, Fayenatic london, Iridescent, J. Spencer, Malcolma, Neutrality, Nieske, Obankston, RHaworth, Tux rocker, 4 anonymous edits

Prague school *Source:* <http://en.wikipedia.org/w/index.php?oldid=526426261> *Contributors:* Aleksd, Alipir, Babbage, Black Falcon, Ceancata, Charles Matthews, Cmaric, Crisco 1492, Dale Chock, DavidWBrooks, Dr Oldekop, Ducktaper, Everyking, FSII, Ganymead, GarrisonLeMasters, Hmains, Jahsonic, Jlittlet, Jm34harvey, Kwamikagami, Nlight, Omnipaedista, Radimsky, Rbellin, Reo On, Rjwilmsi, Sandius, Sebesta, ToNToNi, Tolkowski, Tony1, Travelbird, WikiHannibal, ΛεΞικόπουλος, 24 anonymous edits

Systemic functional grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=522464276> *Contributors:* 1exec1, AjaxSmack, AlistairMcMillan, Alukin, Annabelle Lukin, Bekind, Beland, Burschik, Bwebste1, C1614, Cnilep, Cometstyles, Conversion script, Damian Yerrick, Daniel Vortisto, Deafnews, E2lise, ErikHaugen, Exorabilis, Gary King, GoingBatty, Graham87, Grumpygiraffe, Hamfish4u, Hannes Hirzel, Hariva, Horatio, Javier Carro, JoannaSerah, Jonh bateman, KYPark, KnightRider, Mandarax, Matiasobera, Maunus, Mayumashu, Michael Hardy, Michje, Micko.madrid, Mistex2000, Mr. Absurd, Narssarssuaq, NickelShoe, Olivier, Prof Tournesol, R Lowry, Rintrah, Rjwilmsi, Seanuy, Serge925, Snookerfran, Stevertigo, Taragui, The Wiki ghost, Thomas Bull, Thüringer, Tony1, Trickstar, Unyoyega, WhisperToMe, Zenohockey, 45 anonymous edits

Cognitive grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=473789201> *Contributors:* Action potential, Cewvero, Chick Bowen, Dawynn, Fadesga, Gregbard, Javier Carro, KEJ, Lavintzin, Mark Dingemanse, Mike Dillon, Mukerjee, Mysidia, Plonk2, RobyWayne, Trondtr, Zoicon5, 5 anonymous edits

Construction grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=513844789> *Contributors:* Andres, Anomalocaris, ArglebargleIV, Babajobu, Bekind, Bissinger, Cadr, Cassowary, Chenkeng, Chick Bowen, Cholling, DCDuring, DTOx, Dbachmann, Drilnoth, El Raki, Friedrich Tellberg, Garik, Grafen, Igiffin, Ish ishwar, John Quiggin, Jpop, Junkyardprince, KEJ, Kerotan, Kyoakoa, Mark Dingemanse, Michael Hardy, Mike Dillon, Paxse, PhiRho, Pitam, RHaworth, Reedy, Rjwilmsi, Sandman2007, SchreiberBike, Serapio, Svenonius, Tedermst, The-Pope, Topbanana, Woohookitty, Ziggurat, 91 anonymous edits

Role and reference grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=476942578> *Contributors:* Beland, Burschik, DeadEyeArrow, Elektrik blue 82, Hirzel, Ish ishwar, Javier Carro, MarcusCole12, Nihongogakusha, Pflastertreter, Rangasyd, SimonP, Szyslak, 6 anonymous edits

Emergent grammar *Source:* <http://en.wikipedia.org/w/index.php?oldid=476904559> *Contributors:* Grenadine, Insanityinc, Kyoakoa, Michael Hardy, Rjanag, Rjwilmsi, 1 anonymous edits

Image Sources, Licenses and Contributors

File:Ancient Tamil Script.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Ancient_Tamil_Script.jpg *License:* Creative Commons Attribution 2.0 *Contributors:* Symphonny Symphonny from New York, US

Image:Tepantitla mural, Ballplayer A (Daquella manera).jpg *Source:* [http://en.wikipedia.org/w/index.php?title=File:Tepantitla_mural,_Ballplayer_A_\(Daquella_manera\).jpg](http://en.wikipedia.org/w/index.php?title=File:Tepantitla_mural,_Ballplayer_A_(Daquella_manera).jpg) *License:* Creative Commons Attribution 2.0 *Contributors:* Dodo, FlickrviewR, Madman2001

Image:Cuneiform script2.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Cuneiform_script2.png *License:* Public Domain *Contributors:* Cuneiform_script2.jpg: derivative work: Yjenith (talk)

Image:Girls learning sign language.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Girls_learning_sign_language.jpg *License:* Creative Commons Attribution 2.0 *Contributors:* David Fulmer from Pittsburgh

Image:Braille house09.JPG *Source:* http://en.wikipedia.org/w/index.php?title=File:Braille_house09.JPG *License:* Creative Commons Attribution-Sharealike 3.0,2.5,2.0,1.0 *Contributors:* Kou07kou

File:ASL family.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:ASL_family.jpg *License:* Creative Commons Attribution 2.0 *Contributors:* David Fulmer from Pittsburgh

Image:BBC-artefacts.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:BBC-artefacts.jpg> *License:* GNU Free Documentation License *Contributors:* Original uploader was Chenshiwood at en.wikipedia

Image:Pieter Bruegel the Elder - The Tower of Babel (Vienna) - Google Art Project - edited.jpg *Source:* [http://en.wikipedia.org/w/index.php?title=File:Pieter_Bruegel_the_Elder_-_The_Tower_of_Babel_\(Vienna\)_-_Google_Art_Project_-_edited.jpg](http://en.wikipedia.org/w/index.php?title=File:Pieter_Bruegel_the_Elder_-_The_Tower_of_Babel_(Vienna)_-_Google_Art_Project_-_edited.jpg) *License:* unknown *Contributors:* Dcoetzee, Foundling

Image:Sir William Jones.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Sir_William_Jones.jpg *License:* Public Domain *Contributors:* BishkekRocks, Cirt, Ecummenic, Hsarrasin, Jonsafari, Lokal Profil, Mattes, Mu, Shakko, Thomas Gun, VladiMens

Image:Ferdinand de Saussure by Jullien.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Ferdinand_de_Saussure_by_Jullien.png *License:* Public Domain *Contributors:* "F. Jullien Genève", maybe Frank-Henri Jullien (1882–1938)

Image>Noam chomsky cropped.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File>Noam_chomsky_cropped.jpg *License:* Copyrighted free use *Contributors:* Created by John Soares, uploaded to Commons by Stevertigo, then modified by Verdy p.

Image:Brain Surface Gyri.SVG *Source:* http://en.wikipedia.org/w/index.php?title=File:Brain_Surface_Gyri.svg *License:* Creative Commons Attribution-Share Alike *Contributors:* James.mcd.nz

Image:Illu01 head neck.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Illu01_head_neck.jpg *License:* Public Domain *Contributors:* Arcadian

Image:Spectrogram -iua-.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Spectrogram_-iua-.png *License:* Creative Commons Attribution 2.0 *Contributors:* Ishwar, Mike.lifeguard, Moyogo, 2 anonymous edits

Image:Real-time MRI - Speaking (Chinese).ogv *Source:* [http://en.wikipedia.org/w/index.php?title=File:Real-time_MRI_-_Speaking_\(Chinese\).ogv](http://en.wikipedia.org/w/index.php?title=File:Real-time_MRI_-_Speaking_(Chinese).ogv) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Martin.uecker

Image:Ancient Tamil Script.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Ancient_Tamil_Script.jpg *License:* Creative Commons Attribution 2.0 *Contributors:* Symphonny Symphonny from New York, US

Image:ManSpec.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:ManSpec.png> *License:* Creative Commons Zero *Contributors:* Grechukk

Image:Hangul wi.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Hangul_wi.svg *License:* unknown *Contributors:* Kjoonlee, Kwj2772, Mechamind90, Sarang, Waldir

Image:KSL wi.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:KSL_wi.jpg *License:* GNU Free Documentation License *Contributors:* Original uploader was Kwamikagami at en.wikipedia

File:Basic constituent structure analysis English sentence.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Basic_constituent_structure_analysis_English_sentence.svg *License:* Public Domain *Contributors:* Adam majewski, AnonMoos

Image:Speech Client WEB (5).jpg *Source:* [http://en.wikipedia.org/w/index.php?title=File:Speech_Client_WEB_\(5\).jpg](http://en.wikipedia.org/w/index.php?title=File:Speech_Client_WEB_(5).jpg) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Madmike1952

Image:Lakhovsky Conversation.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Lakhovsky_Conversation.jpg *License:* Public Domain *Contributors:* AndreasPraefcke, Bukk, Dia^, Mattes, Ticketautomat, 1 anonymous edits

Image:Winnipeg Forks - Plains Cree Incription.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Winnipeg_Forks_-_Plains_Cree_Incription.jpg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Ardfern, Diderot, G.dallorto, Man yi, Skeezix1000, Tar-ba-gan, 2 anonymous edits

image:Beowulf.firstpage.jpeg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Beowulf.firstpage.jpeg> *License:* Public Domain *Contributors:* Caravaca, Davepape, EugeneZelenko, Evrik, Jastrow, Jonathunder, Leos ván, Neddyseagoon, Ranveig, Semnoz, Verica Atrebatum, 3 anonymous edits

File:Human Language Families Map.PNG *Source:* http://en.wikipedia.org/w/index.php?title=File:Human_Language_Families_Map.PNG *License:* GNU Free Documentation License *Contributors:* Industrius, Kristaga, Mtll, Neutrality, Phoenix B 1of3, Stern, Waggers, 2 anonymous edits

File:Linguistic diversity.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Linguistic_diversity.png *License:* Public Domain *Contributors:* Davius

Image:Sociolinguistics dialect variation.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Sociolinguistics_dialect_variation.svg *License:* GNU Free Documentation License *Contributors:* FrancisTyers, Pfcfdayelise

File:Clause trees 1.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Clause_trees_1.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Clause trees 2.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Clause_trees_2.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Clause trees 3'.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Clause_trees_3'.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:The house at the end of the street.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:The_house_at_the_end_of_the_street.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Phrases 2.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Phrases_2.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

Image:Nikolai Trubetzkoy.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Nikolai_Trubetzkoy.jpg *License:* Public Domain *Contributors:* Frank C. Müller, Glossologist, Sergei, UV, Лист рівнодення

Image:Phonological Diagram of modern Arabic and Hebrew vowels.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Phonological_Diagram_of_modern_Arabic_and_Hebrew_vowels.png *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Javier Carro, Mo-Al, Moxyfire, Paradocitor, Shlomital

Image:Phonetic Diagram of modern Arabic and Hebrew vowels.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Phonetic_Diagram_of_modern_Arabic_and_Hebrew_vowels.png *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Conscious, Moxyfire, Paradocitor, Shlomital, WikipedaMaster

File:Roma jakobson theory.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Roma_jakobson_theory.png *License:* Public Domain *Contributors:* Artist2426

File:Formal languages.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Formal_languages.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:MithrandirMage

Image:Turnstile state machine colored.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Turnstile_state_machine_colored.svg *License:* Creative Commons Zero *Contributors:* User:Chetvorno

Image:Torniqueterevolution.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Torniqueterevolution.jpg> *License:* Creative Commons Attribution-Share Alike *Contributors:* Sebasgui

File:UML state machine Fig5.png *Source:* http://en.wikipedia.org/w/index.php?title=File:UML_state_machine_Fig5.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Miroslavek (talk)

Image:SdlStateMachine.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:SdlStateMachine.png> *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Manu31415, Theon144

File:Finite state machine example with comments.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Finite_state_machine_example_with_comments.svg *License:* Public Domain *Contributors:* Macguy314

File:Fsm parsing word nice.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Fsm_parsing_word_nice.svg *License:* Public Domain *Contributors:* en:User:Thowa, redrawn by User:Stannered

File:DFAexample.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:DFAexample.svg> *License:* Public Domain *Contributors:* Cepheus

File:Fsm mealy model door control.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Fsm_mealy_model_door_control.jpg *License:* Public Domain *Contributors:* Original uploader was Thowa at en.wikipedia

File:Finite State Machine Logic.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Finite_State_Machine_Logic.svg *License:* Public Domain *Contributors:* jjbeard

File:4 bit counter.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:4_bit_counter.svg *License:* Public Domain *Contributors:* Gargan

File:Parse tree 1.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Parse_tree_1.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Parse2.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Parse2.jpg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

Image:Leftmostderivations jaredwf.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Leftmostderivations_jaredwf.png *License:* Public Domain *Contributors:* Original uploader was Jaredwf at en.wikipedia

File:This tree is illustrating the relation (PSG).png *Source:* [http://en.wikipedia.org/w/index.php?title=File:This tree is illustrating the relation \(PSG\).png](http://en.wikipedia.org/w/index.php?title=File:This tree is illustrating the relation (PSG).png) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

Image:Chomsky-hierarchy.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Chomsky-hierarchy.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* J. Finkelstein

Image:Maquina.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:Maquina.png> *License:* Public Domain *Contributors:* Schadel (<http://turing.izt.uam.mx>)

Image:Turing machine 2a.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Turing_machine_2a.svg *License:* GNU Free Documentation License *Contributors:* User:Nynexman4464

Image:Turing machine 2b.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Turing_machine_2b.svg *License:* Public Domain *Contributors:* User:Nynexman4464. Original uploader was Nynexman4464 at en.wikipedia

Image:State diagram 3 state busy beaver 2B.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:State_diagram_3_state_busy_beaver_2B.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Diego Queiroz

Image:State diagram 3 state busy beaver 4.JPG *Source:* http://en.wikipedia.org/w/index.php?title=File:State_diagram_3_state_busy_beaver_4_.JPG *License:* GNU Free Documentation License *Contributors:* User:Wvbailey

File:Lego Turing Machine.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Lego_Turing_Machine.jpg *License:* Creative Commons Attribution 3.0 *Contributors:* TomT0m

Image:Pushdown-overview.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Pushdown-overview.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochgem

Image:Pushdown-step.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Pushdown-step.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochgem

Image:Pda-example.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Pda-example.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochgem

Image:Pda-steps.svg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Pda-steps.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochgem

File:Basic english syntax tree.svg *Source:* http://en.wikipedia.org/w/index.php?title=File:Basic_english_syntax_tree.svg *License:* Public Domain *Contributors:* Raster: Cadr at English Wikipedia SVG: Beao

Image:ApBp.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:ApBp.png> *License:* Public domain *Contributors:* Watasenia at en.wikipedia

Image:HeSmashedTheVase1.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:HeSmashedTheVase1.png> *License:* Public Domain *Contributors:* Watasenia (talk) (Uploads)

Image:Government and Binding Theory basic tree.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Government_and_Binding_Theory_basic_tree.png *License:* Public Domain *Contributors:* Cadr

Image:the mother of John.png *Source:* http://en.wikipedia.org/w/index.php?title=File:The_mother_of_John.png *License:* Public Domain *Contributors:* Shevaun384

Image:Walks-avm.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:Walks-avm.png> *License:* unknown *Contributors:* Lizmarie

Image:She-avm.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:She-avm.png> *License:* unknown *Contributors:* Lizmarie

Image:Head-subj-avm.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:Head-subj-avm.png> *License:* Public Domain *Contributors:* Lizmarie

Image:Head-subj-tree.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:Head-subj-tree.png> *License:* unknown *Contributors:* Lizmarie

File:We are trying to understand the difference (2).jpg *Source:* [http://en.wikipedia.org/w/index.php?title=File:We are trying to understand the difference_\(2\).jpg](http://en.wikipedia.org/w/index.php?title=File:We are trying to understand the difference_(2).jpg) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

Image:Quranic-arabic-corpus.png *Source:* <http://en.wikipedia.org/w/index.php?title=File:Quranic-arabic-corpus.png> *License:* Creative Commons Attribution 3.0 *Contributors:* Arabismo

File:Conventions.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Conventions.jpg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Semantic dependencies.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Semantic_dependencies.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Morphological dependencies 1.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Morphological_dependencies_1.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Morphological dependencies 2'.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Morphological_dependencies_2'.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Prosodic dependencies'.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Prosodic_dependencies'.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Dg-new-1.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Dg-new-1.jpg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Dg-new-2.jpg *Source:* <http://en.wikipedia.org/w/index.php?title=File:Dg-new-2.jpg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Syntactic functions 1.png *Source:* http://en.wikipedia.org/w/index.php?title=File:Syntactic_functions_1.png *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Tjo3ya

File:Meaningtext1.gif *Source:* <http://en.wikipedia.org/w/index.php?title=File:Meaningtext1.gif> *License:* Public Domain *Contributors:* David Beck

Image:Abiword grammar.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Abiword_grammar.jpg *License:* GNU General Public License *Contributors:* German, Jonsafari, Webaware

Image:Opened Qur'an.jpg *Source:* http://en.wikipedia.org/w/index.php?title=File:Opened_Qur'an.jpg *License:* Creative Commons Attribution 2.0 *Contributors:* Flickr.com user "el7bara"

License

Creative Commons Attribution-Share Alike 3.0 Unported
[//creativecommons.org/licenses/by-sa/3.0/](http://creativecommons.org/licenses/by-sa/3.0/)