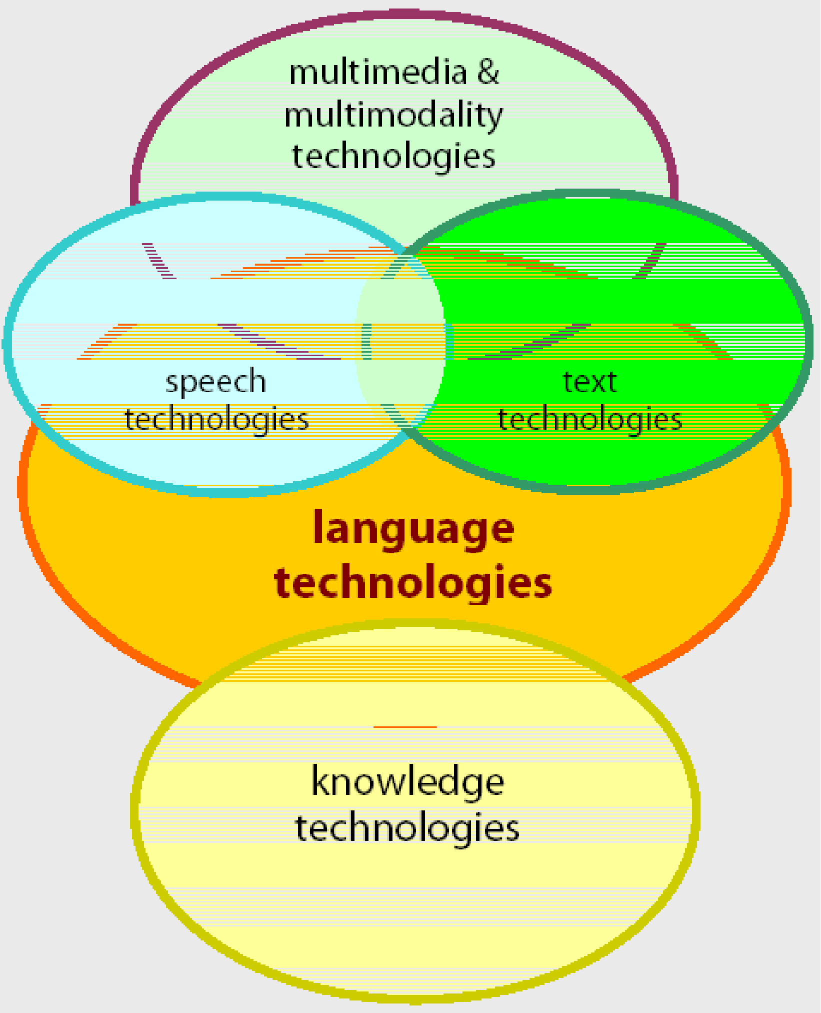
Introduction to NLP

Scope

- ***Language technologies*** are information technologies that are specialized for dealing with the most complex information medium in our world: human language (Human Language Technology).



Applications

- Although existing LT systems are far from achieving human ability, they have numerous possible applications.
- The goal is to create software products that have some knowledge of human language.
- Such products are going to change our lives.
- They are urgently needed for improving human-machine interaction since the main obstacle in the interaction between human and computer is merely a communication problem.
- Today's computers do not understand our language but computer languages are difficult to learn and do not correspond to the structure of human thought.
- Even if the language the machine understands and its domain of discourse are very restricted, the use of human language can increase the acceptance of software and the productivity of its users.

Applications

Friendly technology should listen and speak

- Applications of natural language interfaces
  - Database queries, information retrieval from texts, so-called expert systems, and robot control.
- Spoken language needs to be combined with other modes of communication such as pointing with mouse or finger.
  - If such multimodal communication is finally embedded in an effective general model of cooperation, we have succeeded in turning the machine into a partner.
  - The ultimate goal of research is the omnipresent access to all kinds of technology and to the global information structure by natural interaction.

Applications

Machines can also help people communicate with each other

- *One of the original aims of language technology has always been fully automatic translation between human languages.*
  - Still far away from achieving the ambitious goal of translating unrestricted texts.
  - Nevertheless, they have been able to create software systems that simplify the work of human translators and clearly improve their productivity.
  - Less than perfect automatic translations can also be of great help to information seekers who have to search through large amounts of texts in foreign languages.
- *The most serious bottleneck for e-commerce is the volume of communication between business and customers or among businesses.*
  - Language technology can help to sort, filter and route incoming email.
  - It can also assist the customer relationship agent to look up information and to compose a response.
  - In cases where questions have been answered before, language technology can find appropriate earlier replies and automatically respond.

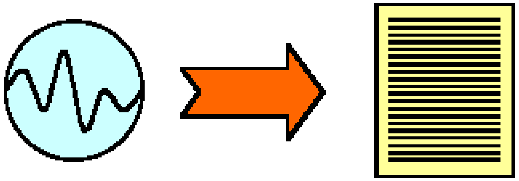
Applications

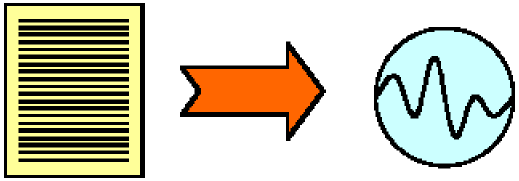
Language is the fabric of the web

- Although the new media combine text, graphics, sound and movies, the whole world of multimedia information can only be structured, indexed and navigated through language.
  - For browsing, navigating, filtering and processing the information on the web, we need software that can get at the contents of documents.
- Language technology for content management is a necessary precondition for turning the wealth of digital information into collective knowledge.
- The increasing multilinguality of the web constitutes an additional challenge for language technology.
  - The global web can only be mastered with the help of multilingual tools for indexing and navigating.
  - Systems for crosslingual information and knowledge management will surmount language barriers for e-commerce, education and international cooperation.

Technologies

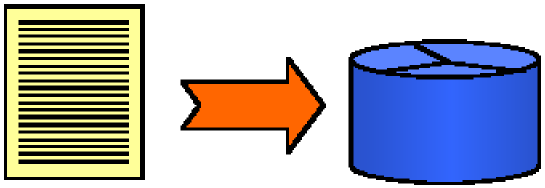
- **Speech recognition**
  - Spoken language is recognized and transformed in into text as in dictation systems, into commands as in robot control systems, or into some other internal representation.
- **Speech synthesis**
  - Utterances in spoken language are produced from text (text-to-speech systems) or from internal representations of words or sentences (concept-to-speech systems)

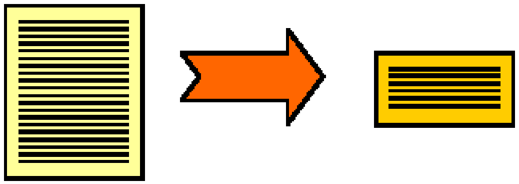




Technologies

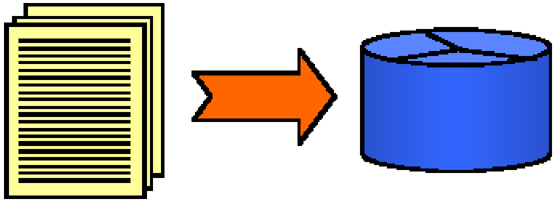
- **Text categorization**
  - This technology assigns texts to categories. Texts may belong to more than one category, categories may contain other categories. Filtering is a special case of categorization with just two categories.
- **Text Summarization**
  - The most relevant portions of a text are extracted as a summary. The task depends on the needed lengths of the summaries. Summarization is harder if the summary has to be specific to a certain query.



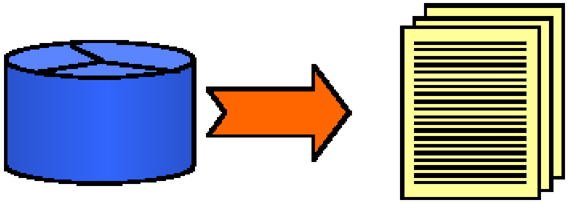


Technologies

- **Text Indexing**
  - As a precondition for document retrieval, texts are stored in an indexed database. Usually a text is indexed for all word forms or – after lemmatization – for all lemmas. Sometimes indexing is combined with categorization and summarization.

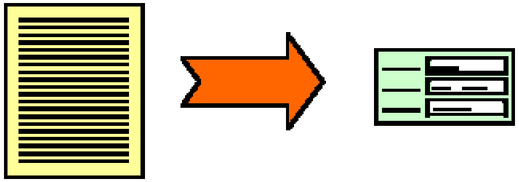


- **Text Retrieval**
  - Texts are retrieved from a database that best match a given query or document. The candidate documents are ordered with respect to their expected relevance. Indexing, categorization, summarization and retrieval are often subsumed under the term information retrieval.

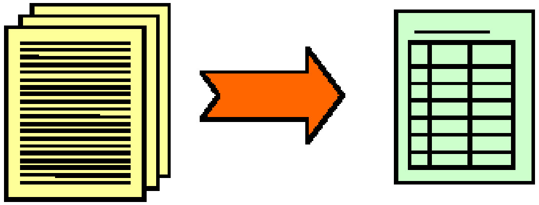


Technologies

- **Information Extraction**
  - Relevant information pieces of information are discovered and marked for extraction. The extracted pieces can be: the topic, named entities such as company, place or person names, simple relations such as prices, destinations, functions etc. or complex relations describing accidents, company mergers or football matches.

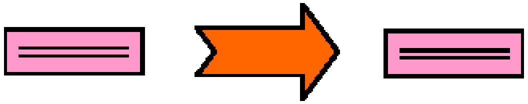


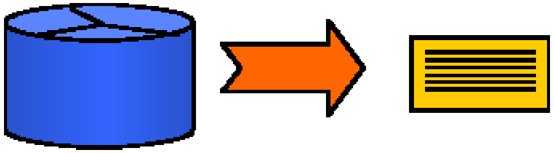
- **Data Fusion and Text Data Mining**
  - Extracted pieces of information from several sources are combined in one database. Previously undetected relationships may be discovered.



Technologies

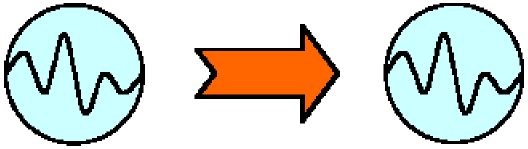
- **Question Answering**
  - Natural language queries are used to access information in a database. The database may be a base of structured data or a repository of digital texts in which certain parts have been marked as potential answers.
- **Report Generation**
  - A report in natural language is produced that describes the essential contents or changes of a database. The report can contain accumulated numbers, maxima, minima and the most drastic changes.

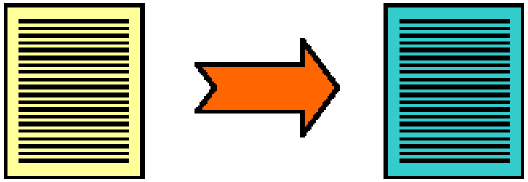




Technologies

- **Spoken Dialogue Systems**
  - The system can carry out a dialogue with a human user in which the user can solicit information or conduct purchases, reservations or other transactions.
- **Translation Technologies**
  - Technologies that translate texts or assist human translators. Automatic translation is called machine translation. Translation memories use large amounts of texts together with existing translations for efficient look-up of possible translations for words, phrases and sentences.





Methods and Resources

- The methods of language technology come from several disciplines:
  - computer science,
  - computational and theoretical linguistics,
  - mathematics,
  - electrical engineering and
  - psychology.

Methods and Resources

- **Generic CS Methods**
  - Programming languages, algorithms for generic data types, and software engineering methods for structuring and organizing software development and quality assurance.
- **Specialized Algorithms**
  - Dedicated algorithms have been designed for parsing, generation and translation, for morphological and syntactic processing with finite state automata/transducers and many other tasks.
- **Nondiscrete Mathematical Methods**
  - Statistical techniques have become especially successful in speech processing, information retrieval, and the automatic acquisition of language models. Other methods in this class are neural networks and powerful techniques for optimization and search.

Methods and Resources

- **Logical and Linguistic Formalisms**
  - For deep linguistic processing, constraint based grammar formalisms are employed. Complex formalisms have been developed for the representation of semantic content and knowledge.
- **Linguistic Knowledge**
  - Linguistic knowledge resources for many languages are utilized: dictionaries, morphological and syntactic grammars, rules for semantic interpretation, pronunciation and intonation.
- **Corpora and Corpus Tools**
  - Large collections of application-specific or generic collections of spoken and written language are exploited for the acquisition and testing of statistical or rule-based language models.

Introduction to NLP

From: Chapter 1 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, By Daniel Jurafsky and James H. Martin

http://www.cs.colorado.edu/~martin/SLP/slp-ch1.pdf

Chapter 1. Introduction to NLP

From: Chapter 1 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- The HAL 9000 computer in Stanley Kubrick's film *2001: A Space Odyssey*
  - HAL is an artificial agent capable of such advanced language processing behavior as speaking and understanding English, and at a crucial moment in the plot, even reading lips.
- The language-related parts of HAL
  - Speech recognition
  - Natural language understanding (and, of course, lip-reading),
  - Natural language generation
  - Speech synthesis
  - Information retrieval
  - information extraction and
  - inference

Background

- Solving the language-related problems and others like them, is the main concern of the fields known as Natural Language Processing, Computational Linguistics, and Speech Recognition and Synthesis, which together we call **Speech and Language Processing(SLP)**.
- Applications of language processing
  - **spelling correction**,
  - **grammar checking**,
  - **information retrieval**, and
  - **machine translation**.

1.1 Knowledge in Speech and Language Processing

- By SLP, we have in mind those computational techniques that process spoken and written human language, *as language*.
- What distinguishes these language processing applications from other data processing systems is their use of *knowledge of language*.
- Unix wc program
  - When used to count bytes and lines, wc is an ordinary data processing application.
  - However, when it is used to count the words in a file it requires *knowledge about what it means to be a word*, and thus becomes a language processing system.

1.1 Knowledge in Speech and Language Processing

- Both the tasks of being capable of analyzing an incoming audio signal and recovering the exact sequence of words and generating its response require knowledge about **phonetics and phonology**, which can help model how words are pronounced in colloquial speech (Chapters 4 and 5).
- Producing and recognizing the variations of individual words (e.g., recognizing that *doors* is plural) requires knowledge about **morphology**, which captures information about the shape and behavior of words in context (Chapters 2 and 3).

1.1 Knowledge in Speech and Language Processing

- **Syntax:** the knowledge needed to order and group words together

HAL, the pod bay door is open.
HAL, is the pod bay door open?

I'm I do, sorry that afraid Dave I'm can't.
(Dave, I'm sorry I'm afraid I can't do that.)

1.1 Knowledge in Speech and Language Processing

- **Lexical semantics:** knowledge of the meanings of the component words
- **Compositional semantics:** knowledge of how these components combine to form larger meanings
  - To know that Dave's command is actually about opening the pod bay door, rather than an inquiry about the day's lunch menu.

1.1 Knowledge in Speech and Language Processing

- **Pragmatics:** the appropriate use of the kind of polite and indirect language

No or
No, I won't open the door.

I'm sorry, I'm afraid, I can't.
I won't.

1.1 Knowledge in Speech and Language Processing

- **discourse conventions:** knowledge of correctly structuring these such conversations
  - HAL chooses to engage in a structured conversation relevant to Dave's initial request. HAL's correct use of the word ***that*** in its answer to Dave's request is a simple illustration of the kind of between-utterance device common in such conversations.

Dave, I'm sorry I'm afraid I can't do that.

1.1 Knowledge in Speech and Language Processing

- Phonetics and Phonology **—** The study of linguistic sounds
- Morphology **—**The study of the meaningful components of words
- Syntax **—**The study of the structural relationships between words
- Semantics **—** The study of meaning
- Pragmatics **—** The study of how language is used to accomplish goals
- Discourse**—**The study of linguistic units larger than a single utterance

1.2 Ambiguity

- A perhaps surprising fact about the six categories of linguistic knowledge is that most or all tasks in speech and language processing can be viewed as resolving **ambiguity** at one of these levels.
- We say some input is ambiguous
  - if there are multiple alternative linguistic structures than can be built for it.
- The spoken sentence, *I made her duck*, has five different meanings.
  - (1.1) I cooked waterfowl for her.
  - (1.2) I cooked waterfowl belonging to her.
  - (1.3) I created the (plaster?) duck she owns.
  - (1.4) I caused her to quickly lower her head or body.
  - (1.5) I waved my magic wand and turned her into undifferentiated waterfowl.

1.2 Ambiguity

- These different meanings are caused by a number of ambiguities.
  - *Duck* can be a verb or a noun, while *her* can be a dative pronoun or a possessive pronoun.
  - The word *make* can mean *create* or *cook*.
  - Finally, the verb *make* is syntactically ambiguous in that it can be transitive (1.2), or it can be ditransitive (1.5).
  - Finally, *make* can take a direct object and a verb (1.4), meaning that the object (*her*) got caused to perform the verbal action (*duck*).
  - In a spoken sentence, there is an even deeper kind of ambiguity; the first word could have been *eye* or the second word *maid*.

1.2 Ambiguity

- Ways to **resolve** or **disambiguate** these ambiguities:
  - Deciding whether *duck* is a verb or a noun can be solved by **part-of-speech tagging** . ▶
  - Deciding whether *make* means “create” or “cook” can be solved by **word sense disambiguation**.
  - Resolution of part-of-speech and word sense ambiguities are two important kinds of **lexical disambiguation**.
- A wide variety of tasks can be framed as lexical disambiguation problems.
  - For example, a text-to-speech synthesis system reading the word *lead* needs to decide whether it should be pronounced as in *lead pipe* or as in *lead me on*.
- Deciding whether *her* and *duck* are part of the same entity (as in (1.1) or (1.4)) or are different entity (as in (1.2)) is an example of **syntactic disambiguation** and can be addressed by **probabilistic parsing**.
- Ambiguities that don’t arise in this particular example (like whether a given sentence is a statement or a question) will also be resolved, for example by **speech act interpretation**.

1.3 Models and Algorithms

- The most important model:
  - **state machines**,
  - **formal rule systems**,
  - **logic**,
  - **probability theory** and
  - other machine learning tools
- The most important algorithms of these models:
  - **state space search** algorithms and
  - **dynamic programming** algorithms

1.3 Models and Algorithms

- State machines are
  - formal models that consist of *states*, *transitions* among states, and an *input representation*.
- Some of the variations of this basic model:
  - **Deterministic** and **non-deterministic finite-state automata**,
  - **finite-state transducers**, which can write to an output device,
  - **weighted automata**, **Markov models**, and **hidden Markov models**, which have a probabilistic component.

1.3 Models and Algorithms

- Closely related to the above procedural models are their declarative counterparts: *formal rule systems*.
  - **regular grammars** and **regular relations**, **context-free grammars**, **feature-augmented grammars**, as well as probabilistic variants of them all.
- *State machines and formal rule systems are the main tools used when dealing with knowledge of phonology, morphology, and syntax.*
- The algorithms associated with both state-machines and formal rule systems typically involve a *search through a space of states representing hypotheses about an input*.
- Representative tasks include
  - searching through a space of phonological sequences for a likely input word in speech recognition, or
  - searching through a space of trees for the correct syntactic parse of an input sentence.
- Among the algorithms that are often used for these tasks are well-known graph algorithms such as **depth-first search**, as well as heuristic variants such as **best-first**, and **A* search**.
- The dynamic programming paradigm is critical to the computational tractability of many of these approaches by ensuring that redundant computations are avoided.

1.3 Models and Algorithms

- The third model that plays a critical role in capturing knowledge of language is *logic*.
- We will discuss
  - **first order logic**, also known as the **predicate calculus**, as well as
  - such related formalisms as feature-structures,
  - semantic networks, and
  - conceptual dependency.
- *These logical representations have traditionally been the tool of choice when dealing with knowledge of semantics, pragmatics, and discourse* (although, as we will see, applications in these areas are increasingly relying on the simpler mechanisms used in phonology, morphology, and syntax).

1.3 Models and Algorithms

- Each of the other models (state machines, formal rule systems, and logic) can be augmented with probabilities.
- One *major use of probability theory* is to solve the many kinds of ambiguity problems that we discussed earlier;
  - almost any speech and language processing problem can be recast as: "*given N choices for some ambiguous input, choose the most probable one*".
- Another *major advantage of probabilistic models* is that
  - they are one of a class of **machine learning** models.
- Machine learning research has focused on ways to automatically learn the various representations described above;
  - automata, rule systems, search heuristics, classifiers.
- These systems can be trained on large corpora and can be used as a powerful modeling technique, especially in places where we don't yet have good causal models.

1.4 Language, Thought, and Understanding

- The effective use of language is intertwined with our general cognitive abilities.
- **Turing Test** by Alan Turing (1950) ▶
  - He suggested an empirical test, a game, in which a computer's use of language would form the basis for determining if it could think. If the machine could win the game it would be judged intelligent.

1.4 Language, Thought, and Understanding

- ELIZA program (Weizenbaum, 1966)
  - ELIZA was an early natural language processing system capable of carrying on a limited form of conversation with a user.
- Consider the following session with a version of ELIZA that imitated the responses of a Rogerian psychotherapist.

```
User1: You are like my father in some ways.
ELIZA1: WHAT RESEMBLANCE DO YOU SEE
User2: You are not very aggressive but I think you don’t want me to notice that.
ELIZA2: WHAT MAKES YOU THINK I AM NOT AGGRESSIVE
User3: You don’t argue with me.
ELIZA3: WHY DO YOU THINK I DON’T ARGUE WITH YOU
User4: You are afraid of me.
ELIZA4: DOES IT PLEASE YOU TO BELIEVE I AM AFRAID OF YOU
```

1.4 Language, Thought, and Understanding

- ELIZA is a remarkably simple program that makes use of pattern-matching to process the input and translate it into suitable outputs.
- The success of this simple technique in this domain is due to the fact that ELIZA doesn’t actually need to *know* anything to mimic a Rogerian psychotherapist.
- Eliza ▶
- A. L. I. C. E. Artificial Intelligence Foundation ▶
- Loebner Prize competition, since 1991,
  - An event has attempted to put various computer programs to the Turing test. ▶

1.5 The State of the Art and the Near-term Future

- Some current applications and near-term possibilities
  - A Canadian computer program accepts daily weather data and generates weather reports that are passed along unedited to the public in English and French (Chandioux, 1976).
  - The *Babel Fish* translation system from Systran handles over 1,000,000 translation requests a day from the AltaVista search engine site. ▶
  - A visitor to Cambridge, Massachusetts, asks a computer about places to eat using only spoken language. The system returns relevant information from a database of facts about the local restaurant scene (Zue et al., 1991).

1.5 The State of the Art and the Near-term Future

- Somewhat more speculative scenarios
  - A computer reads hundreds of typed student essays and grades them in a manner that is indistinguishable from human graders (Landauer et al., 1997).
  - An automated reading tutor helps improve literacy by having children read stories and using a speech recognizer to intervene when the reader asks for reading help or makes mistakes (Mostow and Aist, 1999).
  - A computer equipped with a vision system watches a short video clip of a soccer match and provides an automated natural language report on the game (Wahlster, 1989).
  - A computer predicts upcoming words or expands telegraphic speech to assist people with a speech or communication disability (Newell et al., 1998; McCoy et al., 1998).

1.6 Some Brief History

- Speech and language processing encompasses a number of different but overlapping fields in these different departments:
  - **computational linguistics** in linguistics,
  - **natural language processing** in computer science,
  - **speech recognition** in electrical engineering,
  - **computational psycholinguistics** in psychology.

1.6 Some Brief History

Foundational Insights: 1940s and 1950s

- Two foundational paradigms:
  - the automaton and
  - probabilistic or information-theoretic models
- Turing's work led first to the McCulloch-Pitts neuron (McCulloch and Pitts, 1943),
  - a simplified model of the neuron as a kind of computing element that could be described in terms of propositional logic,
- And then to the work of Kleene (1951) and (1956) on
  - finite automata and regular expressions.
- Shannon (1948) applied probabilistic models of discrete Markov processes to automata for language. (*continued*)

1.6 Some Brief History

Foundational Insights: 1940s and 1950s

- Chomsky (1956), drawing the idea of a finite state Markov process from Shannon’s work, first considered finite-state machines as a way to characterize a grammar, and defined a finite-state language as a language generated by a finite-state grammar.
- These early models led to the field of formal language theory, which used algebra and set theory to define formal languages as sequences of symbols.
  - This includes the context-free grammar, first defined by Chomsky (1956) for natural languages but independently discovered by Backus (1959) and Naur et al. (1960) in their descriptions of the ALGOL programming language.

1.6 Some Brief History

Foundational Insights: 1940s and 1950s

- The second foundational insight of this period was the development of probabilistic algorithms for speech and language processing, which dates to Shannon's other contribution:
  - the metaphor of the **noisy channel** and **decoding** for the transmission of language through media like communication channels and speech acoustics.
  - Shannon also borrowed the concept of **entropy** from thermodynamics as a way of measuring the information capacity of a channel, or the information content of a language, and performed the first measure of the entropy of English using probabilistic techniques.
  - It was also during this early period that the sound spectrograph was developed (Koenig et al., 1946), and foundational research was done in instrumental phonetics that laid the groundwork for later work in speech recognition.
    - This led to the first machine speech recognizers in the early 1950s.

1.6 Some Brief History

The Two Camps: 1957–1970

- By the end of the 1950s and the early 1960s, SLP had split very cleanly into two paradigms: *symbolic* and *stochastic*.
- The symbolic paradigm took off from two lines of research.
  - The **first** was the work of Chomsky and others on formal language theory and generative syntax throughout the late 1950s and early to mid 1960s, and the work of many linguistics and computer scientists on parsing algorithms, initially top-down and bottom-up and then via dynamic programming.
  - One of the earliest complete parsing systems was Zelig Harris's Transformations and Discourse Analysis Project (TDAP), which was implemented between June 1958 and July 1959 at the University of Pennsylvania (Harris, 1962). (*continued*)

1.6 Some Brief History

The Two Camps: 1957–1970

- The second line of research was the new field of artificial intelligence.
  - In the summer of 1956 John McCarthy, Marvin Minsky, Claude Shannon, and Nathaniel Rochester brought together a group of researchers for a two-month workshop on what they decided to call artificial intelligence (AI).
  - Although AI always included a minority of researchers focusing on stochastic and statistical algorithms (include probabilistic models and neural nets), the **major focus of the new field was the work on reasoning and logic** typified by Newell and Simon's work on the Logic Theorist and the General Problem Solver.
  - At this point **early natural language understanding systems were built**.
    - These were simple systems that worked in single domains mainly by a combination of pattern matching and keyword search with simple heuristics for reasoning and question-answering.
    - By the late 1960s more formal logical systems were developed.

1.6 Some Brief History

The Two Camps: 1957–1970

- The stochastic paradigm took hold mainly in departments of statistics and of electrical engineering.
  - By the late 1950s the Bayesian method was beginning to be applied to the **problem of optical character recognition**.
  - Bledsoe and Browning (1959) built a Bayesian system for text-recognition that used a large dictionary and computed the likelihood of each observed letter sequence given each word in the dictionary by multiplying the likelihoods for each letter.
  - Mosteller and Wallace (1964) applied Bayesian methods to the problem of authorship attribution on *The Federalist* papers.
  - The 1960s also saw the rise of the first serious testable psychological models of human language processing based on transformational grammar, as well as the first on-line corpora: the Brown corpus of American English, a 1 million word collection of samples from 500 written texts from different genres (newspaper, novels, non-fiction, academic, etc.), which was assembled at Brown University in 1963–64 (Kučera and Francis, 1967; Francis, 1979; Francis and Kučera, 1982), andWilliam S. Y.Wang's 1967 DOC (Dictionary on Computer), an on-line Chinese dialect dictionary.

1.6 Some Brief History

Four Paradigms: 1970–1983

- The next period saw an explosion in research in SLP and the development of a number of **research paradigms** that still dominate the field.
- The **stochastic** paradigm played a huge role in the development of *speech recognition* algorithms in this period,
  - particularly the use of the *Hidden Markov Model* and the metaphors of the noisy channel and decoding, developed independently by Jelinek, Bahl, Mercer, and colleagues at IBM's Thomas J. Watson Research Center, and by Baker at Carnegie Mellon University, who was influenced by the work of Baum and colleagues at the Institute for Defense Analyses in Princeton.
  - AT&T's Bell Laboratories was also a center for work on speech recognition and synthesis; see Rabiner and Juang (1993) for descriptions of the wide range of this work.

1.6 Some Brief History

Four Paradigms: 1970–1983

- The **logic-based** paradigm was begun by the work of Colmerauer and his colleagues on Q-systems and metamorphosis grammars (Colmerauer, 1970, 1975),
  - the forerunners of Prolog, and Definite Clause Grammars (Pereira andWarren, 1980).
  - Independently, Kay's (1979) work on functional grammar, and shortly later, Bresnan and Kaplan's (1982) work on LFG, established the importance of *feature structure unification*.

1.6 Some Brief History

Four Paradigms: 1970–1983

- The **natural language understanding** field took off during this period,
  - beginning with Terry Winograd's SHRDLU system, which simulated a robot embedded in a world of toy blocks (Winograd, 1972a).
    - The program was able to accept natural language text commands *(Move the red block on top of the smaller green one)* of a hitherto unseen complexity and sophistication.
    - His system was also the first to attempt to build an extensive (for the time) grammar of English, based on Halliday's systemic grammar.
  - Winograd's model made it clear that the problem of parsing was well-enough understood to begin to focus on semantics and discourse models.
  - Roger Schank and his colleagues and students (in what was often referred to as the *Yale School*) built a series of language understanding programs that focused on human conceptual knowledge such as scripts, plans and goals, and human memory organization (Schank and Albelson, 1977; Schank and Riesbeck, 1981; Cullingford, 1981; Wilensky, 1983; Lehnert, 1977).
  - This work often used network-based semantics (Quillian, 1968; Norman and Rumelhart, 1975; Schank, 1972; Wilks, 1975c, 1975b; Kintsch, 1974) and began to incorporate Fillmore's notion of *case roles* (Fillmore, 1968) into their representations (Simmons, 1973).
- The logic-based and natural-language understanding paradigms were unified on systems that used predicate logic as a semantic representation, such as the LUNAR question-answering system (Woods, 1967, 1973).

1.6 Some Brief History

Four Paradigms: 1970–1983

- The **discourse modeling** paradigm focused on four key areas in discourse.
  - Grosz and her colleagues introduced the study of **substructure in discourse**, and of **discourse focus** (Grosz, 1977a; Sidner, 1983),
  - a number of researchers began to work on **automatic reference resolution** (Hobbs, 1978),
  - and the **BDI** (Belief-Desire-Intention) framework for logic-based work on speech acts was developed (Perrault and Allen, 1980; Cohen and Perrault, 1979).

1.6 Some Brief History

Empiricism and Finite State Models Redux: 1983–1993

- This next decade saw the return of two classes of models which had lost popularity in the late 1950s and early 1960s, partially due to theoretical arguments against them such as Chomsky's influential review of Skinner's *Verbal Behavior* (Chomsky, 1959b).
  - The first class was finite-state models, which began to receive attention again after work on finite-state phonology and morphology by Kaplan and Kay (1981) and finite-state models of syntax by Church (1980).
  - The second trend in this period was what has been called the "return of empiricism"; most notably here was the rise of probabilistic models throughout speech and language processing, influenced strongly by the work at the IBM Thomas J. Watson Research Center on probabilistic models of speech recognition.
    - These probabilistic methods and other such data-driven approaches spread into part-of-speech tagging, parsing and attachment ambiguities, and connectionist approaches from speech recognition to semantics.
- This period also saw considerable work on natural language generation.

1.6 Some Brief History

The Field Comes Together: 1994–1999

- By the last five years of the millennium it was clear that the field was vastly changing.
  - First, probabilistic and data-driven models had become quite standard throughout natural language processing.
    - Algorithms for parsing, part-of-speech tagging, reference resolution, and discourse processing all began to incorporate probabilities, and employ evaluation methodologies borrowed from speech recognition and information retrieval.
  - Second, the increases in the speed and memory of computers had allowed commercial exploitation of a number of subareas of speech and language processing, in particular
    - speech recognition and spelling and grammar checking.
    - Speech and language processing algorithms began to be applied to Augmentative and Alternative Communication (AAC).
  - Finally, the rise of the Web emphasized the need for language-based information retrieval and information extraction.

1.7 Summary

- A good way to understand the concerns of speech and language processing research is to consider what it would take to create an intelligent agent like *HAL from 2001: A Space Odyssey*.
- Speech and language technology relies on formal models, or representations, of knowledge of language at the levels of phonology and phonetics, morphology, syntax, semantics, pragmatics and discourse.
  - A small number of formal models including state machines, formal rule systems, logic, and probability theory are used to capture this knowledge.

1.7 Summary

- The foundations of speech and language technology lie in computer science, linguistics, mathematics, electrical engineering and psychology.
- The critical connection between language and thought has placed speech and language processing technology at the center of debate over intelligent machines.
- Revolutionary applications of speech and language processing are currently in use around the world.
  - Recent advances in speech recognition and the creation of the World-Wide Web will lead to many more applications.

Chapter 2. Regular Expressions and Automata

From: Chapter 2 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

2.1 Regular Expressions

- In computer science, RE is a language used for specifying text search string.
- A *regular expression* is a *formula* in a *special language* that is used for specifying a simple class of *string*.
- Formally, a regular expression is an algebraic notation for characterizing a set of strings.
- RE search requires
  - a *pattern* that we want to search for, and
  - a *corpus* of texts to search through.

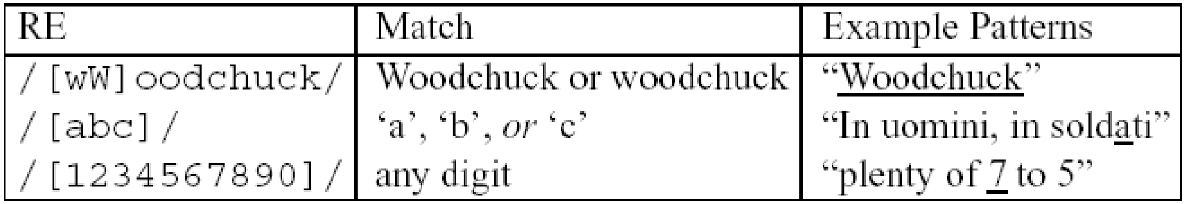
2.1 Regular Expressions

- A RE search function will search through the corpus returning all texts that contain the pattern.
  - In a Web search engine, they might be the entire documents or Web pages.
  - In a word-processor, they might be individual words, or lines of a document. (We take this paradigm.)
    - E.g., the UNIX `grep` command

2.1 Regular Expressions

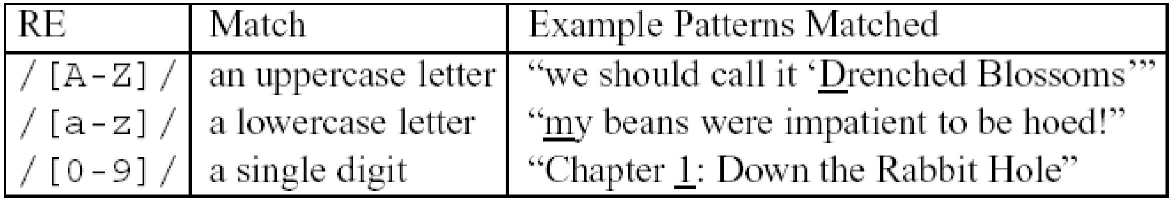
Basic Regular Expression Patterns

- The use of the brackets `[]` to specify a disjunction of characters.



RE	Match	Example Patterns
`/[wW]oodchuck/`	Woodchuck or woodchuck	"Woodchuck"
`/[abc]/`	'a', 'b', *or* 'c'	"In uomini, in soldati"
`/[1234567890]/`	any digit	"plenty of 7 to 5"

- The use of the brackets `[]` plus the dash `-` to specify a range.

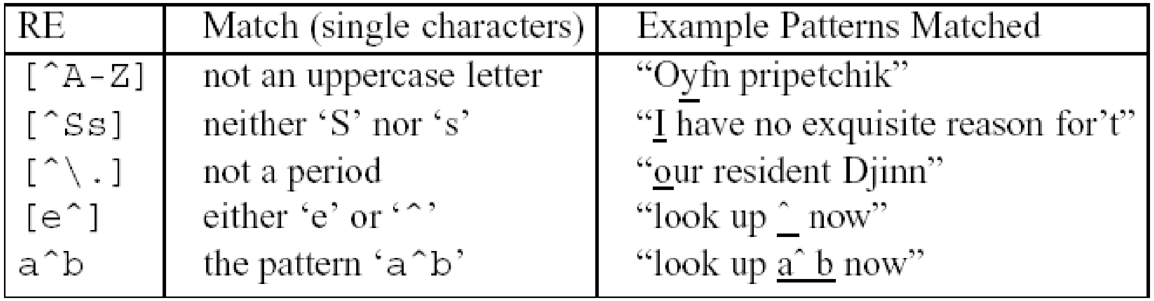


RE	Match	Example Patterns Matched
`/[A-Z]/`	an uppercase letter	"we should call it 'Drenched Blossoms'"
`/[a-z]/`	a lowercase letter	"my beans were impatient to be hoed!"
`/[0-9]/`	a single digit	"Chapter 1: Down the Rabbit Hole"

2.1 Regular Expressions

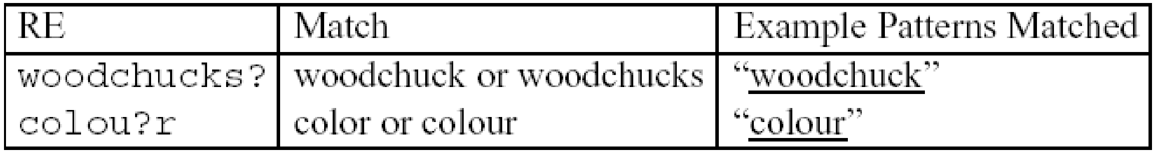
Basic Regular Expression Patterns

- Uses of the caret ^ for negation or just to mean ^



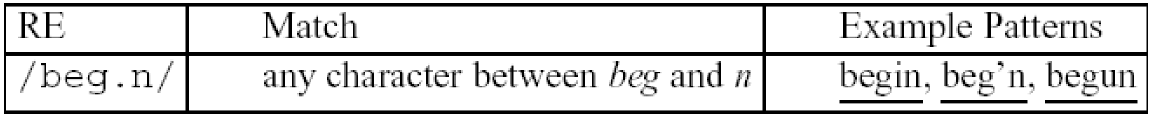
RE	Match (single characters)	Example Patterns Matched
`[^A-Z]`	not an uppercase letter	“Oyfn pripetchik”
`[^Ss]`	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
`[^\.]`	not a period	“our resident Djinn”
`[e^]`	either ‘e’ or ‘^’	“look up ^ now”
`a^b`	the pattern ‘`a^b`’	“look up a^ b now”

- The question-mark `?` marks optionality of the previous expression.



RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	“woodchuck”
`colou?r`	color or colour	“colour”

- The use of period `.` to specify any character



RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg’n, begun

2.1 Regular Expressions

Disjunction, Grouping, and Precedence

- Disjunction

```
/cat|dog
```

- Precedence

```
/gupp(y|ies)
```

- Operator precedence hierarchy

```
()
+ ? { }
the  ^my  end$
|
```

2.1 Regular Expressions

A Simple Example

- To find the English article *the*

```
/the/

/[tT]he/

/\b[tT]he\b/

/[^a-zA-Z][tT]he[^a-zA-Z]/

/^|[^a-zA-Z][tT]he[^a-zA-Z]/
```

2.1 Regular Expressions

A More Complex Example

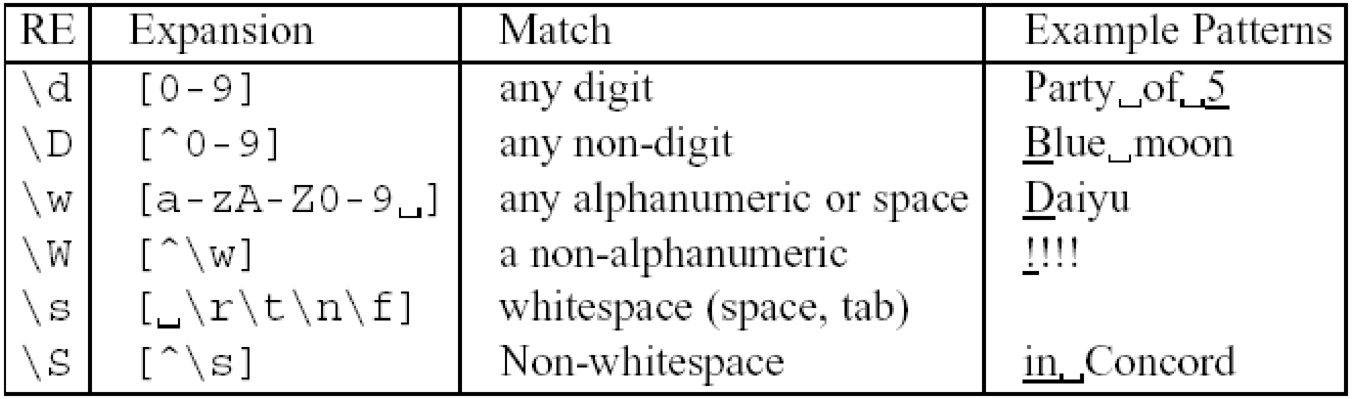
- "any PC with more than ***500 MHz*** and ***32 Gb*** of disk space for less than ***$1000***"

```
/$[0-9]+/

/$[0-9]+\.[0-9][0-9]/

/\b$[0-9]+(\.[0-9][0-9])?\b/

/\b[0-9]+ *(MHz|[Mm]egahertz|GHz|[Gg]igahertz)\b/

/\b[0-9]+ *(Mb|[Mm]egabytes?)\b/

/\b[0-9](\.[0-9]+)? *(Gb|[Gg]igabytes?)\b/

/\b(Win95|Win98|WinNT|Windows *(NT|95|98|2000)?)\b/

/\b(Mac|Macintosh|Apple)\b/
```

2.1 Regular Expressions

Advanced Operators

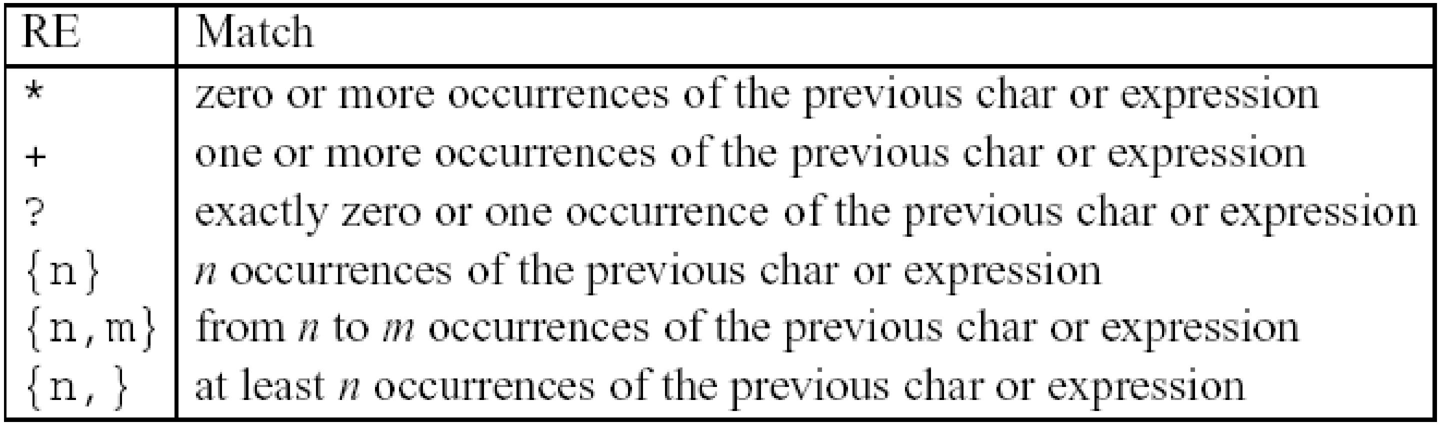
Aliases for common sets of characters



RE	Expansion	Match	Example Patterns
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9␣]	any alphanumeric or space	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

2.1 Regular Expressions
Advanced Operators

Regular expression operators for counting

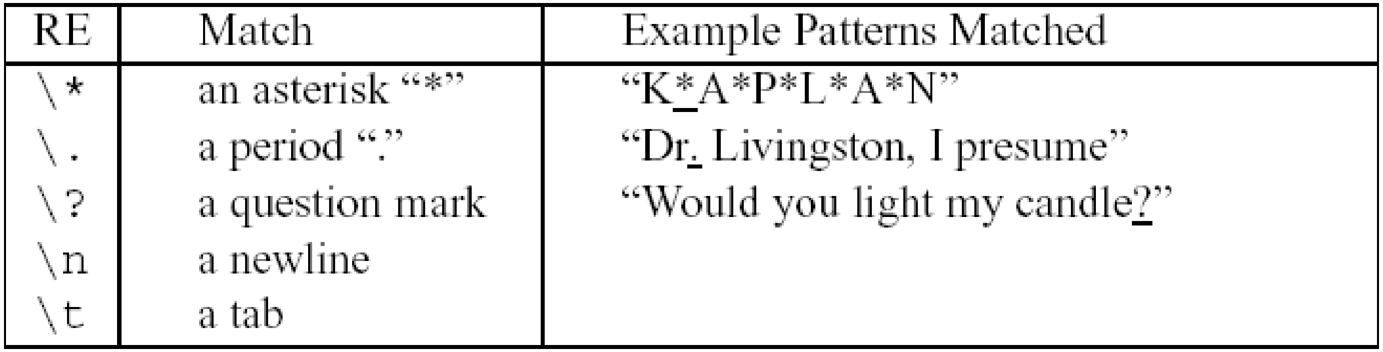


RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression

2.1 Regular Expressions

Advanced Operators

Some characters that need to be backslashed



RE	Match	Example Patterns Matched
`\*`	an asterisk “*”	“K*A*P*L*A*N”
`\.`	a period “.”	“Dr. Livingston, I presume”
`\?`	a question mark	“Would you light my candle?”
`\n`	a newline	
`\t`	a tab	

2.1 Regular Expressions

Regular Expression Substitution, Memory, and ELIZA

```
s/regexp1/regexp2/
```

- E.g. *the 35 boxes* → *the <35> boxes*

```
s/([0-9]+)/<\1>/
```

- The following pattern matches "*The bigger they were, the bigger they will be*", not *"The bigger they were, the faster they will be"*

```
/the (.*)er they were, the\1er they will be/
```

- The following pattern matches "*The bigger they were, the bigger they were*", not *"The bigger they were, the bigger they will be"*

```
/the (.*)er they (.*), the\1er they \2/
```

registers

2.1 Regular Expressions

Regular Expressions Substitution, Memory, and ELIZA

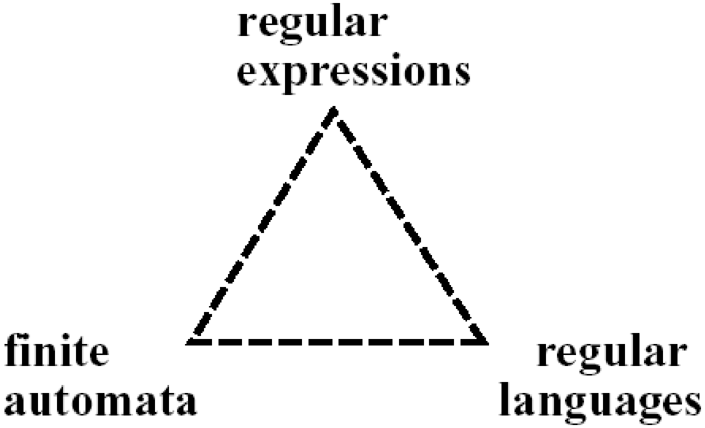
- Eliza worked by having a cascade of regular expression substitutions that each match some part of the input lines and changed them
  - *my → YOUR, I’m → YOU ARE …*

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

User$_1$: Men are all alike.
ELIZA$_1$: IN WHAT WAY
User$_2$: They’re always bugging us about something or other.
ELIZA$_2$: CAN YOU THINK OF A SPECIFIC EXAMPLE
User$_3$: Well, my boyfriend made me come here.
ELIZA$_3$: YOUR BOYBRIEND MADE YOU COME HERE
User$_4$: He says I’m depressed much of the time.
ELIZA$_4$: I AM SORRY TO HEAR YOU ARE DEPRESSED

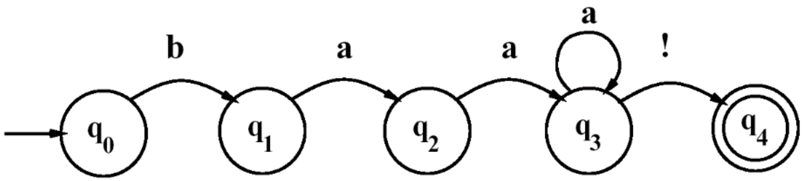
2.2 Finite-State Automata

- An RE is one way of describing a FSA.
- An RE is one way of characterizing a particular kind of formal language called a **regular language**.

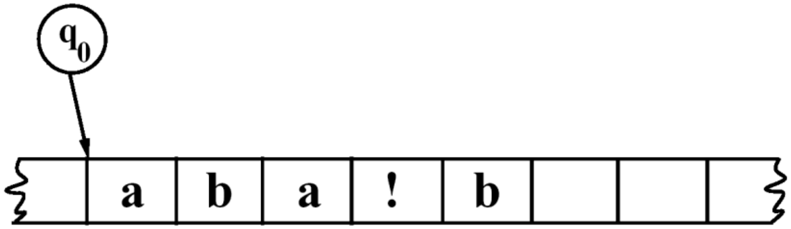


2.2 Finite-State Automata

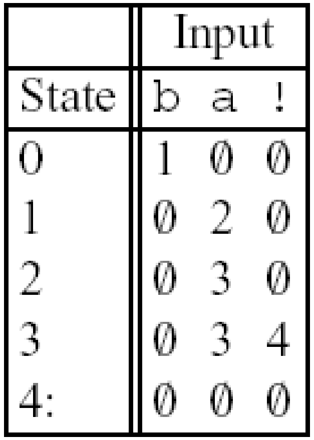
Using an FSA to Recognize Sheeptalk



/baa+!/



A tape with cells



	Input		
State	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅

The transition-state table

- Automaton (finite automaton, finite-state automaton (FSA))
- State, start state, final state (accepting state)

2.2 Finite-State Automata

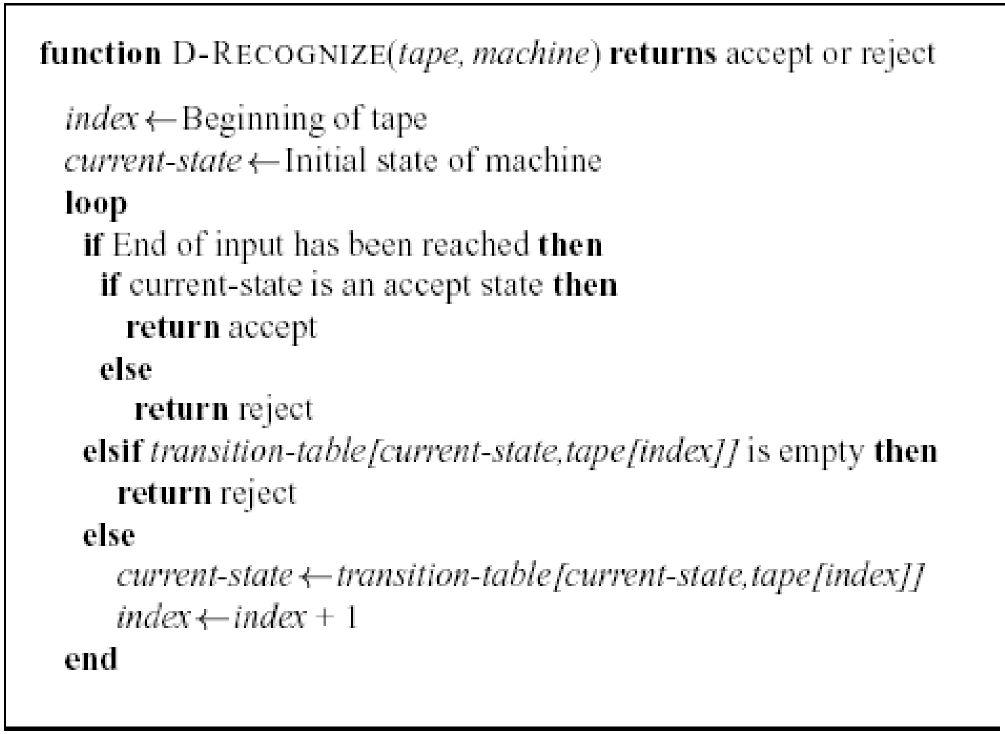
Using an FSA to Recognize Sheeptalk

- A finite automaton is formally defined by the following five parameters:
  - Q: a finite set of N states q_0, q_1, …, q_N
  - Σ: a finite input alphabet of symbols
  - q_0: the start state
  - F: the set of final states, $F \subseteq Q$
  - $\delta(q,i)$: the transition function or transition matrix between states. Given a state $q \in Q$ and input symbol $i \in \Sigma$, $\delta(q,i)$ returns a new state $q' \in Q$. δ is thus a relation from $Q \times \Sigma$ to Q;

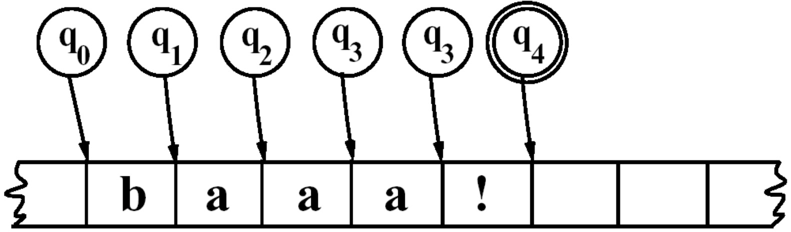
2.2 Finite-State Automata

Using an FSA to Recognize Sheeptalk

- An algorithm for deterministic recognition of FSAs.



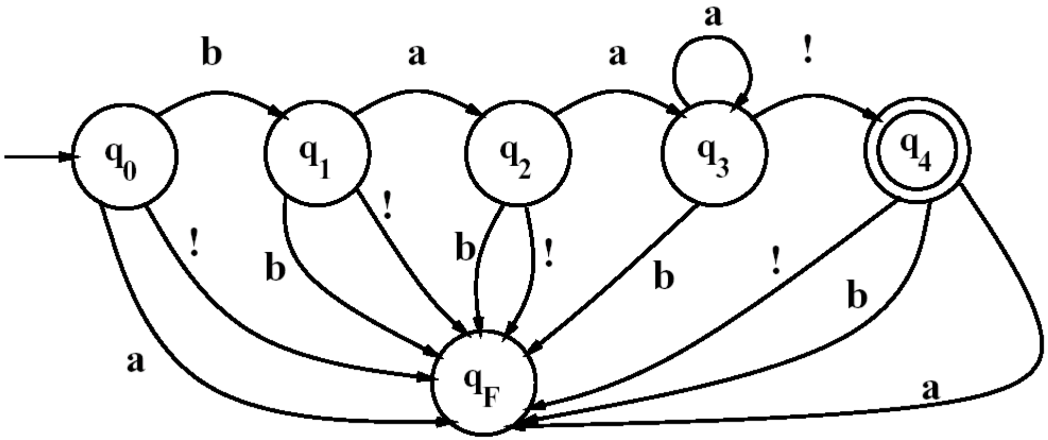
```
function D-RECOGNIZE(tape, machine) returns accept or reject

  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
    if End of input has been reached then
      if current-state is an accept state then
        return accept
      else
        return reject
    elsif transition-table[current-state,tape[index]] is empty then
      return reject
    else
      current-state ← transition-table[current-state,tape[index]]
      index ← index + 1
  end
```



2.2 Finite-State Automata

Using an FSA to Recognize Sheeptalk

- Adding a fail state



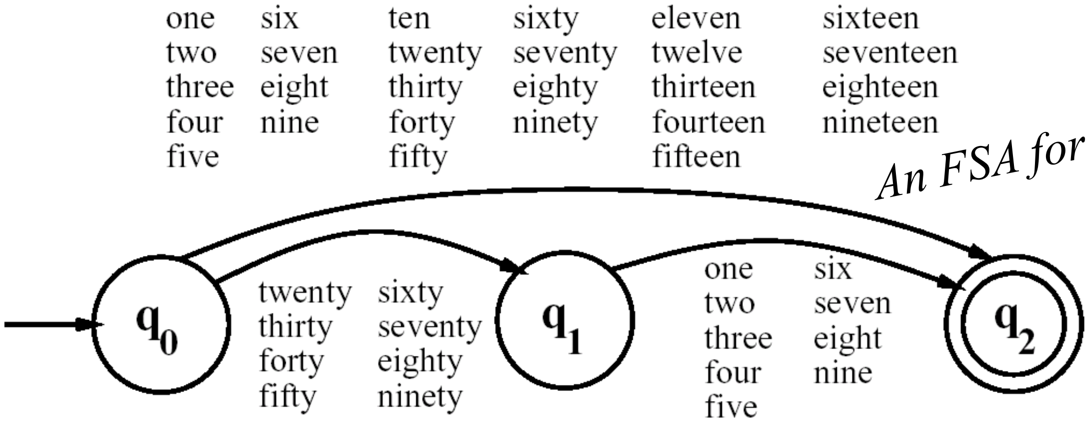
2.2 Finite-State Automata

Formal Languages

- **Key concept #1: Formal Language:** A model which can both generate and recognize all and only the strings of a formal language acts as a *definition* of the formal language.
- A **formal language** is a set of strings, each string composed of symbols from a finite symbol-set call an **alphabet**.
- The usefulness of an automaton for defining a language is that it can express an infinite set in a closed form.
- A formal language may bear no resemblance at all to a real language (natural language), but
  - We often use a formal language to model part of a natural language, such as parts of the phonology, morphology, or syntax.
- The term **generative grammar** is used in linguistics to mean a grammar of a formal language.

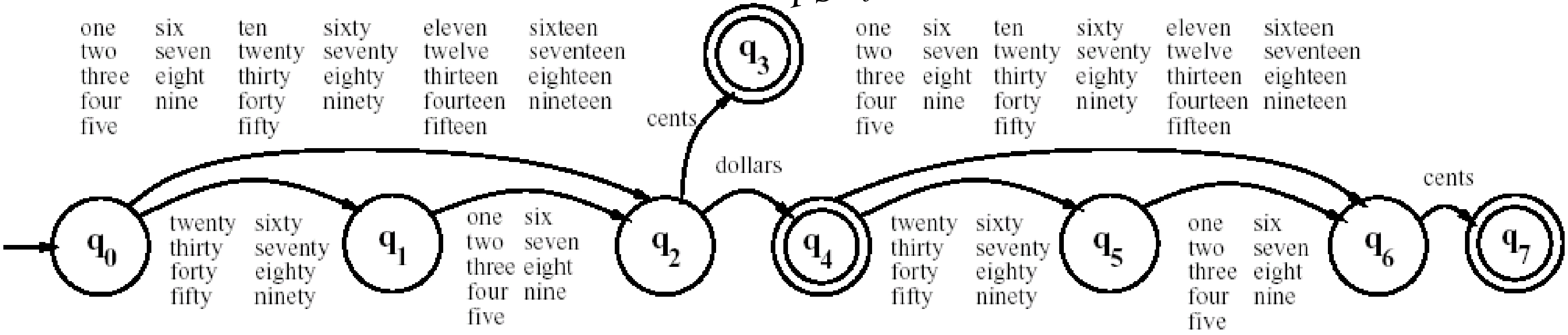
2.2 Finite-State Automata

Another Example


one two three four five
six seven eight nine
ten twenty thirty forty fifty
sixty seventy eighty ninety
eleven twelve thirteen fourteen fifteen
sixteen seventeen eighteen nineteen
q0
twenty thirty forty fifty
sixty seventy eighty ninety
q1
one two three four five
six seven eight nine
q2

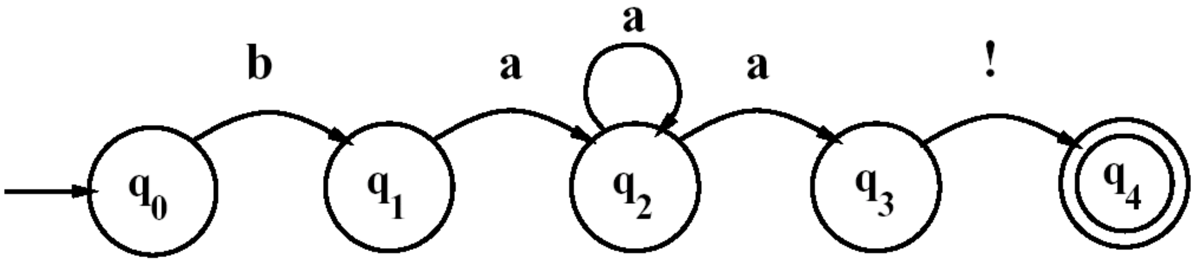
An FSA for the words of English numbers 1-99

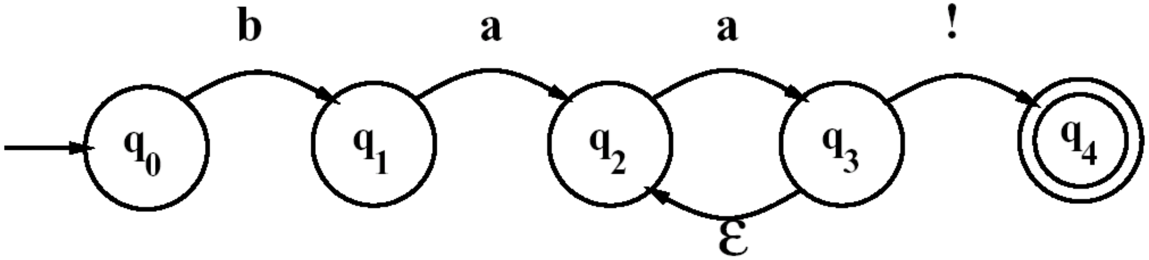
FSA for the simple dollars and cents


one two three four five
six seven eight nine
ten twenty thirty forty fifty
sixty seventy eighty ninety
eleven twelve thirteen fourteen fifteen
sixteen seventeen eighteen nineteen
q3
cents
dollars
one two three four five
six seven eight nine
ten twenty thirty forty fifty
sixty seventy eighty ninety
eleven twelve thirteen fourteen fifteen
sixteen seventeen eighteen nineteen
cents
q0
twenty thirty forty fifty
sixty seventy eighty ninety
q1
one two three four five
six seven eight nine
q2
q4
twenty thirty forty fifty
sixty seventy eighty ninety
q5
one two three four five
six seven eight nine
q6
q7

2.2 Finite-State Automata

Non-Deterministic FSAs

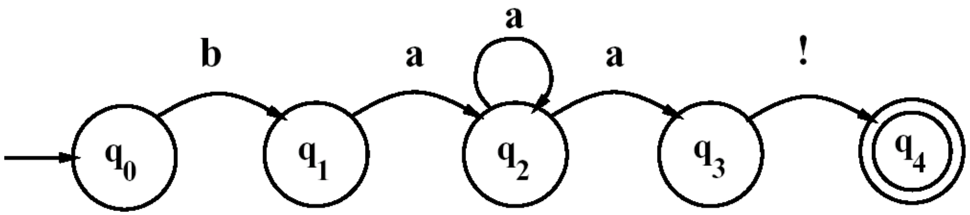


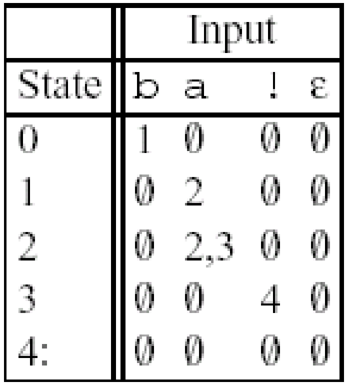


2.2 Finite-State Automata

Using an NFSA to Accept Strings

- Solutions to the problem of multiple choices in an NFSA
  - Backup
  - Look-ahead
  - Parallelism

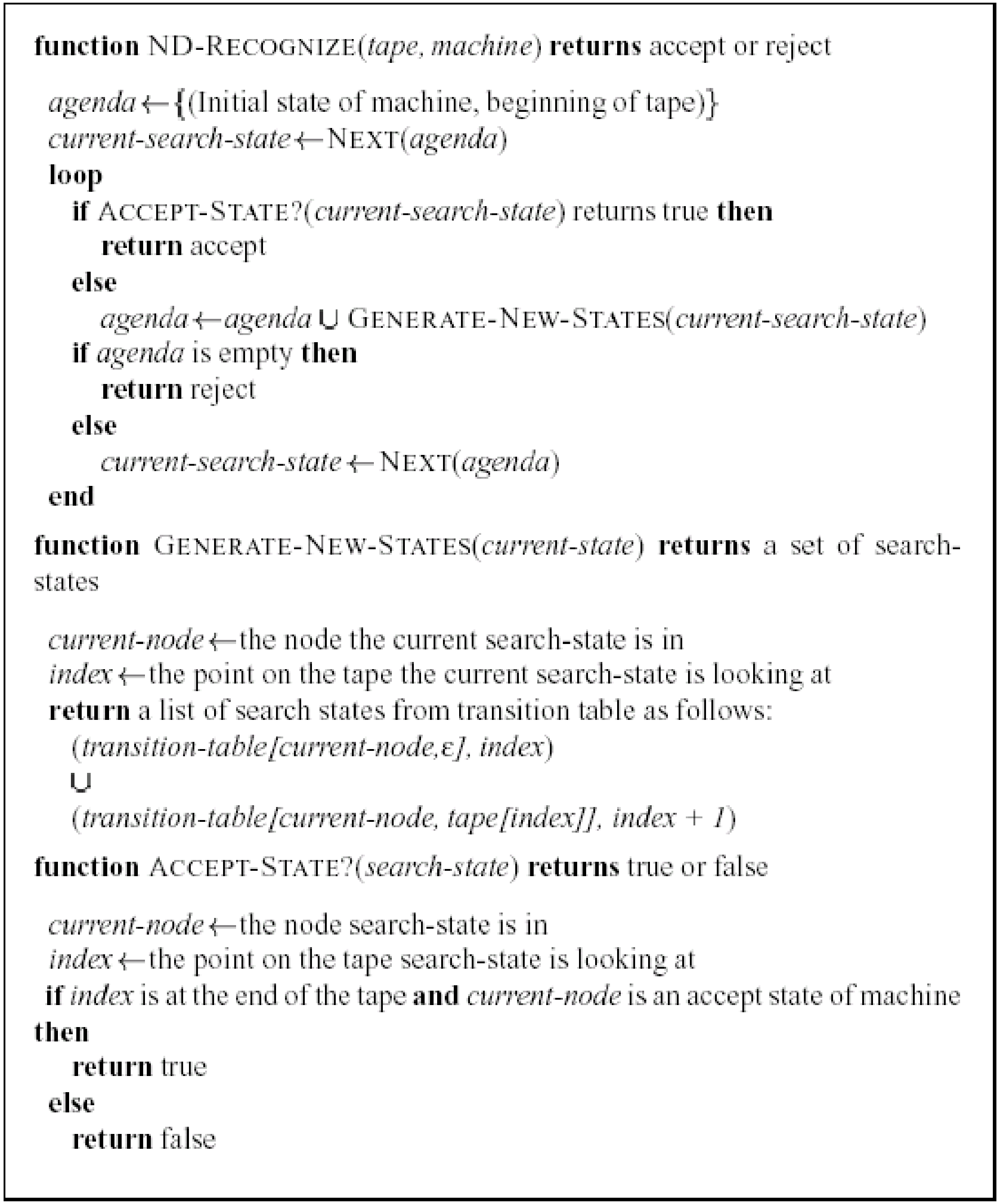




	Input			
State	b	a	!	ε
0	1	∅	∅	∅
1	∅	2	∅	∅
2	∅	2,3	∅	∅
3	∅	∅	4	∅
4:	∅	∅	∅	∅

2.2 Finite-State Automata

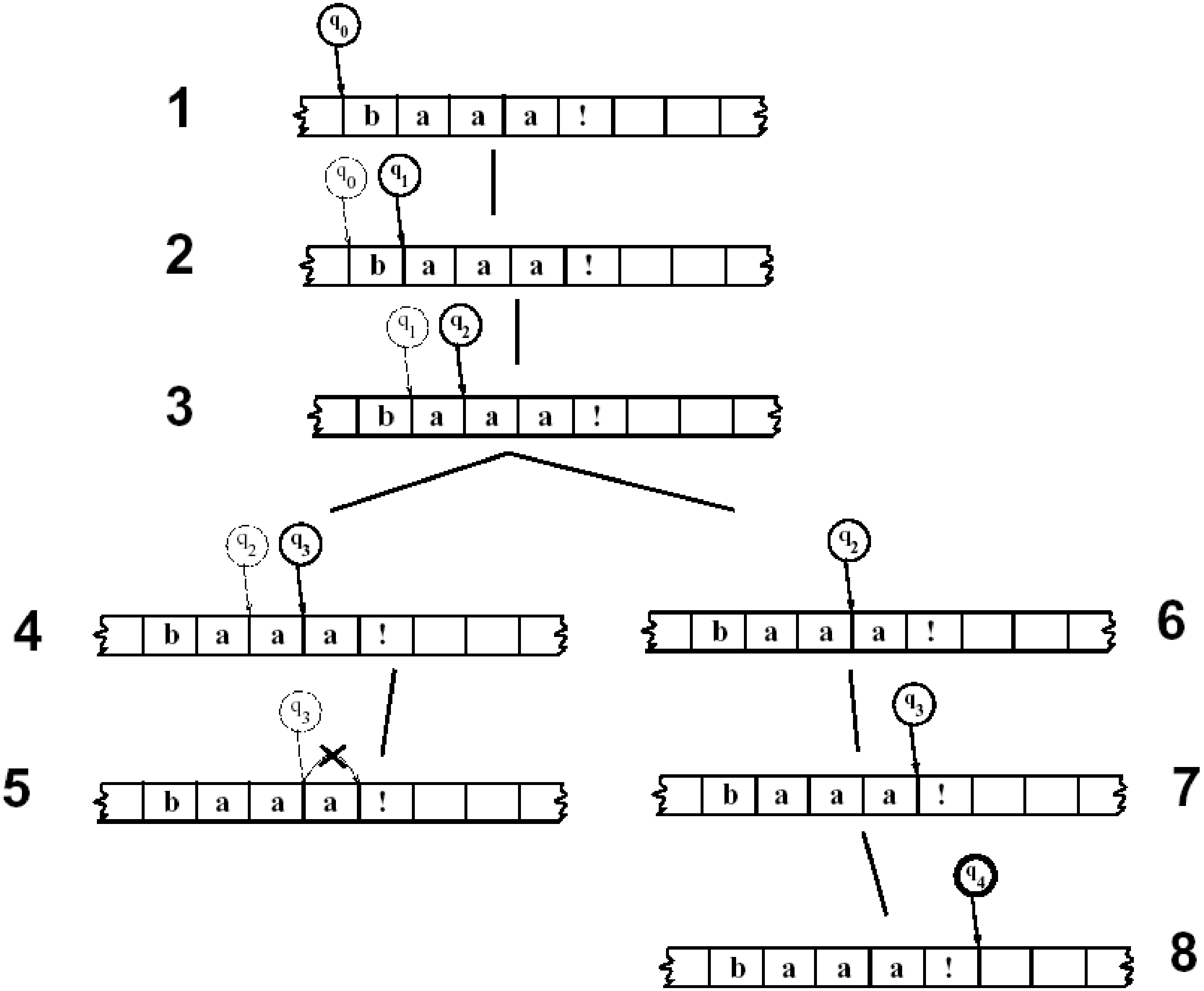
Using an NFSA to Accept Strings



```
function ND-RECOGNIZE(tape, machine) returns accept or reject

  agenda ← {(Initial state of machine, beginning of tape)}
  current-search-state ← NEXT(agenda)
  loop
    if ACCEPT-STATE?(current-search-state) returns true then
       return accept
    else
       agenda ← agenda ∪ GENERATE-NEW-STATES(current-search-state)
    if agenda is empty then
       return reject
    else
       current-search-state ← NEXT(agenda)
  end

function GENERATE-NEW-STATES(current-state) returns a set of search-
states

  current-node ← the node the current search-state is in
  index ← the point on the tape the current search-state is looking at
  return a list of search states from transition table as follows:
    (transition-table[current-node,ε], index)
    ∪
    (transition-table[current-node, tape[index]], index + 1)

function ACCEPT-STATE?(search-state) returns true or false

  current-node ← the node search-state is in
  index ← the point on the tape search-state is looking at
 if index is at the end of the tape and current-node is an accept state of machine
then
    return true
  else
    return false
```

2.2 Finite-State Automata

Using an NFSA to Accept Strings



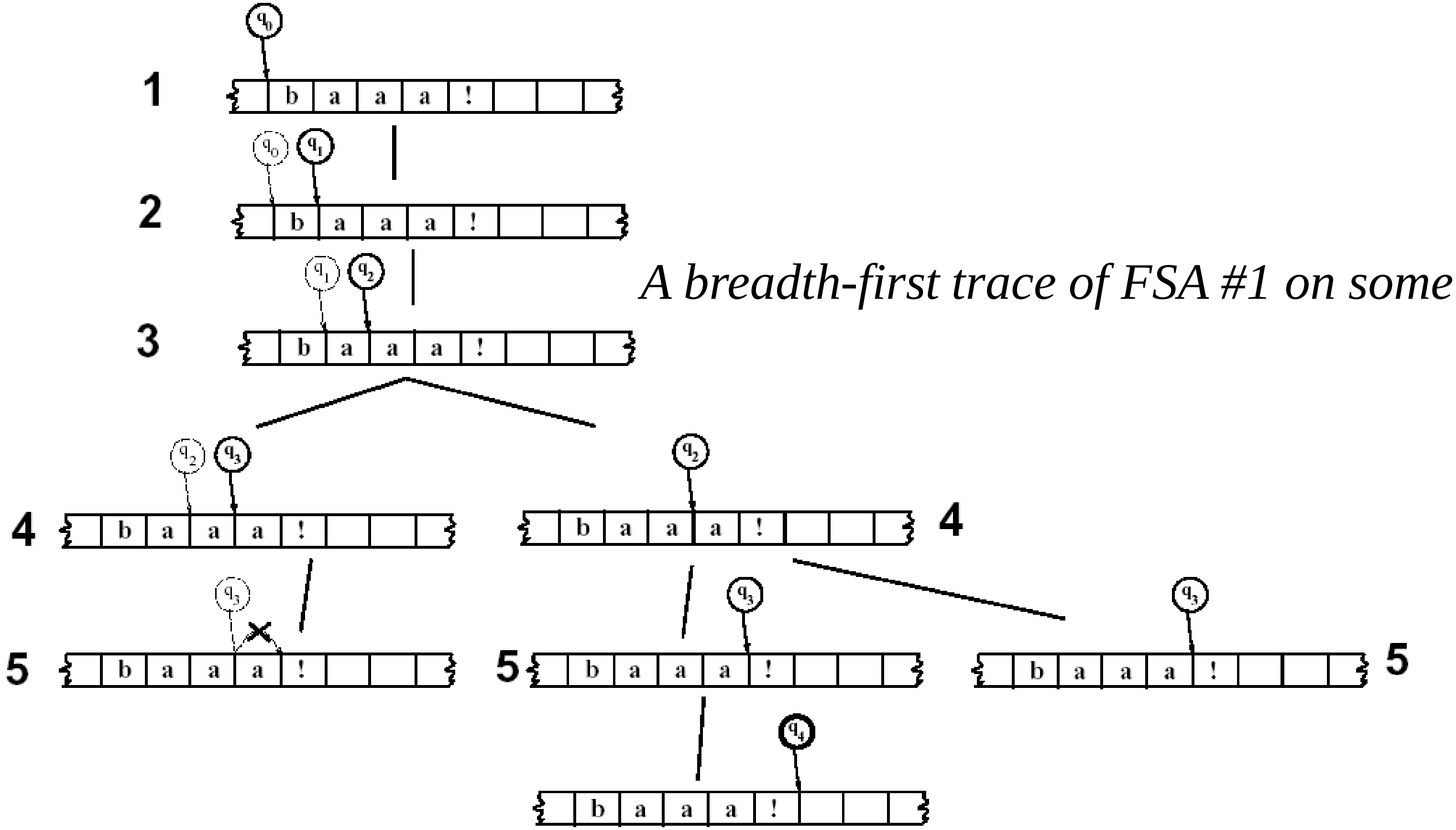
2.2 Finite-State Automata

Recognition as Search

- Algorithms such as ND-RECOGNIZE are known as **state-space search**
- **Depth-first search** or **Last In First Out** (**LIFO**) strategy
- **Breadth-first search** or **First In First Out** (**FIFO**) strategy
- More complex search techniques such as **dynamic programming** or **A***

2.2 Finite-State Automata

Recognition as Search



A breadth-first trace of FSA #1 on some sheeptalk

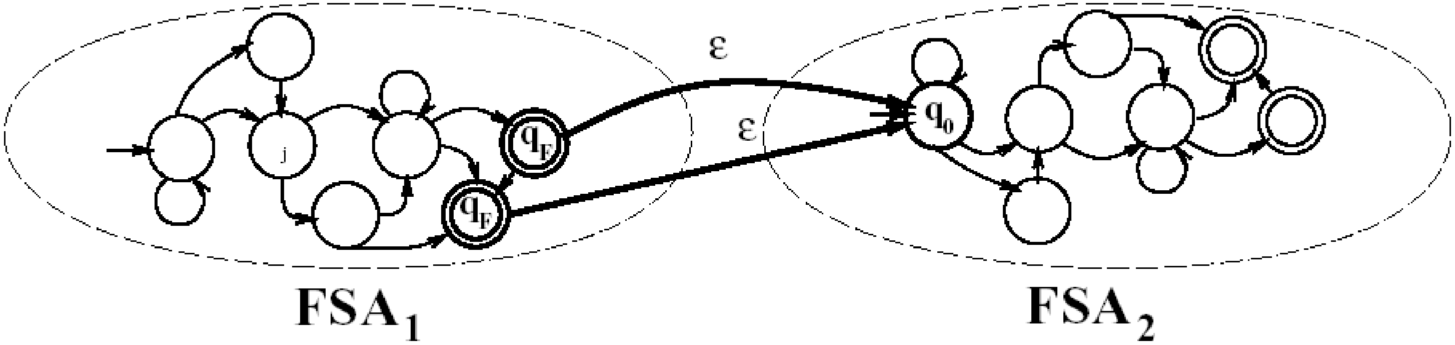
2.3 Regular Languages and FSAs

- The class of languages that are definable by regular expressions is exactly the same as the class of languages that are characterizable by FSA (D or ND).
  - These languages are called **regular languages**.
- The regular languages over Σ is formally defined as:
  1. ϕ is an RL
  2. $\forall a \in \Sigma$, $\{a\}$ is an RL
  3. If L_1 and L_2 are RLs, then so are:
     a) $L_1 \cdot L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$, the **concatenation** of L_1 and L_2

     b) $L_1 \cup L_2$, the **union** of L_1 and L_2

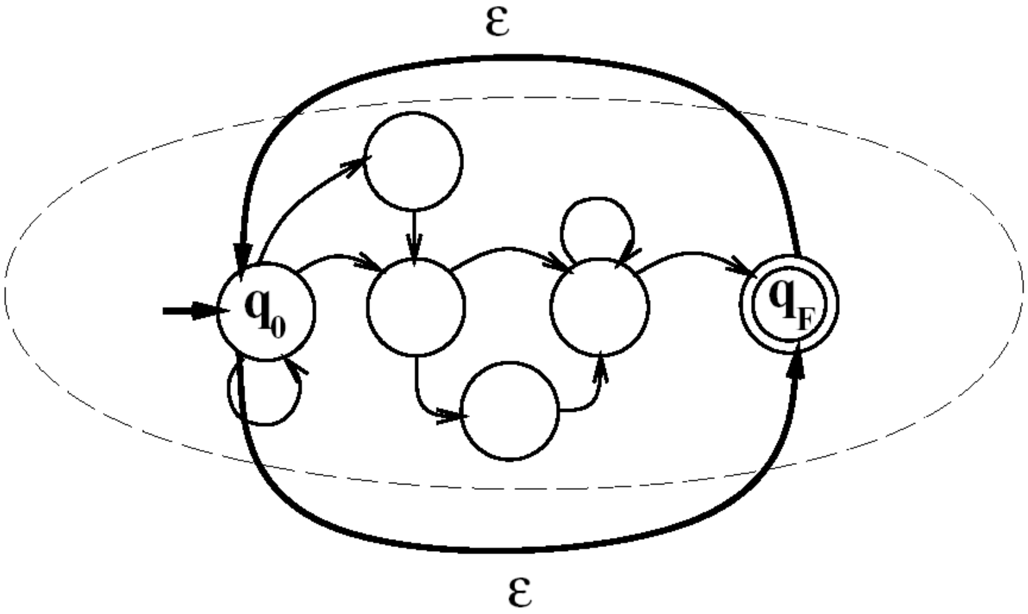
     c) $L_1{}^*$, the **Kleene closure** of L_1

2.3 Regular Languages and FSAs



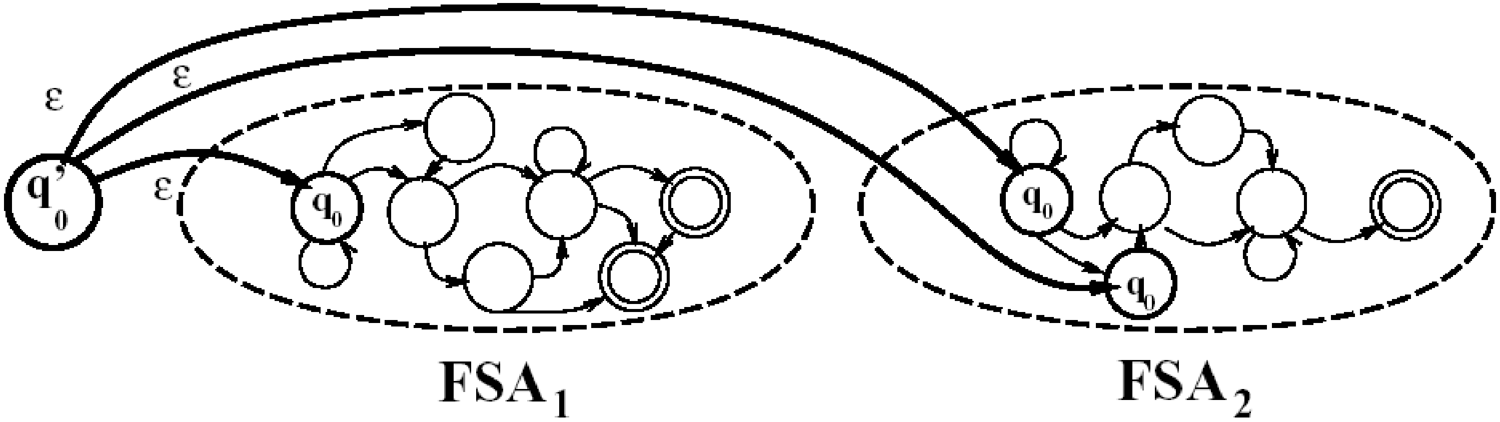
The concatenation of two FSAs

2.3 Regular Languages and FSAs



*The closure (Kleene *) of an FSAs*

2.3 Regular Languages and FSAs



The union (|) *of two FSAs*

Chapter 3. Morphology and Finite-State Transducers

From: Chapter 3 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- The problem of recognizing that *foxes* breaks down into the two morphemes *fox* and *-es* is called ***morphological parsing***.
- Similar problem in the information retrieval domain: ***stemming***
- Given the **surface** or **input form** *going*, we might want to produce the parsed form: `VERB-go + GERUND-ing`
- In this chapter
  - morphological knowledge and
  - The **finite-state transducer**
- It is quite inefficient to list all forms of noun and verb in the dictionary because the productivity of the forms.
- Morphological parsing is necessary more than just IR, but also
  - Machine translation
  - Spelling checking

3.1 Survey of (Mostly) English Morphology

- Morphology is the study of the way words are built up from smaller meaning-bearing units, **morphemes**.
- Two broad classes of morphemes:
  - **The stems:** the "main" morpheme of the word, supplying the main meaning, while
  - **The affixes:** add "additional" meaning of various kinds.
- Affixes are further divided into **prefixes**, **suffixes**, **infixes**, and **circumfixes**.
  - Suffix: *eat-s*
  - Prefix: *un-buckle*
  - Circumfix: *ge-sag-t* (said) *sagen* (to say) (in German)
  - Infix: *hingi* (borrow) *h**um**ingi* (the agent of an action))in Philippine language Tagalog)

3.1 Survey of (Mostly) English Morphology

- Prefixes and suffixes are often called **concatenative morphology.**
- A number of languages have extensive **non-concatenative morphology**
  - The Tagalog infixation example
  - **Templatic morphology** or **root-and-pattern morphology**, common in Arabic, Hebrew, and other Semitic languages
- Two broad classes of ways to form words from morphemes:
  - **Inflection:** the combination of a word stem with a grammatical morpheme, usually resulting in a word of the same class as the original tem, and usually filling some syntactic function like agreement, and
  - **Derivation**: the combination of a word stem with a grammatical morpheme, usually resulting in a word of a *different* class, often with a meaning hard to predict exactly.

3.1 Survey of (Mostly) English Morphology
Inflectional Morphology

- In English, only nouns, verbs, and sometimes adjectives can be inflected, and the number of affixes is quite small.
- Inflections of nouns in English:
  - An affix marking **plural**,
    - `cat(-s), thrush(-es), ox (oxen), mouse (mice)`
    - `ibis(-es), waltz(-es), finch(-es), box(-es), butterfly(-lies)`
  - An affix marking **possessive**
    - `llama’s, children’s, llamas’, Euripides’ comedies`

3.1 Survey of (Mostly) English Morphology

Inflectional Morphology

- Verbal inflection is more complicated than nominal inflection.
  - English has three kinds of verbs:
    - **Main verbs**, *eat, sleep, impeach*
    - **Modal verbs**, *can will, should*
    - **Primary verbs**, *be, have, do*
  - Morphological forms of regular verbs

stem	walk	merge	try	map
-*s* form	walks	merges	tries	maps
-*ing* principle	walking	merging	trying	mapping
Past form or –*ed* participle	walked	merged	tried	mapped

  - These regular verbs and forms are significant in the morphology of English because of their *majority* and being *productive*.

3.1 Survey of (Mostly) English Morphology

Inflectional Morphology

- Morphological forms of irregular verbs

stem	eat	catch	cut
-*s* form	eats	catches	cuts
-*ing* principle	eating	catching	cutting
Past form	ate	caught	cut
–*ed* participle	eaten	caught	cut

3.1 Survey of (Mostly) English Morphology
Derivational Morphology

- **Nominalization** in English:
  - The formation of new nouns, often from verbs or adjectives

Suffix	Base Verb/Adjective	Derived Noun
-action	computerize (V)	computerization
-ee	appoint (V)	appointee
-er	kill (V)	killer
-ness	fuzzy (A)	fuzziness

  - Adjectives derived from nouns or verbs

Suffix	Base Noun/Verb	Derived Adjective
-al	computation (N)	computational
-able	embrace (V)	embraceable
-less	clue (A)	clueless

3.1 Survey of (Mostly) English Morphology

Derivational Morphology

- Derivation in English is more complex than inflection because
  - Generally less productive
    - A nominalizing affix like *–ation* can not be added to absolutely every verb. *eatation*(*)
  - There are subtle and complex meaning differences among nominalizing suffixes. For example, *sincerity* has a subtle difference in meaning from *sincereness*.

3.2 Morphological Processes in Mandarin

- Reduplication
  - 請你嚐這個菜
  - 請你嚐嚐這個菜
- Reduplcation

3.2 Finite-State Morphological Parsing

- Parsing English morphology

Input	Morphological parsed output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(caught +V +PAST-PART) or (catch +V +PAST)

Stems and morphological features

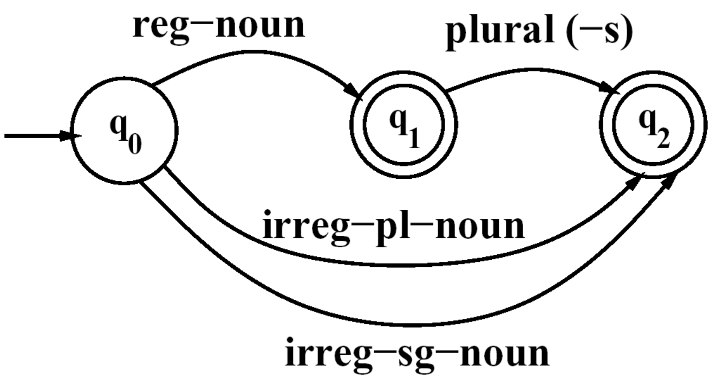
3.2 Finite-State Morphological Parsing

- We need at least the following to build a morphological parser:
   1. **Lexicon**: the list of stems and affixes, together with basic information about them (Noun stem or Verb stem, etc.)
   2. **Morphotactics**: the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. E.g., the rule that English plural morpheme follows the noun rather than preceding it.
   3. **Orthographic rules**: these **spelling rules** are used to model the changes that occur in a word, usually when two morphemes combine (e.g., the *y→ie* spelling rule changes *city* + -*s* to *cities*).

3.2 Finite-State Morphological Parsing

The Lexicon and Morphotactics

- A lexicon is a repository for words.
  - The simplest one would consist of an explicit list of every word of the language. ***Incovenient or impossible!***
  - Computational lexicons are usually structured with
    - a list of each of the stems and
    - Affixes of the language together with a representation of morphotactics telling us how they can fit together.
  - The most common way of modeling morphotactics is the finite-state automaton.

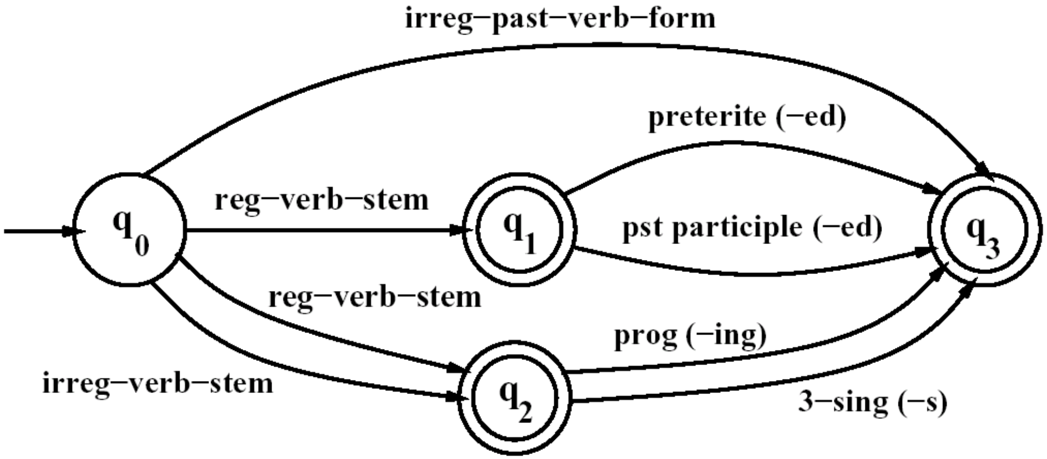


An FSA for English nominal inflection

Reg-noun	Irreg-pl-noun	Irreg-sg-noun	plural
fox fat fog fardvark	geese sheep Mice	goose sheep mouse	-s

3.2 Finite-State Morphological Parsing

The Lexicon and Morphotactics



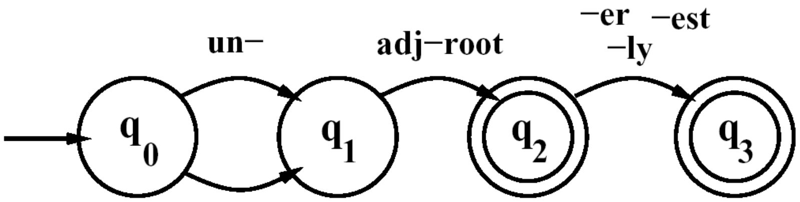
An FSA for English verbal inflection

Reg-verb-stem	Irreg-verb-stem	Irreg-past-verb	past	Past-part	Pres-part	3sg
walk fry talk impeach	cut speak sing sang spoken	caught ate eaten	-ed	-ed	-ing	-s

3.2 Finite-State Morphological Parsing

The Lexicon and Morphotactics

- English derivational morphology is more complex than English inflectional morphology, and so automata of modeling English derivation tends to be quite complex.
  - Some even based on CFG
- A small part of morphosyntactics of English adjectives

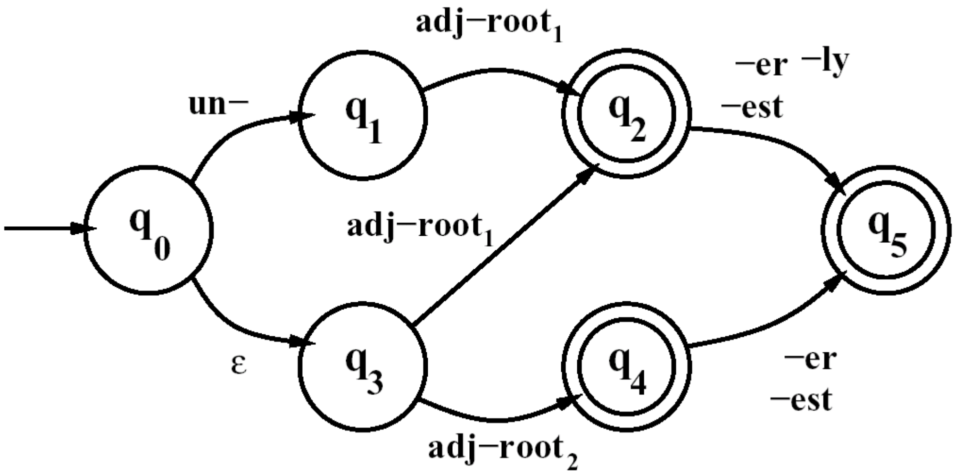


An FSA for a fragment of English adjective Morphology #1

big, bigger, biggest
cool, cooler, coolest, coolly
red, redder, reddest
clear, clearer, clearest, clearly, unclear, unclearly
happy, happier, happiest, happily
unhappy, unhappier, unhappiest, unhappily
real, unreal, really

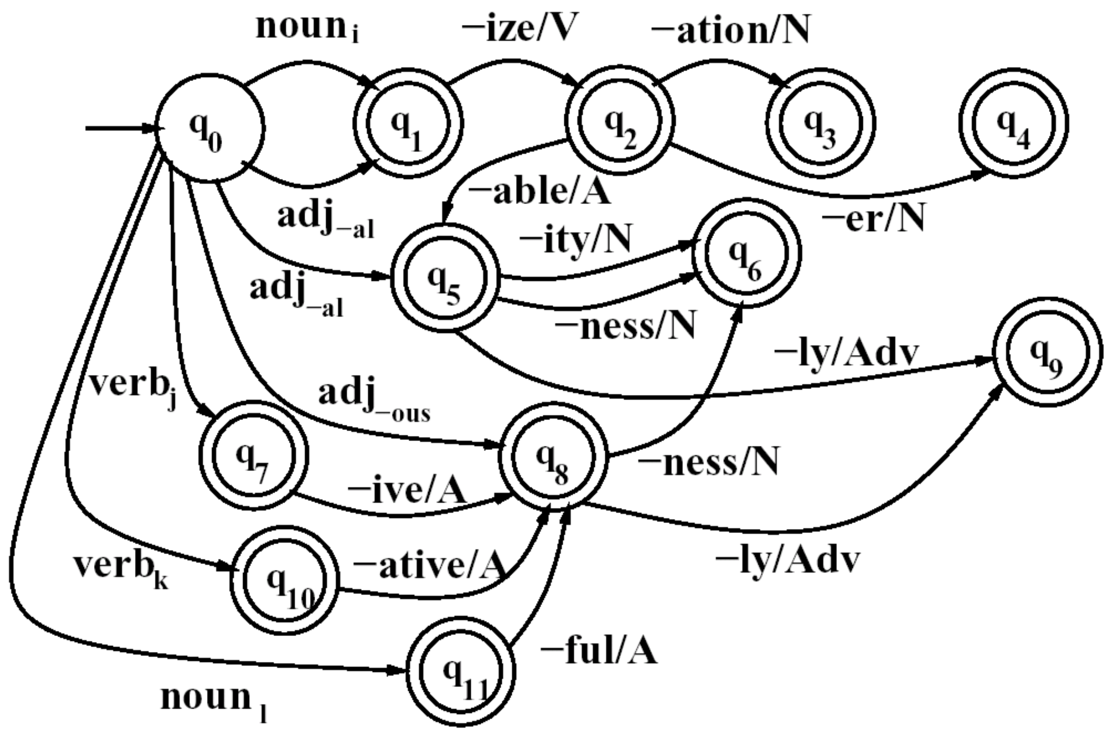
3.2 Finite-State Morphological Parsing

- The FSA#1 recognizes all the listed adjectives, and ungrammatical forms like *unbig*, *redly,* and *realest.*
- Thus #1 is revised to become #2.
- The complexity is expected from English derivation.



An FSA for a fragment of English adjective Morphology #2

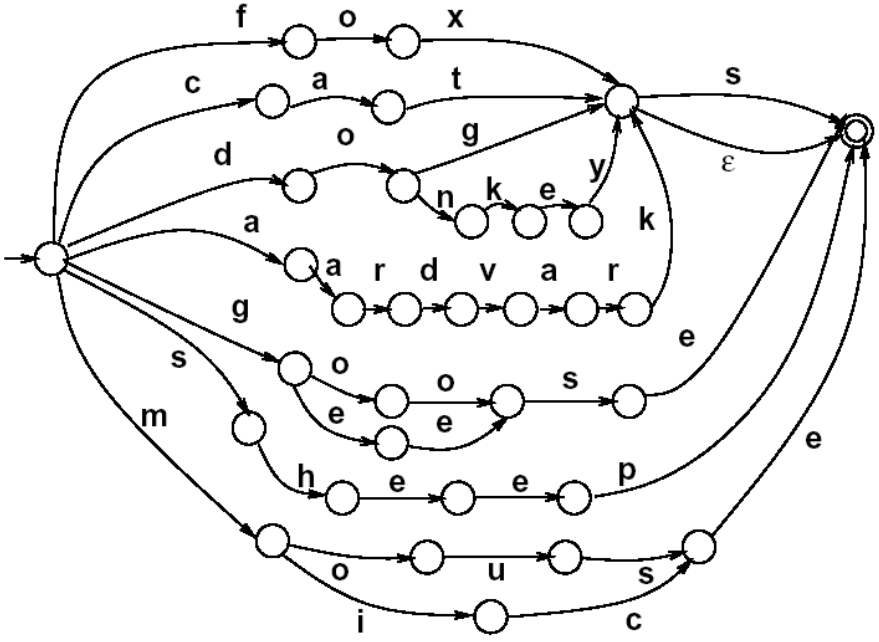
3.2 Finite-State Morphological Parsing



An FSA for another fragment of English derivational morphology

3.2 Finite-State Morphological Parsing

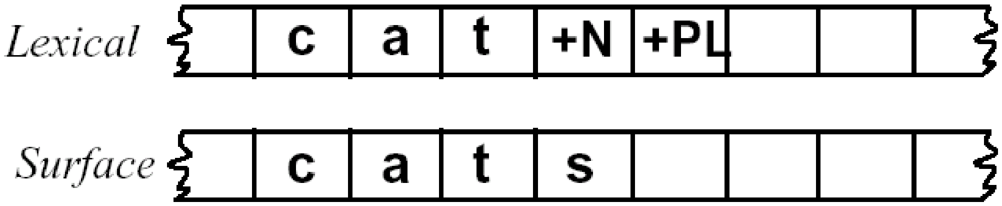
- We can now use these FSAs to solve the problem of **morphological recognition**:
  - Determining whether an input string of letters makes up a legitimate English word or not
  - We do this by taking the morphotactic FSAs, and plugging in each “sub-lexicon” into the FSA.
  - The resulting FSA can then be defined as the level of the individual letter.



3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- Given the input, for example, *cats*, we would like to produce `cat +N +PL`.
- Two-level morphology, by Koskenniemi (1983)
  - Representing a word as a correspondence between a **lexical level**
    - Representing a simple concatenation of morphemes making up a word, and
  - The **surface level**
    - Representing the actual spelling of the final word.
- Morphological parsing is implemented by building mapping rules that maps letter sequences like *cats* on the surface level into morpheme and features sequence like `cat +N +PL` on the lexical level.



3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- The automaton we use for performing the mapping between these two levels is the **finite-state transducer** or **FST**.
  - A transducer maps between one set of symbols and another;
  - An FST does this via a finite automaton.
- Thus an FST can be seen as a two-tape automaton which **recognizes** or **generates** ***pairs*** of strings.
- The FST has a more general function than an FSA:
  - An FSA defines a formal language
  - An FST defines a relation between sets of strings.
- Another view of an FST:
  - A machine reads one string and generates another.

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- **FST as recognizer:**
  - a transducer that takes a pair of strings as input and output *accept* if the string-pair is in the string-pair language, and a *reject* if it is not.
- **FST as generator:**
  - a machine that outputs pairs of strings of the language. Thus the output is a yes or no, and a pair of output strings.
- **FST as transducer:**
  - A machine that reads a string and outputs another string.
- **FST as set relater:**
  - A machine that computes relation between sets.

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- A formal definition of FST (based on the **Mealy machine** extension to a simple FSA):
  - Q: a finite set of N states $q_0, q_1, \ldots, q_N$
  - Σ: a finite alphabet of complex symbols. Each complex symbol is composed of an input-output pair $i : o$; one symbol I from an input alphabet I, and one symbol o from an output alphabet O, thus $\Sigma \subseteq I \times O$. I and O may each also include the epsilon symbol ε.
  - q_0: the start state
  - F: the set of final states, $F \subseteq Q$
  - $\delta(q, i{:}o)$: the transition function or transition matrix between states. Given a state $q \in Q$ and complex symbol $i{:}o \in \Sigma$, $\delta(q, i{:}o)$ returns a new state q' $\in Q$. δ is thus a relation from $Q \times \Sigma$ to Q.

3.2 Finite-State Morphological Parsing

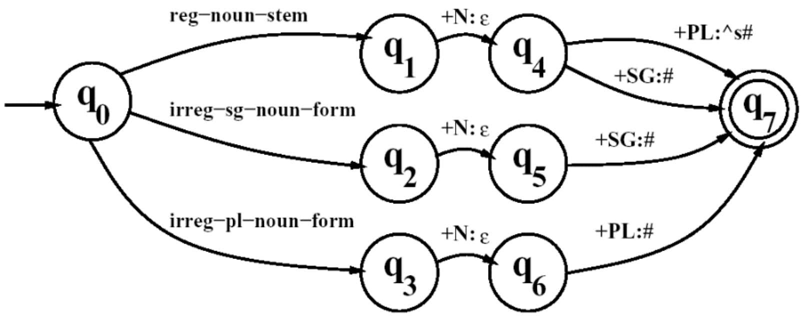
Morphological Parsing with FST

- FSAs are isomorphic to regular languages, FSTs are isomorphic to **regular relations.**
- Regular relations are sets of pairs of strings, a natural extension of the regular language, which are sets of strings.
- FSTs are closed under union, but generally they are not closed under difference, complementation, and intersection.
- Two useful closure properties of FSTs:
  - **Inversion:** If T maps from I to O, then the inverse of T, T^{-1} maps from O to I.
  - **Composition:** If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from I_2 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- Inversion is useful because it makes it easy to convert a FST-as-parser into an FST-as-generator.
- Composition is useful because it allows us to take two transducers than run in series and replace them with one complex transducer.
  - $T_1 \circ T_2(S) = T_2(T_1(S))$

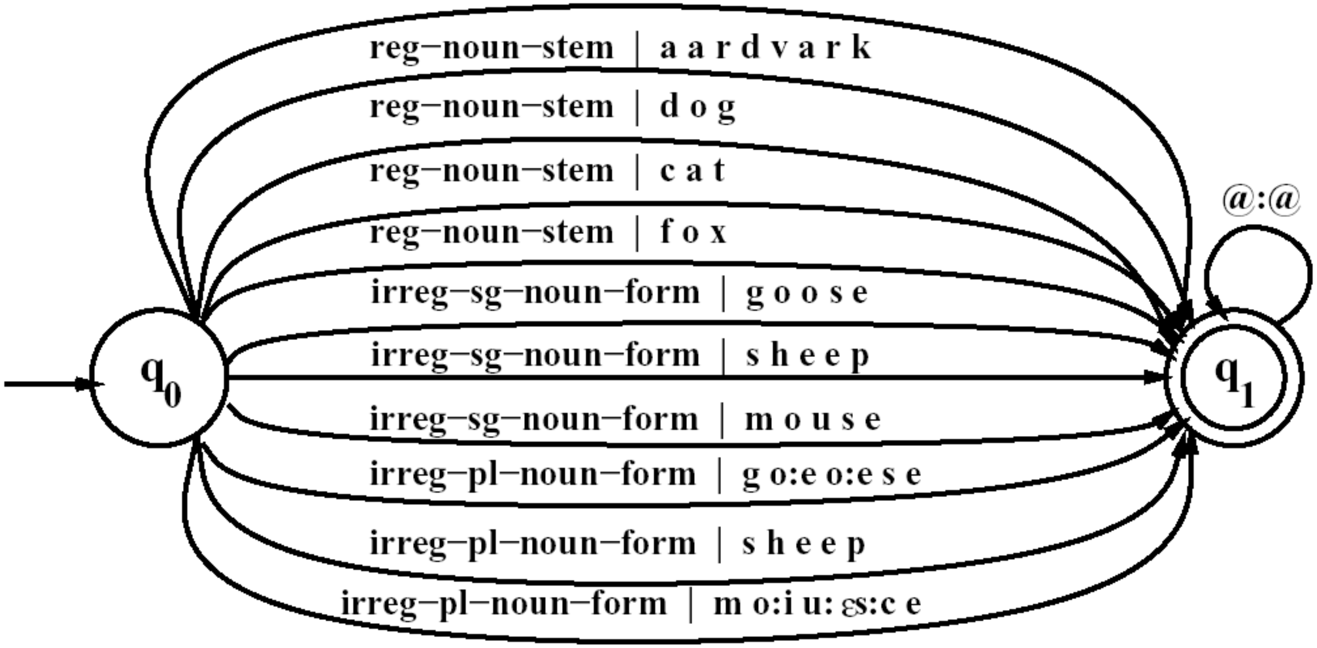


A transducer for English nominal number inflection T_{num}

Reg-noun	Irreg-pl-noun	Irreg-sg-noun
fox	g o:e o:e s e	goose
fat	sheep	sheep
fog	m o:i u:εs:c e	mouse
aardvark		

3.2 Finite-State Morphological Parsing

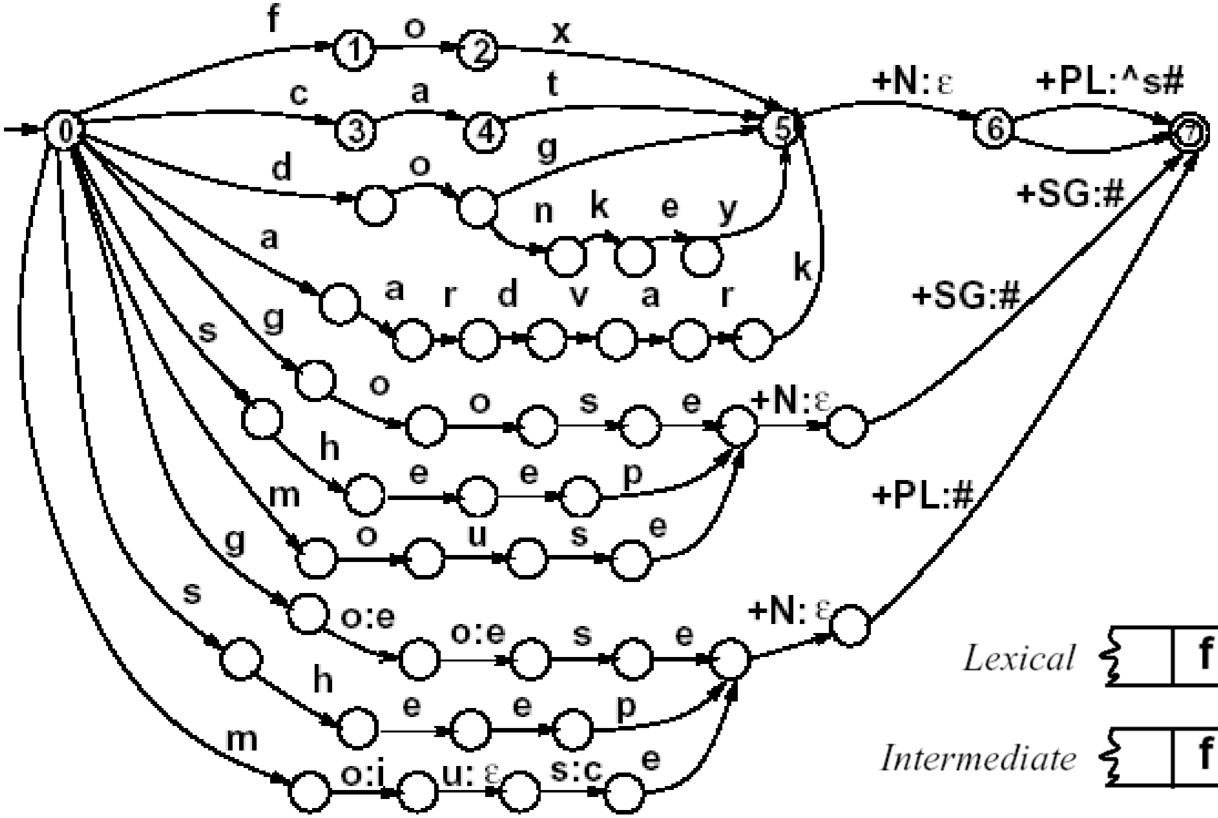
Morphological Parsing with FST


reg−noun−stem | a a r d v a r k
reg−noun−stem | d o g
reg−noun−stem | c a t
reg−noun−stem | f o x
irreg−sg−noun−form | g o o s e
irreg−sg−noun−form | s h e e p
irreg−sg−noun−form | m o u s e
irreg−pl−noun−form | g o:e o:e s e
irreg−pl−noun−form | s h e e p
irreg−pl−noun−form | m o:i u: ε s:c e
@:@
q0
q1

The transducer T_{stems}, which maps roots to their root-class

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST



^: morpheme boundary
#: word boundary

Lexical	f	o	x	+N	+PL			
Intermediate	f	o	x	^	s	#		

A fleshed-out English nominal inflection FST

$T_{lex} = T_{num} \circ T_{stems}$

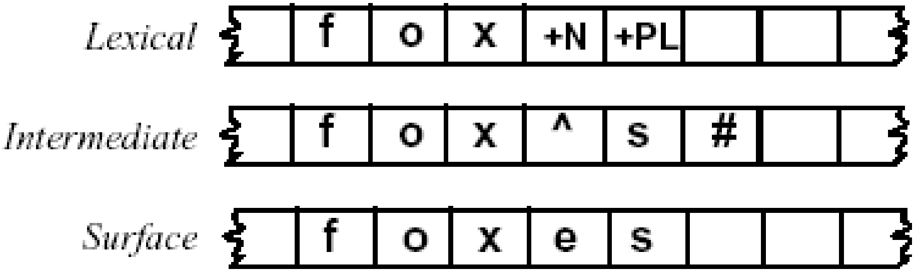
3.2 Finite-State Morphological Parsing

Orthographic Rules and FSTs

- **Spelling rules** (or **orthographic rules**)

Name	**Description of Rule**	**Example**
Consonant doubling	1-letter consonant doubled before *-ing/-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*, *-z*, *-x*, *-ch*, *-sh*, before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	Verb ending with *vowel* + *-c* add *-k*	panic/panicked

  - These spelling changes can be thought as taking as input a simple concatenation of morphemes and producing as output a slightly-modified concatenation of morphemes.



Lexical		f	o	x	+N	+PL			
Intermediate		f	o	x	^	s	#		
Surface		f	o	x	e	s			

3.2 Finite-State Morphological Parsing

Orthographic Rules and FSTs

- “insert an *e* on the surface tape just when the lexical tape has a morpheme ending in *x* (or *z*, etc) and the next morphemes is -*s*”

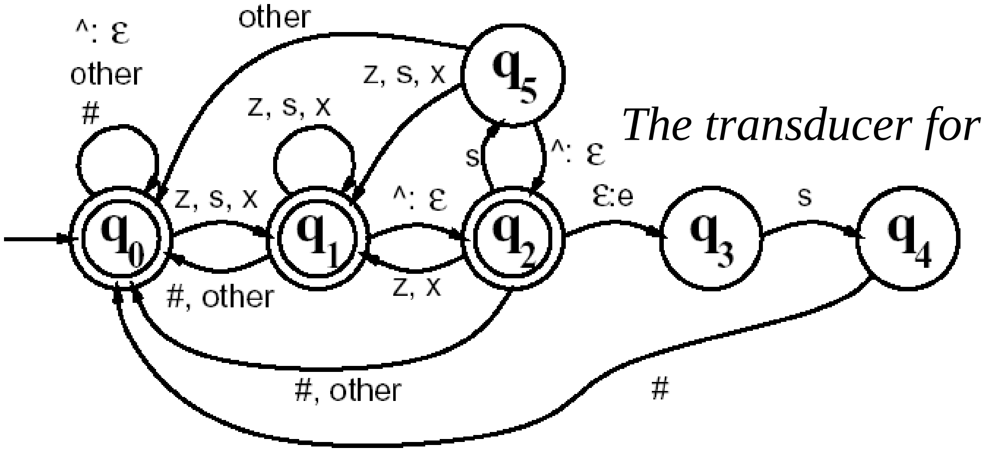
$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{array}{c} x \\ s \\ z \end{array} \right\} \hat{}\ \underline{\ \ }s\#$$

- “rewrite *a* and *b* when it occurs between *c* and *d*”

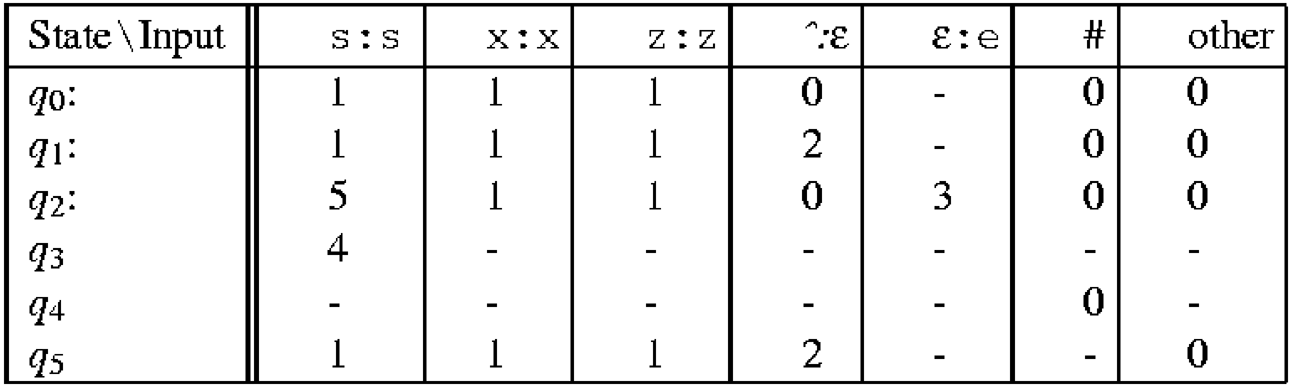
$$a \rightarrow b \ / \ c \ \underline{\ \ }d$$

3.2 Finite-State Morphological Parsing

Orthographic Rules and FSTs

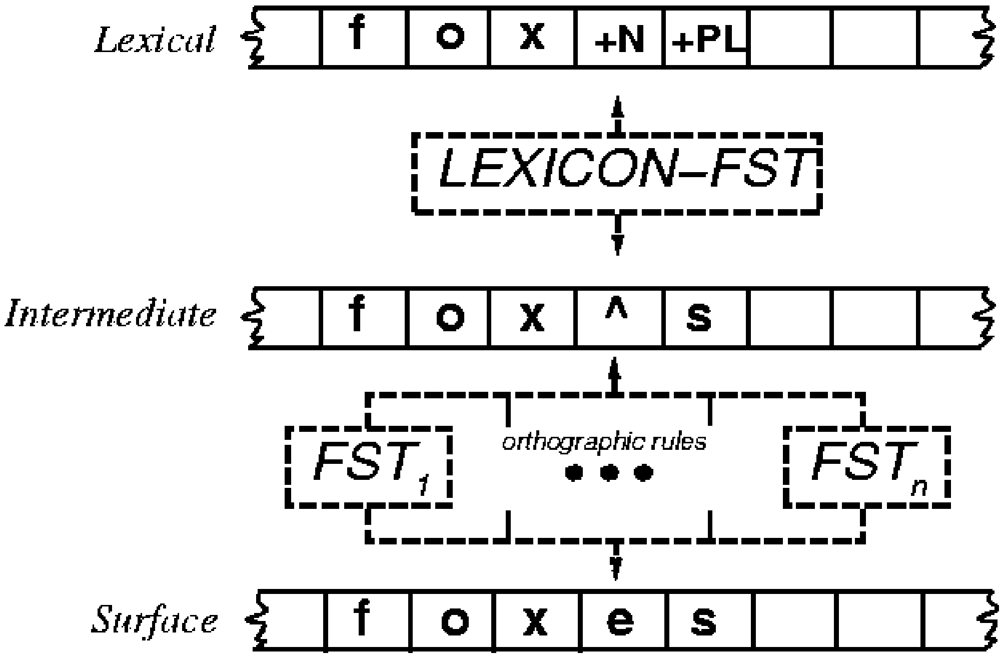


The transducer for the E-insertion rule

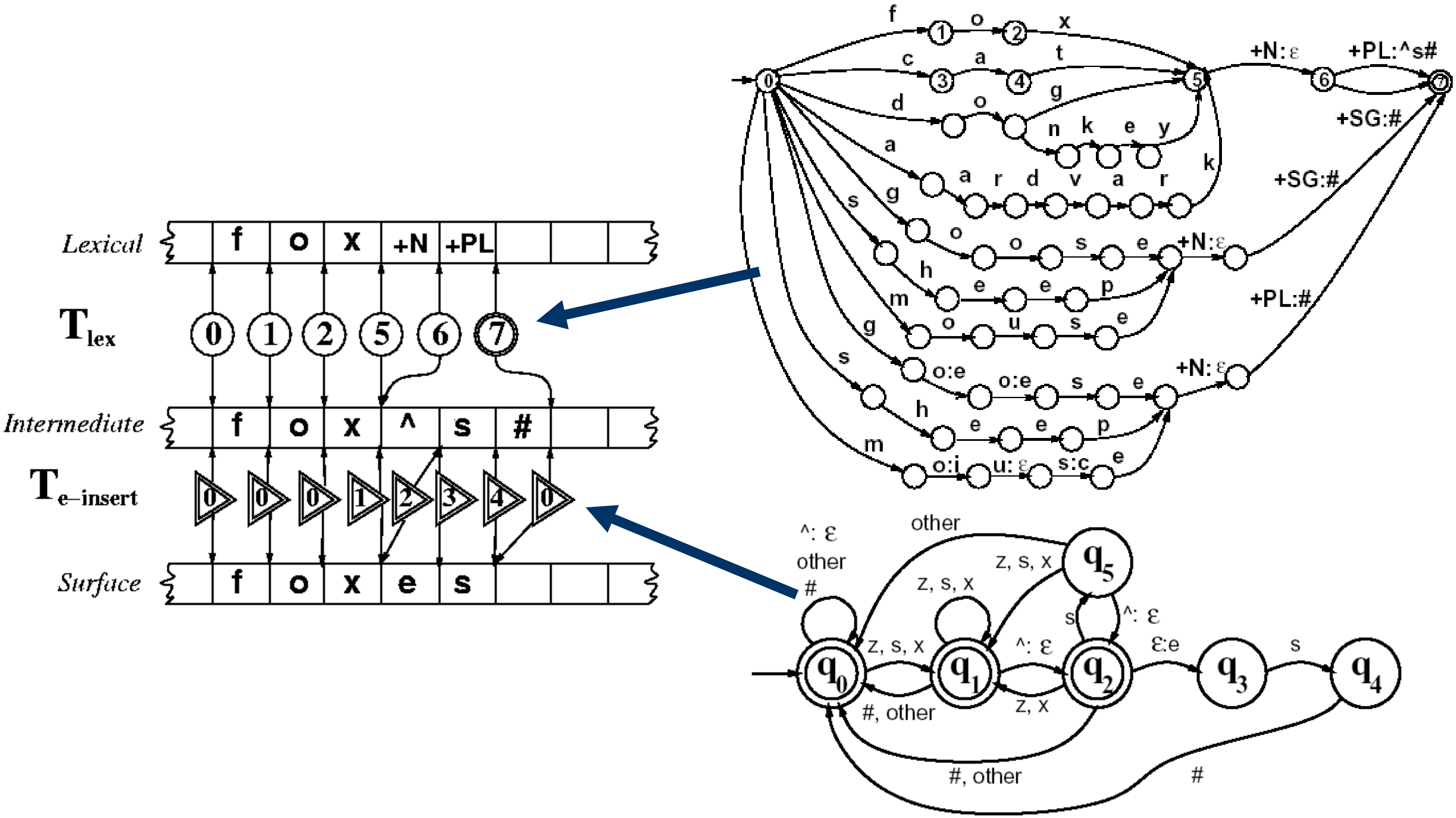


State\Input	s:s	x:x	z:z	^:ε	ε:e	#	other
q_0:	1	1	1	0	-	0	0
q_1:	1	1	1	2	-	0	0
q_2:	5	1	1	0	3	0	0
q_3	4	-	-	-	-	-	-
q_4	-	-	-	-	-	0	-
q_5	1	1	1	2	-	-	0

3.3 Combining FST Lexicon and Rules



3.3 Combining FST Lexicon and Rules



3.3 Combining FST Lexicon and Rules

- The power of FSTs is that the exact same cascade with the same state sequences is used
  - when machine is generating the surface form from the lexical tape, or
  - When it is parsing the lexical tape from the surface tape.
- Parsing can be slightly more complicated than generation, because of the problem of **ambiguity.**
  - For example, *foxes* could be `fox +V +3SG` as well as `fox +N +PL`

3.4 Lexicon-Free FSTs: the Porter Stemmer

- Information retrieval
- One of the mostly widely used **stemmming** algorithms is the simple and efficient Porter (1980) algorithm, which is based on a series of simple cascaded rewrite rules.
  - ATIONAL → ATE (e.g., relational → relate)
  - ING → εif stem contains vowel (e.g., motoring → motor)
- Problem:
  - Not perfect: error of commision, omission
- Experiments have been made
  - Some improvement with smaller documents
  - Any improvement is quite small

Chapter 8. Word Classes and Part-of-Speech Tagging

From: Chapter 8 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- **Part of speech**:
  - Noun, verb, pronoun, preposition, adverb, conjunction, particle, and article
- Recent lists of **POS** (also know as **word classes**, **morphological class**, or **lexical tags**) have much larger numbers of word classes.
  - 45 for Penn Treebank
  - 87 for the Brown corpus, and
  - 146 for the C7 tagset
- The significance of the POS for language processing is that it gives a significant amount of information about the word and its neighbors.
- POS can be used in stemming for IR, since
  - Knowing a word's POS can help tell us which morphological affixes it can take.
  - They can help an IR application by helping select out nouns or other important words from a document.

8.1 English Word Classes

- Give a more complete definition of the classes of POS.
  - Traditionally, the definition of POS has been based on morphological and syntactic function.
  - While, it has tendencies toward semantic coherence (e.g., nouns describe “people, places, or things and adjectives describe properties), this is not necessarily the case.
- Two broad subcategories of POS:
  1. **Closed class**
  2. **Open class**

8.1 English Word Classes

1. **Closed class**
   - Having relatively fixed membership, e.g., prepositions
   - **Function words:**
     - Grammatical words like *of*, *and*, or *you*, which tend to be very short, occur frequently, and play an important role in grammar.
2. **Open class**
   - Four major open classes occurring in the languages of the world: **nouns**, **verbs**, **adjectives**, and **adverbs.**
     - Many languages have no adjectives, e.g., the native American language Lakhota, and Chinese

8.1 English Word Classes
Open Class: Noun

- **Noun**
  - The name given to the lexical class in which the words for most people, places, or things occur
  - Since lexical classes like noun are defined functionally (morphological and syntactically) rather than semantically,
    - some words for people, places, or things may not be nouns, and conversely
    - some nouns may not be words for people, places, or things.
  - Thus, nouns include
    - Concrete terms, like *ship*, and *chair*,
    - Abstractions like *bandwidth* and *relationship*, and
    - Verb-like terms like *pacing*
  - Noun in English
    - Things to occur with determiners (*a goat*, *its bandwidth*, *Plato's Republic*),
    - To take possessives (*IBM's annual revenue*), and
    - To occur in the plural form (*goats*, *abaci*)

8.1 English Word Classes

Open Class: Noun

- Nouns are traditionally grouped into **proper nouns** and **common nouns.**
  - **Proper nouns:**
    - *Regina*, *Colorado*, and *IBM*
    - Not preceded by articles, e.g., *the book is upstairs*, but *Regina is upstairs*.
  - **Common nouns**
    - **Count nouns:**
      - Allow grammatical enumeration, i.e., both singular and plural (*goat/goats*), and can be counted (*one goat*/ *two goats*)
    - **Mass nouns:**
      - Something is conceptualized as a homogeneous group, *snow*, *salt*, and *communism*.
      - Appear without articles where singular nouns cannot (*Snow is white* but not **Goal is white*)

8.1 English Word Classes

Open Class: Verb

- **Verbs**
  - Most of the words referring to actions and processes including main verbs like *draw*, *provide*, *differ*, and *go.*
  - A number of morphological forms: non-3rd-person-sg (*eat*), 3rd-person-sg(*eats*), progressive (*eating*), past participle (*eaten*)
  - A subclass: **auxiliaries** (discussed in closed class)

8.1 English Word Classes

Open Class: Adjectives

- **Adjectives**
  - Terms describing properties or qualities
  - Most languages have adjectives for the concepts of color (*white*, *black*), age (*old*, *young*), and value (*good*, *bad*), but
  - There are languages without adjectives, e.g., Chinese.

8.1 English Word Classes

Open Class: Adverbs

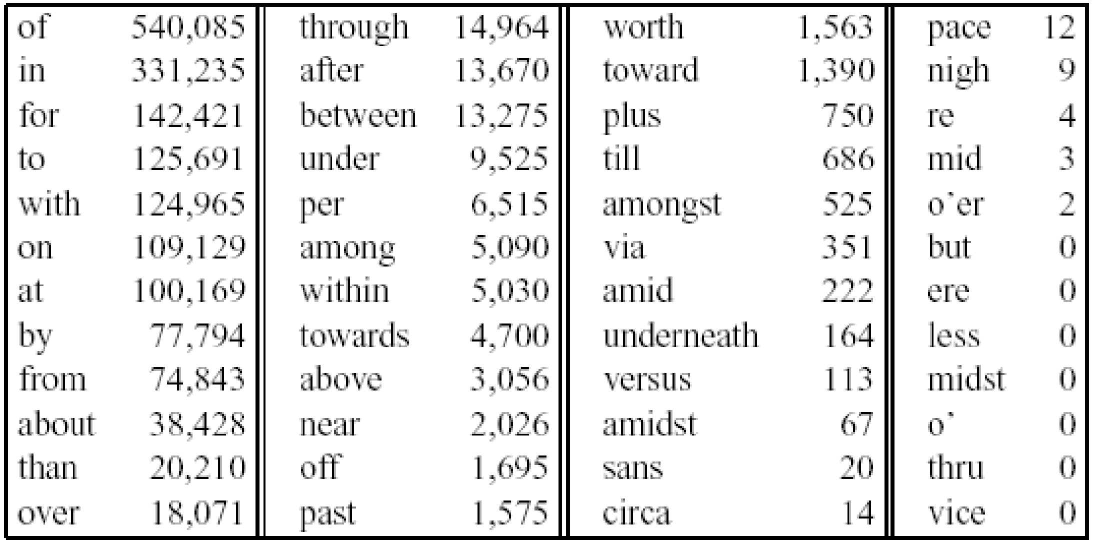
- **Adverbs**
  - Words viewed as modifying something (often verbs)
    - **Directional (or locative) adverbs:** specify the direction or location of some action, *hoe, here, downhill*
    - **Degree adverbs:** specify the extent of some action, process, or property, *extremely, very, somewhat*
    - **Manner adverb:** describe the manner of some action or process, *slowly, slinkily, delicately*
    - **Temporal adverbs:** describe the time that some action or event took place, *yesterday, Monday*

8.1 English Word Classes
Open Classes

- Some important closed classes in English
  - **Prepositions:** on, under, over, near, by, at, from, to, with
  - **Determiners:** a, an, the
  - **Pronouns:** she, who, I, others
  - **Conjunctions:** and, but, or, as, if, when
  - **Auxiliary verbs:** can, may, should, are
  - **Particles:** up, down, on, off, in, out, at, by
  - **Numerals:** one, two, three, first, second, third

8.1 English Word Classes
Open Classes: Prepositions

- **Prepositions** occur before nouns, semantically they are relational
  - Indicating spatial or temporal relations, whether literal (*on it*, *before then*, *by the house*) or metaphorical (*on time, with gusto*, *beside herself*)
  - Other relations as well



of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

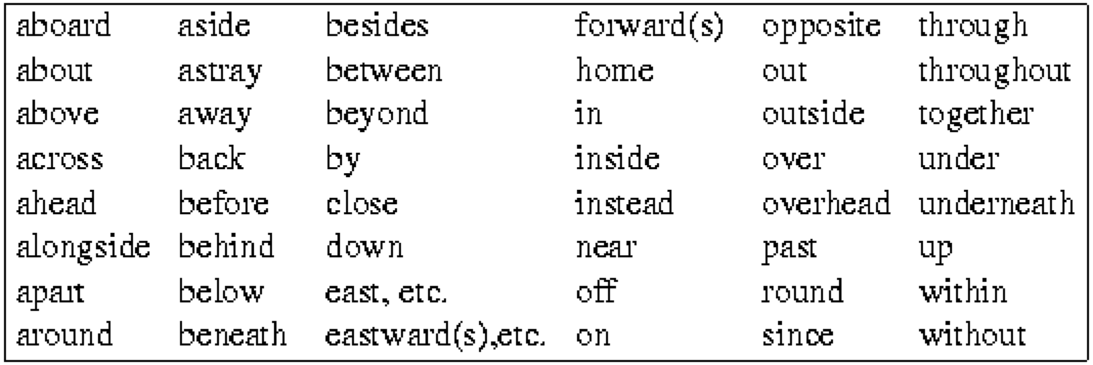
Preposition (and particles) of English from CELEX

8.1 English Word Classes
Open Classes: Particles

- A **particle** is a word that resembles a preposition or an adverb, and that often combines with a verb to form a larger unit call a **phrasal verb**

  So I *went on* for some days cutting and hewing timber …

  Moral reform is the effort to *throw off* sleep …



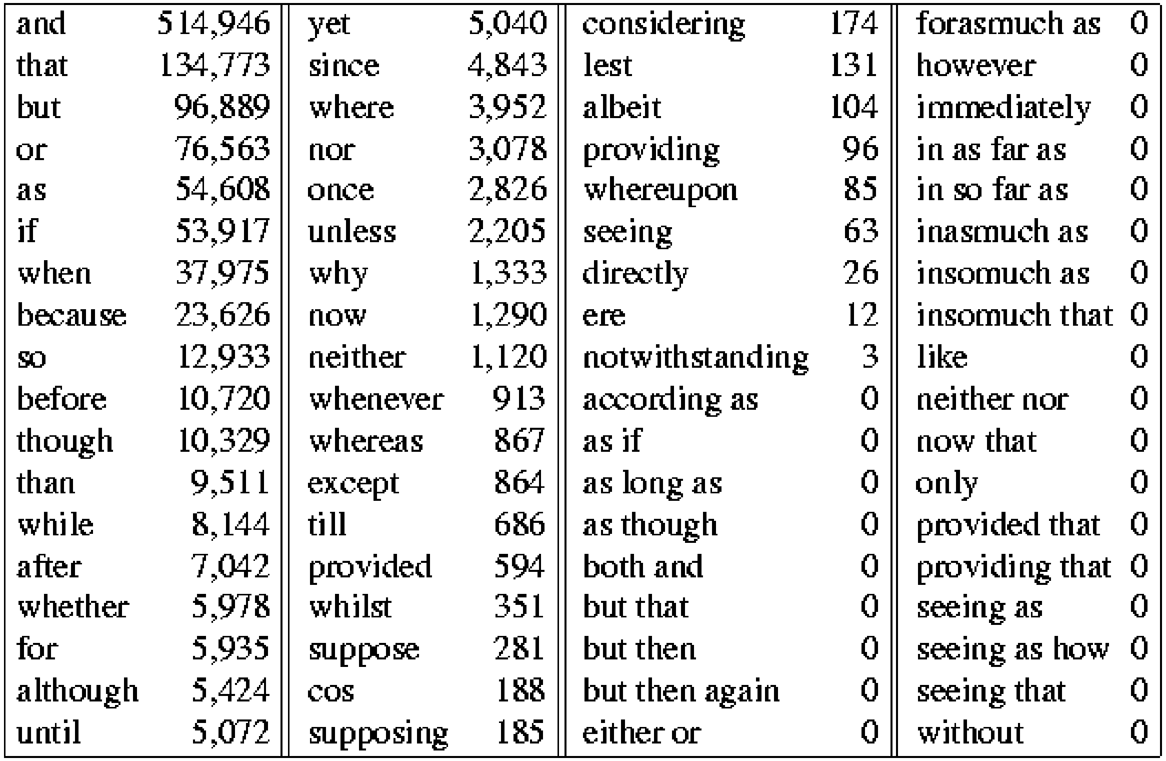
aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

English single-word particles from Quirk, et al (1985)

8.1 English Word Classes

Open Classes: Particles and Conjunctions

- English has three: *a*, *an*, and *the*
  - Articles begin a noun phrase.
  - Articles are frequent in English.
- **Conjunctions** are used to join two phrases, clauses, or sentences.
  - *and*, *or*, or, *but*
  - Subordinating conjunctions are used when one of the elements is of some sort of embedded status. *I thought* ***that*** *you might like some milk*…**complementizer**

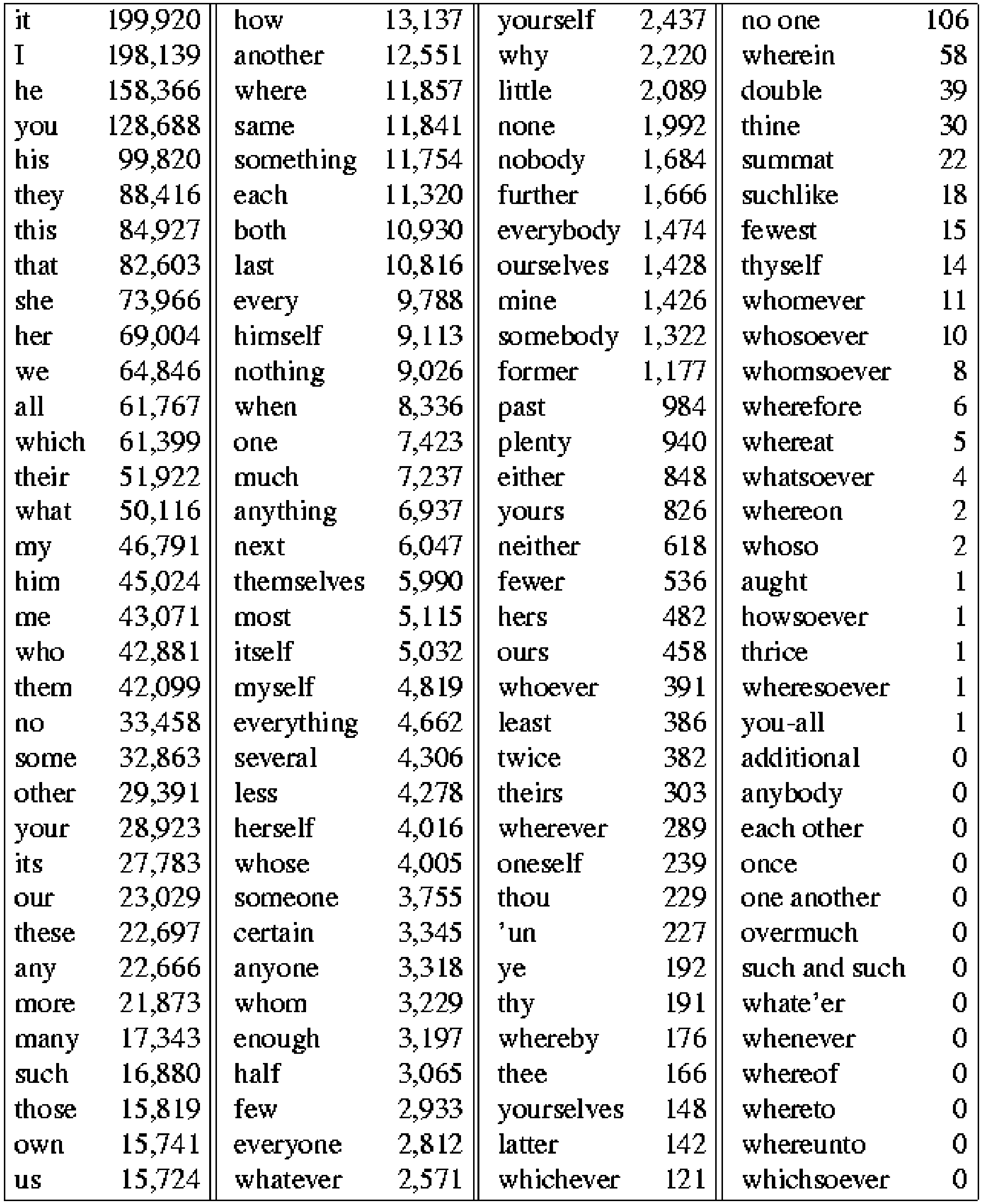


and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

Coordinating and subordinating conjunctions of English
From the CELEX on-line dictionary.

8.1 English Word Classes
Open Classes: Pronouns

- **Pronouns** act as a kind of shorthand for referring to some noun phrase or entity or event.
  - **Personal pronouns:** persons or entities (*you, she, I, it, me, etc*)
  - **Possessive pronouns:** forms of personal pronouns indicating actual possession or just an abstract relation between the person and some objects.
  - **Wh-pronouns:** used in certain question forms, or may act as complementizer.

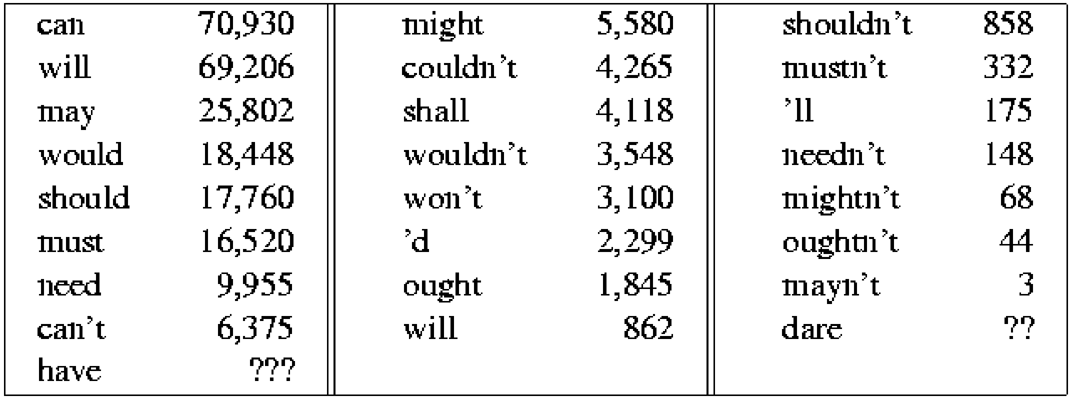


it	199,920	how	13,137	yourself	2,437	no one	106
I	198,139	another	12,551	why	2,220	wherein	58
he	158,366	where	11,857	little	2,089	double	39
you	128,688	same	11,841	none	1,992	thine	30
his	99,820	something	11,754	nobody	1,684	summat	22
they	88,416	each	11,320	further	1,666	suchlike	18
this	84,927	both	10,930	everybody	1,474	fewest	15
that	82,603	last	10,816	ourselves	1,428	thyself	14
she	73,966	every	9,788	mine	1,426	whomever	11
her	69,004	himself	9,113	somebody	1,322	whosoever	10
we	64,846	nothing	9,026	former	1,177	whomsoever	8
all	61,767	when	8,336	past	984	wherefore	6
which	61,399	one	7,423	plenty	940	whereat	5
their	51,922	much	7,237	either	848	whatsoever	4
what	50,116	anything	6,937	yours	826	whereon	2
my	46,791	next	6,047	neither	618	whoso	2
him	45,024	themselves	5,990	fewer	536	aught	1
me	43,071	most	5,115	hers	482	howsoever	1
who	42,881	itself	5,032	ours	458	thrice	1
them	42,099	myself	4,819	whoever	391	wheresoever	1
no	33,458	everything	4,662	least	386	you-all	1
some	32,863	several	4,306	twice	382	additional	0
other	29,391	less	4,278	theirs	303	anybody	0
your	28,923	herself	4,016	wherever	289	each other	0
its	27,783	whose	4,005	oneself	239	once	0
our	23,029	someone	3,755	thou	229	one another	0
these	22,697	certain	3,345	’un	227	overmuch	0
any	22,666	anyone	3,318	ye	192	such and such	0
more	21,873	whom	3,229	thy	191	whate’er	0
many	17,343	enough	3,197	whereby	176	whenever	0
such	16,880	half	3,065	thee	166	whereof	0
those	15,819	few	2,933	yourselves	148	whereto	0
own	15,741	everyone	2,812	latter	142	whereunto	0
us	15,724	whatever	2,571	whichever	121	whichsoever	0

Pronouns of English from the CELEX on-line dictionary.

8.1 English Word Classes
Open Classes: Auxiliary Verbs

- **Auxiliary verbs:** mark certain semantic feature of a main verb, including
  - whether an action takes place in the present, past or future (tense),
  - whether it is completed (aspect),
  - whether it is negated (polarity), and
  - whether an action is necessary, possible, suggested, desired, etc (mood).
  - Including **copula** verb *be*, the two verbs *do* and *have* along with their inflection forms, as well as a class of **modal verbs.**



can	70,930	might	5,580	shouldn't	858
will	69,206	couldn't	4,265	mustn't	332
may	25,802	shall	4,118	'll	175
would	18,448	wouldn't	3,548	needn't	148
should	17,760	won't	3,100	mightn't	68
must	16,520	'd	2,299	oughtn't	44
need	9,955	ought	1,845	mayn't	3
can't	6,375	will	862	dare	??
have	???				

English modal verbs from the CELEX on-line dictionary.

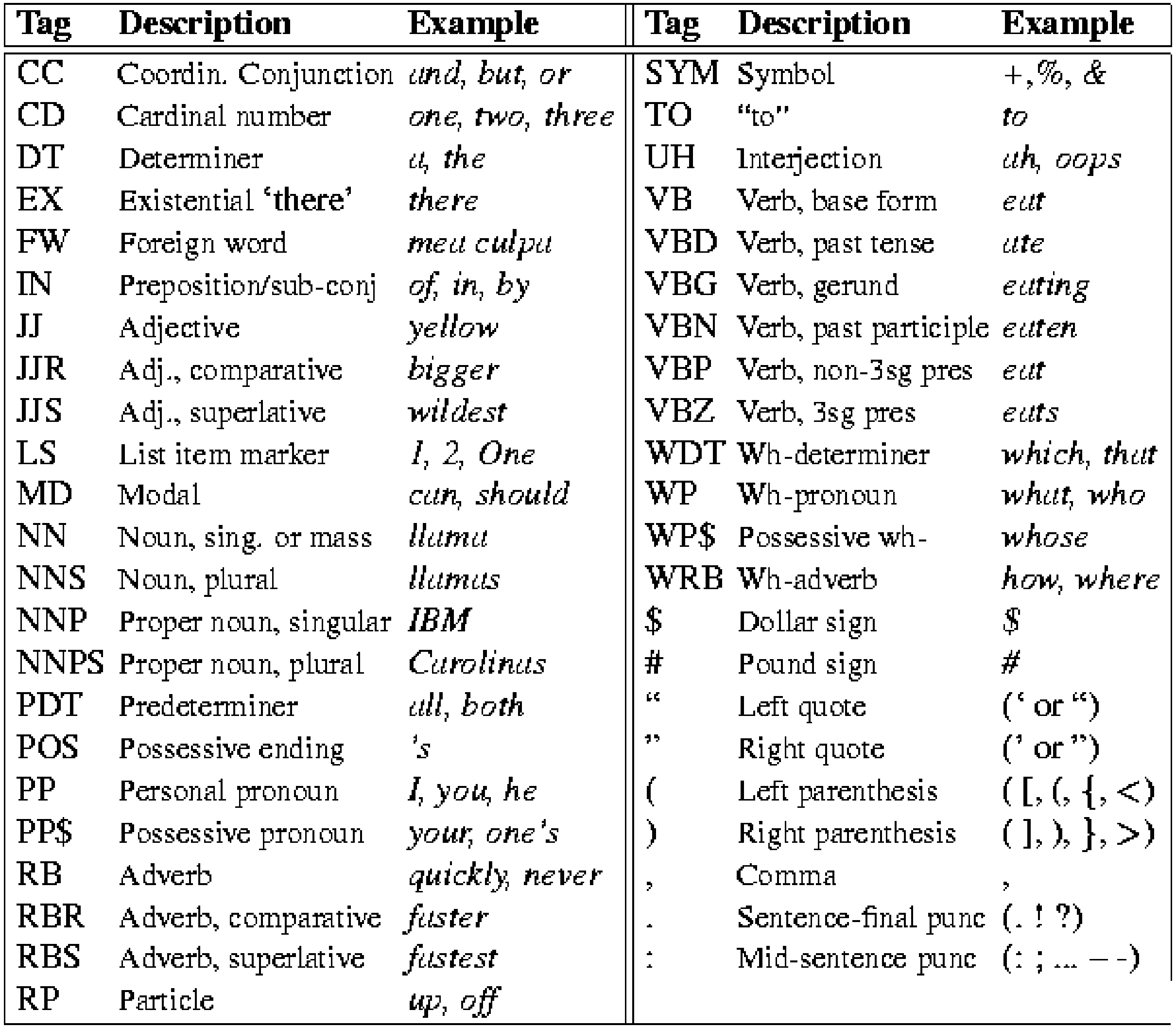
8.1 English Word Classes

Open Classes: Others

- **Interjections:** *oh, ah, hey, man, alas*
- **Negatives:** *no, not*
- **Politeness markers:** *please, thank you*
- **Greetings:** *hello, goodbye*
- Existential **there:** *there are two on the table*

8.2 Tagsets for English

- There are a small number of popular tagsets for English, many of which evolved from the 87-tag tagset used for the Brown corpus.
  - Three commonly used
    - **The small 45-tag Penn Treebank tagset**
    - The medium-sized 61 tag C5 tageset used by the Lancaster UCREL project's CLAWS tagger to tag the British National Corpus, and
    - The larger 146-tag C7 tagset



Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Interjection	*uh, oops*
EX	Existential ‘there’	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*’s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one’s*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Penn Treebank POS tags

8.2 Tagsets for English

The/DT grand/JJ jury/NN commented/VBD on/IN a /DT number/NN of/IN other/JJ topics/NNS ./.

- Certain syntactic distinctions were not marked in the Penn Treebank tagset because
  - Treebank sentences were parsed, not merely tagged, and
  - So some syntactic information is represented in the phrase structure.
- For example, prepositions and subordinating conjunctions were combined into the single tag *IN*, since the tree-structure of the sentence disambiguated them.

8.3 Part-of-Speech Tagging

- POS tagging (tagging)
  - The process of assigning a POS or other lexical marker to each word in a corpus.
  - Also applied to punctuation marks
  - Thus, tagging for NL is the same process as **tokenization** for computer language, although tags for NL are much more ambiguous.
  - Taggers play an increasingly important role in speech recognition, NL parsing and IR

8.3 Part-of-Speech Tagging

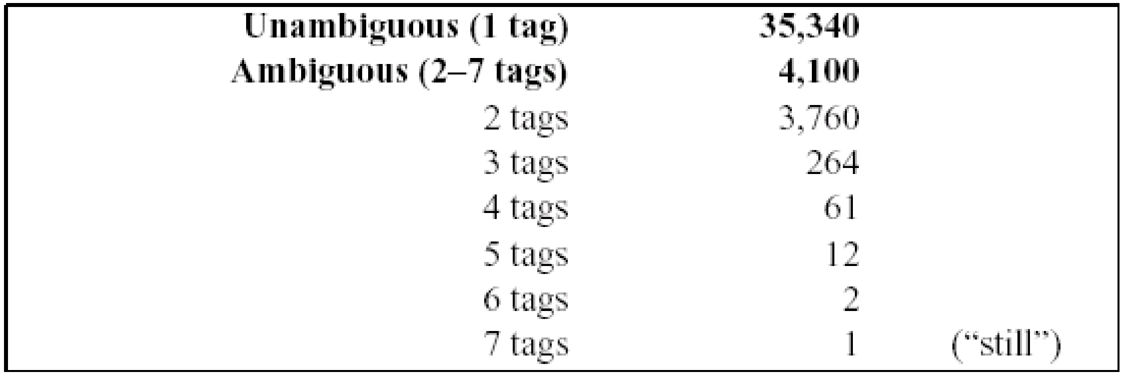
- The input to a tagging algorithm is a string of words and a specified **tagset** of the kind described previously.

```
VB    DT   NN  .
Book that flight .

VBZ  DT  NN   VB      NN   ?
Does that flight serve dinner ?
```

- Automatically assigning a tag to a word is not trivial
  - For example, *book* is ambiguous: it can be a verb or a noun
  - Similarly, *that* can be a determiner, or a complementizer
- The problem of POS-tagging is to resolve the ambiguities, choosing the proper tag for the context.

8.3 Part-of-Speech Tagging

- How hard is the tagging problem?



Unambiguous (1 tag)	**35,340**	
Ambiguous (2–7 tags)	**4,100**	
2 tags	3,760	
3 tags	264	
4 tags	61	
5 tags	12	
6 tags	2	
7 tags	1	(“still”)

The number of word types in Brown corpus by degree of ambiguity.

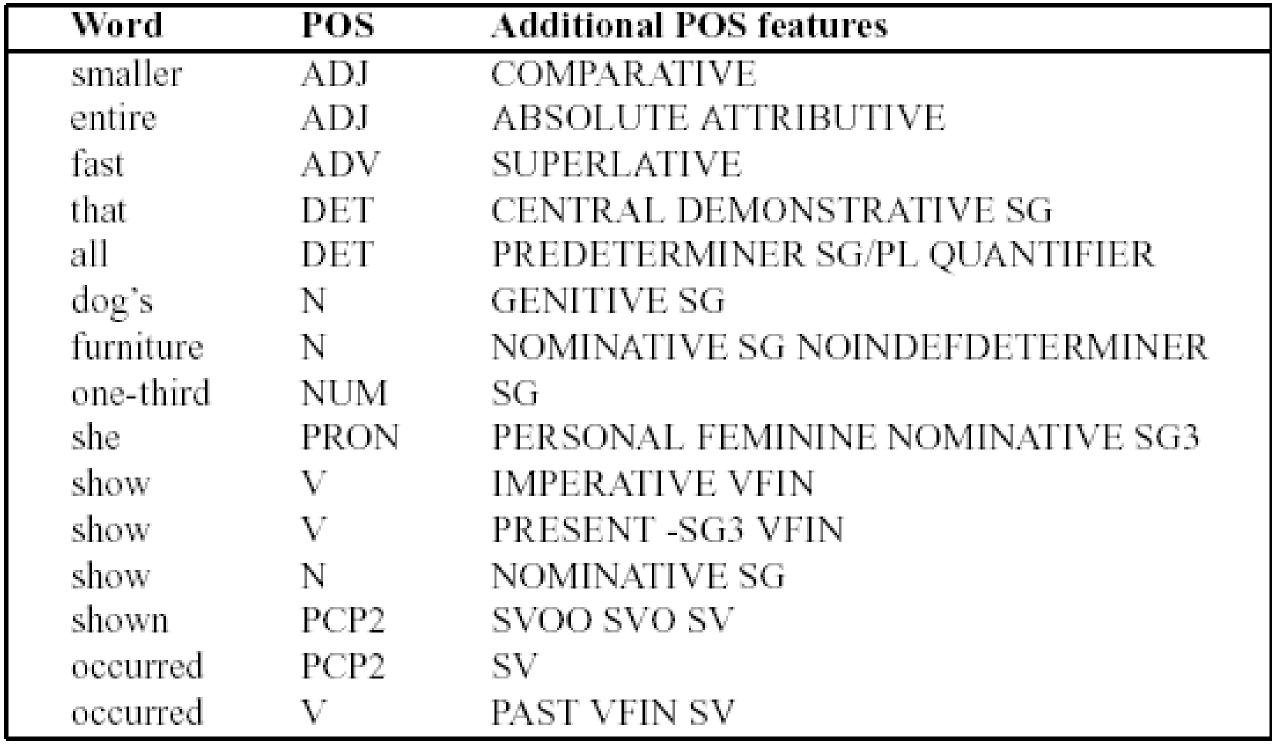
- Many of the 40% ambiguous tokens are easy to disambiguate, because
  - The various tags associated with a word are not equally likely.

8.3 Part-of-Speech Tagging

- Many tagging algorithms fall into two classes:
  - **Rule-based** taggers
    - Involve a large database of hand-written disambiguation rule specifying, for example, that *an ambiguous word is a noun rather than a verb if it follows a determiner*.
  - **Stochastic** taggers
    - Resolve tagging ambiguities by using a training corpus to *count the probability of a given word having a given tag in a given context*.
- The **Brill tagger**, called the **transformation-based tagger,** shares features of both tagging architecture.

8.4 Rule-Based Part-of-Speech Tagging

- The earliest algorithms for automatically assigning POS were based on a two-stage architecture
  - First, use a dictionary to assign each word a list of potential POS.
  - Second, use large lists of hand-written disambiguation rules to winnow down this list to a single POS for each word
- The **ENGTWOL** tagger (1995) is based on the same two stage architecture, with much more sophisticated lexicon and disambiguation rules than before.
  - Lexicon:
    - 56000 entries
    - A word with multiple POS is counted as separate entries



Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

Sample lexical entries from the ENGTWOL lexicon.

8.4 Rule-Based Part-of-Speech Tagging

- In the first stage of tagger,
  - each word is run through the two-level lexicon transducer and
  - the entries for all possible POS are returned.
- A set of about 1,100 constraints are then applied to the input sentences to rule out incorrect POS.

```
Pavlov      PALOV N NOM SG PROPER
had         HAVE V PAST VFIN SVO
            HAVE PCP2 SVO
shown       SHOW PCP2 SVOO SVO SV
that        ADV
             PRON DEM SG
             DET CENTRAL DEM SG
             CS
salivation  N NOM SG
…
```

8.4 Rule-Based Part-of-Speech Tagging

- A simplified version of the constraint:

```
ADVERBIAL-THAT RULE
Given input: “that”
if
   (+1 A/ADV/QUANT); /* if next word is adj, adverb, or quantifier */
   (+2 SENT-LIM);          /* and following which is a sentence boundary, */
   (NOT -1 SVOC/A);      /* and the previous word is not a verb like */
                                     /* ‘consider’ which allows adj as object complements */
then eliminate non-ADV tags
else eliminate ADV tags
```

8.5 HMM Part-of-Speech Tagging

- We are given a sentence, for example, like

  Secretariat is expected to **race** tomorrow.

  - What is the best sequence of tags which corresponds to this sequence of words?

- We want: out of all sequences of *n* tags t_1^n the single tag sequence such that $P(t_1^n \mid w_1^n)$ is highest.

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)$$

ˆ means "our estimate of the correct tag sequence".

- It is not clear how to make the equation operational
  - that is, for a given tag sequence t_1^n and word sequence w_1^n, we don't know how to directly compute $P(t_1^n \mid w_1^n)$.

8.5 HMM Part-of-Speech Tagging

$$\text{Q}\; P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \arg\max_{t_1^n} \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

But $P(w_1^n)$ doesn't change for each tag sequence.

$$\hat{t}_1^n = \arg\max_{t_1^n} \underbrace{P(w_1^n \mid t_1^n)}_{\textit{likelihood}}\underbrace{P(t_1^n)}_{\textit{prior}}$$

Still too hard to compute directly

8.5 HMM Part-of-Speech Tagging

Assumption 1: the probability of a word appearing is dependent only on its own part of speech tag:

$$P(w_1^n \mid t_1^n) \approx \prod_{i=1}^{n} P(w_i^n \mid t_i^n)$$

Assumption 2: the probability of a tag appearing is dependent only on the previous tag:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

8.5 HMM Part-of-Speech Tagging

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n) \approx \arg\max_{t_1^n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

- This equation contains two kinds of probabilities,
  - tag transition probabilities and
  - word likelihoods.
- The tag transition probabilities:

$$P(t_i \mid t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN \mid DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

$$P(w_i \mid t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

- The word likelihood probabilities:

$$P(is \mid VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

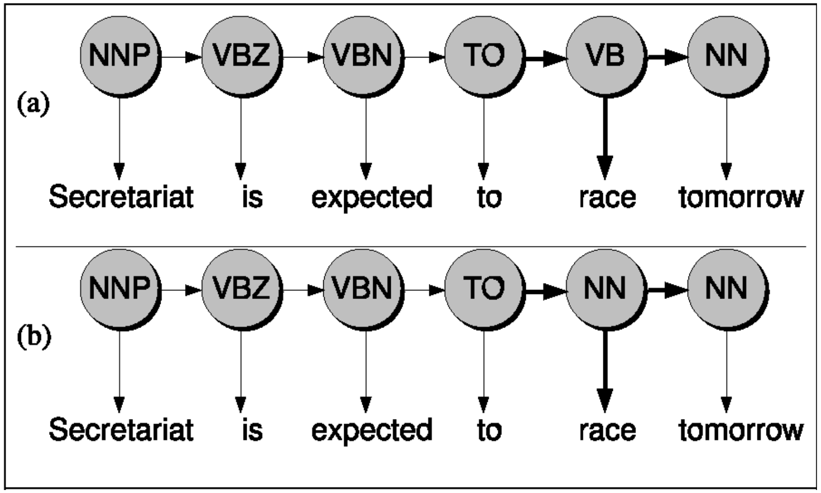
8.5 HMM Part-of-Speech Tagging

Computing the most-likely tag sequence: A motivating example

(8.36)Secretariat/NNP is/BEZ expected/VBN to/TO race/VB tomorrow/NR

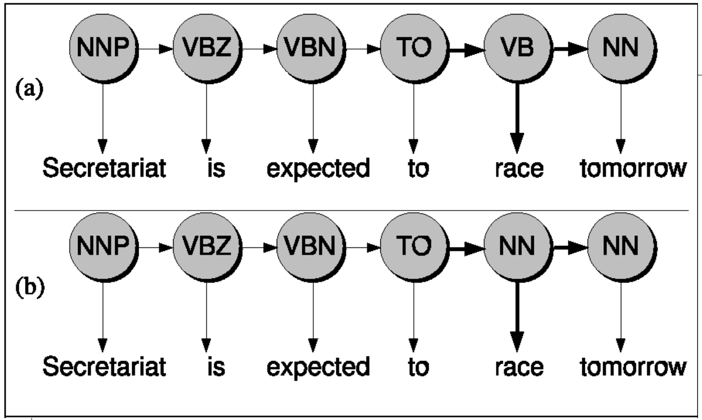
(8.37)People/NNS continue/VB to/TO inquire/VB the/AT reason/NN for/IN the/AT race/NN for/IN outer/JJ space/NN

- Let's look at how race can be correctly tagged as a VB instead of an NN in (8.36).



8.5 HMM Part-of-Speech Tagging

Computing the most-likely tag sequence: A motivating example



$$P(NN\,|\,TO) = .00047$$

$$P(VB\,|\,TO) = .83$$

$$P(race\,|\,NN) = .00057$$

$$P(race\,|\,VB) = .00012$$

$$P(NR\,|\,VB) = .0027$$

$$P(NR\,|\,NN) = .0012$$

$$P(VB\,|\,TO)P(NR\,|\,VB)P(race\,|\,VB) = .00000027$$

$$P(NN\,|\,TO)P(NR\,|\,NN)P(race\,|\,NN) = .00000000032$$

8.5 HMM Part-of-Speech Tagging

Formalizing Hidden Markov Model taggers

Chapter 9. Context-Free Grammars for English

From: Chapter 9 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- **Syntax:** the way words are arranged together
- Main ideas of syntax:
  - **Constituency**
    - Groups of words may behave as a single unit or phrase, called **constituent**, e.g., NP
    - CFG, a formalism allowing us to model the constituency facts
  - **Grammatical relations**
    - A formalization of ideas from traditional grammar about SUBJECT and OBJECT
  - **Subcategorization and dependencies**
    - Referring to certain kind of relations between words and phrases, e.g., the verb *want* can be followed by an infinite, as in *I want to fly to Detroit*.

Background

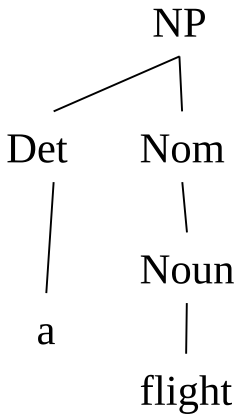
- All of the kinds of syntactic knowledge can be modeled by various kinds of CFG-based grammars.
- CFGs are thus backbone of many models of the syntax of NL.
  - Being integral to most models of NLU, of grammar checking, and more recently speech understanding
- They are powerful enough to express sophisticated relations among the words in a sentence, yet computationally tractable enough that efficient algorithms exists for parsing sentences with them. (Ch. 10)
- Also probability version of CFG (Ch. 12)
- Example sentences from the Air Traffic Information System (ATIS) domain

9.1 Constituency

- **NP:**
  - A sequence of words surrounding at least one noun, e.g.,
    - three parties from Brooklyn *arrive*
    - a high-class spot such as Mindy's *attracts*
    - the Broadway coppers *love*
    - They *sit*
    - Harry the Horse
    - the reason he comes into the Hot Box
- Evidences of constituency
  - The above NPs can all appear in similar syntactic environment, e.g., before, a verb.
  - **Preposed** or **postposed** constructions, e.g., the PP, *on September seventeenth*, can be placed in a number of different locations
    - On September seventeenth, I'd like to fly from Atlanta to Denver.
    - I'd like to fly on September seventeenth from Atlanta to Denver.
    - I'd like to fly from Atlanta to Denver On September seventeenth.

9.2 Context-Free Rules and Trees

- **CFG (or Phrase-Structure Grammar):**
  - The most commonly used mathematical system for modeling constituent structure in English and other NLs
  - Terminals and non-terminals
  - Derivation
  - Parse tree
  - Start symbol

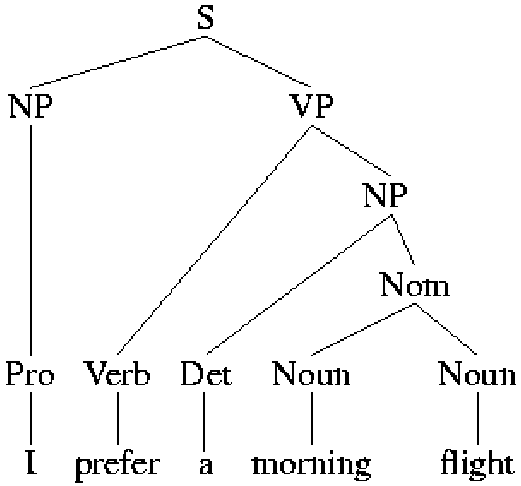


9.2 Context-Free Rules and Trees

Noun → *flight* | *breeze* | *trip* | *morning* | …
Verb → *is* | *prefer* | *like* | *need* | *want* | *fly* …
Adjective → *cheapest* | *non-stop* | *first* | *latest* | *other* | *direct* | …
Pronoun → *me* | *I* | *you* | *it* | …
Proper-Noun → *Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American* | …
Determiner → *the* | *a* | *an* | *this* | *these* | *that* | …
Preposition → *from* | *to* | *on* | *near* | …
Conjunction → *and* | *or* | *but* | …

The lexicon for L_0

S → *NP VP*	I + want a morning flight
NP → *Pronoun*	I
| *Proper-Noun*	Los Angeles
| *Det Nominal*	a + flight
Nominal → *Noun Nominal*	morning + flight
| *Noun*	flights
VP → *Verb*	do
| *Verb NP*	want + a flight
| *Verb NP PP*	leave + Boston + in the morning
| *Verb PP*	leaving + on Thursday
PP → *Preposition NP*	from + Los Angeles



The grammar for L_0

9.2 Context-Free Rules and Trees

- **Bracket notation** of parse tree
- Grammatical vs. ungrammatical sentences
- The use of formal languages to model NLs is called **generative grammar**, since the language is defined by the set of possible sentences “generated” by the grammar.
- The formal definition of a CFG is a 4-tuple.

9.3 Sentence-Level Constructions

- There are a great number of possible overall sentences structures, but four are particularly common and important:
  - *Declarative structure, imperative structure, yes-n-no-question structure*, and *wh-question structure*.
- Sentences with **declarative** structure
  - A subject NP followed by a VP
    - *The flight should be eleven a.m. tomorrow.*
    - *I need a flight to Seattle leaving from Baltimore making a stop in Minneapolis.*
    - *The return flight should leave at around seven p.m.*
    - *I would like to find out the flight number for the United flight that arrives in San Jose around ten p.m.*
    - *I'd like to fly the coach discount class.*
    - *I want a flight from Ontario to Chicago.*
    - *I plan to leave on July first around six thirty in the evening.*

9.3 Sentence-Level Constructions

- Sentence with **imperative** structure
  - Begin with a VP and have no subject.
  - Always used for commands and suggestions
    - *Show the lowest fare.*
    - *Show me the cheapest fare that has lunch.*
    - *Give me Sunday's flight arriving in Las Vegas from Memphis and New York City.*
    - *List all flights between five and seven p.m.*
    - *List all flights from Burbank to Denver.*
    - *Show me all flights that depart before ten a.m. and have first class fares.*
    - *Show me all the flights leaving Baltimore.*
    - *Show me flights arriving within thirty minutes of each other.*
    - *Please list the flights from Charlotte to Long Beach arriving after lunch time.*
    - *Show me the last flight to leave.*
  - $S \rightarrow VP$

9.3 Sentence-Level Constructions

- Sentences with **yes-no-question** structure
  - Begin with auxiliary, followed by a subject *NP*, followed by a *VP*.
    - *Do any of these flights have stops?*
    - *Does American's flight eighteen twenty five serve dinner?*
    - *Can you give me the same information for United?*
  - *S* → *Aux NP VP*

9.3 Sentence-Level Constructions

- The **wh-subject-question** structure
  - Identical to the declarative structure, except that the first NP contains some wh-word.
    - *What airlines fly from Burbank to Denver?*
    - *Which flights depart Burbank after noon and arrive in Denver by six p.m.?*
    - *Which flights serve breakfast?*
    - *Which of these flights have the longest layover Nashville?*
  - $S \rightarrow$ *Wh-NP VP*
- The **wh-non-subject-question** structure
    - *What flights do you have from Burbank to Tacoma Washington?*
  - $S \rightarrow$ *Wh-NP Aux NP VP*

9.4 The Noun Phrase

- View the NP as revolving around a **head**, the central noun in the NP.
  - The syntax of English allows for both pre-nominal (pre-head) modifiers and post-nominal (post-head) modifiers.

9.4 The Noun Phrase

Before the Head Noun

- NPs can begin with a determiner,
  - *a stop, the flights, that fare, this flight, those flights, any flights, some flights*
- Determiners can be optional,
  - *Show me* ***flights*** *from San Francisco to Denver on weekdays.*
- **Mass nouns** don’t require determination.
  - Substances, like *water* and *snow*
  - Abstract nouns, *music*, *homework*,
  - In the ATIS domain, *breakfast*, *lunch*, *dinner*
    - *Does this flight server* ***dinner****?*

9.4 The Noun Phrase

Before the Head Noun

- **Predeterminers:**
  - Word classes appearing in the NP before the determiner
    - ***all** the flights, **all** flights*
- **Postdeterminers:**
  - Word classes appearing in the NP between the determiner and the head noun
    - **Cardinal numbers:** *two friends, one stop*
    - **Ordinal numbers:** *the first one, the next day, the second leg, the last flight, the other American flight, and other fares*
    - **Quantifiers:** *many fares*
      - The quantifiers, *much* and *a little* occur only with noncount nouns.

9.4 The Noun Phrase

Before the Head Noun

- Adjectives occur after quantifiers but before nouns.
  - *a* ***first-class*** *fare, a* ***nonstop*** *flight, the* ***longest*** *layover, the* ***earliest*** *lunch flight*
- Adjectives can be grouped into a phrase called an **adjective phrase** or **AP.**
  - AP can have an adverb before the adjective
    - *the* ***least*** *expensive fare*
- $NP \rightarrow (Det)\ (Card)\ (Ord)\ (Quant)\ (AP)\ Nominal$

9.4 The Noun Phrase

After the Head Noun

- A head noun can be followed by **postmodifiers.**
  - Prepositional phrases
    - *All flights from Cleveland*
  - Non-finite clauses
    - *Any flights arriving after eleven a.m.*
  - Relative clauses
    - *A flight that serves breakfast*

9.4 The Noun Phrase

After the Head Noun

- PP postmodifiers
  - *any stopovers [for Delta seven fifty one]*
  - *all flight [from Cleveland] [to Newark]*
  - *arrival [in San Jose] [before seven a.m.]*
  - *a reservation [on flight six oh six] [from Tampa] [to Montreal]*
  - *Nominal → Nominal PP (PP) (PP)*

9.4 The Noun Phrase

After the Head Noun

- The three most common kinds of **non-finite** postmodifiers are the gerundive (-*ing*), -*ed*, and infinitive form.
  - A gerundive consists of a VP begins with the gerundive (-*ing*)
    - *any of those [leaving on Thursday]*
    - *any flights [arriving after eleven a.m.]*
    - *flights [arriving within thirty minutes of each other]*

$$
\begin{aligned}
&\textit{Nominal} \rightarrow \textit{Nominal GerundVP} \\
&\textit{GerundVP} \rightarrow \textit{GerundV NP} \mid \textit{GerundV PP} \mid \textit{GerundV} \mid \textit{GerundV NP PP} \\
&\textit{GerundV} \rightarrow \textit{being} \mid \textit{preferring} \mid \textit{ariving} \mid \textit{leaving} \mid \ldots
\end{aligned}
$$

  - Examples of two other common kinds
    - *the last flight to arrive in Boston*
    - *I need to have dinner served*
    - *Which is the aircraft used by this flight?*

9.4 The Noun Phrase

After the Head Noun

- A postnominal relative clause (more correctly a **restrictive relative clause**)
  - is a clause that often begins with a **relative pronoun** (*that* and *who* are the most common).
  - The relative pronoun functions as the subject of the embedded verb,
    - *a flight that serves breakfast*
    - *flights that leave in the morning*
    - *the United flight that arrives in San Jose around ten p.m.*
    - *the one that leaves at ten thirty five*

Nominal → *Nominal RelClause*
RelClause → *(who* | *that) VP*

9.4 The Noun Phrase

After the Head Noun

- The relative pronoun may also function as the object of the embedded verb,
  - *the earliest American Airlines flight that I can get*
- Various postnominal modifiers can be combined,
  - *a flight [from Phoenix to Detroit] [leaving Monday evening]*
  - *I need a flight [to Seattle] [leaving from Baltimore] [making a stop in Minneapolis]*
  - *evening flights [from Nashville to Houston] [that serve dinner]*
  - *a friend [living in Denver] [that would like to visit me here in Washington DC]*

9.5 Coordination

- NPs and other units can be **conjoined** with **coordinations** like *and*, *or*, and *but*.
  - *Please repeat [$_{NP}$ [$_{NP}$ the flight] and [$_{NP}$ the coast]]*
  - *I need to know [$_{NP}$ [$_{NP}$ the aircraft] and [$_{NP}$ flight number]]*
  - *I would like to fly from Denver stopping in [$_{NP}$ [$_{NP}$ Pittsburgh] and [$_{NP}$ Atlanta]]*
  - *NP → NP and NP*
  - *VP → VP and VP*
  - *S → S and S*

9.6 Agreement

- Most verbs in English can appear in two forms in the present tense:
  - 3sg, or non-3sg

Do [$_{NP}$ any flights] stop in Chicago?
Do [$_{NP}$ all of these flights] offer first class service?
Do [$_{NP}$ I] get dinner on this flight?
Do [$_{NP}$ you] have a flight from Boston to Forth Worth?
Does [$_{NP}$ this flight] stop in Dallas?
Does [$_{NP}$ that flight] serve dinner?
Does [$_{NP}$ Delta] fly from Atlanta to San Francisco?

What flight *leave* in the morning?
What flight *leaves* from Pittsburgh?

*[What flight] *leave* in the morning?
*Does [$_{NP}$ you] have a flight from Boston to Fort Worth?
*Do [$_{NP}$ this flight] stop in Dallas?

S → Aux NP VP

S → 3sgAux 3sgNP VP
S → Non3sgAux Non3sgNP VP
3sgAux → does | has | can | …
Non3sgAux → do | have | can | …
3sgNP → (Det) (Card) (Ord) (Quant) (AP) SgNominal
Non3sgNP → (Det) (Card) (Ord) (Quant) (AP) PlNominal
SgNominal → SgNoun | SgNoun SgNoun
PlNominal → PlNoun | SgNoun PlNoun
SgNoun → flight | fare | dollar | reservation | …
PlNoun → flights | fares | dollars | reservation | …

9.6 Agreement

- Problem for dealing with number agreement:
  - it doubles the size of the grammar.
- The rule proliferation also happen for the noun’s **case:**
  - For example, English pronouns have **nominative** (*I*, *she*, *he*, *they*) and **accusative** (*me*, *her*, *him*, *them*) versions.
- A more significant problem occurs in languages like German or French
  - Not only N-V agreement, but also **gender agreement.**
- A way to deal with these agreement problems without exploding the size of the grammar:
  - By effectively **parameterizing** each non-terminal of the grammar with **feature-structures.**

9.7 The Verb Phrase and Subcategorization

- The VP consists of the verb and a number of other constituents.

VP → *Verb*	disappear
VP → *Verb NP*	prefer a morning flight
VP → *Verb NP PP*	leave Boston in the morning
VP → *Verb PP*	leaving on Thursday

- An entire embedded sentence, called sentential complement, can follow the verb.

You [$_{VP}$ [$_{V}$ said [$_{S}$ there were two flights that were the cheapest]]]
You [$_{VP}$ [$_{V}$ said [$_{S}$ you had a two hundred sixty six dollar fare]]]
[$_{VP}$ [$_{V}$ Tell] [$_{NP}$ me] [$_{S}$ how to get from the airport in Philadelphia to downtown]]
I [$_{VP}$ [$_{V}$ think [$_{S}$ I would like to take the nine thirty flight]]

VP → *Verb S*

9.7 The Verb Phrase and Subcategorization

- Another potential constituent of the VP is another VP
  - Often the case for verbs like *want*, *would like*, *try*, *intent*, *need*

I want [$_{VP}$ to fly from Milwaukee to Orlando]
Hi, I want [$_{VP}$ to arrange three flights]
Hello, I'm trying [$_{VP}$ to find a flight that goes from Pittsburgh to Denver after two p.m.]

- Recall that verbs can also be followed by *particles*, word that resemble a preposition but that combine with the verb to form a *phrasal verb*, like *take off*.
  - These particles are generally considered to be an integral part of the verb in a way that other post-verbal elements are not;
  - Phrasal verbs are treated as individual verbs composed of two words.

9.7 The Verb Phrase and Subcategorization

- A VP can have many possible kinds of constituents, not every verb is compatible with every VP.
  - *I want a flight …*
  - *I want to fly to …*
  - **I found to fly to Dallas.*
- The idea that verbs are compatible with different kinds of complements
  - Traditional grammar **subcategorize** verbs into two categories (transitive and intransitive).
  - Modern grammars distinguish as many as 100 subcategories

Frame	Verb	Example
ϕ	eat, sleep	I want to eat
NP	prefer, **find** leave	Find [$_{NP}$ the flight from Pittsburgh to Boston]
NP NP	show, give, **find**	Show [$_{NP}$ me] [$_{NP}$ airlines with flights from Pittsburgh]
PP_{from} PP_{to}	fly, travel	I would like to fly [$_{PP}$ from Boston] [$_{PP}$ to Philadelphia]
NP $_{PPwith}$	help, load	Can you help [$_{NP}$ me] [$_{PP}$ with a flight]
VPto	prefer, want, need	I would prefer [$_{VPto}$ to go by United airlines]
VPbrst	can, would, might	I can [$_{VPbrst}$ fo from Boston]
S	mean	Does this mean [$_{S}$ AA has a hub in Boston?]

9.7 The Verb Phrase and Subcategorization

Verb-with-NP-complement → *find* | *leave* | *repeat* | …
Verb-with-S-complement → *think* | *believe* | *say* | …
Verb-with-Inf-VP-complement → *want* | *try* | *need* | …
VP → *Verb-with-no-complement* disappear
VP → *Verb-with-NP-complement NP* prefer a morning flight
VP → *Verb-with-S-complement S* said there were two flights

9.8 Auxiliaries

- **Auxiliaries** or **helping verbs**
  - A subclass of verbs
  - Having particular syntactic constraints which can be viewed as a kind of subcategorization
  - Including the **modal** verb, *can*, *could many*, *might*, *must*, *will*, *would*, *shall*, and *should*
  - The **perfect** auxiliary *have*,
  - The **progressive** auxiliary *be*, and
  - The **passive** auxiliary *be*.

9.8 Auxiliaries

- Modal verbs subcategorize for a *VP* whose head verb is a bare stem.
  - *can go in the morning, will try to find a flight*
- The perfect verb *have* subcategorizes for a *VP* whose head verb is the past participle form:
  - *have booked 3 flights*
- The progressive verb *be* subcategorizes for a *VP* whose head verb is the gerundive participle:
  - *am going from Atlanta*
- The passive verb *be* subcategorizes for a *VP* who head verb is the past participle:
  - *was delayed by inclement weather*

9.8 Auxiliaries

- A sentence may have multiple auxiliary verbs, but they must occur in a particular order.
  - modal < perfect < progressive < passive

modal perfect	could have been a contender
modal passive	will be married
perfect progressive	have been feasting
modal perfect passive	might have been prevented

9.9 Spoken Language Syntax

- *Skip*

9.10 Grammar Equivalence and Normal Form

- Two grammars are equivalent if they generate the same set of strings.
- Two kinds of equivalence
  - **Strong equivalence**
    - If two grammars generate the same set of strings *and* if they assign the same phrase structure to each sentence
  - **Weak equivalence**
    - Two grammars generate the same set of strings but do not assign the same phrase structure to each sentence.

9.10 Grammar Equivalence and Normal Form

- It is useful to have a **normal form** for grammars.
  - A CFG is in **Chomsky normal form** (CNF) if it is ε-free and if in addition each production is either of the form $A \rightarrow B\ C$ or $A \rightarrow a$
- Any grammar can be converted into a weakly-equivalent CNF grammar.
  - For example $A \rightarrow B\ C\ D$ can be converted into the following CNF rules:
    - $A \rightarrow B\ X$
    - $X \rightarrow C\ D$

9.11 Finite-State and Context-Free Grammars

- Recursion problem with finite-state grammars
  - Recursion cannot be handled in finite automata
  - Recursion is quite common in a complete model of NP

Nominal → Nominal PP

(Det)(Card)(Ord)(Quant)(AP)Nominal
*(Det)(Card)(Ord)(Quant)(AP)Nomina (PP)**
*(Det)(Card)(Ord)(Quant)(AP)Nomina (P NP)**

*(Det)(Card)(Ord)(Quant)(AP)Nomina (RelClause|GerundVP|PP)**

- An augmented version of the FSA: the recursive transition network or RTN

Chapter 10. Parsing with CFGs

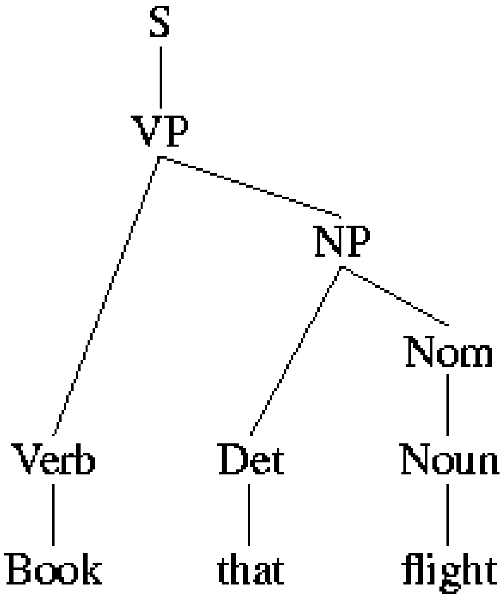
From: Chapter 10 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

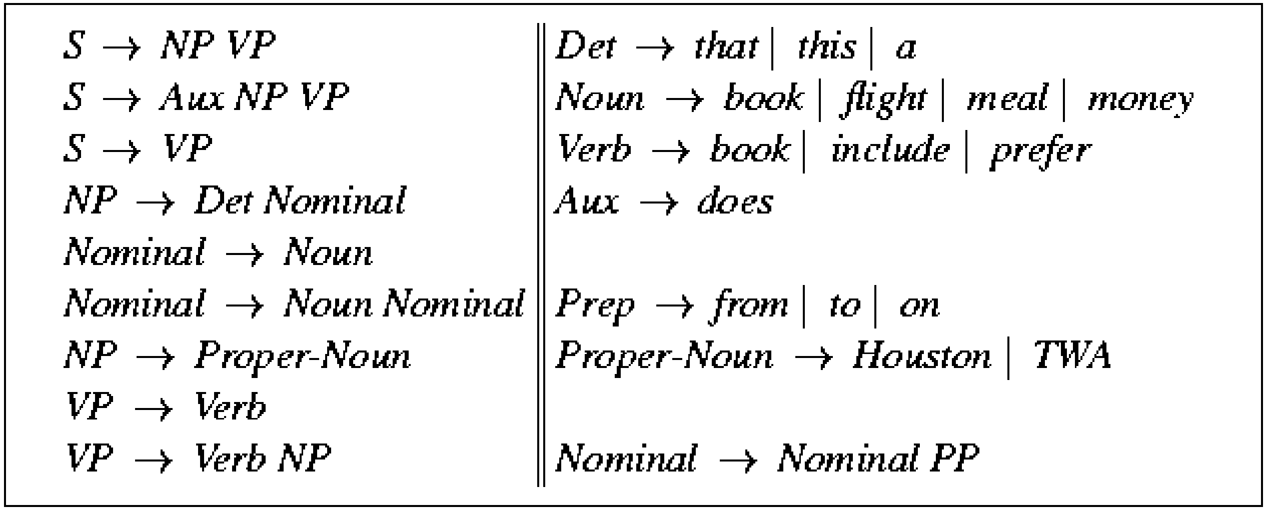
Background

- Syntactic parsing
  - The task of recognizing a sentence and assigning a syntactic structure to it
- Since CFGs are a declarative formalism, they do not specify how the parse tree for a given sentence should be computed.
- Parse trees are useful in applications such as
  - Grammar checking
  - Semantic analysis
  - Machine translation
  - Question answering
  - Information extraction

10.1 Parsing as Search

- The parser can be viewed as searching through the space of all possible parse trees to find the correct parse tree fo the sentence.
- ***How can we use the grammar to produce the parse tree***

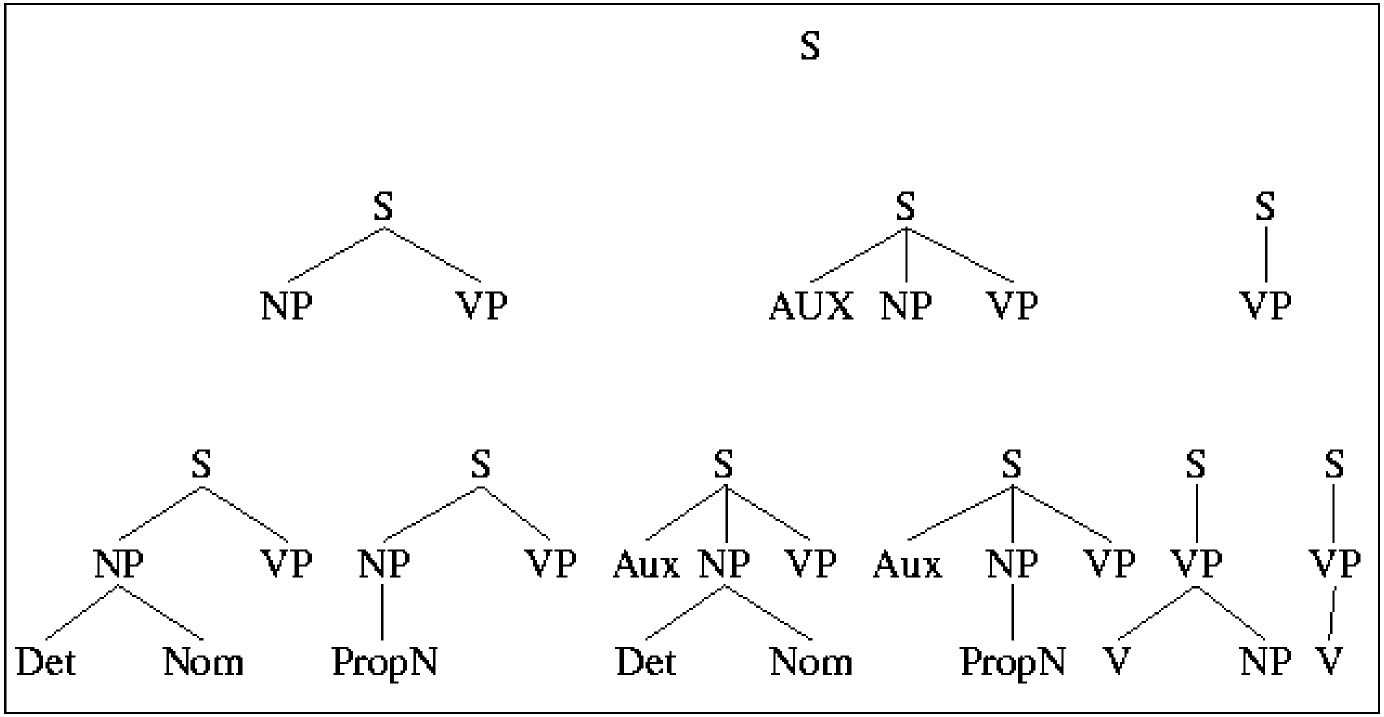




S → NP VP	*Det → that | this | a*
S → Aux NP VP	*Noun → book | flight | meal | money*
S → VP	*Verb → book | include | prefer*
NP → Det Nominal	*Aux → does*
Nominal → Noun	
Nominal → Noun Nominal	*Prep → from | to | on*
NP → Proper-Noun	*Proper-Noun → Houston | TWA*
VP → Verb	
VP → Verb NP	*Nominal → Nominal PP*

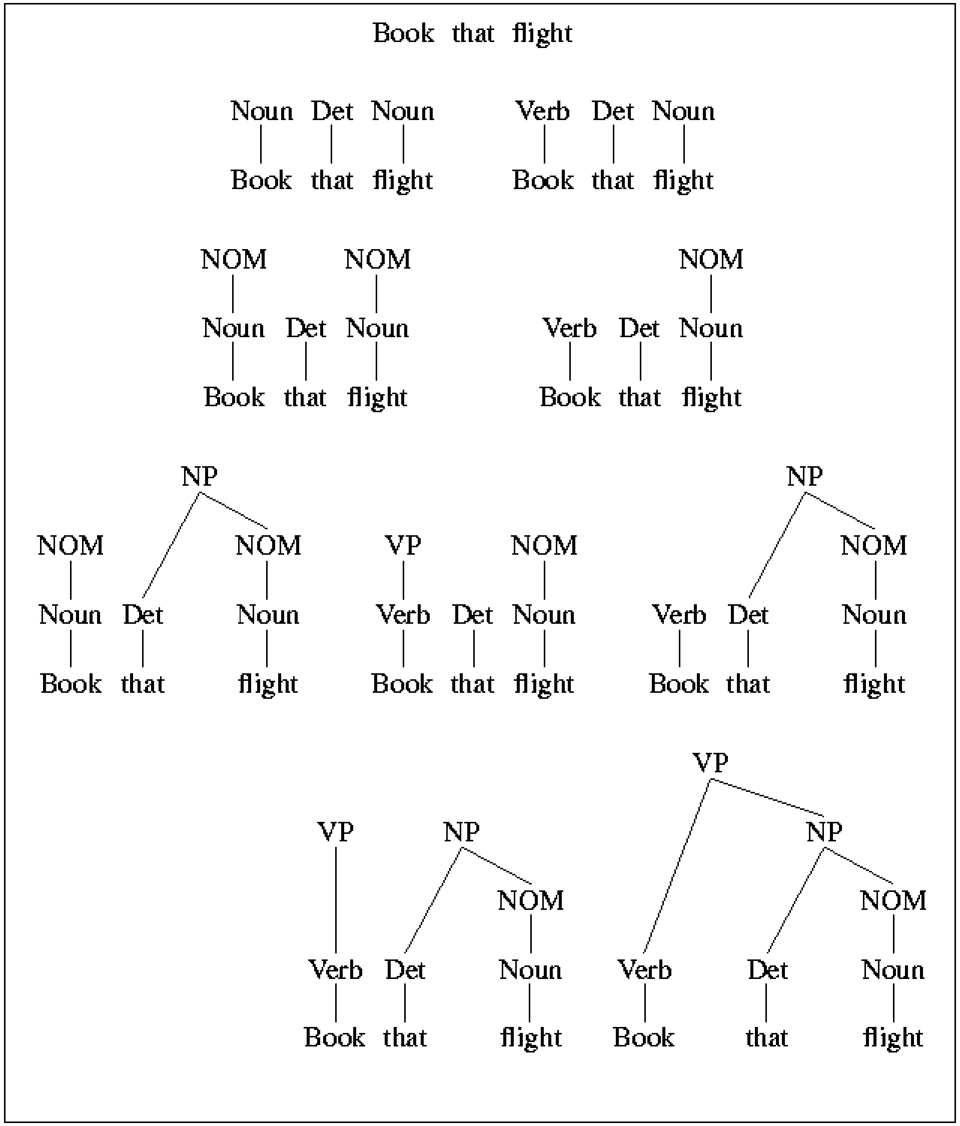
10.1 Parsing as Search

- Top-down parsing



10.1 Parsing as Search

- Bottom-up parsing

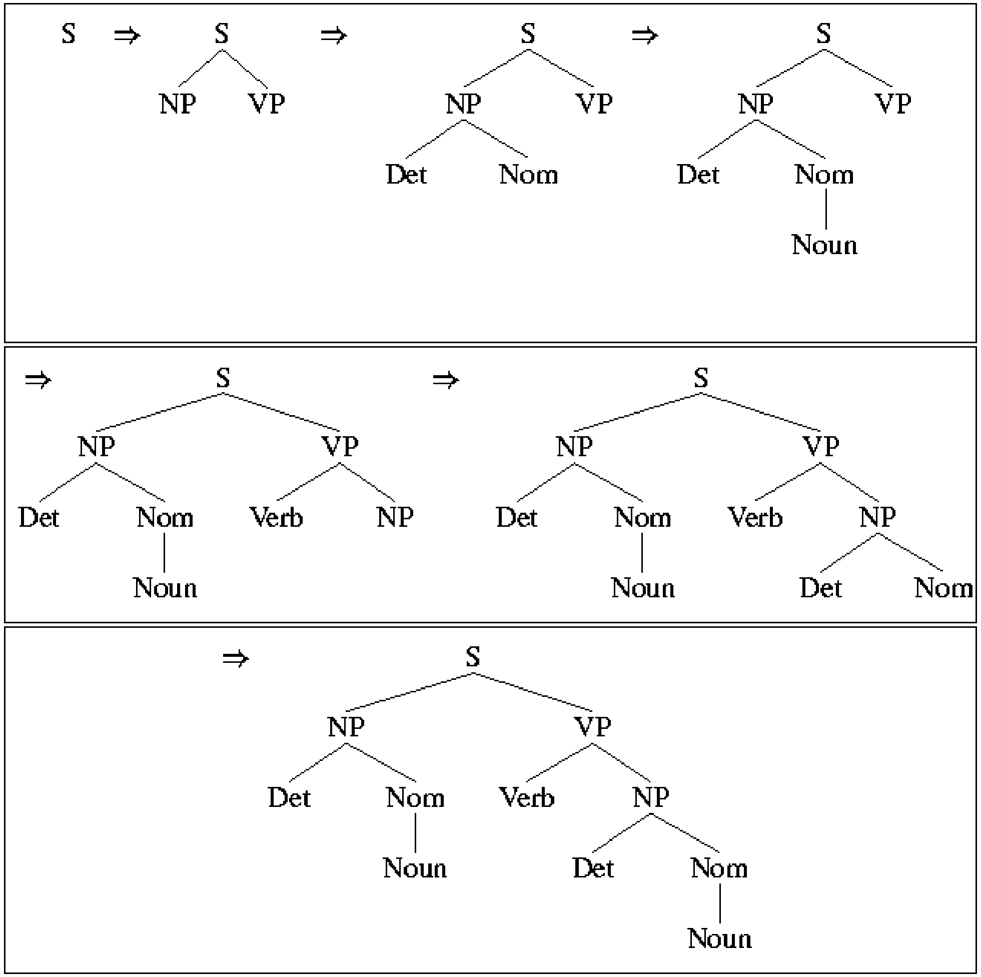


10.1 Parsing as Search

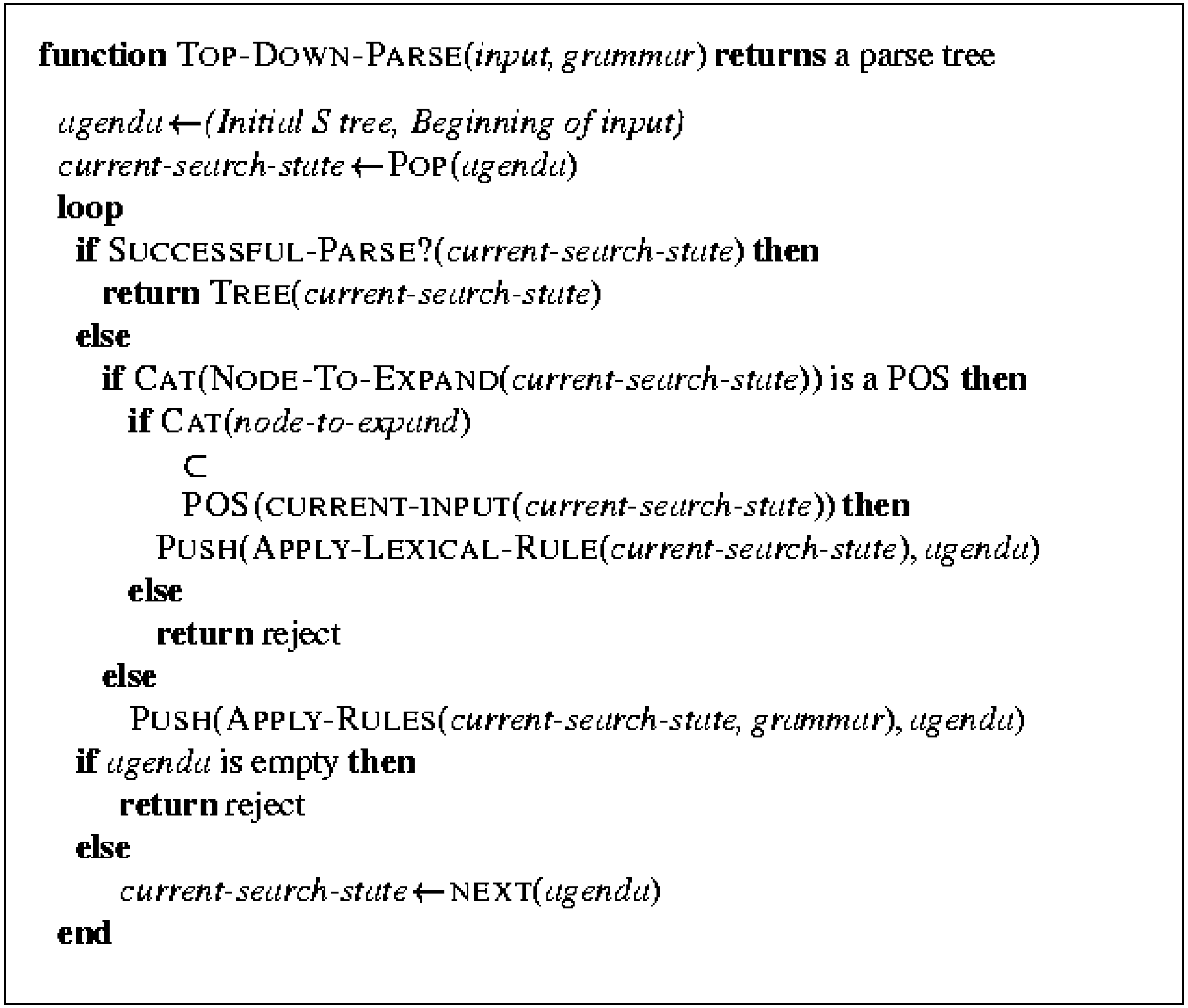
- Comparisons
  - The top-down strategy never wastes time exploring trees that cannot result in an *S*.
  - The bottom-up strategy, by contrast, trees that have no hope to leading to an *S*, or fitting in with any of their neighbors, are generated with wild abandon.
    - The left branch of Fig. 10.4 is completely wasted effort.
  - Spend considerable effort on *S* trees that are not consistent with the input.
    - The first four of the six trees in Fig. 10.3 cannot match the word *book*.

10.2 A Basic Top-Down Parser

- Use depth-first strategy



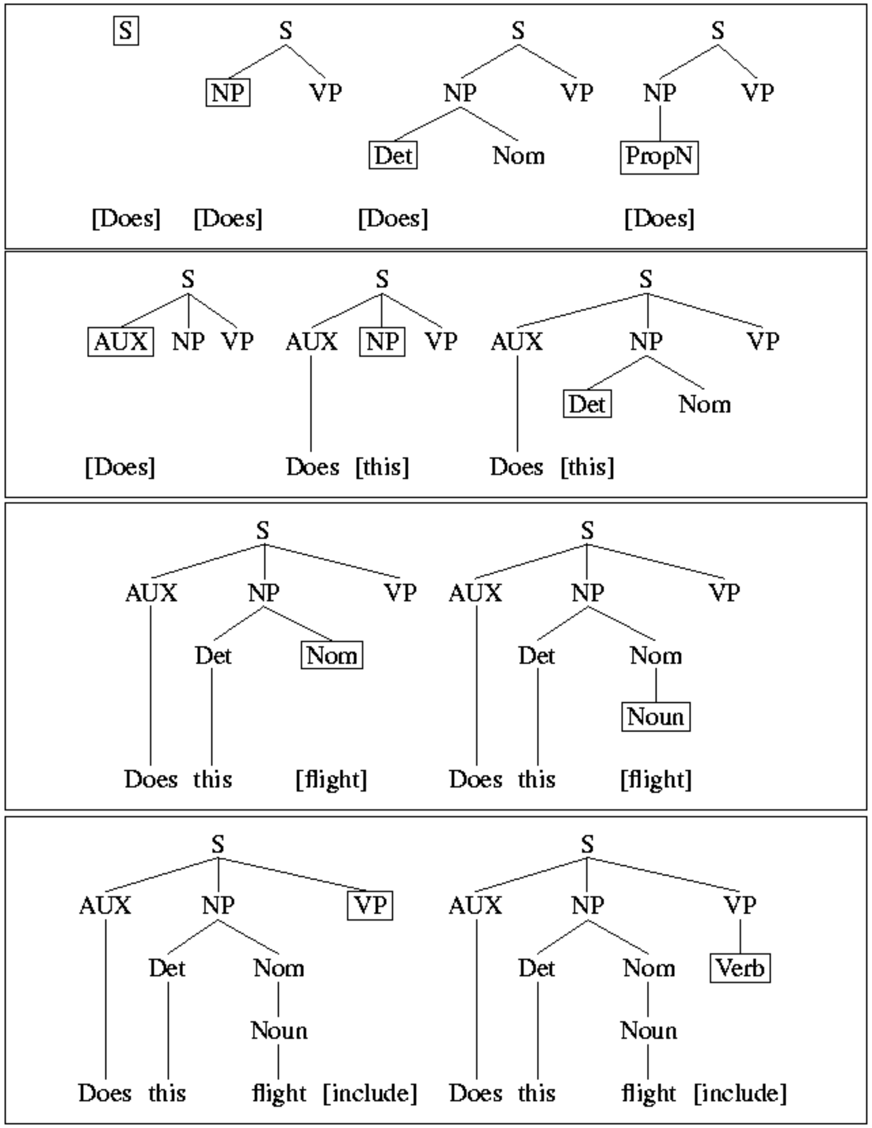
10.2 A Basic Top-Down Parser



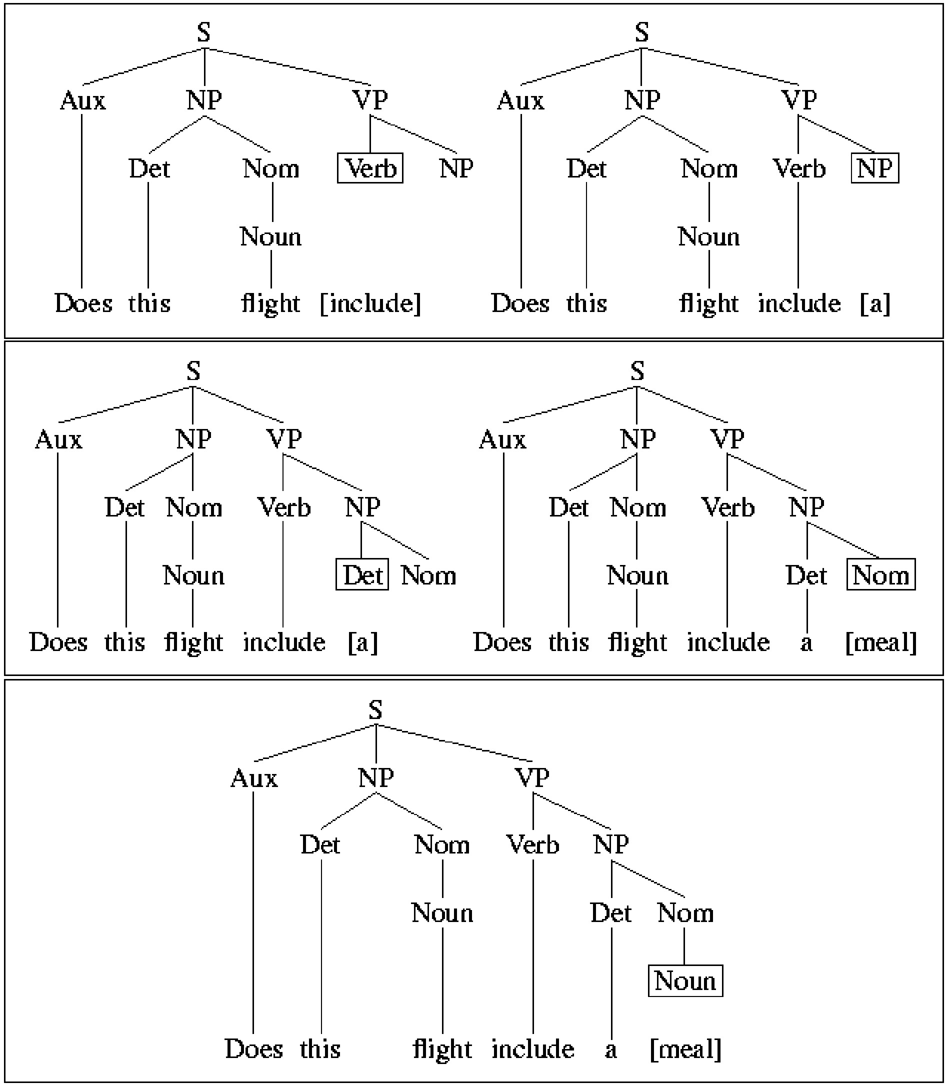
```
function TOP-DOWN-PARSE(input, grammar) returns a parse tree

 agenda ← (Initial S tree, Beginning of input)
 current-search-state ← POP(agenda)
 loop
   if SUCCESSFUL-PARSE?(current-search-state) then
     return TREE(current-search-state)
   else
     if CAT(NODE-TO-EXPAND(current-search-state)) is a POS then
       if CAT(node-to-expand)
            ⊂
            POS(CURRENT-INPUT(current-search-state)) then
          PUSH(APPLY-LEXICAL-RULE(current-search-state), agenda)
       else
          return reject
     else
       PUSH(APPLY-RULES(current-search-state, grammar), agenda)
   if agenda is empty then
      return reject
   else
      current-search-state ← NEXT(agenda)
 end
```

10.2 A Basic Top-Down Parser

- A top-down, depth-first, left-to-right derivation



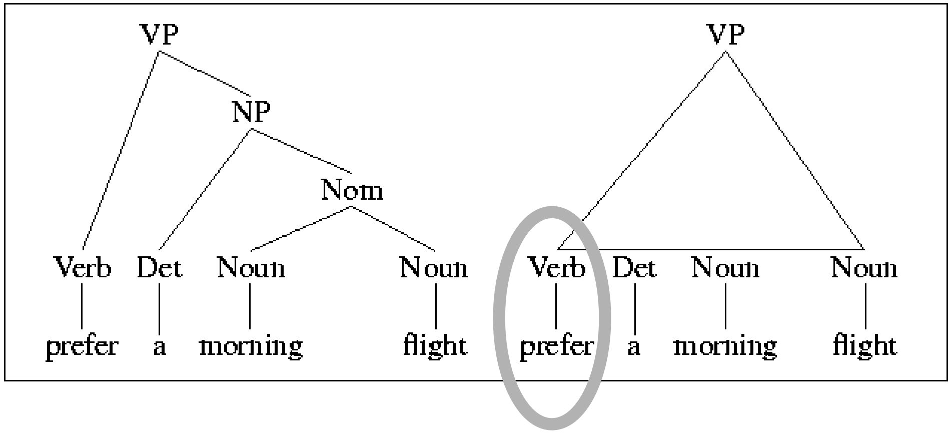
10.2 A Basic Top-Down Parser



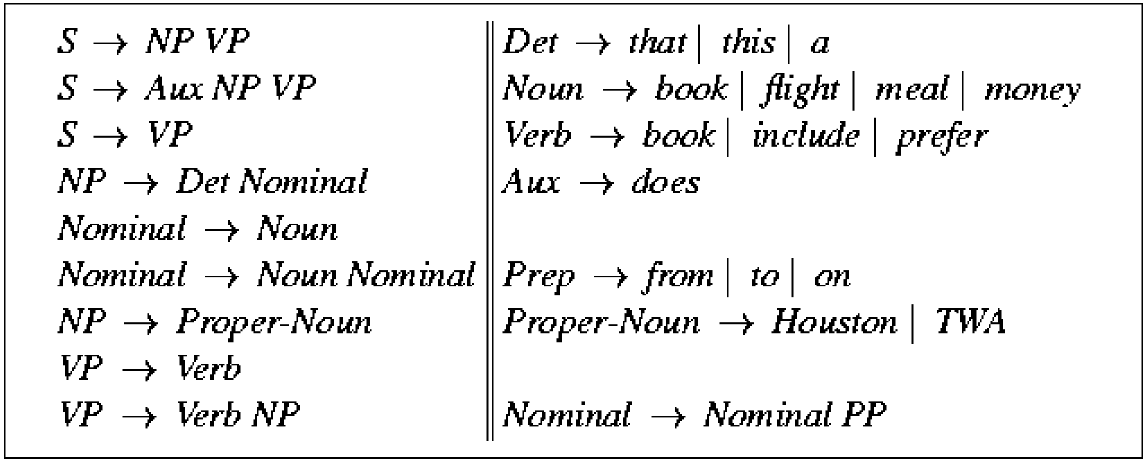
10.2 A Basic Top-Down Parser

- Adding bottom-up filtering
- Left-corner notion
  - For nonterminals *A* and *B*, *B* is a left-corner of *A* if the following relation holds: $A \overset{*}{\Rightarrow} B\alpha$
- Using the left-corner notion, it is easy to see that only the $S \rightarrow Aux\ NP\ VP$ rule is a viable candidate Since the word *Does* can not server as the left-corner of other two *S*-rules.

$$S \rightarrow NP\ VP$$
$$A \rightarrow Aux\ NP\ VP$$
$$S \rightarrow VP$$



10.2 A Basic Top-Down Parser



S → *NP VP*
S → *Aux NP VP*
S → *VP*
NP → *Det Nominal*
Nominal → *Noun*
Nominal → *Noun Nominal*
NP → *Proper-Noun*
VP → *Verb*
VP → *Verb NP*

Det → *that* | *this* | *a*
Noun → *book* | *flight* | *meal* | *money*
Verb → *book* | *include* | *prefer*
Aux → *does*

Prep → *from* | *to* | *on*
Proper-Noun → *Houston* | *TWA*

Nominal → *Nominal PP*

Category	**Left Corners**
S	Det, Proper-Noun, Aux, Verb
NP	Det, Proper-Noun
Nominal	Noun
VP	Verb

10.3 Problems with the Basic Top-Down Parser

- Problems with the top-down parser
  - Left-recursion
  - Ambiguity
  - Inefficiency reparsing of subtrees
- Then, introducing the Earley algorithm

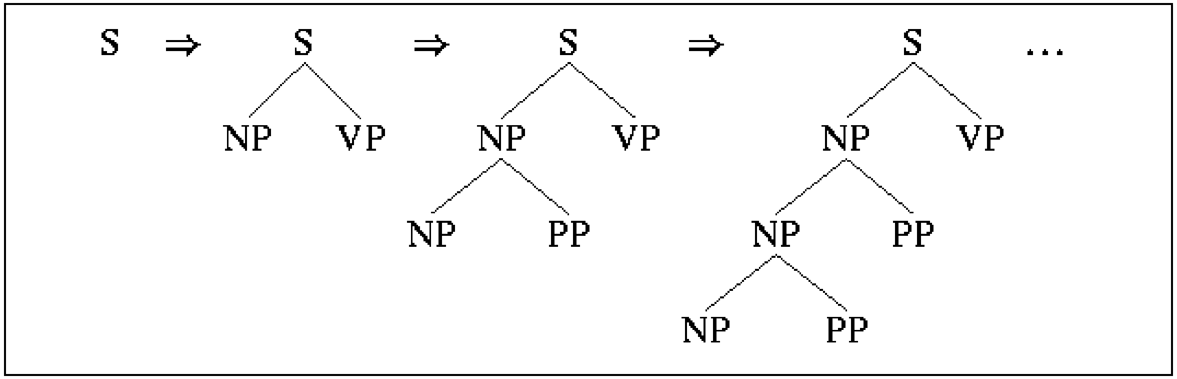
10.3 Problems with the Basic Top-Down Parser
Left-Recursion

- Exploring infinite search space, when **left-recursive grammars** are used
- A grammar is left-recursive if it contains at least one NT *A*, such that $A \overset{*}{\Rightarrow} \alpha A \beta$, for some α and β and $\alpha \overset{*}{\Rightarrow} \varepsilon$.

$NP \rightarrow Det\ Nominal$
$Det \rightarrow NP\ `s$

$NP \rightarrow NP\ PP$
$VP \rightarrow VP\ PP$
$S \rightarrow S\ and\ S$

Left-recursive rules



10.3 Problems with the Basic Top-Down Parser

Left-Recursion

- Two reasonable methods for dealing with left-recursion in a backtracking top-down parser:
  - Rewriting the grammar
  - Explicitly managing the depth of the search during parsing
- Rewrite each rule of left-recursion

$A \rightarrow A\beta \mid \alpha$
$\Rightarrow$
$A \rightarrow \alpha A'$
$A' \rightarrow \beta A \mid \varepsilon$

10.3 Problems with the Basic Top-Down Parser

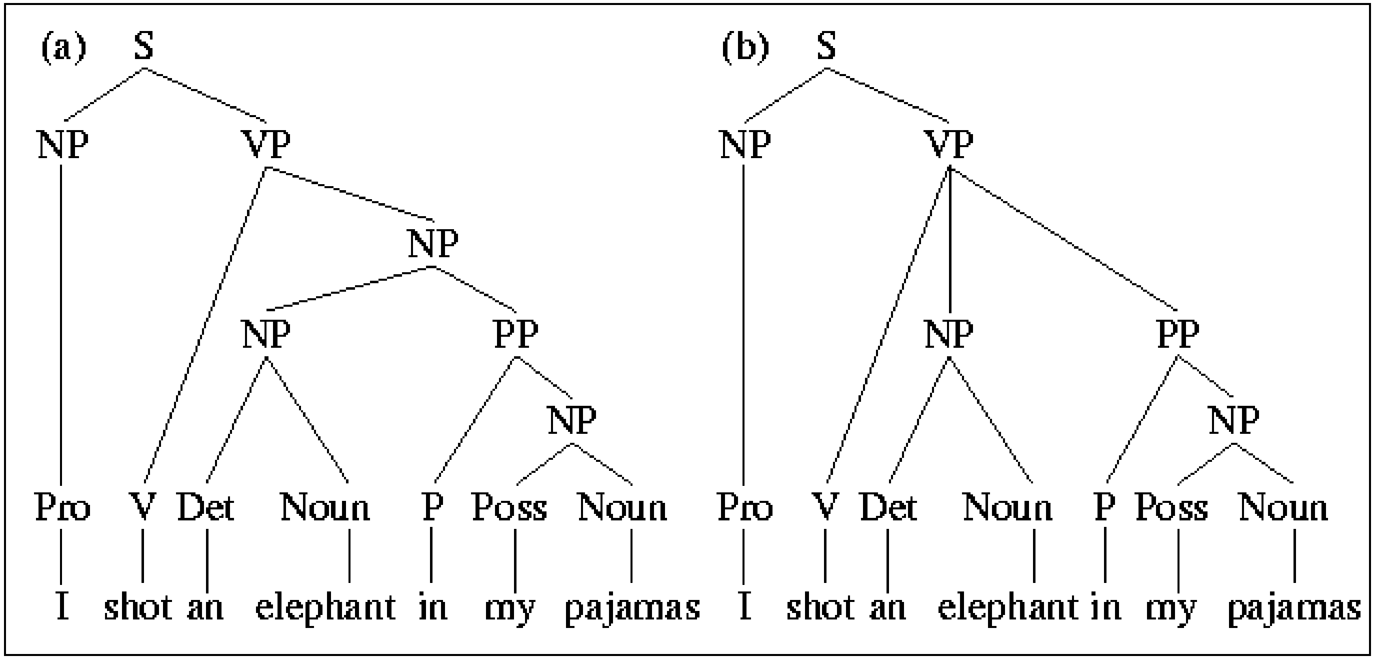
Ambiguity

- Common structural ambiguity
  - Attachment ambiguity
  - Coordination ambiguity
  - NP bracketing ambiguity

10.3 Problems with the Basic Top-Down Parser

Ambiguity

- Example of PP attachment



10.3 Problems with the Basic Top-Down Parser

Ambiguity

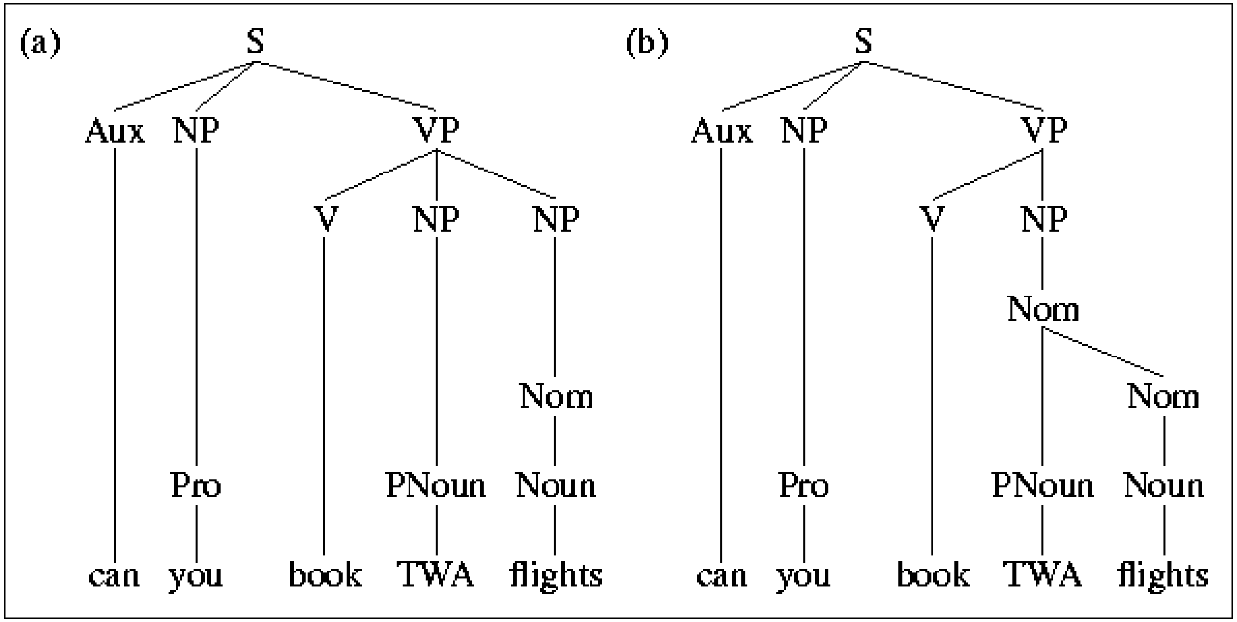
We saw the Eiffel Tower *flying to*

- The gerundive-VP *flying to Paris* can be
  - part of a gerundive sentence, or
  - an adjunct modifying the VP

10.3 Problems with the Basic Top-Down Parser

Ambiguity

- The sentence “Can you book TWA flights” is ambiguous
  - “Can you book flights on behalf of TWA”
  - “Can you book flights run by TWA”



10.3 Problems with the Basic Top-Down Parser

Ambiguity

- Coordination ambiguity
  - Different set of phrases that can be conjoined by a conjunction like *and*.
  - For example *old men and women* can be
    - [*old* [*men and women*]] or [*old men*] *and* [*women*]
- Parsing sentence thus requires disambiguation:
  - Choosing the correct parse from a multitude of possible parser
  - Requiring both statistical (Ch 12) and semantic knowledge (Ch 17)

10.3 Problems with the Basic Top-Down Parser

Ambiguity

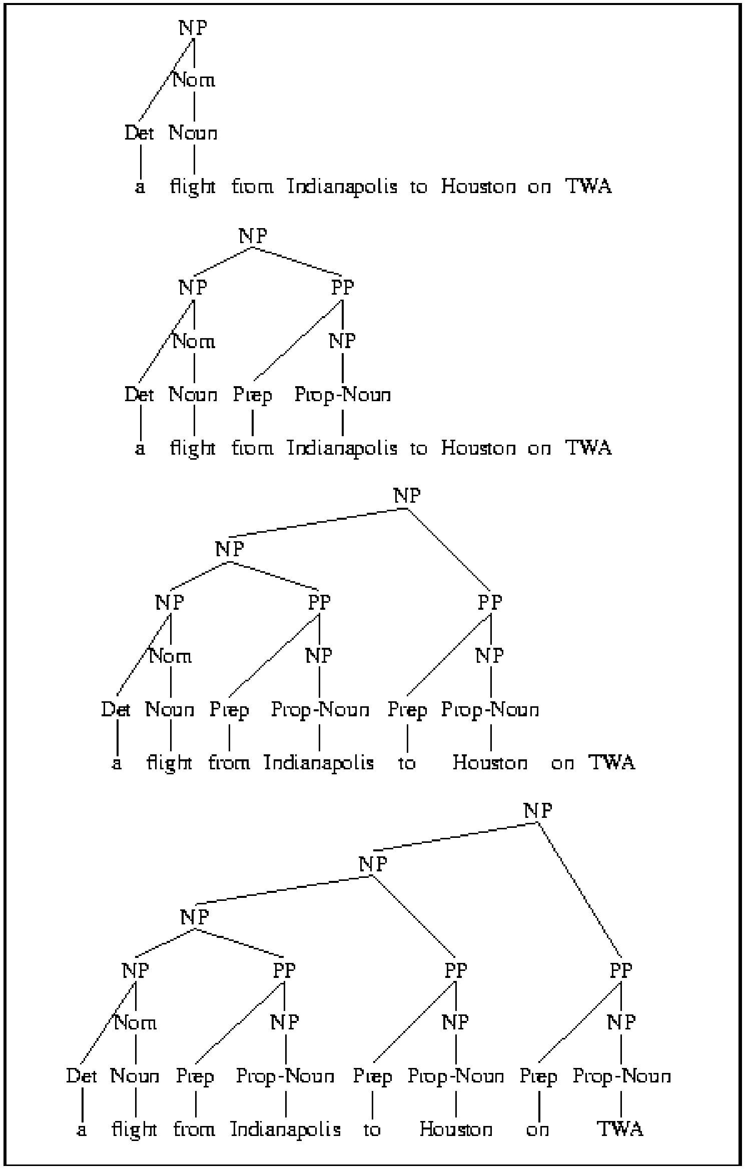
- Parsers which do not incorporate disambiguators may simply return all the possible parse trees for a given input.
- We do not want all possible parses from the robust, highly ambiguous, wide-coverage grammars used in practical applications.
- Reason:
  - Potentially exponential number of parses that are possible for certain inputs
  - Given the ATIS example:
    - Show me the meal on Flight UA 386 from San Francisco to Denver.
  - The three PP's at the end of this sentence yield a total of 14 parse trees for this sentence.

10.3 Problems with the Basic Top-Down Parser

Repeated Parsing Subtrees

- The parser often builds valid parse trees for portion of the input, then discards them during backtracking, only to find that it has to rebuild them again.

a flight	4
From Indianapolis	3
To Houston	2
On TWA	1
A flight from Indianapolis	3
A flight from Indianapolis to Houston	2
A flight from Indianapolis to Houston on TWA	1


NP
Nom
Det
Noun
a flight from Indianapolis to Houston on TWA
NP
NP
PP
Nom
NP
Det
Noun
Prep
Prop-Noun
a flight from Indianapolis to Houston on TWA
NP
NP
NP
PP
PP
Nom
NP
NP
Det
Noun
Prep
Prop-Noun
Prep
Prop-Noun
a flight from Indianapolis to Houston on TWA
NP
NP
NP
NP
PP
PP
PP
Nom
NP
NP
NP
Det
Noun
Prep
Prop-Noun
Prep
Prop-Noun
Prep
Prop-Noun
a flight from Indianapolis to Houston on TWA

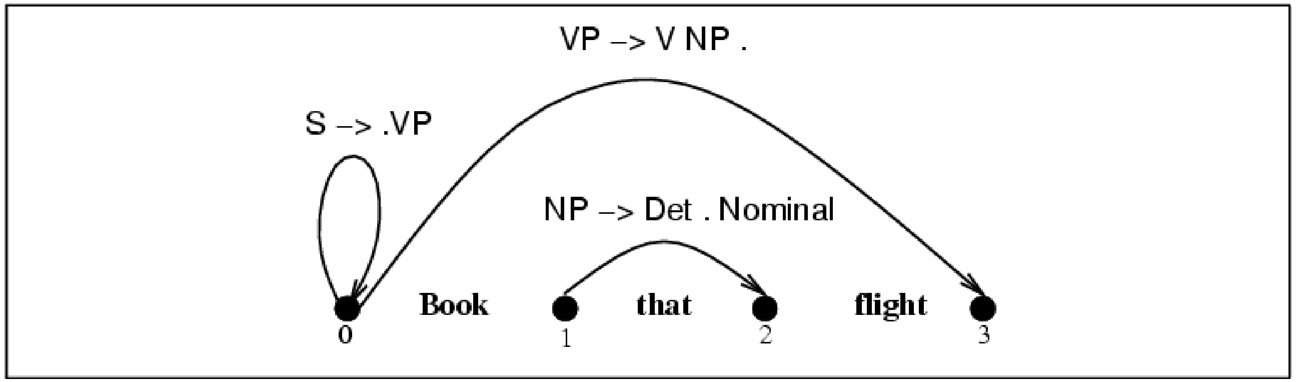
10.4 The Earley Algorithm

- Solving three kinds of problems afflicting standard bottom-up or top-down parsers
- Dynamic programming providing a framework for solving this problem
  - Systematically fill in tables of solutions to sub-problems.
  - When complete, the tables contain solution to all sub-problems needed to solve the problem as a whole.
  - Reducing an exponential-time problem to a polynomial-time one by eliminating the repetitive solution of sub-problems inherently iin backtracking approaches
  - $O(N^3)$, where N is the number of words in the input

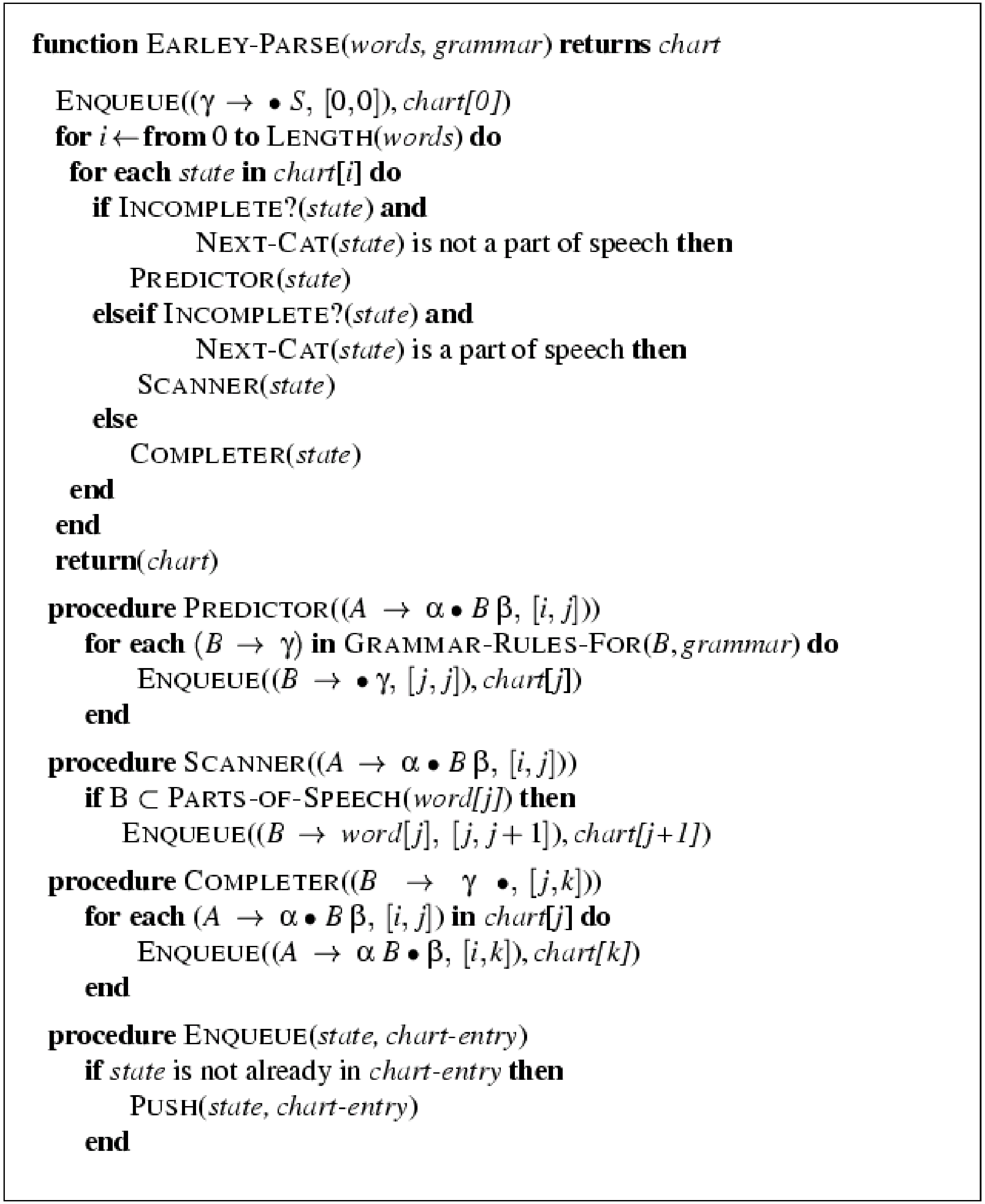
10.4 The Earley Algorithm

- (10.7) Book that flight

$S \rightarrow \bullet VP$, [0,0]
$NP \rightarrow Det\ \ \bullet Nominal$, [1,2]
$VP \rightarrow V\ \ NP \bullet$, [0,3]

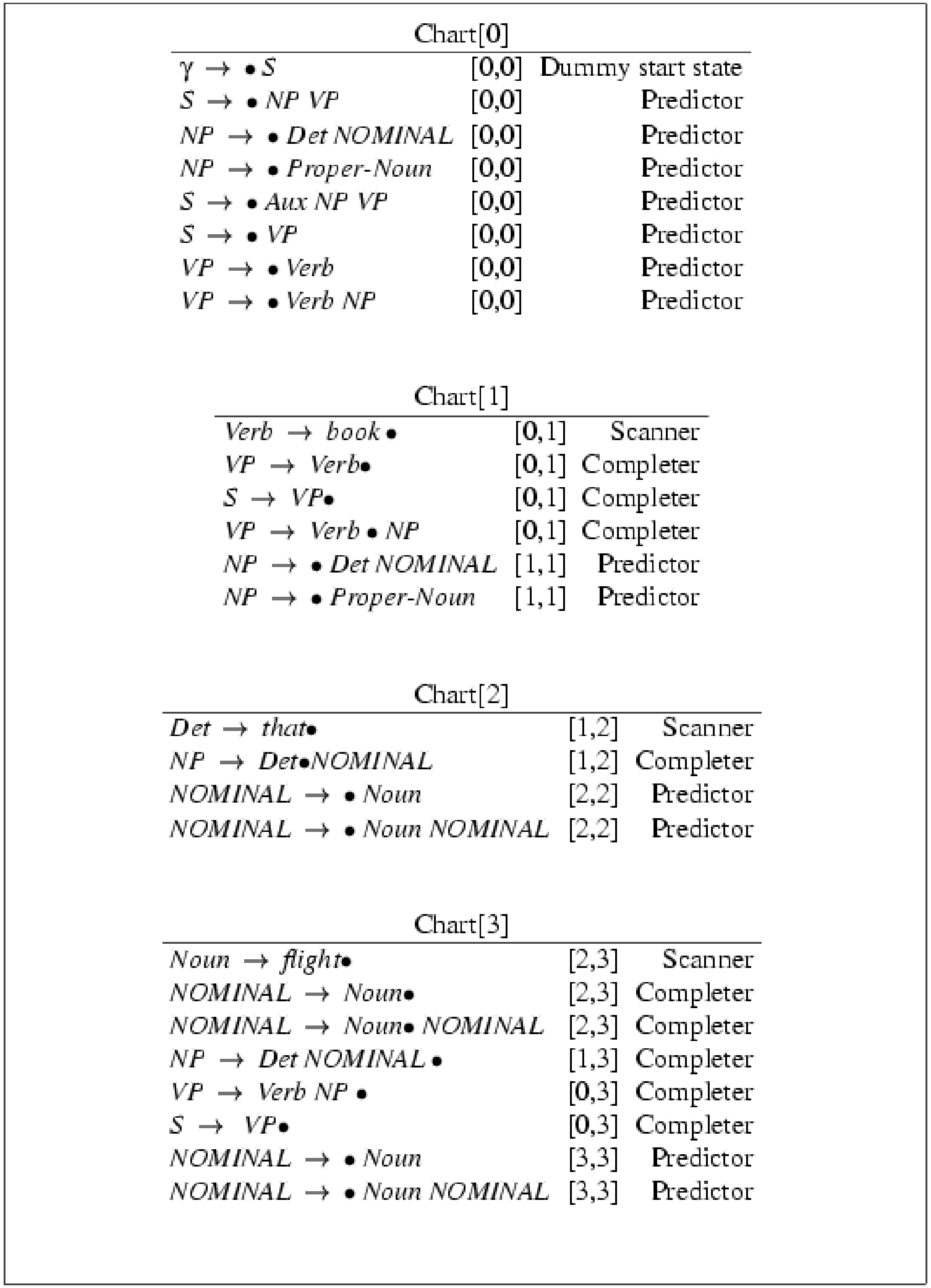


10.4 The Earley Algorithm



```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE(($\gamma \to \bullet S$, [0,0]), chart[0])
  for $i \leftarrow$ from 0 to LENGTH(words) do
   for each state in chart[i] do
     if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
     elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
        SCANNER(state)
     else
        COMPLETER(state)
   end
  end
  return(chart)

 procedure PREDICTOR(($A \to \alpha \bullet B \beta$, [i, j]))
    for each ($B \to \gamma$) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE(($B \to \bullet \gamma$, [j, j]), chart[j])
    end

 procedure SCANNER(($A \to \alpha \bullet B \beta$, [i, j]))
    if B $\subset$ PARTS-OF-SPEECH(word[j]) then
       ENQUEUE(($B \to word[j]$, [j, j + 1]), chart[j+1])

 procedure COMPLETER(($B \to \gamma \bullet$, [j,k]))
    for each ($A \to \alpha \bullet B \beta$, [i, j]) in chart[j] do
        ENQUEUE(($A \to \alpha B \bullet \beta$, [i,k]), chart[k])
    end

 procedure ENQUEUE(state, chart-entry)
    if state is not already in chart-entry then
       PUSH(state, chart-entry)
    end
```

10.4 The Earley Algorithm



Chart[0]

γ → •S	[0,0]	Dummy start state
S → •NP VP	[0,0]	Predictor
NP → •Det NOMINAL	[0,0]	Predictor
NP → •Proper-Noun	[0,0]	Predictor
S → •Aux NP VP	[0,0]	Predictor
S → •VP	[0,0]	Predictor
VP → •Verb	[0,0]	Predictor
VP → •Verb NP	[0,0]	Predictor

Chart[1]

Verb → book•	[0,1]	Scanner
VP → Verb•	[0,1]	Completer
S → VP•	[0,1]	Completer
VP → Verb•NP	[0,1]	Completer
NP → •Det NOMINAL	[1,1]	Predictor
NP → •Proper-Noun	[1,1]	Predictor

Chart[2]

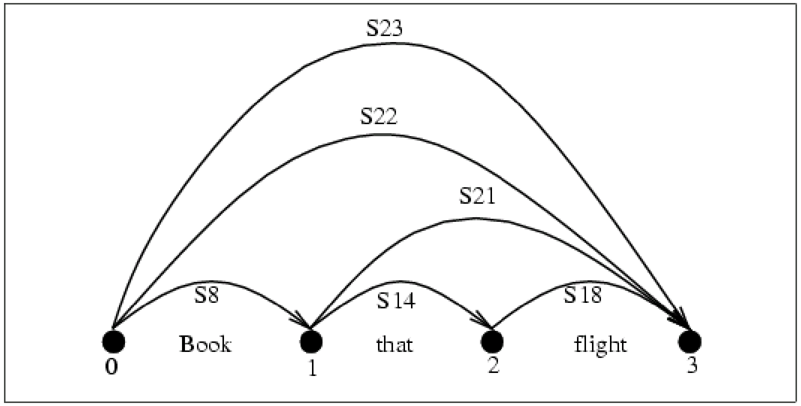
Det → that•	[1,2]	Scanner
NP → Det•NOMINAL	[1,2]	Completer
NOMINAL → •Noun	[2,2]	Predictor
NOMINAL → •Noun NOMINAL	[2,2]	Predictor

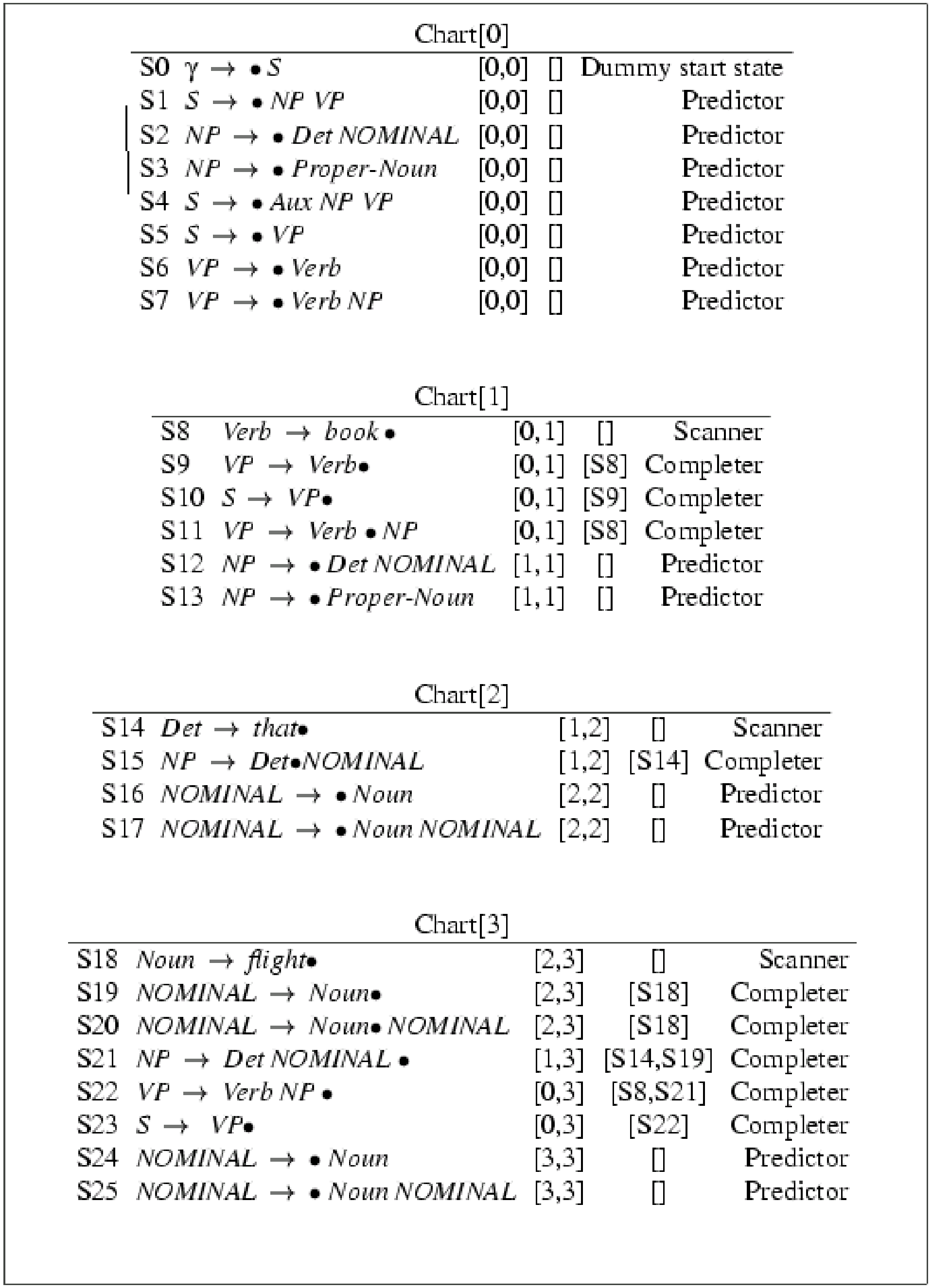
Chart[3]

Noun → flight•	[2,3]	Scanner
NOMINAL → Noun•	[2,3]	Completer
NOMINAL → Noun• NOMINAL	[2,3]	Completer
NP → Det NOMINAL•	[1,3]	Completer
VP → Verb NP•	[0,3]	Completer
S → VP•	[0,3]	Completer
NOMINAL → •Noun	[3,3]	Predictor
NOMINAL → •Noun NOMINAL	[3,3]	Predictor

10.4 The Earley Algorithm

Sequence of state created in Chart while parsing Book that flight including Structural information





Chart[0]

S0	γ → • S	[0,0]	[]	Dummy start state
S1	S → • NP VP	[0,0]	[]	Predictor
S2	NP → • Det NOMINAL	[0,0]	[]	Predictor
S3	NP → • Proper-Noun	[0,0]	[]	Predictor
S4	S → • Aux NP VP	[0,0]	[]	Predictor
S5	S → • VP	[0,0]	[]	Predictor
S6	VP → • Verb	[0,0]	[]	Predictor
S7	VP → • Verb NP	[0,0]	[]	Predictor

Chart[1]

S8	Verb → book •	[0,1]	[]	Scanner
S9	VP → Verb•	[0,1]	[S8]	Completer
S10	S → VP•	[0,1]	[S9]	Completer
S11	VP → Verb • NP	[0,1]	[S8]	Completer
S12	NP → • Det NOMINAL	[1,1]	[]	Predictor
S13	NP → • Proper-Noun	[1,1]	[]	Predictor

Chart[2]

S14	Det → that•	[1,2]	[]	Scanner
S15	NP → Det•NOMINAL	[1,2]	[S14]	Completer
S16	NOMINAL → • Noun	[2,2]	[]	Predictor
S17	NOMINAL → • Noun NOMINAL	[2,2]	[]	Predictor

Chart[3]

S18	Noun → flight•	[2,3]	[]	Scanner
S19	NOMINAL → Noun•	[2,3]	[S18]	Completer
S20	NOMINAL → Noun• NOMINAL	[2,3]	[S18]	Completer
S21	NP → Det NOMINAL •	[1,3]	[S14,S19]	Completer
S22	VP → Verb NP •	[0,3]	[S8,S21]	Completer
S23	S → VP•	[0,3]	[S22]	Completer
S24	NOMINAL → • Noun	[3,3]	[]	Predictor
S25	NOMINAL → • Noun NOMINAL	[3,3]	[]	Predictor

10.5 Finite-State Parsing Methods

- ***Partial parsing*** or ***shallow parsing***
  - Some language processing tasks do not require complete parses.
  - E.g., ***information extraction*** algorithms generally do **not extract all** the possible information in a text; they simply **extract enough** to fill out some sort of template of required data.
- Many partial parsing systems use **cascade** of finite-state automata instead of CFGs.
  - Use FSA to recognize **basic phrases**, such as noun groups, verb groups, locations, etc.
  - FASTUS of SRI

Company Name:	Bridgestone Sports Co.
Verb Group:	said
Noun Group:	Friday
Noun Group:	it
Verb Group	had set up
Noun Group:	a joint venture
Preposition	in
Location:	Taiwan

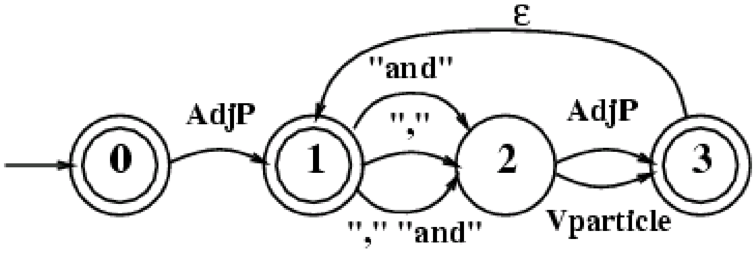
Preposition:	with
Noun Group:	a local concern
Conjunction:	and
Noun Group:	a Japanese trading hounse
Verb Group	to produce
Noun Group:	golf clubs
Verb Group:	to be shipped
Preposition	to
Location:	Japan

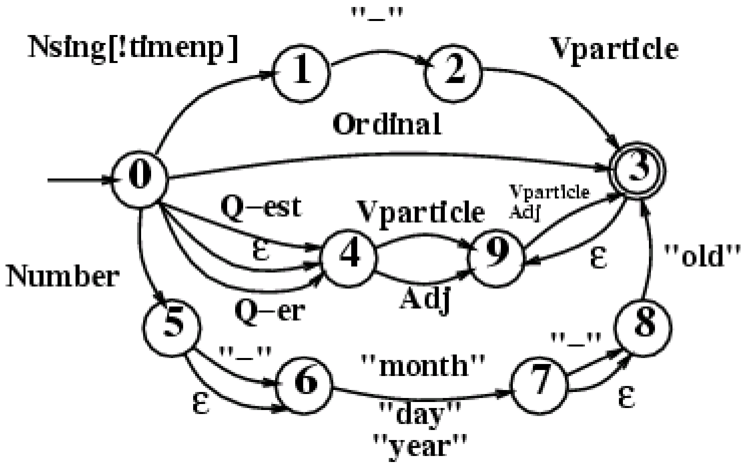
10.5 Finite-State Parsing Methods

- Detection of noun group

NG → Pronoun | Time-NP | Date-NP *she, him, them, yesterday*
NG → (DETP) (Adjs) HdNns | DETP Ving HdNns *the quick and dirty solution,*
the frustrating mathematics problem, the rising index
DETP → DETP-CP | DET-INCP
DETP-CP → …
DETP-INCP → …
Adjs → AdjP …
AdjP → …
HdNns → HdNn …
HdNn → PropN |PreNs …
PreNs → PreN …
PreN → ..

10.5 Finite-State Parsing Methods





Parsing

SLP Chapter 13

Outline

- Parsing with CFGs
  - Bottom-up, top-down
  - CKY parsing
  - Mention of Earley and chart parsing

Parsing

- Parsing with CFGs refers to the task of assigning trees to input strings
- Trees that covers all and only the elements of the input and has an S at the top
- This chapter: find all possible trees
- Next chapter (14): choose the most probable one

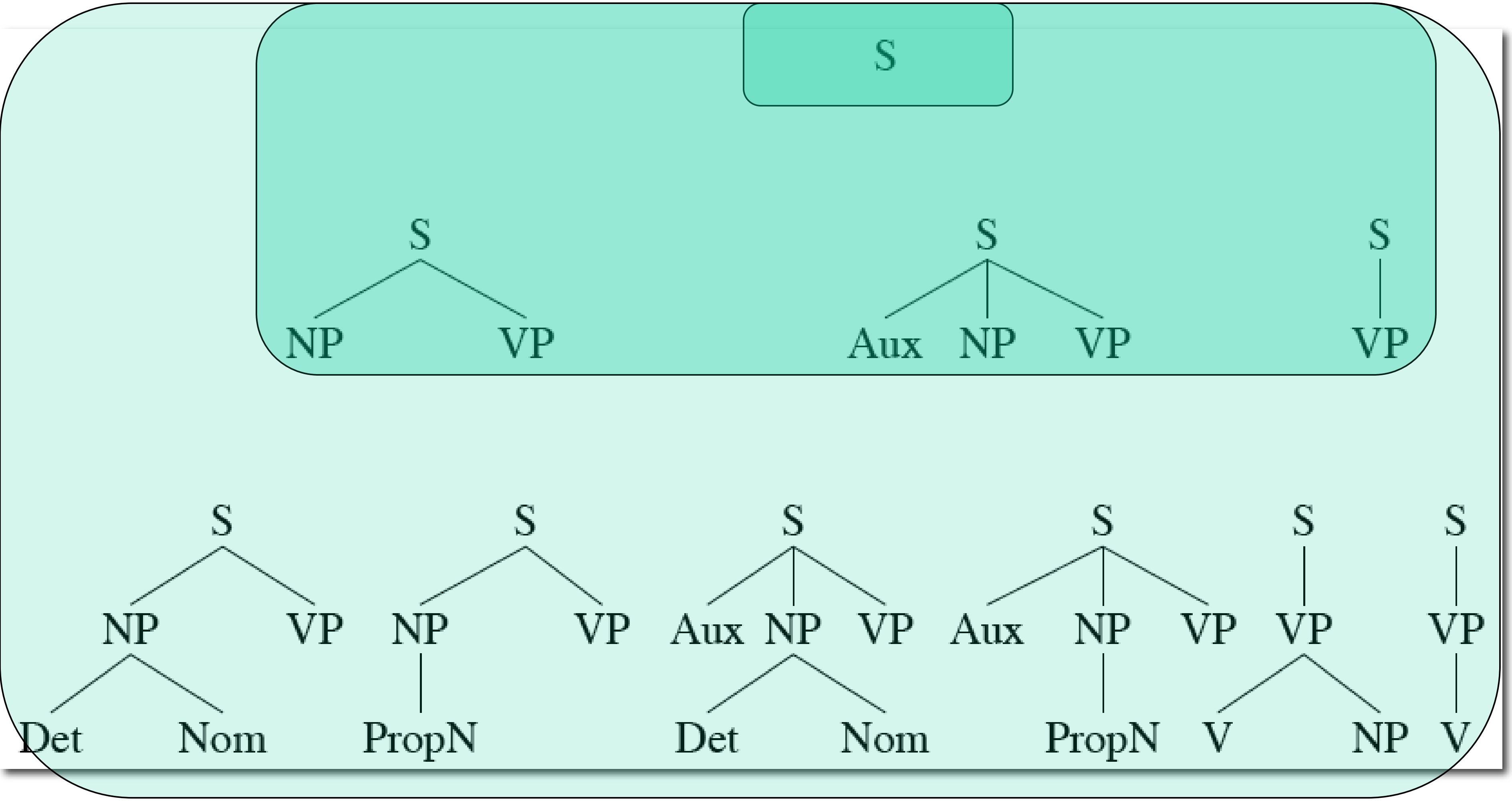
Parsing

- parsing involves a search

Top-Down Search

- We’re trying to find trees rooted with an *S;* start with the rules that give us an *S*.
- Then we can work our way down from there to the words.

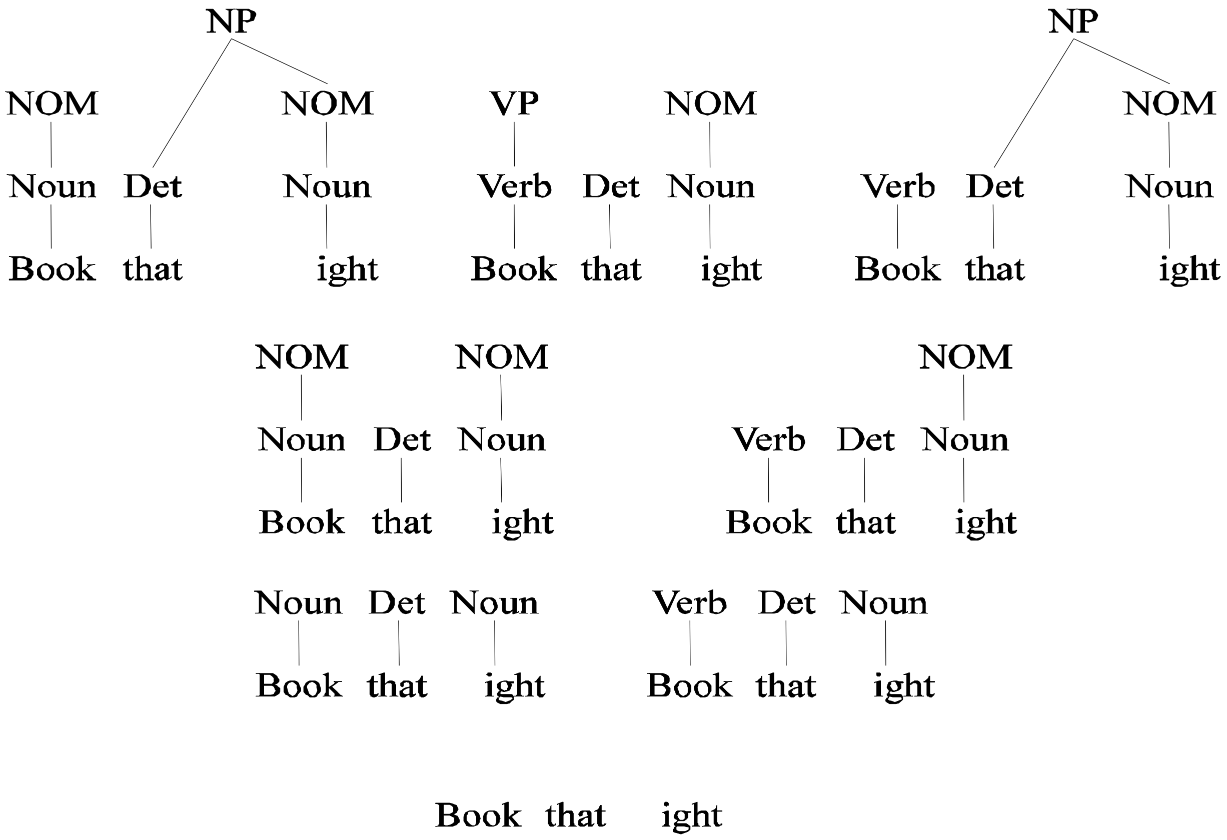
Top Down Space



Bottom-Up Parsing

- We also want trees that cover the input words.
- Start with trees that link up with the words
- Then work your way up from there to larger and larger trees.

Bottom-Up Space


NP
NOM NOM VP NOM NOM
Noun Det Noun Verb Det Noun Verb Det Noun
Book that flight Book that flight Book that flight
NP
NOM NOM NOM
Noun Det Noun Verb Det Noun
Book that flight Book that flight
Noun Det Noun Verb Det Noun
Book that flight Book that flight
Book that flight

Top-Down and Bottom-Up

- Top-down
  - Only searches for trees that can be S’s
  - But also suggests trees that are not consistent with any of the words
- Bottom-up
  - Only forms trees consistent with the words
  - But suggests trees that make no sense globally

Control

- Which node to try to expand next
- Which grammar rule to use to expand a node
- One approach: exhaustive search of the space of possibilities
- Not feasible
  - Time is exponential in the number of non-terminals
  - LOTS of repeated work, as the same constituent is created over and over (shared sub-problems)

Dynamic Programming

- DP search methods fill tables with partial results and thereby
  - Avoid doing avoidable repeated work
  - Solve exponential problems in polynomial time (well, no not really – we'll return to this point)
  - Efficiently store ambiguous structures with shared sub-parts.
- We'll cover two approaches that roughly correspond to bottom-up and top-down approaches.
  - CKY
  - Earley

CKY Parsing

- Consider the rule $A \rightarrow BC$
  - If there is an A somewhere in the input then there must be a B followed by a C in the input.
  - If the A spans from i to j in the input then there must be some k st. i<k<j
    - Ie. The B splits from the C someplace.

Convert Grammar to CNF

- What if your grammar isn't binary?
  - As in the case of the TreeBank grammar?
- Convert it to binary… any arbitrary CFG can be rewritten into Chomsky-Normal Form automatically.
  - The resulting grammar accepts (and rejects) the same set of strings as the original grammar.
  - But the resulting derivations (trees) are different.
  - We saw this in the last set of lecture notes

Convert Grammar to CNF

- More specifically, we want our rules to be of the form

  A → B C

  Or

  A → *w*

  That is, rules can expand to either 2 non-terminals or to a single terminal.

Binarization Intuition

- Introduce new intermediate non-terminals into the grammar that distribute rules with length > 2 over several rules.
  - So… $S \rightarrow A\ B\ C$ *turns into*

$S \rightarrow X\ C$ *and*

$X \rightarrow A\ B$

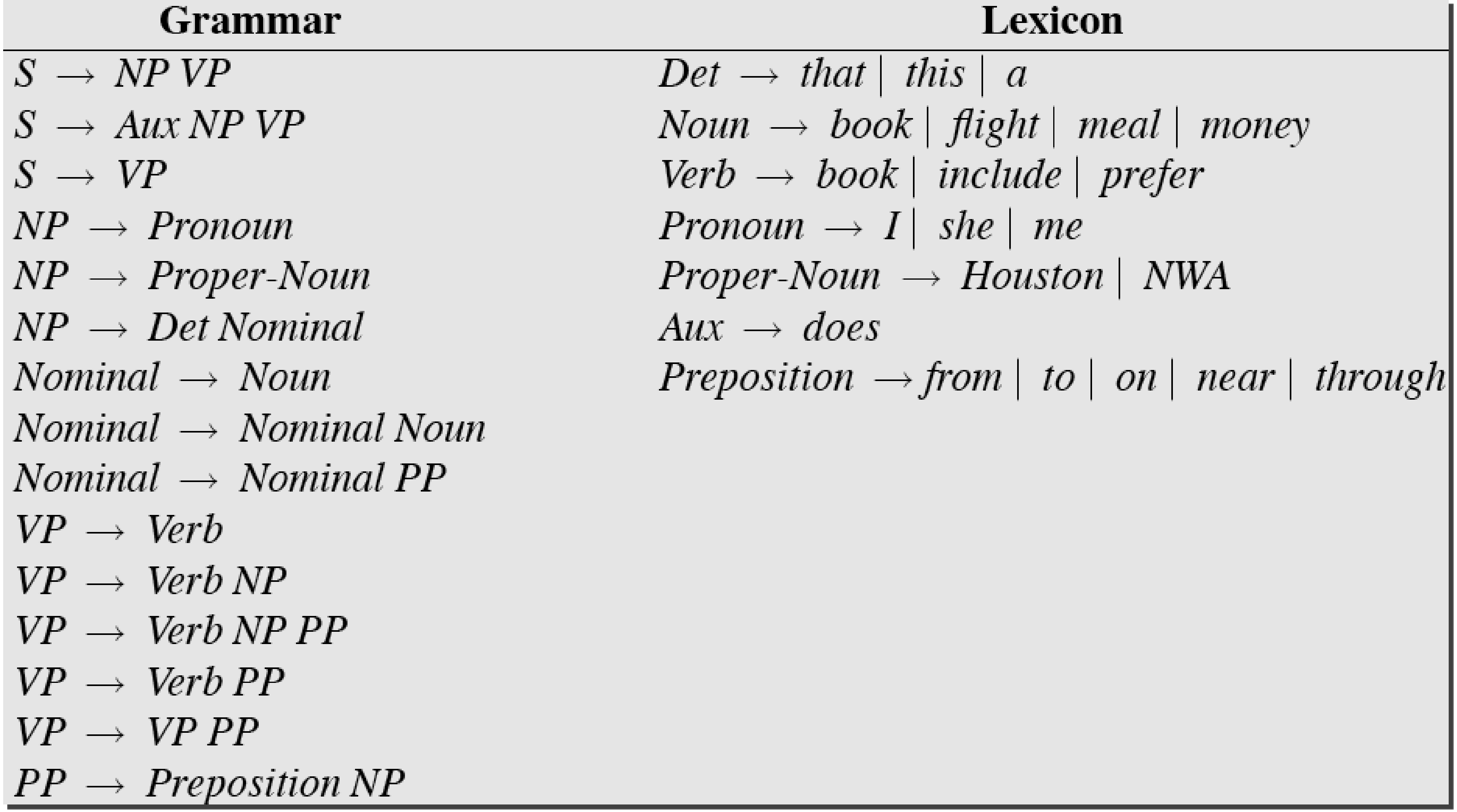
Where X is a symbol that doesn’t occur anywhere else in the the grammar.

Converting grammar to CNF

1. Copy all conforming rules to the new grammar unchanged
2. Convert terminals within rules to dummy non-terminals
3. Convert unit productions
4. Make all rules with NTs on the right binary

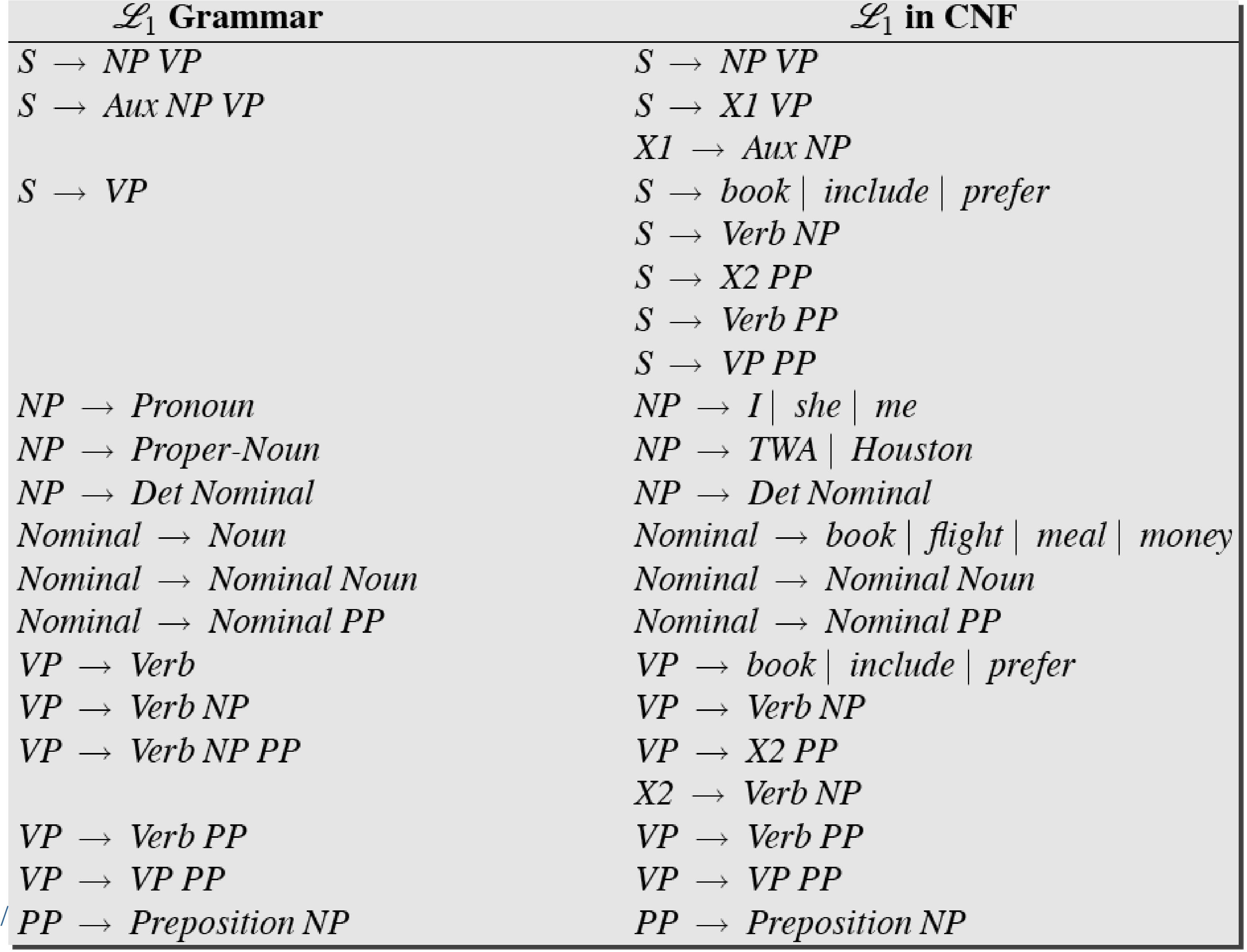
In lecture: what these mean; apply to example on next two slides

Sample L1 Grammar



Grammar	Lexicon
S → *NP VP*	*Det* → *that* | *this* | *a*
S → *Aux NP VP*	*Noun* → *book* | *flight* | *meal* | *money*
S → *VP*	*Verb* → *book* | *include* | *prefer*
NP → *Pronoun*	*Pronoun* → *I* | *she* | *me*
NP → *Proper-Noun*	*Proper-Noun* → *Houston* | *NWA*
NP → *Det Nominal*	*Aux* → *does*
Nominal → *Noun*	*Preposition* → *from* | *to* | *on* | *near* | *through*
Nominal → *Nominal Noun*	
Nominal → *Nominal PP*	
VP → *Verb*	
VP → *Verb NP*	
VP → *Verb NP PP*	
VP → *Verb PP*	
VP → *VP PP*	
PP → *Preposition NP*	

CNF Conversion



$\mathscr{L}_1$ **Grammar**	$\mathscr{L}_1$ **in CNF**
S → *NP VP*	*S* → *NP VP*
S → *Aux NP VP*	*S* → *X1 VP*
	X1 → *Aux NP*
S → *VP*	*S* → *book* | *include* | *prefer*
	S → *Verb NP*
	S → *X2 PP*
	S → *Verb PP*
	S → *VP PP*
NP → *Pronoun*	*NP* → *I* | *she* | *me*
NP → *Proper-Noun*	*NP* → *TWA* | *Houston*
NP → *Det Nominal*	*NP* → *Det Nominal*
Nominal → *Noun*	*Nominal* → *book* | *flight* | *meal* | *money*
Nominal → *Nominal Noun*	*Nominal* → *Nominal Noun*
Nominal → *Nominal PP*	*Nominal* → *Nominal PP*
VP → *Verb*	*VP* → *book* | *include* | *prefer*
VP → *Verb NP*	*VP* → *Verb NP*
VP → *Verb NP PP*	*VP* → *X2 PP*
	X2 → *Verb NP*
VP → *Verb PP*	*VP* → *Verb PP*
VP → *VP PP*	*VP* → *VP PP*
PP → *Preposition NP*	*PP* → *Preposition NP*

CKY

- Build a table so that an A spanning from i to j in the input is placed in cell [i,j] in the table.
- E.g., a non-terminal spanning an entire string will sit in cell [0, n]
  - Hopefully an *S*

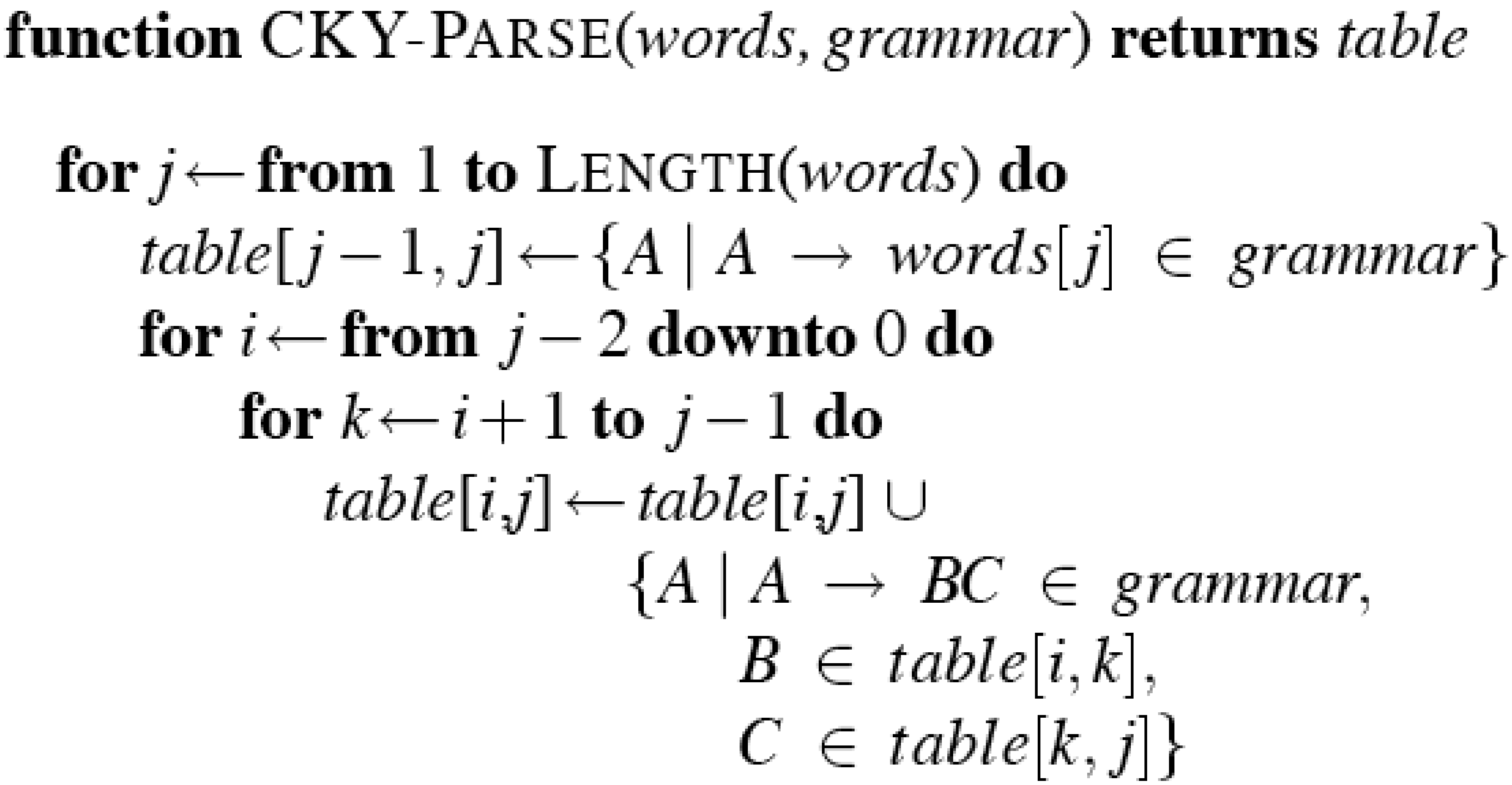
CKY

- If
  - there is an A spanning i,j in the input
  - A → B C is a rule in the grammar
- Then
  - There must be a B in [i,k] and a C in [k,j] for some i<k<j

CKY

- The loops to fill the table a column at a time, from left to right, bottom to top.
  - When we’re filling a cell, the parts needed to fill it are already in the table
    - to the left and below

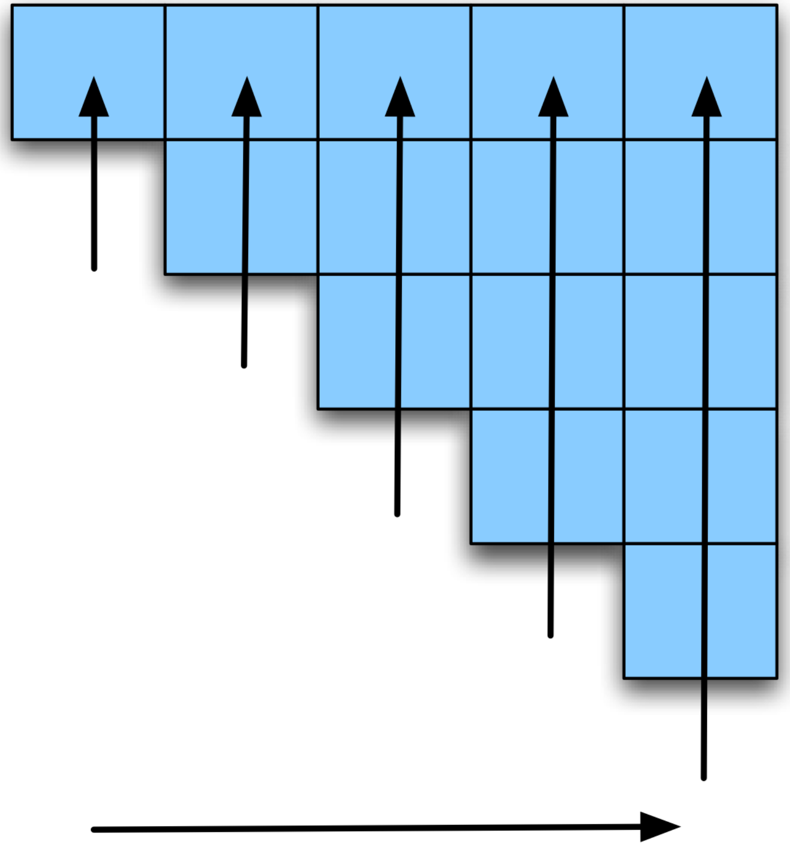
CKY Algorithm



```
function CKY-PARSE($words$, $grammar$) returns $table$

  for $j \leftarrow$ from 1 to LENGTH($words$) do
      $table[j-1, j] \leftarrow \{A \mid A \rightarrow words[j] \in grammar\}$
      for $i \leftarrow$ from $j-2$ downto 0 do
          for $k \leftarrow i+1$ to $j-1$ do
              $table[i,j] \leftarrow table[i,j] \cup$
                          $\{A \mid A \rightarrow BC \in grammar,$
                               $B \in table[i,k],$
                               $C \in table[k,j]\}$
```

Example

Book	the	flight	through	Houston
S, VP, Verb Nominal, Noun [0,1]	[0,2]	S,VP,X2 [0,3]	[0,4]	S,VP,X2 [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, Proper- Noun [4,5]



Go through full example in lecture

CKY Parsing

- Is that really a parser?
- So, far it is only a recognizer
  - Success? an S in cell [0,N]
- To turn it into a parser … see Lecture

CKY Notes

- Since it’s bottom up, CKY populates the table with a lot of worthless constituents.
  - To avoid this we can switch to a top-down control strategy
  - Or we can add some kind of filtering that blocks constituents where they can not happen in a final analysis.

Dynamic Programming Parsing Methods

- **CKY** (Cocke-Kasami-Younger) algorithm based on bottom-up parsing and requires first normalizing the grammar.
- **Earley parser** is based on top-down parsing and does not require normalizing grammar but is more complex.
- More generally, **chart parsers** retain completed phrases in a chart and can combine top-down and bottom-up search.

Conclusions

- Syntax parse trees specify the syntactic structure of a sentence that helps determine its meaning.
  - John ate the spaghetti with meatballs with chopsticks.
  - How did John eat the spaghetti?
    What did John eat?
- CFGs can be used to define the grammar of a natural language.
- Dynamic programming algorithms allow computing a single parse tree in cubic time or all parse trees in exponential time.

Chapter 14. Representing Meaning

From: Chapter 14 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

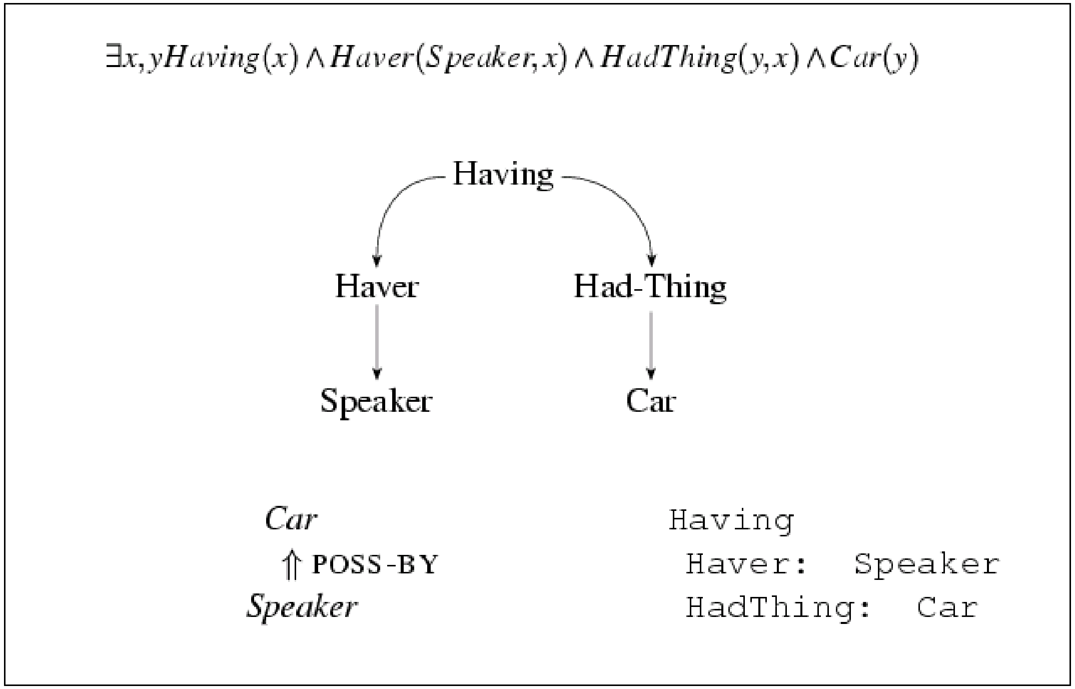
- Meaning representations
  - The meaning of linguistic utterances can be captured in formal structures
- Meaning representation languages
  - The frameworks that are used to specify the syntax and semantics of these representations
- Need for the representations
  - Bridging the gap from linguistic inputs to the kind of non-linguistic knowledge needed to perform a variety of tasks involving the meaning of linguistic inputs
- Language tasks requiring some form of semantic processing
  - Answering an essay question on an exam
  - Deciding what to order at a restaurant by reading a menu
  - Learning to use a new piece of software by reading the manual
  - Realizing that you’ve been insulted
  - Following a recipe

Background

- The tasks require access to representation that link the linguistic elements involved in the task to the non-linguistic *knowledge of the world* needed to successfully accomplish.
- Some of the knowledge of the world needed to perform the above tasks
  - Answering and grading essay questions requires background knowledge about the topic of the question, the desired knowledge level of the students, and how such questions are *normally* answered.
  - Reading a menu and deciding what to order, giving advice about where to go dinner, following a recipe, and generating new recipes all require deep knowledge about food, its preparation, what people like to eat and what restaurants are like.
  - Leaning to use a piece of software by reading a manual, or giving advice about how to do the same, requires deep knowledge about current computers, the specific software in question, similar software applications, and knowledge about users in general.

Background

- **Semantic analysis**
  - Taking linguistic inputs and constructing meaning representations that are made up of the *same kind stuff* that is used to represent this kind of everyday common sense knowledge of the world



$\exists x,y Having(x) \wedge Haver(Speaker,x) \wedge HadThing(y,x) \wedge Car(y)$

```
Having
 Haver:   Speaker
 HadThing:  Car
```

Background

- All above representations share a common foundation of the notion that a meaning representation consists of structures composed from a set of symbols.
  - These symbols structures correspond to objects, relations among objects, in some world being represented.

14.1 Compositional Desiderata for Representations

- Considering the issue of *why* meaning representations are needed and *what* they should do for us
  - Verifiability
  - Unambiguous representations
  - Canonical form
  - Inference and variables
  - Expressiveness

14.1 Compositional Desiderata for Representations

- Verifiability
  - It must be possible to use the representation to determine the relationship between the meaning of a sentence and the world we know it.
  - The most straightforward way
    - Compare, or match the representation of the meaning of an input against the representation in its **KB**, its store of information about its world.

(14.1) Does Maharani serve vegetarian food?
Serves(*Maharani*, *VegetarianFood*)

14.1 Compositional Desiderata for Representations

- Unambiguous representations
  - Single linguistic input can legitimately have different meaning representations assigned to them.
  - (14.2) *I wanna eat someplace that's close to ICSI*.
    - Ordinary interpretation
    - Godzilla's interpretation
  - Regardless of any ambiguity in the raw input, it is critical that a meaning representation language support representations that have a single unambiguity interpretation.
  - **Vagueness**, a concept closely related to ambiguity
    - (14.3) *I want to eat Italian food.*

14.1 Compositional Desiderata for Representations

- Canonical Form
  - Inputs that mean the same thing should have the same meaning representation
    - (14.4) *Does Maharani have vegetarian food*?
    - (14.5) *Do they have vegetarian food at Maharani*?
    - (14.6) *Are vegetarian dishes served at Maharani*?
    - (14.7) *Does Maharani serve vegetarian fare*?
  - Simplify various reasoning tasks
  - Complicate the task of semantic analysis
  - **Word sense** and **word sense disambiguation**
    - (14.8) Maharani serves vegetarian food.
    - (14.9) Vegetarian dishes are served by Maharani.

14.1 Compositional Desiderata for Representations

- Inference and Variables
  - **Inference**: refer generically to a system’s ability to draw valid conclusions based on the meaning representation of inputs and its store of background knowledge
    - (14.11) *I’d like to find a restaurant where I can get vegetarian food*.
    - *Serves*(*x*, *VegetarianFood*)

14.1 Compositional Desiderata for Representations

- Expressiveness
  - To be useful, a meaning representation scheme must be expressive enough to handle an extremely wide range of subject matter.
  - Ideal situation: having a single meaning representation language that could adequately represent the meaning of any sensible natural language utterance

14.2 Meaning Structure of Language

- Various methods by which human language convey meaning
  - Form-meaning associations,
  - Word-order regularities,
  - Tense systems,
  - Conjunction and quantifiers, and
  - A fundamental predicate argument structure
- The last one has great practical influence on the nature of meaning representation languages.

14.2 Meaning Structure of Language

Predicate-Argument Structure

- All human language have a form of predicate-argument arrangement at the core of their semantic structure.
- This predicate-argument structure asserts
  - The specific relationships hold among the various concepts underlying the constituent words and phrases that make up sentences
- This underlying structure permits the creation of a single composite meaning representation from the meanings of the various parts of an input
- One of the most important jobs of a grammar is to help organize this predicate-argument structure.

14.2 Meaning Structure of Language

Predicate-Argument Structure

(14.12) I want Italian food.		*NP want NP*
(14.13) I want to spend less than five dollars.	➡	*NP want Inf-VP*
(14.14) I want it to be close by here.		*NP want NP Inf-VP*

- The syntactic frames specify the number, position, and syntactic category of the arguments that are expected to accompany a verb.
- This kind of information is valuable in capturing a variety important facts about syntax.
- Two extensions of these frames into the semantic realm:
  - Semantic roles
  - Semantic restrictions on these roles

14.2 Meaning Structure of Language

Predicate-Argument Structure

- Notion of semantic role
  - By looking at (14.12) through (14.14)
    - The pre-verbal argument plays the role of the entity doing the *wanting*, while
    - The post-verbal argument plays the role of concept that is *wanted*.
  - By noticing these regularities and labeling them accordingly, we can associate *the surface arguments of a verb* with a set of *discrete roles in its underlying semantics*.
    - **Linking** of arguments in the surface structure with the semantic roles
  - The study of roes associated with specific verbs and across classes of verbs is referred to as **thematic role** or **case role** analysis.

14.2 Meaning Structure of Language

Predicate-Argument Structure

- Notion of semantic restrictions
  - Only certain kinds, or *categories*, of concepts can play the role of wanter — **selection restriction**
- Predicate-argument structures other than verbs
  - (14.15) an Italian restaurant under fifteen dollars

    Under(*ItalianRestaurant*, $15)
  - (14.16) Make a reservation for this evening for a table for two person at 8.

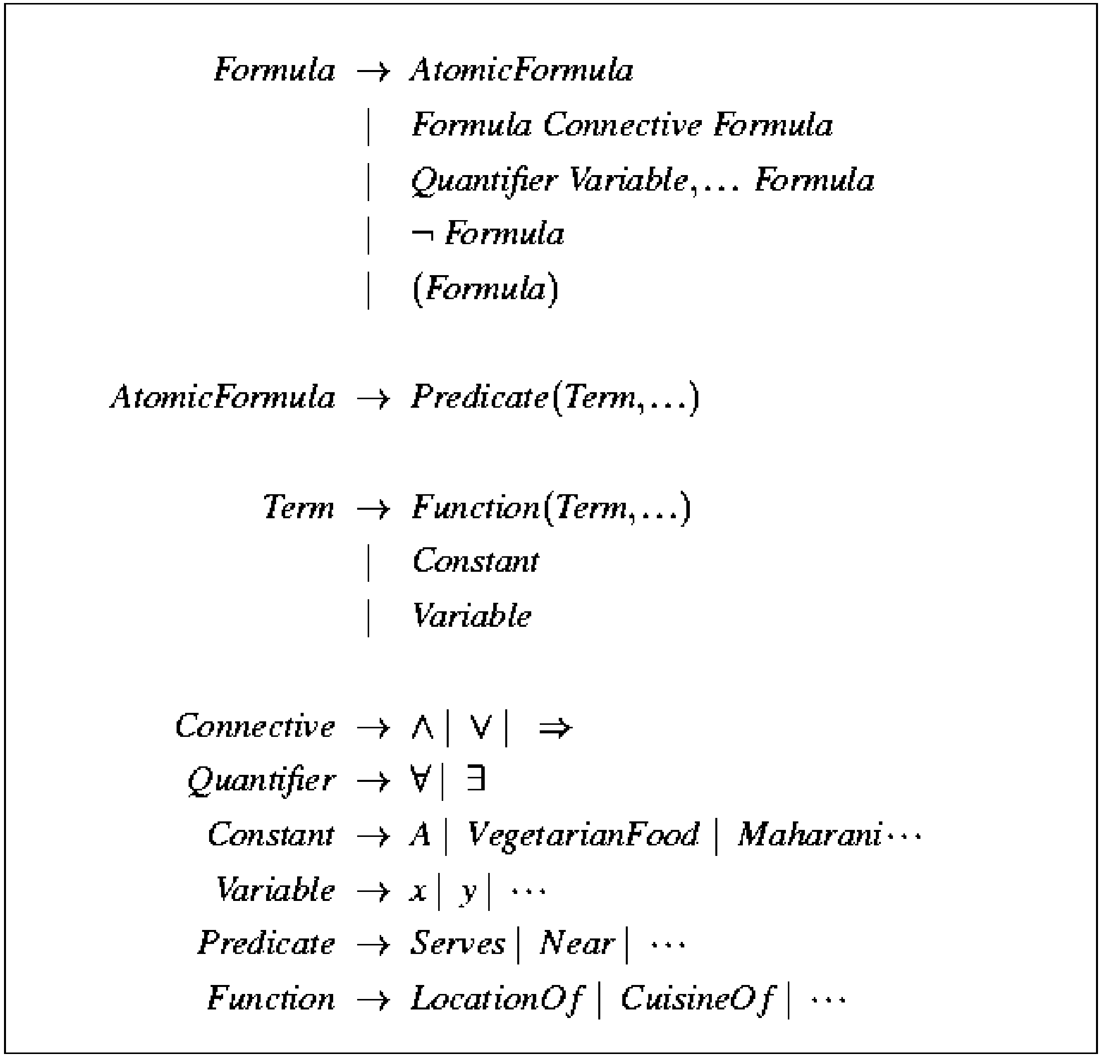
    Reservation(*Hearer*, *Today*, 8*PM*, *2*)
- Useful meaning representation language must support
  - Variable arity predicate-argument structures
  - The semantic labeling of arguments to predicates
  - The statement of semantic constraints on the fillers of argument roles

14.3 First Order Predicate Calculus

- FOPC is a flexible, well-understood, and computationally tractable approach to the representation of knowledge that satisfies many of the requirement raised previously for a meaning representation language.
- It provides a sound computational basis for the verifiability, inference, and expressiveness requirement.
- The most attractive feature of FOPC
  - It makes very few specific commitments as to how tings ought to be represented.

14.3 First Order Predicate Calculus

Elements of FOPC



$$
\begin{array}{rcl}
\textit{Formula} & \rightarrow & \textit{AtomicFormula} \\
 & | & \textit{Formula Connective Formula} \\
 & | & \textit{Quantifier Variable},\ldots\ \textit{Formula} \\
 & | & \neg\ \textit{Formula} \\
 & | & (\textit{Formula}) \\
 & & \\
\textit{AtomicFormula} & \rightarrow & \textit{Predicate}(\textit{Term},\ldots) \\
 & & \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term},\ldots) \\
 & | & \textit{Constant} \\
 & | & \textit{Variable} \\
 & & \\
\textit{Connective} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid \textit{VegetarianFood} \mid \textit{Maharani} \cdots \\
\textit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \cdots
\end{array}
$$

14.3 First Order Predicate Calculus

Elements of FOPC

- **Term**
  - **Constants**
    - Specific objects in the world being described
    - *A*, *B*, *Maharani*, *Harry*
  - **Functions**
    - Concepts often expressed in English as genitives,
      - *the location of Maharani*, or *Maharani's location*
      - *LocationOf*(*Maharani*)
    - Referring to unique objects, though appearing similarly as predicates
  - **Variables**
    - Objects without having to make references to any particular objects
    - Depicted as single lower-case letters

14.3 First Order Predicate Calculus

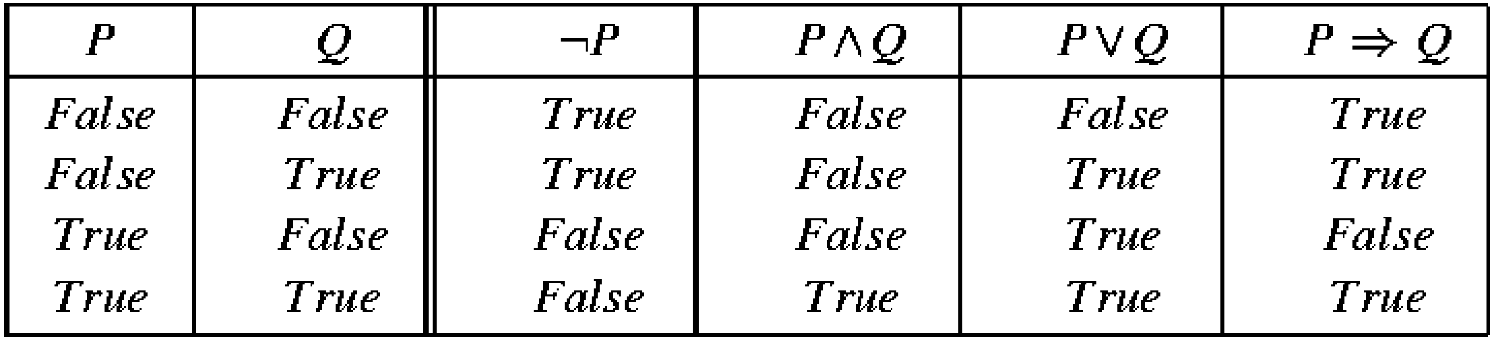
Elements of FOPC

- Relations hold among objects
  - Predicates are symbols refer to, or name, the relations that hold among some fixed number of objects in a given domain.
    - *Serves*(*Maharani*,*VegetarianFood*)
    - *Restaurant*(*Maharani*)
  - Atomic formula
  - Complex formula, through the use of **logical connectives**
    - *Have*(*Speaker*, *FiveDollars*) ∧ ¬*Have*(*Speaker*, *LotOfTime*)

14.3 First Order Predicate Calculus

The Semantics of FOPC

- The various objects, properties, and relations presented on a FOPC knowledge base acquire their meanings by virtue of their correspondence to objects, properties, and relations out in the external would being modeled by the knowledge base.
  - FOPC sentences can therefore be assigned a value of *True* or *False*
- The interpretations of formulas involving logical connectives is based on the meaning of the components in the formulas combined with the meaning of connectives they contain.



P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

14.3 First Order Predicate Calculus

Variables and Quantifiers

- Variables are used in FOPC
  - To refer to particular anonymous objects and
  - To refer generically to all objects in a collection
- The two uses are made possible through the use of **quantifiers**.
  - (14.19) a restaurant that serves Mexican food near ICSI

    $\exists x\ Restaurant(x) \land$

    $\quad Serves(x, MexicanFood) \land Near(LocationOf(x), LocationOf(ICSI))$

    $Restaurant(AyCaramba) \land$

    $\quad Serves(AyCaramba, MexicanFood) \land$

    $\quad Near(LocationOf(AyCaramba), LocationOf(ICSI))$
  - (14.20) All vegetarian restaurants serve vegetarian food.

    $\forall x\ VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

14.3 First Order Predicate Calculus

Inferences

- Inference
  - The ability to add valid new propositions to a knowledge base, or
  - To determine the truth of propositions not explicitly contained within a knowledge base.
- **Modus ponens**

$$\frac{\begin{array}{l}\alpha\\ \alpha \Rightarrow \beta\end{array}}{\beta}$$

$$\frac{\begin{array}{l}VegetarianRestaurant(Rudys)\\ \forall x\ VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)\end{array}}{Serves(Rudys, VegetarianFood)}$$

14.3 First Order Predicate Calculus

Inferences

- **Forward chaining** systems
  - **Production** systems
- **Backward chaining** systems
  - **Prolog** programming language

14.4 Some Linguistically Relevant Concepts

Categories

- Selectional restrictions expressed in the form of semantically-based categories
- The most common way to represent categories is to create a unary predicate for each category of interest.

  VegetarianRestaurant(*Mahrani*)
- Using this method, it is difficult to make assertions about categories themselves.

  MostPopular(*Maharani*, *VegetarianRestaurant*)
  - Not a legal FOPC formula since the arguments to predicates in FOPC must be *Terms*, not other predicates.
- Solution: **reification**

  ISA(*Maharani*, *VegetarianRestaurant*)

  AKO(*VegetarianRestaurant*, *Restaurant*)

14.4 Some Linguistically Relevant Concepts

Events

- So far,

  Reservation(*Heater*, *Maharani*, *Today*, *8PM*, *2*)

- Problems
  - Determining the correct number of roles for any given event
  - Representing facts about the roles associated with an event
  - Ensuring that all the correct inferences can be derived directly from the representation of an event
  - Ensuring that no incorrect inferences can be derived from the representation of an event

14.4 Some Linguistically Relevant Concepts

Events

(14.22) I ate.
(14.23) I ate a turkey sandwich.
(14.24) I ate a turkey sandwich at my desk.
(14.25) I ate at my desk.
(14.26) I ate lunch.
(14.27) I ate a turkey sandwich for lunch.
(14.28) I ate a turkey sandwich for lunch at my desk.

$Eating_1(Speaker)$
$Eating_2(Speaker, TurkeySanswich)$
$Eating_3(Speaker , TurkeySanswich, Desk)$
$Eating_4(Speaker, Desk)$
$Eating_5(Speaker, Lunch)$
$Eating_6(Speaker, , TurkeySanswich, Lunch)$
$Eating_7(Speaker , TurkeySanswich, Lunch, Desk)$

- Problem
  - High cost
- **Meaning postulates**

  $\forall w, x, y, z\ Eating_7(w, x, y, z) \Rightarrow Eating_6\ (w, x, y)$
  - Scalability problem

14.4 Some Linguistically Relevant Concepts

Events

$\exists w, x, y\ Eating_1(Speaker, w, x, y)$

$\exists\ w, x\ Eating(Speaker, TurkeySanswich, w, x)$

$\exists\ w\ Eating(Speaker\ , TurkeySanswich, w, Desk)$

$\exists\ w, x\ \ Eating(Speaker,w, x, Desk)$

$\exists\ w, x\ Eating(Speaker, w, Lunch, x)$

$\exists\ w\ Eating(Speaker, , TurkeySanswich, Lunch, w)$

$Eating(Speaker\ , TurkeySanswich, Lunch, Desk)$

- Deficiencies:
  - Too many commitments, and
  - Does not let us individuate events

14.4 Some Linguistically Relevant Concepts

Events

$\exists w, x\ Eating(Speaker, w, x, Desk)$……….(a)
$\exists\ w, x\ Eating(Speaker, w, Lunch, x)$……...(b)
$\exists\ w, x\ Eating(Speaker, w, Lunch, Desk)$….(c)

- Given the independent facts a and b, it does not conclude c, using the current representation.
- Employ reification to elevate events to objects that can be quantified and related to another objects via sets of defined relations.

$\exists w, x\ ISA(w, Eating) \wedge Eater(w, Speaker) \wedge Eatean(w, TurkeySandwich)$
$\exists w, x\ ISA(w, Eating) \wedge Eater(w, Speaker)$
$\exists w, x\ ISA(w, Eating)$
$\quad\quad \wedge Eater(w, Speaker) \wedge Eatean(w, TurkeySandwich)$
$\quad\quad \wedge MealEaten(w, Lunch)$

14.4 Some Linguistically Relevant Concepts

Representing Time

- The representation of time information in a useful form is the domain of **temporal logic**.
- The most straightforward theory of time hold that it flows inexorably forward, and that events are associated with either points or intervals in time, as on timeline.

14.4 Some Linguistically Relevant Concepts

Representing Time

(14.29) I arrived in New York.

$\exists\, i, e, w, t\ ISA(w, Arriving) \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork)$
$\wedge\ IntervalOf(w, i) \wedge EndPoint(i,e) \wedge Precedes(e, Now)$

(14.30) I am arriving in New York.

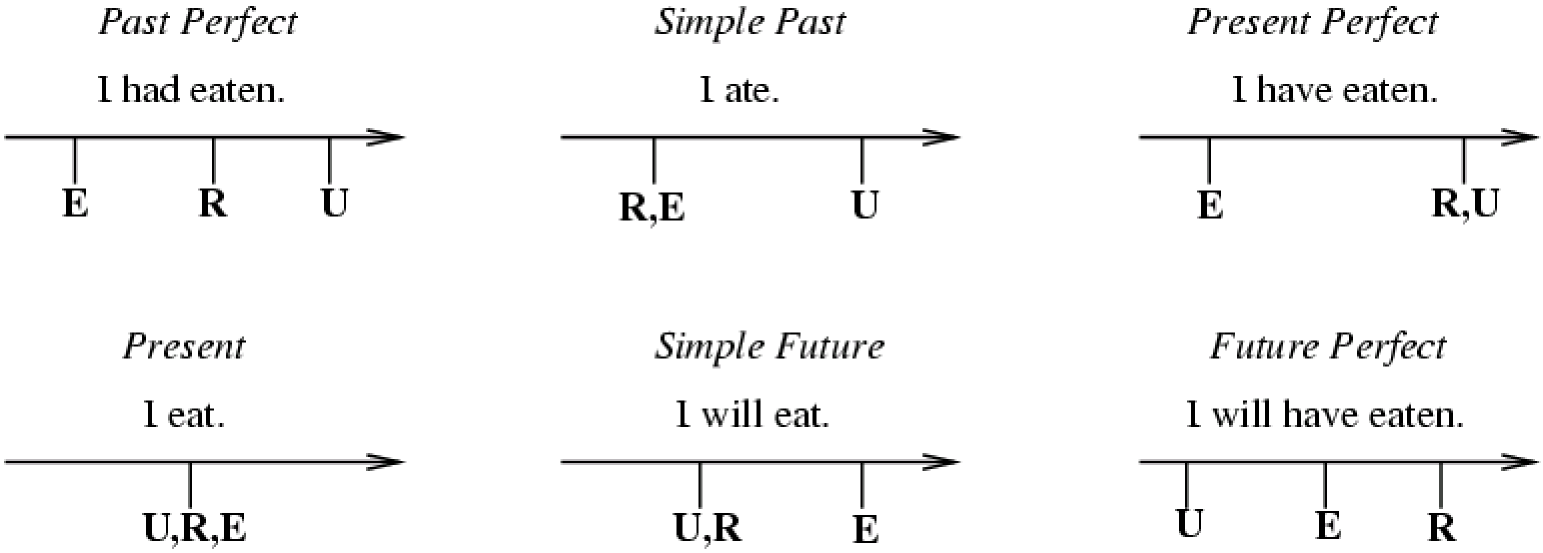
$\exists\, i, e, w, t\ ISA(w, Arriving) \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork)$
$\wedge\ IntervalOf(w, i) \wedge MemberOf(i, Now)$

(14.29) I will arrive in New York.

$\exists\, i, e, w, t\ ISA(w, Arriving) \wedge Arriver(w, Speaker) \wedge Destination(w, NewYork)$
$\wedge\ IntervalOf(w, i) \wedge EndPoint(i,e) \wedge Precedes(Now, e)$

14.4 Some Linguistically Relevant Concepts

Representing Time



14.4 Some Linguistically Relevant Concepts

Aspect

- **Aspect**
  - Concerning a cluster of related topics, including
    - whether an event has ended or is ongoing,
    - whether it is conceptualized as happening at a point in time or over some interval, and
    - Whether or not any particular state in the world comes about because of it.
- Four general classes
  - **Stative:** I know my departure time.
  - **Activity:** John is flying.
  - **Accomplishment:** Sally booked her flight.
  - **Achievement:** She found her gate.

14.4 Some Linguistically Relevant Concepts

Representing Beliefs

- Words and expressions for *world creating activity*
  - Their meaning representations contain logical formulas not intended to be taken as true in the real world, but rather as part of some kind of hypothetical world.
  - For example, *believe, want, imagine,* and *know*
  - (14.72) I believe that Mary ate British food.

$$\exists\, u, v\ ISA(u, Believing) \wedge ISA(\mathrm{v}, Eating) \wedge Believer(u, Speaker) \wedge BelievedProp(u, v) \wedge Eater(v, Mary) \wedge Eaten(v, BritishFood)$$

  - This results in a statement that there actually was an eating of British food by Mary.

14.4 Some Linguistically Relevant Concepts

Representing Beliefs

$Believing(Speaker, Eating(Mary, BritishFood))$

- Problem: It is not even valid FOPC.
- Solution
- Augment FOPC with *operator* that allow us to make statement about full logic formula.
  - Introduce an operator called *Believes* that takes two FOPC formulas as it arguments
    - A formula designating a believer, and
    - A formula designating the believed propositions.

$Believes(Speaker, \exists v\ ISA(\mathrm{v}, Eating) \land Eater(v, Mary) \land Eaten(v, BritishFood))$

Chapter 15. Semantic Analysis

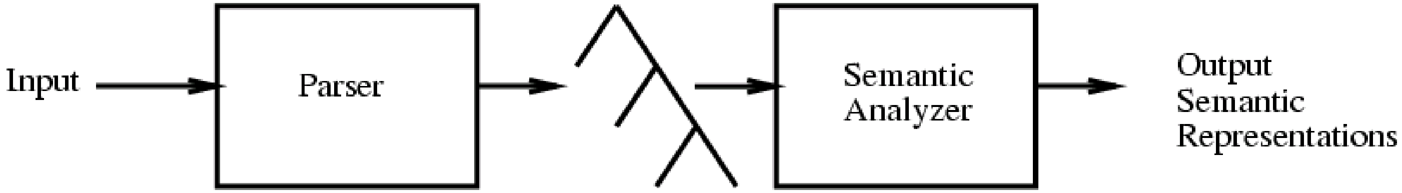
From: Chapter 15 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

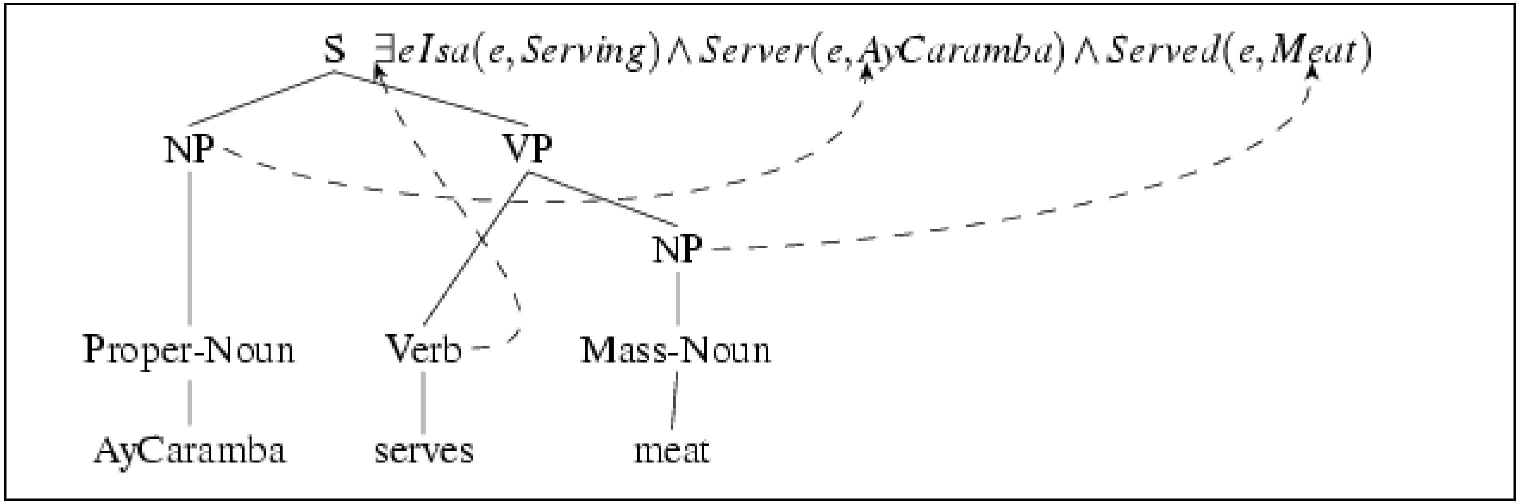
- **Semantic analysis**
  - The process whereby meaning representations are composed and assigned to linguistic inputs
- Among the source of knowledge typically used are
  - the meanings of words,
  - the meanings associated with grammatical structures,
  - knowledge about the structure of the discourse,
  - Knowledge about the context in which the discourse is occurring, and
  - Common-sense knowledge about the topic at hand
- **Syntax-driven semantic analysis**
  - Assigning meaning representations to input based solely on static knowledge from the lexicon and the grammar.

15.1 Syntax-Driven Semantic Analysis

- **Principle of compositionality**
  - The meaning of a sentence can be composed from the meanings of its parts.
- In syntax-driven semantic analysis, the composition of meaning representations is guided by the *syntactic components* and *relations* provided by the grammars discussed previously.
- Assumption:
  - Do not resolve the ambiguities arising from the previous stages.



15.1 Syntax-Driven Semantic Analysis


S
$\exists e\, Isa(e, Serving) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$
NP
VP
NP
Proper-Noun
Verb
Mass-Noun
AyCaramba
serves
meat

- What does it mean for syntactic constituents to have meaning?
- What do these meanings have to be like so that they can be composed into larger meanings?

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- Augmenting CFG rules with **semantic attachment**

  $A \rightarrow \alpha_1 \ldots \alpha_n \{f(\alpha_j.sem, \ldots \alpha_k.sem)\}$

- Walk through the semantic attachment with the example just shown

  PropoerNoun → *AyCaramba* {*AyCaramba*}

  MassNoun → *meat* {*meat*}

  NP → *ProperNoun* {*ProperNoun.sem*}

  NP → *MassNoun* {*MassNoun.sem*}

  $Verb \rightarrow serves\ \{\exists e, x, y\ Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, y)\}$

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- To move *Verb* upwards to *VP*, the *VP* semantic attachment must have two capabilities:
  - The means to *know* exactly which variables within the *Verb*'s semantic attachment are to be replaced by the semantics of the *Verb*'s arguments, and
  - The ability to perform such a replacement.
- FOPC does not provide any advice about when and how such things are done.
  - **Lambda notation** provides exactly the kind of formal parameter functionality we needed.

    $\lambda x P(x)$

    x: variable

    $P(x)$: FOPC expression using the variable x

15.1 Syntax-Driven Semantic Analysis
Semantic Augmentation to CFG Rules

$\lambda x P(x)(\mathrm{A})$ (λ-reduction)

$\Rightarrow P(\mathrm{A})$

$\lambda x \lambda y Near(x, y)$

$\lambda x \lambda y Near(x, y)(ICSI)$

$\Rightarrow \lambda y Near(ICSI, y)$

$\lambda y Near(ICSI, y)(AyCaramba)$

$\Rightarrow Near(ICSI, AyCaramba)$

- **Currying**
  - A way of converting a predicate with multiple arguments into a sequence of single argument predicates

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- With λ-notation

  $Verb \rightarrow serves\ \{\lambda x\ \exists e, y\ Isa(e, Serving) \wedge Server(e, y) \wedge Served(e, x)\}$

  $VP \rightarrow Verb\ NP\ \{Verb.sem(NP.sem)\}$

  $\Rightarrow \exists e, y\ Isa(e, Serving) \wedge Server(e, y) \wedge Served(e, Meat)$

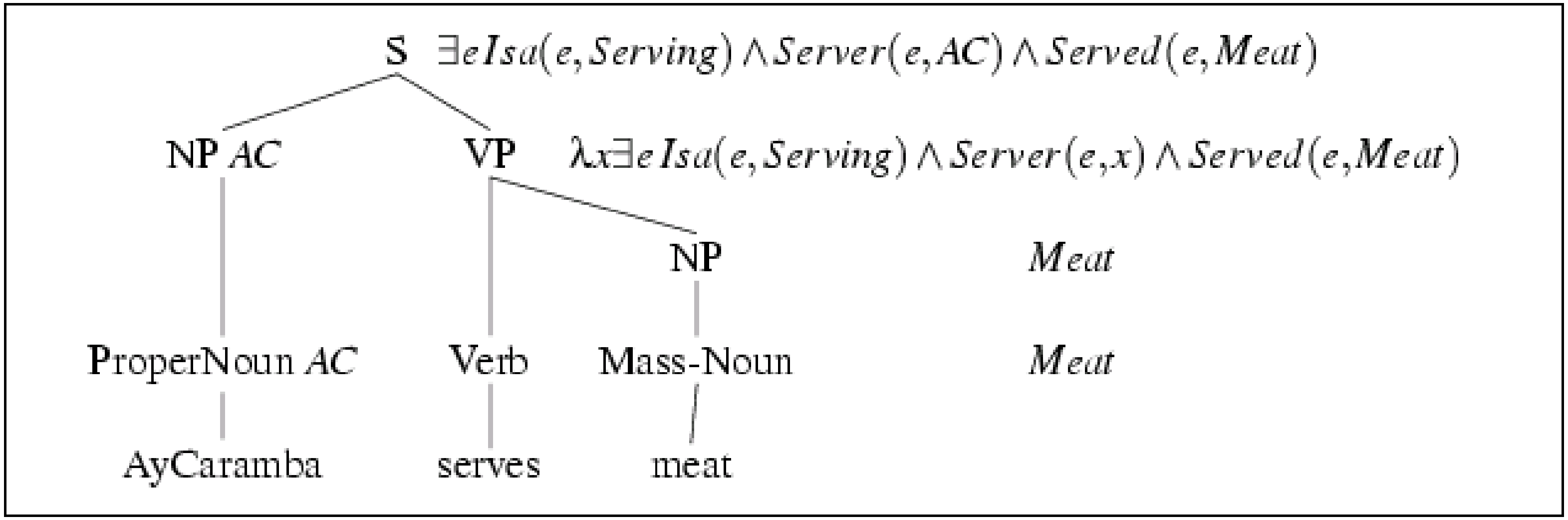
  $S \rightarrow NP\ VP\ \{VP.sem(NP.sem)\}$

  – Revise *Verb* attachment

  $Verb \rightarrow serves\ \{\lambda x\ \lambda y\ \exists e\ Isa(e, Serving) \wedge Server(e, y) \wedge Served(e, x)\}$

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules


S $\exists e\, Isa(e, Serving) \wedge Server(e, AC) \wedge Served(e, Meat)$
NP AC
VP $\lambda x \exists e\, Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, Meat)$
NP
$Meat$
ProperNoun AC
Verb
Mass-Noun
$Meat$
AyCaramba
serves
meat

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- (15.1) A restaurant serves meat.
  - Subject

    $\exists x\ Isa(x, Restaurant)$
  - Embed in the *Server* predicate:

    $\exists e\ Isa(e, Serving) \wedge Server(e, \mathbf{\exists x\ Isa(x, Restaurant)}) \wedge Served(e, Meat)$

    Not a valid FOPC
- Solve this problem by introducing the notion of a **complex-term**.
  - A complex term: $<$ *Quantifier variable body* $>$

    $\exists e\ Isa(e, Serving) \wedge Server(e, \mathbf{<\exists x\ Isa(x, Restaurant>})) \wedge Served(e, Meat)$
  - Rewriting a predicate using a complex-term

    $P(<Quantifier\ variable\ body>)$

    $\Rightarrow$

    $Quantifier\ variable\ body\ Connective\ P(variable)$

$Server(e,<\exists x\ Isa(x, Restaurant>)$

$\Rightarrow$

$\exists x\ Isa(x, Restaurant) \wedge Server(e,x)$

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- The connective that is used to attach the extracted formula to the front of the new expression depends on the type of the quantifier being used:
  - $\wedge$ is used with $\exists$, and $\Rightarrow$ is used with $\forall$.
- It is useful to access the three components of complex terms.

  NP → *Det Nominal* {<*Det.sem x Nominal.sem*(*x*)>}

  Det → *a* {∃}

  Nominal → *Noun* {λ *x Isa*(*x*, *Noun.sem*)>}

  Noun → *restaurant* {*Restaurant*}

15.1 Syntax-Driven Semantic Analysis

Semantic Augmentation to CFG Rules

- We have introduced five concrete mechanisms that instantiate the abstract functional characterization of semantic attachments
  1. The association of normal FOPC expressions with lexical items
  2. The association of function-like λ-expressions with lexical items
  3. The copying of semantic values from children to parents
  4. The function-like application of λ-expressions to the semantics of one or more children of a constituent
  5. The use of complex-terms to allow quantified expressions to be temporarily treated as terms
- **Quasi-logical forms** or **intermediate representations**
  – The use of λ-expressions and complex-terms motivated by the gap between the syntax of FOPC and the syntax of English

15.1 Syntax-Driven Semantic Analysis

Quantifier Scoping and the Translation of Complex-Terms

- (15.3) Every restaurant has a menu.

  $\exists e\ Isa(e, Having)$
  $\quad \wedge\ Haver(e, <\forall x\ Isa(x, Restaurant)>)$
  $\quad \wedge\ Had(e, <\exists y\ Isa(y, Menu)>)$

  $\forall x\ Restaurant(x) \Rightarrow$
  $\quad \exists e, y \wedge Having(e) \wedge Haver(e, x) \wedge Isa(y, Menu) \wedge Had(e, y)$

  $\exists y\ Isa(y, Menu) \wedge \forall x\ Isa(x, Restaurant) \Rightarrow$
  $\quad \exists e\ Having(e) \wedge Haver(e, x) \wedge Had(e, y)$

- The problem of ambguous **quantifier scoping** — a single logical formula with two complex-terms give rise to two distinct and incompatible FOC representations.

15.2 Attachments for a Fragment of English

Sentences

- (15.4) Flight 487 serves lunch.

  $S \rightarrow NP\ VP\ \{DCL(VP.sem(NP.sem))\}$

- (15.5) Serve lunch.

  $S \rightarrow VP\ \{IMP(VP.sem(DummyYou))$

  $IMP(\exists e Serving(e) \wedge Server(e, DummyYou) \wedge Served(e, Lunch)$

  Imperatives can be viewed as a kind of **speech act**.

- (15.6) Does Flight 207 serve lunch?

  $S \rightarrow Aux\ NP\ VP\ \{YNQ(VP.sem(NP.sem))\}$

  $YNQ(\exists e Serving(e) \wedge Server(e, Flt207) \wedge Served(e, Lunch)$

- (15.7) Which flights serve lunch?

  $S \rightarrow WhWord\ NP\ VP\ \{WHQ(NP.sem.var, VP.sem(NP.sem))\}$

  $WHQ(x, \exists e, x\ Isa(e, Serving) \wedge Server(e, x) \wedge Served(e, Lunch) \wedge Isa(x, Flight))$

15.2 Attachments for a Fragment of English

Sentences

- (15.8) How can I go from Minneapolis to Long Beach?

  $S \rightarrow WhWord\ Aux\ NP\ VP\ \{WHQ(WhWord.sem, VP.sem(NP.sem))\}$

  $WHQ(How, \exists e\ Isa(e, Going) \wedge Goer(e, User)$

  $\qquad \wedge Origin(e, Minn) \wedge Destination(e, LongBeach))$

15.2 Attachments for a Fragment of English

NPs: Compound Nominals

- The meaning representations for NPs can be either normal FOPC terms or complex-terms.
- (15.9) Flight schedule
- (15.10) Summer flight schedule

$Nominal \rightarrow Noun$

$Nominal \rightarrow Nominal\ Noun\ \{\lambda x\ Nominal.sem(x) \wedge NN(Noun.sem,x)\}$

$\lambda x\ Isa(x, Schedule) \wedge NN(x, Flight)$

$\lambda x\ Isa(x, Schedule) \wedge NN(x, Flight) \wedge NN(x, Summer)$

15.2 Attachments for a Fragment of English

NP: Genitive NPs

- (Ex.) Atlanta's airport
- (Ex.) Maharani's menu

$NP \rightarrow ComplexDet\ Nominal$

$\{<\exists x Nominal.sem(x) \wedge GN(x, ComplexDet.sem)>\}$

$ComplexDet \rightarrow NP's\ \{NP.sem\}$

$<\exists x\ Isa(x, Airport) \wedge GN(x, Atlanta)>$

15.2 Attachments for a Fragment of English

Adjective Phrases

- (15.11) I don't mind a cheap restaurant.
- (15.12) This restaurant is cheap.
- For pre-nominal case, an obvious and *often incorrect* proposal is:

  $Nominal \rightarrow Adj\ Nominal$

  $\{\lambda x\ Nominal.sem(x) \wedge Isa(x, Adj.sem)\}$

  $Adj \rightarrow cheap\ \{Cheap\}$

  $\lambda x\ Isa(x, Restaurant) \wedge Isa(x, Cheap)$ **intersective semantics**
- ***Wrong***
  - *small elephant* $\Rightarrow \lambda x\ Isa(x, Elephant) \wedge Isa(x, Small)$
  - *former friend* $\Rightarrow \lambda x\ Isa(x, Friend) \wedge Isa(x, Former)$
  - *fake gun* $\Rightarrow \lambda x\ Isa(x, Gun) \wedge Isa(x, Fake)$

Incorrect interactions

15.2 Attachments for a Fragment of English
Adjective Phrases

- The best approach is to simply note the status of a special kind of modification relation and
  - Assume that some further procedure with access to additional relevant knowledge can replace this vague relation with an appropriate representation.

$Nominal \rightarrow Adj\ Nominal$

$\{\lambda x\ Nominal.sem(x) \wedge AM(x, Adj.sem)\}$

$Adj \rightarrow cheap\ \{Cheap\}$

$\lambda x\ Isa(x, Restaurant) \wedge AM(x, Cheap)$

15.2 Attachments for a Fragment of English
VPs: Infinite VPs

- In general, the λ-expression attached to the verb is simply applied to the semantic attachments of the verb's arguments.
- However, some special cases, for example, infinite VP
- (15.13) I told Harry to go to Maharani.
  - The meaning representation could be:

    $\exists e, f, x\ Isa(e, Telling) \wedge Isa(f, Going)$

    $\wedge Teller(e, Speaker) \wedge Tellee(e, Harry) \wedge ToldThing(e, f)$

    $\wedge Goer(f, Harry) \wedge Destination(f, x)$
  - Two things to note:
    - It consists of two events, and
    - One of the participants, *Harry*, plays a role in both of the two events.

15.2 Attachments for a Fragment of English

VPs: Infinite VPs

- A way to represent the semantic attachment of the verb, *tell*

  $\lambda x, y \lambda z \exists e\ Isa(e, Telling) \wedge Teller(e, z) \wedge Tell(e, x) \wedge ToldThing(e, y)$

  – Providing three semantic roles:

    - a person doing the telling,
    - a recipient of the telling, and
    - the proposition being conveyed

  – **Problem:**

    - Harry is not available when the *Going* event is created within the infinite verb phrase.

15.2 Attachments for a Fragment of English

VPs: Infinite VPs

- Solution:

  $VP \rightarrow Verb\ \ NP\ \ VPto \quad \{Verb.sem(NP.sem, VPto.sem)\}$

  $VPto \rightarrow to\ VP\ Verb\ \ NP\ \{VP.sem\}$

  $Verb \rightarrow tell \quad \{\lambda x, y\ \lambda z$

  $\exists e, y.variable\ Isa(e, Telling) \wedge Teller(e, z) \wedge Tellee(e,x)$

  $\wedge ToldThing(e, y.variable) \wedge y(x)\}$

  - The λ-variable *x* plays the role of the *Tellee* of the telling and the argument to the semantics of the infinitive, which is now contained as a λ-expression in the variable *y*.
  - The expression *y*(*x*) represents a λ-reduction that inserts *Harry* into the *Going* event as the *Goer*.
  - The notation *y.variable* is analogous to the notation used for complex-terms variables, and gives us access to the event variable representing *Going* event within the infinitive's meaning representation.

15.2 Attachments for a Fragment of English

Prepositional Phrases

- At an abstract level, PPs serve two functions:
  - They assert binary relations between their heads and the constituents to which they attached, and
  - They signal arguments to constituents that have an argument structure.
- We will consider three places in the grammar where PP serve these roles:
  - Modifiers of NPs
  - Modifiers of VPs, and
  - Arguments to VPs

15.2 Attachments for a Fragment of English
PP: Nominal Modifier

- (15.14) A restaurant on Pearl

  $\exists x\ Isa(x, Restaurant) \wedge On(x, Pearl)$

  $NP \rightarrow Det\ Nominal$

  $Nominal \rightarrow Nominal\ PP \quad \{\lambda z\ Nominal.sem(z) \wedge PP.sem(z)\}$

  $PP \rightarrow P\ NP \quad \{P.sem(NP.sem)\}$

  $P \rightarrow on \quad \{\lambda y\ \lambda x\ On(x,y)\}$

15.2 Attachments for a Fragment of English

PP: VP Modifier

- (Ex.) ate dinner in a hurry

  $VP \rightarrow VP\ PP$

  - The meaning representation of *ate dinner*

    $\lambda x \exists e\ Isa(e, Eating) \wedge Eater(e, x) \wedge Eaten(e, Dinner)$

  - The representation of the PP

    $\lambda x\ In(x, <\exists h\ Hurry(h)>)$

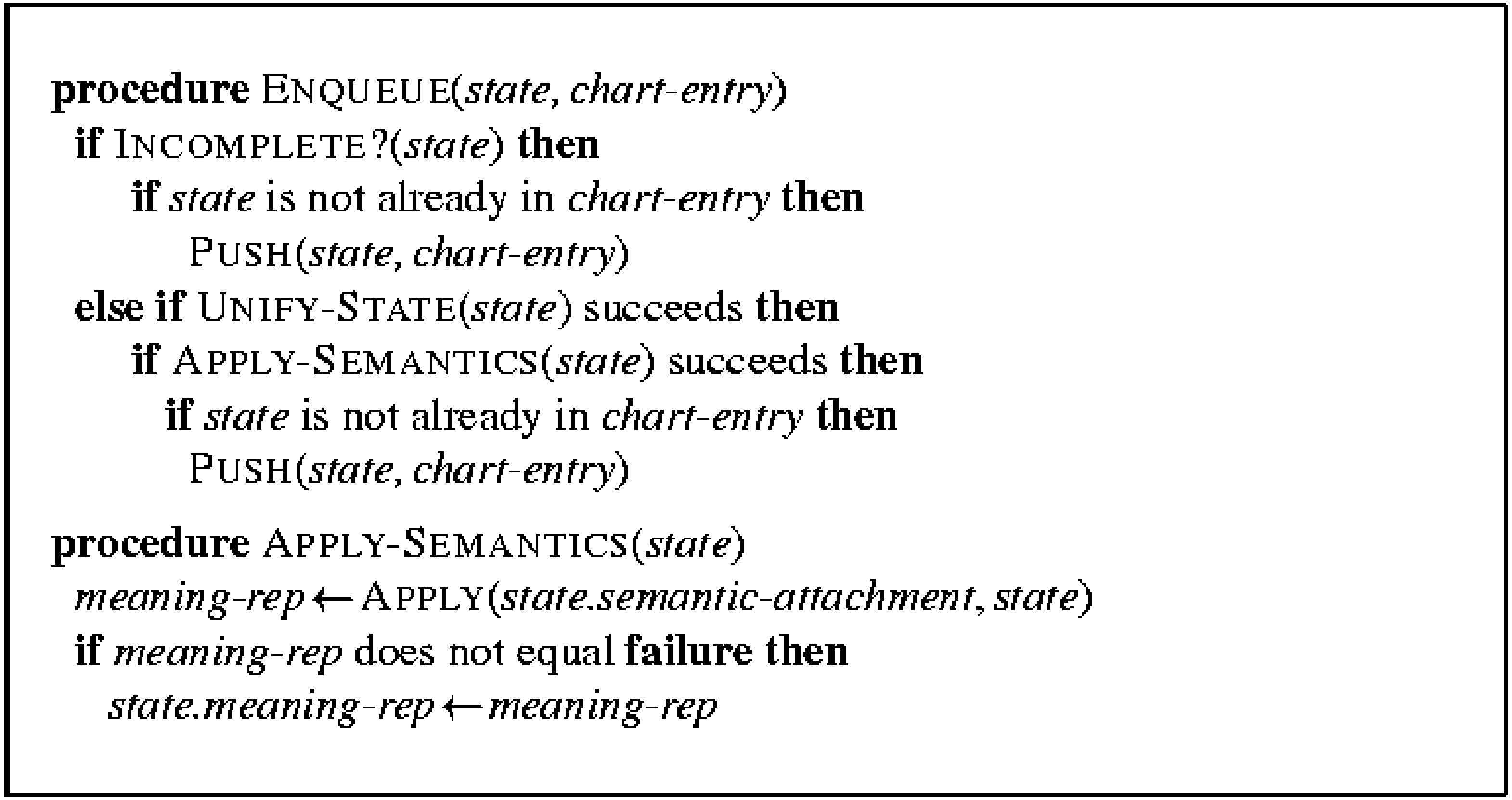
  - The correct representation of the modified VP should contain the conjunction of the two
    - With the *Eating* event variable filling the first argument slot of the *In* expression.

    $VP \rightarrow VP\ PP\ \{\lambda y\ VP.sem(y) \wedge PP.sem(VP.sem.variable)\}$

  - The result of application

    $\lambda y \exists e\ Isa(e, Eating) \wedge Eater(e, y) \wedge Eaten(e, Dinner) \wedge In(e\ , <\exists h\ Hurry(h)>)$

15.3 Integrating Semantic Analysis into the Earley Parser



```
procedure ENQUEUE(state, chart-entry)
  if INCOMPLETE?(state) then
      if state is not already in chart-entry then
          PUSH(state, chart-entry)
  else if UNIFY-STATE(state) succeeds then
      if APPLY-SEMANTICS(state) succeeds then
         if state is not already in chart-entry then
          PUSH(state, chart-entry)

procedure APPLY-SEMANTICS(state)
  meaning-rep ← APPLY(state.semantic-attachment, state)
  if meaning-rep does not equal failure then
     state.meaning-rep ← meaning-rep
```

15.4 Idioms and Compositionality

- (15.16) Coupons are just ***the tip of the iceberg***.
- (15.17) The SEC's allegations are only ***the tip of the iceberg***.
- (15.18) Coronary bypass surgery, hip replacement and intensive-care units are ***the tip of the iceberg***.

  NP → *the tip of the iceberg* {*Beginning*}

  NP → *TipNP of the IcebergNP* {*Beginning*}
- Handling idioms require at least the following changes to the general composition framework:
  - Allow the mixing of lexical items with traditional grammatical constituents.
  - Allow the creation of additional idiom-specific constituents needed to handle the correct range of productivity of the idioms.
  - Permit semantic attachments that introduce logical terms and predicates that are not related to any of the constituents of the rule.

15.5 Robust Semantic Analysis
Information Extraction

- The task of IE about
  - joint venture from business news,
  - understanding weather reports, or
  - summarizing simple information about what happened today on the stock market from a radio report,

  do not require detailed understanding.
- Tasks of IE are characterized by two properties:
  - The desire knowledge can be described by a relatively simple and fixed **template**, or frame, with slots that need to be filled in with material from the text, and
  - Only a small part of the information in the text is relevant for filling in this frame; the rest can be ignored.

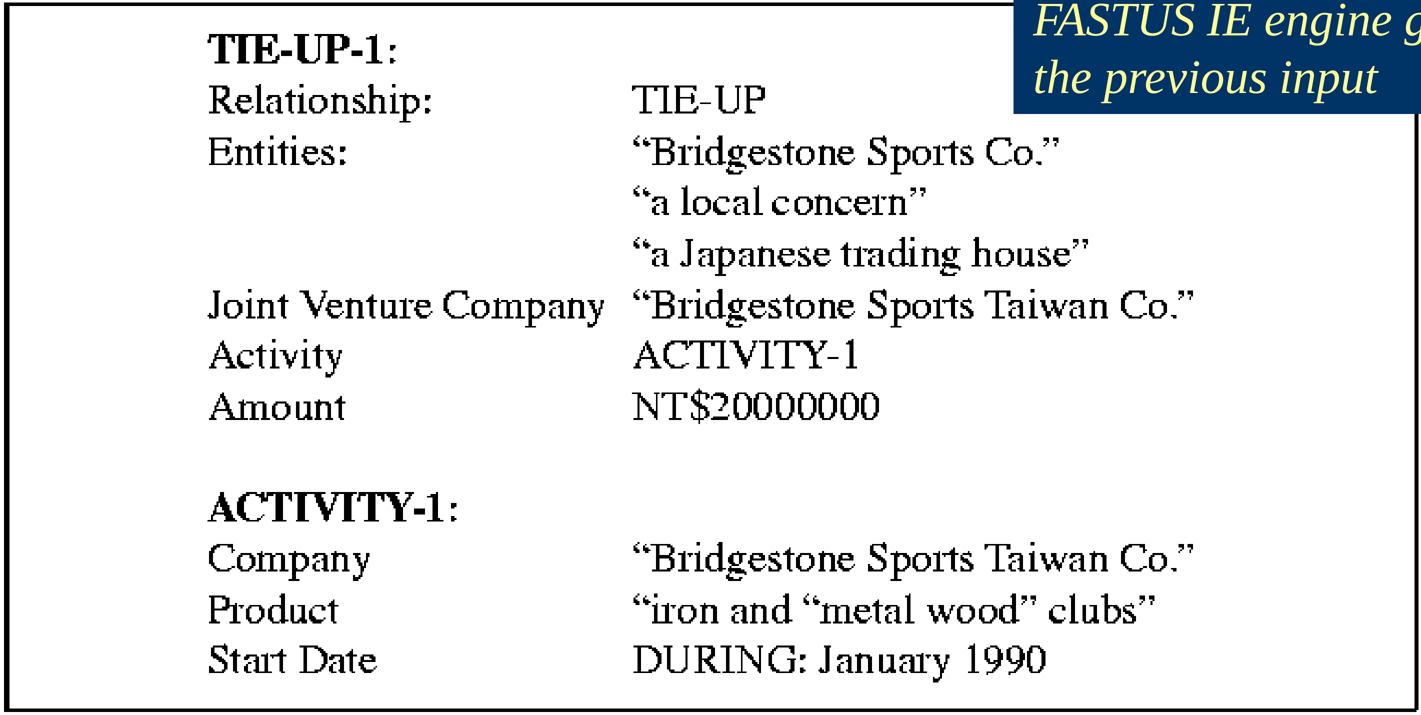
15.5 Robust Semantic Analysis
Information Extraction

- *Message Understanding Conference, MUC*-5 (1993)
  - A US Government-organized IE conference
  - To extract information about international joint venture from business news
  - Sample article

    Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.
    The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990 with production of 20,000 iron and "metal wood" clubs a month.

  - The output of an IE system can be a single template with a certain number of slots filled in, or a more complex hierarchically related set of objects.

15.5 Robust Semantic Analysis
Information Extraction

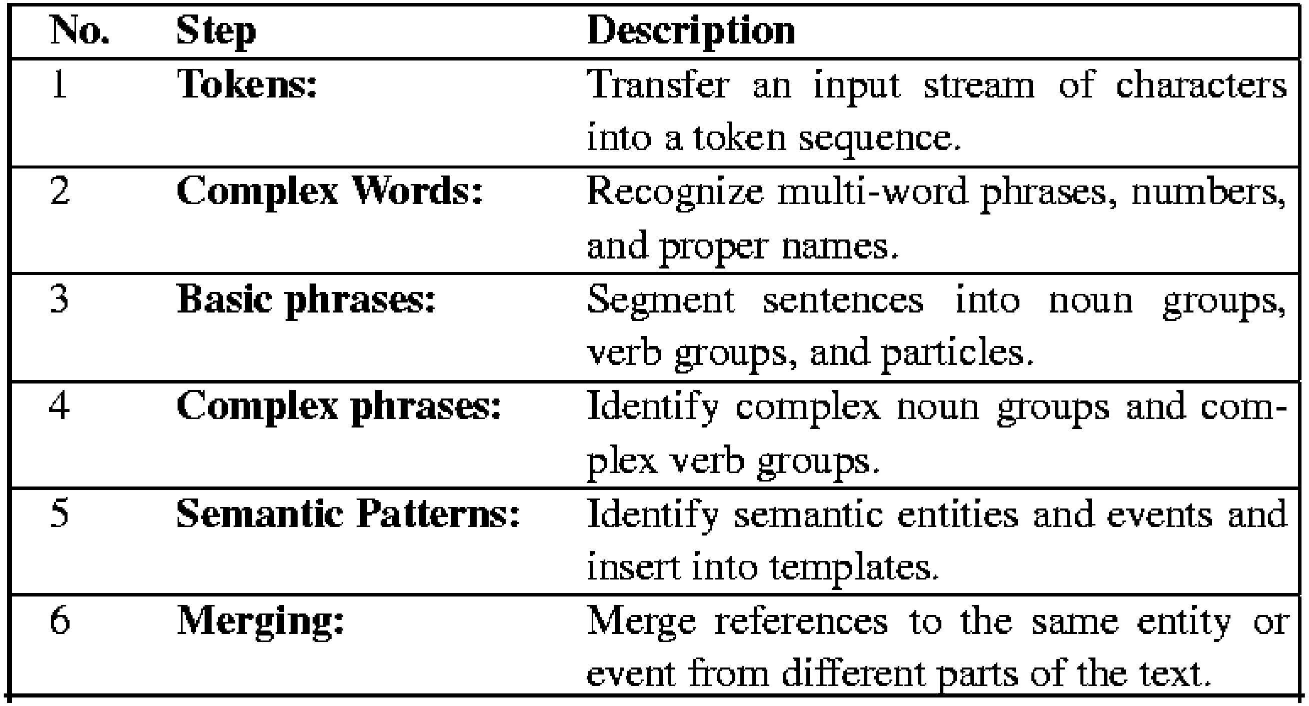


TIE-UP-1:	
Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co." "a local concern" "a Japanese trading house"
Joint Venture Company	"Bridgestone Sports Taiwan Co."
Activity	ACTIVITY-1
Amount	NT$20000000
ACTIVITY-1:	
Company	"Bridgestone Sports Taiwan Co."
Product	"iron and "metal wood" clubs"
Start Date	DURING: January 1990

The templates produced by the FASTUS IE engine given the previous input

15.5 Robust Semantic Analysis
Information Extraction

- Many IE systems are built around **cascades** of FSA, for example FASTUS.



No.	Step	Description
1	**Tokens:**	Transfer an input stream of characters into a token sequence.
2	**Complex Words:**	Recognize multi-word phrases, numbers, and proper names.
3	**Basic phrases:**	Segment sentences into noun groups, verb groups, and particles.
4	**Complex phrases:**	Identify complex noun groups and complex verb groups.
5	**Semantic Patterns:**	Identify semantic entities and events and insert into templates.
6	**Merging:**	Merge references to the same entity or event from different parts of the text.

15.5 Robust Semantic Analysis
Information Extraction

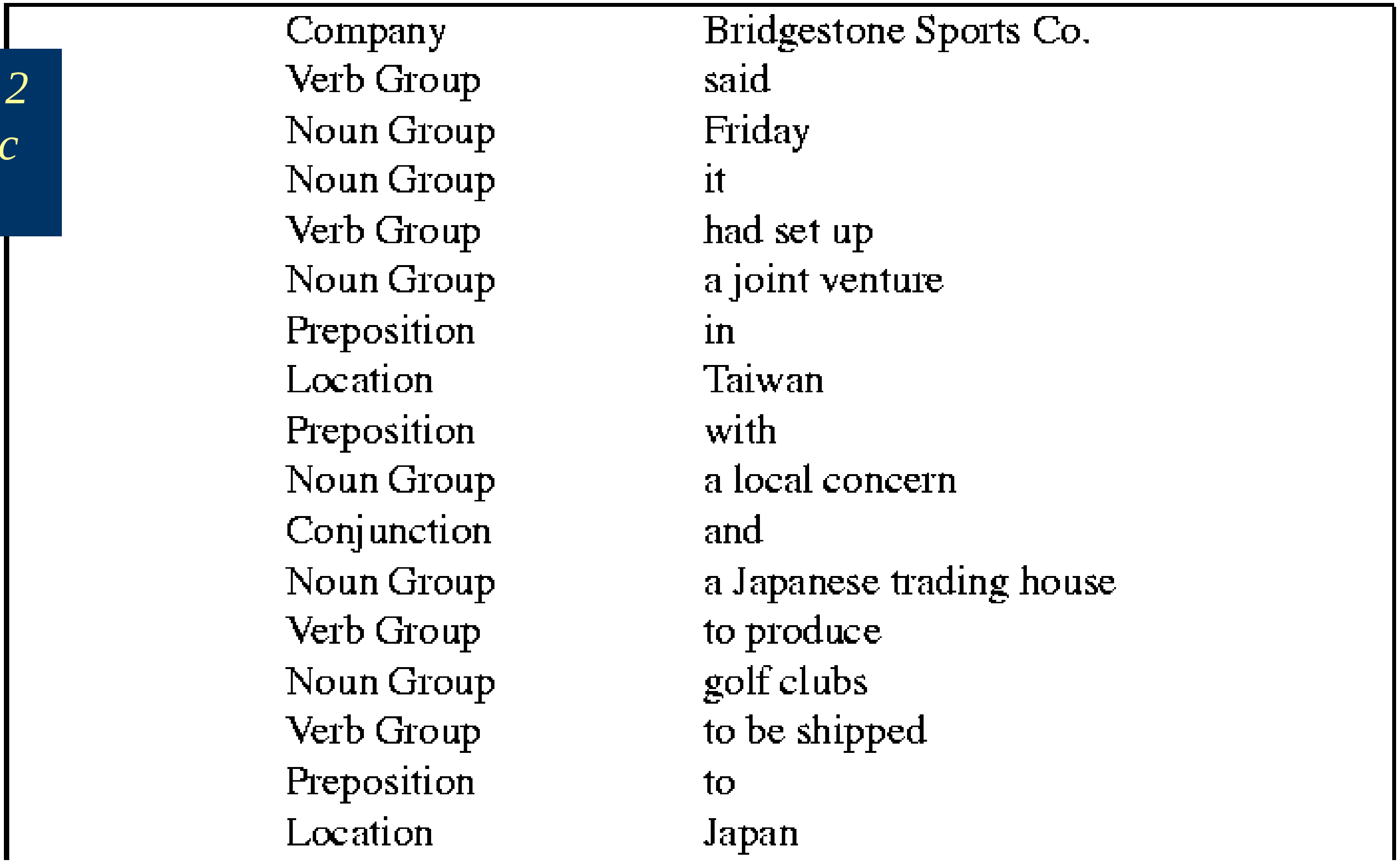
- After tokenization, in FASTUS, the second level recognizes
  - multiwords like, *set up*, and *joint venture*, and
  - names like *Bridgestone Sports Co.*
- The name recognizer is a transducer, composed of a large set of specific mapping designed to handle
  - Locations,
  - Personal names, and
  - Names of organizations, companies, unions, performing groups, etc.
- Typical rules

Performer-Org → (pre-location) Performer-Noun+ Perf-Org-Suffix
pre-location → locname | nationality
locname → city | region
Perf-Org-Suffix → orchestra, company
Performer-Noun → symphony, opera
nationality → Canadian, American, Mexican
city → San Francisco, London

San Francisco Symphony Orchestra
Canadian Opera Company

15.5 Robust Semantic Analysis
Information Extraction

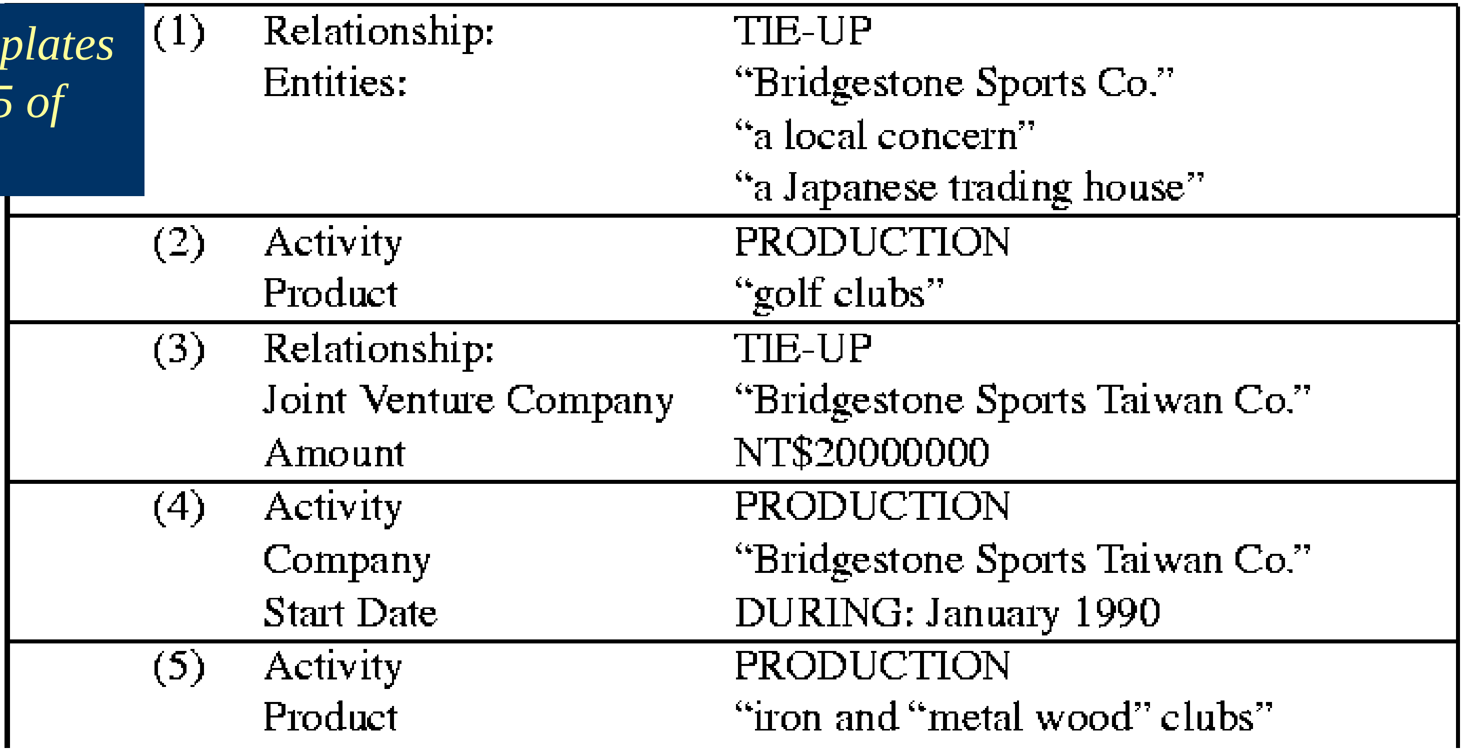
The output of stage 2 of the FASTUS basic phrase extractor



Company	Bridgestone Sports Co.
Verb Group	said
Noun Group	Friday
Noun Group	it
Verb Group	had set up
Noun Group	a joint venture
Preposition	in
Location	Taiwan
Preposition	with
Noun Group	a local concern
Conjunction	and
Noun Group	a Japanese trading house
Verb Group	to produce
Noun Group	golf clubs
Verb Group	to be shipped
Preposition	to
Location	Japan

15.5 Robust Semantic Analysis
Information Extraction

The five partial templates produced by Stage 5 of the FASTUS system



(1)	Relationship:	TIE-UP
	Entities:	“Bridgestone Sports Co.” “a local concern” “a Japanese trading house”
(2)	Activity	PRODUCTION
	Product	“golf clubs”
(3)	Relationship:	TIE-UP
	Joint Venture Company	“Bridgestone Sports Taiwan Co.”
	Amount	NT$20000000
(4)	Activity	PRODUCTION
	Company	“Bridgestone Sports Taiwan Co.”
	Start Date	DURING: January 1990
(5)	Activity	PRODUCTION
	Product	“iron and “metal wood” clubs”

15.5 Robust Semantic Analysis

Chapter 17. Lexical Semantics

From: Chapter 17 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- We made minimal use of the notion of the meaning of a word previously.
- Words and their meanings provided the appropriate bits and pieces necessary to construct adequate meaning representations for entire sentences.
- While words may contribute content to the meanings of sentences, they do not themselves have meanings.
    - Words do not refer to the world, cannot be judged to be true or false, or literal or figurative, or …
- This narrow conception of the role of words in a semantic theory leads to a view of *the lexicon as a simple listing of symbolic fragments devoid of any systematic structure*.

Background

- In this chapter,
  - the lexicon has a highly systematic structure governing what words can mean, and
  - how they can be used.
- This structure consists of
  - Relations among words and their meanings, as well as
  - The internal structure of individual words.
- **Lexical semantics**

Background

- **Lexeme**
  - An individual entry in the lexicon
  - Be thought of as a pairing of particular orthographic and phonological form with some form of symbolic meaning representation
- A **lexicon** = a finite list of lexemes
- Dictionaries are fruitful places to start with.

right *adj*. located near the right hand esp. being on the right when facing the same direction as the observer.
left *adj*. located nearer to this side of the body than the right.
red *adj*. the color of blood or a ruby.
blood *adj*. the red liquid that circulates in the heart, arteries and veins of animal

- Circularity in them
  - Dictionaries entries are often not definitions at all, but rather descriptions of lexemes in terms of other lexemes.

Background

- The approach of dictionaries will fail without some ultimate grounding in the external world.
- Fortunately, there are still wealth of semantic information contained in this kind of definition.
  - *Left* and *right* are similar kinds of lexemes, standing in some kind of alternation, or opposition, to one another.
  - *Red* is a color, it can be applied to both *blood* and *rubies*, and *blood* is a *liquid*.
- Given a sufficiently large database of facts like these, many applications are quite capable of performing sophisticated semantic tasks (even if they do not *really* know their right from their left).

Background

- To summarize:
  - We can capture quite a bit about the semantics of individual lexemes by analyzing and labeling their relations to other lexemes in various settings.
  - We will be interested in accounting for the similarities and differences among different lexemes in similar settings and
  - The nature of relation among lexemes in a single setting (internal structures of lexemes, 16.3).

16.1 Relations among Lexemes and their Senses

- **Homonymy**
  - A relation that holds *between words that have the same form with unrelated meanings.*

  (16.1) Instead, a *bank* can hold the investments in a custodial account in the client’s name.

  (16.2) But as agriculture burgeons on the east *bank*, the river will shrink even more.

16.1 Relations among Lexemes and their Senses

- **Polysemy**
  - Multiple related meanings within a single lexeme

(16.7) While some *banks* furbish sperm only to married women, other are much less restrictive.

  - *Blood bank, egg blank, sperm bank*

16.1 Relations among Lexemes and their Senses

- **Synonym**
  - *Different lexemes with the same meaning*

  (16.14) How big is that plane?

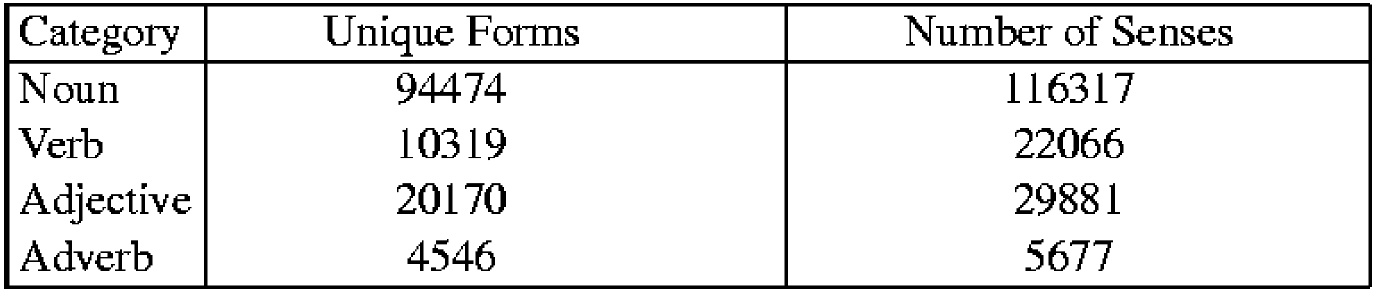
  (16.15) Would I be flying on a large or small plane?

16.1 Relations among Lexemes and their Senses

- **Hyponym**
  - *Pairings where one lexeme denotes a subclass of the other*
  - The more specific lexeme as a **hyponym** of the more general one, and the more general term as a **hypernym** of the more specific one
  - *Car* is a hyponym of *vehicle*, and *vehicle* is hypernym of *car*.

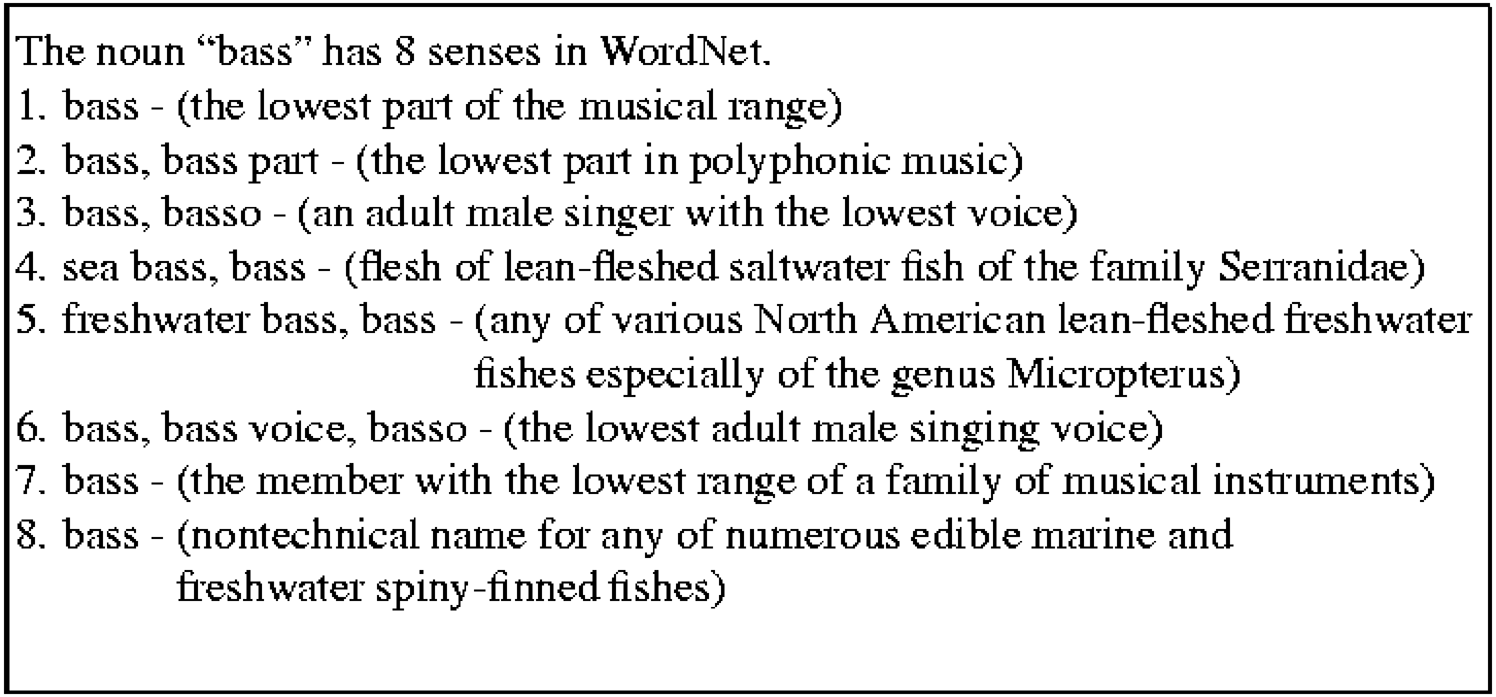
16.2 WORDNET: A Database of Lexical Relations

- WordNet:
  - The most well-developed and widely used lexical DB for English
  - Handcrafting from scratch, rather than mining information from existing dictionaries and thesauri
  - Consisting three separate DBs:
    - One each for nouns and verbs, and
    - A third for adjectives and adverbs



Category	Unique Forms	Number of Senses
Noun	94474	116317
Verb	10319	22066
Adjective	20170	29881
Adverb	4546	5677

16.2 WORDNET: A Database of Lexical Relations

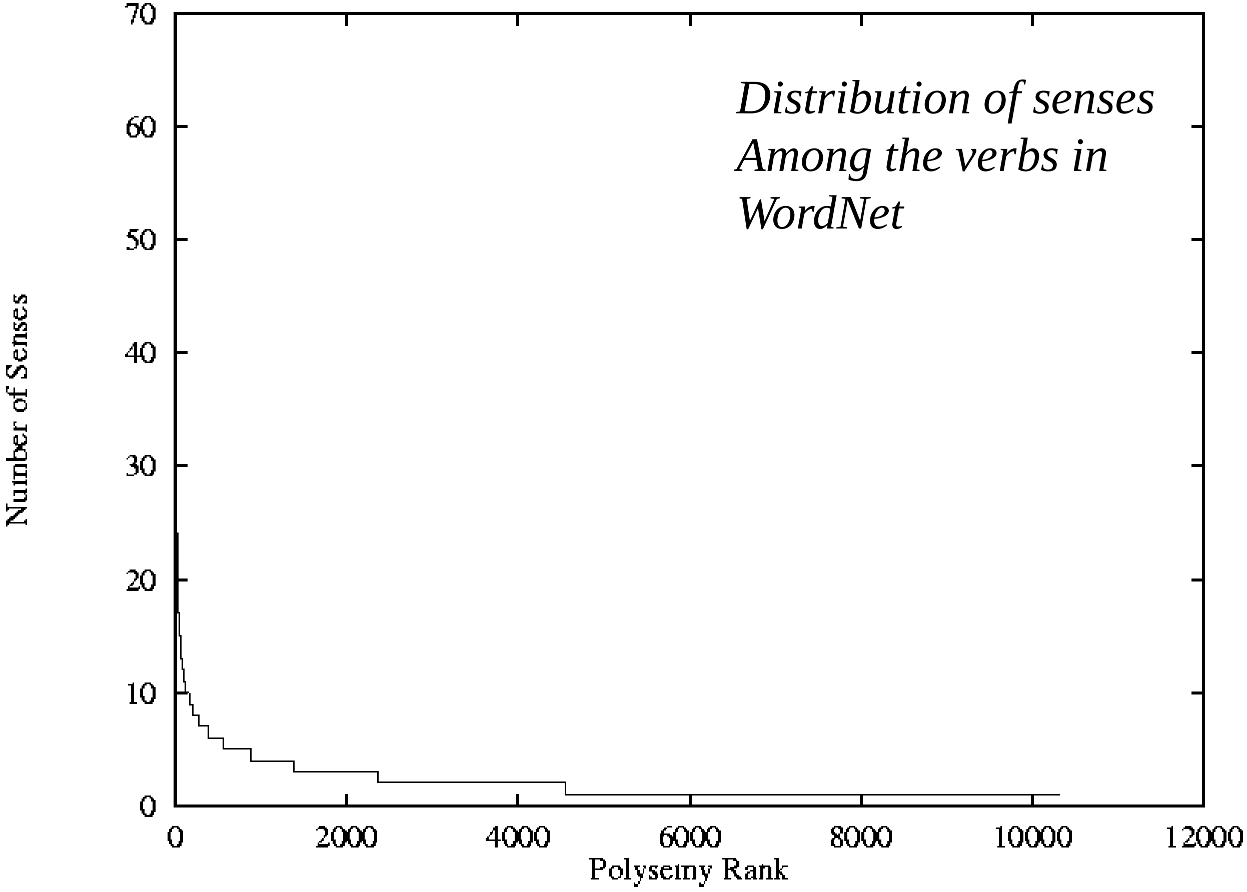


The noun “bass” has 8 senses in WordNet.

1. bass - (the lowest part of the musical range)
2. bass, bass part - (the lowest part in polyphonic music)
3. bass, basso - (an adult male singer with the lowest voice)
4. sea bass, bass - (flesh of lean-fleshed saltwater fish of the family Serranidae)
5. freshwater bass, bass - (any of various North American lean-fleshed freshwater fishes especially of the genus Micropterus)
6. bass, bass voice, basso - (the lowest adult male singing voice)
7. bass - (the member with the lowest range of a family of musical instruments)
8. bass - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

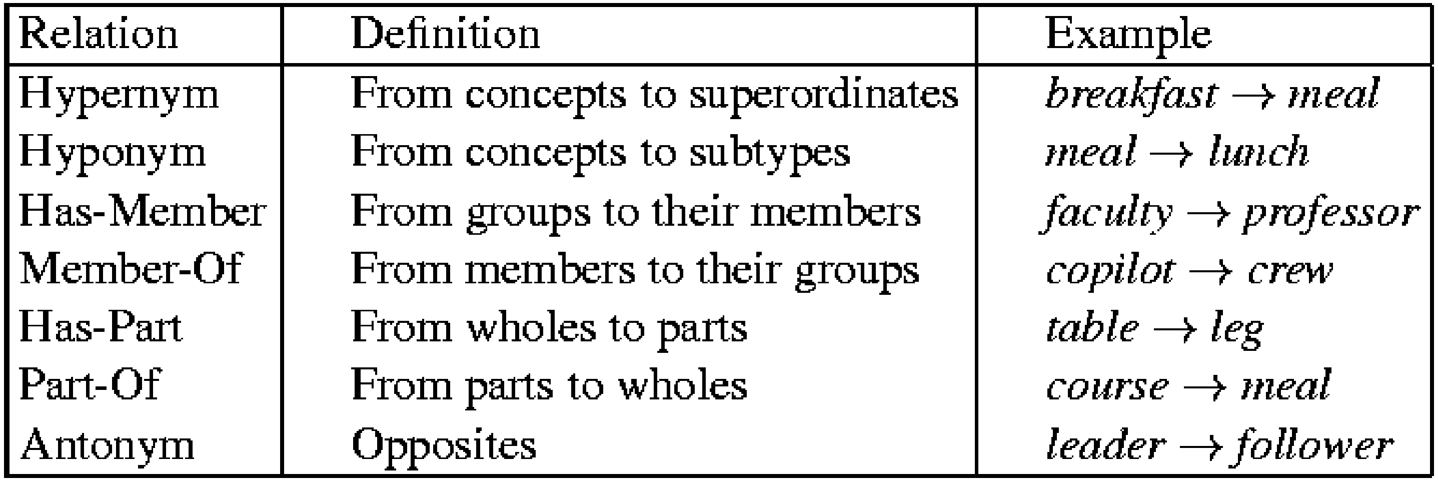
A portion of the WordNet 1.6 entry for the noun *bass*

16.2 WORDNET: A Database of Lexical Relations



Distribution of senses Among the verbs in WordNet

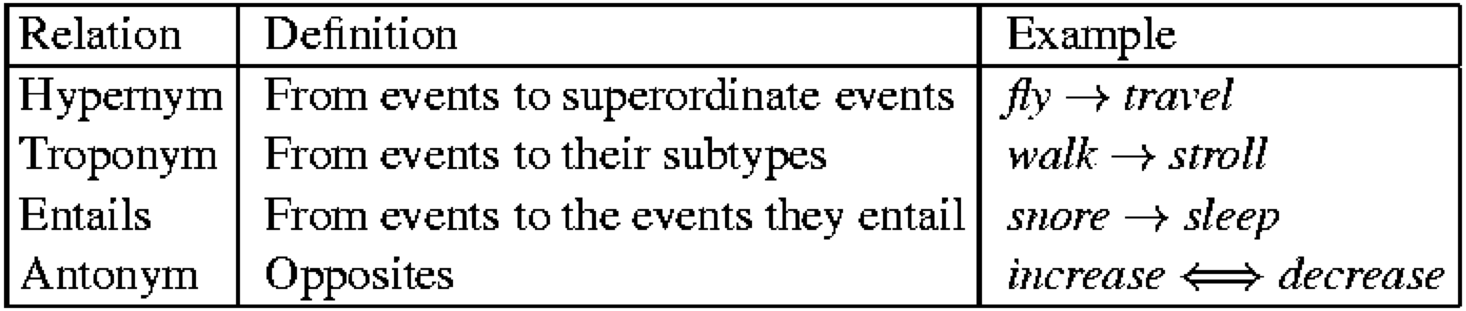
16.2 WordNet: A Database of Lexical Relations



Relation	Definition	Example
Hypernym	From concepts to superordinates	*breakfast* → *meal*
Hyponym	From concepts to subtypes	*meal* → *lunch*
Has-Member	From groups to their members	*faculty* → *professor*
Member-Of	From members to their groups	*copilot* → *crew*
Has-Part	From wholes to parts	*table* → *leg*
Part-Of	From parts to wholes	*course* → *meal*
Antonym	Opposites	*leader* → *follower*

Noun relations in WordNet

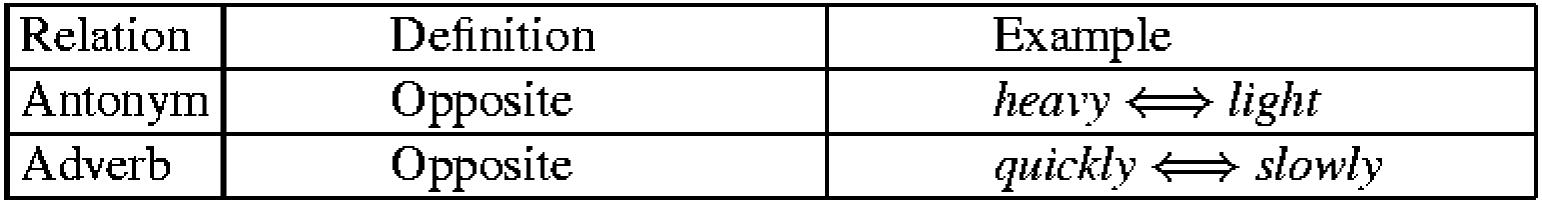
16.2 WORDNET: A Database of Lexical Relations



Relation	Definition	Example
Hypernym	From events to superordinate events	*fly* → *travel*
Troponym	From events to their subtypes	*walk* → *stroll*
Entails	From events to the events they entail	*snore* → *sleep*
Antonym	Opposites	*increase* ⟺ *decrease*

Verb relations in WordNet

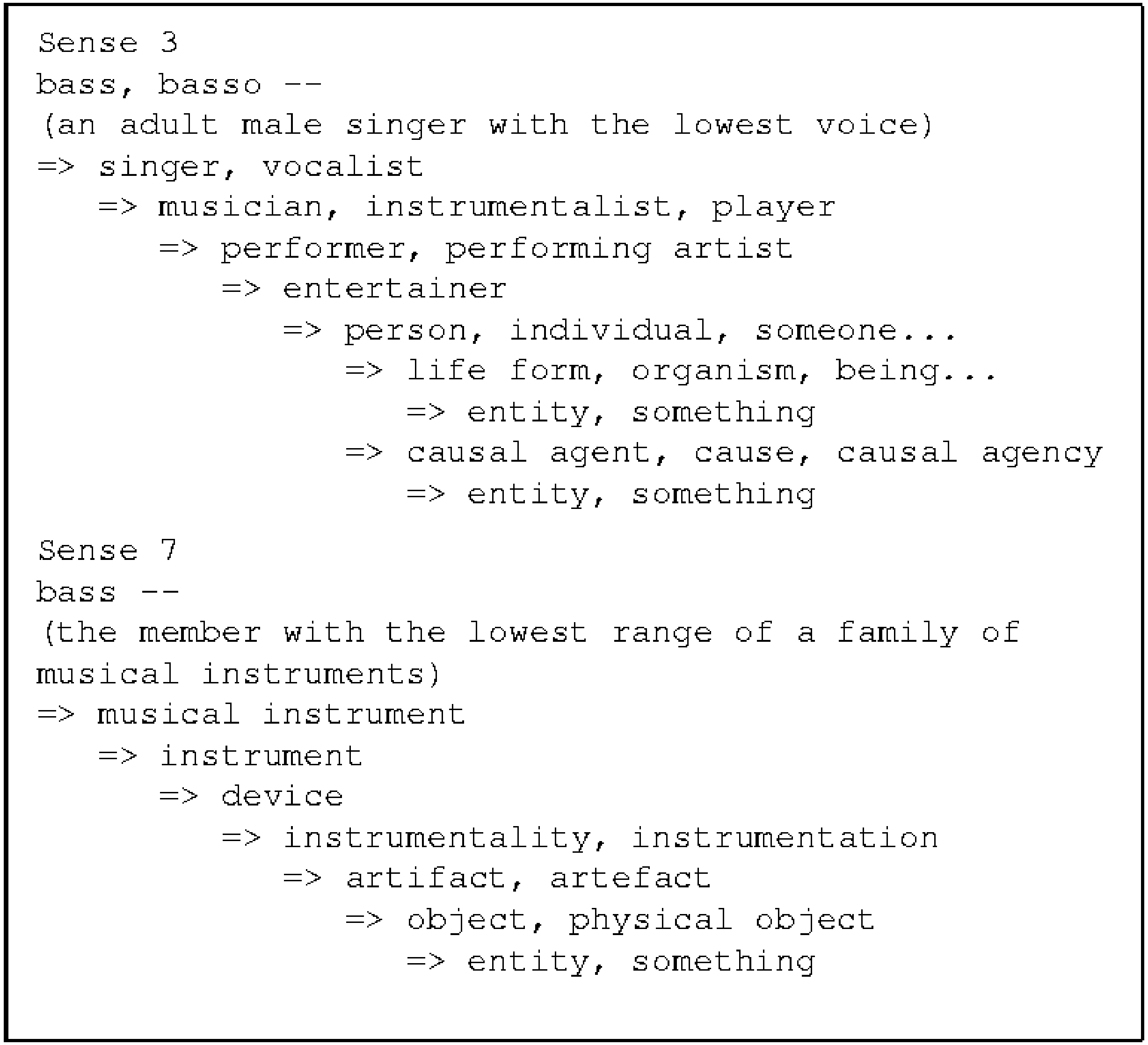
16.2 WORDNET: A Database of Lexical Relations



Relation	Definition	Example
Antonym	Opposite	*heavy* ⟺ *light*
Adverb	Opposite	*quickly* ⟺ *slowly*

Adjective and adverb relations in WordNet

16.2 WORDNET: A Database of Lexical Relations



```
Sense 3
bass, basso --
(an adult male singer with the lowest voice)
=> singer, vocalist
   => musician, instrumentalist, player
      => performer, performing artist
         => entertainer
            => person, individual, someone...
               => life form, organism, being...
                  => entity, something
               => causal agent, cause, causal agency
                  => entity, something

Sense 7
bass --
(the member with the lowest range of a family of
musical instruments)
=> musical instrument
   => instrument
      => device
         => instrumentality, instrumentation
            => artifact, artefact
               => object, physical object
                  => entity, something
```

Hyponym chains for two separate senses o the lexeme bass

16.3 The Internal Structure of Words

- **Thematic roles**
  - First proposed by Gruber (1965) and Fillmore (1968)
  - A set of categories providing a shallow semantic language for characterizing certain arguments of verbs

(16.22) Houston's Billy Hatcher broke a bat.

(16.23) He opened a drawer.

$$\exists\, e, x, y\ \mathit{Isa}(e, \mathit{Breaking}) \wedge \mathit{Breaker}(e, \mathit{BillyHatcher}) \\ \wedge \mathit{BrokenThing}(e,y) \wedge \mathit{Isa}(y,\mathit{BaseballBat})$$

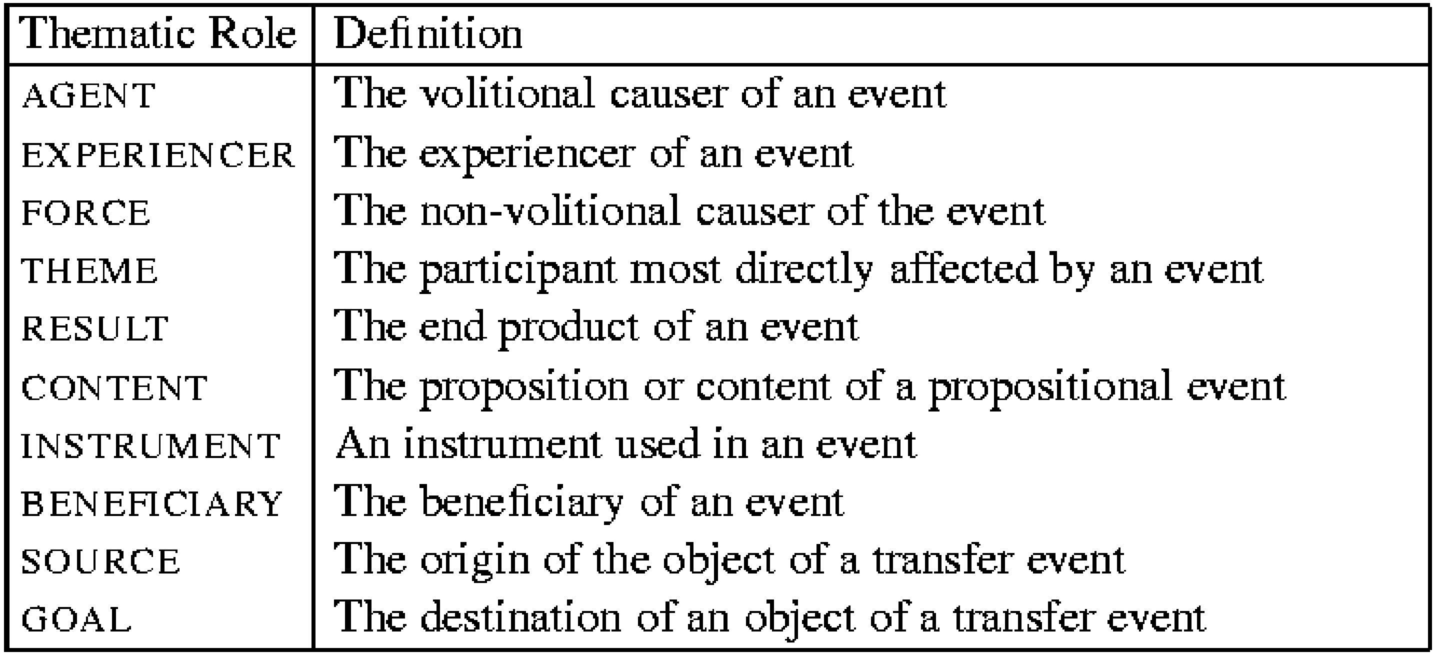
$$\exists\, e, x, y\ \mathit{Isa}(e, \mathit{Opening}) \wedge \mathit{Opener}(h, he) \\ \wedge \mathit{OpenedThing}(e, y) \wedge \mathit{Isa}(y,\mathit{Door})$$

  - The **deep roles** are specific to each possible kind of event: *Breaking* ⇒ *Breaker*; *Open* ⇒ *Opener* …

16.3 The Internal Structure of Words

- But *Breaker* and *Opener* have something in common
  - They are both volitional actors, animate, and they have direct casual relationship for their events.
- A **thematic role** is a way for expressing this commonality.
  - The subject of both these verbs are **agents**.
  - The thematic role for these objects is **theme**.
  - (16.24) A company soccer game least year got so rough that Mr. Cockwell (**experiencer**) broke his collarbone and an associate broke an ankle.
  - (16.25) The quake **(force)** broke glass in several downtown skyscrapers.
  - (16.26) It (**instrument**) broke his jaw.

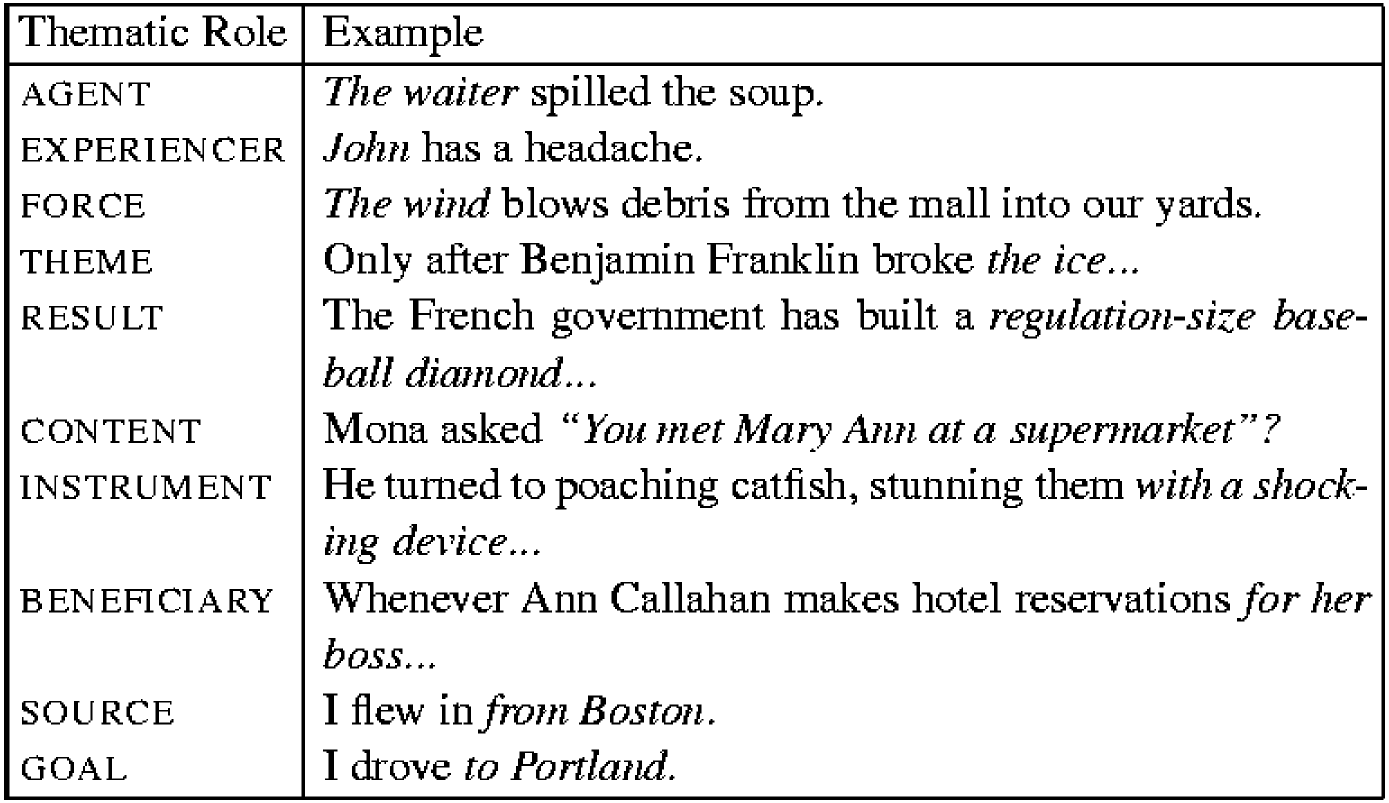
16.3 The Internal Structure of Words



Thematic Role	Definition
AGENT	The volitional causer of an event
EXPERIENCER	The experiencer of an event
FORCE	The non-volitional causer of the event
THEME	The participant most directly affected by an event
RESULT	The end product of an event
CONTENT	The proposition or content of a propositional event
INSTRUMENT	An instrument used in an event
BENEFICIARY	The beneficiary of an event
SOURCE	The origin of the object of a transfer event
GOAL	The destination of an object of a transfer event

Some commonly-used thematic roles with their definitions

16.3 The Internal Structure of Words



Thematic Role	Example
AGENT	*The waiter* spilled the soup.
EXPERIENCER	*John* has a headache.
FORCE	*The wind* blows debris from the mall into our yards.
THEME	Only after Benjamin Franklin broke *the ice...*
RESULT	The French government has built a *regulation-size baseball diamond...*
CONTENT	Mona asked *"You met Mary Ann at a supermarket"?*
INSTRUMENT	He turned to poaching catfish, stunning them *with a shocking device...*
BENEFICIARY	Whenever Ann Callahan makes hotel reservations *for her boss...*
SOURCE	I flew in *from Boston.*
GOAL	I drove *to Portland.*

Prototypical examples of various thematic roles

16.3 The Internal Structure of Words

Applications to Linking Theory and Shallow Semantic Interpretation

- A common use of thematic roles in computational system is as a shallow semantic language. (Ch. 21)
- Another use of thematic roles, was as an intermediary between semantic roles in conceptual structure or common-sense knowledge like *Breaker* and *DrivenThing* and their more language-specific surface grammatical realizations as subject and object.
- **Thematic hierarchy** for assigning the subject role:

AGENT > INSTRUMENT > THEME

(17.27) *John opened the door*
AGENT THEME

(17.28) *John opened the door with the key*
AGENT THEME INSTRUMENT

(17.29) *The key opened the door*
AGENT THEME

(17.30) *The door was opened by John*
THEME AGENT

Chapter 18. Discourse

From: Chapter 18 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- **Discourse**
  - Language does not normally consists of isolated, unrelated sentences, but instead of *collected, related groups of sentences*.
- **Monologue**
  - Characterized by a *speaker* (*writer*), and a *hearer* (*reader*)
  - Communication flows in only one direction, from the speaker to the hearer.
- **Dialogue**
  - Each participant periodically takes turns being a speaker and hearer.
  - Generally consists of different types of communicative acts:
    - Asking questions,
    - Giving answers,
    - Making corrections,
    - And so on

Background

- **HCI**, human-computer interaction
  - Limitations on the ability of computer systems to participate in free, unconstrained conversation
- Language is rife with phenomena that operate at the discourse level.

  (18.1) John went to Bill's car dealership to check out an Acura Integra. ***He*** looked at ***it*** for about an hour.
  - What do pronouns such as *he* and *it* denote?
  - Can we build a computational model for the resolution of *referring expressions*?
- Methods of of interpreting *referring expressions* (18.1)
- Establishing the *coherence* of a discourse (18.2)
- Methods of for determining the *structure* of a discourse (18.3)

Background

- Algorithms for resolving discourse-level phenomena are essential for a wide range of language applications.
  - For instance, interactions with **query interfaces and dialogue interpretation systems** like ATIS frequently contain pronouns and similar types of expressions.

    (18.2) I'd like to get from Boston to San Francisco, on either December 5th or December 6th. It's okay if *it* stops in another city along the way.

    - *It* denotes the flight that the user wants to book in order to perform the appropriate action.
  - **IE systems** must frequently extract information from utterances that contain pronouns

    (18.3) First Union Corp is continuing to wrestle with severe problems unleashed by a botched merger and a troubled business strategy. According to industry insiders at Paine Webber, ***their*** president, John R. Georgius, is planning to retire by the end of the year.
  - **Text summarization systems** employ a procedure for selecting the important sentences from a source document and using them to form a summary.

18.1 Reference Resolution

- Terminology
  - **Reference**
    - The process by which speakers use expressions like *John* and *he* to denote a person named John.
  - **Referring expression**
    - An NL expression used to perform reference
  - **Referent**
    - The entity that is referred to
  - **Corefer**
    - Two referring expressions used to refer to the same entity: *John* and *he* in (18.1)
    - *John*: the **antecedent** of *he*; *he*: an **anaphor** (and thus **anaphoric**) of *John*

18.1 Reference Resolution

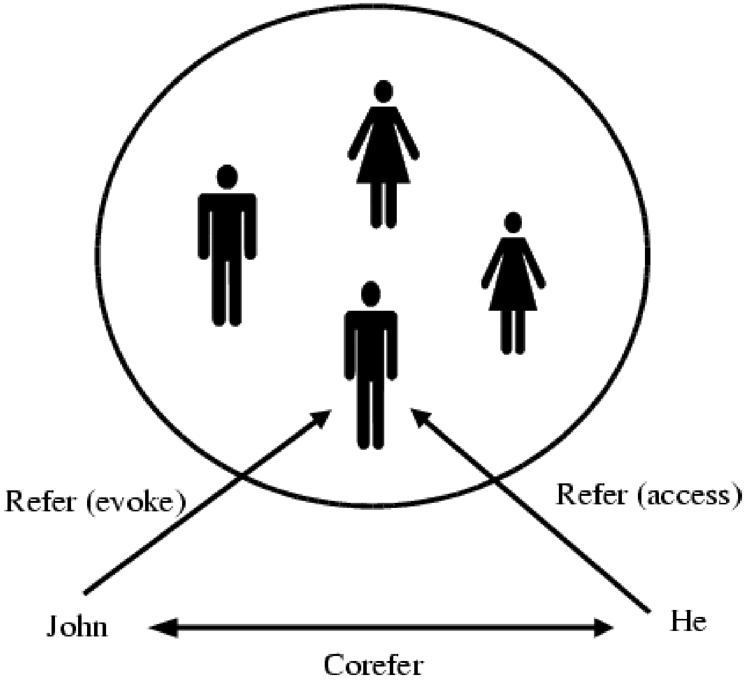
- NLs provide speakers with a variety of ways to refer to entities.
  - Depending op the operative **discourse context**, you might want to say *it*, *this*, *that*, *this car*, *that car*, *the car*, *the Acura*, *the Integra*, or, *my friend's car*, to refer to your friend's Acura Integra.
- **However, you are not free to choose between any of these alternative in any context**.
  - You can not say *it* or *the Acura* if the hearer has no prior knowledge of your friend's car, it has not been mentioned before, and it is not in the immediate surroundings of the discourse participants (i.e., the **situational context** of the discourse).

18.1 Reference Resolution

- Each type of referring expression encodes different signals about the place that the speaker believes the referent occupies within the hearer's set of beliefs.
- A subset of these beliefs that has a special status from the hearer's mental model of the ongoing discourse, which we call a **discourse model**.
  - The discourse model contains
    - representations of the entities that have been referred to in the discourse and
    - the relationships in which they participate.
- There are two components required by a system to successfully produce and interpret referring expressions:
  - A method for constructing a discourse model that evolves with the dynamically-changing discourse it represents, and
  - A method for mapping between the signals that various referring expression encode and the set of beliefs

18.1 Reference Resolution

- Two fundamental operations to the discourse model
  - **Evoke**
    - When a referent is first mentioned in a discourse, we say that a representation for it is **evoked** into the model.
  - **Access**
    - Upon subsequent mention, this representation is **accessed** from the model.



18.1 Reference Resolution

- We restrict our discussion to reference to entities, although discourses include reference to many other types of referents.

(18.4) According to John, Bob bought Sue an Integra, and Sue bought Fred a Legend.

a. But *that* turned out to be a lie. (a speech act)

b. But *that* was false. (a proposition)

c. *That* struck me a funny way to describe the situation. (a manner of description)

d. *That* caused Sue to become rather poor. (an event)

e. *That* caused them both to become rather poor. (a combination of several events)

18.1 Reference Resolution

Reference Phenomena

- Five types of referring expression
  - *Indefinite NPs*
  - *Definite NPs*
  - *Pronouns,*
  - *Demonstratives*, and
  - *One-anaphora*
- Three types of referents that complicate the reference resolution problem
  - *Inferrables*
  - *Discountinuous sets*, and
  - *Generics*

18.1 Reference Resolution
Reference Phenomena

- **Indefinite NPs**
  - Introducing entities new to the hearer into the discourse context

    (18.5) I saw *an Acura Integra* today. (evoke)

    (18.6) Some *Acura Integra* were being unloaded at the local dealership today.

    (18.7) I saw *this awesome Acura Integra* today.

    (18.8) I am going to the dealership to buy an Acura Integra today. (specific/non-specific ambiguity)

18.1 Reference Resolution
Reference Phenomena

- **Definite NPs**
  - Refer to an entity that is identifiable to the hearer, either because
    - it has already been mentioned in the discourse context,
    - it is contained in the hearer's set of beliefs about the world, or
    - the uniqueness of the objects is implied by the description itself.

  (18.9) I saw an Acura Integra today. *The Integra* was white and needed to be washed. (context)

  (18.10) *The Indianapolis 500* is the most popular car race in the US. (belief)

  (18.11) *The faster car in the Indianapolis 500* was an Integra. (uniqueness)

18.1 Reference Resolution

Reference Phenomena

- **Pronouns**
  - Another form of definite reference is pronominalization.

    (18.12) I saw an Acura Integra today. *It* was white and needed to be washed.
  - The constraints on using pronominal reference are stronger than for full definite NPs,
    - Requiring that the referent have a high degree of activation or **salience** in the discourse model.
  - Pronouns usually refer back no further than one or two sentences back in the ongoing discourse,
    - whereas definite NPs can often refer further back

    (18.13) a. John went to Bob's party, and parked next to a beautiful Acura Integra.

    b. He went inside and talked to Bob for more than an hour.

    c. Bob told him that he recently got engaged.

    d. ?? He also said that he bought *it* yesterday.

    d'. He also said that he bought *the Acura Integra* yesterday.

18.1 Reference Resolution

- **Pronoun**
  - Pronouns can also participate in **cataphora**, in which they are mentioned before there referents are.

    (18.14) Before *he* bought *it*, John checked over the Integra very carefully.
  - Pronouns can also appear in quantified context in whu=ich they are considered to be **bound**.

    (18.15) Every woman bought *her* Acura Integra at the local dealership.

18.1 Reference Resolution

- **Demonstratives**
  - Demonstrative pronouns, like *this* and *that*, can appear either alone or as determiner, for instance *this Acura*, *that Acura*.
  - The choice between two demonstratives is generally associated with some notion of spatial proximity:
    - *This* indicates closeness and *that* signaling distance

  (18.16) [John shows Bob an Acura Integra and a Mazda Miata]

  Bob (pointing): I like *this* better than *that*.

  (18.17) I bought an Integra yesterday. It's similar to the one I bought five years ago. *That one* was really nice, but I like *this one* even better.

18.1 Reference Resolution

- **One anaphora**
  - *One*-anaphora, blends properties of definite and indefinite reference.

    (18.18) I saw no less than 6 Acura Integra today. Now I want *one*.

    - *One of them*
  - One may evoke a new entity into the discourse model, but it is necessarily dependent on an existing referent for the description of this new entity.
  - Should be distinguished from the formal, non-specific pronoun usage in (10.19), and its meaning as the number one in (18.20)

    (18.19) *One* shouldn’t pay more than twenty thousand dollars for an Acura.

    (18.20) John has two Acura, but I only have *one*.

18.1 Reference Resolution

- **Inferrables**
  - A referring expression that does not refer to any entity that has been explicitly evoked in the text, but instead one that is inferentially related to an evoked entity.

    (18.21) I almost bough an Acura Integra today, but *a door* had a dent and *the engine* seemed noisy.
  - Inferrables can also specify the results of processes described by utterances in a discourse.

    (18.22) Mix the flour, and water.

    a. Kneed *the dough* until smooth and shiny.

    b. Spread *the paste* over the blueberries.

    c. Stir *the batter* until all lumps are gone.

18.1 Reference Resolution

- **Discontinuous Sets**
  - Plural referring expressions, like *they* and *them*, refer to set of entities that are
    - Evoked together using another plural expressions (*their Acura*) or
    - A conjoinded NPs (*John and Mary*)

    (18.23) John and Mary love their Acuras. *They* drive *them* all the time.

    (18.24) John has an Acura, and Mary has a Mazda. *They* drive *them* all the time. (a *pairwise* or *respective* reading)

18.1 Reference Resolution

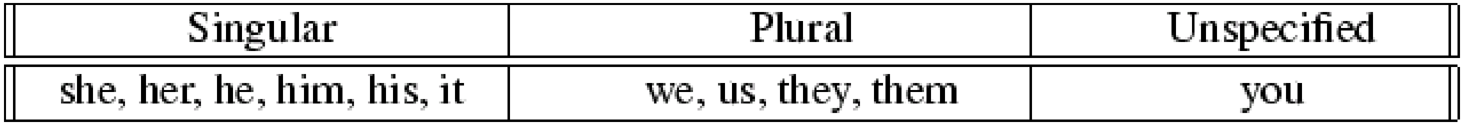
- **Generics**
  - The existence of *generics* makes the reference problem even more complicated.

    (18.25) I saw no less than 6 Acura Integra today. *They* are the coolest cars.
  - The most natural reading: *they* ⇒ the class of Integra in general

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Number agreement**



Singular	Plural	Unspecified
she, her, he, him, his, it	we, us, they, them	you

(18.26) John has a new Acura. It is red.

(18.27) John has three new Acura. They are red.

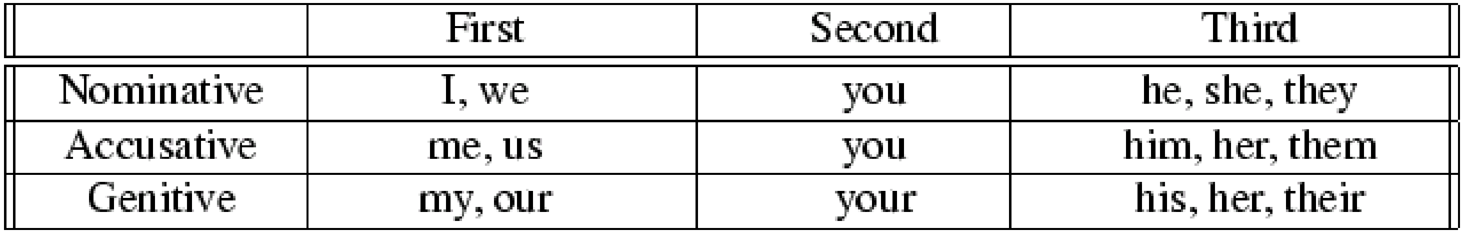
(18.28) * John has a new Acura. They are red.

(18.29) * John has three new Acura. It is rad.

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Person and Case Agreement**



	First	Second	Third
Nominative	I, we	you	he, she, they
Accusative	me, us	you	him, her, them
Genitive	my, our	your	his, her, their

(18.30) You and I have Acura. We love them.

(18.31) John and Mary has Acuras. They love them.

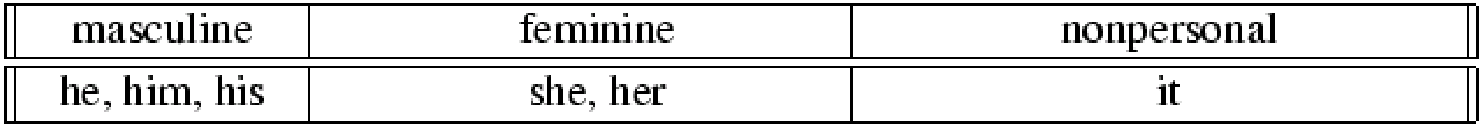
(18.32) * John and Mary has Acuras. We love them.

(18.29) * You and I have Acura. They love them.

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Gender Agreement**



masculine	feminine	nonpersonal
he, him, his	she, her	it

(18.34) John has an Acura. He is attractive. (He=John)

(18.35) John and an Acura. It is attractive. (It=the Acura)

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Syntactic constraints**
  - Reference relations may also be constrained by the syntactic relationships between a referential expression and a possible antecedent NP when both occur in the same sentence.
  - **reflexives:** *himself*, *herself*, *themselves*

    (18.36) John bought himself a new Acura. (himself=John)

    (18.37) John bought him a new Acura. (him≠John)

    (18.38) John said that Bill bought him a new Acura. (him ≠ Bill)

    (18.39) John said that Bill bought himself a new Acura. (himself = Bill)

    (18.40) He said that he bought John a new Acura. (He≠John; he≠John)

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Syntactic constraints**

  (18.41) John wanted a new car. Bill bought him a new Acura. [him=John]

  (18.42) John wanted a new car. He bought him a new Acura. [He=John; him≠John]

  (18.43) John set the pamphlets about Acuras next to himself. [himself=John]

  (18.44) John set the pamphlets about Acuras next to him. [him=John]

18.1 Reference Resolution

Syntactic and Semantic Constraints on Coreference

- **Selectional Restrictions**
  - The selectional restrictions that a verb places on its arguments may be responsible for eliminating referents.

    (18.45) John parked his Acura in the garage. *He* had driven *it* a around for hours.

    (18.46) John bought a new Acura. It drinks gasoline like you would not believe. (violation of selectional restriction: metaphorical use of *drink*)
  - Comprehensive knowledge is required to resolve the pronoun *it*.

    (18.47) John parked his Acura in the garage. *It* is incredibly messy, with old bike and car parts lying around everywhere.
  - One's knowledge about certain thing (Beverly Hills, here) is required to resolve pronouns.

    (18.48) John parked his Acura in downtown Beverly Hills. *It* is incredibly messy, with old bike and car parts lying around everywhere.

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Recency**
  - Entities introduced in recent utterances are more salient than those introduced from utterances further back.

    (18.49) John has an Integra. Bill has a Legend. Mary likes to drive it. (it=Legend)

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Grammatical Role**
  - Many theories specify a salience hierarchy of entities that is ordered by grammatical position of the referring expressions which denote them.
  - Subject > object > others

    (18.50) John went to the Acura dealership with Bill. He bought an Integra. [he=John]

    (18.51) Bill went to the Acura dealership with John. He bought an Integra. [he= Bill]

    (18.52) Bill and John went to the Acura dealership. He bought an Integra. [he= ??]

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Repeated Mention**
  - Some theories incorporate the idea that entities that have been focused on in the prior discourse are more likely to continue to be focused on the subsequent discourse, and hence references to them are more likely to be pronominalized.

    (18.53) John needed a car to get to his new job. He decided that he wanted something sporty. Bill went to the Acura dealership with him. He bought an Integra. [he=John]

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Parallelism**
  - The are strong preferences that appear to be induced by parallelism effects.

    (18.54) Mary went with Sue to the Acura dealership. Sally went with her to the Mazda dealership. [her=Sue]
  - This suggests that we might want a heuristic saying that ***non-subject pronouns prefer non-subject referents***. However, such a heuristic may not work.

    (18.55) Mary went with Sue to the Acura dealership. Sally told her not to buy anything. [her=Mary]

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Verb Semantics**
  - Certain verbs appear to place a semantically-oriented emphasis on one of their argument positions, which can have the effect of biasing the manner in which subsequent pronouns are interpreted.

    (18.56) John telephoned Bill. He lost the pamphlet on Acuras. [He=John]

    (18.57) John criticized Bill. He lost the pamphlet on Acuras. [He=Bill]
  - Some researchers have claimed this effect results from what has been called "implicit causality" of a verb:
    - The implicit cause of "criticizing" event is considered to be its object, whereas
    - The implicit cause of a "telephoning" event is considered to be its subject.

18.1 Reference Resolution

Preferences in Pronoun Interpretation

- **Verb Semantics**
  - Similar preferences have been articulated in terms of the thematic roles.

    (18.58) John seized the Acura pamphlet from Bill. *He* loves reading about car. (*Goal=John*, Source=Bill)

    (18.59) John passed the Acura pamphlet to Bill. *He* loves reading about car. (*Goal=Bill*, Source=John)

    (18.60) The car dealer admired John. He knows Acuras inside and out. (*Stimulus=John*, Experiencer=Bill)

    (18.61) The car dealer impressed John. He knows Acuras inside and out. (*Stimulus=the car dealer*, Experiencer=John)

18.1 Reference Resolution

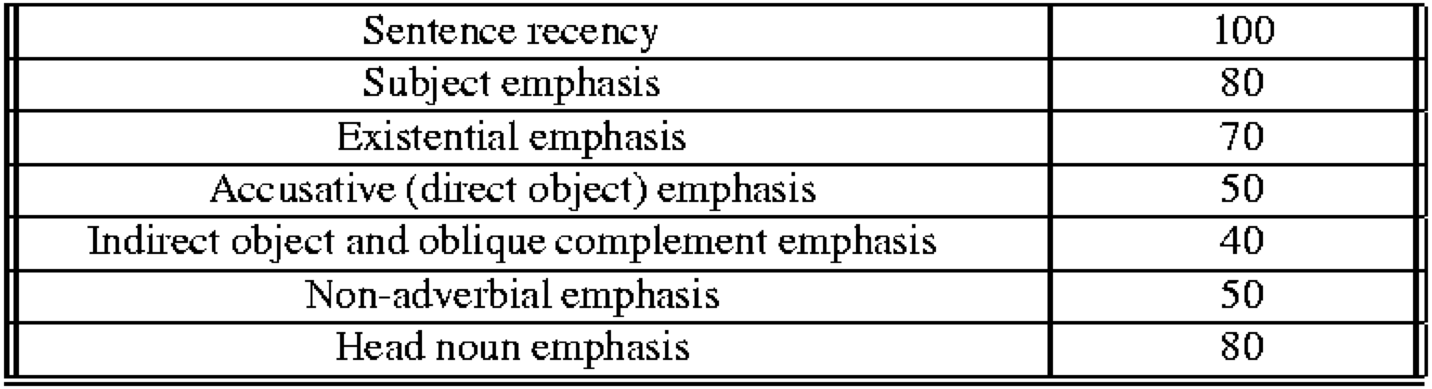
An Algorithm for Pronoun Resolution

- Lappin and Leass (1994) describe a straightforward algorithm for pronoun interpretation that takes many of the preferences into consideration.
  - It employs a simple weighting scheme integrating the effects of recency and syntactically-based preferences; no semantic preferences are employed beyond those enforced by agreement.
  - Two types of operations performed by the algorithm:
    - Discourse model update and pronoun resolution
  - When an NP evoking a new entity is encountered, a representation for it must be added to the discourse model and a degree of salience **(**a **salience value**) computed for it.
  - The salience value is calculated as the sum of the weights assigned by a set of **salience factors**. (see next page)

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- The weights that each factor assigns to an entity in the discourse model are cut in half each time a new sentence is processed.
- This, along with the added affect of the sentence recency weight, capturing the Recency preference described previously.



Sentence recency	100
Subject emphasis	80
Existential emphasis	70
Accusative (direct object) emphasis	50
Indirect object and oblique complement emphasis	40
Non-adverbial emphasis	50
Head noun emphasis	80

Fig. 18.5

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- The next five factors can be view as a way of encoding a grammatical role preference scheme using the following hierarchy:
  - subject > existential predicate nominal > object > indirect object or oblique > demarcated adverbial PP

    (18.62) *An Acura Integra* is parked in the lot. (subject)

    (18.63) There is *an Acura Integra* parked in the lot. (existential predicate nominal)

    (18.64) John parked *an Acura Integra* in the lot. (object)

    (18.65) John gave *his Acura Integra* a bath. (indirect object)

    (18.66) Inside *his Acura Integra*, John showed Susan is new CD player. (demarcated adverbial PP)

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- The head noun emphasis factor penalizes referents which are embedded in larger NP, again by promoting the weights of referents that are not.

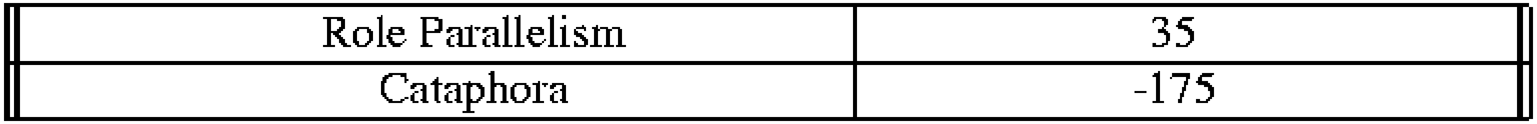
  (18.67) The owner's manual for *an Acura Integra* is on John's desk.

- It could be that several NPs in the preceding discourse refer to the same referent, each being assigned a different level of salience, and thus we need a way in which to combine the contributions of each.
  – L&L associate with each referent an equivalence class that contains all the NPs having been determined to refer to it.

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- Once we have updated the discourse model with new potential referents and recalculated the salience values associated with them, we are ready to consider the process of resolving any pronouns that exists within a new sentence.



Role Parallelism	35
Cataphora	-175

Fig. 18.6

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- The pronoun resolution algorithm
  - Assume that the DM has been updated to reflect the initial salience values for referents.

1. Collect the potential referents (up to four sentences back)
2. Remove potential referents that do not agree in number or gender with the pronouns.
3. Remove potential referents that do not pass intrasentential syntactic coreference constraints.
4. Computed the total salience value of the referent by adding any applicable values from Fig. 18.6 to the existing salience value previously computed during the discourse model update step (i.e., the sum of the applicable values from Fig. 18.5)
5. Select the referent with the highest salience value. In the case of ties, select the closest referent in terms of string position (computed without bias to direction)

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

(18.68) John saw a beautiful Acura Integra at the dealership.
He showed it to Bob.
He bought it.

- We first process the first sentence to collect potential referents and computed their initial salience values.
  - No pronouns to be resolved in this sentence.

	Rec	Subj	Exist	Obj	Ind-Obj	Non-Adv	HeadN	Total
John	100	80				50	80	310
Integra	100			50		50	80	280
dealership	100					50	80	230

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

(18.68) John saw a beautiful Acura Integra at the dealership.
He showed it to Bob.
He bought it.

- We move on to the next sentence.
  - Gender filtering: he ⇒ John

Referent	Phrases	Value
John	{*John*}	155
Integra	{*a beautiful Acura Integra*}	140
dealership	{*the dealership*}	115

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

(18.68) John saw a beautiful Acura Integra at the dealership.
He showed it to Bob.
He bought it.

- After he is resolved in the second sentence, the DM is updated as below.
  - The pronoun in the current sentence (100); Subject position (80); Not in adverbial (50); Not embedded (80)
  - Total 310 added to the current weight for John to become 465

Referent	**Phrases**	**Value**
John	{*John, he*$_1$}	465
Integra	{*a beautiful Acura Integra*}	140
dealership	{*the dealership*}	115

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- For the next pronoun *it* in the second sentence
  - The referent Integra satisfies parallelism: 140+35= 175
  - The referent dealership: 115
  - ∴ the Integra is taken to be the referent
- Update the DM:
  - *it* receives 100+50+50+80=280
  - Update to become 420
  - Bob= 100+40+50+80=270

Referent	**Phrases**	**Value**
John	{*John*, *he*$_1$}	465
Integra	{*a beautiful Acura Integra*, *it*$_1$}	420
Bob	{*Bob*}	270
dealership	{*the dealership*}	115

18.1 Reference Resolution

An Algorithm for Pronoun Resolution

- Move on to the next sentence, the DM becomes as follows.
  - According to the weights, it is clear to resolve *he* and *it* in the last sentence.

Referent	**Phrases**	**Value**
John	{*John*, *he*$_1$}	232.5
Integra	{*a beautiful Acura Integra*, *it*$_1$}	210
Bob	{*Bob*}	135
dealership	{*the dealership*}	57.5

18.1 Reference Resolution

A Tree Search Algorithm

- Hobbs (1978)

18.1 Reference Resolution

A Centering Algorithm

- Centering theory has an explicit representation of a DM, and incorporate an additional claim:
  - That there is a single entity being “centered” on any given point in the discourse which is to be distinguished from all other entities that have been evoked.
- Two main representations tracked in the DM:
  - The *backward looking center* of U_n, $C_b(U_n)$
    - Representing the entity currently being focused on in the discourse after U_n is interpreted.
  - The *forward looking center* of U_n, $C_f(U_n)$
    - Forming an ordered list containing the entities mentioned in U_n all of which could serve as the C_b of the following utterance.
  - By definition, $C_b(U_{n+1})$ is the most highly ranked element of $C_f(U_n)$ mentioned in U_{n+1}.
  - For simplicity, we use the grammatical role hierarchy in L&L algorithm to order $C_f(U_n)$.
    - subject > existential predicate nominal > object > indirect object or oblique > demarcated adverbial PP

18.1 Reference Resolution

A Centering Algorithm

- The following rules are used by the algorithm
  - Rule 1: If any element of $C_f(U_n)$ is realized by a pronoun in utterance U_{n+1}, then $C_b(U_{n+1})$ must be realized as a pronoun also.
  - Rule 2: Transition states are ordered. Continue > Retain > Smooth-shift > Rough-shift

	$C_b(U_{n+1}) = C_b(U_n)$ or undefined $C_b(U_n)$	$C_b(U_{n+1}) \neq C_b(U_n)$
$C_b(U_{n+1}) = C_p(U_{n+1})$	Continue	Smooth-Shift
$C_b(U_{n+1}) \neq C_p(U_{n+1})$	Retain	Rough-Shift

18.1 Reference Resolution

A Centering Algorithm

- The algorithm:
  1. Generate possible C_b-C_f combinations for each possible set of reference assignments.
  2. Filter by constraints,. E.g., syntactic coreference constraints, selectional restrictions, centering rules and constraints.
  3. Rank by transition ordering.

18.1 Reference Resolution

A Centering Algorithm

(18.68) John saw a beautiful Acura Integra at the dealership.
He showed it to Bob.
He bought it.

$C_f(U_1)$: {John, Integra, dealership}
$C_p(U_1)$: John
$Cb(U_1)$: undefined

$C_f(U_2)$: {John, Integra, Bob}
$C_p(U_2)$: John
$Cb(U_2)$: John
Result: Continue $C_p(U_2)= C_b(U_2)$;
undefined $C_b(U_1)$

$C_f(U_2)$: {John, dealership, Bob}
$C_p(U_2)$: John
$Cb(U_2)$: John
Result: Continue $C_p(U_2)= C_b(U_2)$;
undefined $C_b(U_1)$

$C_f(U_3)$: {John, Acura}
$C_p(U_3)$: Bob
$Cb(U_3)$: Bob
Result: Continue $C_p(U_3)= C_b(U_3)=C_b(U_2)$

$C_f(U_3)$: {Bob, Acura}
$C_p(U_3)$: Bob
$Cb(U_3)$: Bob
Result: Smooth-shift $C_p(U_3)= C_b(U_3)$;
$C_b(U_3)\neq C_b(U_2)$