5

Words, Parts of Speech, and Morphology

5.1 Words

5.1.1 Parts of Speech

We can divide the lexicon into **parts of speech** (POS), that is, classes whose words share common grammatical properties. The concept of part of speech dates back to the classical antiquity philosophy and teaching. Plato made a distinction between the verb and the noun. After him, the word classification further evolved and parts of speech grew in number until Dionysius Thrax fixed and formulated them under a form that we still use today. Aelius Donatus popularized the list of the eight parts of speech: noun, pronoun, verb, participle, conjunction, adverb, preposition, and interjection, in his work *Ars grammatica*, a reference reading in the Middle Ages:

> "Partes orationis quot sunt? Octo. Quae? Nomen pronomen verbum adverbium participium coniunctio praepositio interiectio."

The word parsing comes from the Latin phrase *partes orationis* 'parts of speech'. It corresponds to the identification of the words' parts of speech in a sentence. In natural language processing, POS tagging is the automatic annotation of words with grammatical categories also called POS tags. Parts of speech are also sometimes called lexical categories.

Most European languages have inherited the Greek and Latin part-of-speech classification with a few adaptations. The word categories as they are taught today roughly coincide in English, French, and German in spite of some inconsistencies. This is not new. To manage the nonexistence of articles in Latin, Latin grammarians tried to get the Greek article into the Latin pronoun category.

The definition of the parts of speech is sometimes arbitrary and has been a matter of debate. From Dionysius Thrax, tradition has defined the parts of speech using morphological and grammatical properties. We shall adopt essentially this viewpoint here. However, words of a certain part of speech share semantic properties and some grammars contain statements like a noun denotes a thing and a verb an action.

Parts of speech can be clustered into two main classes: the **closed class** and the **open class**. Closed class words are relatively stable over time and have a functional role. They include words such as articles, like English *the*, French *le*, or German *der*, which change very slowly. Among the closed class, there are the determiners, the pronouns, the prepositions, the conjunctions, and the auxiliary and modal verbs (Table 5.1).

Open class words form the bulk of a vocabulary. They appear or disappear with the evolution of the language. If a new word is created, say a *hedgedog*, breed of a hedgehog and a Yorkshire terrier, it will belong to an open class category: here a noun. The main categories of the open class are the nouns, the adjectives, the verbs, and the adverbs (Table 5.2). We can add interjection to this list. Interjections are words such as *ouch*, *ha*, *oh*, and so on, that express sudden surprise, pain, or pleasure.

Table 5.1. Closed class categories.

Part of speech	English	French	German
Determiners	*the, several, my*	*le, plusieurs, mon*	*der, mehrere, mein*
Pronouns	*he, she, it*	*il, elle, lui*	*er, sie, ihm*
Prepositions	*to, of*	*vers, de*	*nach, von*
Conjunctions	*and, or*	*et, ou*	*und, oder*
Auxiliaries and modals	*be, have, will, would*	*être, avoir, pouvoir*	*sein, haben, können*

Table 5.2. Open class categories.

Part of speech	English	French	German
Nouns	*name, Frank*	*nom, François*	*Name, Franz*
Adjectives	*big, good*	*grand, bon*	*groß, gut*
Verbs	*to swim*	*nager*	*schwimmen*
Adverbs	*rather, very, only*	*plutôt, très, uniquement*	*fast, nur, sehr, endlich*

5.1.2 Features

Basic categories can be further refined, that is **subcategorized**. Nouns, for instance, can be split into singular nouns and plural nouns. In French and German, nouns can also be split according to their gender: masculine and feminine for French, and masculine, feminine, and neuter for German.

Genders do not correspond in these languages and can shape different visions of the world. Sun is a masculine entity in French – *le soleil* – and a feminine one in German – *die Sonne*. In contrast, moon is a feminine entity in French – *la lune* – and a masculine one in German – *der Mond*.

Additional properties that can further specify main categories are often called the **features**. Features vary among European languages and include notably the number, gender, person, case, and tense. Each feature has a set of possible values; for instance, the number can be singular or plural.

Word features are different according to their parts of speech. In English, a verb has a tense, a noun has a number, and an adjective has neither tense nor number. In French and German, adjectives have a number but no tense. The feature list of a word defines its part of speech together with its role in the sentence.

5.1.3 Two Significant Parts of Speech: The Noun and the Verb

The Noun. Nouns are divided into proper and common nouns. Proper nouns are names of persons, people, countries, companies, and trademarks, such as: *England, Robert, Citroën*. Common nouns are the rest of the nouns. Common nouns are often used to qualify persons, things, ideas.

A noun definition referring to semantics is a disputable approximation, however. More surely, nouns have certain syntactic features, namely the number, gender, and case (Table 5.3). A noun group is marked with these features, and other words of the group, that is, determiners, adjectives, must agree with the features they share.

Table 5.3. Features of common nouns.

Features\Values	English	French	German
Number	singular, plural *waiter/waiters, book/books*	singular, plural *serveur/serveurs, livre/livres*	singular, plural *Buch/Bücher*
Gender		masculine, feminine *serveur/table*	masculine, feminine, neuter *Ober/Gabel/Tuch*
Case			nominative, accusative, genitive, dative *Junge/Jungen/Jungen/Jungen*

While number and gender are probably obvious, case might be a bit obscure for non-German speakers. Case is a function marker that inflects words such as nouns or adjectives. In German, there are four cases: nominative, accusative, genitive, and dative. The nominative case corresponds to the subject function, the accusative case to the direct object function, and the dative case to the indirect object function. Genitive denotes a possession relation. These cases are still marked in English and French for pronouns.

In addition to these features, the English language makes a distinction between nouns that can have a plural: count nouns, and nouns that cannot: mass nouns. *Milk, water, air* are examples of mass nouns.

Verbs. Semantically, verbs often describe an action, an event, a state, etc. More positively, and as for the nouns, verbs in European languages are marked by their

morphology. This morphology is quite elaborate in a language like French, notably due to the tense system. Verbs can be basically classified into three main types: auxiliaries, modals, and main verbs.

Auxiliaries are helper verbs such as *be* and *have* that enable us to build some of the main verb tenses (Table 5.4). Modal verbs are verbs immediately followed by another verb in the infinitive. They usually indicate a modality, a possibility (Table 5.5). Modal verbs are more specific to English and German. In French, semiauxiliaries correspond to a similar category.

Table 5.4. Auxiliary verbs.

English	French	German
***to be**: am, are, is, was, were* ***to have**: has, have, had* ***to do**: does, did, done*	***être**: suis, es, est, sommes, sont, étais, était* ***avoir**: ai, as, a, avons, ont, avais, avait, avions*	***sein**: bin, bist, ist, war, waren* ***haben**: habe, hast, hat, haben, habt* ***werden**: werde, wirst, wird, wurde*

Table 5.5. Modal verbs.

English	French (semiauxiliaries)	German
can, could, must, may, might, shall, should	***pouvoir**: peux, peut, pouvons, pourrai, pourrais* ***devoir**: dois, doit, devons, devrai, devrais* ***vouloir**: veux, veut, voulons, voudrai, voudrais*	***können**: kann, können, konnte* ***dürfen**: darf, dürfen, dürfte* ***mögen**: mag, mögen, möchte* ***müssen**: muß, müssen, mußte* ***sollen**: soll, sollen, sollte*

Main verbs are all the other verbs. Traditionally, main verbs are categorized according to their complement's function (Table 5.6):

- Copula or link verb – verbs linking a subject to an (adjective) complement. Copulas include **verbs of being** such as *be, être, sein* when not used as auxiliaries, and other verbs such as *seem, sembler, scheinen*.
- Intransitive – verbs taking no object.
- Transitive – verbs taking an object.
- Ditransitive – verbs taking two objects.

Verbs have more features than other parts of speech. First, the verb group shares certain features of the noun (Table 5.7). These features must agree with corresponding ones of the verb's subject.

Verbs have also specific features, namely the tense, the mode, and the voice:

Table 5.6. Verb types.

	English	French	German
Copulas	*Man **is** mortal* *She **seems** intelligent*	*l'homme **est** mortel* *Elle **paraît** intelligente*	*Der Mensch **ist** sterblich* *Sie **scheint** intelligent*
Intransitive verbs	*Frank **sleeps*** *Charlotte **runs***	*François **dort*** *Charlotte **court***	*Franz **schläft*** *Charlotte **rennt***
Transitive verbs	*You **take** the book* *Susan **reads** the paper*	*Tu **prends** le livre* *Suzanne **lis** l'article*	*Du **nimmst** das Buch* *Susan **liest** den Artikel*
Ditransitive verbs	*I **give** my neighbors the notes*	*Je **donne** les notes à mon voisin*	*Ich **gebe** die Notizen meinem Nachbarn*

Table 5.7. Features common to verbs and nouns.

Features\Values	English	French	German
Person	1, 2, and 3 *I am* *you are* *she is*	1, 2, and 3 *je suis* *tu es* *elle est*	1, 2, and 3 *ich bin* *du bist* *sie ist*
Number	singular, plural *I am/we are* *She eats/they eat*	singular, plural *je suis/nous sommes* *elle mange/elles mangent*	singular, plural *ich bin/wir sind* *sie ißt/sie essen*
Gender	–	masculine, feminine *il est mangé/elle est mangée*	–

- **Tense** locates the verb, and the sentence, in time. Tense systems are elaborate in English, French, and German, and do not correspond. Tenses are construed using form variations (Table 5.8) or auxiliaries (Table 5.9). Tenses are a source of significant form variation in French.
- **Mood** enables the speaker to present or to conceive the action in various ways (Table 5.10).
- **Voice** characterizes the sequence of syntactic groups. Active voice corresponds to the "subject, verb, object" sequence. The reverse sequence corresponds to the passive voice. This voice is possible only for transitive verbs. Some constructions in French and German use a reflexive pronoun. They correspond to the pronominal voice.

5.2 Lexicons

A lexicon is a list of words, and in this context, lexical entries are also called the **lexemes**. Lexicons often cover a particular domain. Some focus on a whole language, like English, French, or German, while some specialize in specific areas such

Table 5.8. Tenses constructed using inflection.

	English	French	German
Base	*I like to* ***sing***	*j'aime* ***chanter***	*Ich* ***singe*** *gern*
Present	*I* ***sing*** *everyday*	*Je* ***chante*** *tous les jours*	*Ich* ***singe*** *alltags*
Preterit (Simple past)	*I* ***sang*** *in my youth*	*Je* ***chantai*** *dans ma jeunesse*	*Ich* ***sang*** *in meiner Jugend*
Imperfect	–	*Je* ***chantais*** *dans ma jeunesse*	–
Future	–	*Je* ***chanterai*** *plus tard*	–
Present participle	*I am* ***singing***	*En* ***chantant*** *tous les jours*	***Singend***
Past participle	*I have* ***sung*** *before*	*J'ai* ***chanté***	*Ich habe* ***gesungen***

Table 5.9. Some tenses constructed using auxiliaries. Values do not correspond across languages.

	English	French	German
Present progressive	*I am singing*	–	–
Future	*I shall (will) sing*	–	*Ich werde singen*
Present perfect	*I have sung*	*J'ai chanté*	*Ich habe gesungen*
Pluperfect	*I had sung*	*J'avais chanté*	*Ich hatte gesungen*
Passé antérieur	–	*J'eus chanté*	–
Future perfect	*I will have sung*	*J'aurai chanté*	*Ich werde gesungen haben*
Futur antérieur	*I would have sung*	*J'aurais chanté*	*Ich würde gesungen haben*
Past progressive	*I was singing*	–	–
Future progressive	*I will be singing*	–	–
Present perfect progressive	*I have been singing*	–	–
Future perfect progressive	*I will have been singing*	–	–
Past perfect progressive	*I had been singing*	–	–

Table 5.10. Moods (Present only).

	English	French	German
Indicative	*I am singing*	*Je chante*	*Ich singe*
Imperative	*sing*	*chante*	*singe*
Conditional	*I should (would) sing*	*Je chanterais*	*Ich würde singen*
Subjunctive	Rare, it appears in expressions such as: *God save the queen*	*Il faut que je chante*	*Ich singe*

as proper names, technology, science, and finance. In some applications, lexicons try to be as exhaustive as is humanly possible. This is the case of Internet crawlers, which index all the words of all the Web pages they can find. Computerized lexicons are now embedded in many popular applications such as in spelling checkers, thesauruses, or definition dictionaries of word processors. They are also the first building block of most language processing programs.

Several options can be taken when building a computerized lexicon. They range from a collection of words – a word list – to words carefully annotated with their pronunciation, morphology, and syntactic and semantic labels. Words can also be related together using semantic relationships and definitions.

A key point in lexicon building is that many words are ambiguous both syntactically and semantically. Therefore, each word may have as many entries as it has syntactic or semantic readings. Table 5.11 shows words that have two or more parts of speech and senses. In this chapter, we only examine the syntactic part. Chap. 13 will cover semantic issues.

Table 5.11. Word ambiguity.

	English	**French**	**German**
Part of speech	*can* modal	*le* article	*der* article
	can noun	*le* pronoun	*der* pronoun
Semantic	*great* big	*grand* big	*groß* big
	great notable	*grand* notable	*groß* notable

Many computerized lexicons are now available from the industry and from sources on the Internet. English sources are the most numerous at present, but the situation is rapidly changing for other languages. Most notable ones in English include word lists derived from the *Longman Dictionary of Contemporary English* (Procter 1978) and the *Oxford Advanced Learner's Dictionary* (Hornby 1974). Table 5.12 shows the first lines of letter *A* of an electronic version of the OALD.

BDLex – standing for *Base de Données Lexicale* – is an example of a simple French lexicon (Pérennou and de Calmès 1987). BDLex features a list of words in a lemmatized form together with their part of speech and a syntactic type (Table 5.13).

5.2.1 Encoding a Dictionary

Letter trees (de la Briandais 1959) or tries (pronounce try ees) are a useful data structures to store large lexicons and to search words quickly. The idea behind a trie is to store the words as trees of characters and to share branches as far as the letters of two words are identical. Figure 5.1 shows a graphical representation of a trie encoding the words *bin*, *dark*, *dawn*, *tab*, *table*, *tables*, and *tablet*.

In Prolog, we can represent this trie as embedded lists, where each branch is a list. The first element of a branch is the root letter: the first letter of all the subwords that correspond to the branch. The leaves of the trie are the lexical entries, here the

Table 5.12. The first lines the *Oxford Advanced Learner's Dictionary*.

Word	Pronunciation	Syntactic tag	Syllable count or verb pattern (for verbs)
a	@	S-*	1
a	EI	Ki$	1
a fortiori	eI ,fOtI'OraI	Pu$	5
a posteriori	eI ,p0sterI'OraI	OA$,Pu$	6
a priori	eI ,praI'OraI	OA$, Pu$	4
a's	Eiz	Kj$	1
ab initio	&b I'nISI@U	Pu$	5
abaci	'&b@saI	Kj$	3
aback	@'b&k	Pu%	2
abacus	'&b@k@s	K7%	3
abacuses	'&b@k@sIz	Kj%	4
abaft	@'bAft	Pu$,T-$	2
abandon	@'b&nd@n	H0%,L@%	36A,14
abandoned	@'b&nd@nd	Hc%,Hd%,OA%	36A,14
abandoning	@'b&nd@nIN	Hb%	46A,14
abandonment	@'b&nd@nm@nt	L@%	4
abandons	@'b&nd@nz	Ha%	36A,14
abase	@'beIs	H2%	26B
abased	@'beIst	Hc%,Hd%	26B
abasement	@'beIsm@nt	L@%	3

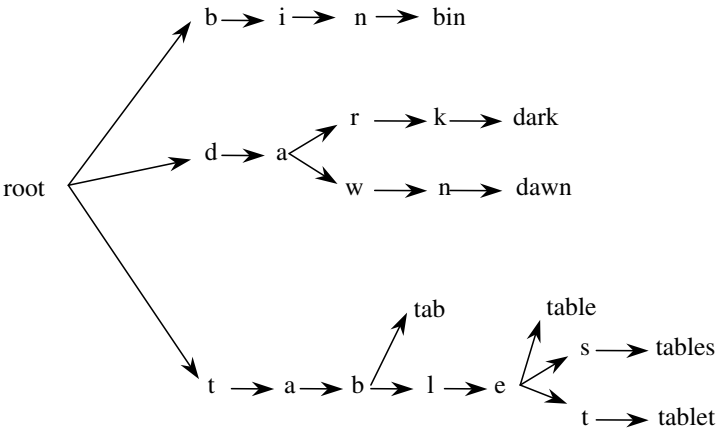

Fig. 5.1. A letter tree encoding the words *tab*, *table*, *tablet*, and *tables*.

Table 5.13. An excerpt from BDLex. Digits encode accents on letters. The syntactical tags of the verbs correspond to their conjugation type taken from the *Bescherelle* reference.

Entry	Part of speech	Lemma	Syntactic tag
a2	Prep	a2	Prep_00_00;
abaisser	Verbe	abaisser	Verbe_01_060_**;
abandon	Nom	abandon	Nom_Mn_01;
abandonner	Verbe	abandonner	Verbe_01_060_**;
abattre	Verbe	abattre	Verbe_01_550_**;
abbe1	Nom	abbe1	Nom_gn_90;
abdiquer	Verbe	abdiquer	Verbe_01_060_**;
abeille	Nom	abeille	Nom_Fn_81;
abi3mer	Verbe	abi3mer	Verbe_01_060_**;
abolition	Nom	abolition	Nom_Fn_81;
abondance	Nom	abondance	Nom_Fn_81;
abondant	Adj	abondant	Adj_gn_01;
abonnement	Nom	abonnement	Nom_Mn_01;
abord	Nom	abord	Nom_Mn_01;
aborder	Verbe	aborder	Verbe_01_060_**;
aboutir	Verbe	aboutir	Verbe_00_190_**;
aboyer	Verbe	aboyer	Verbe_01_170_**;
abre1ger	Verbe	abre1ger	Verbe_01_140_**;
abre1viation	Nom	abre1viation	Nom_Fn_81;
abri	Nom	abri	Nom_Mn_01;
abriter	Verbe	abriter	Verbe_01_060_**;

words themselves that we represent as atoms. Of course, these entries could contain more information, such as the part of speech, the pronunciation, etc.

```
[
 [b, [i, [n, bin]]]
 [d, [a, [r, [k, dark]],
         [w, [n, dawn]]]]
 [t, [a, [b, tab,
             [l, [e, table,
                     [s, tables],
                     [t, tablet]]]]]]]
]
```

5.2.2 Building a Trie in Prolog

The `make_trie/2` predicate builds a trie from a lexicon represented as an ordered list of atoms.

```
% make_trie(+WordList, -Trie)
make_trie([Word | WordList], Trie) :-
```

```
    make_trielist(Word, Word, WordTrie),
    make_trie(WordList, [WordTrie], Trie).

% make_trie(+WordList, -Trie, -FinalTrie)
make_trie([], T, T) :- !.
make_trie([Word | WordList], Trie, FinalTrie) :-
    insert_word_in_trie(Word, Word, Trie, NewTrie),
    make_trie(WordList, NewTrie, FinalTrie).
```

The `make_trie/2` predicate uses `make_trielist/3` to transform an atom into a trie representing a single word. The `make_trielist/3` predicate takes the word and the lexical entry as an input:

```
?- make_trielist(tab, noun, TL).
TL = [t, [a, [b, noun]]]

%make_trielist(+Word, +Leave, -WordTtrie)
% Creates the trie for a single word.
% Leaf contains the type of the word.
make_trielist(Word, Leaf, WordTrie) :-
    atom_chars(Word, CharList),
    make_trielist_aux(CharList, Leaf, WordTrie).

make_trielist_aux([X], Leaf, [X, Leaf]) :- !.
make_trielist_aux([X | L], Leaf, [X | [LS]]) :-
    make_trielist_aux(L, Leaf, LS).
```

Finally, `make_trie/2` inserts a word trie into the lexicon trie using `insert_word_in_trie/4`:

```
%Inserts a word in a trie.
%The Leaf argument contains the type of the word
%insert_word_in_trie(+Word, +Leaf, +Trie, -NewTrie)
insert_word_in_trie(Word, Leaf, Trie, NewTrie) :-
    make_trielist(Word, Leaf, WordTrie),
    insert_wordtrie_in_trie(WordTrie, Trie, NewTrie).

%Inserts a word trie in a trie
%insert_wordtrie_in_trie(+WordTrie, +Trie, -NewTrie)
insert_wordtrie_in_trie([H | [T]],
      [[H, Leaf | BT] | LT], [[H, Leaf | NB] | LT]) :-
    atom(Leaf),
    !,
    insert_wordtrie_in_trie(T, BT, NB).
% Traverses a segment shared between the trie and
% the word and encounters a leaf.
```

```
% It assumes that the leaf is an atom.

insert_wordtrie_in_trie([H | [T]], [[H | BT] | LT],
    [[H | NB] | LT]) :-
  !,
  insert_wordtrie_in_trie(T, BT, NB).
% Traverses a segment shared between the trie and
% the word.

insert_wordtrie_in_trie([H | T], [[HT | BT] | LT],
    [[HT | BT] | NB]) :-
  !,
  insert_wordtrie_in_trie([H | T], LT, NB).
% Traverses a nonshared segment

insert_wordtrie_in_trie(RW, RT, NB) :-
  append(RT, [RW], NB),
  !.
% Appends the remaining part of the word to the trie.
```

5.2.3 Finding a Word in a Trie

The rules to find a word in a trie are easier to write. A first rule compares the first letter of the word to the trie and unifies with the branch starting with this letter. It continues recursively with the remaining characters of the word. A second rule extracts the lexical entries that we assume to be atoms.

```
% Checks if a word is in a trie
% is_word_in_trie(+WordChars, +Trie, -Lex)
is_word_in_trie([H | T], Trie, Lex) :-
  member([H | Branches], Trie),
  is_word_in_trie(T, Branches, Lex).
is_word_in_trie([], Trie, LexList) :-
  findall(Lex, (member(Lex, Trie), atom(Lex)),
    LexList),
  LexList \= [].
% We assume that the word lexical entry is an atom
```

5.3 Morphology

5.3.1 Morphemes

From a morphological viewpoint, a language is a set of morphemes divided into **lexical** and **grammatical** morphemes. Lexical morphemes correspond to the word stems

and form the bulk of the vocabulary. Grammatical morphemes include grammatical words and the affixes. In European languages, words are made of one or more morphemes (Table 5.14). The affixes are concatenated to the stem (bold): before it – the prefixes (underlined) – and after it – the suffixes (double underlined). When a prefix and a suffix surrounding the stem are bound together, it is called a circumfix, as in the German part participle (wavy underlines).

Table 5.14. Morpheme decomposition. We replaced the stems with the corresponding lemmas.

	Word	**Morpheme decomposition**
English	*disentangling*	*dis*+*en*+***tangle***+*ing*
	rewritten	*re*+***write***+*en*
French	*désembrouillé*	*dé*+*em*+***brouiller***+*é*
	récrite	*re*+***écrire***+*te*
German	*entwirrend*	*ent*+***wirren***+*end*
	wiedergeschrieben	*wieder*+*ge*+***schreiben***+*en*

Affixing grammatical morphemes to the stem is general property of most European languages, which is **concatenative morphology** (Fig. 5.2). Although there are numerous exceptions, it enables us to analyze the structure of most words.

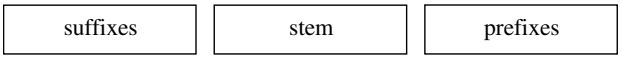

Fig. 5.2. Concatenative morphology where prefixes and suffixes are concatenated to the stem.

Concatenative morphology is not universal, however. The Semitic languages, like Arabic or Hebrew, for instance, have a **templatic morphology** that interweaves the grammatical morphemes to the stem. There are also examples of nonconcatenative patterns in European languages like in irregular verbs of German. The verb *singen* 'sing' has the forms *sangst* 'you sang' and *gesungen* 'sung' where the stem [s–ng] is embedded into the grammatical morphemes [–a–st] for the second-person preterit (Fig. 5.3) and [ge–u–en] for the past participle (Fig. 5.4).

5.3.2 Morphs

Grammatical morphemes represent syntactic or semantic functions whose realizations in words are called **morphs**. Using an object-oriented terminology, morphemes would be the classes, while morphs would be the objects. The **allomorphs** correspond to the set of all the morphs in a morpheme class.

The plural morpheme of English and French nouns is generally realized with an *s* suffix – an *s* added at the end of the noun. It can also be *es* or nothing (∅) in English and *ux* in French. In German, the plural morpheme can take several shapes, such as suffixes *e*, *en* , *er*, *s*, or an umlaut on the first vowel of the word (Table 5.15):

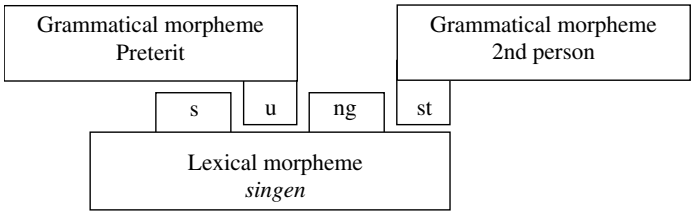

Fig. 5.3. Embedding of the stem into the grammatical morphemes in the German verb *sangst* (second-person preterit of *singen*). After Simone (1998, p. 144).

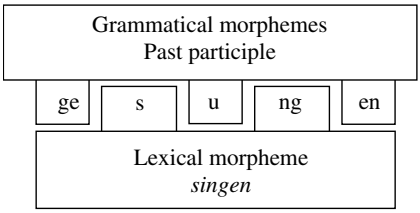

Fig. 5.4. Embedding of the stem into the grammatical morphemes in the German verb *gesungen* (past participle of *singen*). After Simone (1998, p. 144).

- In English, suffixes *-s*, *-es*, etc.
- In French, *-s*, *-ux*, etc.
- In German, an umlaut on the first vowel and the *-e* suffix, or simply the *-e* suffix.

Table 5.15. Plural morphs.

	Plural of nouns	**Morpheme decomposition**
English	*hedgehogs*	*hedgehog+s*
	churches	*church+es*
	sheep	*sheep+∅*
French	*hérissons*	*hérisson+s*
	chevaux	*cheval+ux*
German	*Gründe*	*Grund+(¨)e*
	Hände	*Hand+(¨)e*
	Igel	*Igel+∅*

Plurals also offer exceptions. Many of the exceptions, such as *mouse* and *mice*, are not predictable and have to be listed in the lexicon.

5.3.3 Inflection and Derivation

Some Definitions. We saw in Chap. 1 that morphology can be classified into **inflection**, the form variation of a word according to syntactic features such as gender,

number, person, tense, etc., and **derivation**, the creation of a new word – a new meaning – by concatenating a word with a specific affix. A last form of construction is the **composition (compounding)** of two words to give a new one, for instance, *part of speech*, *can opener*, *pomme de terre.* Composition is more obvious in German, where such new words are not separated with a space, for example, *Führerschein*. In English and French, some words are formed in this way, such as *bedroom*, or are separated with a hyphen, *centre-ville*. However, the exact determination of other compounded words – separated with a space – can be quite tricky.

Inflection. Inflection corresponds to the application of a grammatical feature to a word, such as putting a noun into the plural or a verb into the past participle (Table 5.16). It is also governed by its context in the sentence; for instance, the word is bound to agree in number with some of its neighbors.

Inflection is relatively predictable – regular – depending on the language. Given a lemma, its part of speech, and a set of grammatical features, it is possible to construct a word form using rules, for instance, gender, plural, or conjugation rules. The past participle of regular English, French, and German verbs can be respectively formed with an *ed* suffix, an *é* suffix, and the *ge* prefix and the *t* suffix. Morphology also includes frequent exceptions that can sometimes also be described by rules.

Table 5.16. Verb inflection with past participle.

	English	**French**	**German**
Base form	*work*	*travailler, chanter*	*arbeiten*
	sing	*paraître*	*singen*
Past participle (regular)	*worked*	*travaillé, chanté*	*gearbeitet*
Past participle (exception)	*sung*	*paru*	*gesungen*

Inflectional systems are similar in European languages but show differences according to the syntactic features. In English, French, and German, nouns are inflected with plurals and are consequently decorated with a specific suffix. However, in French and other Romance languages, verbs are inflected with future. Verb *chanterons* is made of two morphs: *chant* 'sing' and *-erons*. The first one is the stem (root) of *chanter*, and the second one is a suffix indicating the future tense, the first person, and the plural number. In English and German, this tense is rendered with an auxiliary: *we shall sing* or *wir werden singen*.

Derivation. Derivation is linked to lexical semantics and involves another set of affixes (Table 5.17). Most affixes can only be attached to a specific lexical category (part of speech) of words: some to nouns, others to verbs, etc. Some affixes leave the derived word in the same category, while some others entail a change of category. For instance, some affixes transform adjectives into adverbs, nouns into adjectives, and verbs into nouns (Table 5.18). Derivation rules can be combined and are sometimes

complex. For instance, the word *disentangling* features two prefixes: *dis-* and *en-*, and a suffix *-ing*.

Table 5.17. Derivational affixes.

	English	French	German
Prefixes	***fore**see*, ***un**pleasant*	***pré**voir*, ***dé**plaisant*	***vorher**sehen*, ***un**angenehm*
Suffixes	*manage**able**, rigor**ous***	*gér**able**, rigour**eux***	*vorsicht**ich**, streit**bar***

Table 5.18. Derivation related to part of speech.

	Adjectives	Adverbs	Nouns	Adjectives	Verbs	Nouns
English	*recent* *frank*	*recently* *frankly*	*air* *base*	*aerial* *basic*	*compute*	*computation*
French	*récent* *franc*	*récemment* *franchement*	*lune* *air*	*lunaire* *aérien*	*calculer*	*calcul*
German	*glücklich* *möglich*	*glücklicherweise* *möglicherweise*	*Luft* *Grund*	*luftig* *gründlich*	*rechnen*	*Rechnung*

Some semantic features of words, such as the contrary or the possibility, can be roughly associated to affixes, and so word meaning can be altered using them (Table 5.19). However, derivation is very irregular. Many words cannot be generated as simply, because the word does not exist or sounds weird. In addition, some affixes cannot be mapped to clear semantic features.

Table 5.19. Word derivation.

	Word	Contrary	Possibility
English	*pleasant* *do*	*unpleasant* *undo*	**pleasable* *doable*
French	*plaisant* *faire*	*déplaisant* *défaire*	**plaisable* *faisable*
German	*angenehm* *tun*	*unangenehm* **untun*	**angenehmbar* *tunlichst*

Compounding is a feature of German, Dutch, and the Scandinavian languages. It resembles the English noun sequences with the difference that nouns are not separated with a white space. English open compounds (e.g., *word processor*) are.

Morphological Processing. Morphological processing includes parsing and generation (Table 5.20). Parsing consists in splitting an inflected, derived, or com-

pounded word into morphemes; this process is also called a **lemmatization**. Lemmatization refers to transforming a word into its canonical dictionary form, for example, *retrieving* into *retrieve*, *recherchant* into *rechercher*, or *suchend* into *suchen*. Stemming consists of removing the suffix from the rest of the word. Taking the previous examples, this yields *retriev*, *recherch*, and *such*. Lemmatization and stemming are often mistaken. Conversely, generation consists of producing a word – a lexical form – from a set of morphemes.

Table 5.20. Morphological generation and parsing.

Generation →					
English		**French**		**German**	
dog+s	*dogs*	*chien+s*	*chiens*	*Hund+e*	*Hunde*
work+ing	*working*	*travailler+ant*	*travaillant*	*arbeiten+end*	*arbeitend*
un+do	*undo*	*dé+faire*	*défaire*		
← Parsing					

In French, English, and German, derivation operates on open class words. In English and French, a word of this class consists of a stem preceded by zero or more derivational prefixes and followed by zero or more derivational suffixes. An inflectional suffix can be appended to the word. In German, a word consists of one or more stems preceded by zero or more derivational prefixes and followed zero or more derivational suffixes. An inflectional prefix and an inflectional suffix can be appended to the word (Table 5.21). As we saw earlier, these rules are general principles of concatenative morphology that have exceptions.

Table 5.21. Open class word morphology, where `*` denotes zero or more elements and `?` denotes an optional element.

English and French	`prefix* stem suffix* inflection?`
German	`inflection? prefix* stem* suffix* inflection?`

Ambiguity. Word lemmatization is often ambiguous. An isolated word can lead to several readings: several bases and morphemes, and in consequence several categories and features as exemplified in Table 5.22.

Lemmatization ambiguities are generally resolved using the word context in the sentence. Usually only one reading is syntactically or semantically possible, and others are not. The correct reading of a word's part of speech is determined considering the word's relations with the surrounding words and with the rest of the sentence. From a human perspective, this corresponds to determining the word's function in the sentence. As we saw in the introduction, this process has been done by genera-

Table 5.22. Lemmatization ambiguities.

	Words	Words in context	Lemmatization
English	*Run*		
		1. *A **run** in the forest*	1. **run**: noun singular
		2. *Sportsmen **run** everyday*	2. **run**: verb present third person plural
French	*Marche*		
		1. *Une **marche** dans la forêt*	1. **marche**: noun singular feminine
		2. *Il **marche** dans la cour*	2. **marcher**: verb present third person singular
German	*Lauf*		
		1. *Der **Lauf** der Zeit*	1. **Der Lauf**: noun, sing, masc
		2. ***Lauf** schnell!*	2. **laufen**: verb, imperative, singular

tions of pupils dating as far back as the schools of ancient Greece and the Roman Empire.

5.3.4 Language Differences

Paper lexicons do not include all the words of a language but only lemmas. Each lemma is fitted with a morphological class to relate it to a model of inflection or possible exceptions. A French verb will be given a class of conjugation or its exception pattern – one among a hundred. English or German verbs will be marked as regular or strong and in this latter case will be given their irregular forms. Then, a reader can apply morphological rules to produce all the lexical forms of the language.

Automatic morphological processing tries to mimic this human behavior. Nevertheless, it has not been so widely implemented in English as in other languages. Programmers have often preferred to pack all the English words into a single dictionary instead of implementing a parser to do the job. This strategy is possible for European languages because morphology is finite: there is a finite number of noun forms, adjective forms, or verb forms. It is clumsy, however, to extend it to languages other than English because it considerably inflates the size of dictionaries.

Statistics from Xerox (Table 5.23) show that techniques available for storing English words are very costly for many other languages. It is not a surprise that the most widespread morphological parser – KIMMO – was originally built for Finnish, one of the most inflection-rich languages. In addition, while English inflection is tractable by means of storing all the forms in a lexicon, it is often necessary to resort to a morphological parser to deal with forms such as: *computer, computerize, computerization, recomputerize* (Antworth 1994), which cannot all be foreseen by lexicographers.

Table 5.23. Some language statistics from a Xerox promotional flyer.

Language	Number of stems	Number of inflected forms		Lexicon size (kb)
English	55,000	240,000		200–300
French	50,000	5,700,000		200–300
German	50,000	350,000 infinite	or (compounding)	450
Japanese	130,000	200 20,000,000	suffixes word forms	500 500
Spanish	40,000	3,000,000		200–300

5.4 Morphological Parsing

5.4.1 Two-Level Model of Morphology

Using a memory expensive method, lemmatization can be accomplished with a lexicon containing all the words with all their possible inflections. A dictionary lookup yields then the lemma of each word in a text. Although it has often been used for English, this method is not very efficient for many other languages. We now introduce the two-level model of Kimmo Koskenniemi (1983), which is universal and has been adopted by many morphological parsers.

The two-level morphology model enables us to link the **surface form** of a word – the word as it is actually in a text – to its **lexical** or **underlying form** – its sequence of morphemes. Karttunen (1983) did the first implementation of this model, which he named KIMMO. A later implementation – PC-KIMMO 2 – was carried out by Antworth (1995) in C. PC-KIMMO 2 is available from the Summer Institute of Linguistics through the Internet.

Table 5.24 shows examples of correspondence between surface forms and lexical forms. Morpheme boundaries in lexical forms are denoted by $+$.

Table 5.24. Surface and lexical forms.

Generation: Lexical to surface form →		
English	*dis+en+tangle+ed*	*disentangled*
	happy+er	*happier*
	move+ed	*moved*
French	*dés+em+brouiller+é*	*désembrouillé*
	dé+chanter+erons	*déchanterons*
German	*ent+wirren+end*	*entwirrend*
	wieder+ge+schreiben+en	*wiedergeschrieben*
Parsing: ← Surface to lexical form		

In the two-level model, the mapping between the surface and lexical forms is synchronous. Both strings need to be aligned with a letter-for-letter correspondence.

That is, the first letter of the first form is mapped to the first letter of the second form, and so on. To maintain the alignment, possible null symbols are inserted in either form and are denoted ε or 0, if the Greek letters are not available. They reflect a letter deletion or insertion. Table 5.25 shows aligned surface and lexical forms.

Table 5.25. Correspondence between lexical and surface forms.

English	`dis+en+tangle+ed`	`happy+er`	`move+ed`
	↕ …	↕ …	↕ …
	`dis0en0tangl00ed`	`happi0er`	`mov00ed`
French	`dé+chanter+erons`	`cheval+ux`	`cheviller+é`
	↕ …	↕ …	↕ …
	`dé0chant000erons`	`cheva00ux`	`chevill000é`
German	`singen+st`	`Grund+¨e`	`Igel+Ø`
	↕ …	↕ …	↕ …
	`singe00st`	`Gründ00e`	`Igel00`

5.4.2 Interpreting the Morphs

Considering inflection only, it is easier to interpret the morphological information using grammatical features rather than morphs. Most morphological parsers represent the lexical form as a concatenation of the stem and its features instead of morphs. For example, the Xerox parser output for *disentangle*, *happier*, and *Gründe* is:

```
disentangle+Verb+PastBoth+123SP
happy+Adj+Comp
Grund+Noun+Masc+Pl+NomAccGen
```

where the feature `+Verb` denotes a verb, `+PastBoth`, either past tense or past participle, and `+123SP` any person, singular or plural; `+Adj` denotes an adjective and `+Comp`, a comparative; `+Noun` denotes a noun, `+Masc` masculine, `+Pl`, plural, and `+NomAccGen` either nominative, accusative, or genitive. (All these forms are ambiguous, and the Xerox parser shows more than one interpretation per form.)

Given these new lexical forms, the parser has to align the feature symbols with letters or null symbols. The principles do not change, however (Fig. 5.5).

5.4.3 Finite-State Transducers

The two-level model is commonly implemented using finite-state transducers (FST). Transducers are automata that accept, translate, or generate pairs of strings. The arcs are labeled with two symbols: the first symbol is the input and the second is the output. The input symbol is transduced into the output symbol as a transition occurs on the arc. For instance, the transducer in Fig. 5.6 accepts or generates the string *abbbc* and translates into *zyyyx*.

Lexical	d	i	s	e	n	t	a	n	g	l	e	+Verb	+PastBoth	+123sp
Surface	d	i	s	e	n	t	a	n	g	l	0	0	e	d

Lexical	h	a	p	p	y	+Adj	+Comp
Surface	h	a	p	p	i	e	r

Lexical	G	r	u	n	d	+Noun	+Masc	+Pl	+NomAccGen
Surface	G	r	ü	n	d	0	0	0	e

Fig. 5.5. Alignments with features.

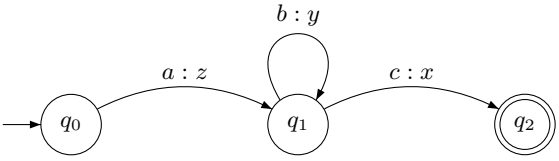

Fig. 5.6. A transducer.

Finite-state transducers have a formal definition, which is similar to that of finite-state automata. A FST consists of five components $(Q, \Sigma, q_0, F, \delta)$, where:

1. Q is a finite set of states.
2. Σ is a finite set of symbol or character pairs $i : o$, where i is a symbol of the input alphabet and o of the output alphabet. As we saw, both alphabets may include epsilon transitions.
3. q_0 is the start state, $q_0 \in Q$.
4. F is the set of final states, $F \subseteq Q$.
5. δ is the transition function $Q \times \Sigma \rightarrow Q$, where $\delta(q, i, o)$ returns the state where the automaton moves when it is in state q and consumes the input symbol pair $i : o$.

The quintuple, which defines the automaton in Fig. 5.6 is $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a : z, b : y, c : x\}$, $\delta = \{\delta(q_0, a : z) = q_1, \delta(q_1, b : y) = q_1, \delta(q_1, c : x) = q_2\}$, and $F = \{q_2\}$.

5.4.4 Conjugating a French Verb

Morphological FSTs encode the lexicon and express all the legal transitions. Arcs are labeled with pairs of symbols representing letters of the surface form – the word – and the lexical form – the set of morphs.

Table 5.26 shows the future tense of regular French verb *chanter*, where suffixes are specific to each person and number, but are shared by all the verbs of the so-called first group. The first group accounts for the large majority of French verbs. Table 5.27 shows the aligned forms and Fig. 5.7 the corresponding transducer. The arcs are annotated by the input/output pairs, where the left symbol corresponds to the lexical form and the right one to the surface form. When the lexical and surface characters are equal, as in `c:c`, we just use a single symbol in the arc.

Table 5.26. Future tense of French verb *chanter*.

Number\Person	First	Second	Third
singular	*chanterai*	*chanteras*	*chantera*
plural	*chanterons*	*chanterez*	*chanteront*

Table 5.27. Aligned lexical and surface forms.

Number\Pers.	First	Second	Third
singular	`chanter+erai`	`chanter+eras`	`chanter+era`
	`chant000erai`	`chant000eras`	`chant000era`
plural	`chanter+erons`	`chanter+erez`	`chanter+eront`
	`chant000erons`	`chant000erez`	`chant000eront`

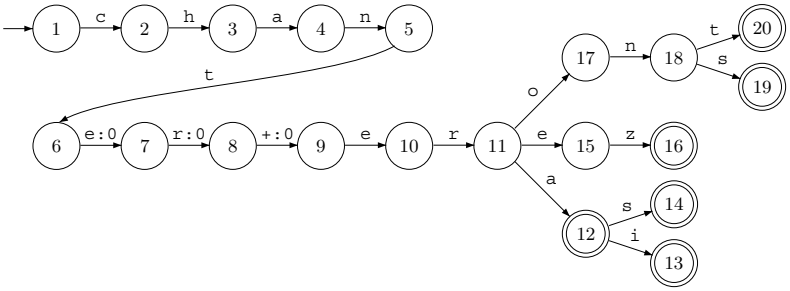

Fig. 5.7. A finite-state transducer describing the future tense of *chanter*.

This transducer can be generalized to any regular French verb of the first group by removing the stem part and inserting a self-looping transition on the first state (Fig. 5.8).

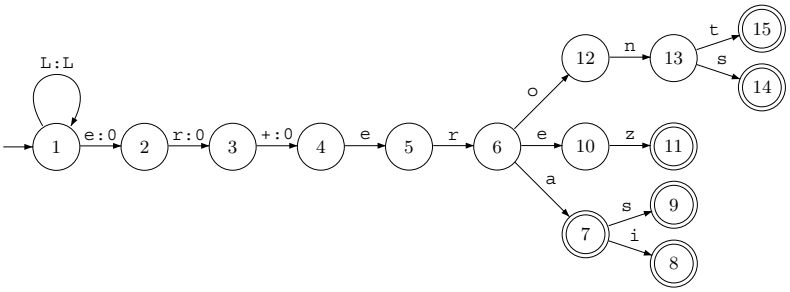

Fig. 5.8. A finite-state transducer describing the future tense of French verbs of the first group.

The transducer in Fig. 5.8 also parses and generates forms that do not exist. For instance, we can forge an imaginary French verb **palimoter* that still can be conjugated by the transducer. Conversely, the transducer will successfully parse the improbable **palimoterons*. This process is called overgeneration (both in parsing and generation).

Overgeneration is not that harmful provided that inputs are well formed. However, it can lead to some wrong parses. Consider English and German comparatives that are formed with *-er* suffix. Raw implementation of a comparative transducer would rightly parse *greater* as *great*+*er* but could also parse *better* or *reader*. Overgeneration is reduced by a lexical lookup, where the parse result is searched in a dictionary. This eliminates nonexistent words. It can also be limited by a set of constraints on affixes restricting the part of speech of the word to which they can be appended – here adjectives.

5.4.5 Prolog Implementation

Finite-state transducers can easily be implemented in Prolog. In this section, we implement the future tense of regular French verbs corresponding to Fig. 5.8, and we remove null symbols by inserting a mute transition in the surface form. The transducer has four parameters: the start state, normally 1, a final state, together with a lexical form and a surface one:

```
transduce(+Start, ?Final, ?Lexical, ?Surface).
```

The transducer parses surface forms:

```
?- transduce(1, Final, Lexical, [r, ê, v, e, r, a]).
  Final = 7,
  Lexical = [r, ê, v, e, r, +, e, r, a]
```

It also generates surface forms from lexical ones:

```
?- transduce(1, Final,
    [r, ê, v, e, r, +, e, r, e, z], Surface).
  Final = 11,
  Surface = [r, ê, v, e, r, e, z]
```

Finally, the transducer conjugates verbs (generates the verbal forms):

```
?- transduce(1, 11, [r, ê, v, e, r | L], Surface).
  L = [+, e, r, e, z],
  Surface = [r, ê, v, e, r, e, z]
```

Here is the Prolog code:

```
% arc(Start, End, LexicalChar, SurfaceChar)
%  describes the automaton

arc(1, 1, C, C) :- letter(C).
arc(1, 2, e, 0).   arc(2, 3, r, 0).  arc(3, 4, +, 0).
arc(4, 5, e, e).   arc(5, 6, r, r).  arc(6, 7, a, a).
arc(7, 8, i, i).   arc(7, 9, s, s).
arc(6, 10, e, e).  arc(10, 11, z, z).
arc(6, 12, o, o).  arc(12, 13, n, n).
arc(13, 14, s, s). arc(13, 15, t, t).


% final_state(S)
%  gives the stop condition

final_state(7).   final_state(8).   final_state(9).
final_state(11).  final_state(14).  final_state(15).


% letter(+L)
%  describes the French lower-case letters

letter(L) :-
  name(L, [Code]),
  97 =< Code, Code =< 122, !.
letter(L) :-
  member(L,
    [à, â, ä, ç, é, è, ê, ë, î, ï, ô, ö, ù, û,
    ü, 'œ']),
  !.
```

```
% transduce(+Start, ?Final, ?LexicalString,
%    ?SurfaceString)
%  describes the transducer. The first and second
%  rules include mute transitions and
%  enable to remove 0s

transduce(Start, Final, [U | LexicalString],
    SurfaceString) :-
  arc(Start, Next, U, 0),
  transduce(Next, Final,LexicalString,SurfaceString).
transduce(Start, Final, LexicalString,
    [S | SurfaceString]) :-
  arc(Start, Next, 0, S),
  transduce(Next, Final,LexicalString,SurfaceString).
transduce(Start, Final, [U | LexicalString],
    [S | SurfaceString]) :-
  arc(Start, Next, U, S),
  U \== 0,
  S \== 0,
  transduce(Next, Final,LexicalString,SurfaceString).
transduce(Final, Final, [], []) :-
  final_state(Final).
```

We can associate a final state to a part of speech. For instance, state 11 corresponds to the second-person plural of the future.

5.4.6 Ambiguity

In the transducer for future tense, there is no ambiguity. That is, a surface form has only one lexical form with a unique final state. This is not the case with the present tense (Table 5.28), and

(je) chante 'I sing'
(il) chante 'he sings'

have the same surface form but correspond respectively to the first- and third-person singular.

Table 5.28. Present tense of French verb *chanter*.

Number\Person	First	Second	Third
singular	*chante*	*chantes*	*chante*
plural	*chantons*	*chantez*	*chantent*

This corresponds to the transducer in Fig. 5.9, where final states 5 and 7 are the same. The implementation in Prolog is similar to that of the future tense. Using

backtracking, the transducer can yield all the final states reflecting the morphological ambiguity.

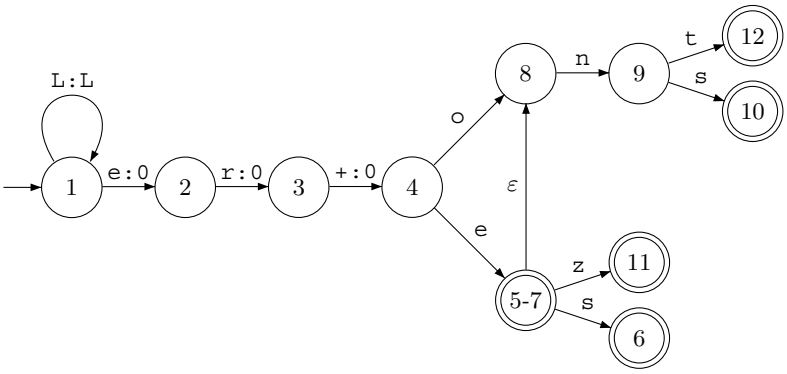

Fig. 5.9. A finite-state transducer encoding the present tense of verbs of the first group.

5.4.7 Operations on Finite-State Transducers

Finite-state transducers have mathematical properties similar to those of finite-state automata. In addition, they can be inverted and composed:

- Let T be transducer. The inversion T^{-1} reverses the input and output symbols of the transition function. The transition function of the transducer in Fig. 5.6 is then $\delta = \{\delta(q_0, z : a) = q_1, \delta(q_1, y : b) = q_1, \delta(q_1, x : c) = q_2\}$.
- Let T_1 and T_2 be two transducers. The composition $T_1 \circ T_2$ is a transducer, where the output of T_1 acts as the input of T_2.

Both the inversion and composition operations result in new transducers. This is obvious for the inversion. The proof is slightly more complex for the composition. Let $T_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ and $T_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ be two transducers. The composition $T_3 = T_1 \circ T_2$ is defined by $(\Sigma, Q_1 \times Q_2, \langle q_1, q_2 \rangle, F_1 \times F_2, \delta_3)$. The transition function δ_3 is built using the transition functions δ_1 and δ_2, and generating all the pairs where they interact (Kaplan and Kay 1994):

$$\delta_3(\langle s_1, s_2 \rangle, i, o) = \{\langle t_1, t_2 \rangle | \exists c \in \Sigma \cup \varepsilon, t_1 \in \delta_1(s_1, i, c) \wedge t_2 \in \delta_2(s_2, c, o)\}.$$

The inversion property enables transducers to operate in generating or parsing mode. They accept both surface and lexical strings. Each symbol of the first string is mapped to the symbol of the second string. So you can walk through the automaton and retrieve the lexical form from the surface form, or conversely, as we saw with the Prolog example.

Composition enables us to break down morphological phenomena. It is sometimes easier to formulate a solution then using intermediate forms between the surface and lexical forms. The correspondence between the word form and the sequence

of morphemes is not direct but is obtained as a cascade of transductions. Composition enables us to compact the cascade and to replace the transducers involved in it by a single one (Karttunen et al. 1992). We will see an example of it with French irregular verbs in Sect. 5.5.3.

5.5 Morphological Rules

5.5.1 Two-Level Rules

Originally, Koskenniemi (1983) used declarative rules to describe morphology. These two-level rules enumerate the correspondences between lexical characters and surface ones and the context where they occur. Context corresponds to left and right characters of the current character and can often be expressed in terms of vowels (*V*) or consonants (*C*).

In the two-level formalism, a rule is made of a correspondence pair (`lexical:surface`), a rule operator, and the immediate left and right context. Operators can be $\Rightarrow$, $\Leftarrow$, $\Leftrightarrow$, or $/\Leftarrow$, and mean respectively only in that context, always in that context, always and only, and never in that context. Left and right contexts where the rule applies are separated by the symbol __ (Table 5.29).

Table 5.29. Two-level rules.

Rules	Description
`a:b` $\Rightarrow$ `lc __ rc`	`a` is transduced as `b` **only** when it has `lc` to the left and `rc` to the right
`a:b` $\Leftarrow$ `lc __ rc`	`a` is **always** transduced as `b` when it has `lc` to the left and `rc` to the right
`a:b` $\Leftrightarrow$ `lc __ rc`	`a` is transduced as `b` **always and only** when it has `lc` to the left and `rc` to the right
`a:b` $/\Leftarrow$ `lc __ rc`	`a` is **never** transduced as `b` when it has `lc` to the left and `rc` to the right

In English, the comparative *happier* is decomposed into two morphemes *happy* + *er*, where the lexical y corresponds to a surface i (Table 5.30). This correspondence occurs more generally when y is preceded by a consonant and followed by *-er*, *-ed*, or *-s*. This can be expressed by three rules, where `C` represents any consonant:

1. `y:i` $\Leftarrow$ `C:C __ +:0 e:e r:r`
2. `y:i` $\Leftarrow$ `C:C __ +:e s:s`
3. `y:i` $\Leftarrow$ `C:C __ +:0 e:e d:d`

Once written, all the rules are applied in parallel. This parallel application is the main distinctive feature of the two-level morphology compared with other, older models. This means that when processing a string, every rule must be successfully

Table 5.30. The `y:i` transduction rules.

Examples	`happy+er`	`party+s`	`marry+ed`
	`happi0er`	`parties`	`marri0ed`
Rules	`Cy+er`	`Cy+s`	`Cy+ed`
	`Ci0er`	`Cies`	`Ci0ed`

applied to the current pair of characters `lexical:surface` before moving to the next pair (Fig. 5.10).

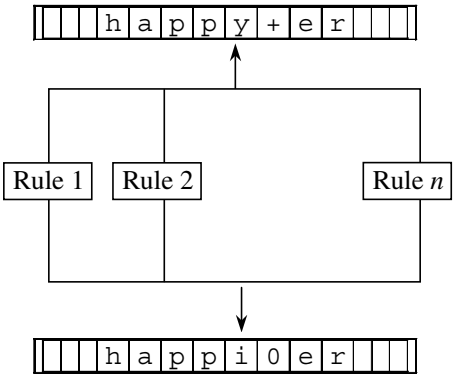

Fig. 5.10. Applying the rules in parallel.

The left and right contexts of a rule can use a wildcard, the ANY symbol `@`, which stands for any alphabetical character, as in

`y:x` ⇐ `__` `@:c`

This rule means that a lexical `y` corresponds to a surface `x` when it is before a surface `c`. The corresponding lexical character in the right context is not specified in the rule, however, the unspecified character represented by the ANY symbol must be compatible with the correspondence rule that can apply to it. The ANY symbol is not, strictly speaking, any character then, but any character so that it forms a "feasible pair", here with `c`.

5.5.2 Rules and Finite-State Transducers

It has been demonstrated that any two-level rule can be compiled into an equivalent transducer (Johnson 1972, Kaplan and Kay 1994). Rule 1, for instance, corresponds to the automaton in Fig. 5.11, where the pair `@:@` denotes any pair that cannot pass the other transitions.

In practice, morphological phenomena are easier to describe and to understand using individual rules rather than writing a complex transducer. For this reason, the

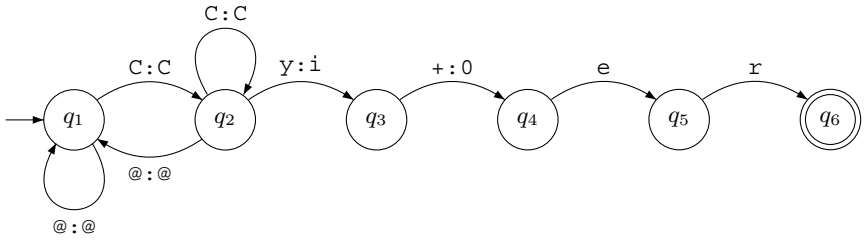

Fig. 5.11. A transducer to parse the `y:i` correspondence.

development of parsers based on the two-level method uses this strategy (Karttunen 1994). It consists in writing a collection of rules to model a language's morphology and compiling them into as many transducers. The parallel transducers are then combined into a single one using the transducer intersection (Fig. 5.12).

However, while the intersection of two finite automata defines a finite-state automaton, it is not always the case for finite-state transducers. Kaplan and Kay (1994) demonstrated that when surface and lexical pairs have the same length – without ε – the intersection is a transducer. This property is sufficient to intersect the rules in practical applications. In fact, transducers obtained from two-level rules are intersected by treating the ε symbol as an ordinary symbol (Beesley and Karttunen 2003, p. 55). Parallel application of rules or the transducer intersection removes one of their major harmful the side effects: their application outside of their intended context.

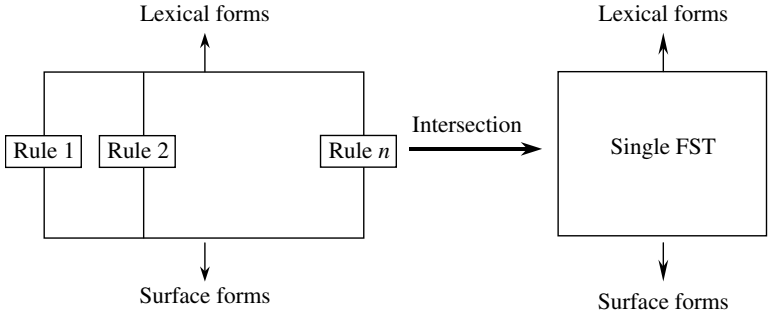

Fig. 5.12. A set of two-level rules intersected into a single FST. After Karttunen et al. (1992).

Originally, rules were compiled by hand. However, this problem can quickly become intractable, especially when it comes to managing conflicting rules or when rule contexts interfere with transduced symbols. To solve it, we can use a compiler that creates transducers automatically from two-level rules. The Xerox XFST is an example of it. It is a publicly available tool, and to date it is the only serious implementation of a morphological rule compiler.

5.5.3 Rule Composition: An Example with French Irregular Verbs

When developing a complete morphological parser, it is often convenient to introduce intermediate levels between the lexical and surface strings. This is especially true when the lexical and surface forms are distant and involve complex morphological relations. Intermediate levels enable us then to decompose the morphological system into smaller parts that are easier to treat.

Chanod (1994) gives an example of decomposition with the notoriously difficult morphology of French irregular verbs (Bescherelle 1980). The French verb system has about 100 models of inflection – paradigms. Two of them are said to be regular, the first and second group, and gather the vast majority of the verbs. The third group is made of irregular verbs and gathers the rest. The irregular group contains the most frequent verbs: *faire* 'do', *savoir* 'know', *connaître* 'know', *dormir* 'sleep', *courir* 'run', *battre* 'beat', *écrire* 'write', etc.

Table 5.31 shows the conjugation of some irregular verbs. We can see that there is a set of regular suffixes: *s*, *s*, *t*, *ons*, *ez*, and *ent*, and that most irregularities, also called alternations, occur at the junction of the stem and the suffix. The stem and suffix can be directly concatenated, as in *courir*, but not in *dormir*, *peindre*, or *battre*.

Table 5.31. Conjugation of irregular French verbs, present tense. *Courir* has regular suffixes in underlined bold characters. In the other verbs, irregular infections are shown in bold characters.

Infinitive	**courir**	**dormir**	**battre**	**peindre**	**écrire**
First person singular	cour**s**	**dors**	**bats**	**peins**	écris
Second person singular	cour**s**	**dors**	**bats**	**peins**	écris
Third person singular	cour**t**	**dort**	**bat**	**peint**	écrit
First person plural	cour**ons**	dormons	battons	peignons	**écrivons**
Second person plural	cour**ez**	dormez	battez	peignez	**écrivez**
Third person plural	cour**ent**	dorment	battent	peignent	**écrivent**

Although apparently complex, general rules can model these alternations using local contexts corresponding to specific substrings. In the case of *dormir*, a general principle in French makes it impossible to have an *m* followed by an *s* or *t*. It then must be deleted in the three singular persons. For *battre*, the pairs *tt* or *dt* do not occur in the end of a word or before a final *s*. Such rules are not tied to one specific verb but can be applied across a variety of inflection paradigms and persons. Figure 5.5.3 shows the rule sequence that produces the correct surface form of *dors*.

The verbs *peindre* and *écrire* are more complex cases because their conjugation uses two stems: *pein* and *peign* – *écri* and *écriv*. Chanod (1994) solves these difficulties using a transduction between the infinitive and a first intermediate form that will then be regular. Then *peindre*+IndP+SG+P1 is associated to *peign*+IndP+SG+P1, and *écrire*+IndP+SG+P1 to *écriv*+IndP+SG+P1. The second intermediate form uses two-level rules to obtain the correct surface forms: *v* or *gn* must be followed by a vowel or deleted (Fig. 5.5.3). The rule that Chanod uses is, in fact:

Lexical form: stem	dormir	+IndP +SG +P1
Intermediate form: inflection	dorm	+IndP +SG +P1
Intermediate form:: deletion of *m* followed by *s*	dorm	s
Surface form:	dor	s

Fig. 5.13. Sequence of rules applied to *dormir*. After Chanod (1994).

```
n:0 ⇔ g __ [s|t]
```

Lexical form: stem	peindre	+IndP +SG +P1
Intermediate form: inflection	peign	+IndP +SG +P1
Intermediate form: Depalatalisation of *gn*	peign	s
Surface form:	pein	s

Fig. 5.14. Sequence of rules applied to *peindre*. After Chanod (1994).

The FST resulting from the surface, lexical, and intermediate levels are ultimately combined with the lexicon and composed into a single transducer (Fig. 5.15).

5.6 Application Examples

The Xerox language tools give a good example of what morphological parsers and part-of-speech taggers can do. These parsers are available for demonstration on the Internet using a Web browser. Xerox tools let you enter English, French, German, Italian, Portuguese, and Spanish words, and the server returns the context-free morphological analysis for each term (Tables 5.32–5.34). You can also type in phrases or sentences and Xerox taggers will disambiguate their part of speech. In addition to demonstrations, Xerox lists examples of industrial applications that make use of its tools.

5.7 Further Reading

Dionysius Thrax fixed the parts of speech for Greek in the 2nd century BCE. They have not changed since and his grammar is still interesting to read, see Lallot (1998).

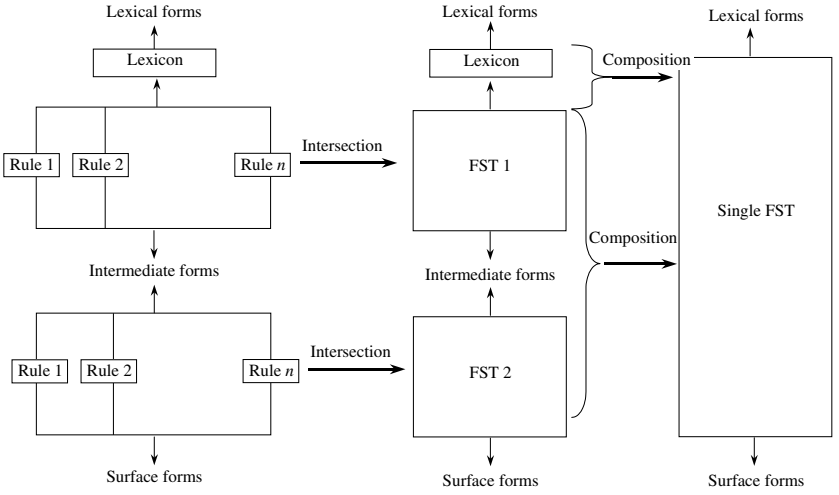

Fig. 5.15. Intersection and composition of finite-state transducers. After Karttunen (1994).

Table 5.32. Xerox morphological parsing in English.

Input Term(s): works	Input Term(s): round	Input Term(s): this
work+Vsg3	round+Vb	this+Psg
work+Npl	round+Prep	this+Dsg
	round+Adv	this+Adv
	round+Adj	
	round+Nsg	

Table 5.33. Xerox morphological parsing in French.

Input Term(s): <'>etions	Input Term(s): porte
étions 1	porte 6
être+IndI+PL+P1+Verb	porter+SubjP+SG+P1+Verb
	porter+SubjP+SG+P3+Verb
	porter+Imp+SG+P2+Verb
	porter+IndP+SG+P1+Verb
	porter+IndP+SG+P3+Verb
	porte+Fem+SG+Noun

Table 5.34. Xerox morphological parsing in German.

Input Term(s): arbeite	Input Term(s): die
arbeiten+V+IMP+PRÄS+SG2	die+ART+DEF+PL+AKK
arbeiten+V+IND+PRÄS+SG1	die+ART+DEF+PL+NOM
arbeiten+V+KONJ+PRÄS+SG1	die+ART+DEF+SG+AKK+FEM
arbeiten+V+KONJ+PRÄS+SG3	die+ART+DEF+SG+NOM+FEM
	die+PRON+DEM+PL+AKK
	die+PRON+DEM+PL+NOM
	die+PRON+DEM+SG+AKK+FEM
	die+PRON+DEM+SG+NOM+FEM
	die+PRON+RELAT+PL+AKK
	die+PRON+RELAT+PL+NOM
	die+PRON+RELAT+SG+AKK+FEM
	die+PRON+RELAT+SG+NOM+FEM

A short and readable introduction in French to the history of parts of speech is Ducrot (1995).

Accounts on finite-state morphology can be found in Sproat (1992) and Ritchie et al. (1992). Roche and Schabes (1997) is useful book that describes fundamental algorithms and applications of finite-state machines in language processing, especially for French. Kornai (1999) covers other aspects and languages. Kiraz (2001) on the morphology of Semitic languages: Syriac, Arabic, and Hebrew. Beesley and Karttunen (2003) is an extensive description of the two-level model in relation with the Xerox tools. It contains a CD-ROM with the Xerox rule compiler.

Antworth (1995) provides a free implementation of KIMMO named PC-KIMMO 2 with source and executable programs. The system is available from the Internet (http://www.sil.org). It comes with an English lexicon and English morphological rules. It is open to extensions and modifications. General-purpose finite-state transducers toolkits are also available. They include the FSA utilities (van Noord and Gerdemann 2001), the FSM library (Mohri et al. 1998), and Unitex (http://www-igm.univ-mlv.fr/~unitex/).

Exercises

5.1. Find a dictionary on the Web in English, French, German, or another language you would like to study and extract all the articles, conjunctions, prepositions, and pronouns.

5.2. Implement a morphological parser to analyze regular plurals of nouns in English or French.

5.3. Add a lexical look-up to Exercise 5.2.

5.4. Implement a morphological parser to analyze plurals of nouns in English or French, taking a list of exceptions into account.

5.5. Implement a morphological parser to analyze regular preterits of verbs in English or German.

5.6. Implement a morphological parser to conjugate French verbs of first group in the imperfect tense.

5.7. Implement a morphological parser to conjugate regular German verbs in the present tense.

5.8. Build a morphological parser implementing regular English verb inflection: *-s*, *-ed*, *-ing*.

5.9. Some verbs have their final *-e* deleted, for instance, *chase* (*chase+ed, chase+ing*). In the KIMMO formalism, the *-e* deletion rule is expressed as e:0 $\Leftrightarrow$ C:C __ 0:+ V:V. Draw the corresponding transducer and write the Prolog rules that will parse these verbs.

5.10. Break the following words into morphemes: *computer*: *computers*, *computerize, computerization*, *recomputerize*.

5.11. Build a morphological parser that will parse words derived from *computer*: *computers*, *computerize, computerization*, *recomputerize*.

5.12. Break the following words into morphemes: *chanter*: *enchanter*, *rechanter*, *déchanter*, *désenchanter*.

5.13. Build a morphological parser that will parse words derived from *chanter*: *enchanter*, *rechanter*, *déchanter*, and *désenchanter*.