A Short Introduction to Formal Syntax and Morphology

Sylvain Schmitz
LSV, ENS Cachan & CNRS
March 9, 2011 (r1315M)

These notes cover the *first* part of an introductory course on computational linguistics, also known as **MPRI 2-27-1**: *Logical and computational structures for linguistic modeling*. Among their prerequisites are

- classical notions of formal language theory, in particular regular and context-free languages, and more generally the Chomsky hierarchy,
- a basic command of English and French morphology and syntax, in order to understand the examples;
- some acquaintance with logic and proof theory also is advisable, and is at any rate an actual prerequisite for the *second* part of this course, which covers semantics and discourse.

Several courses at MPRI provide an in-depth treatment of subjects we can only hint at. The interested student might consider attending

MPRI 1-18: *Tree automata and applications*,

MPRI 2-16: *Finite automata modelization*, and

MPRI 2-1: *Linear logic*.

Contents

Further Reading

Interested students will find a good general textbook on natural language processing in Jurafsky and Martin (2009). The present notes have a strong bias towards formal language theory—reference textbooks in this domain include (Harrison, 1978; Berstel, 1979; Sakarovitch, 2009; Comon et al., 2007)—, but this is hardly representative of the general field of natural language processing and computational linguistics. In particular, the overwhelming importance of statistical approaches in the current body of research makes the textbook of Manning and Schütze (1999) another recommended reference.

The main journal of natural language processing is *Computational Linguistics*. As often in computer science, the main conferences of the field have equivalent if not greater importance than journal outlets, and one will find among the major conferences *ACL* ("Annual Meeting of the Association for Computational Linguistics"), *EACL* ("European Chapter of the ACL"), *NAACL* ("North American Chapter of the ACL"), and *CoLing* ("International Conference on Computational Linguistics"). A very good point in favor of the ACL community is their early adoption of open access; one will find all the ACL publications online at `http://www.aclweb.org/anthology/`.

Notations

We use the following notations in this document. First, as is customary in linguistic texts, we prefix agrammatical or incorrect examples with an asterisk, like

**ationhospitalmis* or **sleep man to is the*.

These notes also contain some exercises, and a difficulty appreciation is indicated as a number of asterisks in the margin next to each exercise—a single asterisk denotes a straightforward application of the definitions.

Relations. We only consider binary **relations**, i.e. subsets of $A \times B$ for some sets A and B (although the treatment of e.g. rational relations in Section 1.2.1 can be generalized to n-ary relations). The **inverse** of a relation R is $R^{-1} = \{(b, a) \mid (a, b) \in R\}$, its **domain** is $R^{-1}(B)$ and its **range** is $R(A)$. Beyond the usual union, intersection and complement operations, we denote the **composition** of two relations $R_1 \subseteq A \times B$ and $R_2 \subseteq B \times C$ as $R_1 \fatsemi R_2 = \{(a, c) \mid \exists b \in B, (a, b) \in R_1 \wedge (b, c) \in R_2\}$. The **reflexive transitive closure** of a relation is noted $R^{\circledast} = \bigcup_i R^i$, where $R^0 = \mathrm{Id}_A = \{(a, a) \mid a \in A\}$ is the **identity** over A, and $R^{i+1} = R \fatsemi R^i$.

Monoids. A **monoid** $\langle \mathbb{M}, \cdot, 1_{\mathbb{M}} \rangle$ is a set of elements $\mathbb{M}$ along with an associative operation $\cdot$ and a neutral element $1_{\mathbb{M}} \in \mathbb{M}$. We are often dealing with the **free monoid** $\langle \Sigma^*, \cdot, \varepsilon \rangle$ generated by concatenation $\cdot$ of elements from a finite set Σ. A monoid is **commutative** if $a \cdot b = b \cdot a$ for all a, b in $\mathbb{M}$.

We lift $\cdot$ to subsets of $\mathbb{M}$ by $L_1 \cdot L_2 = \{m_1 \cdot m_2 \mid m_1 \in L_1, m_2 \in L_2\}$. Then for $L \subseteq \mathbb{M}$, $L^0 = \{1_{\mathbb{M}}\}$ and $L^{i+1} = L \cdot L^i$, and we define the **Kleene star** operator by $L^* = \bigcup_i L^i$.

Semirings. A **semiring** $\langle \mathbb{K}, \oplus, \odot, 0_{\mathbb{K}}, 1_{\mathbb{K}} \rangle$ is endowed with two binary operations, an addition $\oplus$ and a multiplication $\odot$ such that

- $\langle \mathbb{K}, \oplus, 0_{\mathbb{K}} \rangle$ is a commutative monoid for addition with $0_{\mathbb{K}}$ for neutral element,
- $\langle \mathbb{K}, \odot, 1_{\mathbb{K}} \rangle$ is a monoid for multiplication with $1_{\mathbb{K}}$ for neutral element,
- multiplication distributes over addition, i.e. $a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c)$ and $(a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$ for all a, b, c in $\mathbb{K}$,
- $0_{\mathbb{K}}$ is a zero for multiplication, i.e. $a \odot 0_{\mathbb{K}} = 0_{\mathbb{K}} \odot a = 0_{\mathbb{K}}$ for all a in $\mathbb{K}$.

Among the semirings of interest are the

- boolean semiring $\langle \mathbb{B}, \vee, \wedge, 0, 1 \rangle$ where $\mathbb{B} = \{0, 1\}$,
- probabilistic semiring $\langle \mathbb{R}_+, +, \cdot, 0, 1 \rangle$ where $\mathbb{R}_+ = [0, +\infty)$ is the set of positive reals (sometimes restricted to $[0, 1]$ when in presence of a probability distribution),
- tropical semiring $\langle \mathbb{R}_+ \uplus \{+\infty\}, \min, +, +\infty, 0 \rangle$,
- rational semiring $\langle \mathrm{Rat}(\Delta^*), \cup, \cdot, \emptyset, \{\varepsilon\} \rangle$ where $\mathrm{Rat}(\Delta^*)$ is the set of rational sets over some alphabet Δ.

String Rewrite Systems. A **string rewrite system** or **semi-Thue systems** over an alphabet Σ is a relation $R \subseteq \Sigma^* \times \Sigma^*$. The elements (u, v) of R are called **string rewrite rules** and noted $u \to v$. The **one step derivation relation** generated by R, noted $\overset{R}{\Rightarrow}$, is the relation over Σ^* defined for all w, w' in Σ^* by $w \overset{R}{\Rightarrow} w'$ iff there exist x, y in Σ^* such that $w = xuy$, $w' = xvy$, and $u \to v$ is in R. The **derivation relation** is the reflexive transitive closure $\overset{R}{\Rightarrow}^{\circledast}$.

See also the monograph by Book and Otto (1993).

Prefixes. The **prefix ordering** $\leq_{\text{pref}}$ over Σ^* is defined by $u \leq_{\text{pref}} v$ iff there exists v' in Σ^* such that $v = uv'$. We note $\text{Pref}(v) = \{u \mid u \leq_{\text{pref}} v\}$ the set of prefixes of v, and $u \wedge v$ the longest common prefix of u and v.

Term Rewrite Systems. For ranked alphabets, trees, terms, contexts, substitutions, term rewrite systems, etc., see Comon et al. (2007).

Chapter 1

Morphology

We consider in this chapter how to represent sets of words of a natural language in linguistically meaningful and computationally efficient ways.

The purpose of morphology is to describe the mechanisms that underlie the formation of words. Intuitively, one can recognize the existence of a relation between the words *sings* and *singing*, and further find that the same relation holds between *dances* and *dancing*. A **morphological analysis** of these words

- splits them into basic components, called **morphemes**: here the **stems** *sing* and *dance*, and the **affixes** *-s* and *-ing*, standing for singular third person and present participle, and thus
- recognizes them as **inflected forms** of the **lemmas** *to sing* and *to dance*.

Already observe some difficulties in the word formation rules: the realization of the present participle of *dance* is not **danceing*; and some words may be outright irregular, e.g. *sang* and *sung* for the preterit and past participle forms of *to sing*. We will consider formal systems describing the derivation of words.

Beyond the simple enumeration of words, we usually want to retrieve some linguistic information that will be helpful for further processing: are we dealing with a noun or a verb (its **category**)? is it plural or singular (its **number**)? what is its **part-of-speech** (**POS**) tag? etc. Table 1.1 illustrates the kind of information one can expect to find in a morphological lexicon. If our rules are well-designed, we should be able to extract this information from the various derivations that account for the word.

Table 1.1: Example of entries in English along with their POS tags (from the Penn Treebank tagset) and some morphological features.

Input	Lemma	POS tag	Features
race	race	NN	[cat=n; num=sg; case=nom|acc]
races	race	NNS	[cat=n; num=pl; case=nom|acc]
race	to race	VB	[cat=v; mode=inf]
race	to race	VBP	[cat=v; pers=1|2; num=sg|pl; tense=pres; mode=ind]
race	to race	VBP	[cat=v; pers=3; num=pl; tense=pres; mode=ind]
races	to race	VBZ	[cat=v; pers=3; num=sg; tense=pres; mode=ind]
race	to race	VB	[cat=v; pers=2; num=sg; tense=pres; mode=imp]
raced	to race	VBD	[cat=v; tense=past; mode=ind]
raced	to race	VBN	[cat=v; mode=ppart]
racing	to race	VBG	[cat=v; mode=ger]

1.1 *Background:* Linguistic Aspects

1.1.1 A Bit of English Morphology

Most of the discussion is based on (Jurafsky and Martin, 2009, Section 3.1).

Morphology is the study of the rules of word composition from their basic meaning-bearing elements, known as **morphemes**.

One usually distinguishes between the main morphemes, called **stems** (like *sing*, *dance*, etc.), that carry the main meaning, from **affixes** (like *-s*, *-ing*, etc.). Four main types of composition rules are commonly considered, and we will briefly review them in the following.

Inflection. Inflectional morphology composes a word stem along with a grammatical morpheme. **Inflection** can mark various syntactic features like case, tense, mood, genre, or number.

The various inflected forms of *race* and *to race* in Table 1.1 show the regular inflections of nouns and verbs in English: the *-s* plural marking for nouns, and the *-s* present third person, *-ed* preterit or past participle, and *-ing* present participle for verbs. Short gradable adjectives can take a comparative suffix *-er* (as in *cuter*) and a superlative suffix *-est* (as in *cutest*).

The regular rules are **productive** in the sense that new-formed words fall prey to them, for instance *to twit/twits/twitted/twitting*. In contrast, there are few irregular nouns and verbs, but they tend to be very frequent words. For nouns, *ox/oxen*, *mouse/mice*, *sheep/sheep* are a few examples, and for verbs *to sing/sang/sung*, *to eat/ate/eaten*, or *to cut/cut/cut*.

Derivation. A combination of a stem with an affix that results in a different lemma is called a **derivation**. The obtained word is often of a different category—e.g. from noun to verb with *hospital* and *hospitalize*, and back to noun with *hospitalization*—, but this is not mandatory—e.g. *pseudohopitalization*. Beyond prefixes and suffixes, English can employ expletives such as *bloody*, *motherfuckin(g)*, *sodding* etc. as infixes: *absobloominglutely*, *Massafriggingchussets*.

See McCarthy (1982) on the composition rules for infixed expletives.

Compounding. Like derivation, **compounding** results in a different lemma, but links several stems: *doghouse*, *bed-time*, *rock 'n' roll*, *bull's eye*, etc. We will not treat compounding, which in practical processing is mostly a **tokenization** issue. Note that there are also finite-state approaches to tokenization (see e.g. Karttunen et al., 1996, Section 4.1).

Check Zwicky (1985) to get a better idea of what a clitic is.

Cliticization. A **clitic** is a morpheme that acts syntactically like a word, but is bound to another word like an affix. English has auxiliary verbs that may become simple clitics: *has/'s*, *have/'ve*, *had/'d*, *am/'m*, *is/'s*, *are/'re*, *will/'ll*, and *would/'d*. Such simple clitics are usually replaced by their expanded forms prior any further processing by the tokenizer.

The possessive marker *'s* can also be seen as a special clitic, only applicable to nouns.

Orthographic Rules. In addition to morpheme composition rules, a morphological description has to take some **orthographic rules** into account. These are often caused by phonological issues.

An *-s* suffix is turned into *-es* in *ibises*, *waltzes*, *thrushes*, *finches*, *boxes* for nouns, and similarly *tosses*, *waltzes*, *washes*, *catches*, *boxes* for verbs. An ending *y* is turned into *ies* in *butterflies* and *tries*. Regarding *-ed* and *-ing* suffixes, ending consonant

letters are doubled as in *begging*, while silent ending *e*'s are deleted as in *dancing*.

Other examples of orthographic rules include *in-* prefixes before some consonants (*b*, *p*, *m*) turning into *im-*, e.g. *impractical*, but this does not apply to *un-* prefixes (*unperturbable*).

A Formal Approach Let us define the **morphological analysis** problem as the problem, given a single word in isolation, of recovering all its possible structures and morphological features. The exact formulation of course depends on how word structures and morphological features are formalized; we will consider a particular case where a word is decomposed into a sequence of stems, affixes, and feature structures. For instance, from *hospitalized*, we want to recover the sequence $\mathrm{hopital{+}ize{+}ed[cat{=}v; mode{=}ppart]}$. Note that we also want to recover the sequence $\mathrm{hopital{+}ize{+}ed[cat{=}v; tense{=}past; mode{=}ind]}$, i.e. we need to account for ambiguity.

To solve the morphological analysis problem, if the set of words is finite, we can store all the forms in a plain dictionary and simply lookup the various entries. A much more efficient structure is a *trie* or a directed acyclic graph with the word structure and morphological information attached to the nodes.

Of course, a finite set approach is linguistically unsatisfying: limits to affix stacking are more of a performance issue, and are easily violated by extreme or playful uses, like *antidisestablishmentarianism* or **mystery-y-ish-y*. In order to represent infinite sets of words, we switch naturally to automata with outputs, i.e. **transducers**. We first review some basic results on transducers in Section 1.2.1, before returning to morphological analysis in Section 1.2.2.

See Zwicky and Pullum (1987) for a discussion of "playful" morphology.

1.1.2 Part-of-Speech Tags

Part-of-speech tags are refinements of the usual, basic categories like *noun* or *verb*. Different sets of tags (or **tagsets**) will provide different amounts of morphological or syntactic information about a word; for instance, one can see in Table 1.1 that, in the Penn Treebank tagset (Marcus et al., 1993), VBP tags verbs in present tense, indicative mode, except for the third person singular case, which uses the VBZ tag. Table 1.2 presents the 36 POS tags of the Penn Treebank tagset, 12 further tags for punctuation and currency symbols being omitted.

The best source on the Penn Treebank tagset are the tagging guidelines used by the annotators of the Penn Treebank project (Santorini, 1990). Beware that in those early guidelines NNP, NNPS and PRP were noted NP, NPS and PP.

By **POS tagging** we refer to the process of associating a POS tag to each word of a sentence. There are two important differences with the morphological analysis problem:

1. we only care about the POS tag, not about other morphological information,
2. the tagging of a word depends on its surrounding context: we should take it into account in order to accurately disambiguate between different possible tags.

For instance, we have several possible tags for *hospitalized*, but the tagging is unambiguous in the context of

> He/PRP hospitalized/VBD his/PP$ mother/NN ./.
> His/PP$ mother/NN was/VBD hospitalized/VBN on/IN Saturday/NNP ./.
> He/PRP 's/VBZ visiting/VBG his/PP$ hospitalized/JJ mother/NN ./.

Note that syntactic context is not always enough:

Table 1.2: The Penn Treebank POS tagset, punctuation excepted (Marcus et al., 1993).

Tag	POS	Tag	POS
CC	Coordinating conjunction	PP$	Possessive pronoun
CD	Cardinal number	RB	Adverb
DT	Determiner	RBR	Adverb, comparative
EX	Existential *there*	RBS	Adverb, superlative
FW	Foreign word	RP	Particle
IN	Preposition or subordinating conjunction	SYM	Symbol
JJ	Adjective	TO	*to*
JJR	Adjective, comparative	UH	Interjection
JJS	Adjective, superlative	VB	Verb, base form
LS	List item marker	VBD	Verb, past tense
MD	Modal	VBG	Verb, gerund or present participle
NN	Noun, singular or mass	VBN	Verb, past participle
NNP	Proper noun, singular	VBP	Verb, present, non-3rd person singular
NNPS	Proper noun, plural	VBZ	Verb, 3rd person present
NNS	Noun, plural	WDT	Wh-determiner
PDT	Predeterminer	WP	Wh-pronoun
POS	Possessive ending	WP$	Possessive wh-pronoun
PRP	Personal pronoun	WRB	Wh-adverb

*Gator/NN attacks/VBZ puzzle/NN experts/NNS
Gator/NN attacks/NNS puzzle/VBP experts/NNS

In fact, newspaper headlines—like the previous example, from *AOL News*, spotted in a *New York Times* "On Language" column—can be quite puzzling. Another example, from *The Guardian*, spotted on *Language Log*:

May/NNP axes/VBZ Labour/NNP police/NN beat/NN pledge/NN

where *May*—Theresa May, British Home Secretary—could also be a MD, *axes* a NNS, and each of *Labour*, *police*, *beat*, and *pledge* VBPs. A last example, from the *BBC news*, also spotted on *Language Log*:

Council/NN hires/VBZ ban/NN bid/NN taxi/NN firm/NN

where *hires* could also be a NNS, each of *ban*, *bid*, and *taxi* could VBPs, and *firm* a JJ. This illustrates the amount of ambiguity that POS taggers have to cope with.

The POS tagging task can in fact be decomposed into two subtasks:

1. training a POS tagger from already-tagged data, for instance the annotated texts from the *Wall Street Journal* present in the Penn Treebank corpus, which represent approximately 50,000 sentences and 1,2 million tokens, and
2. using the tagger on tokenized text.

We describe two finite-state approaches for tackling the POS tagging tasks in Section 1.3.

1.2 Finite-State Morphology

1.2.1 *Background:* Rational Transductions

Whereas the theory of regular languages is typically presented on rational and recognizable subsets of Σ^* for Σ a finite alphabet, with Kleene's Theorem stating their

equality, the landscape of results changes on products of monoids, for instance on $\Sigma^* \times \Delta^*$ for Σ and Δ two finite alphabets: rational and recognizable sets do not coincide. Nevertheless, rational subsets of $\Sigma^* \times \Delta^*$, aka **rational relations**, have an operational presentation through finite-state devices, namely **finite transducers**.

We only review basic results on rational relations, and briefly mention two subclasses with applications in computational morphology, namely the **length preserving relations** and **sequential functions**. We will also draw parallels in Section 1.3.2 between hidden Markov models, which are probabilistic models commonly employed in statistical language processing, and **recognizable series**.

*To learn more, go attend **MPRI 2-16** or check the textbooks of Berstel (1979), Sakarovitch (2009), and Berstel and Reutenauer (2010).*

Rational Relations Observe that $\langle \Sigma^* \times \Delta^*, \cdot, (\varepsilon, \varepsilon) \rangle$ with $(u, v) \cdot (u', v') = (uu', vv')$ is a monoid, generated by $(\{\varepsilon\} \times \Delta^*) \cup (\Sigma^* \times \{\varepsilon\})$, but not *freely* generated (e.g. $(a, b) = (a, \varepsilon) \cdot (\varepsilon, b) = (\varepsilon, b) \cdot (a, \varepsilon)$). We use $u{:}v$ as a shorthand for $(u, v) \in \Sigma^* \times \Delta^*$.

The rational subsets of $\Sigma^* \times \Delta^*$ are defined as per usual: the finite subsets are rational, and we take their closure by union, concatenation and Kleene star. Note that subsets of $\Sigma^* \times \Delta^*$ are relations, hence the name **rational relations** between Σ^* and Δ^*. We can also define **rational expressions** over $\Sigma^* \times \Delta^*$ using the abstract syntax

$$E ::= \emptyset \mid \varepsilon{:}\varepsilon \mid a{:}\varepsilon \mid \varepsilon{:}b \mid E^* \mid E_1 + E_2 \mid E_1 \cdot E_2$$

with a in Σ and b in Δ.

A **finite-state transducer** is a finite automaton over $\Sigma^* \times \Delta^*$: $\mathcal{T} = \langle Q, \Sigma^* \times \Delta^*, \delta, I, F \rangle$ with Q a finite set of states, $\delta \subseteq Q \times \Sigma^* \times \Delta^* \times Q$ a finite transition relation, and I and F the initial and final subsets of Q. The behavior of $\mathcal{T}$ is a relation in $\Sigma^* \times \Delta^*$ defined by

$$[\![\mathcal{T}]\!] = \{u{:}v \mid \delta(I, u{:}v) \cap F \neq \emptyset\}$$

and is called a **rational transduction**. Without loss of generality, we can always assume our finite transducers to be **normalized**, i.e. to be over $(\{\varepsilon\} \times \Delta^*) \cup (\Sigma^* \times \{\varepsilon\})$.

Exercise 1.1. Show that the range $R(\Sigma^*)$ of a rational transduction R is a rational language. **(∗)**

Exercise 1.2. Show that if L is a rational language over Σ, then Id_L is a rational transduction over $\Sigma^* \times \Sigma^*$. **(∗)**

Exercise 1.3. Show that rational transductions are closed under inverse and composition. **(∗∗)**

Exercise 1.4. Let R be a relation over $\Sigma^* \times \Delta^*$. Show that R is a rational relation iff it is a rational transduction. **(∗∗)**

Remark 1.1 (Non-closure)**.** Rational relations are not closed under intersection:

$$\{c^n{:}a^n b^m \mid m, n \geq 0\} \cap \{c^n{:}a^m b^n \mid m, n \geq 0\} = \{c^n{:}a^n b^n \mid n \geq 0\}$$

has a non-rational language $\{a^n b^n \mid n \geq 0\}$ for range. Thus rational relations are not closed under complement either.

Similarly, $R = \mathrm{Id}_{\{a,b\}^*} \cdot ba{:}ab \cdot \mathrm{Id}_{\{a,b\}^*}$ is a rational relation, but its reflexive transitive closure $R^{\circledast}$ is not: let $L = \{(ab)^n \mid n \geq 0\}$, then

$$\mathrm{Id}_L \fatsemi R^{\circledast}(\{a, b\}^*) \cap \{a^n b^m \mid m, n \geq 0\} = \{a^n b^n \mid n \geq 0\}$$

is not a rational language.

Sequential Functions The equivalence of deterministic and nondeterministic finite state automata breaks when entering the realm of rational relations. The closest substitute for a deterministic transducer is called a **sequential transducer**. Formally, a sequential transducer from Σ to Δ is a tuple $\mathcal{T} = \langle Q, \Sigma, \Delta, q_0, \delta, \eta, \iota, \rho \rangle$ where $\delta : Q \times \Sigma \to Q$ is a partial transition function, $\eta : Q \times \Sigma \to \Delta^*$ a partial transition output function with the same domain as δ, $\iota \in \Delta^*$ is an initial output, and $\rho : Q \to \Delta^*$ is a partial final output function. $\mathcal{T}$ defines a partial **sequential function** $[\![\mathcal{T}]\!] : \Sigma^* \to \Delta^*$ with

Historically, what we call here "sequential" was named "subsequential" by Schützenberger (1977), but we follow the more recent practice initiated by Sakarovitch (2009).

$$[\![\mathcal{T}]\!](w) = \iota \cdot \eta(q_0, w) \cdot \rho(\delta(q_0, w))$$

for all w in Σ^* for which $\delta(q_0, w)$ and $\rho(\delta(q_0, w))$ are defined, where $\eta(q, \varepsilon) = \varepsilon$ and $\eta(q, wa) = \eta(q, w) \cdot \eta(\delta(q, w), a)$ for all w in Σ^* and a in Σ.

(**) Exercise 1.5. Show that sequential transducers are closed under composition.

Normalization. Let us note $\mathcal{T}_{(q)}$ for the sequential transducer with q for initial state. The longest common prefix of all the outputs from state q can be written formally as $\bigwedge_{v \in \Sigma^*} [\![\mathcal{T}_{(q)}]\!](v)$. A sequential transducer is **normalized** if this value is ε for all $q \in Q$ such that $\mathrm{dom}([\![\mathcal{T}_{(q)}]\!]) \neq \emptyset$.

(**) Exercise 1.6. Show that any sequential transducer can be normalized.

Minimization. The **translation** of a sequential function f by a word w in Σ^* is defined by

$$\mathrm{dom}(w^{-1}f) = w^{-1}\mathrm{dom}(f) \qquad w^{-1}f(u) = \left(\bigwedge_{v \in \Sigma^*} f(wv) \right)^{-1} \cdot f(wu)$$

for all u in $\mathrm{dom}(w^{-1}f)$.

Theorem 1.2 (Raney, 1958)**.** *A function $f : \Sigma^* \to \Delta^*$ is sequential iff the set of translations $\{w^{-1}f \mid w \in \Sigma^*\}$ is finite.*

As in the finite automata case where minimal automata are isomorphic with residual automata, the minimal sequential transducer for a sequential function f is defined as the **translation transducer** $\langle Q, \Sigma, \Delta, q_0, \delta, \eta, \iota, \rho \rangle$ where

- $Q = \{w^{-1}f \mid w \in \Sigma^*\}$ (which is finite according to Theorem 1.2),
- $q_0 = \varepsilon^{-1}f$,
- $\iota = \bigwedge_{v \in \Sigma^*} f(v)$ if $\mathrm{dom}(f) \neq \emptyset$ and $\iota = \varepsilon$ otherwise,
- $\delta(w^{-1}, a) = (wa)^{-1}f$,
- $\eta(w^{-1}f, a) = \bigwedge_{v \in \Sigma^*}(w^{-1}f)(av)$ if $\mathrm{dom}((wa)^{-1}f) \neq \emptyset$ and $\eta(w^{-1}f, a) = \varepsilon$ otherwise, and
- $\rho(w^{-1}f) = (w^{-1}f)(\varepsilon)$ if $\varepsilon \in \mathrm{dom}(w^{-1}f)$, and is otherwise undefined.

Recognizable Series The idea of relations in $\Sigma^* \times \Delta^*$ can be extended to map words of Σ^* with values in a semiring $\mathbb{K}$.

A finite **weighted automaton** (or **automaton with multiplicity**, or **$\mathbb{K}$-automaton**) in a semiring $\mathbb{K}$ is a generalization of a finite automaton: $\mathcal{A} = \langle Q, \Sigma, \mathbb{K}, \delta, I, F \rangle$ where $\delta \subseteq Q \times \Sigma \times \mathbb{K} \times Q$ is a weighted transition relation, and I and F are maps from Q to $\mathbb{K}$ instead of subsets of Q. A run

$$\rho = q_0 \xrightarrow{a_1,k_1} q_1 \xrightarrow{a_2,k_2} q_2 \cdots q_{n-1} \xrightarrow{a_n,k_n} q_n$$

defines a **monomial** $[\![\rho]\!] = kw$ where $w = a_1 \cdots a_n$ is the **word label** of ρ and $k = I(q_0)k_1 \cdots k_n F(q_n)$ its **multiplicity**. The behavior $[\![\mathcal{A}]\!]$ of $\mathcal{A}$ is the sum of the monomials for all runs in $\mathcal{A}$: it is a formal power series on Σ^* with coefficients in $\mathbb{K}$, i.e. a map $\Sigma^* \to \mathbb{K}$. The **coefficient** of a word w in $[\![\mathcal{A}]\!]$ is denoted $\langle [\![\mathcal{A}]\!], w \rangle$ and is the sum of the multiplicities of all the runs with w for word label:

$$\langle [\![\mathcal{A}]\!], a_1 \cdots a_n \rangle = \sum_{q_0 \xrightarrow{a_1,k_1} q_1 \cdots q_{n-1} \xrightarrow{a_n,k_n} q_n} I(q_0)k_1 \cdots k_n F(q_n) \ .$$

A matrix **$\mathbb{K}$-representation** for $\mathcal{A}$ is $\langle I, \mu, F \rangle$, where I is seen as a row matrix in $\mathbb{K}^{1\times Q}$, the morphism $\mu : \Sigma^* \to \mathbb{K}^{Q\times Q}$ is defined by $\mu(a)(q, q') = k$ iff $(q, a, k, q') \in \delta$, and F is seen as a column matrix in $\mathbb{K}^{Q\times 1}$. Then

$$\langle [\![\mathcal{A}]\!], w \rangle = I\mu(w)F \ .$$

*There is a notion of $\mathbb{K}$-**rational** series, which coincide with the $\mathbb{K}$-recognizable ones (Schützenberger, 1961).*

A series is **$\mathbb{K}$-recognizable** if there exists a $\mathbb{K}$-representation for it.

The **support** of a series $[\![\mathcal{A}]\!]$ is $\mathrm{supp}([\![\mathcal{A}]\!]) = \{w \in \Sigma^* \mid \langle [\![\mathcal{A}]\!], w \rangle \neq 0_{\mathbb{K}}\}$. This corresponds to the language of the underlying automaton of $\mathcal{A}$.

Exercise 1.7 (Product of series)**.** Let $\mathbb{K}$ be a commutative semiring. Show that $\mathbb{K}$-recognizable series are closed under product: given two $\mathbb{K}$-recognizable series s and s', show that $s \odot s'$ with $\langle s \odot s', w \rangle = \langle s, w \rangle \odot \langle s', w \rangle$ for all w in Σ^* is $\mathbb{K}$-recognizable. What can you tell about the support of $s \odot s'$? **(∗∗)**

Exercise 1.8 (Rational relations as series)**.** Given a relation R in $\Sigma^* \times \Delta^*$, define the series $[\![R]\!]$ by $\langle [\![R]\!], w \rangle = R(w)$ for all w in Σ^*. Show that R is rational iff $[\![R]\!]$ is a $\mathrm{Rat}(\Delta^*)$-recognizable series over Σ. **(∗∗)**

See Berstel (1979, Corollary III.7.2) or Sakarovitch (2009, Theorem IV.1.7).

1.2.2 Morphological Analysis

Let us return to the problem of morphological analysis. We assume that we have at our disposal a **lexicon** of all the possible stems, affixes, and feature structures, and want to model how these morphemes can be combined.

Actually, we use in our examples POS tags as shorthand notations for both morphological features and inflectional affixes. For instance, the morphological analysis of *hospitalized* will rather be $\mathrm{hospital}{-}\mathrm{ize}[{-}\mathrm{vbd}]$, where the POS tag VBG, noted $[{-}\mathrm{vbd}]$, is a shorthand notation for both the inflectional affix *-ed* and the features [cat=v; tense=past; mode=ind]. Accordingly, our lexicon gathers stems, derivational affixes, and POS tags.

As we will see, using finite-state methods solely, we can

1. model the various possible morpheme associations, and their various possible orderings; this is called **morphotactics** (e.g. $[{-}\mathrm{vbd}]$ can follow any verb stem, but the suffix *-ation* should be applied to verbs ending with *-ize* as in *hospitalization*),

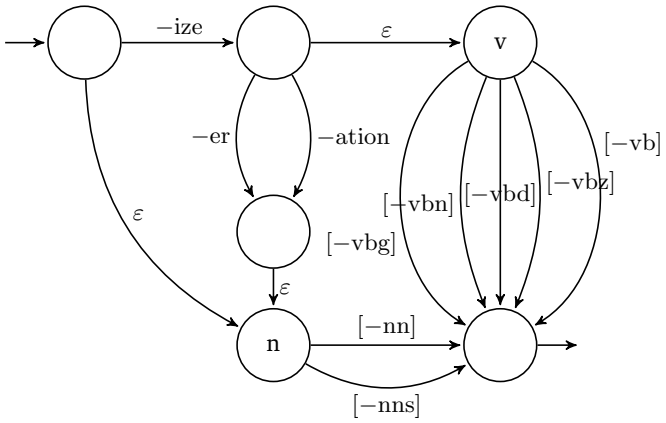


Figure 1.1: A finite state automaton for some morphotactics applicable to nouns ending with *-al*.

2. implement the morphological rules (e.g. $[-\mathrm{vbd}]$ translates into *-ed* for regular verbs, but behaves differently for irregular ones) and orthographic rules (e.g. *hospitalized* and not **hospitalizeed*).

Affix Selection and Morphotactics The issue of finite-state morphotactics is rather straightforward from a formal language viewpoint: the various orderings can be stored in a simple finite state automaton over the lexicon as alphabet. This automaton can then be turned into one over the Latin alphabet plus POS tags and a morpheme boundary marker "$-$"—which we will denote by Σ from now on—, and further minimized.

The automaton of Figure 1.1 presents morphotactics that derive for instance *hospitalizations* ($\mathrm{hospital}-\mathrm{ize}-\mathrm{ation}[-\mathrm{nns}]$) or *hospitalized* ($\mathrm{hospital}-\mathrm{ize}[-\mathrm{vbd}]$).

What is the linguistic value of such a finite automaton representation?

- For one thing, it merely stores information about the possible combinations without providing any rationale. Consider for instance the rules of adjective suffixing with the comparative *-er*: the general rule is that adjectives of two syllables or less can use it, like *sadder* (produced by $\mathrm{sad}[-\mathrm{jjr}]$) or *nicer* ($\mathrm{nice}[-\mathrm{jjr}]$), but not the adjectives with more than two syllables, like **curiouser* or **eleganter* (see Pesetsky, 1985, for a related discussion). Contrast these affixation constraints with those of *un-* on adjectives: we can have *unwell*, *unhappy*, or *uncheerful*, but not **unill*, **unsad* or **unsorrowful*. The explanation is that *un-* can only apply to an adjective with a positive connotation (usually attributed to Zimmer, 1964). Such phonological and semantic explanations are absent from the automaton model.

- Another issue comes from duplication of some parts of the automaton (Karttunen, 1983; Sproat, 1992). For example, contrast *enjoyable* ($\mathrm{en}-\mathrm{joy}-\mathrm{able}$) and *enrichable* ($\mathrm{en}-\mathrm{rich}-\mathrm{able}$) with the incorrect **joyable* and **richable*: in these examples, the *-able* suffix is only acceptable if the *en-* prefix is present, resulting in duplication in the automaton in order to record whether *en-* was added or not. This seems to indicate that the finite state model is not perfectly adequate; for instance a more compact representation would be obtained through pushdown automata.

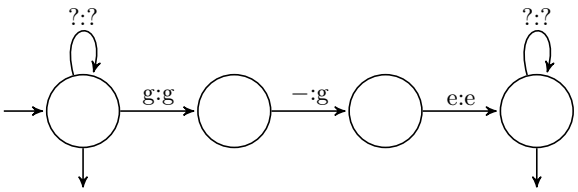


Figure 1.2: A transducer for rewrite rule (1.2). Question marks ?:? stand for a:a for all $a \in \Sigma$.

Morphological and Orthographic Rules A natural way to represent morphological and orthographic rules is to use **string rewrite systems**. The formation of *begged* out of the stem *beg* and the affix *-ed* can be explained as an application of the morphological rule

$$[-\mathrm{vbn}] \to -\mathrm{ed} \tag{1.1}$$

followed by the orthographic rule

$$\mathrm{g}-\mathrm{e} \to \mathrm{gge} \tag{1.2}$$

to $\mathrm{beg}[-\mathrm{vbn}]$. The one-step derivation relation $\overset{R}{\Rightarrow}$ defined by a finite string rewrite system R is a rational relation, which can also be expressed as $\mathrm{Id}_{\Sigma^*} \cdot (\sum_{u \to v \in R} \mathrm{u{:}v}) \cdot \mathrm{Id}_{\Sigma^*}$. For instance this corresponds to

$$\mathrm{Id}_{\Sigma^*} \cdot \mathrm{g}-\mathrm{e{:}gge} \cdot \mathrm{Id}_{\Sigma^*} \tag{1.3}$$

for the one-rule system consisting of rule (1.2), which can be implemented by a finite transducer, as the one of Figure 1.2 for (1.2).

The remainder of this section is dedicated to the translation from string rewriting formalisms into finite state transducers: Section 1.2.3 presents a formalism of cascaded **phonological rules** to this end.

See Koskenniemi and Church (1988) for another formalism of parallel rewrites.

1.2.3 Phonological Rules

Chomsky and Halle (1968) introduced a particular notation for the rewrite rules used in phonology: the general form of a **phonological rule** is

$$\alpha \to \beta \;/\; \lambda\_\_\_\_\rho \tag{1.4}$$

where α, β, λ, and ρ are rational languages over Σ (for instance represented by rational expressions). Such a rule stands roughly for "rewrite α into β in the context of λ, ρ", i.e. for the (generally infinite) string rewrite system

$$\{x\, u\, y \to x\, v\, y \mid (x, u, v, y) \in \lambda \times \alpha \times \beta \times \rho\} \tag{1.5}$$

—but not quite, as we will see later when considering the implicit restrictions on derivations assumed by phonologists. The same formalism can also be employed for the treatment of morphological and orthographic rules.

Example 1.3. Let us focus for instance on past participle inflection: (1.2) can be restated with this notation as

$$- \to \mathrm{g} \;/\; \mathrm{g}\_\_\_\_\mathrm{e}\ . \tag{1.6}$$

Keeping with past participle composition, we would also need

$$- \to \varepsilon \;/\; \_\_\_\_ \tag{1.7}$$

in order to obtain *faxed* from $\mathrm{fax}{-}\mathrm{ed}$ (contexts are left blank for the rational expression ε), and

$$\mathrm{e}- \to \varepsilon \;/\; \_\_\_\_\mathrm{e} \tag{1.8}$$

in order to obtain *danced* from $\mathrm{dance}{-}\mathrm{ed}$. On the other hand, the formation of *sung* for *to sing* requires the addition of

$$\mathrm{sing}[-\mathrm{vbn}] \to \mathrm{sung} \;/\; \_\_\_\_\;. \tag{1.9}$$

Observe that we should be able to **order** our rules if we want to avoid spurious rewrites, like **beged* or **singged*. In our case, we should apply (1.9), then (1.1), then (1.6) and (1.8), and (1.7) last. Furthermore, we should make the rewrites **obligatory**: if (1.9) or (1.6) can be applied, then they should be. But not all rules should be obligatory: for instance, with

$$\mathrm{prove}[-\mathrm{vbn}] \to \mathrm{proven} \;/\; \_\_\_\_ \tag{1.10}$$

both *proven* and *proved* are accepted as past participles of *to prove*, thus (1.10) should be **optional**. Adding some derivational morphology to the mix allows to witness another issue with rule application:

$$[-\mathrm{vbg}] \to -\mathrm{ing} \;/\; \_\_\_\_ \tag{1.11}$$

$$\mathrm{e}- \to \varepsilon \;/\; \_\_\_\_\mathrm{i} \tag{1.12}$$

could model gerund inflection in *dancing*. In order to obtain *passivizing*, we need to apply (1.11) once to $\mathrm{passive}{-}\mathrm{ize}[-\mathrm{vbg}]$, and (1.12) on all the applicable factors of $\mathrm{passive}{-}\mathrm{ize}{-}\mathrm{ing}$: we are actually applying the *transitive closure* of the one-step derivation relation defined by rule (1.12).

To sum up, a **phonological rule system** consists of a finite sequence $\mathcal{P} = r_1 \cdots r_n$ of phonological rules r_i, each rule r_i defining a rewrite relation $[\![r_i]\!]$ over $\Sigma^* \times \Sigma^*$, such that the behavior of $\mathcal{P}$ is the composition $[\![\mathcal{P}]\!] = [\![r_1]\!] \fatsemi \cdots \fatsemi [\![r_n]\!]$.

Restrictions on Rewrites In the optional case, the rewrite relation defined by a phonological rule of form (1.4) seems to be exactly the derivation relation of the system (1.5). General string rewrite systems are already Turing-complete in the finite case (in fact, three rules are enough to yield undecidable accessibility (Matiyasevicha and Sénizergues, 2005)), thus there is no hope to be able to compute the effect of a phonological rule without further restricting derivations.

A related open problem is to decide termination of one-rule string rewrite systems, see McNaughton (1995); Sénizergues (1996) for partial solutions.

Fortunately, linguists give a particular semantics to rewrites: after the application of a rule like (1.4), the newly written word from β cannot be later rewritten. Moreover, rewrites are constrained to occur left-to-right (or right-to-left or simultaneously, but we will only consider the first case).

Formally, given a phonological rule $r = \alpha \to \beta \;/\; \lambda\_\_\_\_\rho$, a **derivation** $w_0 \overset{r}{\Rightarrow} w_1 \overset{r}{\Rightarrow} \cdots \overset{r}{\Rightarrow} w_n$ is such that for each $0 \leq i < n$, $w_i = x_i u_i y_i$ and $w_{i+1} = x_i v_i y_i$ for some $(x_i, u_i, v_i, y_i) \in (\Sigma^* \cdot \lambda) \times \alpha \times \beta \times (\rho \cdot \Sigma^*)$. A derivation is **left-to-right** if for each $0 \leq i < n-1$, $|x_i v_i| \leq |x_{i+1}|$; we only consider left-to-right derivations in the following. It is furthermore

leftmost if for each $0 \leq i < n$ and for all (z, z') in $\Sigma^* \times \Sigma^+$ such that $zz' = x_i u_i$ and either $i = 0$, or $i > 0$ and $|z| \geq |x_{i-1} v_{i-1}|$, $(z, z' y_i) \notin (\Sigma^* \lambda \alpha) \times (\rho \Sigma^*)$,

irreducible if either $n = 0$ and $w_0 \notin \Sigma^* \lambda \alpha \rho \Sigma^*$, or $n > 0$ and for all z, z' in Σ^* with $y_{n-1} = zz'$, $(x_{n-1} v_{n-1} z, z') \notin (\Sigma^* \lambda \alpha) \times (\rho \Sigma^*)$.

Given a phonological rule r, its **behavior** $[\![r]\!]$ is a relation over Σ^* defined by $w \, [\![r]\!] \, w'$ iff

- there exists a left-to-right derivation $w = w_0 \overset{r}{\Rightarrow} w_1 \overset{r}{\Rightarrow} \cdots \overset{r}{\Rightarrow} w_n = w'$ if r is optional, or iff
- there exists a left-to-right, leftmost, and irreducible derivation $w = w_0 \overset{r}{\Rightarrow} w_1 \overset{r}{\Rightarrow} \cdots \overset{r}{\Rightarrow} w_n = w'$ if r is obligatory.

Note that the definition of left-to-right derivations justifies the context notation in phonological rules: using directly rules of form $\lambda \alpha \rho \to \lambda \beta \rho$ in left-to-right mode would not allow to later rewrite the factor matched by ρ.

Phonological Rules as Rational Relations We show in this section that the behavior of a phonological rule r is a rational relation. We only consider the simpler case of optional rules; see Kaplan and Kay (1994) and Mohri and Sproat (1996) for the obligatory case.

First observe that, in a left-to-right derivation $w_0 \overset{r}{\Rightarrow} w_1 \overset{r}{\Rightarrow} \cdots \overset{r}{\Rightarrow} w_n$, since each of the n rewrites has to occur to the right of the previous one, we can decompose each w_i as

$$\begin{aligned} w_0 &= z_0 u_0 z_1 u_1 z_2 \cdots z_{n-2} u_{n-1} z_{n-1} \\ w_1 &= z_0 v_0 z_1 u_1 z_2 \cdots z_{n-2} u_{n-1} z_{n-1} \\ &\vdots \\ w_n &= z_0 v_0 z_1 v_1 z_2 \cdots z_{n-2} v_{n-1} z_{n-1} \end{aligned}$$

verifying

$$(z_0 v_0 z_1 \cdots z_{i-1}, u_i, v_i, z_i \cdots z_{n-2} u_{n-1} z_{n-1}) \in (\Sigma^* \cdot \lambda) \times \alpha \times \beta \times (\rho \cdot \Sigma^*) \quad (1.13)$$

for each $0 \leq i < n$.

The second observation is that right contexts can be checked against w_0, whereas left contexts should be checked against w_n. Hence a decomposition of $[\![r]\!]$ as the composition of three relations

$$[\![r]\!] = \mathrm{right}_\rho \fatsemi \mathrm{replace}_{\alpha,\beta} \fatsemi \mathrm{left}_\lambda \quad (1.14)$$

that respectively check the right contexts, perform the rewrites, and check the left contexts.

It remains to implements these relations as rational transductions. Let us introduce a fresh delimiter symbol $\#$ and the projection $\pi_\# : (\Sigma \uplus \{\#\})^* \to \Sigma^*$. The relation right_ρ nondeterministically introduces $\#$s before factors in ρ. The relation $\mathrm{replace}_{\alpha,\beta}$ replaces a factor $u\#$ in $\alpha\#$ by a factor $\#v$ in $\#\beta$. The relation left_λ erases $\#$s after factors in $\pi_\#^{-1}(\lambda)$.

Exercise 1.9. Propose transducer constructions for each of right_ρ, $\mathrm{replace}_{\alpha,\beta}$, and left_λ. **(***)**

(***) **Exercise 1.10.** Given a rational relation R in $\Sigma^* \times \Delta^*$, build a phonological rule system $\mathcal{P}$ of optional rules such that, for all (w, w') in $\Sigma^* \times \Delta^*$, $\$w\$ \llbracket\mathcal{P}\rrbracket \$w'\$$ iff $w \; R \; w'$, where $\$$ is an end-of-string marker neither in Σ nor in Δ.

Deduce that the **morphological analysis problem** for phonological rule systems, i.e. given a phonological rule system $\mathcal{P}$ of optional rules and a word w', to decide whether there exists w such that $w \llbracket\mathcal{P}\rrbracket w'$, is PSPACE-hard.

1.3 Part-of-Speech Tagging

Recall that the POS tagging task consists in assigning the appropriate part-of-speech tag to a word in the context of its sentence. The program that performs this task, the **POS tagger**, can be learned from an annotated corpus like the Penn Treebank—called **supervised** learning.

Formally, we are given a finite tagset Θ and an annotated corpus. For benchmarking purposes, the corpus is typically partitioned into

- a **training corpus**, on which the tagger is trained,
- optionally a **development corpus**, used to tune the tagger training algorithm, and
- a **test corpus**, on which the performance of the tagger is measured.

Thus the training corpus is made of sequences of (word, POS tag) pairs in $\Sigma \times \Theta$, where Σ is the set of words in the training corpus. A consequence of this subdivision is that Σ is likely to be a strict subset of the set of words in the entire corpus; in particular, there are bound to be **unknown words** in the test corpus. For instance, Brants (2000) reports that 2.9% of the words in his 10%-sized test set from the Penn Treebank corpus were unknown; unsurprisingly, tagging accuracies tend to be significantly lower for unknown words.

The accuracy of taggers trained on a corpus similar enough to the test set, for instance using a partitioned corpus, is quite high: Brill (1992) reports tagging accuracy scores around 95% using his rule-based tagger on the Brown corpus, while Brants (2000) reports an overall 96.7% accuracy on the WSJ parts of the Penn Treebank with his trigram-HMM tagger (these values are not directly comparable due to differing tagsets and corpora). One has to contrast such numbers with the mean inter-annotator agreement rate: Marcus et al. (1993) report that on average two linguists agree over 96.6% of the tags. Hence the accuracy scores of taggers trained on a corpus similar to the test set is pretty much optimal!

1.3.1 Rule-Based Tagging

This section is partly based on Roche and Schabes (1995).

The most famous rule-based POS tagging technique is due to Brill (1992). He introduced a three-parts technique comprising:

1. a lexical tagger, which associates a unique POS tag to each word from an annotated training corpus. This lexical tagger simply associates to each known word its most probable tag according to the training corpus annotation, i.e. a unigram maximum likelihood estimation;
2. an unknown word tagger, which attempts to tag unknown words based on suffix or capitalization features. It works like the contextual tagger, using

the presence of a capital letter and bounded sized suffixes in its rules: for instance, a *-able* suffix usually denotes an adjective;

3. a contextual tagger, on which we focus here. It consists of a cascade of **contextual rules** of form $uav \to ubv$ for a, b in Θ and uv in $\Theta^{\leq k}$ for some predefined k, which correct tag assignments based on the u, v contexts. We present in this section how such rules are learned from the training or the development corpus, and how they can be compiled into sequential transducers.

As in Section 1.2.3, the rewrite semantics of these rules are not quite the usual ones.

Learning Contextual Rules

We start with an example by Roche and Schabes (1995): Let us suppose the following sentences were tagged by the lexical tagger

*Chapman/NNP killed/VBN John/NNP Lennon/NNP
*John/NNP Lennon/NNP was/VBD shot/VBD by/IN Chapman/NNP
He/PRP witnessed/VBD Lennon/NNP killed/VBN by/IN Chapman/NNP

with mistakes in the first two sentences: *killed* should be tagged as a past tense form, and *shot* as a past participle form.

The contextual tagger learns contextual rules of form $uav \to ubv$ for a, b in Θ and uv in $\Theta^{\leq k}$ for some predefined k; in practice, $k = 2$ or $k = 3$. The learning algorithm simply consists in comparing the effect of all possible contextual rules on the tagging accuracy, and keeping the one with the best results. The learning phase always terminate since a rule is kept only if it actually improves tagging accuracy, and there is only a finite number of possible pairs in $\Sigma \times \Theta$ for each token of the training corpus. In fact, Brill (1992) reports that 71 rules are enough when learning on 5% of the Brown corpus; Roche and Schabes (1995) obtain 280 rules on 90% of the Brown corpus.

Brill (1992) and Roche and Schabes (1995) use slightly different templates *than the one parametrized by k we present here.*

For instance, a first contextual rule could be

$$\mathrm{nnp\,vbn} \to \mathrm{nnp\,vbd} \tag{1.15}$$

resulting in a new tagging

Chapman/NNP killed/VBD John/NNP Lennon/NNP
*John/NNP Lennon/NNP was/VBD shot/VBD by/IN Chapman/NNP
*He/PRP witnessed/VBD Lennon/NNP killed/VBD by/IN Chapman/NNP

A second contextual rule could be

$$\mathrm{vbd\,in} \to \mathrm{vbn\,in} \tag{1.16}$$

resulting in the correct tagging

Chapman/NNP killed/VBD John/NNP Lennon/NNP
John/NNP Lennon/NNP was/VBD shot/VBN by/IN Chapman/NNP
He/PRP witnessed/VBD Lennon/NNP killed/VBN by/IN Chapman/NNP

Contextual Rules as Sequential Functions

As stated before, our goal is to compile the entire sequence of contextual rules learned from a corpus into a single sequential function.

Let us first formalize the semantics of Brill's contextual rules. Let $\mathcal{C} = r_1 r_2 \dots r_n$ be a finite sequence of rewrite rules in $\Theta^* \times \Theta^*$. In practice the rules constructed in Brill's contextual tagger are length-preserving and modify a single letter, but this is not a useful consideration from a theoretical viewpoint. Each rule $r_i = u_i \to v_i$ defines a **leftmost rewrite relation** $\overset{r_i}{\underset{\text{lm}}{\Longrightarrow}}$ defined by

$$w \overset{r_i}{\underset{\text{lm}}{\Longrightarrow}} w' \text{ iff } \exists x, y \in \Sigma^*, w = x u_i y \wedge w' = x v_i y \wedge (\forall z, z' \in \Sigma^*, w \neq z u_i z' \vee x \leq_{\text{pref}} z) \tag{1.17}$$

Note that the domain of $\overset{r_i}{\underset{\text{lm}}{\Longrightarrow}}$ is $\Theta^* \cdot u_i \cdot \Theta^*$. The **behavior** of a single rule is then

$$[\![r_i]\!] = \overset{r_i}{\underset{\text{lm}}{\Longrightarrow}} \cup \mathrm{Id}_{\overline{\Theta^* \cdot u_i \cdot \Theta^*}} , \tag{1.18}$$

i.e. it applies $\overset{r_i}{\underset{\text{lm}}{\Longrightarrow}}$ on $\Theta^* \cdot u_i \cdot \Theta^*$ and the identity on its complement $\overline{\Theta^* \cdot u_i \cdot \Theta^*}$. The behavior of $\mathcal{C}$ is then the composition

$$[\![\mathcal{C}]\!] = [\![r_1]\!] \fatsemi [\![r_2]\!] \fatsemi \cdots \fatsemi [\![r_n]\!] . \tag{1.19}$$

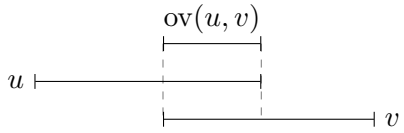
A naive implementation of $\mathcal{C}$ would try to match each u_i at every position of the input string w in Σ^*, resulting in an overall complexity of $O(|w| \cdot \sum_i |u_i|)$. However, one often faces the problem of tagging a *set* of sentences $\{w_1, \dots, w_m\}$, which yields $O((\sum_i |u_i|) \cdot (\sum_j |w_j|))$. As shown in Roche and Schabes' experiments, compiling $\mathcal{C}$ into a single sequential transducer $\mathcal{T}$ results in practice in huge savings, with overall complexities in $O(|w| + |\mathcal{T}|)$ and $O(|\mathcal{T}| + \sum_j |w_j|)$ respectively.

By (1.19) and the closure of sequential functions under composition, it suffices to prove that $[\![r_i]\!]$ is a sequential function for each i in order to prove that $[\![\mathcal{C}]\!]$ is a sequential function. Since each $[\![r_i]\!]$ is a rational function, being the union of two rational functions over disjoint domains, our efforts are not doomed from the start.

See (Simon, 1994; Crochemore and Hancart, 1997).

Sequential Transducer of a Rule Intuitively, the sequential transducer for $[\![r_i]\!]$ is related to the **string matching automaton** for u_i, i.e. the automaton for the language $\Theta^* u_i$. This insight yields a *direct* construction of the minimal sequential transducer of a contextual rule, with $|u_i| + 1$ states in most cases. Let us recall a few definitions:

Definition 1.4 (Overlap, Border)**.** The **overlap** $\mathrm{ov}(u, v)$ of two words u and v is the longest suffix of u which is simultaneously a prefix of v:



A word u is a **border** of a word v if it is both a prefix and a suffix of v, i.e. if there exist v_1, v_2 in Θ^* such that $v = u v_1 = v_2 u$. For $v \neq \varepsilon$, the longest border of v different from v itself is denoted $\mathrm{bord}(v)$.

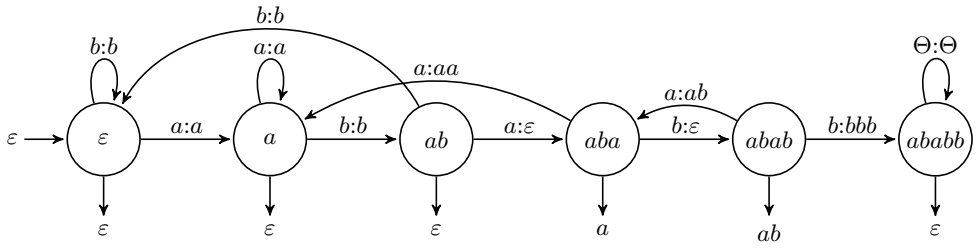
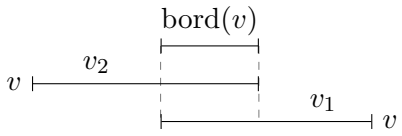


Figure 1.3: The sequential transducer constructed for $ababb \to abbbb$.



Exercise 1.11. Show that for all u, v in Θ^* and a in Θ, **(∗)**

$$\mathrm{ov}(ua, v) = \begin{cases} \mathrm{ov}(u,v)\cdot a & \text{if } \mathrm{ov}(u,v)\cdot a \leq_{\mathrm{pref}} v \\ \mathrm{bord}(\mathrm{ov}(u,v)\cdot a) & \text{otherwise.} \end{cases} \tag{1.20}$$

Definition 1.5 (Transducer of a Contextual Rule)**.** The sequential transducer $\mathcal{T}_r$ associated with a contextual rule $r = u \to v$ with $u \neq \varepsilon$ is defined as

$$\mathcal{T}_r = \langle \mathrm{Pref}(u), \Theta, \Theta, \varepsilon, \delta, \eta, \varepsilon, \rho \rangle$$

with the set of prefixes of u as state set, ε as initial state and initial output, and for all a in Θ and w in $\mathrm{Pref}(u)$,

$$\delta(w,a) = \begin{cases} wa & \text{if } wa \leq_{\mathrm{pref}} u \\ w & \text{if } w = u \\ \mathrm{bord}(wa) & \text{otherwise} \end{cases}$$

$$\rho(w) = \begin{cases} \varepsilon & \text{if } w \leq_{\mathrm{pref}} (u \wedge v) \\ (u \wedge v)^{-1} \cdot w & \text{if } (u \wedge v) <_{\mathrm{pref}} w <_{\mathrm{pref}} u \\ \varepsilon & \text{otherwise, i.e. if } w = u \end{cases}$$

$$\eta(w,a) = \begin{cases} a & \text{if } wa \leq_{\mathrm{pref}} (u \wedge v) \\ \varepsilon & \text{if } (u \wedge v) <_{\mathrm{pref}} wa <_{\mathrm{pref}} u \\ (u \wedge v)^{-1} \cdot v & \text{if } wa = u \\ a & \text{if } w = u \\ \rho(w)a \cdot \rho(\mathrm{bord}(wa))^{-1} & \text{otherwise.} \end{cases}$$

For instance, the sequential transducer for the rule $ababb \to abbbb$ is shown in Figure 1.3 (one can check that $ababb \wedge abbbb = ab$, $\mathrm{bord}(b) = \varepsilon$, $\mathrm{bord}(aa) = a$, $\mathrm{bord}(abb) = \varepsilon$, $\mathrm{bord}(abaa) = a$, and $\mathrm{bord}(ababa) = aba$).

Proposition 1.6. *Let $r = u \to v$ with $u \neq \varepsilon$. Then $[\![\mathcal{T}_r]\!] = [\![r]\!]$.*

Proof. Let us first consider the case of input words in $\overline{\Theta^* \cdot u \cdot \Theta^*}$:
Claim 1.6.1. For all w in $\overline{\Theta^* \cdot u \cdot \Theta^*}$,

$$\delta(\varepsilon, w) = \mathrm{ov}(w,u) \qquad\qquad \eta(\varepsilon, w) = w \cdot \rho(\mathrm{ov}(w,u))^{-1}\ .$$

Proof of Claim 1.6.1. By induction on w: since $u \neq \varepsilon$, the base case is $w = \varepsilon$ with

$$\delta(\varepsilon, \varepsilon) = \varepsilon = \mathrm{ov}(\varepsilon, u) \qquad \eta(\varepsilon, \varepsilon) = \varepsilon = \varepsilon \cdot \varepsilon^{-1} = \varepsilon \cdot \rho(\varepsilon)^{-1} \ .$$

For the induction step, we consider wa in $\overline{\Theta^* \cdot u \cdot \Theta^*}$ for some w in Θ^* and a in Θ, and we get

$$\begin{aligned}
\delta(\varepsilon, wa) &= \delta(\delta(\varepsilon, w), a) && \text{(by def.)} \\
&= \delta(\mathrm{ov}(w, u), a) && \text{(by ind. hyp.)} \\
&= \mathrm{ov}(wa, u) && \text{(by (1.20))} \\
\eta(\varepsilon, wa) &= \eta(\varepsilon, w) \cdot \eta(\delta(\varepsilon, w), a) && \text{(by def.)} \\
&= w \cdot \rho(\delta(\varepsilon, w))^{-1} \cdot \eta(\delta(\varepsilon, w), a) && \text{(by ind. hyp.)} \\
&= w \cdot \rho(w')^{-1} \cdot \eta(w', a) \ ; && \text{(by setting } w' = \delta(\varepsilon, w))
\end{aligned}$$

we need to do a case analysis for this last equation:

Case $\boldsymbol{w'a \not\leq_{\mathbf{pref}} u}$ Then $\eta(w', a) = \rho(w') \cdot a \cdot \rho(\mathrm{border}(w'a))^{-1}$, which yields

$$\begin{aligned}
\eta(\varepsilon, wa) &= w \cdot \rho(w')^{-1} \cdot \rho(w') \cdot a \cdot \rho(\delta(\varepsilon, wa))^{-1} \\
&= wa \cdot \rho(\delta(\varepsilon, wa))^{-1} \ .
\end{aligned}$$

Case $\boldsymbol{w'a <_{\mathbf{pref}} u}$ Then $\delta(\varepsilon, wa) = w'a$, and we need to further distinguish between several cases:

- **$\boldsymbol{w'a \leq_{\mathbf{pref}} (u \wedge v)}$** then $\rho(w') = \varepsilon$, $\eta(w', a) = a$, and $\rho(w'a) = \varepsilon$, thus
$$\eta(\varepsilon, wa) = wa = wa \cdot \varepsilon^{-1} = wa \cdot \rho(w'a)^{-1} \ ,$$
- **$\boldsymbol{w' = (u \wedge v)}$** then $\rho(w') = \varepsilon$, $\eta(w', a) = \varepsilon$, and $\rho(w'a) = (u \wedge v)^{-1} \cdot w'a = a$, thus
$$\eta(\varepsilon, wa) = w = wa \cdot a^{-1} = wa \cdot \rho(w'a)^{-1} \ ,$$
- **$\boldsymbol{(u \wedge v) <_{\mathbf{pref}} w'}$** then $\rho(w') = (u \wedge v)^{-1} \cdot w'$, $\eta(w', a) = \varepsilon$, and $\rho(w'a) = (u \wedge v)^{-1} \cdot w'a$, thus
$$\begin{aligned}
\eta(\varepsilon, wa) &= w \cdot ((u \wedge v)^{-1} \cdot w')^{-1} = wa \cdot a^{-1} \cdot ((u \wedge v)^{-1} \cdot w')^{-1} \\
&= wa \cdot \rho(w'a)^{-1} \ . \qquad \text{[1.6.1]}
\end{aligned}$$

Claim 1.6.1 yields that $[\![\mathcal{T}_r]\!]$ coincides with $[\![r]\!]$ on words in with $\overline{\Theta^* \cdot u \cdot \Theta^*}$, i.e. is the identity over $\overline{\Theta^* \cdot u \cdot \Theta^*}$. Then, since $u \neq \varepsilon$, a word in $\Theta^* \cdot u \cdot \Theta^*$ can be written as waw' with w in $\overline{\Theta^* \cdot u \cdot \Theta^*}$, a in Θ with wa in $\Theta^* \cdot u$, and w' in Θ^*. Let $u = u'a$; Claim 1.6.1 implies that

$$\delta(\varepsilon, w) = u' \qquad \eta(\varepsilon, w) = w \cdot \rho(u')^{-1} \ .$$

Thus, by definition of $\mathcal{T}_r$, $\delta(\varepsilon, wa) = u'a = u$ and

$$\eta(\varepsilon, wa) = \eta(\varepsilon, w) \cdot \eta(u', a) = w \cdot \rho(u')^{-1} \cdot (u \wedge v)^{-1} \cdot v \ ;$$

if $\boldsymbol{(u \wedge v) <_{\mathbf{pref}} u'}$

$$\eta(\varepsilon, wa) = w \cdot ((u \wedge v)^{-1} \cdot u')^{-1} \cdot (u \wedge v)^{-1} \cdot v = w \cdot u'^{-1} \cdot v = wa \cdot u^{-1} \cdot v \ ;$$

otherwise i.e. if $u' = (u \wedge v)$:

$$\eta(\varepsilon, wa) = w \cdot u'^{-1} \cdot v = wa \cdot u^{-1} \cdot v \ .$$

Thus in all cases $[\![\mathcal{T}_r]\!](wa) = [\![r]\!](wa)$, and since $\mathcal{T}_r$ starting in state u (i.e. $\mathcal{T}_{r(u)}$) implements the identity over Θ^*, we have more generally $[\![\mathcal{T}_r]\!] = [\![r]\!]$. □

Lemma 1.7. *Let $r = u \to v$. Then $\mathcal{T}_r$ is normalized.*

Proof. Let $w \in \mathrm{Prefix}(u)$ be a state of $\mathcal{T}_r$; we need to show that $\bigwedge [\![\mathcal{T}_{r(w)}]\!](\Theta^*) = \varepsilon$.

If $(u \wedge v) <_{\mathbf{pref}} w <_{\mathbf{pref}} u$ let $u' = w^{-1}u \in \Theta^+$, and consider the two outputs

$$[\![\mathcal{T}_{r(w)}]\!](u') = \eta(w, u')\rho(u) = (u \wedge v)^{-1}v \quad [\![\mathcal{T}_{r(w)}]\!](\varepsilon) = \rho(w) = (u \wedge v)^{-1}w \ .$$

Since $(u \wedge v) <_{\mathrm{pref}} u$ we can write u as $(u \wedge v)au''u'$, and either $v = (u \wedge v)bv'$ or $v = u \wedge v$, for some $a \neq b$ in Θ and u'', v' in Θ^*; this yields $w = (u \wedge v)au''$ and thus $[\![\mathcal{T}_{r(w)}]\!](u') \wedge [\![\mathcal{T}_{r(w)}]\!](\varepsilon) = \varepsilon$.

otherwise $\rho(w) = \varepsilon$, which yields the lemma. □

Proposition 1.8. *Let $r = u \to v$ with $u \neq \varepsilon$ and $u \neq v$. Then $\mathcal{T}_r$ is the minimal sequential transducer for $[\![r]\!]$.*

Proof. Let $w <_{\mathrm{pref}} w'$ be two different states in $\mathrm{Prefix}(u)$; we proceed to prove that $[\![w^{-1}\mathcal{T}_r]\!] \neq [\![w'^{-1}\mathcal{T}_r]\!]$, hence that no two states of $\mathcal{T}_r$ can be merged. By Lemma 1.7 it suffices to prove that $[\![\mathcal{T}_{r(w)}]\!] \neq [\![\mathcal{T}_{r(w')}]\!]$, thus to exhibit some $x \in \Theta^*$ such that $[\![\mathcal{T}_{r(w)}]\!](x) \neq [\![\mathcal{T}_{r(w')}]\!](x)$. We perform a case analysis:

if $w' \leq_{\mathbf{pref}} (u \wedge v)$ then $w <_{\mathrm{pref}} (u \wedge v)$ thus $[\![\mathcal{T}_{r(w)}]\!](x) = x$ for all $x \notin w^{-1} \cdot \Theta^* \cdot u \cdot \Theta^*$; consider

$$[\![\mathcal{T}_{r(w)}]\!](w'^{-1}u) = w'^{-1}u \neq w'^{-1}v = [\![\mathcal{T}_{r(w')}]\!](w'^{-1}u) \ ;$$

if $w \leq_{\mathbf{pref}} (u \wedge v)$ and $w' = u$ then $[\![\mathcal{T}_{r(w')}]\!](x) = x$ for all x and we consider

$$[\![\mathcal{T}_{r(w)}]\!](w^{-1}u) = w^{-1}v \neq w^{-1}v = [\![\mathcal{T}_{r(w')}]\!](w^{-1}u) \ ;$$

otherwise that is if $w \leq_{\mathrm{pref}} (u \wedge v)$ and $(u \wedge v) <_{\mathrm{pref}} w' <_{\mathrm{pref}} u$, or $(u \wedge v) <_{\mathrm{pref}} w <_{\mathrm{pref}} w' \leq_{\mathrm{pref}} u$, we have $\rho(w) \neq \rho(w')$ thus

$$[\![\mathcal{T}_{r(w)}]\!](\varepsilon) \neq [\![\mathcal{T}_{r(w')}]\!](\varepsilon) \ .$$ □

Exercise 1.12. Define the minimal sequential transducers for $r = u \to v$ in the cases $u = \varepsilon$ and $u = v$. **(∗)**

1.3.2 HMM Tagging

Other approaches to the POS tagging problem rely on probabilistic models to find an appropriate tag sequence given a word sequence. A simple formalism to this end is that of **hidden Markov models** (**HMM**), where the observed sequences of symbols (here the words) depend on hidden sequences of states (here the tags) that spanned them.

We need to define a notion of probabilities for sequences. Consider n variables $Y_1, \ldots, Y_n$ with values in Σ, and a sequence w of n words in Σ. Variable Y_i is the

act of observing the ith word in the sequence of n words. The probability of a particular sequence $w = a_1 \cdots a_n$ is then

$$\begin{aligned} & p(a_1 \cdots a_n) \\ & \quad = \Pr(Y_1 = a_1, Y_2 = a_2, \ldots, Y_n = a_n) \\ & \quad = \Pr(Y_1 = a_1) \cdot \Pr(Y_2 = a_2 | Y_1 = a_1) \cdots \Pr(Y_n = a_n | Y_1 = a_1, \ldots, Y_{n-1} = a_{n-1}) \\ & \quad = \prod_{i=1}^{n} \Pr(Y_i = a_i | Y_1 = a_1, \ldots, Y_{i-1} = a_{i-1}) \ . \end{aligned}$$

Add an extra variable Y_0 and a "beginning-of-sequence" marker \$ with $\Pr(Y_0 = \$) = 1$; we obtain a simpler expression

$$p(a_1 \cdots a_n) = \prod_{i=1}^{n} p(a_i | \$ a_1 \cdots a_{i-1}) \ . \tag{1.21}$$

Hidden Markov model provide a means to define the probability of an observed sequence as the result of another, hidden, sequence of states.

Given a set S, $\mathrm{Disc}(S)$ denotes the set of discrete probability distributions over S, i.e. $\{p : S \to [0,1] \mid \sum_{e \in S} p(e) = 1\}$.

Definition 1.9 (HMM)**.** A **hidden Markov model** is a tuple $\mathcal{H} = \langle Q, \Sigma, S, T, E \rangle$ where Q is a finite set of states, Σ a finite output alphabet, $S \in \mathrm{Disc}(Q)$ the starting state probabilities, $T : Q \to \mathrm{Disc}(Q)$ the transition probabilities, and $E : Q \to \mathrm{Disc}(\Sigma)$ the emission probabilities.

The entries of S represent the conditional probability $S(q) = p(q|\$)$ of starting a sequence of states in state q, T the conditional probability $T(q)(q') = p(q'|q)$ of moving to q' when in q, and E the conditional probability $E(q)(a) = p(a|q)$ of emitting a when in q. The probability for a run $\rho = q_1 \cdots q_n$ to occur is defined to be

$$p(\rho) = \prod_{i=1}^{n} p(q_i | \$ q_1 \cdots q_{i-1}) = \prod_{i=1}^{n} p(q_i | q_{i-1}) = S(q_1) \cdot \prod_{i=2}^{n} T(q_{i-1})(q_i)$$

(with $q_0 = \$$), i.e. the conditional probability distribution of the next state depends only upon the current state—the **Markov property**—, while the probability for this run to emit $w = a_1 \cdots a_n$ is defined to depend solely on the currently visited states,

$$p(w|\rho) = \prod_{i=1}^{n} p(a_i | q_i) = \prod_{i=1}^{n} E(q_i)(a_i) \ ;$$

and the probability of w is thus

$$p(w) = \sum_{\rho \in Q^n} p(w|\rho) \cdot p(\rho) \ .$$

Observe that a HMM defines a discrete probability distribution over Σ^n for all n:

$$\forall n, \ \sum_{w \in \Sigma^n} p(w) = 1 \ . \tag{1.22}$$

Example 1.10. Consider the HMM defined by $Q = \{q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, and

$$S = \begin{pmatrix} 0.5 & 0.5 & 0 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix} \qquad E = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0.5 & 0.5 \end{pmatrix}.$$

It starts with probability 0.5 in either q_1 or q_2. Supposing it starts in q_2, it remains there with probability 0.5 and emits a, or moves to q_3 and emits a or b. The run q_2q_2 has probability $p(q_2q_2) = 0.25$ and emits aa with probability $p(aa|q_2q_2) = 1$. There are other runs that emit aa, for instance q_3q_3 is such that $p(aa|q_3q_3) = 0.25$, but $p(q_3q_3) = 0$, and in fact there are only two other runs with non-null probability that emit aa: $p(q_1q_1) = 0.5$ with $p(aa|q_1q_1) = 1$ and $p(q_2q_3) = 0.25$ with $p(aa|q_2q_3) = 0.5$, thus we have $p(aa) = 0.875$.

Constructing HMMs from N-Grams

As we have seen in (1.21), the probability of a given sequence is a complex expression that involves the full history of the sequence at each step. The idea of **N-grams** is to approximate this full history by considering only the last $N - 1$ events as conditioning the current one, i.e. by replacing (1.21) with

$$p(a_1 \cdots a_n) \approx \prod_{i-1}^{n} p(a_i|a_{i-N+1} \cdots a_{i-1}) \,, \tag{1.23}$$

(with the convention that $a_j = \$$ is a dummy observation for each $j \leq 0$). In the particular cases of $N = 1$, $N = 2$, and $N = 3$, N-grams are called **unigrams**, **bigrams**, and **trigrams** repectively.

Maximum Likelihood Estimation Suppose now that we have an annotated corpus made of sequences of (word, POS tag) pairs in $\Sigma \times \Theta$. Then we can estimate the probability of a given tag t appearing after $N - 1$ other tags $t_1 \cdots t_{N-1}$ by counting the number of occurrences $C(t_1 \cdots t_{N-1}t)$ of the sequence $t_1 \cdots t_{N-1}t$ and dividing by the number of occurrences of $C(t_1 \cdots t_{N-1})$ of $t_1 \cdots t_{N-1}$:

$$p(t|t_1 \cdots t_{N-1}) = \frac{C(t_1 \cdots t_{N-1}t)}{C(t_1 \cdots t_{N-1})} \tag{1.24}$$

(assuming we pad our corpus sequences with dummy \$s both on the left and on the right); this is called a **maximum likelihood estimation**.

We can build a HMM from such estimations by setting $Q = (\Theta \uplus \{\$\})^N$, i.e. using states of form $q = t_1 \cdots t_N$, and computing the next state probabilities as

$$p(t'_1 \cdots t'_N|t_1 \cdots t_N) = \begin{cases} p(t'_N|t_2 \cdots t_N) & \text{if } \forall 1 \leq i \leq N-1, t'_i = t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1.25}$$

the initial state probabilities being the particular case $p(t'_1 \cdots t'_N|\$^N)$, and the emission probabilities as

$$p(a|t_1 \cdots t_N) = \frac{1}{|\Sigma|^{N-1}} \sum_{a_1 \cdots a_{N-1} \in \Sigma^{N-1}} \frac{C((a_1, t_1) \cdots (a, t_N))}{\sum_{a_N \in \Sigma} C((a_1, t_1) \cdots (a_N, t_N))} \tag{1.26}$$

estimated from occurrences of sequences of pairs. One can then reconstruct a sequence of tags from a sequence of states by projection on the Nth component.

*The statistical distribution of words in corpora can be approximated by **Zipf's law** (see Manning and Schütze, 1999, Section 1.4.3).*

See Jurafsky and Martin (2009, Section 4.5) and Manning and Schütze (1999, Chapter 6).

Smoothing Maximum likelihood estimations are accurate if there are enough occurrences in the training corpus. Nevertheless, some valid sequences of tags or of pairs of tags and words will invariably be missing, and be assigned a zero probability. Furthermore, the estimations are also unreliable for observations with low occurrence counts.

The idea of **smoothing** is to compensate data sparseness by moving some of the probability mass from the higher counts towards the lower and null ones. This can be performed in rather crude ways (for instance add 1 to the counts on the numerators of (1.24) and (1.26) and normalize, called **Laplace smoothing**), or more involved ones that take into account the probability of observations with a single occurrence (**Good-Turing discounting**) or the probabilities of $(N-1)$-grams (**interpolation** and **backoff**). A common side-effect of all these techniques is that there are no zero-probability values left in the constructed HMMs.

HMM Decoding

Recall the POS tagging problem: find the best possible sequence of tags $t_1 \cdots t_n$, given a sequence of words $w = a_1 \cdots a_n$. Let us assume we are given a HMM model where we can reconstruct the sequence $t_1 \cdots t_n$ from the most probable execution $\rho = q_1 \cdots q_n$ that emits w, i.e. we want to compute

$$\rho = \operatorname*{argmax}_{\rho' \in Q^n} p(\rho'|w) \ , \tag{1.27}$$

which is also known as **HMM decoding**. By Bayes' inversion rule, this is the same as

$$\begin{aligned} \rho &= \operatorname*{argmax}_{\rho' \in Q^n} \frac{p(w|\rho')\, p(\rho')}{p(w)} \\ &= \operatorname*{argmax}_{\rho' \in Q^n} p(w|\rho')\, p(\rho') \ . \end{aligned} \tag{1.28}$$

The usual procedure to compute the result of (1.28) is to use the **Viterbi algorithm**, a dynamic programming algorithm. We also present another approach based on weighted automata products and shortest path algorithms, like **Dijkstra's algorithm**.

The Viterbi Algorithm Let $w = a_1 \cdots a_n$, $0 \leq i < n$, and consider the maximal joint probability $V(i+1, q)$ among all sequences of $i+1$ states ending in a given state q and of a sequence of emissions $a_1 \cdots a_{i+1}$:

$$V(i+1, q) = \max_{\rho' \in Q^i} p(a_1 \cdots a_{i+1}|\rho' q)\, p(\rho' q) \ . \tag{1.29}$$

For $i = 0$, this probability is clearly

$$V(1, q) = p(a_1|q)\, p(q|\$) = E(q)(a_1)\, S(q) \ . \tag{1.30}$$

Then, for $1 \leq i < n$,

$$\begin{aligned} V(i+1,q) &= \max_{\rho' \in Q^{i-1}, q' \in Q} p(a_1 \cdots a_i a_{i+1} | \rho' q' q)\, p(\rho' q' q) \\ &= \max_{\rho' \in Q^{i-1}, q' \in Q} p(a_1 \cdots a_i | \rho' q')\, p(a_{i+1}|q)\, p(\rho' q')\, p(q|q') \\ &= \max_{q' \in Q} V(i,q')\, p(a_{i+1}|q)\, p(q|q') \\ &= E(q)(a_{i+1}) \max_{q' \in Q} V(i,q')\, T(q')(q) \ . \end{aligned} \tag{1.31}$$

Let us introduce backpointers for the best choice at each step $1 \leq i < n$ and for each state q:

$$B(i,q) = \operatorname*{argmax}_{q' \in Q} V(i,q')\, T(q')(q) \ . \tag{1.32}$$

Let $\rho = q_1 \cdots q_n$, then the last state q_n of the most likely explanation is

$$q_n = \operatorname*{argmax}_{q \in Q} V(n,q) \ , \tag{1.33}$$

and we can work our way back from there using

$$q_i = B(i, q_{i+1}) \ , \tag{1.34}$$

for each $1 \leq i < n$ to reconstruct ρ.

Example 1.11. For the HMM of Example 1.10 and the input aa, we obtain

$$V = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.25 & 0.125 \end{pmatrix} \qquad\qquad B = (q_1\ q_2\ q_2)$$

from which we reconstruct the most likely state sequence $q_1 q_1$.

Complexity of the Viterbi Algorithm. The algorithm proceeds by computing $V(i+1,q)$ for each $0 \leq i < n$ and q in Q; the computation given by (1.31) of one of these probabilities is in $O(|Q|)$. The complexity of the computation of V dominates the other operations, and the overall complexity is thus in $O(|w|\,|Q|^2)$.

Shortest Path Approach A HMM defines a rational series $[\![\mathcal{H}]\!]$ on the probabilistic semiring. Indeed, set $Q' = Q \uplus \{q_0\}$, $I(q_0) = 1$ and $I(q \neq q_0) = 0$ and define the representation $\langle I, \mu, \bar{1} \rangle$ where, for all a in Σ and q, q' in Q,

$$\mu(a)(q_0,q_0) = 0 \quad \mu(a)(q_0,q') = S(q') \cdot E(q')(a) \quad \mu(a)(q,q') = T(q)(q') \cdot E(q')(a) \tag{1.35}$$

combines the transition and emission probabilities. Observe that the support of this series is **prefix-closed**: if $\langle [\![\mathcal{H}]\!], uv \rangle \neq 0$, then $\langle [\![\mathcal{H}]\!], u \rangle \neq 0$—this is reflected by the $\bar{1}$ final matrix in the representation. Figure 1.4 shows the probabilistic automaton that corresponds to the HMM of Example 1.10.

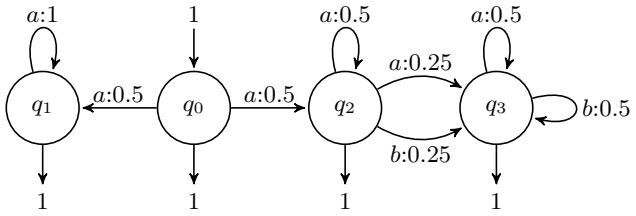


Figure 1.4: The probabilistic automaton for the HMM of Example 1.10.

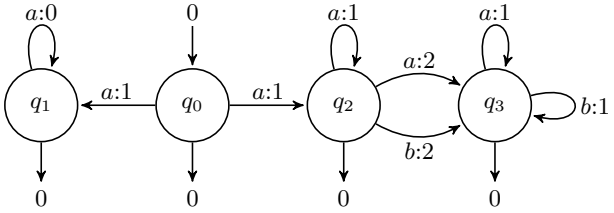


Figure 1.5: The tropical automaton $-\log \mathcal{H}$ for the HMM $\mathcal{H}$ of Example 1.10.

Our HMM decoding problem then reduces to choosing the path of maximal weight labeled by $w = a_1 \cdots a_n$ in the probabilistic automaton associated to $\mathcal{H}$ by (1.35):

$$\begin{aligned} \rho &= \operatorname*{argmax}_{\rho' \in Q^n} p(w|\rho')\, p(\rho') && (1.28) \\ &= \operatorname*{argmax}_{q_1 \cdots q_n} \prod_{i=1}^{n} p(a_i|q_i)\, p(q_i|q_{i-1}) \\ &= \operatorname*{argmax}_{q_1 \cdots q_n} S(q_1)\, E(q_1)(a_1) \prod_{i=2}^{n} E(q_i)(a_i)\, T(q_{i-1})(q_i)\ . && (1.36) \end{aligned}$$

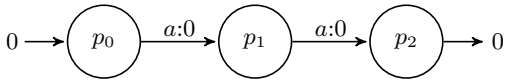
For performance reasons, we rather look for the path of minimal weight in the tropical automaton $-\log \mathcal{H}$ of representation $\langle -\log I, -\log \mu, \bar{0} \rangle$. (See Figure 1.5.) From a practical standpoint, this allows to use addition instead of multiplication, and avoids issues with the floating-point representation of real numbers close to zero. From a theoretical standpoint, a solution to (1.36) becomes

$$\rho = \operatorname*{argmin}_{q_1 \cdots q_n} \left(-\log S(q_1) - E(q_1)(a_1) - \sum_{i=2}^{n} \log E(q_i)(a_i) + \log T(q_{i-1})(q_i) \right), \tag{1.37}$$

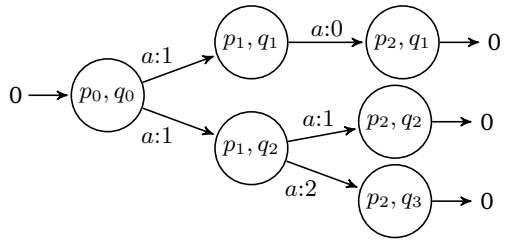
i.e. a path with weight $\langle [\![-\log \mathcal{H}]\!], w \rangle$ assigned by the tropical automaton to w.

We can effectively build the product of our weighted automaton $-\log \mathcal{H}$ with an automaton $\mathcal{W}$ for the singleton language $\{w\}$ (Figure 1.6a). The transition labels in the resulting weighted automaton $-\log \mathcal{H} + \mathcal{W}$ (Figure 1.6b) are then useless, and we can see it more simply as a weighted graph with weights in $\mathbb{R}_+$. Adding a single sink node s with edges $((p, q), 0, s)$ for each final state (p, q) of the product automaton (Figure 1.6c) then allows to use a single-pair shortest path algorithm between (p_0, q_0) and s to find a solution for (1.37) ($q_1 q_1$ in the example of Figure 1.6).

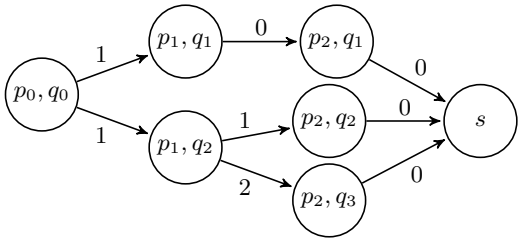
Complexity of the Shortest Path Approach. Recall that Dijkstra's algorithm with Fibonacci heaps runs in $O(m + n \log n)$ in a graph with m edges and n vertices.



(a) Tropical automaton $\mathcal{W}$ for aa. Only $+\infty$ and 0 weights are used.



(b) Product automaton $-\log \mathcal{H} + \mathcal{W}$.



(c) Weighted graph with sink state.

Figure 1.6: Construction steps for the tagging algorithm on aa.

The product automaton $-\log \mathcal{H} + \mathcal{W}$ has at most $n = |w| \cdot |Q| + 1$ states (there is a single initial state (p_0, q_0) by construction).

In practice, due to smoothing, the transition relation of $-\log \mathcal{H}$ is complete except that q_0 does not have any incoming transition: the number of transitions with weight different from $+\infty$ is $|Q|^2 + |Q|$. Luckily, the situation is not as bad with $-\log \mathcal{H} + \mathcal{W}$: its number of transitions is not n^2 but $m = (|w| - 1) \cdot |Q|^2 + |Q|$. Indeed, there are in total $|Q|$ outgoing transitions from (p_0, q_0) to each (p_1, q) with q in Q, and after that for each $1 \leq i < |w|$ there are in total $|Q|^2$ transitions between some state (p_i, q) and some state (p_{i+1}, q') with q, q' in Q.

Overall, we obtain a complexity of

$$O\big((|w| - 1)\,|Q|^2 + |Q| + (|w|\,|Q| + 1)\log(|w|\,|Q| + 1)\big) \\ = O\big(|w|\,|Q|^2 + |w|\,|Q|\,\log(|w|\,|Q|)\big)\,,$$

which is close enough to that of the Viterbi algorithm.

Sequential Series Approach One might rightly think that, even if we end up with similar complexities, the weighted automata approach induces quite a bit of extra machinery, and is of limited practical interest compared to the Viterbi algorithm.

There is however one case where the weighted automata approach yields practical advantages: when the automaton $-\log \mathcal{H}$ can be **determinized**. Recall that a solution ρ of (1.37) is a path with weight $\langle [\![-\log \mathcal{H}]\!], w \rangle$. One could thus issue the state information along with this weight and hope to perform HMM tagging deterministically, with a $O(|w| + |\mathcal{A}|)$ complexity where $\mathcal{A}$ is the determinized and minimized automaton for $-\log \mathcal{H}$.

However, unlike the rule-based tagging technique of Section 1.3.1, there is no general determinization procedure for (weighted automata translations of) HMMs. Consider for instance the automaton of Figure 1.5 for the HMM of Example 1.10: it implements the series over the tropical semiring defined by

$$\langle s, w\rangle = \begin{cases} 1 & \text{if } w = a^n, \\ n + 2 + |w'| & \text{if } w = a^n b w',\ w' \in \{a, b\}^*. \end{cases} \tag{1.38}$$

A related open problem is the decidability of sequentiality for rational series over the tropical semiring; see Lombardy and Sakarovitch (2006).

The set of translations $w^{-1}s$ for all $w \in \Sigma^*$ is not finite: for each m and n, one gets a different $\langle (a^n)^{-1}s, a^m b\rangle = n + m + 1$; thus by the pendant of Theorem 1.2 for sequential series, s is not sequential (see Lombardy and Sakarovitch, 2006, Theorem 8, where $w^{-1}s$ is noted $[w^{-1}s]^\sharp$).

Still, an incomplete determinization algorithm that might work in practice is described by Mohri (1997), and can be followed by a minimization step.

Chapter 2

Generative Syntax

Syntax deals with how words are arranged into sentences. An important body of linguistics proposes **constituent** analyses for sentences, where for instance

> Those torn books are completely worthless.

can be decomposed into a **noun phrase** *those torn books* and a **verb phrase** *are completely worthless*. These two constituants can be recursively decomposed until we reach the individual words, effectively describing a tree:

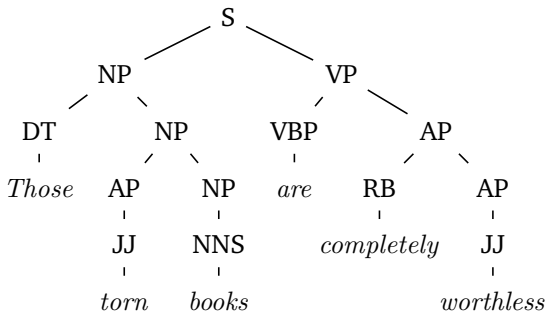


Figure 2.1: A context-free derivation tree.

You have probably recognized in this example a derivation tree for a **context-free grammar** (CFG). Context-free grammars, proposed by Chomsky (1956), constitute the primary example of a *generative* formalism for syntax, which we take to include all string- or term-rewrite systems.

See Pullum and Scholz (2001) for an account of the differences between generative and model-theoretic approaches to syntax.

Definition 2.1 (Phrase-Structured Grammars)**.** A **phrase-structured grammar** is a tuple $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ where N is a finite *nonterminal alphabet*, Σ a finite *terminal alphabet* disjoint from N, $V = N \uplus \Sigma$ the *vocabulary*, $P \subseteq V^* \times V^*$ a finite set of rewrite rules or *productions*, and S a *start symbol* or *axiom* in N.

A phrase-structure grammar defines a string rewrite system over V. Strings α in V^* s.t. $S \Rightarrow^* \alpha$ are called **sentential forms**, whereas strings w in Σ^* s.t. $S \Rightarrow^* w$ are called **sentences**. The *language* of $\mathcal{G}$ is its set of sentences, i.e.

$$L(\mathcal{G}) = L_{\mathcal{G}}(S) \qquad\qquad L_{\mathcal{G}}(A) = \{w \in \Sigma^* \mid A \Rightarrow^* w\} \ .$$

Different restrictions on the shape of productions lead to different classes of grammars; we will not recall the entire **Chomsky hierarchy** (Chomsky, 1959) here, but only define **context-free grammars** (aka **type 2 grammars**) as phrase-structured grammars with $P \subseteq N \times V^*$.

Example 2.2. The derivation tree of Figure 2.1 corresponds to the context-free grammar with

$$N = \{\text{S}, \text{NP}, \text{AP}, \text{VP}, \text{DT}, \text{JJ}, \text{NNS}, \text{VBP}, \text{RB}\} ,$$
$$\Sigma = \{\mathit{those}, \mathit{torn}, \mathit{books}, \mathit{are}, \mathit{completely}, \mathit{worthless}\} ,$$

$$\begin{aligned} P = \{ \quad & \text{S} \to \text{NP VP}, & & \text{NP} \to \text{DT NP} \mid \text{AP NP} \mid \text{NNS}, \\ & \text{VP} \to \text{VBP AP}, & & \text{AP} \to \text{RB AP} \mid \text{JJ}, \\ & \text{DT} \to \mathit{Those}, & & \text{JJ} \to \mathit{torn} \mid \mathit{worthless}, \\ & \text{NNS} \to \mathit{books}, & & \text{VBP} \to \mathit{are}, \\ & \text{RB} \to \mathit{completely}\} , & & \\ S = \text{S} . & & & \end{aligned}$$

Note that it also generates sentences such as *Those books are torn.* or *Those completely worthless books are completely completely torn.* Also note that this grammar describes POS tagging information; a different formalization could set $\Sigma = \{\text{DT}, \text{JJ}, \text{NNS}, \text{VBP}, \text{RB}\}$ and delegate the POS tagging issues to an external device, such as the sequential transducers and HMMs of Section 1.3.

The Parsing Problem Context-free grammars have a number of appreciable computational properties:

- both their **uniform membership** problem—i.e. given $\langle \mathcal{G}, w \rangle$ does $w \in L(\mathcal{G})$—and their **emptiness** problem—i.e. given $\langle \mathcal{G} \rangle$ does $L(\mathcal{G}) = \emptyset$—are PTIME-complete (Jones and Laaser, 1976),
- their **fixed grammar membership** problem—i.e. for a fixed $\mathcal{G}$, given $\langle w \rangle$ does $w \in L(\mathcal{G})$—is by very definition LOGCFL-complete (Sudborough, 1978),
- they have a natural notion of **derivation trees**, which constitute a local **regular tree language** (Thatcher, 1967).

Recall that our motivation in context-free grammars lies in their ability to model constituency through their *derivation trees*. Thus much of the linguistic interest in context-free grammars revolves around a variant of the membership problem: given $\langle \mathcal{G}, w \rangle$, compute the set of derivation trees of $\mathcal{G}$ that yield w—the **parsing problem**.

The monograph of Grune and Jacobs (2007) is a rather exhaustive resource on context-free parsing.

2.1 Context-Free Parsing

Outside the realm of deterministic parsing algorithms for restricted classes of CFGs, for instance for LL(k) or LR(k) grammars (Knuth, 1965; Kurki-Suonio, 1969; Rosenkrantz and Stearns, 1970)—which are often studied in computer science curricula—, there exists quite a variety of methods for *general* context-free parsing. Possibly the best known of these is the CKY algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967), which in its most basic form works with complexity $O(|\mathcal{G}|\,|w|^3)$ on grammars in Chomsky normal form. Both the CKY algorithm(s) and the advanced methods (Earley, 1970; Lang, 1974; Graham et al., 1980; Tomita, 1986; Billot and Lang, 1989) can be seen as refinement of the construction first described by Bar-Hillel et al. (1961) to prove the closure of context-free languages under intersection with recognizable sets, which will be central in this section, including the part on probabilistic parsing in Section 2.1.2.

The asymptotically best parsing algorithm is that of Valiant (1975), with complexity $\Theta(B(|w|))$ where $B(n)$ is the complexity of n-dimensional boolean matrix multiplication, currently in $O(n^{2.376})$ (Coppersmith and Winograd, 1990). A converse reduction from boolean matrix multiplication to context-free parsing by Lee (2002) shows that any improvement for one problem would also yield one for the other.

Ambiguity and Parse Forests The key issue in general parsing and parsing for natural language applications is grammatical **ambiguity**: the existence of several derivation trees sharing the same string yield.

The following sentence is a classical example of a PP attachment ambiguity, illustrated by the two derivation trees of Figure 2.2:

She watches a man with a telescope.

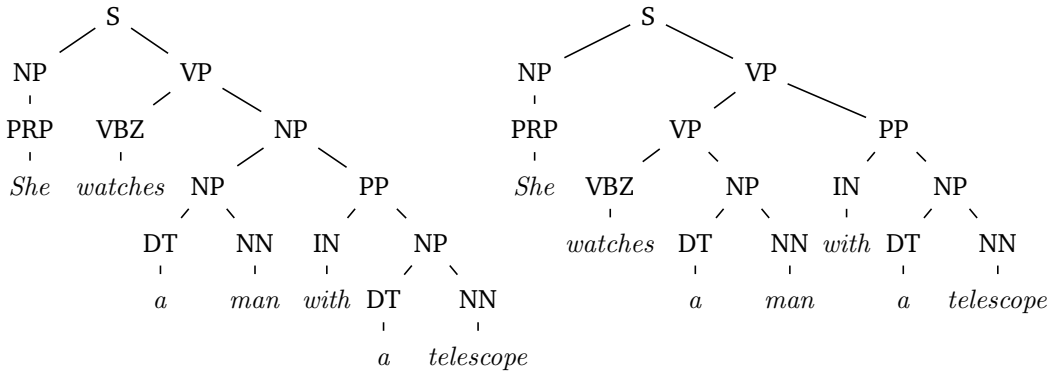


Figure 2.2: An ambiguous sentence.

In the case of a **cyclic** CFG, with a nonterminal A verifying $A \Rightarrow^+ A$, the number of different derivation trees for a single sentence can be infinite. For **acyclic** CFGs, it is finite but might be exponential in the length of the grammar and sentence:

Example 2.3 (Wich, 2005)**.** The grammar with rules

$$S \to a\ S \mid a\ A \mid \varepsilon, \qquad\qquad A \to a\ S \mid a\ A \mid \varepsilon$$

has exactly 2^n different derivation trees for the sentence a^n.

Such an explosive behavior is not unrealistic for CFGs in natural languages: Moore (2004) reports an average number of 7.2×10^{27} different derivations for sentences of 5.7 words on average, using a CFG extracted from the Penn Treebank.

The solution in order to retain polynomial complexities is to represent all these derivation trees as the language of a finite tree automaton (or using a CFG). This is sometimes called a **shared forest** representation.

Definition 2.4 (Finite Tree Automata)**.** A **finite tree automaton** (NTA) is a tuple $\mathcal{A} = \{Q, \Sigma, \delta, F\}$ where Q is a finite set of *states*, Σ a *ranked alphabet*, δ a finite *transition relation* in $\bigcup_n Q \times \Sigma_n \times Q^n$, and $I \subseteq Q$ a set of *initial states*.

See Comon et al. *(2007).*

The semantics of a NTA can be defined by term rewrite systems over $\mathcal{F} = Q \uplus \Sigma$ where the states in Q have arity 0: either **bottom-up**:

$$R_B = \{a^{(n)}(q_1^{(0)}, \ldots, q_n^{(0)}) \to q^{(0)} \mid (q, a^{(n)}, q_1, \ldots, q_n) \in \delta\}$$
$$L(\mathcal{A}) = \{t \in T(\Sigma) \mid \exists q \in I, t \xrightarrow{R_B}{}^* q\}\ ,$$

or **top-down**:

$$R_T = \{q^{(0)} \to a^{(n)}(q_1^{(0)}, \ldots, q_n^{(0)}) \mid (q, a^{(n)}, q_1, \ldots, q_n) \in \delta\}$$
$$L(\mathcal{A}) = \{t \in T(\Sigma) \mid \exists q \in I, q \xrightarrow{R_T}{}^* t\}\ .$$

Example 2.5. The 2^n derivation trees for a^n in the grammar of Example 2.3 are generated by the $O(n)$-sized automaton $\langle\{q_S, q_a, q_\varepsilon, q_1, \ldots, q_n\}, \{S, A, a, \varepsilon\}, \delta, \{q_S\}\rangle$ with rules

$$\begin{aligned}\delta = &\{(q_S, S^{(2)}, q_a, q_1), (q_a, a^{(0)}), (q_\varepsilon, \varepsilon^{(0)})\}\\ \cup\ &\{(q_i, X, q_a, q_{i+1}) \mid 1 \leq i < n, X \in \{S^{(2)}, A^{(2)}\}\}\\ \cup\ &\{(q_n, X, q_\varepsilon) \mid X \in \{S^{(1)}, A^{(1)}\}\}\ .\end{aligned}$$

2.1.1 Tabular Parsing

We briefly survey the principles of general context-free parsing using **dynamic** or **tabular** algorithms. For more details, see Nederhof and Satta (2004).

Parsing as Intersection

The basic construction underlying all the tabular parsing algorithms is the *intersection* grammar of Bar-Hillel et al. (1961). It consists in an intersection between an $(|w|+1)$-sized automaton with language $\{w\}$ and the CFG under consideration. The intersection approach is moreover quite convenient if several input strings are possible, for instance if the input of the parser is provided by a speech recognition system.

A landmark paper on the importance of the Bar-Hillel et al. *(1961) construction for parsing is Lang (1994). We provide here a tree automaton variant; we will see a more general, weighted CFG variant in Section 2.1.2.*

Theorem 2.6 (Bar-Hillel et al., 1961)**.** *Let $\mathcal{G} = \langle N, \Sigma, P, S\rangle$ be a CFG and $\mathcal{A} = \langle Q, \Sigma, \delta, I, F\rangle$ be a NFA. The set of derivation trees of $\mathcal{G}$ with a word of $L(\mathcal{A})$ as yield is generated by the NTA $\mathcal{T} = \langle (V \uplus \{\varepsilon\}) \times Q \times Q, \Sigma \uplus N \uplus \{\varepsilon\}, \delta', \{S\} \times I \times F\rangle$ with*

$$\begin{aligned}\delta' = &\{((A, q_0, q_m), A^{(m)}, (X_1, q_0, q_1), \ldots, (X_m, q_{m-1}, q_m))\\ &\qquad \mid m \geq 1, A \to X_1 \cdots X_m \in P, q_0, q_1, \ldots, q_m \in Q\}\\ \cup\ &\{((A, q, q), A^{(1)}, (\varepsilon, q, q)) \mid A \to \varepsilon \in P, q \in Q\}\\ \cup\ &\{((\varepsilon, q, q), \varepsilon^{(0)}) \mid q \in Q\}\\ \cup\ &\{((a, q, q'), a^{(0)}) \mid (q, a, q') \in \delta\}\ .\end{aligned}$$

The size of the resulting NTA is in $O(|\mathcal{G}| \cdot |Q|^{m+1})$ where m is the maximal arity of a nonterminal in N. We can further reduce this NTA to only keep useful states, in linear time on a RAM machine. It is also possible to determinize and minimize the resulting tree automaton.

In order to reduce the complexity of this construction to $O(|\mathcal{G}| \cdot |Q|^3)$, one can put the CFG in **quadratic form**, so that $P \subseteq N \times V^{\leq 2}$. This changes the shape of trees, and thus the linguistic analyses, but the transformation is reversible:

Lemma 2.7. *Given a CFG $\mathcal{G} = \langle \Sigma, N, P, S\rangle$, one can construct in time $O(|\mathcal{G}|)$ an equivalent CFG $\mathcal{G}' = \langle \Sigma, N', P', S\rangle$ in quadratic form s.t. $V \subseteq V'$, $L_{\mathcal{G}}(X) = L_{\mathcal{G}'}(X)$ for all X in V, and $|\mathcal{G}'| \leq 5 \cdot |\mathcal{G}|$.*

Proof. For every production $A \to X_1 \cdots X_m$ of P with $m \geq 2$, add productions

$$\begin{aligned}A &\to [X_1][X_2 \cdots X_m]\\ [X_2 \cdots X_m] &\to [X_2][X_3 \cdots X_m]\\ &\vdots\\ [X_{m-1}X_m] &\to [X_{m-1}][X_m]\end{aligned}$$

and for all $1 \leq i \leq m$

$$[X_i] \to X_i \ .$$

Thus an $(m+1)$-sized production is replaced by $m-1$ productions of size 3 and m productions of size 2, for a total less than $5m$. Formally,

$$\begin{aligned}
N' &= N \cup \{[\beta] \mid \beta \in V^+ \text{ and } \exists A \in N, \alpha \in V^+, A \to \alpha\beta \in P\} \\
&\qquad \cup \{[X] \mid X \in V \text{ and } \exists A \in N, \alpha, \beta \in V^*, A \to \alpha X \beta \in P\} \\
P' &= \{A \to \alpha \in P \mid |\alpha| \leq 1\} \\
&\quad \cup \{A \to [X][\beta] \mid A \to X\beta \in P, X \in V \text{ and } \beta \in V^+\} \\
&\quad \cup \{[X\beta] \to [X][\beta] \mid [X\beta] \in N', X \in V \text{ and } \beta \in V^+\} \\
&\quad \cup \{[X] \to X \mid [X] \in N' \text{ and } X \in V\} \ .
\end{aligned}$$

Grammar $\mathcal{G}'$ est clearly in quadratic form with $N \subseteq N'$ and $|\mathcal{G}'| \leq 5 \cdot |\mathcal{G}|$. It remains to show equivalence, which stems from $L_{\mathcal{G}}(X) = L_{\mathcal{G}'}(X)$ for all X in V. Obviously, $L_{\mathcal{G}}(X) \subseteq L_{\mathcal{G}'}(X)$. Conversely, by induction on the length n of derivations in $\mathcal{G}'$, we prove that

$$X \Rightarrow_{\mathcal{G}'}^{n} w \text{ implies } X \Rightarrow_{\mathcal{G}}^{*} w \tag{2.1}$$

$$[\alpha] \Rightarrow_{\mathcal{G}'}^{n} w \text{ implies } \alpha \Rightarrow_{\mathcal{G}}^{*} w \tag{2.2}$$

for all X in V, w in Σ^*, and $[\alpha]$ in $N' \backslash N$. The base case $n = 0$ implies X in Σ and the lemma holds. Suppose it holds for all $i < n$.

From the shape of the productions in $\mathcal{G}'$, three cases can be distinguished for a derivation

$$X \Rightarrow_{\mathcal{G}'} \beta \Rightarrow^{n-1}_{\mathcal{G}'} w \ :$$

1. $\beta = \varepsilon$ implies immediately $X \Rightarrow_{\mathcal{G}}^{*} w = \varepsilon$, or
2. $\beta = Y$ in V implies $X \Rightarrow_{\mathcal{G}}^{*} w$ by induction hypothesis (2.1), or
3. $\beta = [Y][\gamma]$ with $[Y]$ and $[\gamma]$ in N' implies again $X \Rightarrow_{\mathcal{G}}^{*} w$ by induction hypothesis (2.2) and context-freeness, since in that case $X \to Y\gamma$ is in P.

Similarly, a derivation

$$[\alpha] \Rightarrow_{\mathcal{G}'} \beta \Rightarrow_{\mathcal{G}'}^{n-1} w$$

implies $\alpha \Rightarrow^* w$ by induction hypothesis (2.1) if $|\alpha| = 1$ and thus $\beta = \alpha$, or by induction hypothesis (2.2) and context-freeness if $\alpha = Y\gamma$ with Y in V and γ in V^+, and thus $\beta = [Y][\gamma]$. □

Parsing as Deduction

In practice, we want to perform at least some of the reduction of the tree automaton constructed by Theorem 2.6 *on the fly*, in order to avoid constructing states and transitions that will be later discarded as useless.

Bottom-Up Tabular Parsing One way is to restrict ourselves to **co-accessible** states, by which we mean states q of the NTA such that there exists at least one tree t with $t \xrightarrow{R_B}{}^{*} q$. This is the principle underlying the classical CKY parsing algorithm (but here we do not require the grammar to be in Chomsky normal form).

We describe the algorithm using deduction rules (Pereira and Warren, 1983; Sikkel, 1997), which conveniently represent how new tabulated **items** can be constructed from previously computed ones: in this case, items are states (A, q, q') in $V \times Q \times Q$ of the constructed NTA. Side conditions constrain how a deduction rule can be applied.

$$\frac{(X_1, q_0, q_1), \ldots, (X_m, q_{m-1}, q_m)}{(A, q_0, q_m)} \left\{ \begin{array}{l} A \to X_1 \cdots X_m \in P \\ q_0, q_1, \ldots, q_m \in Q \end{array} \right. \quad \text{(Internal)}$$

$$\frac{}{(a, q, q')} \left\{ \; (q, a, q') \in \delta \right. \quad \text{(Leaf)}$$

The construction of the NTA proceeds by creating new states following the rules, and transitions of δ' as output to the deduction rules, i.e. an application of (Internal) outputs if $m \geq 1$ $((A, q_0, q_m), A^{(m)}, (X_1, q_0, q_1), \ldots, (X_m, q_{m-1}, q_m))$, or if $m = 0$ $((A, q_0, q_0), A^{(1)}, (\varepsilon, q_0, q_0))$, and one of (Leaf) outputs $((a, q, q'), a^{(0)})$. We only need to add states (ε, q, q) and transitions $((\varepsilon, q, q), \varepsilon^{(0)})$ for each q in Q in order to obtain the co-accessible part of the NTA of Theorem 2.6.

The algorithm performs the deduction closure of the system; the intersection itself is non-empty if an item in $\{S\} \times I \times F$ appears at some point. The complexity depends on the "free variables" in the premices of the rules and on the side constraints; here it is dominated by the (Internal) rule, with at most $|\mathcal{G}| \cdot |Q|^{m+1}$ applications.

We could similarly construct a system of **top-down** deduction rules that only construct **accessible** states of the NTA, starting from (S, q_i, q_f) with q_i in I and q_f in F, and working its way towards the leaves.

(∗) Exercise 2.1. Give the deduction rules for top-down tabular parsing.

Earley Parsing The algorithm of Earley (1970) uses a mix of accessibility and co-accessibility. An *Earley item* is a triple $(A \to \alpha \cdot \beta, q, q')$, q, q' in Q and $A \to \alpha\beta$ in P, constructed iff

This invariant proves the correctness of the algorithm. For a more original proof using abtract interpretation, see Cousot and Cousot (2003).

1. there exists both (i) a run of $\mathcal{A}$ starting in q and ending in q' with label v and (ii) a derivation $\alpha \Rightarrow^* v$, and furthermore
2. there exists (i) a run in $\mathcal{A}$ from some q_i in I to q with label u and (ii) a derivation $S \underset{\text{lm}}{\Longrightarrow}^* uA\gamma$ for some γ in V^*.

$$\frac{}{(S \to \cdot\alpha, q_i, q_i)} \left\{ \begin{array}{l} S \to \alpha \in P \\ q_i \in I \end{array} \right. \quad \text{(Init)}$$

$$\frac{(A \to \alpha \cdot B\alpha', q, q')}{(B \to \cdot\beta, q', q')} \left\{ \; B \to \beta \in P \right. \quad \text{(Predict)}$$

$$\frac{(A \to \alpha \cdot a\alpha', q, q')}{(A \to \alpha a \cdot \alpha', q, q'')} \left\{ \; (q', a, q'') \in \delta \right. \quad \text{(Scan)}$$

$$\frac{\begin{array}{c}(A \to \alpha \cdot B\alpha', q, q') \\ (B \to \beta\cdot, q', q'')\end{array}}{(A \to \alpha B \cdot \alpha', q, q'')} \quad \text{(Complete)}$$

The intersection is non empty if an item $(S \to \alpha\cdot, q_i, q_f)$ is obtained for some q_i in I and q_f in F.

The algorithm run as a recognizer works in $O(|\mathcal{G}|^2 \cdot |Q|^3)$ regardless of the arity of symbols in $\mathcal{G}$ ((Complete) dominates this complexity), and can be further optimized to run in $O(|\mathcal{G}| \cdot |Q|^3)$, which is the object of Exercise 2.2. This cubic complexity in the size of the automaton can be understood as the effect of an on-the-fly quadratic form transformation into $\mathcal{G}' = \langle N', \Sigma, P', S' \rangle$ with

$$\begin{aligned}
N' &= \{S'\} \uplus \{[A \to \alpha \cdot \beta] \mid A \to \alpha\beta \in P\} \\
P' &= \{S' \to [S \to \alpha\cdot] \mid S \to \alpha \in P\} \\
&\quad \cup\ \{[A \to \alpha B \cdot \alpha'] \to [A \to \alpha \cdot B\alpha']\ [B \to \beta\cdot] \mid B \to \beta \in P\} \\
&\quad \cup\ \{[A \to \alpha a \cdot \alpha'] \to [A \to \alpha \cdot a\alpha']\ a \mid a \in \Sigma\} \\
&\quad \cup\ \{[A \to \cdot\alpha'] \to \varepsilon\}\ .
\end{aligned}$$

Note that the transformation yields a grammar of quadratic size, but can be modified to yield one of linear size—this is the same simple trick as that of Exercise 2.2. It is easier to output a NTA for this transformed grammar $\mathcal{G}'$:

- create state $([S \to \cdot\alpha], q_i, q_i)$ and transition $(([S \to \cdot\alpha], q_i, q_i), \varepsilon^{(0)})$ when applying (Init),
- create state $([B \to \cdot\beta], q', q')$ and transition $(([B \to \cdot\beta], q', q'), \varepsilon^{(0)})$ when applying (Predict),
- create states $([A \to \alpha a \cdot \alpha'], q, q'')$ and (a, q', q''), and transitions $(([A \to \alpha a \cdot \alpha'], q, q''), [A \to \alpha a \cdot \alpha']^{(2)}, ([A \to \alpha \cdot a\alpha'], q, q'), (a, q', q''))$ and $((a, q', q''), a^{(0)})$ when applying (Scan),
- create state $([A \to \alpha B \cdot \alpha'], q, q'')$ and transition $(([A \to \alpha B \cdot \alpha'], q, q''), [A \to \alpha B \cdot \alpha']^{(2)}, ([A \to \alpha \cdot B\alpha'], q, q'), ([B \to \beta\cdot], q', q'')$ when applying (Complete).

We finally need to add states (S', q_i, q_f) for q_i in I and q_f in F, and transitions $((S', q_i, q_f), S'^{(1)}, ([S \to \alpha\cdot], q_i, q_f)$ for each $S \to \alpha$ in P.

Exercise 2.2. How should the algorithm be modified in order to run in time $O(|\mathcal{G}| \cdot |Q|^3)$ instead of $O(|\mathcal{G}|^2 \cdot |Q|^3)$? **(∗)**

Exercise 2.3. Show that the Earley recognizer works in time $O(|\mathcal{G}| \cdot |Q|^2)$ if the grammar is unambiguous and the automaton deterministic. **(∗)**

A related open problem is whether fixed grammar membership can be solved in time $O(|w|)$ if $\mathcal{G}$ is unambiguous. See Leo (1991) for a partial answer in the case where $\mathcal{G}$ is LR-Regular.

The presentation of this section follows closely Nederhof and Satta (2008).

2.1.2 Probabilistic Parsing

Probabilistic approaches to parsing are helpful on (at least) two different grounds:

1. the first is *ambiguity* issues; in order to choose between the various possible parses of a sentence, like the PP attachment ambiguity of Figure 2.2, we can resort to several techniques: heuristics, semantic processing, and what interests us in this section, probabilities learned from a corpus.
2. the second is *robustness* of the parser: rather than discarding a sentence as agrammatical or returning a partial parse, a probabilistic parser with smoothed probabilities will still propose several parses with low probabilities.

Weighted and Probabilistic CFGs

Definition 2.8 (Weighted Context-Free Grammars)**.** A **weighted context-free grammar** $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$ over a semiring $\mathbb{K}$ ($\mathbb{K}$-CFG) is a context-free grammar $\langle N, \Sigma, P, S \rangle$ along with a mapping $\rho : P \to \mathbb{K}$, which is extended in a natural way into a morphism from $\langle P^*, \cdot, \varepsilon \rangle$ to $\langle \mathbb{K}, \odot, 1_\mathbb{K} \rangle$. The *weight* of a leftmost derivation $\alpha \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^* \beta$ is then defined as $\rho(\pi)$. It would be natural to define the weight of a sentential form γ as the sum of the weights $\rho(\pi)$ with $S \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^* \gamma$, i.e.

Considering leftmost *derivations is only important if* $\langle \mathbb{K}, \odot, 1_\mathbb{K} \rangle$ *is non-commutative.*

$$\rho(\gamma) = \sum_{\pi \in P^*, S \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^* \gamma} \rho(\pi) \ .$$

However this sum might be infinite in general, and lead to weights outside $\mathbb{K}$. We therefore restrict ourselves to **acyclic** $\mathbb{K}$-CFGs, such that $A \Rightarrow^+ A$ is impossible for all A in N, ensuring that there exist only finitely many derivations for each sentential form. An acyclic $\mathbb{K}$-CFG $\mathcal{G}$ then defines a formal series $[\![\mathcal{G}]\!]$ with coefficients $\langle [\![\mathcal{G}]\!], w \rangle = \rho(w)$.

A $\mathbb{K}$-CFG $\mathcal{G}$ is **reduced** if each nonterminal A in $N \backslash \{S\}$ is **useful**, which means that there exist π_1, π_2 in P^*, u, v in Σ^*, and γ in V^* such that $S \underset{\text{lm}}{\overset{\pi_1}{\Longrightarrow}}{}^* uA\gamma \underset{\text{lm}}{\overset{\pi_2}{\Longrightarrow}}{}^* uv$ and $\rho(\pi_1\pi_2) \neq 0_\mathbb{K}$.

A $\mathbb{R}_+$-CFG $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$ is a **probabilistic context-free grammar** (PCFG) if ρ is a mapping $P \to [0, 1]$.

The following exercise shows that the treatment of probabilistic context-free parsing generalizes that of HMM decoding in Section 1.3.2:

(**) Exercise 2.4. A **right linear** $\mathbb{K}$-CFG $\mathcal{G}$ has its productions in $N \times (\Sigma^* \cup \Sigma^* \cdot N)$. Show that a series s over Σ is $\mathbb{K}$-recognizable iff there exists an acyclic right linear $\mathbb{K}$-CFG for it.

Proper and Consistent PCFGs Definition 2.8 makes no provision on the kind of probability distributions defined by a PCFG. We define here two such conditions, properness and consistency (Booth and Thompson, 1973).

A PCFG is **proper** if for all A in N,

$$\sum_{p = A \to \alpha \in P} \rho(p) = 1 \ , \tag{2.3}$$

i.e. ρ can be seen as a mapping from N to $\mathrm{Disc}(\{p \in P \mid p = A \to \alpha\})$.

The **partition function** Z maps each nonterminal A to

$$Z(A) = \sum_{w \in \Sigma^*, A \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^* w} \rho(\pi) \ . \tag{2.4}$$

A PCFG is **convergent** if

$$Z(S) < \infty \ ; \tag{2.5}$$

in particular, it is **consistent** if

$$Z(S) = 1 \ , \tag{2.6}$$

i.e. ρ defines a discrete probability distribution over the derivations of terminal strings. The intuition behind proper inconsistent grammars is that some of the probability mass is lost into infinite, non-terminating derivations.

Equation (2.4) can be decomposed using commutativity of multiplication into

$$Z(A) = \sum_{p=A\to\alpha\in P} \rho(p)\cdot Z(\alpha) \qquad \text{for all } A \text{ in } N \tag{2.7}$$

$$Z(a) = 1 \qquad \text{for all } a \text{ in } \Sigma \uplus \{\varepsilon\} \tag{2.8}$$

$$Z(X\alpha) = Z(X)\cdot Z(\beta) \qquad \text{for all } (X,\beta) \text{ in } V\times V^*. \tag{2.9}$$

This describes a monotone system of equations with the $Z(A)$ for A in N as variables.

Example 2.9. Properness and consistency are two distinct notions. For instance, the PCFG

$$S \xrightarrow{p_1} S\,S \qquad \rho(p_1) = q$$
$$S \xrightarrow{p_2} a \qquad \rho(p_2) = (1-q)$$

is proper for all $0 \le q \le 1$, but $Z(S)$ is the least solution of $x = qx^2 + 1 - q$, thus if $q \le \frac{1}{2}$ the grammar is consistent, but otherwise $Z(S) = \frac{1-q}{q} < 1$.

Conversely,

$$S \xrightarrow{p_1} A \qquad \rho(p_1) = \frac{q}{1-q}$$
$$A \xrightarrow{p_2} A\,A \qquad \rho(p_2) = q$$
$$A \xrightarrow{p_3} a \qquad \rho(p_3) = 1-q$$

is improper but consistent for $\frac{1}{2} < q < 1$.

See Booth and Thompson (1973); Gecse and Kovács (2010) for ways to check for consistency, and Etessami and Yannakakis (2009) for ways to compute $Z(A)$. In general, $Z(A)$ has to be approximated:

Remark 2.10 (Etessami and Yannakakis, 2009, Theorem 3.2)**.** The partition function of S can be irrational even when ρ maps productions to rationals in $[0,1]$:

$$S \xrightarrow{p_1} S\,S\,S\,S\,S \qquad \rho(p_1) = \frac{1}{6}$$
$$S \xrightarrow{p_2} a \qquad \rho(p_2) = \frac{1}{2}\,.$$

The associated equation is $x = \frac{1}{6}x^5 + \frac{1}{2}$, which has no rational root.

Normalization Given $Z(A)$ for all A in N, one can furthermore **normalize** any reduced convergent PCFG $\mathcal{G} = \langle N,\Sigma,P,S,\rho\rangle$ with $Z(S) > 0$ into a proper and consistent PCFG $\mathcal{G}' = \langle N,\Sigma,P,S,\rho'\rangle$. Define for this

$$\rho'(p = A\to\alpha) = \frac{\rho(p)Z(\alpha)}{Z(A)}\,. \tag{2.10}$$

Exercise 2.5. Show that in a reduced convergent PCFG with $Z(S) > 0$, for each α in V^*, one has $0 < Z(\alpha) < \infty$. (This justifies that (2.10) is well-defined.) **(∗)**

(∗) Exercise 2.6. Show that $\mathcal{G}'$ is a proper PCFG.

(∗∗) Exercise 2.7. Show that for all γ in V^*, π in P^*, and w in Σ^* with $\gamma \overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}{}^* w$,

$$\rho'(\pi) = \frac{\rho(\pi)}{Z(\gamma)} \; . \tag{2.11}$$

Equation (2.11) shows that $\mathcal{G}'$ is consistent, since

$$Z'(S) = \sum_{w \in \Sigma^*, S \overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}{}^* w} \rho'(\pi) = \sum_{w \in \Sigma^*, S \overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}{}^* w} \frac{\rho(\pi)}{Z(S)} = \frac{Z(S)}{Z(S)} = 1 \; . \tag{2.12}$$

It also yields for all w in Σ^*

$$\rho'(w) = \frac{\rho(w)}{Z(S)} \; , \tag{2.13}$$

thus the ratios between derivation weights are preserved (see Nederhof and Satta, 2003, for details).

Example 2.11. Considering again the first grammar of Example 2.9, if $q > \frac{1}{2}$, then ρ' with $\rho'(p_1) = \frac{q\,Z(S)^2}{Z(S)} = 1 - q$ and $\rho'(p_2) = q$ fits.

Learning PCFGs

As in Section 1.3.2 we rely on an annotated corpus for **supervised** learning. Again we consider the Penn Treebank (Marcus et al., 1993) as an example of such an annotated corpus, made of n trees.

Maximum Likelihood Estimation Assuming the treebank to be well-formed, i.e. that the labels of internal nodes and those of leaves are disjoint, we can collect all the labels of internal tree nodes as nonterminals, all the labels of tree leaves as terminals, and all elementary subtrees (i.e. all the subtrees of height one) as productions. Introducing a new start symbol S' with productions $S' \to S$ for each label S of a root node ensures a unique start symbol. The treebank itself can then be seen as a multiset of leftmost derivations $D = \{\pi_1, \ldots, \pi_n\}$.

Let $C(p, \pi)$ be the count of occurrences of production p inside derivation π, and $C(A, \pi) = \sum_{p=A\to\alpha\in P} C(p, \pi)$. Summing over the entire treebank, we get $C(p, D) = \sum_{\pi\in D} C(p, \pi)$ and $C(A, D) = \sum_{\pi\in D} C(A, \pi)$. The estimated probability of a production is then (see e.g. Chi and Geman, 1998)

$$\rho(p = A \to \alpha) = \frac{C(p, D)}{C(A, D)} \; . \tag{2.14}$$

(∗∗) Exercise 2.8. Show that the obtained PCFG is proper and consistent.

Preprocessing the Treebank The PCFG estimated from a treebank is typically not very good: the linguistic annotations are too coarse-grained, and nonterminals do not capture enough context to allow for a precise parsing.

Refining Nonterminals. For instance, PP attachment ambiguities are typically resolved as high attachments (i.e. to the VP) when the verb expects a PP complement, as with the following *hurled... into* construction, and a low attachment (i.e. to the NP) otherwise, as in the following *sip of ...* construction:

[NP He] [VP[VP hurled [NP the ball]] [PP into the basket]].
[NP She] [VP took [NP[NP a sip] [PP of water]]].

A PCFG cannot assign different probabilities to the attachment choices if the extracted rules are the same.

In practice, the tree annotations are refined in two directions: from the lexical leaves by tracking the **head** information, and from the root by remembering the **parent** or **grandparent** label. This greatly increases the sets of nonterminals and rules, thus some smoothing techniques are required to compensate for data sparseness. Figure 2.3 illustrates this idea by associating lexical head and parent information to each internal node. Observe that the PP attachment probability is now specific to a production

$$\text{VP}[\text{S}, \textit{hurled}, \text{VBD}] \rightarrow \text{VP}[\text{VP}, \textit{hurled}, \text{VBD}]\ \text{PP}[\text{VP}, \textit{into}, \text{IN}]\ ,$$

allowing to give it a higher probability than that of

$$\text{VP}[\text{S}, \textit{took}, \text{VBD}] \rightarrow \text{VP}[\text{VP}, \textit{took}, \text{VBD}]\ \text{PP}[\text{VP}, \textit{of}, \text{IN}]\ .$$

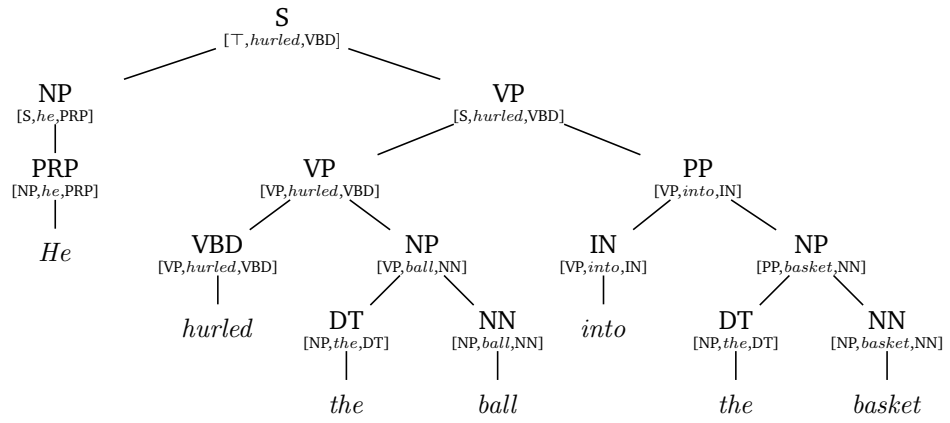


Figure 2.3: A derivation tree refined with lexical and parent information.

Binary Rules. Another issue, which is more specific to the kind of linguistic analyses found in the Penn Treebank, is that trees are mostly *flat*, resulting in a very large number of long, different rules, like

$$\text{VP} \rightarrow \text{VBP PP PP PP PP PP ADVP PP}$$

for sentence

This mostly happens because we [VP go [PP from football] [PP in the fall] [PP to lifting] [PP in the winter] [PP to football] [ADVP again] [PP in the spring]].

The *WSJ* part of the Penn Treebank yields about 17,500 distinct rules, causing important data sparseness issues in probability estimations. A solution is to transform the resulting grammar into **quadratic form** prior to probability estimation, for instance by having rules

$$\text{VP} \rightarrow \text{VBP VP'} \qquad\qquad \text{VP'} \rightarrow \text{PP} \mid \text{PP VP'} \mid \text{ADVP VP'}\ .$$

Parser Evaluation The usual measure of constituent parser performance is called PARSEVAL (Black et al., 1991). It supposes that some **gold standard** derivation trees are available for sentences, as in a test subcorpus of the *Wall Street Journal* part of the Penn Treebank, and compares the candidate parses with the gold ones. The comparison is constituent-based: correctly identified constituents start and end at the expected point and are labeled with the appropriate nonterminal symbol. The evaluation measures the

labeled recall which is the number of correct constituents in the candidate parse of a sentence, divided by the number of constituents in the gold standard analysis of the sentence,

labeled precision which is the number of correct constituents in the candidate parse of a sentence divided by the number of constituents in the same candidate parse.

Current probabilistic parsers on the *WSJ* treebank obtain a bit more than 90% precision and recall. Beware however that long sentences are often parsed incorrectly, i.e. have at least one misparsed constituent.

Probabilistic Parsing as Intersection

We generalize in this section the intersective approaches of Theorem 2.6 and Section 1.3.2. More precisely, we show how to construct a product grammar from a weighted grammar and a weighted automaton over a commutative semiring, and then use a generalized version of Dijkstra's algorithm due to Knuth (1977) to find the most probable parse in this grammar.

Weighted Product We generalize here Theorem 2.6 to the weighted case. Observe that it also answers Exercise 1.7 since $\mathbb{K}$-automata are equivalent to right-linear $\mathbb{K}$-CFGs according to Exercise 2.4.

Theorem 2.12. *Let $\mathbb{K}$ be a commutative semiring, $\mathcal{G} = \langle N, \Sigma, P, S, \rho\rangle$ an acyclic $\mathbb{K}$-CFG, and $\mathcal{A} = \langle Q, \Sigma, \mathbb{K}, \delta, I, F\rangle$ a $\mathbb{K}$-automaton. Then the $\mathbb{K}$-CFG $\mathcal{G}' = \langle \{S'\} \uplus (N \times Q \times Q), \Sigma, P', S', \rho'\rangle$ with*

We abuse notation and write $A \xrightarrow{k} \alpha$ for a production $p = A \to \alpha$ with $\rho(p) = k$.

$$
\begin{aligned}
P' = \{ & S' \xrightarrow{I(q_i)\odot F(q_f)} (S, q_i, q_f) \mid q_i, q_f \in Q\} \\
\cup\ & \{(A, q_0, q_m) \xrightarrow{k} (X_1, q_0, q_1)\cdots(X_m, q_{m-1}, q_m) \\
& \qquad\qquad \mid m \geq 1, A \xrightarrow{k} X_1 \cdots X_m \in P, q_0, \ldots, q_m \in Q\} \\
\cup\ & \{(a, q, q') \xrightarrow{k} a \mid (q, a, k, q') \in \delta\}
\end{aligned}
$$

is acyclic and such that, for all w in Σ^*, $\langle [\![\mathcal{G}']\!], w\rangle = \langle [\![\mathcal{G}]\!], w\rangle \odot \langle [\![\mathcal{A}]\!], w\rangle$.

As with Theorem 2.6, the construction of Theorem 2.12 works in time $O(|\mathcal{G}| \cdot |Q|^{m+1})$ with m the maximal length of a rule rightpart in $\mathcal{G}$. Again, this complexity can be reduced by first transforming $\mathcal{G}$ into **quadratic form**, thus yielding a $O(|\mathcal{G}| \cdot |Q|^3)$ construction.

(∗) Exercise 2.9. Modify the quadratic form construction of Lemma 2.7 for the weighted case.

Finding the Most Probable Parse The weighted CFG $\mathcal{G}'$ constructed by Theorem 2.12 can be reduced by a generalization of the usual CFG reduction algorithm to the weighted case. Here we rather consider the issue of finding the best parse in this intersection grammar $\mathcal{G}'$, assuming we are working on the probabilistic semiring—we could also work on the tropical semiring.

Non Recursive Case. The easiest case is that of a **non recursive** $\mathbb{K}$-CFG $\mathcal{G}'$, i.e. where there does not exist a derivation $A \Rightarrow^+ \delta A \gamma$ for any A in N and δ, γ in V^* in the underlying grammar. This is necessarily the case with Theorem 2.12 if $\mathcal{G}$ is acyclic and $\mathcal{A}$ has a finite support language. Then a **topological sort** of the nonterminals of $\mathcal{G}'$ for the partial ordering $B \prec A$ iff there exists a production $A \to \alpha B \beta$ in P' with α, β in V'^* can be performed in linear time, yielding a total order $(N', <)$: $A_1 < A_2 < \cdots < A_{|N'|}$. We can then compute the probability $M(S')$ of the most probable parse by computing for $j = 1, \ldots, |N'|$

$$M(A_j) = \max_{A \xrightarrow{k} X_1 \cdots X_m} k \cdot M(X_1) \cdots M(X_m) \tag{2.15}$$

in the probabilistic semiring, with $M(a) = 1$ for each a in Σ. The topological sort ensures that the maximal values $M(X_i)$ in the right-hand side have already been computed.

Knuth's Algorithm. In the case of a recursive PCFG, the topological sort approach fails. We can nevertheless use an extension of Dijkstra's algorithm to weighted CFGs proposed by Knuth (1977):

Algorithm 1: Most probable derivation.

Data: $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$
1 **foreach** $a \in \Sigma$ **do**
2 | $M(a) = 1$
3 **end**
4 $D \longleftarrow \Sigma$
5 **while** $D \neq V$ **do**
6 | **foreach** $A \in V \backslash D$ **do**
7 | | $\nu(A) \longleftarrow \max_{A \xrightarrow{k} X_1 \cdots X_m \text{ s.t. } X_1, \ldots, X_m \in D} k \cdot M(X_1) \cdots M(X_m)$
8 | **end**
9 | $A \longleftarrow \operatorname{argmax}_{V \backslash D} \nu(A)$
10 | $M(A) \longleftarrow \nu(A)$
11 | $D \longleftarrow D \uplus \{A\}$
12 **end**
13 **return** $M(S)$

The set $D \subseteq V$ is the set of symbols X for which $M(X)$, the probability of the most probable tree rooted in X, has been computed. Using a priority queue for extracting elements of $V \backslash D$ in time $\log |N|$ at line 9, and tracking which productions to consider for the computation of $\nu(A)$ at line 7, the time complexity of the algorithm is in $O(|P| \log |N| + |\mathcal{G}|)$.

The correctness of the algorithm relies on the fact that $M(A) = \nu(A)$ at line 10; assuming the opposite, there must exist a shortest derivation $B \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^* w$ with $\rho(\pi) > \nu(A)$ for some $B \notin D$. We can split this derivation into $B \underset{\text{lm}}{\overset{p}{\Longrightarrow}}{}^* X_1 \cdots X_m$ and

$X_i \overset{\pi_i}{\underset{\text{lm}}{\Rightarrow}}{}^* w_i$ with $w = w_1 \cdots w_m$ and $\pi = p\pi_1 \cdots \pi_m$, thus with $\rho(\pi) = \rho(p) \cdot \rho(\pi_1) \cdots \rho(\pi_m)$. If each X_i is already in D, then $M(X_i) \geq \rho(\pi_i)$ for all i, thus $\rho(\pi) \leq \nu(B)$ computed at line 7, and finally $\rho(\pi) \leq \nu(B) \leq \nu(A)$ by line 10—a contradiction. Therefore there must be one X_i not in D for some i, but in that case $\rho(\pi_i) \geq \rho(\pi) > \nu(A)$ and π_i is strictly shorter than π, a contradiction.

2.2 Mildly Context-Sensitive Languages

Recall that **context-sensitive languages** (aka **type-1 languages**) are defined by phrase structure grammars with rules of form $\lambda A \rho \to \lambda \alpha \rho$ with A in N, λ, ρ in V^*, and α in V^+. Their expressive power is equivalent to that of **linear bounded automata** (LBA), i.e. Turing machines working in linear space. Such grammars are not very useful from a computational viewpoint: membership is PSPACE-complete, and emptiness is undecidable.

Still, one would like to use string- and tree-generating formalisms with greater expressive power than context-free grammars. The rationale is twofold:

- some natural language constructs are not context-free, the Swiss-German account by Shieber (1985) being the best known example. Such fragments typically involve so-called **limited cross-serial dependencies**, as in the languages $\{a^n b^m c^n d^m \mid n, m \geq 0\}$ or $\{ww \mid w \in \{a, b\}^*\}$.
- the class of regular tree languages is not rich enough to account for the desired linguistic analyses (e.g. Kroch and Santorini, 1991, for Dutch).

This second argument is actually the strongest: the class of tree structures and how they are combined—which ideally should relate to how semantics compose—in context-free grammars are not satisfactory from a linguistic modeling point of view.

Based on his experience with **tree-adjoining grammars** (TAGs) and weakly equivalent formalisms (head grammars, a version of combinatory categorial grammars, and linear indexed grammars; see Joshi et al., 1991), Joshi (1985) proposed an *informal* definition of which properties a class of formal languages should have for linguistic applications: **mildly context-sensitive languages** (MCSLs) were "roughly" defined as the extensions of context-free languages that accommodate

1. *limited cross-serial dependencies*, while preserving
2. constant growth—a requisite nowadays replaced by **semilinearity**, which demands the Parikh image of the language to be a semilinear subset of $\mathbb{N}^{|\Sigma|}$ (Parikh, 1966), and
3. *polynomial time recognition*.

A possible *formal* definition for MCSLs is the class of languages generated by **multiple context-free grammars** (MCFGs, Seki et al., 1991), or equivalently **linear context-free rewrite systems** (LCFRSs, Weir, 1992), **multi-component tree adjoining grammars** (MCTAGs), and quite a few more.

We will however concentrate on two strict subclasses: tree adjoining languages (TALs, Section 2.2.1) and well-nested MCSLs (wnMCSLs, Section 2.2.2); Figure 2.4 illustrates the relationship between these classes. As in Section 2.1 our main focus will be on the corresponding tree languages, representing linguistic constituency analyses and sentence composition.

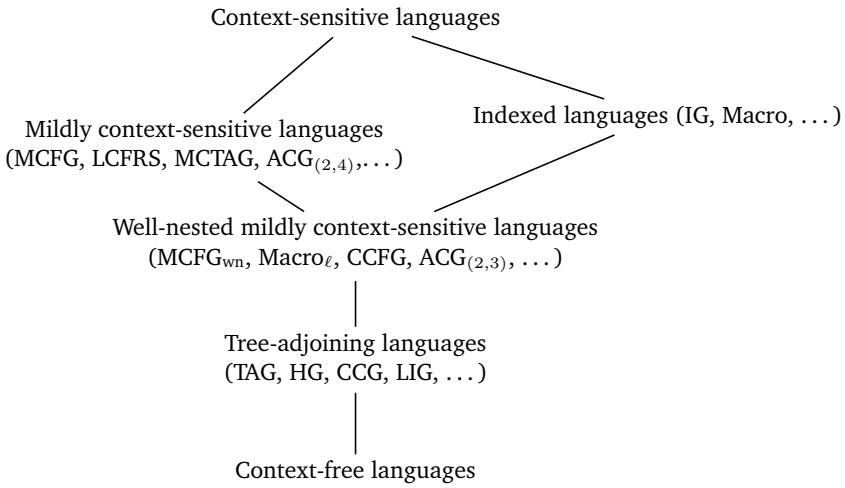


Figure 2.4: Hierarchies between context-free and full context-sensitive languages.

2.2.1 Tree Adjoining Grammars

Tree-adjoining grammars are a restricted class of term rewrite systems (we will see later that they are more precisely a subclass of the linear monadic context-free tree grammars). They have first been defined by Joshi et al. (1975) and subsequentely extended in various ways; see Joshi and Schabes (1997) for the "standard" definitions.

Definition 2.13 (Tree Adjoining Grammars)**.** A **tree adjoining grammar** (TAG) is a tuple $\mathcal{G} = \langle N, \Sigma, T_\alpha, T_\beta, S\rangle$ where N is a finite *nonterminal* ranked alphabet with arities > 0, Σ a finite *terminal* ranked alphabet with arities 0 and $N \cap \Sigma = \emptyset$, T_α and T_β two finite sets of finite **initial** and **auxiliary** trees, where $T_\alpha \cup T_\beta$ is called the set of **elementary** trees, and S in N a *start symbol*.

Given the nonterminal alphabet N, define

- $N{\downarrow} = \{A{\downarrow} \mid A \in N\}$ the set of **substitution** labels, all with arity 0,
- $N\star = \{A\star \mid A \in N\}$ the set of **foot** variables, all with arity 0, and
- $N^{\text{na}} = \{A^{\text{na}} \mid A \in N\}$ the set of **null adjunction** labels, with the same arities for A^{na} as for A.

Then

- $T_\alpha \subseteq T(N \cup N{\downarrow} \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\})$ is a finite set of finite trees α with nonterminal or null adjunction symbols as internal node labels, and terminal symbols or ε or substitution symbols as leaf labels;
- $T_\beta \subseteq T(N \cup N{\downarrow} \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\}, N\star)$ is a finite set of finite trees $\beta[A\star]$ defined similarly, except for the additional condition that they should have *exactly* one leaf, called the **foot node**, labeled by a variable $A\star$ of $N\star$ iff the root is labeled A.

The semantics of a TAG is that of a finite term rewrite system with rules

$$R_{\mathcal{G}} = \{A{\downarrow}^{(0)} \to \alpha \mid \alpha \in T_\alpha \text{ has } A \text{ for root label}\} \quad \text{(substitution)}$$
$$\cup\ \{A^{(m)}(x_1, \ldots, x_m) \to \beta[A^{\text{na}(m)}(x_1, \ldots, x_m)] \mid A^{(m)} \in N_m, \beta[A\star] \in T_\beta\}\ . \quad \text{(adjunction)}$$

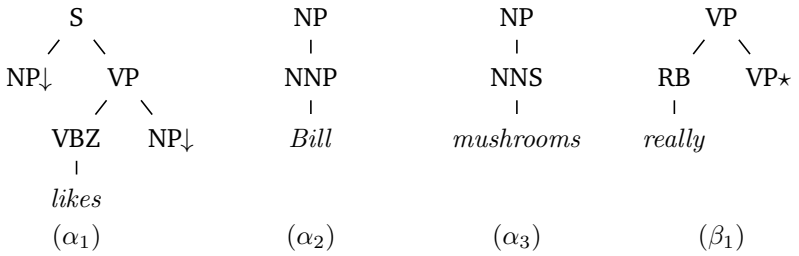


Figure 2.5: A tree adjoining grammar.

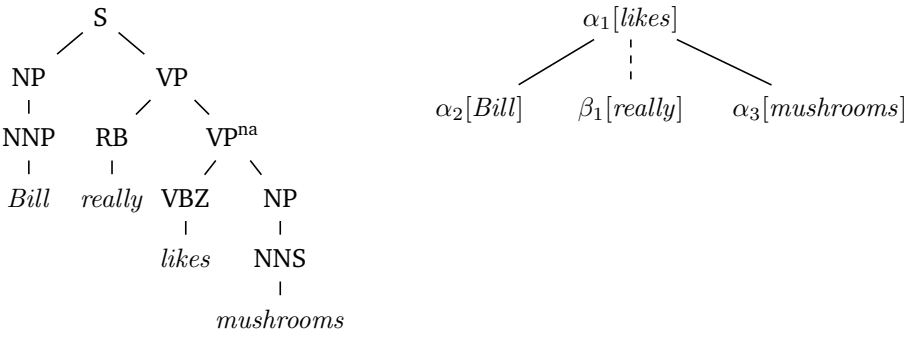


Figure 2.6: A derived tree and the corresponding derivation tree for the TAG of Example 2.14.

A **derivation** starts with an initial tree in T_α and applies rules from $R_\mathcal{G}$ until no substitution node is left:

$$L_T(\mathcal{G}) = \{t \in T(N \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\}) \mid \exists \alpha \in T_\alpha \text{ rooted by } S, \alpha \overset{R_\mathcal{G}}{\Longrightarrow}{}^* t\}$$

is the **tree language** of $\mathcal{G}$, while its **string language** is

$$L(\mathcal{G}) = \text{yield}(L_T(\mathcal{G}))$$

the set of yields of all its trees.

Example 2.14. Figure 2.5 presents a tree adjoining grammar with

$$\begin{aligned} N &= \{\text{S}, \text{NP}, \text{VP}, \text{VBZ}, \text{NNP}, \text{NNS}, \text{RB}\}\ , \\ \Sigma &= \{\textit{likes}, \textit{Bill}, \textit{mushrooms}, \textit{really}\}\ , \\ T_\alpha &= \{\alpha_1, \alpha_2, \alpha_3\}\ , \\ T_\beta &= \{\beta_1\}\ , \\ S &= \text{S}\ . \end{aligned}$$

Its sole S-rooted initial tree is α_1, on which one can substitute α_2 or α_3 in order to get *Bill likes mushrooms* or *mushrooms likes mushrooms*; the adjunction of β_1 on the VP node of α_1 also yields *Bill really likes mushrooms* (see Figure 2.6) or *mushrooms really really really likes Bill*. In the TAG literature, a tree in $T(N \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\})$ obtained through the substitution and adjunction operations is called a **derived tree**, while a **derivation tree** records how the rewrites took place (see Figure 2.6 for an example; children of an elementary tree are shown in addressing order, with plain lines for substitutions and dashed lines for adjunctions).

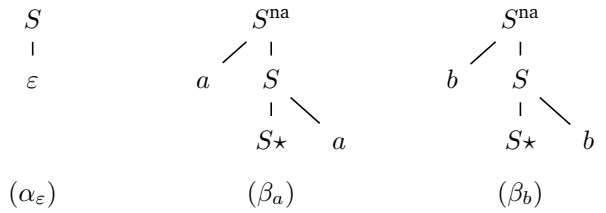


Figure 2.7: A TAG for L_{copy}.

Example 2.15 (Copy Language)**.** The **copy language** $L_{\text{copy}} = \{ww \mid w \in \{a, b\}^*\}$ is generated by the TAG of Figure 2.7 with $N = \{S\}$, $\Sigma = \{a, b\}$, $T_\alpha = \{\alpha_\varepsilon\}$, and $T_\beta = \{\beta_a, \beta_b\}$.

Exercise 2.10. Give a TAG for the language $\{a^n b^m c^n d^m \mid n, m \geq 0\}$. **(∗)**

Linguistic Analyses Using TAGs

Starting in particular with Kroch and Joshi (1985)'s work, the body of literature on linguistic analyses using TAGs and their variants is quite large. As significant evidence of the practical interest of TAGs, the XTAG project (XTAG Research Group, 2001) has published a large TAG for English, with a few more than 1,000 elementary unanchored trees. This particular variant of TAGs, a **lexicalized**, **feature-based** TAG, uses finite **feature structures** and **lexical anchors**. We will briefly survey the architecture of this grammar, and give a short account of it how treats some long-distance dependencies in English.

Lexicalized Grammar A TAG is **lexicalized** if all its elementary trees have at least one terminal symbol as a leaf. In linguistic modeling, it will actually have one distinguished terminal symbol, called the **anchor**, plus possibly some other terminal symbols, called **coanchors**. An anchor serves as head word for at least a part of the elementary tree, as *likes* for α_1 in Figure 2.5. Coanchors serve for particles, prepositions, etc., whose use is mandatory in the syntactic phenomenon modeled by the elementary tree, as *by* for α_5 in Figure 2.8.

Subcategorization Frames. Each elementary tree then instantiates a **subcategorization frame** for its anchor, i.e. specifications of the number and categories of the arguments of a word. For instance, *to like* is a **transitive verb** taking a NP subject and a NP complement, as instantiated by α_1 in Figure 2.5; similarly, *to think* takes a clausal S complement, as instantiated by β_2 in Figure 2.8. These first two examples are **canonical** instantiations of the subcategorization frames of *to like* and *to think*, but there are other possible instantiations, for instance **interrogative** with α_4 or **passive** with α_5 for *to like*.

*A more principled organization of the trees for subcategorization frames and their various instantiations can be obtained thanks to a **meta grammar** describing the set of elementary trees (see e.g. Crabbé, 2005).*

Example 2.16. Extend the TAG of Figure 2.5 with the trees of Figure 2.8. This new grammar is now able to generate

> mushrooms are liked by Bill
> mushrooms think Bill likes Bill
> who does Bill really think Bill really likes

In a feature-based grammar, both the obligatory adjunction of a single β_3 on the S node of α_4, and that of a single β_4 on the VP node of α_5 are controlled through the feature structures, and there is no overgeneration from this simple grammar.

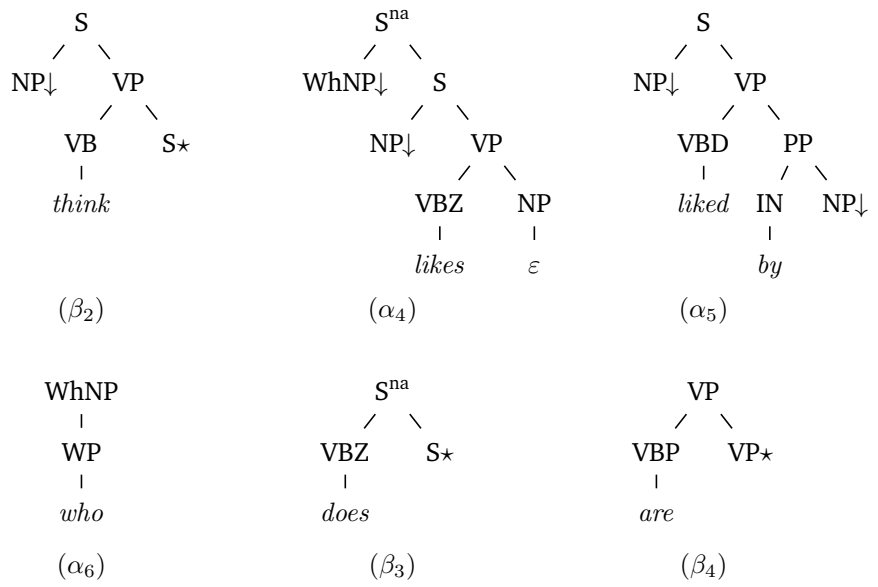


Figure 2.8: More elementary trees for the tree adjoining grammar of Example 2.14.

Syntactic Lexicon. In practice, elementary trees as the ones of Figure 2.5 are not present as such in the XTAG grammar. It rather contains **unanchored** versions of these trees, with a specific marker $\diamond$ for the anchor position. For instance, α_2 in Figure 2.5 would be stored as a context NP(NNP($\diamond$)) and enough information to know that *Bill* anchors this tree.

The anchoring information is stored in a **syntactic lexicon** associating with each lexical entry classes of trees that it anchors. The XTAG project has developed a naming ontology for these classes based on subcategorization frame and type of construction (e.g. canonical, passive, ...).

Long-Distance Dependencies Let us focus on α_4 in Figure 2.8. The "move" of the object NP argument of *likes* into sentence-first position as a WhNP is called a **long-distance dependency**. Observe that a CFG analysis would be difficult to come with, as this "move" crosses through the VP subtree of *think*—see the dotted dependency in the derived tree of Figure 2.9. We leave the question of syntax/semantics interfaces using derivation trees to the second half of the course.

See Schabes and Shieber (1994) for an alternative definition of adjunction, which yields more natural derivation trees. Among the possible interfaces to semantics, let us mention the use of feature structures (Gardent and Kallmeyer, 2003; Kallmeyer and Romero, 2004), or better a mapping from the derivation structures to logical ones (de Groote, 2001).

2.2.2 Well-Nested MCSLs

The class of **well-nested MCSLs** is at the junction of different extensions of context-free languages that still lie below full context-sensitive ones Figure 2.4. This provides characterizations both in terms of

- **well-nested multiple context-free grammars** (or equivalently well-nested linear context-free rewrite systems) (Kanazawa, 2009), and in terms of
- **linear macro grammars** (Seki and Kato, 2008), a subclass of the macro grammars of Fischer (1968), also characterized via linear context-free tree grammars (Rounds, 1970) or linear macro tree transducers (Engelfriet and Vogler, 1985).

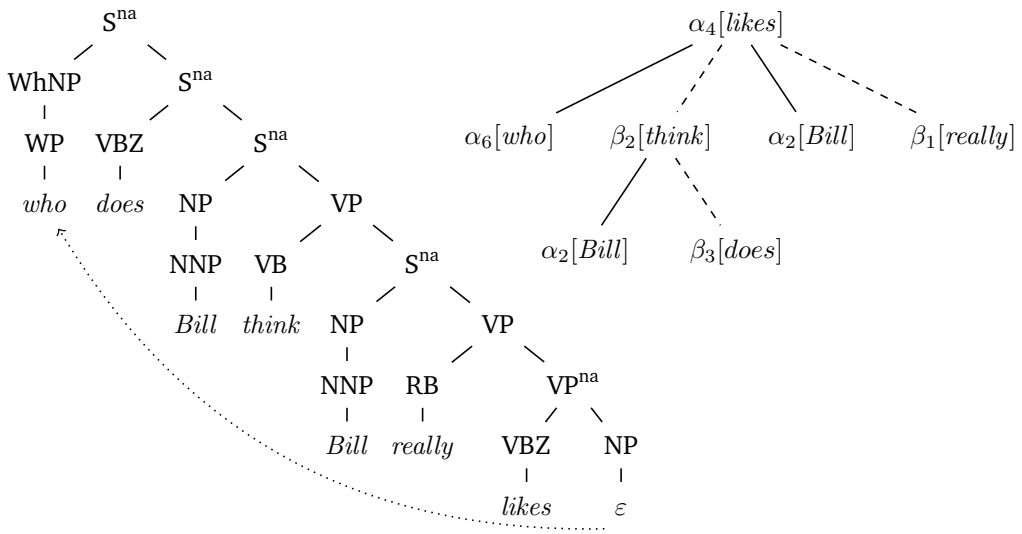


Figure 2.9: Derived and derivation trees for *Who does Bill think Bill really likes?* using the TAG of Figures 2.5 and 2.8.

We concentrate on the latter view, which opens some interesting perspectives towards **two-level syntax** (see e.g. Shieber, 2006; de Groote, 2001).

Definition 2.17 (Macro Tree Transducers)**.** A **macro tree transducer** (MTT) is a tuple $\mathcal{T} = \langle Q, \Sigma, \Delta, I, R\rangle$ consisting of three finite ranked alphabets Q, Σ, and Δ of *states*, *input*, and *output* symbols, a set $I \subseteq Q_1$ of initial states, all with arity 1, and a set R of rewrite rules over $T(Q \cup \Sigma \cup \Delta, \mathcal{X} \cup \mathcal{Y})$ for $\mathcal{X}, \mathcal{Y}$ two infinite countable sets of input variables and parameters, each rule being of form

$$q^{(n+1)}(a^{(m)}(x_1, \ldots, x_m), y_1, \ldots, y_n) \to t$$

where $q^{(n+1)}$ is in Q_{n+1}, $a^{(m)}$ in Σ_m, the input variables x_i in $\mathcal{X}_m$, and the parameters y_j in $\mathcal{Y}_n$, and t is a tree in $\mathrm{RHS}(Q, \Delta, m, n)$ defined by the abstract syntax

$$t ::= y_j \mid a^{(r)}(t, \ldots, t) \mid q^{(p+1)}(x_i, t, \ldots, t)$$

where y_j is in $\mathcal{Y}_n$, $a^{(r)}$ in Δ_r, $q^{(p+1)}$ in Q_{p+1}, and x_i in $\mathcal{X}_m$.

The semantics $[\![\mathcal{T}]\!]$ of a MTT is a relation in $T(\Sigma) \times T(\Delta)$ defined by

$$[\![\mathcal{T}]\!] = \{(t, t') \in T(\Sigma) \times T(\Delta) \mid \exists q_i^{(1)} \in I, q_i^{(1)}(t) \overset{R}{\Rightarrow}{}^{*}\ t'\}\ .$$

To do; this will be in next year's course...

Chapter 3

Categorial Grammars

The last approach to formal syntax we will consider in these notes is also one of the oldest: categorial grammars were indeed introduced by Bar-Hillel in 1953, based on earlier ideas of Ajdukiewicz (1935).

See the lecture notes of Retoré (2005) for a more detailed treatment of categorial grammars.

In their barest form, categorial grammars are defined using residuation types, which are usually called syntactic types or categories. Consider a finite set of **primitive types** Γ. We define **syntactic types** over C as terms γ defined by the abstract syntax

$$C ::= p \mid C \setminus C \mid C \,/\, C \qquad \text{(syntactic types)}$$

where p is in Γ; let $C(\Gamma)$ be the set of syntactic types over Γ. The second part of the course will better emphasize the interest of categorial grammars for semantics representation. Indeed, one can apply the Curry-Howard isomorphism and associate lambda terms (modeling semantics) to syntactic types (modeling syntax).

The **interpretation** of sequences of syntactic types over the free semigroup $\langle \Sigma^+, \cdot \rangle$ relies on a finite **lexical** relation ℓ between Σ and $C(\Gamma)$, mapping words to set of syntactic types, so that the interpretation of $[\![C]\!]_\ell$ a syntactic type C given ℓ is a subset of Σ^+:

$$\begin{aligned} [\![p]\!]_\ell &= \ell^{-1}(p) \\ [\![C_1 \setminus C_2]\!]_\ell &= ([\![C_1]\!]_\ell)^{-1} \cdot [\![C_2]\!]_\ell \\ [\![C_1 \,/\, C_2]\!]_\ell &= [\![C_1]\!]_\ell \cdot ([\![C_2]\!]_\ell)^{-1} \,. \end{aligned}$$

Having a distinguished axiom type S then allows to define a language over Σ as the interpretation $[\![S]\!]_\ell$. Categorial grammars are interested in quasi orderings $\vdash$ of **derivability** between sequences of types, such that if $\gamma \vdash \gamma'$ is derivable then $[\![\gamma]\!]_\ell \subseteq [\![\gamma']\!]_\ell$.

*The set of operators $\{\setminus, /\}$ itself can be expanded; for instance it is quite common to introduce an associative **product** $\bullet$ with interpretation $[\![C_1 \bullet C_2]\!]_\ell = [\![C_1]\!]_\ell \cdot [\![C_2]\!]_\ell$ (we will see it in the full Lambek calculus in Section 3.2). Thus we are considering a **product-free** fragment. (See also Morrill, 1994; Steedman, 2000; Moortgat, 1997, for further extended sets of operators.)*

Definition 3.1. A (product-free) **categorial grammar** $\mathcal{C} = \langle \Sigma, \Gamma, S, \vdash, \ell \rangle$ comprises a finite alphabet Σ, a finite set of primitive types Γ, a distinguished syntactic type S in $C(\Gamma)$, a derivability quasi ordering $\vdash$ over $(C(\Gamma))^+$, and a finite lexical relation ℓ in $\Sigma \times C(\Gamma)$.

The language of $\mathcal{C}$ is defined as

$$L(\mathcal{C}) = \{a_1 \cdots a_n \in \Sigma^+ \mid n > 0, \exists C_1 \in \ell(a_1), \ldots, \exists C_n \in \ell(a_n), C_1 \cdots C_n \vdash S\} \,.$$

We present two different systems to define derivability quasi orderings in sections 3.1 and 3.2.

3.1 AB Categorial Grammars

The derivability quasi ordering for AB categorial grammars (named after Ajdukiewicz and Bar-Hillel) can be defined by a **string rewrite system** R over the free semigroup $\langle C(\Gamma)^+, \cdot \rangle$ with the two **cancellation** rule schemata

$$B \cdot (B \setminus A) \to A \qquad (\setminus \mathsf{E})$$
$$(A \,/\, B) \cdot B \to A \qquad (/\mathsf{E})$$

for all A, B in $C(\Gamma)$, so that $\vdash$ is the reflexive transitive closure of the single-step rewrite relation $\overset{R}{\Rightarrow}$—which is by definition a quasi ordering.

Example 3.2. Let $\Gamma = \{n, s\}$ and let us consider the following lexical relation:

Σ	$C(\Gamma)$
Bill, *John*, *Mary*, *mushrooms*	n
the, *white*	$n \,/\, n$
works	$n \setminus s$
likes	$(n \setminus s) \,/\, n$
thinks	$(n \setminus s) \,/\, s$
tells	$((n \setminus s) \,/\, s) \,/\, n$
really	$(n \setminus s) \,/\, (n \setminus s)$
who	$(n \setminus n) \,/\, (n \setminus s)$

We can derive sentences such as

Bill really likes mushrooms.
John thinks Bill likes mushrooms.
Bill who likes mushrooms likes white mushrooms.
John tells Mary Bill likes mushrooms.

Observe that the principles of **lexicalization** put forward in Section 2.2.1 for TAGs are also at work here: the syntactic types associated to *works*, *likes*, *thinks*, and *tells* reflect their subcategorization frames (only a subject, a subject and a nominal object, a subject and a clausal object, and a subject and both a nominal and a clausal object resp.).

3.1.1 Alternative Views

Axiomatic View Other definitions are possible; for instance by algebraic laws over $(C(\Gamma))^+$, where the laws left implicit in the string rewrite definition have to be expressed. More precisely, by definition of a semigroup, the rules are taken modulo associativity of $\cdot$, and by definition of a string rewrite system, $\vdash$ is monotone wrt. concatenation:

$$A \vdash A \qquad \text{(reflexivity)}$$
$$A \vdash B \text{ and } B \vdash C \text{ imply } A \vdash C \qquad \text{(transitivity)}$$
$$A \cdot (B \cdot C) \vdash (A \cdot B) \cdot C \qquad \text{(associativity)}$$
$$(A \cdot B) \cdot C \vdash A \cdot (B \cdot C) \qquad \text{(associativity)}$$
$$A \vdash B \text{ implies } A \cdot C \vdash B \cdot C \qquad \text{(left monotonicity)}$$
$$A \vdash B \text{ implies } C \cdot A \vdash C \cdot B \qquad \text{(right monotonicity)}$$

for all A, B, C in $(C(\Gamma))^+$.

Proof-Theoretic View Yet another presentation would be as a natural deduction sequent calculus: a **sequent** $\gamma \vdash C$ pairs up a non empty sequence γ in $(C(\Gamma))^+$ with a syntactic type C in $C(\Gamma)$. The derivability quasi ordering is then defined by the following substructural proof system

$$\frac{}{C \vdash C}\ (\mathsf{Id}) \qquad \frac{\beta \vdash B \quad \alpha \vdash B \setminus A}{\beta\alpha \vdash A}\ (\setminus\mathsf{E}) \qquad \frac{\alpha \vdash A \,/\, B \quad \beta \vdash B}{\alpha\beta \vdash A}\ (/\mathsf{E})$$

where the two rules ($\setminus\mathsf{E}$) and ($/\mathsf{E}$) are non-commutative versions of the traditional modus ponens rule (which we will recall later as rule ($\to\mathsf{E}$)).

3.1.2 Equivalence with Context-Free Grammars

The equivalence of AB categorial grammars and context-free grammars is originally due to Bar-Hillel et al. (1960).

From AB Categorial Grammars to CFGs The encoding relies on a **subformula property** for the cancellation rules: the resulting types are always subtypes of the left-hand types. Thus, given an AB categorial grammar $\mathcal{C} = \langle \Sigma, \Gamma, S, \vdash, \ell \rangle$, the set of types that can appear during a derivation is in $\mathrm{sub}(\ell(\Sigma))^+$. A second property is a **context-freeness** one: a derivation of form $\beta \overset{R}{\Rightarrow}{}^n A_1 \cdots A_m$ can be decomposed into m subderivations $\beta_i \overset{R}{\Rightarrow}{}^{n_i} A_i$ with $n = n_1 + \cdots + n_m$ and $\beta = \beta_1 \cdots \beta_m$.

Using these two properties, it is straightforward to check that the CFG $\mathcal{G} = \langle \mathrm{sub}(\ell(\Sigma)), \Sigma, P, S \rangle$ with

$$\begin{aligned} P = &\{A \to B\ (B \setminus A) \mid (B \setminus A) \in \mathrm{sub}(\ell(\Sigma))\} \\ \cup\ &\{A \to (A \,/\, B)\ B \mid (A \,/\, B) \in \mathrm{sub}(\ell(\Sigma))\} \\ \cup\ &\{A \to a \mid (a, A) \in \ell\} \end{aligned}$$

encodes $\mathcal{C}$.

From CFGs to AB Categorial Grammars Recall that any CFG can be transformed into an equivalent CFG in (quadratic) **Greibach normal form** (GNF, Greibach, 1965), i.e. such that all its productions are of form

$$\begin{aligned} &S \to \varepsilon && S \text{ the axiom} \\ &A \to a\alpha && a \in \Sigma, \alpha \in (N \setminus \{S\})^{\leq 2} \end{aligned}$$

This yields a straightforward encoding of a CFG $\mathcal{G}$ with $\varepsilon \notin L(\mathcal{G})$ in GNF into an AB categorial grammar $\mathcal{C} = \langle N, \Sigma, \vdash, S, \ell \rangle$ with

$$\begin{aligned} \ell = &\{(a, (A \,/\, C) \,/\, B) \mid A \to aBC \in P\} \\ \cup\ &\{(a, A \,/\, B) \mid A \to aB \in P\} \\ \cup\ &\{(a, A) \mid A \to a \in P\}\ . \end{aligned}$$

Exercise 3.1. Fill out the missing details of the proof of equivalence between AB categorial grammars and context-free grammars. **(**)**

Exercise 3.2. The Dyck language D_n over n pairs of parentheses $a_i, \bar{a}_i$ is generated by the CFG with productions $\{S \to a_i S \bar{a}_i \mid 1 \leq i \leq n\} \cup \{S \to SS\} \cup \{S \to \varepsilon\}$. Give an AB categorial grammar for the language $D_n\$$ where $\$$ is an endmarker distinct from all the $a_i, \bar{a}_i$. **(**)**

3.1.3 Structural Limitations

There exist some structural limitations to AB categorial grammars. Consider for instance *that* introducing a subordination as in *the mushrooms that Bill likes*; in this frame the type for *that* would be $(n \setminus n) \,/\, (s \,/\, n)$ since *Bill likes* is intuitively of type $s \,/\, n$. However, we cannot derive $n \cdot ((n \setminus s) \,/\, n) \vdash s \,/\, n$ using only the cancellation rules, although it would be correct wrt. the free semigroup interpretation.

Several extensions were defined in order to circumvent the limitations of AB categorial grammars (often sparked by semantic rather than syntactic motivations):

type raising $B \to (A \,/\, B) \setminus A$ and $B \to A \,/\, (B \setminus A)$,

composition $(A \,/\, B)(B \,/\, C) \to A \,/\, C$ and $(C \setminus B)(B \setminus A) \to C \setminus A$,

Geach rules $A \,/\, B \to (A \,/\, C) \,/\, (B \,/\, C)$ and $B \setminus A \to (C \setminus B) \setminus (C \setminus A)$.

All these extensions are captured by the Lambek calculus.

3.2 Lambek Grammars

The **Lambek calculus** (Lambek, 1958) generalizes all the extensions of AB categorial rules by proposing instead to add introduction rules to the intuitionistic fragment of Section 3.1.1.

3.2.1 *Background:* Substructural Proof Systems

Let us first recall the implicative and conjunctive fragment of propositional calculus, with a presentation based on natural deduction for intuitionistic logic. A **proposition** C is defined in this fragment as

$$C ::= p \mid C \to C \mid C \wedge C \qquad \text{(propositions)}$$

where p is taken from a set Γ of atomic propositions. An **assumption** a sequence of propositions, and a **judgement** has the form $\gamma \vdash C$, meaning that from assumptions γ one can conclude proposition C. A version of the propositional calculus is then defined by the rules

$$\frac{}{C \vdash C}\ (\mathsf{Id})$$

$$\frac{\alpha\beta \vdash A}{\beta\alpha \vdash A}\ (\mathsf{Ex}) \qquad \frac{\alpha A A \vdash B}{\alpha A \vdash B}\ (\mathsf{Con}) \qquad \frac{\alpha \vdash B}{\alpha A \vdash B}\ (\mathsf{W})$$

$$\frac{\alpha B \vdash A}{\alpha \vdash B \to A}\ (\to\mathsf{I}) \qquad \frac{\beta \vdash B \quad \alpha \vdash B \to A}{\beta\alpha \vdash A}\ (\to\mathsf{E})$$

$$\frac{\alpha \vdash A \quad \beta \vdash B}{\alpha\beta \vdash A \wedge B}\ (\wedge\mathsf{I}) \qquad \frac{\alpha \vdash A \wedge B \quad \beta A B \vdash C}{\alpha\beta \vdash C}\ (\wedge\mathsf{E})$$

where (Ex), (Con), and (W) are the **structural rules** of *exchange*, *contraction*, and *weakening*, respectively.

There exists a rich literature on **substructural logics**, in particular **linear logic** (Girard, 1987) allows to restrict the use of the (Con) and (W) rules. If we completely forbid these two rules, then the fragment of propositional calculus we just saw corresponds to the multiplicative fragment of intuitionistic linear logic, which displays linear implication $\multimap$ instead of implication, and tensor product $\otimes$ instead of conjunction.

*See e.g. Troelstra (1992) for a textbook on linear logic, and the course **MPRI 2-1**.*

One can go a step further and also forbid the exchange rule (Ex). It has however the effect of refining the implication rules ($\to$I) and ($\to$E) into left and right implications, while conjunction becomes a form of concatenation, which we denote by $\bullet$:

One can go one more step further and define a non-associative *calculus, where sequents left parts are terms instead of sequences. This results in a variant called the **non-associative Lambek calculus** (Lambek, 1961).*

$$\frac{}{C \vdash C}\ (\mathsf{Id})$$

$$\frac{B\alpha \vdash A}{\alpha \vdash B \setminus A}\ (\setminus\mathsf{I}),\ \alpha \neq \varepsilon \qquad \frac{\beta \vdash B \quad \alpha \vdash B \setminus A}{\beta\alpha \vdash A}\ (\setminus\mathsf{E})$$

$$\frac{\alpha B \vdash A}{\alpha \vdash A \,/\, B}\ (/\mathsf{I}),\ \alpha \neq \varepsilon \qquad \frac{\alpha \vdash A \,/\, B \quad \beta \vdash B}{\alpha\beta \vdash A}\ (/\mathsf{E})$$

$$\frac{\alpha \vdash A \quad \beta \vdash B}{\alpha\beta \vdash A \bullet B}\ (\bullet\mathsf{I}) \qquad \frac{\beta \vdash A \bullet B \quad \alpha A B \gamma \vdash C}{\alpha\beta\gamma \vdash C}\ (\bullet\mathsf{E})$$

Note that this system simply adds insertion counterparts to ($\setminus$E) and (/E) and product rules to the system of Section 3.1.1. What we have just defined is a natural deduction version of the Lambek calculus.

Example 3.3. Here is a derivation of a type raising rule from Section 3.1.3:

$$\dfrac{\dfrac{\dfrac{}{A \,/\, B \vdash A \,/\, B}\ (\mathsf{Id}) \qquad \dfrac{}{B \vdash B}\ (\mathsf{Id})}{(A \,/\, B)B \vdash A}\ (/\mathsf{E})}{B \vdash (A \,/\, B) \setminus A}\ (\setminus\mathsf{I})$$

Exercise 3.3. Show that the composition and Geach rules from Section 3.1.3 are also derivable in this natural deduction version of the Lambek calculus. **(∗)**

3.2.2 Lambek Calculus

Lambek (1958) actually presents his calculus in Gentzen sequent style, with rules

$$\frac{}{C \vdash C}\ (\mathsf{Id}) \qquad \frac{\beta \vdash B \quad \alpha B \gamma \vdash A}{\alpha\beta\gamma \vdash A}\ (\mathsf{Cut})$$

$$\frac{B\alpha \vdash A}{\alpha \vdash B \setminus A}\ (\setminus\mathsf{R}),\ \alpha \neq \varepsilon \qquad \frac{\beta \vdash B \quad \alpha A \gamma \vdash C}{\alpha\beta(B \setminus A)\gamma \vdash C}\ (\setminus\mathsf{L})$$

$$\frac{\alpha B \vdash A}{\alpha \vdash A \,/\, B}\ (/\mathsf{R}),\ \alpha \neq \varepsilon \qquad \frac{\alpha A \gamma \vdash C \quad \beta \vdash B}{\alpha(A \,/\, B)\beta\gamma \vdash C}\ (/\mathsf{L})$$

$$\frac{\alpha \vdash A \quad \beta \vdash B}{\alpha\beta \vdash A \bullet B}\ (\bullet\mathsf{R}) \qquad \frac{\alpha A B \beta \vdash C}{\alpha(A \bullet B)\beta \vdash C}\ (\bullet\mathsf{L})$$

Again, one can recognize a non-commutative variation of the multiplicative fragment of intuitionistic linear logic.

Cut Elimination The Lambek calculus enjoys **cut elimination**, i.e. for any proof in the sequent calculus, there exists a proof that does not employ the (Cut) rule. A byproduct of cut elimination is that cut-free proofs have the **subformula property**, in the following strong sense: each application of the rules besides (Cut) adds one symbol from $\{\backslash, /, \bullet\}$ to the sequent. Thus working our way backward from a sequent $\gamma \vdash C$ to be proven, there are only finitely many cut-free proofs possible: the calculus is decidable.

The Lambek calculus is in fact NPTIME*-complete (Pentus, 2006).*

(∗) Exercise 3.4. Show that the decision procedure sketched above is in NPTIME.

Let us prove the cut elimination property. Suppose both $\beta \vdash B$ and $\alpha B \gamma \vdash A$ are provable in the cut-free calculus; we want to show that $\alpha\beta\gamma \vdash A$ is also provable. The proof proceeds by induction on the sum of the sizes of the sequents—defined as their number of symbols from $\{\backslash, /, \bullet\}$—and consists mostly of a large case analysis depending on the last rule employed to obtain the sequents before the cut:

1. if either sequent is the result of (Id), then the other is already the result of the cut,
2. if $\beta \vdash B$ is the result of a rule that did not introduce the main connective of B, i.e. rule ($\backslash$L), ($/$L), or ($\bullet$L), then there is a premise of form $\beta' \vdash B$ of smaller size, which by induction hypothesis yields $\alpha\beta'\gamma \vdash A$ in the cut-free calculus, and later $\alpha\beta\gamma \vdash A$ by the same rule application that lead from $\beta' \vdash B$ to $\beta \vdash B$,
3. if $\alpha B \gamma \vdash A$ is the result of a rule that did not introduce the main connective of B, then there is a premise of form $\alpha' B \gamma' \vdash A'$ of smaller size, which by induction hypothesis yields a cut-free proof of $\alpha'\beta\gamma' \vdash A'$, and an application of the rule that lead from $\alpha' B \gamma' \vdash A'$ to $\alpha B \gamma \vdash A$ yields the result,
4. if $B = C \bullet D$ is the result of ($\bullet$R) and ($\bullet$L), and we can replace

$$\dfrac{\dfrac{\beta' \vdash C \qquad \beta'' \vdash D}{\beta'\beta'' \vdash C \bullet D}\,(\bullet\mathsf{R}) \qquad \dfrac{\alpha C D \gamma \vdash A}{\alpha(C \bullet D)\gamma \vdash A}\,(\bullet\mathsf{L})}{\alpha\beta'\beta''\gamma \vdash A}\,(\mathsf{Cut})$$

   by the proof

$$\dfrac{\beta' \vdash C \qquad \dfrac{\beta'' \vdash D \qquad \alpha C D \gamma \vdash A}{\alpha C \beta'' \gamma \vdash A}\,(\mathsf{Cut})}{\alpha\beta'\beta''\gamma \vdash A}\,(\mathsf{Cut})$$

   with both (Cut) applications are on *smaller* sequents, thus provable in the cut-free calculus by induction hypothesis,
5. if $B = C \,/\, D$ is the result of ($/$R) and ($/$L), and we can replace

$$\dfrac{\dfrac{\beta' D \vdash C}{\beta' \vdash C \,/\, D}\,(/\mathsf{R}) \qquad \dfrac{\alpha C \gamma \vdash A \qquad \beta'' \vdash D}{\alpha(C \,/\, D)\beta''\gamma \vdash A}\,(/\mathsf{L})}{\alpha\beta'\beta''\gamma \vdash A}\,(\mathsf{Cut})$$

by the proof

$$\dfrac{\beta'' \vdash D \qquad \dfrac{\beta' D \vdash C \qquad \alpha C \gamma \vdash A}{\alpha \beta' D \gamma \vdash A}\,(\mathsf{Cut})}{\alpha \beta' \beta'' \gamma \vdash A}\,(\mathsf{Cut})$$

where both (Cut) applications are on *smaller* sequents, thus provable in the cut-free calculus by induction hypothesis,

6. if $B = C \setminus D$ is the result of ($\setminus$R) and ($\setminus$L), the case is symmetric to case 5.

Encoding Natural Deduction The natural deduction rules ($\setminus$E), and ($\bullet$E) can be obtained as (the case of ($/$E) being symmetric to that of ($\setminus$E)):

$$\dfrac{\alpha \vdash B \setminus A \qquad \dfrac{\beta \vdash B \qquad \dfrac{}{A \vdash A}\,(\mathsf{Id})}{\beta(B \setminus A) \vdash A}\,(\setminus\mathsf{L})}{\beta\alpha \vdash A}\,(\mathsf{Cut}) \qquad \dfrac{\beta \vdash A \bullet B \qquad \dfrac{\alpha A B \gamma \vdash C}{\alpha(A \bullet B)\gamma \vdash C}\,(\bullet\mathsf{L})}{\alpha\beta\gamma \vdash C}\,(\mathsf{Cut})$$

Conversely, one can prove that the Lambek calculus is actually equivalent to its natural deduction presentation (see e.g. Retoré, 2005, Section 2.6).

3.2.3 Equivalence with Context-Free Grammars

Although the Lambek calculus is strictly more expressive than the two cancellation rules ($\setminus$E) and ($/$E) of AB categorial grammars, **Lambek grammars**, i.e. the categorial grammars that employ the (product-free) Lambek calculus for the derivability quasi ordering $\vdash$, are not more expressive: they define exactly the context-free languages. This result was conjectured by Chomsky in the 1960s but remained open until the 1992 proof of Pentus.

We merely give a taste of the proof in the product-free case (Pentus, 1997). It defines the **norm** $\|\gamma\|$ of a product-free type sequence γ in $(C(\Gamma))^+$ as its number of atomic type occurrences, i.e.

$$\|p\| = 1 \quad \|C \setminus C'\| = \|C \,/\, C'\| = \|C\| + \|C'\| \quad \|C_1 \cdots C_n\| = \|C_1\| + \cdots + \|C_n\|,$$

and uses it to define the finite sets of types and type sequences

$$C_m(\Gamma) = \{C \in C(\Gamma) \mid \|C\| \leq m\} \qquad L_m(\Gamma) = \{\gamma \in (C(\Gamma))^+ \mid \|\gamma\| \leq 2m\}$$

for all $m \geq 0$. The **(m, Γ)-bounded Lambek calculus** is then defined by the two rules

$$\dfrac{}{\gamma \vdash C}\,(\mathsf{Ax}) \qquad \dfrac{\beta \vdash B \quad \alpha B \gamma \vdash A}{\alpha\beta\gamma \vdash A}\,(\mathsf{Cut})$$

where $\gamma \vdash C$ in (Ax) is any sequent in $L_m(\Gamma) \times C_m(\Gamma)$ provable in the product-free Lambek calculus (thus there are only finitely many such axioms for a fixed (m, Γ) pair).

Theorem 3.4 (Pentus, 1997)**.** *Let $B_1, \ldots, B_n, A$ be types in $C_m(\Gamma)$. If $B_1 \cdots B_n \vdash A$ is provable in the product-free Lambek calculus, then it is also provable in the (m, Γ)-bounded Lambek calculus.*

Thus, given a Lambek categorial grammar $\mathcal{C} = \langle \Sigma, \Gamma, S, \vdash, \ell \rangle$, there exist $m \geq 0$ and $\Gamma' \subseteq \Gamma$ s.t. $S \in C_m(\Gamma')$ and $\ell(\Sigma) \subseteq C_m(\Gamma')$. We can construct a context-free grammar $\mathcal{G} = \langle C_m(\Gamma'), \Sigma, P, S \rangle$ with

$$\begin{aligned} P = &\{C \to \gamma \mid C \in C_m(\Gamma'), \gamma \in L_m(\Gamma'), \gamma \vdash C \text{ provable}\} \\ &\cup \{A \to a \mid (a, A) \in \ell\} \ . \end{aligned}$$

Context-free derivations then simulate the action of the (Cut) rule in the (m, Γ')-calculus.

(**) Exercise 3.5. Prove using Theorem 3.4 the equivalence of $\mathcal{C}$ and $\mathcal{G}$ as defined above.

Chapter 4

References

Ajdukiewicz, K., 1935. Die syntaktische Konnexität. *Studia Philisophica*, 1:1–27. Cited on pages 49, 50.

Bar-Hillel, Y., 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29 (1):47–58. doi:10.2307/410452. Cited on pages 49, 50.

Bar-Hillel, Y., Gaifman, C., and Shamir, E., 1960. On categorial and phrase-structure grammars. *Bulletin of the research council of Israel*, 9F:1–16. Cited on page 51.

Bar-Hillel, Y., Perles, M., and Shamir, E., 1961. On formal properties of simple phrase-structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft, und Kommunikationsforschung*, 14:143–172. Cited on pages 30, 32.

Berstel, J., 1979. *Transductions and Context-Free Languages*. Teubner Studienbücher: Informatik. Teubner. ISBN 3-519-02340-7. http://www-igm.univ-mlv.fr/~berstel/LivreTransductions/LivreTransductions.html. Cited on pages 2, 9, 11.

Berstel, J. and Reutenauer, C., 2010. *Noncommutative Rational Series With Applications*. Cambridge University Press. http://www-igm.univ-mlv.fr/~berstel/LivreSeries/LivreSeries.html. Cited on page 9.

Billot, S. and Lang, B., 1989. The structure of shared forests in ambiguous parsing. In *ACL'89*, pages 143–151. ACL Press. doi:10.3115/981623.981641. Cited on page 30.

Black, E., Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T., 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *HLT '91*, pages 306–311. ACL Press. doi:10.3115/112405.112467. Cited on page 40.

Book, R. and Otto, F., 1993. *String Rewriting Systems*. Texts and monographs in Computer Science. Springer. ISBN 3-540-97965-4. Cited on page 3.

Booth, T.L. and Thompson, R.A., 1973. Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450. doi:10.1109/T-C.1973.223746. Cited on pages 36, 37.

Brants, T., 2000. TnT – a statistical part-of-speech tagger. In *ANLP 2000*, pages 224–231. doi:10.3115/974147.974178. Cited on page 16.

Brill, E., 1992. A simple rule-based part of speech tagger. In *ANLP '92*, pages 152–155. ACL Press. doi:10.3115/974499.974526. Cited on pages 16, 17, 18.

Chi, Z. and Geman, S., 1998. Estimation of probabilistic context-free grammars. *Computational Linguistics*, 24(2):299–305. http://www.aclweb.org/anthology/J98-2005.pdf. Cited on page 38.

Chomsky, N., 1956. Three models for the description of language. *IEEE Transactions on Information Theory*, 2(3):113–124. doi:10.1109/TIT.1956.1056813. Cited on page 29.

Chomsky, N., 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167. doi:10.1016/S0019-9958(59)90362-6. Cited on page 29.

Chomsky, N. and Halle, M., 1968. *The Sound Pattern of English*. Harper and Row. Cited on page 13.

Cocke, J. and Schwartz, J.T., 1970. *Programming languages and their compilers*. Courant Institute of Mathematical Sciences, New York University. Cited on page 30.

Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M., 2007. *Tree Automata Techniques and Applications*. http://tata.gforge.inria.fr/. Cited on pages 2, 4, 31.

Coppersmith, D. and Winograd, S., 1990. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280. doi:10.1016/S0747-7171(08)80013-2. Cited on page 30.

Cousot, P. and Cousot, R., 2003. Parsing as abstract interpretation of grammar semantics. *Theoretical Computer Science*, 290(1):531–544. doi:10.1016/S0304-3975(02)00034-8. Cited on page 34.

Crabbé, B., 2005. Grammatical development with XMG. In Blache, P., Stabler, E., Busquets, J., and Moot, R., editors, *LACL'05*, volume 3492 of *Lecture Notes in Computer Science*, pages 84–100. Springer. ISBN 978-3-540-25783-7. doi:10.1007/11422532_6. Cited on page 45.

Crochemore, M. and Hancart, C., 1997. Automata for matching patterns. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 2. Linear Modeling: Background and Application, chapter 9, pages 399–462. Springer. ISBN 3-540-60648-3. Cited on page 18.

de Groote, P., 2001. Towards abstract categorial grammars. In *ACL 2001*, pages 252–259. ACL Press. doi:10.3115/1073012.1073045. Cited on pages 46, 47.

Earley, J., 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102. doi:10.1145/362007.362035. Cited on pages 30, 34.

Engelfriet, J. and Vogler, H., 1985. Macro tree transducers. *Journal of Computer and System Sciences*, 31:71–146. doi:10.1016/0022-0000(85)90066-2. Cited on page 46.

Etessami, K. and Yannakakis, M., 2009. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66. doi:10.1145/1462153.1462154. Cited on page 37.

Fischer, M.J., 1968. Grammars with macro-like productions. In *SWAT '68*, pages 131–142. IEEE Computer Society. doi:10.1109/SWAT.1968.12. Cited on page 46.

Gardent, C. and Kallmeyer, L., 2003. Semantic construction in feature-based TAG. In *EACL'03*, pages 123–130. ACL Press. ISBN 1-333-56789-0. doi:10.3115/1067807.1067825. Cited on page 46.

Gecse, R. and Kovács, A., 2010. Consistency of stochastic context-free grammars. *Mathematical and Computer Modelling*, 52(3–4):490–500. doi:10.1016/j.mcm.2010.03.046. Cited on page 37.

Girard, J.Y., 1987. Linear logic. *Theoretical Computer Science*, 50(1):1–101. doi:10.1016/0304-3975(87)90045-4. Cited on page 53.

Graham, S.L., Harrison, M., and Ruzzo, W.L., 1980. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems*, 2(3):415–462. doi:10.1145/357103.357112. Cited on page 30.

Greibach, S.A., 1965. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52. doi:10.1145/321250.321254. Cited on page 51.

Grune, D. and Jacobs, C.J.H., 2007. *Parsing Techniques*. Monographs in Computer Science. Springer, second edition. ISBN 0-387-20248-X. Cited on page 30.

Harrison, M.A., 1978. *Introduction to Formal Language Theory*. Series in Computer Science. Addison-Wesley. ISBN 0-201-02955-3. Cited on page 2.

Jones, N.D. and Laaser, W.T., 1976. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3(1):105–117. doi:10.1016/0304-3975(76)90068-2. Cited on page 30.

Joshi, A.K., Levy, L.S., and Takahashi, M., 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163. doi:10.1016/S0022-0000(75)80019-5. Cited on page 43.

Joshi, A.K., 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In Dowty, D.R., Karttunen, L., and Zwicky, A.M., editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, chapter 6, pages 206–250. Cambridge University Press. Cited on page 42.

Joshi, A.K., Vijay-Shanker, K., and Weir, D., 1991. The convergence of mildly context-sensitive grammatical formalisms. In Sells, P., Shieber, S., and Wasow, T., editors, *Foundational Issues in Natural Language Processing*. MIT Press. http://repository.upenn.edu/cis_reports/539. Cited on page 42.

Joshi, A.K. and Schabes, Y., 1997. Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages*, volume 3: Beyond Words, chapter 2, pages 69–124. Springer. ISBN 3-540-60649-1. http://citeseer.ist.psu.edu/joshi97treeadjoining.html. Cited on page 43.

Jurafsky, D. and Martin, J.H., 2009. *Speech and Language Processing*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, second edition. ISBN 978-0-13-187321-6. Cited on pages 2, 6, 24.

Kallmeyer, L. and Romero, M., 2004. LTAG semantics with semantic unification. In Rambow, O. and Stone, M., editors, *TAG+7*, pages 155–162. http://www.cs.rutgers.edu/TAG+7/papers/kallmeyer-c.pdf. Cited on page 46.

Kanazawa, M., 2009. The pumping lemma for well-nested multiple context-free languages. In Diekert, V. and Nowotka, D., editors, *DLT 2009*, volume 5583 of *Lecture Notes in Computer Science*, pages 312–325. Springer. doi:10.1007/978-3-642-02737-6_25. Cited on page 46.

Kaplan, R.M. and Kay, M., 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378. http://www.aclweb.org/anthology/J94-3001.pdf. Cited on page 15.

Karttunen, L., 1983. KIMMO: a general morphological processor. In Dalrymple, M., Doron, E., Goggin, J., Goodman, B., and McCarthy, J., editors, *Texas Linguistic Forum*, volume 22, pages 165–186. Department of Linguistics, The University of Texas at Austin. http://www2.parc.com/istl/members/karttune/publications/archive/kimmo/kimmo-gmp.pdf. Cited on page 12.

Karttunen, L., Chanod, J.P., Grefenstette, G., and Schiller, A., 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2:305–328. doi:10.1017/S1351324997001563. Cited on page 6.

Kasami, T., 1965. An efficient recognition and syntax analysis algorithm for context free languages. Scientific Report AF CRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachussetts. Cited on page 30.

Knuth, D.E., 1965. On the translation of languages from left to right. *Information and Control*, 8(6):607–639. doi:10.1016/S0019-9958(65)90426-2. Cited on page 30.

Knuth, D.E., 1977. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5. doi:10.1016/0020-0190(77)90002-3. Cited on pages 40, 41.

Koskenniemi, K. and Church, K.W., 1988. Complexity, two-level morphology and Finnish. In *CoLing '88*, pages 335–340. ACL Press. doi:10.3115/991635.991704. Cited on page 13.

Kroch, A.S. and Joshi, A.K., 1985. The linguistic relevance of tree adjoining grammars. Technical Report MS-CIS-85-16, University of Pennsylvania, Department of Computer and Information Science. http://repository.upenn.edu/cis_reports/671/. Cited on page 45.

Kroch, A.S. and Santorini, B., 1991. The derived constituent structure of the West Germanic verb-raising construction. In Freidin, R., editor, *Principles and Parameters in Comparative Grammar*, chapter 10, pages 269–338. MIT Press. Cited on page 42.

Kurki-Suonio, R., 1969. Notes on top-down languages. *BIT Numerical Mathematics*, 9 (3):225–238. doi:10.1007/BF01946814. Cited on page 30.

Lambek, J., 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170. doi:10.2307/2310058. Cited on pages 52, 53.

Lambek, J., 1961. On the calculus of syntactic types. In Jakobson, R., editor, *Structure of Language and its Mathematical Aspects*, volume 12 of *Proceedings of Symposia in Applied Mathematics*, pages 166–178. AMS. ISBN 0-8218-1312-9. Cited on page 53.

Lang, B., 1974. Deterministic techniques for efficient non-deterministic parsers. In Loeckx, J., editor, *ICALP'74*, volume 14 of *Lecture Notes in Computer Science*, pages 255–269. Springer. doi:10.1007/3-540-06841-4_65. Cited on page 30.

Lang, B., 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10 (4):486–494. doi:10.1111/j.1467-8640.1994.tb00011.x. Cited on page 32.

Lee, L., 2002. Fast context-free grammar parsing requires fast boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15. doi:10.1145/505241.505242. Cited on page 30.

Leo, J.M.I.M., 1991. A general context-free parsing algorithm running in linear time on every LR(k) grammar without using lookahead. *Theoretical Computer Science*, 82(1): 165–176. doi:10.1016/0304-3975(91)90180-A. Cited on page 35.

Lombardy, S. and Sakarovitch, J., 2006. Sequential? *Theoretical Computer Science*, 356 (1):224–244. doi:10.1016/j.tcs.2006.01.028. Cited on page 28.

Manning, C.D. and Schütze, H., 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. ISBN 978-0-262-13360-9. Cited on pages 2, 24.

Marcus, M.P., Marcinkiewicz, M.A., and Santorini, B., 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330. http://www.aclweb.org/anthology/J93-2004.pdf. Cited on pages 7, 8, 16, 38.

Matiyasevicha, Y. and Sénizergues, G., 2005. Decision problems for semi-Thue systems with a few rules. *Theoretical Computer Science*, 330(1):145–169. doi:10.1016/j.tcs.2004.09.016. Cited on page 14.

McCarthy, J.J., 1982. Prosodic structure and expletive infixation. *Language*, 58(3):574–590. doi:10.2307/413849. Cited on page 6.

McNaughton, R., 1995. Well behaved derivations in one-rule semi-Thue systems. Technical Report 95-15, Department of Computer Science, Rensselaer Polytechnic Institute. http://www.cs.rpi.edu/research/ps/95-15.ps. Cited on page 14.

Mohri, M. and Sproat, R., 1996. An efficient compiler for weighted rewrite rules. In *ACL '96*, pages 231–238. ACL Press. doi:10.3115/981863.981894. Cited on page 15.

Mohri, M., 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311. http://www.cs.nyu.edu/~mohri/pub/cl1.pdf. Corrected version from the author's webpage. Cited on page 28.

Moore, R.C., 2004. Improved left-corner chart parsing for large context-free grammars. In *New Developments in Parsing Technology*, pages 185–201. Springer. doi: 10.1007/1-4020-2295-6_9. Cited on page 31.

Moortgat, M., 1997. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3–4):349–385. doi:10.1007/BF00159344. Cited on page 49.

Morrill, G.V., 1994. *Type Logical Grammar*. Kluwer Academic Publishers. ISBN 0-7923-3095-1. Cited on page 49.

Nederhof, M.J. and Satta, G., 2003. Probabilistic parsing as intersection. In *IWPT 2003*, pages 137–148. http://www.cs.st-andrews.ac.uk/∼mjn/publications/2003b.pdf. Cited on page 38.

Nederhof, M.J. and Satta, G., 2004. Tabular parsing. In Martín-Vide, C., Mitrana, V., and Paun, G., editors, *Formal Languages and Applications*, volume 148 of *Studies in Fuzziness and Soft Computing*, pages 529–549. Springer. arXiv:cs.CL/0404009. Cited on page 32.

Nederhof, M.J. and Satta, G., 2008. Probabilistic parsing. In Bel-Enguix, G., Jiménez-López, M., and Martín-Vide, C., editors, *New Developments in Formal Languages and Applications*, volume 113 of *Studies in Computational Intelligence*, pages 229–258. Springer. doi:10.1007/978-3-540-78291-9_7. Cited on page 35.

Parikh, R.J., 1966. On context-free languages. *Journal of the ACM*, 13(4):570–581. doi:10.1145/321356.321364. Cited on page 42.

Pentus, M., 1997. Product-free Lambek calculus and context-free grammars. *Journal of Symbolic Logic*, 62(2):648–660. doi:10.2307/2275553. Cited on page 55.

Pentus, M., 2006. Lambek calculus is NP-complete. *Theoretical Computer Science*, 357: 186–201. doi:10.1016/j.tcs.2006.03.018. Cited on page 54.

Pereira, F.C.N. and Warren, D.H.D., 1983. Parsing as deduction. In *ACL '83*, pages 137–144. ACL Press. doi:10.3115/981311.981338. Cited on page 34.

Pesetsky, D., 1985. Morphology and logical form. *Linguistic Inquiry*, 16(2):193–246. http://www.jstor.org/stable/4178430. Cited on page 12.

Pullum, G.K. and Scholz, B.C., 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In de Groote, P., Morrill, G., and Retoré, C., editors, *LACL 2001*, volume 2099 of *Lecture Notes in Computer Science*, pages 17–43. Springer. doi:10.1007/3-540-48199-0_2. Cited on page 29.

Raney, G.N., 1958. Sequential functions. *Journal of the ACM*, 5(2):177–180. doi: 10.1145/320924.320930. Cited on page 10.

Retoré, C., 2005. The logic of categorial grammars: Lecture notes. Technical Report RR-5703, INRIA. http://hal.inria.fr/inria-00070313/. Cited on pages 49, 55.

Roche, E. and Schabes, Y., 1995. Deterministic part-of-speech tagging with finite-state transducers. *Computational Linguistics*, 21(2):227–253. http://www.aclweb.org/anthology/J95-2004.pdf. Cited on pages 16, 17, 18.

Rosenkrantz, D.J. and Stearns, R.E., 1970. Properties of deterministic top-down grammars. *Information and Control*, 17(3):226–256. doi:10.1016/S0019-9958(70)90446-8. Cited on page 30.

Rounds, W.C., 1970. Mappings and grammars on trees. *Theory of Computing Systems*, 4 (3):257–287. doi:10.1007/BF01695769. Cited on page 46.

Sakarovitch, J., 2009. *Elements of Automata Theory*. Cambridge University Press. ISBN 978-0-521-84425-3. Translated from *Éléments de théorie des automates*, Vuibert, 2003. Cited on pages 2, 9, 10, 11.

Santorini, B., 1990. Part-of-speech tagging guidelines for the Penn Treebank project (3rd revision). Technical Report MS-CIS-90-47, University of Pennsylvania, Department of Computer and Information Science. http://repository.upenn.edu/cis_reports/570/. Cited on page 7.

Schabes, Y. and Shieber, S.M., 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124. http://www.aclweb.org/anthology/J94-1004. Cited on page 46.

Schützenberger, M.P., 1961. On the definition of a family of automata. *Information and Control*, 4(2–3):245–270. doi:10.1016/S0019-9958(61)80020-X. Cited on page 11.

Schützenberger, M.P., 1977. Sur une variante des fonctions séquentielles. *Theoretical Computer Science*, 4(1):47–57. doi:0.1016/0304-3975(77)90055-X. Cited on page 10.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T., 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229. doi:10.1016/0304-3975(91)90374-B. Cited on page 42.

Seki, H. and Kato, Y., 2008. On the generative power of multiple context-free grammars and macro grammars. *IEICE Transactions on Information and Systems*, E91-D(2):209–221. doi:10.1093/ietisy/e91-d.2.209. Cited on page 46.

Sénizergues, G., 1996. On the termination problem for one-rule semi-Thue system. In Ganzinger, H., editor, *RTA '96*, volume 1103 of *Lecture Notes in Computer Science*, pages 302–316. Springer. doi:10.1007/3-540-61464-8_61. Cited on page 14.

Shieber, S.M., 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8(3):333–343. doi:10.1007/BF00630917. Cited on page 42.

Shieber, S.M., 2006. Unifying synchronous tree-adjoining grammars and tree transducers via bimorphisms. In *EACL'06*. ACL Press. ISBN 1-932432-59-0. http://www.aclweb.org/anthology/E06-1048. Cited on page 47.

Sikkel, K., 1997. *Parsing Schemata - a framework for specification and analysis of parsing algorithms*. Texts in Theoretical Computer Science - An EATCS Series. Springer. ISBN 3-540-61650-0. Cited on page 34.

Simon, I., 1994. String matching algorithms and automata. In Karhumäki, J., Maurer, H., and Rozenberg, G., editors, *Results and Trends in Theoretical Computer Science: Colloquium in Honor of Arto Salomaa*, volume 812 of *Lecture Notes in Computer Science*, pages 386–395. Springer. ISBN 978-3-540-58131-4. doi:10.1007/3-540-58131-6_61. Cited on page 18.

Sproat, R.W., 1992. *Morphology and Computation*. ACL–MIT Press series in natural-language processing. MIT Press. ISBN 0-262-19314-0. Cited on page 12.

Steedman, M., 2000. *The Syntactic Process*. MIT Press. ISBN 0-262-69268-6. Cited on page 49.

Sudborough, I.H., 1978. On the tape complexity of deterministic context-free languages. *Journal of the ACM*, 25(3):405–414. doi:10.1145/322077.322083. Cited on page 30.

Thatcher, J.W., 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1 (4):317–322. doi:10.1016/S0022-0000(67)80022-9. Cited on page 30.

Tomita, M., 1986. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers. ISBN 0-89838-202-5. Cited on page 30.

Troelstra, A.S., 1992. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI. http://standish.stanford.edu/bin/detail?fileID=1846861073. Cited on page 53.

Valiant, L.G., 1975. General context-free recognition in less than cubic time. *Journal of Computer and System Sciences*, 10(2):308–314. doi:10.1016/S0022-0000(75)80046-8. Cited on page 30.

Weir, D.J., 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *ACL '92*, pages 136–143. ACL Press. doi:10.3115/981967.981985. Cited on page 42.

Wich, K., 2005. *Ambiguity Functions of Context-Free Grammars and Languages*. PhD thesis, Institut fur Formale Methoden der Informatik, Universität Stuttgart. ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/DIS-2005-01/DIS-2005-01.pdf. Cited on page 31.

XTAG Research Group, 2001. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, University of Pennsylvania, Institute for Research in Cognitive Science. http://www.cis.upenn.edu/~xtag/. Cited on page 45.

Younger, D.H., 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208. doi:10.1016/S0019-9958(67)80007-X. Cited on page 30.

Zimmer, K., 1964. *Affixal negation in English and other languages*. William Clowes. Supplement to *Word* 20:2, monograph 5. Cited on page 12.

Zwicky, A.M., 1985. Clitics and particles. *Language*, 61(2):283–305. doi:10.2307/414146. Cited on page 6.

Zwicky, A.M. and Pullum, G.K., 1987. Plain morphology and expressive morphology. In Aske, J., Beery, N., Michaelis, L., and Filip, H., editors, *Berkeley Linguistics Society '87*, pages 330–340. http://www.ling.ed.ac.uk/~gpullum/bls_1987.pdf. Cited on page 7.