MPRI course 2-27-1, year 2014–2015

Logical and Computational Structures for Linguistic Modelling

Sylvain Schmitz
LSV, ENS Cachan & CNRS & INRIA
April 8, 2015 (`r5867M`)

These notes cover the contents of an introductory course on computational linguistics, also known as **MPRI 2-27-1**: *Logical and computational structures for linguistic modelling*. The course is subdivided into two parts: the first, taught this year by Éric Villemonte de la Clergerie, covers grammars and automata for syntax modelling, while the second part focuses on logical approaches to syntax and semantics. Among the prerequisites to the course 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 is also advisable.

These notes are based on numerous articles—and I have tried my best to provide stable hyperlinks to online versions in the references—, and on the excellent material of Benoît Crabbé, Éric Villemonte de la Clergerie, and Philippe de Groote who taught this course with me.

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

MPRI 1-18: *Tree automata and applications*: tree languages and term rewriting systems will be our basic tools in many models;

MPRI 2-16: *Finite automata modelisation*: only the basic theory of weighted automata is used in our course;

MPRI 2-26-1: *Web data management*: you might be surprised at how many concepts are similar, from automata and logics on trees for syntax to description logics for semantics.

MPRI 2-1: *Linear logic*.

Contents

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. 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 $R_1 \mathbin{;} R_2 = \{(a, c) \mid \exists b \in B, (a, b) \in R_1 \wedge (b, c) \in R_2\}$. The **reflexive transitive closure** of a relation is noted $R^\star = \bigcup_i R^i$, where $R^0 = \mathrm{Id}_A = \{(a, a) \mid a \in A\}$ is the **identity** over A, and $R^{i+1} = R \mathbin{;} R^i$.

See Comon et al. *(2007) for missing definitions and notations.*

Terms. A **ranked alphabet** a pair (Σ, r) where Σ is a finite alphabet and $r : \Sigma \to \mathbb{N}$ gives the **arity** of symbols in Σ. The subset of symbols of arity n is noted Σ_n.

Let $\mathcal{X}$ be a set of **variables**, each with arity 0, assumed distinct from Σ. We write $\mathcal{X}_n$ for a set of n distinct variables taken from $\mathcal{X}$.

The set $T(\Sigma, \mathcal{X})$ of **terms** over Σ and $\mathcal{X}$ is the smallest set s.t. $\Sigma_0 \subseteq T(\Sigma, \mathcal{X})$, $\mathcal{X} \subseteq T(\Sigma, \mathcal{X})$, and if $n > 0$, f is in Σ_n, and $t_1, \dots, t_n$ are terms in $T(\Sigma, \mathcal{X})$, then $f(t_1, \dots, t_n)$ is a term in $T(\Sigma, \mathcal{X})$. The set of terms $T(\Sigma, \emptyset)$ is also noted $T(\Sigma)$ and is called the set of **ground terms**.

A term t in $T(\Sigma, \mathcal{X})$ is **linear** if every variable of $\mathcal{X}$ occurs at most once in t. A linear term in $T(\Sigma, \mathcal{X}_n)$ is called a **context**, and the expression $C[t_1, \dots, t_n]$ for $t_1, \dots, t_n$ in $T(\Sigma)$ denotes the term in $T(\Sigma)$ obtained by substituting t_i for x_i for each $1 \leq i \leq n$, i.e. is a shorthand for $C\{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n\}$. We denote $\mathcal{C}^n(\Sigma)$ the set of contexts with n variables, and $\mathcal{C}(\Sigma)$ that of contexts with a single variable—in which case we usually write $\Box$ for this unique variable.

Trees. By **tree** we mean a finite ordered ranked tree t over some set of labels Σ, i.e. a partial function $t : \{0, \ldots, k\}^* \to \Sigma$ where k is the maximal rank, associating to a finite sequence its label. The domain of t is **prefix-closed**, i.e. if $ui \in \mathrm{dom}(t)$ for u in $\mathbb{N}^*$ and i in $\mathbb{N}$, then $u \in \mathrm{dom}(t)$, and **predecessor-closed**, i.e. if $ui \in \mathrm{dom}(t)$ for u in $\mathbb{N}^*$ and i in $\mathbb{N}_{>0}$, then $u(i-1) \in \mathrm{dom}(t)$.

The set Σ can be turned into a ranked alphabet simply by building $k+1$ copies of it, one for each possible rank in $\{0, \ldots, k\}$; we note $a^{(m)}$ for the copy of a label a in Σ with rank m. Because in linguistic applications tree node labels typically denote syntactic categories, which have no fixed arities, it is useful to work under the convention that a denotes the “unranked” version of $a^{(m)}$. This also allows us to view trees as terms (over the ranked version of the alphabet), and conversely terms as trees (by erasing ranking information from labels)—we will not distinguish between the two concepts.

Term Rewriting Systems. A **term rewriting system** over some ranked alphabet Σ is a set of rules $R \subseteq (T(\Sigma, \mathcal{X}))^2$, each noted $t \to t'$. Given a rule $r : t \to t'$ (also noted $t \xrightarrow{r} t'$), with t, t' in $T(\Sigma, \mathcal{X}_n)$, the associated one-step rewrite relation over $T(\Sigma)$ is $\overset{r}{\Rightarrow} = \{(C[t\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}], C[t'\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}]) \mid C \in \mathcal{C}(\Sigma), t_1, \ldots, t_n \in T(\Sigma)\}$. We write $\xRightarrow{r_1 r_2}$ for $\xRightarrow{r_1} \mathbin{;} \xRightarrow{r_2}$, and $\overset{R}{\Rightarrow}$ for $\bigcup_{r \in R} \overset{r}{\Rightarrow}$.

Chapter 1

Introduction

If linguistics is about the description and understanding of human language, a computational linguist thrives in developing *computational* models of language. By computational, we mean models that are not only mathematically elegant, but also amenable to an algorithmic treatment. Such models are certainly useful for practical applications in *natural language processing*, which range from text mining, question answering, and text summarization, to automated translation.

In spite of the large impact such technologies have on our lives, the case of computational linguistics is even stronger. Consider that human brains have limited capacity for holding language information (think for instance of dictionaries and common turns of phrase), and that being able to learn, understand, and produce a potentially unbounded number of utterances, we need to rely on some form or other of computation—quite an efficient one at that if you think about it.

A computational model, rather than a “mere” mathematical one, also allows for *experimentation*, and thus validation or refinement of the model. For example, a theoretical linguist might test her predictions about which sentences are grammatical by parsing large corpora of presumably correct text—does the model undergenerate?—, or about the syntax rules of a particular phenomenon by generating random sentences and checking against over-generation. As another example, a psycholinguist might try to match some measured degree of linguistic difficulty of sentences with various aspects of the model: frequency of the lexemes and of the syntactic rules, type and size of the involved rules, degree of ambiguity, etc.

1.1 Levels of Description

Language models are classically divided into several layers, first some specific to speech processing: **phonetics** and **phonology**, then more generally applicable: **morphology**, **syntax**, **semantics**, and **pragmatics**. This forms a *pipeline*, that inputs utterances in oral or written form and outputs meaning representations in context.

1.1.1 From Text to Meaning

Let us give a quick overview of the phases from text to meaning.

Morphology 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*. 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? Modeling morphology often involves (probabilistic) word automata and transducers.

This process is quite prone to ambiguity: in the sentence

> Gator attacks puzzle experts

is *attacks* a verb in third person singular (VBZ) or a plural noun (NNS)? Is *puzzle* a verb (VB) or a noun (NN)? Should crossword experts avoid Florida?

Syntax deals with the structure of sentences: how do we combine words into phrases and sentences?

Constituents and Dependencies. Two main types of analysis are used by syntacticians: one as **constituents**, where the sentence is split into phrases, themselves further split until we reach the word level, as in

> [[She] [watches [a bird]]]

Such a constituent analysis can also be represented as a tree, as on the left of Figure 1.1. Here we introduced part-of-speech tags and syntactic categories to label the internal nodes: for instance, *VBZ* stands for a verb conjugated in present third person, *NP* stands for a noun phrase, and *VP* for a verb phrase.

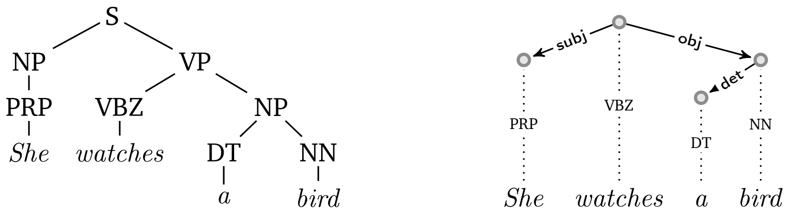


Figure 1.1: Constituent (on the left) and dependency (on the right) analyses.

An alternative analysis, illustrated on the right of Figure 1.1, rather exhibits the **dependencies** between words in the sentence: its **head** is the verb *watches*, with two dependents *She* and *bird*, which play the roles of subject and object respectively. In turn, *bird* governs its determiner *a*. Again, additional labels can decorate the nodes and relations in dependency structures, as shown in Figure 1.1.

Ambiguity. The following sentence is a classical example of a syntactic ambiguity, illustrated by the two derivation trees of Figure 1.2:

> She watches a man with a telescope.

This is called a *PP attachment* ambiguity: who exactly is using a telescope?

Semantics studies meaning. We often use logical languages to describe meaning, like the following (guarded) first-order sentence for “Every man loves a woman”:

$$\forall x.\text{man}(x) \supset \exists y.\text{love}(x, y) \wedge \text{woman}(y)$$

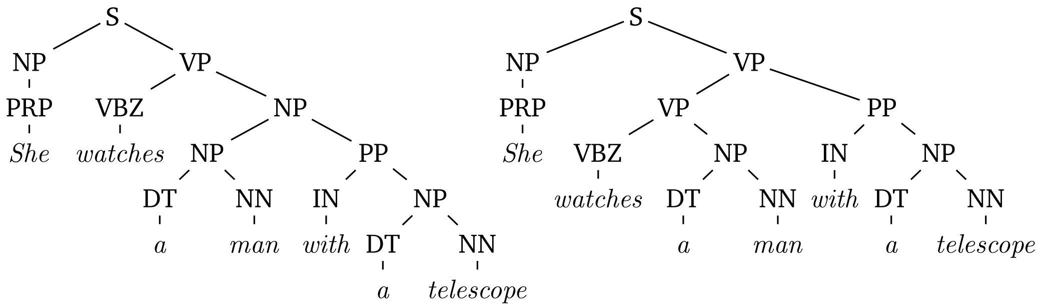


Figure 1.2: An ambiguous sentence.

or the description logic statement

$$\mathsf{Man} \sqsubseteq \exists \mathsf{love}.\mathsf{Woman}\ .$$

Ambiguity is of course present as in every aspect of language: for instance, scope ambiguities, as in this alternate reading of “Every man loves a woman”

$$\exists y.\text{woman}(y) \wedge \forall x.\text{man}(x) \supset \text{love}(x, y)$$

where there exists one particular woman loved by every man in the world.

More difficulties arise when we attempt to build meaning representations *compositionally,* based on syntactic structures, and when intensional phenomena must be modeled. The solutions often mix higher-order logics with possible-worlds semantics and modalities.

Pragmatics considers the ways in which meaning is affected by the *context* of a sentence: it includes the study of **discourse** and of **referential expressions**.

As usual, the models have to account for massive ambiguity, as in this **anaphora** resolution:

Mary asks Eve about her father

where *her* might refer to *Mary* or *Eve*; only the context of the sentence will allow to disambiguate.

1.1.2 Ambiguity at Every Turn

The above succinct presentation should convince the reader that ambiguity permeates every layer of the linguistic entreprise. To better emphasize the importance of ambiguity, let us look at experimental results in real-world syntax grammars: Martin et al. (1987) presents a typical sentence found in a corpus, which when generalized to arbitrary lengths n, exhibits a number of parses related to the Catalan numbers $C_n \sim \frac{4^n}{n^{3/2}\sqrt{\pi}}$. In more recent experiments with treebank-induced grammars, Moore (2004) reports an average number of 7.2×10^{27} different derivations for sentences of 5.7 words on average. The rationale behind these staggering levels of ambiguity is that any formal grammar that accounts for a realistic part of natural language, must allow for so many constructions, that it also yields an enormous number of different analyses: robustness of the model comes at a steep price in ambiguity.

The practical answer to this issue is to refine the models with **weights**, allowing to attach a grammaticality estimation to each structure. Those weights are typically probabilities inferred from frequencies found in large corpora. Stochastic methods are now ubiquitous in natural language processing (Manning and Schütze, 1999), and no purely symbolic model is able to compete with statistical models on practical benchmarks.

1.1.3 Romantics and Revolutionaries

Read (Pereira, 2000; Steedman, 2011).

1.2 Models of Syntax

To conclude this already long introduction, here is a short presentation of the kinds of models employed for describing syntax. Not every one will be covered in class, but there are pointers to the relevant literature.

For each of the two kinds of analyses, using constituents or dependencies, three different flavors of models can be distinguished: *generative* models, which construct structures through rewriting systems; *model-theoretic* approaches rather describe structures in a logical language and allow any model of a formula as an answer; *proof-theoretic* techniques establish the grammaticality of sentences through a proof in some formal deduction system. Finally, *stochastic* methods might be mixed with any of the previous frameworks (Manning and Schütze, 1999). This gives rise to twelve combinations—which should however not be distinguished too strictly, as their borders are often quite blurry.

1.2.1 Constituent Syntax

Generative Syntax The formal description of morpho-syntactic phenomena through rewriting systems can be traced back to 350BC and the Sanskrit grammar of Pāṇini, the *Aṣṭādhyāyī*. This large grammar employs *contextual* rewriting rules like

$$A \to B \;/\; C\_\_D \tag{1.1}$$

for “rewrite A to B in the context $C\_\_D$”, i.e. the rewrite rule

$$CAD \to CBD\ . \tag{1.2}$$

The grammar already features auxiliary symbols (like the labels on the inner nodes of Figure 1.1), and this type of formal systems is therefore already Turing-complete.

The adoption of **phrase-structure grammars** to derive constituent structures stems mostly from Chomsky’s *Three Models for the Description of Language* (1956), which considers the suitability of finite automata, context-free grammars, and transformational grammars for syntactic modeling.

Readers with a computer science background are likely to be rather familiar with context-free grammars from a compilers or formal languages course; it is quite interesting to see that the equivalent BNF notation (Backus, 1959) was developed at roughly the same time to specify the syntax of ALGOL 60 (Ginsburg and Rice, 1962). The focus in linguistics applications is however on *trees*, for which tree languages provide a more appropriate framework (Comon et al., 2007).

Model-Theoretic Syntax Because the focus of linguistic models of syntax is on trees, there is an alternative way of understanding a disjunction of context-free production rules

$$A \to BC \mid DE\;. \tag{1.3}$$

It posits that in a valid tree, a node labeled by A should feature two children, labeled either by B and C or by D and E. In first-order logic, assuming $A, B, \ldots$ to be predicates and using $\downarrow$ and $\to$ to denote the *child* and *right sibling* relations, this could be expressed as

$$\begin{aligned}\forall x.A(x) \supset \exists y.\exists z.x \downarrow y \wedge x \downarrow z \wedge y \to z \wedge \big((B(y) \wedge C(z)) \vee (D(y) \wedge E(z))\big)\\ \wedge\, \forall c.x \downarrow c \supset c = y \vee c = z\;.\end{aligned} \tag{1.4}$$

A constituent tree is valid if it satisfies the constraints stated by the grammar, and a language is the set of models, in a logical sense, of the grammar. See the survey by Pullum (2007) on the early developments of the model-theoretic approach.

Of course, the logical language of context-free rules is rather limited, and more expressive logics can be employed: we will consider **monadic second-order logic** and **propositional dynamic logic** in Chapter 4.

Proof-Theoretic Syntax Yet another way of viewing a context-free rule like (1.3) is as a deduction rules

$$A(xy) :\!-\; B(x), C(y). \tag{1.5}$$

$$A(xy) :\!-\; D(x), E(y). \tag{1.6}$$

(in Prolog-like syntax). Here the variables x and y range over finite strings, and a sentence w is accepted by the grammar if the judgement $S(w)$ (for "$w \in L(S)$") can be derived using the rules

$$\frac{B_1(u_1)\ \ldots\ B_m(u_m)}{A(u_1 \cdots u_m)}\{A(x_1 \cdots x_m) :\!-\; B_1(x_1), \ldots, B_m(x_m) \tag{1.7}$$

where $u_1, \ldots, u_m$ are finite strings.

The interest of this proof-theoretic view is that it is readily generalizable beyond context-free grammars, for instance by removing the restriction to monadic predicates, as in **multiple context-free grammars** (Seki et al., 1991). It also encourages annotations of proofs with terms (as with the Curry-Howard isomorphism) to construct a semantic representation of the sentence, and thus provides an elegant syntax/semantics interface.

1.2.2 Dependency Syntax

Dependency analyses take their roots in the work of Tesnière, and are especially well-suited to language with "relaxed" word order, where **discontinuities** come handy (Mel'čuk, 1988, e.g. Meaning-Text Theory for Czech). It also turns out that several of the best statistical parsing systems today rely on dependencies rather than constituents.

Generative Syntax If we look at the dependency structure of Figure 1.1, we can observe that it can be encoded through rewrite rules of the form

$$h \to L * R \tag{1.8}$$

where L is the list of left dependents and R that of right dependents of the head word h, and $*$ marks the position of this word: more concretely, the rules

$$\text{VBZ} \to \text{PRP} * \text{NN} \tag{1.9}$$
$$\text{PRP} \to * \tag{1.10}$$
$$\text{NN} \to \text{DT} * \tag{1.11}$$

would allow to generate the dependency tree on the right of Figure 1.1. This general idea has been put forward by Gaifman (1965) and Hays (1964).

Conversely, given a constituent tree like the one on the left of Figure 1.1, a dependency tree can be recovered by identifying the head of each phrase as in Figure 1.3. Applying this transformation to a context-free grammar results in a **head lexicalized** grammar, which is a fairly common idea in statistical parsing (e.g. Charniak, 1997; Collins, 2003).

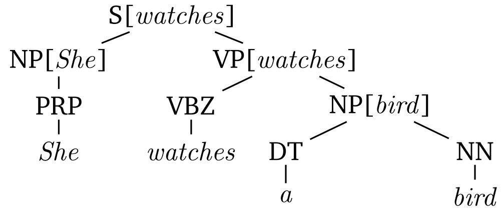


Figure 1.3: A head lexicalized constituent tree.

Model-Theoretic Syntax As with constituency analysis, dependency structures can be described in a model-theoretic framework. Here I do not know much work on the subject, besides a constraint-solving approach for a (positive existential) logic: the **topological dependency grammars** of Duchier and Debusmann (2001), along with related formalisms.

Proof-Theoretic Syntax Regarding the proof-theoretic take on dependency syntax, there is a very rich literature on **categorial grammar**. In the basic system of Bar-Hillel (1953), **categories** are built using left and right quotients over a finite set of symbols A:

$$\gamma ::= A \mid \gamma\backslash\gamma \mid \gamma/\gamma \tag{categories}$$

The proof system then features three deduction rules: one that looks up the possible categories associated to a word in a finite lexicon

$$\frac{}{w \vdash \gamma}\,\text{Lexicon}$$

and two rules to eliminate the $\backslash$ and $/$ connectors:

$$\frac{w_1 \vdash \gamma_1 \qquad w_2 \vdash \gamma_1\backslash\gamma_2}{w_1 \cdot w_2 \vdash \gamma_2}\,\backslash E$$

$$\frac{w_1 \vdash \gamma_2/\gamma_1 \qquad w_2 \vdash \gamma_1}{w_1 \cdot w_2 \vdash \gamma_2}\,/E$$

For instance, the dependencies from the right of Figure 1.1 can be described in a lexicon over $A \overset{\text{def}}{=} \{s, n, d\}$ by

$$\text{She} \vdash n \qquad \text{watches} \vdash (n\backslash s)/n \qquad \text{a} \vdash d \qquad \text{bird} \vdash d\backslash n$$

with a proof

$$\dfrac{\text{She} \vdash n \qquad \dfrac{\text{watches} \vdash (s\backslash n)/n \qquad \dfrac{\text{a} \vdash d \qquad \text{bird} \vdash d\backslash n}{\text{a bird} \vdash n}\,\backslash E}{\text{watches a bird} \vdash n\backslash s}\,/E}{\text{She watches a bird} \vdash s}\,\backslash E$$

By adding *introduction* rules to this proof system, Lambek (1958) has defined the **Lambek calculus**, which can be viewed as a non-commutative variant of linear logic (e.g. Troelstra, 1992). As with constituency analyses, one of the interests of proof-theoretic methods is that it provides an elegant way of building compositional semantics interpretations.

1.3 Further Reading

Interested readers will find a good general textbook on natural language processing by Jurafsky and Martin (2009). The present notes have a strong bias towards logical formalisms, but this is hardly representative of the general field of natural language processing. 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/`.

The more mathematics-oriented linguistics community is scattered around several sub-communities, each with its meeting. Let me mention two special interest groups of the ACL: *SIGMoL* on “Mathematics of Language” and *SIGParse* on “natural language parsing”.

Chapter 2

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 2.1 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 2.1: 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]

2.0.1 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 2.1 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 2.1.1, before returning to morphological analysis in Section 2.1.2.

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

2.0.2 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 2.1 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 2.2 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:

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

Table 2.2: 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

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 2.2.

2.1 Finite-State Morphology

2.1.1 *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 2.2.2 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 2.1. Show that the range $R(\Sigma^*)$ of a rational transduction R is a rational language. **(∗)**

Exercise 2.2. Show that if L is a rational language over Σ, then Id_L is a rational transduction over $\Sigma^* \times \Sigma^*$. **(∗)**

Exercise 2.3. Show that rational transductions are closed under inverse and composition. **(∗∗)**

Exercise 2.4. Let R be a relation over $\Sigma^* \times \Delta^*$. Show that R is a rational relation iff it is a rational transduction. **(∗∗)**

Remark 2.1 (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 $R^\star$ is not: let $L = \{(ab)^n \mid n \geq 0\}$, then

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

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 2.5.** 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 2.6.** 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 2.2 (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 2.2),
- $q_0 = \varepsilon^{-1}f$,
- $\iota = \bigwedge_{v \in \Sigma^*} f(v)$ 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 2.7 (Hadamard Product). 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 2.8 (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).

2.1.2 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 hospital−ize[−vbd], where the POS tag VBG, noted [−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. [−vbd] can follow any verb stem, but the suffix *-ation* should be applied to verbs ending with *-ize* as in *hospitalization*),

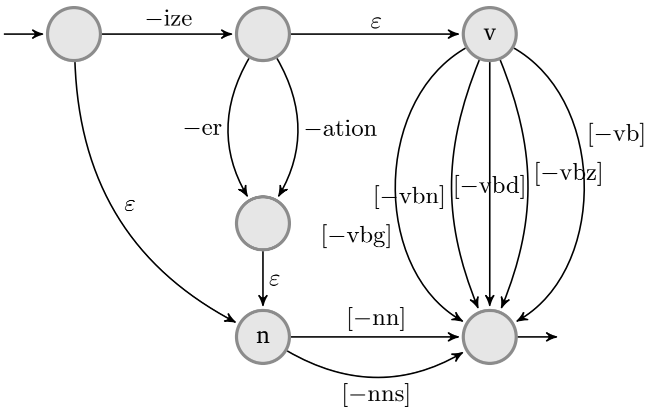


Figure 2.1: A finite state automaton for some morphotactics applicable to nouns ending with *-al*.

2. implement the morphological rules (e.g. [−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 2.1 presents morphotactics that derive for instance *hospitalizations* (hospital−ize−ation[−nns]) or *hospitalized* (hospital−ize[−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 sad[−jjr]) or *nicer* (nice[−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* (en−joy−able) and *enrichable* (en−rich−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.

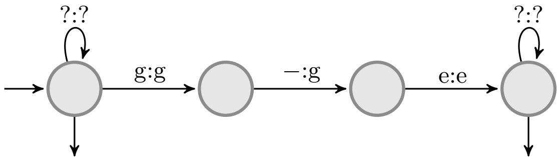


Figure 2.2: A transducer for rewrite rule (2.2). Question marks ?:? stand for $a{:}a$ for all $a \in \Sigma$.

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

$$[-\mathrm{vbn}] \to -\mathrm{ed} \tag{2.1}$$

followed by the orthographic rule

$$\mathrm{g}{-}\mathrm{e} \to \mathrm{gge} \tag{2.2}$$

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

$$\mathrm{Id}_{\Sigma^*} \cdot \mathrm{g}{-}\mathrm{e{:}gge} \cdot \mathrm{Id}_{\Sigma^*} \tag{2.3}$$

for the one-rule system consisting of rule (2.2), which can be implemented by a finite transducer, as the one of Figure 2.2 for (2.2).

The remainder of this section is dedicated to the translation from string rewriting formalisms into finite state transducers: Section 2.1.3 presents a formalism of cascaded **phonological rules** to this end.

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

2.1.3 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

$$\alpha \to \beta \;/\; \lambda \underline{\quad\quad} \rho \tag{2.4}$$

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

$$\{x\,u\,y \to x\,v\,y \mid (x,u,v,y) \in \lambda \times \alpha \times \beta \times \rho\} \tag{2.5}$$

—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 2.3. Let us focus for instance on past participle inflection: (2.2) can be restated with this notation as

$$- \to \mathrm{g} \;/\; \mathrm{g}\underline{\quad\quad}\mathrm{e}\;. \tag{2.6}$$

Keeping with past participle composition, we would also need

$$- \rightarrow \varepsilon \;/\; \underline{\qquad} \tag{2.7}$$

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

$$\mathrm{e}- \rightarrow \varepsilon \;/\; \underline{\qquad}\mathrm{e} \tag{2.8}$$

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

$$\mathrm{sing}[-\mathrm{vbn}] \rightarrow \mathrm{sung} \;/\; \underline{\qquad}\;. \tag{2.9}$$

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 (2.9), then (2.1), then (2.6) and (2.8), and (2.7) last. Furthermore, we should make the rewrites **obligatory**: if (2.9) or (2.6) can be applied, then they should be. But not all rules should be obligatory: for instance, with

$$\mathrm{prove}[-\mathrm{vbn}] \rightarrow \mathrm{proven} \;/\; \underline{\qquad} \tag{2.10}$$

both *proven* and *proved* are accepted as past participles of *to prove*, thus (2.10) should be **optional**. Adding some derivational morphology to the mix allows to witness another issue with rule application:

$$[-\mathrm{vbg}] \rightarrow -\mathrm{ing} \;/\; \underline{\qquad} \tag{2.11}$$

$$\mathrm{e}- \rightarrow \varepsilon \;/\; \underline{\qquad}\mathrm{i} \tag{2.12}$$

could model gerund inflection in *dancing*. In order to obtain *passivizing*, we need to apply (2.11) once to $\mathrm{passive{-}ize}[-\mathrm{vbg}]$, and (2.12) on all the applicable factors of $\mathrm{passive{-}ize{-}ing}$: we are actually applying the *transitive closure* of the one-step derivation relation defined by rule (2.12).

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 (2.4) seems to be exactly the derivation relation of the system (2.5). 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 (2.4), 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 $r = \alpha \rightarrow \beta \;/\; \lambda\underline{\qquad}\rho$, 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

$$(z_0 v_0 z_1 \cdots z_{i-1}, u_i, v_i, z_i \cdots z_{n-2} u_{n-1} z_{n-1}) \in (\Sigma^* \cdot \lambda) \times \alpha \times \beta \times (\rho \cdot \Sigma^*) \qquad (2.13)$$

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

$$[\![r]\!] = \mathrm{right}_\rho \mathbin{⨾} \mathrm{replace}_{\alpha,\beta} \mathbin{⨾} \mathrm{left}_\lambda \qquad (2.14)$$

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

(∗∗∗) **Exercise 2.10.** 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 $\Sigma^* \times \Delta^*$, $\$w\$ \; [\![\mathcal{P}]\!] \; \$w'\$$ 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 $w \; [\![\mathcal{P}]\!] \; w'$, is PSPACE-hard.

2.2 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!

2.2.1 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 2.1.3, 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

$$\mathrm{nnp}\,\mathrm{vbn} \to \mathrm{nnp}\,\mathrm{vbd} \tag{2.15}$$

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

$$\mathrm{vbd}\,\mathrm{in} \to \mathrm{vbn}\,\mathrm{in} \tag{2.16}$$

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 $\mathcal{C} = r_1 r_2 \ldots r_n$ 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

$$w \overset{r_i}{\underset{\text{lm}}{\Longrightarrow}} w' \text{ iff } \exists x, y \in \Sigma^*, w = xu_iy \wedge w' = xv_iy \wedge (\forall z, z' \in \Sigma^*, w \neq zu_iz' \vee x \leq_{\text{pref}} z) \tag{2.17}$$

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

$$\llbracket r_i \rrbracket = \overset{r_i}{\underset{\text{lm}}{\Longrightarrow}} \cup \mathrm{Id}_{\overline{\Theta^* \cdot u_i \cdot \Theta^*}} \,, \tag{2.18}$$

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

$$\llbracket \mathcal{C} \rrbracket = \llbracket r_1 \rrbracket \fatsemi \llbracket r_2 \rrbracket \fatsemi \cdots \fatsemi \llbracket r_n \rrbracket \,. \tag{2.19}$$

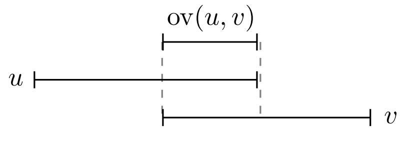
A naive implementation of $\mathcal{C}$ would try to match each u_i at every position of the input string w in Σ^*, resulting in an overall complexity of $O(|w| \cdot \sum_i |u_i|)$. However, one often faces the problem of tagging a *set* of sentences $\{w_1, \ldots, w_m\}$, which yields $O((\sum_i |u_i|) \cdot (\sum_j |w_j|))$. 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 (2.19) and the closure of sequential functions under composition, it suffices to prove that $\llbracket r_i \rrbracket$ is a sequential function for each i in order to prove that $\llbracket \mathcal{C} \rrbracket$ is a sequential function. Since each $\llbracket r_i \rrbracket$ 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 $\llbracket r_i \rrbracket$ 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 2.4 (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 $v = uv_1 = v_2u$. For $v \neq \varepsilon$, the longest border of v different from v itself is denoted $\mathrm{bord}(v)$.

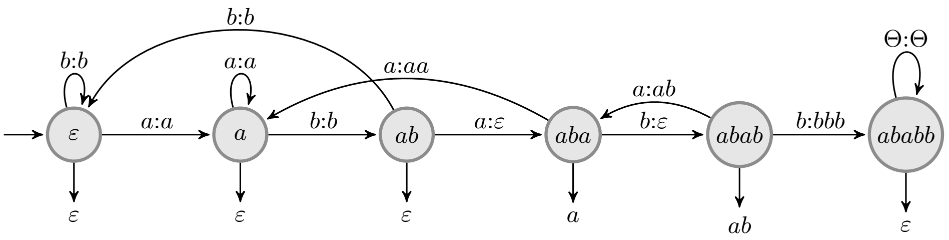
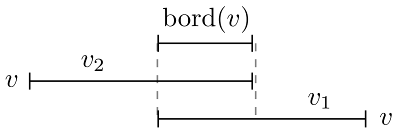


Figure 2.3: The sequential transducer constructed for $ababb \to abbbb$.



Exercise 2.11. Show that for all u, v in Θ^* and a in Θ, **(∗)**

$$\operatorname{ov}(ua, v) = \begin{cases} \operatorname{ov}(u,v)\cdot a & \text{if } \operatorname{ov}(u,v)\cdot a \leq_{\text{pref}} v \\ \operatorname{bord}(\operatorname{ov}(u,v)\cdot a) & \text{otherwise.}\end{cases} \tag{2.20}$$

Definition 2.5 (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

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

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 $\operatorname{Pref}(u)$,

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

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

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

For instance, the sequential transducer for the rule $ababb \to abbbb$ is shown in Figure 2.3 (one can check that $ababb \wedge abbbb = ab$, $\operatorname{bord}(b) = \varepsilon$, $\operatorname{bord}(aa) = a$, $\operatorname{bord}(abb) = \varepsilon$, $\operatorname{bord}(abaa) = a$, and $\operatorname{bord}(ababa) = aba$).

Proposition 2.6. *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 2.6.1. For all w in $\overline{\Theta^* \cdot u \cdot \Theta^*}$,

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

Proof of Claim 2.6.1. 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

$$\begin{aligned}
\delta(\varepsilon,wa) &= \delta(\delta(\varepsilon,w),a) && \text{(by def.)}\\
&= \delta(\mathrm{ov}(w,u),a) && \text{(by ind. hyp.)}\\
&= \mathrm{ov}(wa,u) && \text{(by (2.20))}\\
\eta(\varepsilon,wa) &= \eta(\varepsilon,w)\cdot\eta(\delta(\varepsilon,w),a) && \text{(by def.)}\\
&= w\cdot\rho(\delta(\varepsilon,w))^{-1}\cdot\eta(\delta(\varepsilon,w),a) && \text{(by ind. hyp.)}\\
&= w\cdot\rho(w')^{-1}\cdot\eta(w',a)\ ; && \text{(by setting } w' = \delta(\varepsilon,w))
\end{aligned}$$

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

Case $w'a \not\leq_{\mathrm{pref}} u$ 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 $w'a <_{\mathrm{pref}} u$ Then $\delta(\varepsilon,wa) = w'a$, and we need to further distinguish between several cases:

$w'a \leq_{\mathrm{pref}} (u \wedge v)$ then $\rho(w') = \varepsilon$, $\eta(w',a) = a$, and $\rho(w'a) = \varepsilon$, thus

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

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

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

$(u \wedge v) <_{\mathrm{pref}} w'$ 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

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

[2.6.1]

Claim 2.6.1 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 2.6.1 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 $(u \wedge v) <_{\mathrm{pref}} u'$

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

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

Lemma 2.7. *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 $\bigwedge \llbracket \mathcal{T}_{r(w)} \rrbracket(\Theta^*) = \varepsilon$.

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

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

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 $\llbracket \mathcal{T}_{r(w)} \rrbracket(u') \wedge \llbracket \mathcal{T}_{r(w)} \rrbracket(\varepsilon) = \varepsilon$.

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

Proposition 2.8. *Let $r = u \to v$ with $u \neq \varepsilon$ and $u \neq v$. Then $\mathcal{T}_r$ is the minimal sequential transducer for $\llbracket r \rrbracket$.*

Proof. Let $w <_{\mathrm{pref}} w'$ be two different states in $\mathrm{Prefix}(u)$; we proceed to prove that $\llbracket w^{-1}\mathcal{T}_r \rrbracket \neq \llbracket w'^{-1}\mathcal{T}_r \rrbracket$, hence that no two states of $\mathcal{T}_r$ can be merged. By Lemma 2.7 it suffices to prove that $\llbracket \mathcal{T}_{r(w)} \rrbracket \neq \llbracket \mathcal{T}_{r(w')} \rrbracket$, thus to exhibit some $x \in \Theta^*$ such that $\llbracket \mathcal{T}_{r(w)} \rrbracket(x) \neq \llbracket \mathcal{T}_{r(w')} \rrbracket(x)$. We perform a case analysis:

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

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

if $w \leq_{\mathrm{pref}} (u \wedge v)$ and $w' = u$ then $\llbracket \mathcal{T}_{r(w')} \rrbracket(x) = x$ for all x and we consider

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

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

$$\llbracket \mathcal{T}_{r(w)} \rrbracket(\varepsilon) \neq \llbracket \mathcal{T}_{r(w')} \rrbracket(\varepsilon) \;.$$ □

Exercise 2.12. Define the minimal sequential transducers for $r = u \to v$ in the cases $u = \varepsilon$ and $u = v$. **(∗)**

2.2.2 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

$$p(a_1 \cdots a_n) = \prod_{i=1}^{n} p(a_i | \$ a_1 \cdots a_{i-1}) \; . \tag{2.21}$$

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 2.9 (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:

$$\forall n, \sum_{w \in \Sigma^n} p(w) = 1 \; . \tag{2.22}$$

Example 2.10. 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 (2.21), 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 (2.21) with

$$p(a_1 \cdots a_n) \approx \prod_{i-1}^{n} p(a_i | a_{i-N+1} \cdots a_{i-1}) \; , \tag{2.23}$$

(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 $t_1 \cdots t_{N-1}$:

$$p(t|t_1 \cdots t_{N-1}) = \frac{C(t_1 \cdots t_{N-1}t)}{C(t_1 \cdots t_{N-1})} \tag{2.24}$$

(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

$$p(t'_1 \cdots t'_N | t_1 \cdots t_N) = \begin{cases} p(t'_N | t_2 \cdots t_N) & \text{if } \forall 1 \leq i \leq N-1, t'_i = t_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.25}$$

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

$$p(a|t_1 \cdots t_N) = \frac{1}{|\Sigma|^{N-1}} \sum_{a_1 \cdots a_{N-1} \in \Sigma^{N-1}} \frac{C((a_1, t_1) \cdots (a, t_N))}{\sum_{a_N \in \Sigma} C((a_1, t_1) \cdots (a_N, t_N))} \tag{2.26}$$

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 (2.24) and (2.26) 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

$$\rho = \operatorname*{argmax}_{\rho' \in Q^n} p(\rho' | w) \ , \tag{2.27}$$

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

$$\begin{aligned} \rho &= \operatorname*{argmax}_{\rho' \in Q^n} \frac{p(w|\rho')\, p(\rho')}{p(w)} \\ &= \operatorname*{argmax}_{\rho' \in Q^n} p(w|\rho')\, p(\rho') \ . \end{aligned} \tag{2.28}$$

The usual procedure to compute the result of (2.28) 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 $a_1 \cdots a_{i+1}$:

$$V(i+1, q) = \max_{\rho' \in Q^i} p(a_1 \cdots a_{i+1} | \rho' q)\, p(\rho' q) \ . \tag{2.29}$$

For $i = 0$, this probability is clearly

$$V(1, q) = p(a_1 | q)\, p(q | \$) = E(q)(a_1)\, S(q) \ . \tag{2.30}$$

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

$$\begin{aligned} V(i+1, q) &= \max_{\rho' \in Q^{i-1}, q' \in Q} p(a_1 \cdots a_i a_{i+1} | \rho' q' q)\, p(\rho' q' q) \\ &= \max_{\rho' \in Q^{i-1}, q' \in Q} p(a_1 \cdots a_i | \rho' q')\, p(a_{i+1} | q)\, p(\rho' q')\, p(q | q') \\ &= \max_{q' \in Q} V(i, q')\, p(a_{i+1} | q)\, p(q | q') \\ &= E(q)(a_{i+1}) \max_{q' \in Q} V(i, q')\, T(q')(q) \ . \end{aligned} \tag{2.31}$$

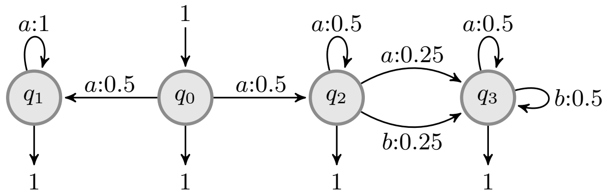


Figure 2.4: The probabilistic automaton for the HMM of Example 2.10.

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

$$B(i, q) = \operatorname*{argmax}_{q' \in Q} V(i, q')\, T(q')(q) \;. \tag{2.32}$$

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

$$q_n = \operatorname*{argmax}_{q \in Q} V(n, q) \;, \tag{2.33}$$

and we can work our way back from there using

$$q_i = B(i, q_{i+1}) \;, \tag{2.34}$$

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

Example 2.11. For the HMM of Example 2.10 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 (2.31) 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,

$$\mu(a)(q_0, q_0) = 0 \quad \mu(a)(q_0, q') = S(q') \cdot E(q')(a) \quad \mu(a)(q, q') = T(q)(q') \cdot E(q')(a) \tag{2.35}$$

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 2.4 shows the probabilistic automaton that corresponds to the HMM of Example 2.10.

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

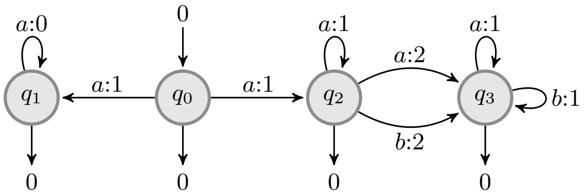


Figure 2.5: The tropical automaton $-\log \mathcal{H}$ for the HMM $\mathcal{H}$ of Example 2.10.

(2.35):

$$\begin{aligned}\rho &= \operatorname*{argmax}_{\rho' \in Q^n} p(w|\rho')\, p(\rho') && (2.28)\\ &= \operatorname*{argmax}_{q_1 \cdots q_n} \prod_{i=1}^{n} p(a_i|q_i)\, p(q_i|q_{i-1}) \\ &= \operatorname*{argmax}_{q_1 \cdots q_n} S(q_1)\, E(q_1)(a_1) \prod_{i=2}^{n} E(q_i)(a_i)\, T(q_{i-1})(q_i)\ . && (2.36)\end{aligned}$$

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 2.5.) 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 (2.36) becomes

$$\rho = \operatorname*{argmin}_{q_1 \cdots q_n} \left(-\log S(q_1) - E(q_1)(a_1) - \sum_{i=2}^{n} \log E(q_i)(a_i) + \log T(q_{i-1})(q_i) \right), \tag{2.37}$$

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

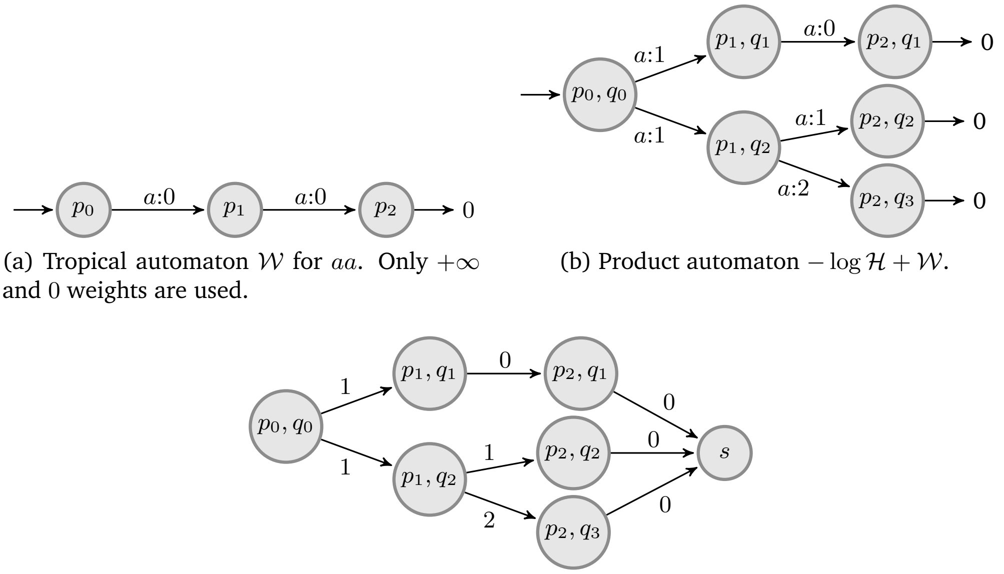
We can effectively build the product of our weighted automaton $-\log \mathcal{H}$ with an automaton $\mathcal{W}$ for the singleton language $\{w\}$ (Figure 2.6a). The transition labels in the resulting weighted automaton $-\log \mathcal{H} + \mathcal{W}$ (Figure 2.6b) are then useless, and we can see 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 2.6c) then allows to use a single-pair shortest path algorithm between (p_0, q_0) and s to find a solution for (2.37) ($q_1 q_1$ in the example of Figure 2.6).

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. 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

$$\begin{aligned} &O\big((|w| - 1)\,|Q|^2 + |Q| + (|w|\,|Q| + 1) \log(|w|\,|Q| + 1)\big) \\ &\qquad\qquad\qquad\qquad = O\big(|w|\,|Q|^2 + |w|\,|Q|\, \log(|w|\,|Q|)\big)\,, \end{aligned}$$



(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 2.6: Construction steps for the tagging algorithm on aa.

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 (2.37) 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 2.2.1, there is no general determinization procedure for (weighted automata translations of) HMMs. Consider for instance the automaton of Figure 2.5 for the HMM of Example 2.10: it implements the series over the tropical semiring defined by

$$\langle s, w\rangle = \begin{cases} 1 & \text{if } w = a^n, \\ n + 2 + |w'| & \text{if } w = a^n b w',\ w' \in \{a, b\}^*. \end{cases} \tag{2.38}$$

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 2.2 for sequential series, s is not sequential (see Lombardy and Sakarovitch, 2006, Theorem 8, where $w^{-1}s$ is noted $[w^{-1}s]^\sharp$).

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

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

Chapter 3

Context-Free 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 constituents can be recursively decomposed until we reach the individual words, effectively describing a tree:

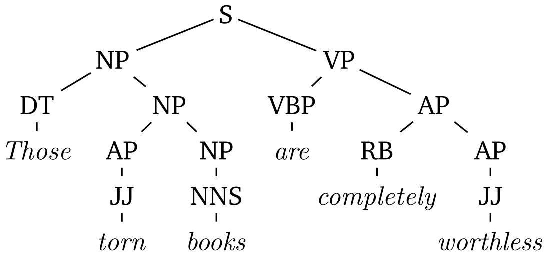


Figure 3.1: A context-free derivation tree.

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-rewriting systems.

3.1 Grammars

Definition 3.1 (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. $S \Rightarrow^\star \alpha$ are called **sentential forms**, whereas strings w in Σ^* s.t. $S \Rightarrow^\star w$ are called **sentences**. The *language* of $\mathcal{G}$ is its set of sentences, i.e.

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

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 3.2. The derivation tree of Figure 3.1 corresponds to the context-free grammar with

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

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

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 part-of-speech tagging information, based on the Penn TreeBank tagset (Santorini, 1990). A different formalization could set $\Sigma = \{\mathrm{DT}, \mathrm{JJ}, \mathrm{NNS}, \mathrm{VBP}, \mathrm{RB}\}$ and delegate the POS tagging issues to an external device.

3.1.1 The Parsing Problem

Context-free grammars enjoy a number of nice 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 P-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).

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

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 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 known to be in $O(n^{2.3727})$ (Williams, 2012). 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.

Parsing Techniques 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 these notes on syntax.

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 3.2:

She watches a man with a telescope.

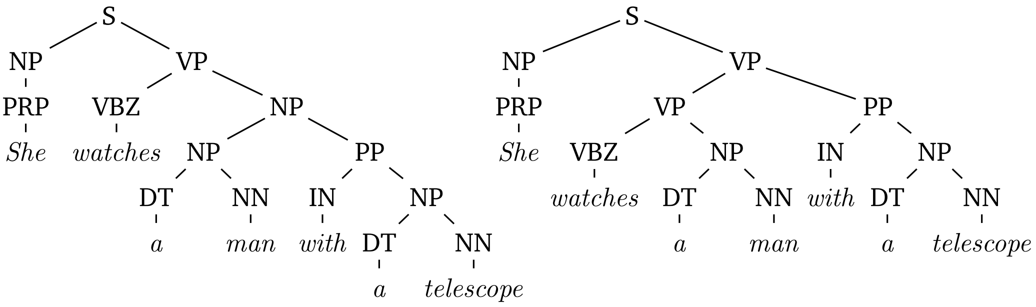


Figure 3.2: An ambiguous sentence.

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 3.3 (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.

3.1.2 *Background:* Tree Automata

Because our focus in linguistics analyses is on *trees*, context-free grammars are mostly useful as a means to define tree languages. Let us first recall basic definitions on regular tree languages.

Definition 3.4 (Finite Tree Automata)**.** A **finite tree automaton** (NTA) is a tuple $\mathcal{A} = \langle Q, \mathcal{F}, \delta, I\rangle$ where Q is a finite set of *states*, $\mathcal{F}$ a *ranked alphabet*, δ a finite *transition relation* in $\bigcup_n Q \times \mathcal{F}_n \times Q^n$, 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 $\mathcal{F} = Q \uplus \mathcal{F}$ where the states in Q have arity 0: either **bottom-up**:

$$\begin{aligned} R_B &= \{a^{(n)}(q_1^{(0)}, \ldots, q_n^{(0)}) \to q^{(0)} \mid (q, a^{(n)}, q_1, \ldots, q_n) \in \delta\} \\ L(\mathcal{A}) &= \{t \in T(\mathcal{F}) \mid \exists q \in I, t \xRightarrow{R_B}{}^{\star} q\}\ , \end{aligned}$$

or **top-down**:

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

A tree language $L \subseteq T(\mathcal{F})$ is **regular** if there exists an NTA $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Example 3.5. The 2^n derivation trees for a^n in the grammar of Example 3.3 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

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

It is rather easy to define the set of derivation trees of a CFG through an NTA. The only slightly annoying point is that nonterminals in a CFG do not have a fixed arity; for instance if $A \to BC \mid a$ are two productions, then an A-labeled node in a derivation tree might have two children B and C or a single child a. This motivates the notation $A^{(r)}$ for an A-labeled node with rank r.

Definition 3.6 (Derived Tree Language)**.** Let $\mathcal{G} = \langle N, \Sigma, P, S\rangle$ be a context-free grammar and let m be its maximal right-hand side length. Its **derived tree language** $T(\mathcal{G})$ is defined as the language of the NTA $\mathcal{A} = \langle V \uplus \{\varepsilon\}, \mathcal{F}, \delta, \{S\}\rangle$, where

$$\begin{aligned}\mathcal{F} &\overset{\text{def}}{=} \{a^{(0)} \mid a \in \Sigma\} \cup \{\varepsilon^{(0)}\} \cup \{A^{(1)} \mid A \to \varepsilon \in P\}\\ &\quad \cup \{A^{(m)} \mid m > 0 \text{ and } \exists A \to X_1 \cdots X_m \in P \text{ with } \forall i.X_i \in V\}\\ \delta &\overset{\text{def}}{=} \{(A, A^{(m)}, X_1, \ldots, X_m) \mid m > 0 \land A \to X_1 \cdots X_m \in P\}\\ &\quad \cup\ \{(A, A^{(1)}, \varepsilon) \mid A \to \varepsilon \in P\}\\ &\quad \cup\ \{(a, a^{(0)}) \mid a \in \Sigma\}\\ &\quad \cup\ \{(\varepsilon, \varepsilon^{(0)})\}\ .\end{aligned}$$

The class of derived tree languages of context-free grammars is a strict subclass of the class of regular tree languages.

$(**)$ **Exercise 3.1** (Local Tree Languages)**.** Let $\mathcal{F}$ be a ranked alphabet and t a term of $T(\mathcal{F})$. We denote by $\mathsf{r}(t)$ the root symbol of t and by $\mathsf{b}(t)$ the set of *local branches* of t, defined inductively by

$$\mathsf{r}(a^{(0)}) \overset{\text{def}}{=} a^{(0)} \qquad \mathsf{b}(a^{(0)}) \overset{\text{def}}{=} \emptyset$$
$$\mathsf{r}(f^{(n)}(t_1, \ldots, t_n)) \overset{\text{def}}{=} f^{(n)} \quad \mathsf{b}(f^{(n)}(t_1, \ldots, t_n)) \overset{\text{def}}{=} \{f^{(n)}(\mathsf{r}(t_1), \ldots, \mathsf{r}(t_n))\} \cup \bigcup_{i=1}^{n} \mathsf{b}(t_i)\ .$$

For instance $\mathsf{b}(\ f(g(a), f(a,b))\ ) = \{f(g,f), g(a), f(a,b)\}$.

A tree language $L \subseteq T(\mathcal{F})$ is **local** if and only if there exist two sets $R \subseteq \mathcal{F}$ of root symbols and $B \subseteq \mathsf{b}(T(\mathcal{F}))$ of local branches, such that $t \in L$ iff $\mathsf{r}(t) \in R$ and $\mathsf{b}(t) \subseteq B$. Let

$$L(R, B) = \{t \in T(\mathcal{F}) \mid \mathsf{r}(t) \in R \text{ and } \mathsf{b}(t) \subseteq B\}\ ;$$

then a tree language L is local if and only if $L = L(\mathsf{r}(L), \mathsf{b}(L))$.

1. Show that $\{ f(g(a), g(b)) \}$ is not a local tree language.
2. Show that any local tree language is the language of some NTA.
3. Show that a tree language included in $T(\mathcal{F})$ is local with $R \subseteq \mathcal{F}_{>0}$ if and only if it is the derived tree language of some CFG.
4. Show that any regular tree language is the homomorphic image of a local tree language by an *alphabetic tree morphism*, i.e. the application of a relabeling to the tree nodes.
5. Given a tree language L, let $\mathrm{Yield}(L) \stackrel{\text{def}}{=} \bigcup_{t \in L} \mathrm{Yield}(t)$ and define inductively $\mathrm{Yield}(a^{(0)}) \stackrel{\text{def}}{=} a$ and $\mathrm{Yield}(f^{(r)}(t_1, \ldots, t_r) \stackrel{\text{def}}{=} \mathrm{Yield}(t_1) \cdots \mathrm{Yield}(t_r)$. Show that, if L is a regular tree language, then $\mathrm{Yield}(L)$ is a context-free language.

3.2 Tabular Parsing

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

3.2.1 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 construction of Bar-Hillel et al. *(1961) for parsing is due to Lang (1994).*

Theorem 3.7 (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*

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

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 3.8. *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

$$X \Rightarrow^n_{\mathcal{G}'} w \text{ implies } X \Rightarrow^\star_{\mathcal{G}} w \tag{3.1}$$

$$[\alpha] \Rightarrow^n_{\mathcal{G}'} w \text{ implies } \alpha \Rightarrow^\star_{\mathcal{G}} w \tag{3.2}$$

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 $X \Rightarrow^\star_{\mathcal{G}} w = \varepsilon$, or
2. $\beta = Y$ in V implies $X \Rightarrow^\star_{\mathcal{G}} w$ by induction hypothesis (3.1), or
3. $\beta = [Y][\gamma]$ with $[Y]$ and $[\gamma]$ in N' implies again $X \Rightarrow^\star_{\mathcal{G}} w$ by induction hypothesis (3.2) and context-freeness, since in that case $X \to Y\gamma$ is in P.

Similarly, a derivation

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

implies $\alpha \Rightarrow^\star w$ by induction hypothesis (3.1) if $|\alpha| = 1$ and thus $\beta = \alpha$, or by induction hypothesis (3.2) and context-freeness if $\alpha = Y\gamma$ with Y in V and γ in V^+, and thus $\beta = [Y][\gamma]$. □

3.2.2 Parsing as Deduction

In practice, we want to perform at least some of the reduction of the tree automaton constructed by Theorem 3.7 *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 $t \overset{R_B}{\Longrightarrow}{}^\star\, q$. 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.

$$\frac{(X_1,q_0,q_1),\ldots,(X_m,q_{m-1},q_m)}{(A,q_0,q_m)}\left\{\begin{array}{l} m>0,\ A\to X_1\cdots X_m\in P\\ q_0,q_1,\ldots,q_m\in Q\end{array}\right. \qquad \text{(Internal)}$$

$$\frac{}{(A,q,q)}\left\{\begin{array}{l} A\to\varepsilon\in P\\ q\in Q\end{array}\right. \qquad \text{(Empty)}$$

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

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 3.7.

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 3.2. 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

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

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

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

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

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

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

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 3.3. 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 3.3. 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 3.3.** 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 3.4.** 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.

Chapter 4

Model-Theoretic Syntax

In contrast with the generative approaches of the first part of the course, we take here a different stance on how to formalise constituent-based syntax. Instead of a more or less operational description using some string or term rewrite system, the trees of our linguistic analyses are now *models* of logical formulæ.

4.0.1 Model-Theoretic vs. Generative

The connections between the classes of tree structures that can be singled out through logical formulæ on the one hand and context-free grammars or finite tree automata on the other hand are well-known, and we will survey some of these bridges. Thus the interest of a model theoretic approach does not reside so much in what can be expressed as in *how* it can be expressed.

Most of this discussion is inspired by Pullum and Scholz (2001).

Local vs. Global View The model-theoretic approach simplifies the specification of global properties of syntactic analyses. Let us consider for instance the problem of finding the **head** of a constituent, which can be used to lexicalise CFGs. Remember that the solution there was to explicitly annotate each nonterminal with the head information of its subtree—which is the only way to percolate the head information up the trees in a context-free grammar. On the other hand, one can write a logic formula postulating the existence of a unique head word for each node of a tree (see (4.19) and (4.20)).

Gradience of Grammaticality Agrammatical sentences can vary considerably in their *degree* of agrammaticality. Rather than a binary choice between grammatical and agrammatical, one would rather have a finer classification that would give increasing levels of agrammaticality to the following sentences:

Practical aspects of the notion of grammaticality gradience have been investigated in the context of ***property grammars****, see e.g. Duchier* et al. *(2009).*

*In a hole in in the ground there lived a hobbit.
*In a hole in in ground there lived a hobbit.
*Hobbit a ground in lived there a the hole in.

One way to achieve this finer granularity with generative syntax is to employ weights as a measure of grammaticality. Note that it is not quite what we obtained through probabilistic methods, because estimated probabilities are not grammaticality judgements per se, but occurrence-based (although smoothing techniques attempt to account for missing events).

A natural way to obtain a gradience of grammaticality using model theoretic methods is to structure formulæ as large conjunctions $\bigwedge_i \varphi_i$, where each conjunct

φ_i implements a specific linguistic notion. A degree of grammaticality can be derived from (possibly weighted) counts of satisfied conjuncts.

Open Lexicon An underlying assumption of generative syntax is the presence of a *finite* lexicon Σ. A specific treatment is required in automated systems in order to handle unknown words.

This limitation is at odds with the diachronic addition of new words to languages, and with the grammaticality of sentences containing **pseudo-words**, as for instance

> Could you hand over the salt, please?
> Could you smurf over the smurf, please?

Again, structuring formulæ in such a way that lexical information only further *constrains* the linguistic trees makes it easy to handle unknown or pseudo-words, which simply do not add any constraint.

Infinite Sentences A debatable point is whether natural language sentences should be limited to finite ones. An example illustrating why this question is not so clear-cut is an expression for "mutual belief" that starts with the following:

> Jones believes that iron rusts, and Smith believes that iron rusts, and Jones believes that Smith believes that iron rusts, and Smith believes that Jones believes that iron rusts, and Jones believes that Smith believes that Jones believes that iron rusts, and...

Dealing with infinite sequences and trees requires to extend the semantics of generative devices (CFGs, PDAs, etc.) and leads to complications. By contrast, logics are not *a priori* restricted to finite models, and in fact the two examples we will see are expressive enough to force the choice of either infinite or finite models. Of course, for practical applications one might want to restrict oneself to finite models.

Algorithmic Costs Formulæ in the logics considered in this chapter are provably more succinct than context-free grammars. The downfall is an algorithmic cost increased in the same proportion, e.g. parsing can require exponential time for PDL (Afanasiev et al., 2005), and non-elementary time for wMSO (Meyer, 1975; Reinhardt, 2002).

4.0.2 Tree Structures

Before we turn to the two logical languages that we consider for model-theoretic syntax, let us introduce the structures we will consider as possible models. Because we work with constituent analyses, these will be **labelled ordered trees**. Given a set A of labels, a **tree structure** is a tuple $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$ where W is a set of nodes, $\downarrow$ and $\rightarrow$ are respectively the **child** and **next-sibling** relations over W, and each P_a for a in A is a unary labelling relation over W. We take W to be isomorphic to some *prefix-closed* and *predecessor-closed* subset of $\mathbb{N}^*$, where $\downarrow$ and $\rightarrow$ can then be defined by

$$\downarrow \overset{\text{def}}{=} \{(w, wi) \mid i \in \mathbb{N} \wedge wi \in W\} \tag{4.1}$$

$$\rightarrow \overset{\text{def}}{=} \{(wi, w(i+1)) \mid i \in \mathbb{N} \wedge w(i+1) \in W\}\ . \tag{4.2}$$

Note that (a) we do not limit ourselves to a single label per node, i.e. we actually work on trees labelled by $\Sigma \overset{\text{def}}{=} 2^A$, (b) we do not bound the rank of our trees, and (c) we do not assume the set of labels to be finite.

See Comon et al. *(2007, Section 8.3.1).*

Binary Trees One way to deal with unranked trees is to look at their encoding as "first child/next sibling" binary trees. Formally, given a tree structure $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$, we construct a **labelled binary tree** t, which is a partial function $\{0,1\}^* \to \Sigma$ with a prefix-closed domain. We define for this $\text{dom}(t) = \text{fcns}(W)$ and $t(w) = \{a \in A \mid P_a(\text{fcns}^{-1}(w))\}$ for all $w \in \text{dom}(t)$, where

$$\text{fcns}(\varepsilon) \overset{\text{def}}{=} \varepsilon \qquad \text{fcns}(w0) \overset{\text{def}}{=} \text{fcns}(w)0 \qquad \text{fcns}(w(i+1)) \overset{\text{def}}{=} \text{fcns}(wi)1 \tag{4.3}$$

for all w in $\mathbb{N}^*$ and i in $\mathbb{N}$ and the corresponding inverse mapping is

$$\text{fcns}^{-1}(\varepsilon) \overset{\text{def}}{=} \varepsilon \quad \text{fcns}^{-1}(w0) \overset{\text{def}}{=} \text{fcns}^{-1}(w)0 \qquad \text{fcns}^{-1}(w1) \overset{\text{def}}{=} \text{fcns}^{-1}(w)+1 \tag{4.4}$$

for all w in $\varepsilon \cup 0\{0,1\}^*$, under the understanding that $(wi)+1 = w(i+1)$ for all w in $\mathbb{N}^*$ and $i \in \mathbb{N}$. Observe that binary trees t produced by this encoding verify $\text{dom}(t) \subseteq 0\{0,1\}^*$.

The tree t can be seen as a **binary structure** $\text{fcns}(\mathfrak{M}) = \langle \text{dom}(t), \downarrow_0, \downarrow_1, (P_a)_{a\in A}\rangle$, defined by

$$\downarrow_0 \overset{\text{def}}{=} \{(w, w0) \mid w0 \in \text{dom}(t)\} \tag{4.5}$$

$$\downarrow_1 \overset{\text{def}}{=} \{(w, w1) \mid w1 \in \text{dom}(t)\} \tag{4.6}$$

$$P_a \overset{\text{def}}{=} \{w \in \text{dom}(t) \mid a \in t(w)\}\ . \tag{4.7}$$

The domains of our constructed binary trees are not necessarily predecessor-closed, which can be annoying. Let # be a fresh symbol not in A; given t a labelled binary tree, its **closure** $\bar{t}$ is the tree with domain

$$\text{dom}(\bar{t}) \overset{\text{def}}{=} \{\varepsilon, 1\} \cup \{0w \mid w \in \text{dom}(t)\} \cup \{0wi \mid w \in \text{dom}(t) \wedge i \in \{0,1\}\} \tag{4.8}$$

and labels

$$\bar{t}(w) \overset{\text{def}}{=} \begin{cases} t(w') & \text{if } w = 0w' \wedge w' \in \text{dom}(t) \\ \{\#\} & \text{otherwise.} \end{cases} \tag{4.9}$$

Note that in $\bar{t}$, every node is either a node not labelled by # with exactly two children, or a #-labelled leaf with no children, or a #-labelled root with two children, thus $\bar{t}$ is a *full* (aka *strict*) binary tree.

4.1 Monadic Second-Order Logic

See Comon et al. *(2007, Section 8.4).*

We consider the **weak monadic second-order logic** (wMSO), over tree structures $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$ and two infinite countable sets of first-order variables $\mathcal{X}_1$ and second-order variables $\mathcal{X}_2$. Its syntax is defined by

$$\psi ::= x = y \mid x \in X \mid x \downarrow y \mid x \rightarrow y \mid P_a(x) \mid \neg\psi \mid \psi \vee \psi \mid \exists x.\psi \mid \exists X.\psi$$

where x, y range over $\mathcal{X}_1$, X over $\mathcal{X}_2$, and a over A. We write $\text{FV}(\psi)$ for the set of variables free in a formula ψ; a formula without free variables is called a **sentence**.

First-order variables are interpreted as nodes in W, while second-order variables are interpreted as *finite* subsets of W (it would otherwise be the full second-order logic). Let $\nu : \mathcal{X}_1 \to W$ and $\mu : \mathcal{X}_2 \to \mathcal{P}_f(W)$ be two corresponding assignments; then the satisfaction relation is defined by

$$\begin{array}{ll}
\mathfrak{M} \models_{\nu,\mu} x = y & \text{if } \nu(x) = \nu(y) \\
\mathfrak{M} \models_{\nu,\mu} x \in X & \text{if } \nu(x) \in \mu(X) \\
\mathfrak{M} \models_{\nu,\mu} x \downarrow y & \text{if } \nu(x) \downarrow \nu(y) \\
\mathfrak{M} \models_{\nu,\mu} x \to y & \text{if } \nu(x) \to \nu(y) \\
\mathfrak{M} \models_{\nu,\mu} P_a(x) & \text{if } P_a(\nu(x)) \\
\mathfrak{M} \models_{\nu,\mu} \neg\psi & \text{if } \mathfrak{M} \not\models_{\nu,\mu} \psi \\
\mathfrak{M} \models_{\nu,\mu} \psi \vee \psi' & \text{if } \mathfrak{M} \models_{\nu,\mu} \psi \text{ or } \mathfrak{M} \models_{\nu,\mu} \psi' \\
\mathfrak{M} \models_{\nu,\mu} \exists x.\psi & \text{if } \exists w \in W, \mathfrak{M} \models_{\nu\{x \leftarrow w\},\mu} \psi \\
\mathfrak{M} \models_{\nu,\mu} \exists X.\psi & \text{if } \exists U \subseteq W, U \text{ finite} \wedge \mathfrak{M} \models_{\nu,\mu\{X \leftarrow U\}} \psi \ .
\end{array}$$

As usual, we define conjunctions as $\psi \wedge \psi' \overset{\text{def}}{=} \neg(\neg\psi \vee \neg\psi')$, implications as $\psi \supset \psi' \overset{\text{def}}{=} \neg\psi \vee \psi'$, and equivalences as $\psi \equiv \psi' \overset{\text{def}}{=} \psi \supset \psi' \wedge \psi' \supset \psi$.

Given a wMSO formula ψ, we are interested in two algorithmic problems: the **satisfiability** problem, which asks whether there exist $\mathfrak{M}$ and ν and μ s.t. $\mathfrak{M} \models_{\nu,\mu} \psi$, and the **model-checking** problem, which given $\mathfrak{M}$ asks whether there exist ν and μ s.t. $\mathfrak{M} \models_{\nu,\mu} \psi$. By modifying the vocabulary to have labels in $A \uplus \mathrm{FV}(\psi)$, these questions can be rephrased on a wMSO *sentence* ψ':

$$\begin{aligned}
\psi' \overset{\text{def}}{=} \exists \mathrm{FV}(\psi).\psi &\wedge \left(\bigwedge_{x \in \mathcal{X}_1 \cap \mathrm{FV}(\psi)} P_x(x) \wedge \forall y. x \neq y \supset \neg P_x(y) \right) \\
&\wedge \left(\bigwedge_{X \in \mathcal{X}_2 \cap \mathrm{FV}(\psi)} \forall y. y \in X \equiv P_X(y) \right) .
\end{aligned}$$

In practical applications of model-theoretic techniques we restrict ourselves to *finite* models for these questions.

Example 4.1. Here are a few useful wMSO formulæ: To allow any label in a finite set $B \subseteq A$:

$$\begin{aligned}
P_B(x) &\overset{\text{def}}{=} \bigvee_{a \in B} P_a(x) \\
P_B(X) &\overset{\text{def}}{=} \forall x. x \in X \supset P_B(x) \ .
\end{aligned}$$

To check whether we are at the root or a leaf or similar constraints:

$$\begin{aligned}
\mathrm{root}(x) &\overset{\text{def}}{=} \neg\exists y. y \downarrow x \\
\mathrm{leaf}(x) &\overset{\text{def}}{=} \neg\exists y. x \downarrow y \\
\mathrm{internal}(x) &\overset{\text{def}}{=} \neg\mathrm{leaf}(x) \\
\mathrm{children}(x, X) &\overset{\text{def}}{=} \forall y. y \in X \equiv x \downarrow y \\
x \downarrow_0 y &\overset{\text{def}}{=} x \downarrow y \wedge \neg\exists z. z \to y \ .
\end{aligned}$$

To use the **monadic transitive closure** of a formula $\psi(u,v)$ with $u,v \in \mathrm{FV}(\psi)$: such a formula $\psi(u,v)$ defines a binary relation over the model, and $[\mathrm{TC}_{u,v}\ \psi(u,v)]$ then defines the transitive reflexive closure of the relation:

$$x\ [\mathrm{TC}_{u,v}\ \psi(u,v)]\ y \overset{\text{def}}{=} \forall X.(x \in X \land \forall uv.(u \in X \land \psi(u,v) \supset v \in X) \supset y \in X) \tag{4.10}$$

For example,

$$\begin{aligned} x \downarrow^\star y &\overset{\text{def}}{=} x\ [\mathrm{TC}_{u,v}\ u \downarrow v]\ y \\ x \rightarrow^\star y &\overset{\text{def}}{=} x\ [\mathrm{TC}_{u,v}\ u \rightarrow v]\ y\ . \end{aligned}$$

4.1.1 Linguistic Analyses in wMSO

Let us illustrate how we can work out a constituent-based analysis using wMSO. Following the ideas on grammaticality expressed at the beginning of the chapter, we define large conjunctions of formulæ expressing various linguistic constraints.

See Rogers (1998) for a complete analysis using wMSO. Monadic second-order logic can also be applied to queries *in treebanks (Kepser, 2004; Maryns and Kepser, 2009).*

Basic Grammatical Labels Let us fix two disjoint finite sets N of grammatical categories and Θ of part-of-speech tags and distinguish a particular category $S \in N$ standing for sentences, and let $N \uplus \Theta \subseteq A$ (we do not assume A to be finite).

Define the formula

$$\mathrm{labels}_{N,\Theta} \overset{\text{def}}{=} \forall x.\mathrm{root}(x) \supset P_S(x)\ , \tag{4.11}$$

which forces the root label to be S;

$$\land\ \forall x.\mathrm{internal}(x) \supset \bigvee_{a \in N \uplus \Theta} P_a(x) \land \bigwedge_{b \in N \uplus \Theta \setminus \{a\}} \neg P_b(x) \tag{4.12}$$

checks that every internal node has exactly one label from $N \uplus \Theta$ (plus potentially others from $A \setminus (N \uplus \Theta)$);

$$\land\ \forall x.\mathrm{leaf}(x) \supset \neg P_{N \uplus \Theta}(x) \tag{4.13}$$

forbids grammatical labels on leaves;

$$\land\ \forall y.\mathrm{leaf}(y) \supset \exists x.x \downarrow y \land P_\Theta(x) \tag{4.14}$$

expresses that leaves should have POS-labelled parents;

$$\land\ \forall x.\exists y_0 y_1 y_2.x \downarrow^\star y_0 \land y_0 \downarrow y_1 \land y_1 \downarrow y_2 \land \mathrm{leaf}(y_2) \supset P_N(x) \tag{4.15}$$

verifies that internal nodes at distance at least two from some leaf should have labels drawn from N, and are thus not POS-labelled by (4.12), and thus cannot have a leaf as a child by (4.13);

$$\land\ \forall x.P_\Theta(x) \supset \neg\exists yz.y \neq z \land x \downarrow y \land x \downarrow z \tag{4.16}$$

discards trees where POS-labelled nodes have more than one child. The purpose of $\mathrm{labels}_{N,\Theta}$ is to restrict the possible models to trees with the particular shape we use in constituent-based analyses.

Open Lexicon Let us assume that some finite part of the lexicon is known, as well as possible POS tags for each known word. One way to express this in an open-ended manner is to define a finite set $L \subseteq A$ disjoint from N and Θ, and a relation $\mathrm{pos} \subseteq L \times \Theta$. Then the formula

$$\mathrm{lexicon}_{L,\mathrm{pos}} \overset{\text{def}}{=} \forall x. \bigvee_{\ell \in L} \left(P_\ell(x) \supset \mathrm{leaf}(x) \land \bigwedge_{\ell' \in L \setminus \{\ell\}} \neg P_{\ell'}(x) \land \forall y.y \downarrow x \supset P_{\mathrm{pos}(\ell)}(y) \right) \tag{4.17}$$

makes sure that only leaves can be labelled by words, and that when a word is known (i.e. if it appears in L), it should have one of its allowed POS tag as immediate parent. If the current POS tagging information of our lexicon is incomplete, then this particular constraint will not be satisfied. For an unknown word however, any POS tag can be used.

Context-Free Constraints It is of course easy to enforce some local constraints in trees. For instance, assume we are given a CFG $\mathcal{G} = \langle N, \Theta, P, S\rangle$ describing the "usual" local constraints between grammatical categories and POS tags. Assume ε belongs to A; then the formula

$$\mathrm{grammar}_{\mathcal{G}} \overset{\mathrm{def}}{=} \forall x.(P_\varepsilon(x) \supset \neg P_{N \uplus \Theta \uplus L}(x)) \wedge \bigvee_{B \in N} P_B(x) \supset \bigvee_{B \to \beta \in P} \exists y.x \downarrow_0 y \wedge \mathrm{rule}_\beta(y) \tag{4.18}$$

forces the tree to comply with the rules of the grammar, where

$$\begin{aligned} \mathrm{rule}_{X\beta}(x) &\overset{\mathrm{def}}{=} P_X(x) \wedge \exists y.x \to y \wedge \mathrm{rule}_\beta(y) && (\text{for } \beta \neq \varepsilon \text{ and } X \in N \uplus \Theta)\\ \mathrm{rule}_X(x) &\overset{\mathrm{def}}{=} P_X(x) \wedge \neg\exists y.x \to y && (\text{for } X \in N \uplus \Theta)\\ \mathrm{rule}_\varepsilon(x) &\overset{\mathrm{def}}{=} P_\varepsilon(x) \wedge \mathrm{leaf}(x)\ . \end{aligned}$$

Again, the idea is to provide a rather permissive set of local constraints, and to be able to spot the cases where these constraints are not satisfied.

Non-Local Dependencies Implementing local constraints as provided by a CFG is however far from ideal. A much more interesting approach would be to take advantage of the ability to use long-distance constraints, and to model subcategorisation frames and modifiers.

The following examples also show that some of the typical **features** used for training statistical models can be formally expressed using wMSO. This means that treebank annotations can be computed very efficiently once a tree automaton has been computed for the wMSO formulæ, in time linear in the size of the treebank.

Head Percolation. The first step is to find which child is the **head** among its siblings; several heuristics have been developed to this end, and a simple way to describe such heuristics is to use a **head percolation** function $h : N \to \{l, r\} \times (N \uplus \Theta)^*$ that describes for a given parent label A a list of potential labels $X_1, \ldots, X_n$ in $N \uplus \Theta$ in order of priority and a direction $d \in \{l, r\}$ standing for "leftmost" or "rightmost": such a value means that the leftmost (resp. rightmost) occurrence of X_1 is the head, this unless X_1 is not among the children, in which case we should try X_2 and so on, and if X_n also fails simply choose the leftmost (resp. rightmost) child (see e.g. Collins, 1999, Appendix A). For instance, the function

$$\begin{aligned} h(\mathrm{S}) &= (r, \mathrm{TO\ IN\ VP\ S\ SBAR} \cdots)\\ h(\mathrm{VP}) &= (l, \mathrm{VBD\ VBN\ VBZ\ VB\ VBG\ VP} \cdots)\\ h(\mathrm{NP}) &= (r, \mathrm{NN\ NNP\ NNS\ NNPS\ JJR\ CD} \cdots)\\ h(\mathrm{PP}) &= (l, \mathrm{IN\ TO\ VBG\ VBN} \cdots) \end{aligned}$$

would result in the correct head annotations in Figure 6.1.

Given such a head percolation function h, we can express the fact that a given node is a head:

$$\mathrm{head}(x) \overset{\mathrm{def}}{=} \mathrm{leaf}(x) \vee \bigvee_{B \in N} \exists y Y . y \downarrow x \wedge \mathrm{children}(y, Y) \wedge P_B(y) \wedge \mathrm{head}_{h(B)}(x, Y) \tag{4.19}$$

$$\begin{aligned}
\mathrm{head}_{d,X\beta}(x, Y) &\overset{\mathrm{def}}{=} \neg\mathrm{priority}_{d,X}(x, Y) \supset (\mathrm{head}_{d,\beta}(x, Y) \wedge \neg P_X(Y))\\
\mathrm{head}_{l,\varepsilon}(x, Y) &\overset{\mathrm{def}}{=} \forall y . y \in Y \supset x \to^{\star} y\\
\mathrm{head}_{r,\varepsilon}(x, Y) &\overset{\mathrm{def}}{=} \forall y . y \in Y \supset y \to^{\star} x\\
\mathrm{priority}_{l,X}(x, Y) &\overset{\mathrm{def}}{=} P_X(x) \wedge \forall y . y \in Y \wedge y \to^{\star} x \supset \neg P_X(y)\\
\mathrm{priority}_{r,X}(x, Y) &\overset{\mathrm{def}}{=} P_X(x) \wedge \forall y . y \in Y \wedge x \to^{\star} y \supset \neg P_X(y) \;.
\end{aligned}$$

where β is a sequence in $(N \uplus \Theta)^*$ and X a symbol in $N \uplus \Theta$.

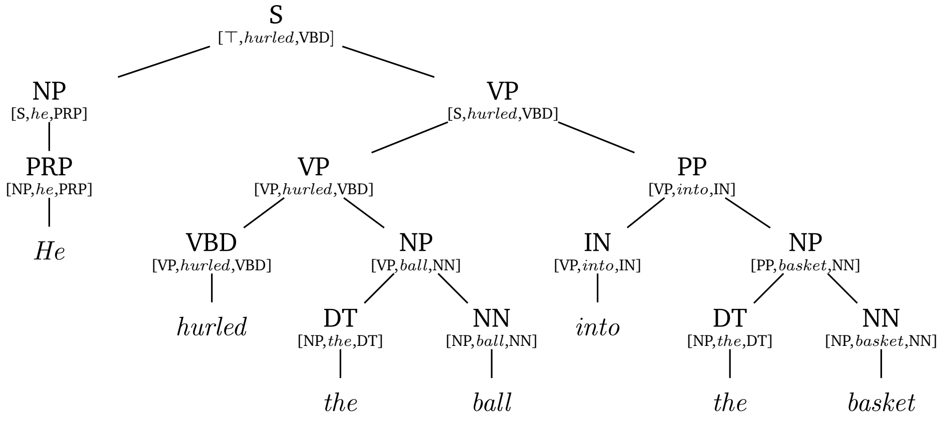


Figure 4.1: A derivation tree refined with lexical and parent information.

Lexicalisation. Using head information, we can also recover lexicalisation information:

$$\mathrm{lexicalise}(x, y) \overset{\mathrm{def}}{=} \mathrm{leaf}(y) \wedge x \, [\mathrm{TC}_{u,v} \; u \downarrow v \wedge \mathrm{head}(v)] \; y \;. \tag{4.20}$$

This formula recovers the lexical information in Figure 6.1.

Exercise 4.1. Propose wMSO formulæ to recover the parent and lexical POS information in constituent trees, as illustrated in Figure 6.1. **(∗)**

Modifiers. Here is a first use of wMSO to *extract* information about a proposed constituent tree: try to find which word is modified by another word. For instance, for an adverb we could write something like

$$\begin{aligned}
\mathrm{modify}_{\mathrm{RB}}(x, y) \overset{\mathrm{def}}{=} \exists x' y' z . z \downarrow x \wedge P_{\mathrm{RB}}(z) \wedge \mathrm{lexicalise}(x', x) \wedge y' \downarrow x'\\
\wedge \neg\mathrm{lexicalise}(y', x) \wedge \mathrm{lexicalise}(y', y)
\end{aligned} \tag{4.21}$$

that finds a maximal head x' and the lexical projection of its parent y'. This formula finds for instance that *really* modifies *likes* in Figure 4.2.

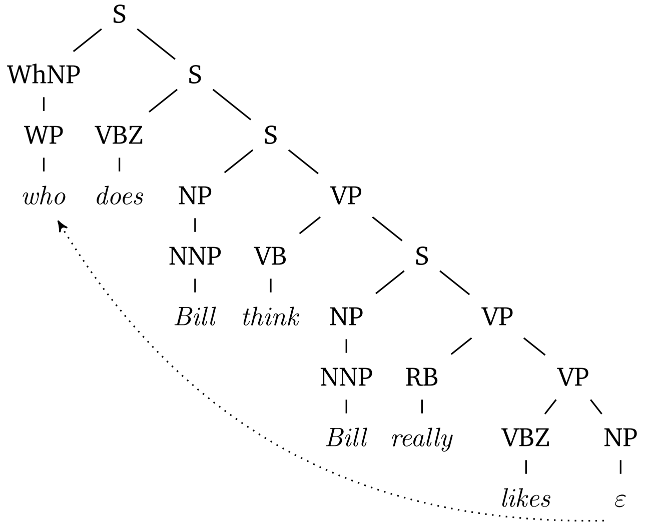


Figure 4.2: Derivation tree for *Who does Bill think Bill really likes?*

($*$) **Exercise 4.2.** Modify (4.21) to make sure that any leaf with a parent tagged by the POS RB modifies either a verb or an adjective.

($**$) **Exercise 4.3.** Consider the ε node in Figure 4.2: modify (4.20) to recover that *who* lexicalises the bottommost NP node.

4.1.2 wS2S

See (Doner, 1970; Thatcher and Wright, 1968; Rabin, 1969; Meyer, 1975) for classical results on wS2S, and more recently (Rogers, 1996, 2003) for linguistic applications.

The classical logics for trees do not use the vocabulary of tree structures $\mathfrak{M}$, but rather that of binary structures $\langle \mathrm{dom}(t), \downarrow_0, \downarrow_1, (P_a)_{a \in A}\rangle$. The weak monadic second-order logic over this vocabulary is called the weak monadic second-order logic of **two successors** (wS2S). The semantics of wS2S should be clear.

The interest of considering wS2S at this point is that it is well-known to have a decidable satisfiability problem, and that for any wS2S sentence ψ one can construct a tree automaton $\mathcal{A}_\psi$—with tower($|\psi|$) as size—that recognises all the finite models of ψ. More precisely, when working with finite binary trees and closed formulæ ψ,

See Comon et al. *(2007, Section 3.3)—their construction is easily extended to handle labelled trees. Using automata over infinite trees, these can also be handled (Rabin, 1969; Weyer, 2002).*

$$L(\mathcal{A}_\psi) = \{\bar{t} \in T(\Sigma \uplus \{\{\#\}\}) \mid t \text{ finite} \wedge t \models \psi\} \; . \tag{4.22}$$

Now, it is easy to translate any wMSO sentence ψ into a wS2S sentence ψ' s.t. $\mathfrak{M} \models \psi$ iff $\mathrm{fcns}(\mathfrak{M}) \models \psi'$. This formula simply has to *interpret* the $\downarrow$ and $\rightarrow$ relations into their binary encodings: let

$$\psi' \overset{\text{def}}{=} \psi \wedge \exists x. \neg(\exists z. z \downarrow_0 x \vee z \downarrow_1 x) \wedge \neg(\exists y. x \downarrow_1 y) \tag{4.23}$$

where the conditions on x ensure it is at the root and does not have any right child, and where ψ uses the macros

$$x \downarrow y \overset{\text{def}}{=} \exists x_0 . x \downarrow_0 x_0 \wedge (x_0 \, [\mathrm{TC}_{u,v} \, u \downarrow_1 v] \, y) \tag{4.24}$$

$$x \rightarrow y \overset{\text{def}}{=} x \downarrow_1 y \; . \tag{4.25}$$

The conclusion of this construction is

Theorem 4.2. *Satisfiability and model-checking for wMSO are decidable.*

Exercise 4.4 (ω Successors)**.** Show that the weak second-order logic of ω successors (**wS$\boldsymbol{\omega}$S**), i.e. with $\downarrow_i \overset{\text{def}}{=} \{(w, wi) \mid wi \in W\}$ defined for every $i \in \mathbb{N}$, has decidable satisfiability and model-checking problems. (∗)

4.2 Propositional Dynamic Logic

An alternative take on model-theoretic syntax is to employ **modal logics** on tree structures. Several properties of modal logics make them interesting to this end: their decision problems are usually considerably simpler, and they allow to express rather naturally how to hop from one point of interest to another.

Propositional dynamic logic (Fischer and Ladner, 1979) is a two-sorted modal logic where the basic relations can be composed using regular operations: on tree structures $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$, its terms follow the abstract syntax

$$\pi ::= \downarrow \mid \rightarrow \mid \pi^{-1} \mid \pi;\pi \mid \pi+\pi \mid \pi^* \mid \varphi? \qquad \text{(path formulæ)}$$
$$\varphi ::= a \mid \top \mid \neg\varphi \mid \varphi\vee\varphi \mid \langle\pi\rangle\varphi \qquad \text{(node formulæ)}$$

where a ranges over A.

Propositional dynamic logic on ordered trees was first defined by Kracht (1995). The name of PDL on trees *is due to Afanasiev* et al. *(2005); this logic is also known as* ***Regular XPath*** *in the XML processing community (Marx, 2005). Various fragments have been considered through the years; see for instance Blackburn* et al. *(1993, 1996); Palm (1999); Marx and de Rijke (2005).*

The **semantics** of a node formula on a tree structure $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$ is a set of tree nodes $[\![\varphi]\!] = \{w \in W \mid \mathfrak{M}, w \models \varphi\}$, while the semantics of a path formula is a binary relation over W:

$$\begin{aligned}
[\![a]\!] &\overset{\text{def}}{=} \{w\in W \mid P_a(w)\} & [\![\downarrow]\!] &\overset{\text{def}}{=} \downarrow\\
[\![\top]\!] &\overset{\text{def}}{=} W & [\![\rightarrow]\!] &\overset{\text{def}}{=} \rightarrow\\
[\![\neg\varphi]\!] &\overset{\text{def}}{=} W\backslash[\![\varphi]\!] & [\![\pi^{-1}]\!] &\overset{\text{def}}{=} [\![\pi]\!]^{-1}\\
[\![\varphi_1\vee\varphi_2]\!] &\overset{\text{def}}{=} [\![\varphi_1]\!]\cup[\![\varphi_2]\!] & [\![\pi_1;\pi_2]\!] &\overset{\text{def}}{=} [\![\pi_1]\!] \mathbin{⨾} [\![\pi_2]\!]\\
[\![\langle\pi\rangle\varphi]\!] &\overset{\text{def}}{=} [\![\pi]\!]^{-1}([\![\varphi]\!]) & [\![\pi_1+\pi_2]\!] &\overset{\text{def}}{=} [\![\pi_1]\!]\cup[\![\pi_2]\!]\\
& & [\![\pi^*]\!] &\overset{\text{def}}{=} [\![\pi]\!]^\star\\
& & [\![\varphi?]\!] &\overset{\text{def}}{=} \mathrm{Id}_{[\![\varphi]\!]}\;.
\end{aligned}$$

Finally, a tree $\mathfrak{M}$ is a **model** for a PDL formula φ if its root is in $[\![\varphi]\!]$, written $\mathfrak{M}, \text{root} \models \varphi$.

We define the classical dual operators

$$\bot \overset{\text{def}}{=} \neg\top \qquad \varphi_1\wedge\varphi_2 \overset{\text{def}}{=} \neg(\neg\varphi_1\vee\neg\varphi_2) \qquad [\pi]\varphi \overset{\text{def}}{=} \neg\langle\pi\rangle\neg\varphi\;. \tag{4.26}$$

We also define

$$\begin{aligned}
\uparrow &\overset{\text{def}}{=} \downarrow^{-1} & \leftarrow &\overset{\text{def}}{=} \rightarrow^{-1}\\
\text{root} &\overset{\text{def}}{=} [\uparrow]\bot & \text{leaf} &\overset{\text{def}}{=} [\downarrow]\bot\\
\text{first} &\overset{\text{def}}{=} [\leftarrow]\bot & \text{last} &\overset{\text{def}}{=} [\rightarrow]\bot\;.
\end{aligned}$$

Exercise 4.5 (Converses)**.** Prove the following equivalences: (∗)

$$(\pi_1;\pi_2)^{-1} \equiv \pi_2^{-1};\pi_1^{-1} \tag{4.27}$$
$$(\pi_1+\pi_2)^{-1} \equiv \pi_1^{-1}+\pi_2^{-1} \tag{4.28}$$
$$(\pi^*)^{-1} \equiv (\pi^{-1})^* \tag{4.29}$$
$$(\varphi?)^{-1} \equiv \varphi?\;. \tag{4.30}$$

(∗) **Exercise 4.6** (Reductions)**.** Prove the following equivalences:

$$\langle \pi_1 ; \pi_2 \rangle \varphi \equiv \langle \pi_1 \rangle \langle \pi_2 \rangle \varphi \tag{4.31}$$

$$\langle \pi_1 + \pi_2 \rangle \varphi \equiv (\langle \pi_1 \rangle \varphi) \vee (\langle \pi_2 \rangle \varphi) \tag{4.32}$$

$$\langle \pi^* \rangle \varphi \equiv \varphi \vee \langle \pi ; \pi^* \rangle \varphi \tag{4.33}$$

$$\langle \varphi_1 ? \rangle \varphi_2 \equiv \varphi_1 \wedge \varphi_2 \ . \tag{4.34}$$

4.2.1 Model-Checking

As with MSO, the main application of PDL on trees is to query treebanks (see e.g. Lai and Bird, 2010).

The model-checking problem for PDL is rather easy to decide. Given a model $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_p)_{p \in A} \rangle$, we can compute inductively the satisfaction sets and relations using standard algorithms. This is a P algorithm.

4.2.2 Satisfiability

See also (Blackburn et al.*, 2001, Section 6.8) for a reduction from a tiling problem and (Harel* et al.*, 2000, Chapter 8) for a reduction from alternating Turing machines.*

Unlike the model-checking problem, the satisfiability problem for PDL is rather demanding: it is ExpTime-complete.

Theorem 4.3 (Fischer and Ladner, 1979)**.** *Satisfiability for PDL is* ExpTime*-hard.*

As with wMSO, it is more convenient to work on binary trees t of the form $\langle \mathrm{dom}(t), \downarrow_0, \downarrow_1, (P_a)_{a \in A \uplus \{0,1\}} \rangle$ that encode our tree structures. Compared with the wMSO case, we add two atomic predicates 0 and 1 that hold on left and right children respectively. The syntax of PDL over such models simply replaces $\downarrow$ and $\rightarrow$ by $\downarrow_0$ and $\downarrow_1$; as with wMSO in Section 4.1.2 we can *interpret* these relations in PDL by

$$\downarrow \stackrel{\text{def}}{=} \downarrow_0 ; \downarrow_1^* \qquad\qquad \rightarrow \stackrel{\text{def}}{=} \downarrow_1 \tag{4.35}$$

and translate any PDL formula φ into a formula

$$\varphi' \stackrel{\text{def}}{=} \varphi \wedge ([\uparrow^* ; \downarrow^* ; \downarrow_0] 0 \wedge \neg 1) \wedge ([\uparrow^* ; \downarrow^* ; \downarrow_1] 1 \wedge \neg 0) \wedge [\uparrow^* ; \mathrm{root}? ; \downarrow_1] \bot \tag{4.36}$$

that checks that φ holds, that the 0 and 1 labels are correct, and verifies $\mathfrak{M}, w \models \varphi$ iff $\mathrm{fcns}(\mathfrak{M}), \mathrm{fcns}(w) \models \varphi'$. The conditions in (4.36) ensure that the tree we are considering is the image of some tree structure by fcns: we first go back to the root by the path $\uparrow^* ; \mathrm{root}?$, and then verify that the root does not have a right child.

Normal Form. Let us write

$$\uparrow_0 \stackrel{\text{def}}{=} \downarrow_0^{-1} \qquad\qquad \uparrow_1 \stackrel{\text{def}}{=} \downarrow_1^{-1} \ ;$$

then using the equivalences of Exercise 4.5 we can reason on PDL with a restricted path syntax

$$\alpha ::= \downarrow_0 \mid \uparrow_0 \mid \downarrow_1 \mid \uparrow_1 \qquad \text{(atomic relations)}$$

$$\pi ::= \alpha \mid \pi ; \pi \mid \pi + \pi \mid \pi^* \mid \varphi ? \qquad \text{(path formulæ)}$$

and using the dualities of (4.26), we can restrict node formulæ to be of form

$$\varphi ::= a \mid \neg a \mid \top \mid \bot \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle \pi \rangle \varphi \mid [\pi] \varphi \ . \qquad \text{(node formulæ)}$$

Lemma 4.4. *For any PDL formula φ, we can construct an equivalent formula φ' in normal form with $|\varphi'| = O(|\varphi|)$.*

Proof sketch. The normal form is obtained by "pushing" negations and converses as far towards the leaves as possible, and can result in the worst-case in doubling the size of φ due to the extra $\neg$ and $^{-1}$ at the leaves. □

Fisher-Ladner Closure

The equivalences found in Exercise 4.6 and their duals allow to simplify PDL formulæ into a reduced normal form we will soon see, which is a form of disjunctive normal form with atomic propositions and atomic modalities for literals. In order to obtain algorithmic complexity results, it will be important to be able to bound the number of possible such literals, which we do now.

The **Fisher-Ladner closure** of a PDL formula in normal form φ is the smallest set S of formulæ in normal form s.t.

1. $\varphi \in S$,
2. if $\varphi_1 \vee \varphi_2 \in S$ or $\varphi_1 \wedge \varphi_2 \in S$ then $\varphi_1 \in S$ and $\varphi_2 \in S$,
3. if $\langle\pi\rangle\varphi' \in S$ or $[\pi]\varphi' \in S$ then $\varphi' \in S$,
4. if $\langle\pi_1;\pi_2\rangle\varphi' \in S$ then $\langle\pi_1\rangle\langle\pi_2\rangle\varphi' \in S$,
5. if $[\pi_1;\pi_2]\varphi' \in S$ then $[\pi_1][\pi_2]\varphi' \in S$,
6. if $\langle\pi_1+\pi_2\rangle\varphi' \in S$ then $\langle\pi_1\rangle\varphi' \in S$ and $\langle\pi_2\rangle\varphi' \in S$,
7. if $[\pi_1+\pi_2]\varphi' \in S$ then $[\pi_1]\varphi' \in S$ and $[\pi_2]\varphi' \in S$,
8. if $\langle\pi^*\rangle\varphi' \in S$ then $\langle\pi\rangle\langle\pi^*\rangle\varphi' \in S$,
9. if $[\pi^*]\varphi' \in S$ then $[\pi][\pi^*]\varphi' \in S$,
10. if $\langle\varphi_1?\rangle\varphi_2 \in S$ or $[\varphi_1?]\varphi_2 \in S$ then $\varphi_1 \in S$.

We write $\mathrm{FL}(\varphi)$ for the Fisher-Ladner closure of φ.

Lemma 4.5. *Let φ be a PDL formula in normal form. Its Fisher-Ladner closure is of size $|\mathrm{FL}(\varphi)| \leq |\varphi|$.*

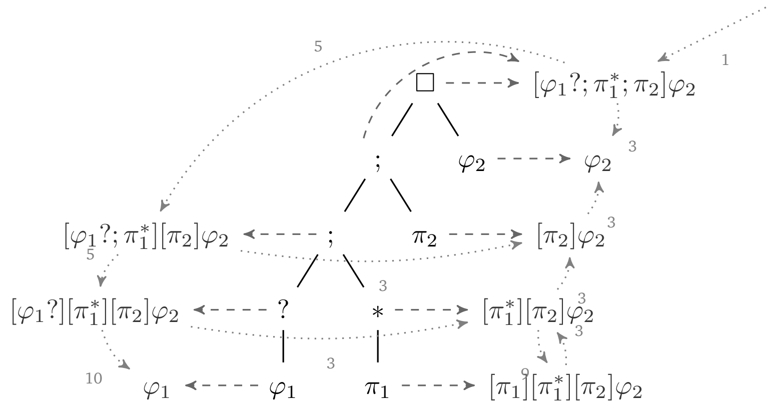

5
1
□
$[\varphi_1?;\pi_1^*;\pi_2]\varphi_2$
;
φ_2
φ_2
3
$[\varphi_1?;\pi_1^*][\pi_2]\varphi_2$
;
π_2
$[\pi_2]\varphi_2$
3
5
3
$[\varphi_1?][\pi_1^*][\pi_2]\varphi_2$
?
$*$
$[\pi_1^*][\pi_2]\varphi_2$
3
3
3
10
φ_1
φ_1
π_1
9
$[\pi_1][\pi_1^*][\pi_2]\varphi_2$

Figure 4.3: The surjection σ from positions in $\varphi \overset{\text{def}}{=} [\varphi_1?;\pi_1^*;\pi_2]\varphi_2$ to $\mathrm{FL}(\varphi)$ (dashed), and the rules used to construct $\mathrm{FL}(\varphi)$ (dotted).

Proof. We construct a surjection σ between positions p in the term φ and the formulæ in S:

- for positions p spanning a node subformula $\mathrm{span}(p) = \varphi_1$, we can map to φ_1 (this corresponds to cases 1—3 and 10 on subformulæ of φ');

- for positions p spanning a path subformula $\mathrm{span}(p) = \pi$, we find the closest ancestor spanning a node subformula (thus of form $\langle\pi'\rangle\varphi_1$ or $[\pi']\varphi_1$). If $\pi = \pi'$ we map p to the same $\langle\pi'\rangle\varphi_1$ or $[\pi']\varphi_1$. Otherwise we consider the parent position p' of p, which is mapped to some formula $\sigma(p')$, and distinguish several cases:
  - for $\sigma(p') = \langle\pi_1;\pi_2\rangle\varphi_2$ we map p to $\langle\pi_1\rangle\langle\pi_2\rangle\varphi_2$ if $\mathrm{span}(p) = \pi_1$ and to $\langle\pi_2\rangle\varphi_2$ if $\mathrm{span}(p) = \pi_2$ (this matches case 4 and the further application of 3);
  - for $\sigma(p') = [\pi_1;\pi_2]\varphi_2$ we map p to $[\pi_1][\pi_2]\varphi_2$ if $\mathrm{span}(p) = \pi_1$ and to $[\pi_2]\varphi_2$ if $\mathrm{span}(p) = \pi_2$ (this matches case 5 and the further application of 3);
  - for $\sigma(p') = \langle\pi_1 + \pi_2\rangle\varphi_2$ and $\mathrm{span}(p) = \pi_i$ with $i \in \{1, 2\}$, we map p to $\langle\pi_i\rangle\varphi_2$ (this matches case 6);
  - for $\sigma(p') = [\pi_1 + \pi_2]\varphi$ and $\mathrm{span}(p) = \pi_i$ with $i \in \{1, 2\}$, we map p to $[\pi_i]\varphi_2$ (this matches case 7);
  - for $\sigma(p') = \langle\pi^*\rangle\varphi_2$, $\mathrm{span}(p) = \pi$ and we map p to $\langle\pi\rangle\langle\pi^*\rangle\varphi_2$ (this matches case 8);
  - for $\sigma(p') = [\pi^*]\varphi_2$, $\mathrm{span}(p) = \pi$ and we map p to $[\pi][\pi^*]\varphi_2$ (this matches case 9).

The function σ we just defined is indeed surjective: we have covered every formula produced by every rule. Figure 4.3 presents an example term and its mapping. □

Reduced Formulæ

Reduced Normal Form. We try now to reduce formulæ into a form where any modal subformula is under the scope of some atomic modality $\langle\alpha\rangle$ or $[\alpha]$. Given a formula φ in normal form, this is obtained by using the equivalences of Exercise 4.6 and their duals, and by putting the formula into disjunctive normal form, i.e.

$$\varphi \equiv \bigvee_i \bigwedge_j \chi_{i,j} \tag{4.37}$$

where each $\chi_{i,j}$ is of form

$$\chi ::= a \mid \neg a \mid \langle\alpha\rangle\varphi' \mid [\alpha]\varphi' \ . \qquad \text{(reduced formulæ)}$$

Observe that all the equivalences we used can be found among the rules of the Fisher-Ladner closure of φ:

Lemma 4.6. *Given a PDL formula φ in normal form, we can construct an equivalent formula $\bigvee_i \bigwedge_j \chi_{i,j}$ where each $\chi_{i,j}$ is a reduced formula in $\mathrm{FL}(\varphi)$.*

Two-Way Alternating Tree Automaton

The presentation follows mostly Calvanese et al. *(2009).*

We finally turn to the construction of a tree automaton that recognises the models of a normal form formula φ. To simplify matters, we use a powerful model for this automaton: a **two-way alternating tree automaton** (2ATA) over finite ranked trees.

Definition 4.7. A **two-way alternating tree automaton** (2ATA) is a tuple $\mathcal{A} = \langle Q, \Sigma, q_i, F, \delta\rangle$ where Q is a finite set of states, Σ is a ranked alphabet with maximal rank k, $q_i \in Q$ is the initial state, and δ is a transition function from pairs of states and symbols (q, a) in $Q \times \Sigma$ to *positive Boolean formulæ* f in $\mathcal{B}_+(\{-1, \dots, k\} \times Q)$, defined by the abstract syntax

$$f ::= (d, q) \mid f \vee f \mid f \wedge f \mid \top \mid \bot \,,$$

where d ranges over $\{-1, \dots, k\}$ and q over Q. For a set $J \subseteq \{-1, \dots, k\} \times Q$ and a formula f, we say that J *satisfies* f if assigning $\top$ to elements of J and $\bot$ to those in $\{-1, \dots, k\} \times Q \backslash J$ makes f true. A 2ATA is able to send copies of itself to a parent node (using the direction -1), to the same node (using direction 0), or to a child (using directions in $\{1, \dots, k\}$).

Given a labelled ranked ordered tree t over Σ, a **run** of $\mathcal{A}$ is a tree ρ labelled by $\mathrm{dom}(t) \times Q$ satisfying

1. ε is in $\mathrm{dom}(\rho)$ with $\rho(\varepsilon) = (\varepsilon, q_i)$,

2. if w is in $\mathrm{dom}(\rho)$, $\rho(w) = (u, q)$ and $\delta(q, t(u)) = f$, then there exists $J \subseteq \{-1, \dots, k\} \times Q$ of form $J = \{(d_0, q_0), \dots, (d_n, q_n)\}$ s.t. $J \models f$ and for all $0 \leq i \leq n$ we have

$$wi \in \mathrm{dom}(\rho) \qquad \rho(wi) = (u'_i, q_i) \qquad u'_i = \begin{cases} u(d_i - 1) & \text{if } d_i > 0 \\ u & \text{if } d_i = 0 \\ u' \text{ where } u = u'j & \text{otherwise} \end{cases}$$

with each $u'_i \in \mathrm{dom}(t)$.

A tree is accepted if there exists a run for it.

Theorem 4.8 (Vardi, 1998)**.** *Given a 2ATA $\mathcal{A} = \langle Q, \Sigma, q_i, F, \delta\rangle$, deciding the emptiness of $L(\mathcal{A})$ can be done in deterministic time $|\Sigma| \cdot 2^{O(k|Q|^3)}$.*

Automaton of a Formula Let φ be a formula in normal form. We want to construct a 2ATA $\mathcal{A}_\varphi = \langle Q, \Sigma, q_i, \delta\rangle$ that recognises exactly the closed models of φ, so that we can test the satisfiability of φ by Theorem 4.8. We assume wlog. that $A \subseteq \mathrm{Sub}(\varphi)$. We define

$$\begin{aligned} Q &\overset{\text{def}}{=} \mathrm{FL}(\varphi) \uplus \{q_i, q_\varphi, q_\#\} \\ \Sigma &\overset{\text{def}}{=} \{\#^{(0)}, \#^{(2)}\} \cup \{a^{(2)} \mid a \subseteq A \uplus \{0, 1\}\} \,. \end{aligned}$$

The transitions of $\mathcal{A}_\varphi$ are based on formula reductions. Let φ' be a formula in $\mathrm{FL}(\varphi)$ which is not reduced: then we can find an equivalent formula $\bigvee_i \bigwedge_j \chi_{i,j}$ where each $\chi_{i,j}$ is reduced. We define accordingly

$$\delta(\varphi', a) \overset{\text{def}}{=} \bigvee_i \bigwedge_j (0, \chi_{i,j})$$

for all such φ' and all $a \subseteq A$, thereby staying in place and checking the various $\chi_{i,j}$. For a reduced formula χ in $\mathrm{FL}(\varphi)$, we set for all $a \subseteq A \uplus \{0,1\}$

$$\delta(p,a) \stackrel{\text{def}}{=} \begin{cases} \top & \text{if } p \in a \\ \bot & \text{otherwise} \end{cases} \qquad \delta(\neg p,a) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } p \in a \\ \top & \text{otherwise} \end{cases}$$

$$\begin{aligned} \delta(\langle{\downarrow_0}\rangle\varphi',a) &\stackrel{\text{def}}{=} (1,\varphi') & \delta([{\downarrow_0}]\varphi',a) &\stackrel{\text{def}}{=} (1,\varphi') \vee (1,q_{\#}) \\ \delta(\langle{\downarrow_1}\rangle\varphi',a) &\stackrel{\text{def}}{=} (2,\varphi') & \delta([{\downarrow_1}]\varphi',a) &\stackrel{\text{def}}{=} (2,\varphi') \vee (2,q_{\#}) \\ \delta(\langle{\uparrow_0}\rangle\varphi',a) &\stackrel{\text{def}}{=} (-1,\varphi') \wedge (0,0) & \delta([{\uparrow_0}]\varphi',a) &\stackrel{\text{def}}{=} ((-1,\varphi') \wedge (0,0)) \vee (-1,q_{\#}) \vee (0,1) \\ \delta(\langle{\uparrow_1}\rangle\varphi',a) &\stackrel{\text{def}}{=} (-1,\varphi') \wedge (0,1) & \delta([{\uparrow_1}]\varphi',a) &\stackrel{\text{def}}{=} ((-1,\varphi') \wedge (0,1)) \vee (-1,q_{\#}) \vee (0,0) \end{aligned}$$

where the subformulæ 0 and 1 are used to check that the node we are coming from was a left or a right son and $q_{\#}$ checks that the node label is $\#$:

$$\delta(q_{\#},\#) \stackrel{\text{def}}{=} \top \qquad \delta(q_{\#},a) \stackrel{\text{def}}{=} \bot\ .$$

The initial state q_i checks that the root is labelled $\#$ and has φ for left son and another $\#$ for right son:

$$\begin{aligned} \delta(q_i,\#) &\stackrel{\text{def}}{=} (1,q_\varphi) \wedge (2,q_{\#}) & \delta(q_i,a) &\stackrel{\text{def}}{=} \bot \\ \delta(q_\varphi,a) &\stackrel{\text{def}}{=} \delta(\varphi,a) \wedge (2,q_{\#})\ . \end{aligned}$$

For any state q beside q_i and $q_{\#}$

$$\delta(q,\#) \stackrel{\text{def}}{=} \bot\ .$$

Corollary 4.9. *Satisfiability of PDL can be decided in* ExpTime.

Proof sketch. Given a PDL formula φ, by Lemma 4.4 construct an equivalent formula in normal form φ' with $|\varphi'| = O(|\varphi|)$. We then construct $\mathcal{A}_{\varphi'}$ with $O(|\varphi|)$ states by Lemma 4.5 and an alphabet of size at most $2^{O(|\varphi|)}$, s.t. $\bar{t}$ is accepted by $\mathcal{A}_{\varphi'}$ iff $t, \mathrm{root} \models \varphi$. By Theorem 4.8 we can decide the existence of such a tree $\bar{t}$ in time $2^{O(|\varphi|^3)}$. The proof carries to satisfiability on tree structures rather than binary trees. □

4.2.3 Expressiveness

See ten Cate and Segoufin (2010).

Monadic Transitive Closure PDL can be expressed in $\mathrm{FO}[\mathrm{TC}^1]$ the **first-order logic with monadic transitive closure**. The translation can be expressed by induction, yielding formulæ $\mathrm{ST}_x(\varphi)$ with one free variable x for node formulæ and $\mathrm{ST}_{x,y}(\pi)$ with two free variables for path formulæ, such that $\mathfrak{M} \models_{x \mapsto w} \mathrm{ST}_x(\varphi)$ iff

$w \in [\![\varphi]\!]_{\mathfrak{M}}$ and $\mathfrak{M} \models_{x\mapsto u, y\mapsto v} \mathrm{ST}_{x,y}(\pi)$ iff $u\ [\![\pi]\!]_{\mathfrak{M}}\ v$:

$$\begin{aligned}
\mathrm{ST}_x(a) &\overset{\text{def}}{=} P_a(x)\\
\mathrm{ST}_x(\top) &\overset{\text{def}}{=} (x = x)\\
\mathrm{ST}_x(\neg\varphi) &\overset{\text{def}}{=} \neg \mathrm{ST}_x(\varphi)\\
\mathrm{ST}_x(\varphi_1 \vee \varphi_2) &\overset{\text{def}}{=} \mathrm{ST}_x(\varphi_1) \vee \mathrm{ST}_x(\varphi_2)\\
\mathrm{ST}_x(\langle\pi\rangle\varphi) &\overset{\text{def}}{=} \exists y.\mathrm{ST}_{x,y}(\pi) \wedge \mathrm{ST}_y(\varphi)\\
\mathrm{ST}_{x,y}(\downarrow) &\overset{\text{def}}{=} x \downarrow y\\
\mathrm{ST}_{x,y}(\rightarrow) &\overset{\text{def}}{=} x \rightarrow y\\
\mathrm{ST}_{x,y}(\pi^{-1}) &\overset{\text{def}}{=} \mathrm{ST}_{y,x}(\pi)\\
\mathrm{ST}_{x,y}(\pi_1;\pi_2) &\overset{\text{def}}{=} \exists z.\mathrm{ST}_{x,z}(\pi_1) \wedge \mathrm{ST}_{z,y}(\pi_2)\\
\mathrm{ST}_{x,y}(\pi_1+\pi_2) &\overset{\text{def}}{=} \mathrm{ST}_{x,y}(\pi_1) \vee \mathrm{ST}_{x,y}(\pi_2)\\
\mathrm{ST}_{x,y}(\pi^*) &\overset{\text{def}}{=} [\mathrm{TC}_{u,v}\,\mathrm{ST}_{u,v}(\pi)](x,y)\\
\mathrm{ST}_{x,y}(\varphi?) &\overset{\text{def}}{=} (x = y) \wedge \mathrm{ST}_x(\varphi)\ .
\end{aligned}$$

It is known that wMSO is strictly more expressive than FO[TC^1] (ten Cate and Segoufin, 2010, Theorem 2). Ten Cate and Segoufin also provide an extension of PDL with a "within" modality that extracts the subtree at the current node; they show that this extension is exactly as expressive as FO[TC^1]. It is open whether FO[TC^1] is strictly more expressive than PDL without this extension.

Exercise 4.7 (Within modality)**.** Let $\mathfrak{M} = \langle W, \downarrow, \rightarrow, (P_a)_{a\in A}\rangle$ be a tree structure and p be a point in $\mathfrak{M}$. We define the *substructure at* p, noted $\mathfrak{M} \restriction p$, as the substructure induced by $W \restriction p \overset{\text{def}}{=} \{w \in W \mid p \downarrow^\star w\}$. The semantics of a PDLW formula Wφ is defined by $\mathfrak{M}, w \models \mathsf{W}\varphi$ iff $\mathfrak{M} \restriction w, w \models \varphi$.

Propose a translation of PDLW formulæ into FO[TC^1]. **(∗∗)**

Conditional PDL A particular fragment of PDL called **conditional PDL** (cPDL) is *equivalent* to FO[$\downarrow^\star, \rightarrow^\star$]:

See Marx (2005).

$$\pi ::= \alpha \mid \alpha^* \mid \pi;\pi \mid \pi+\pi \mid (\alpha;\varphi?)^* \mid \varphi? \qquad \text{(conditional paths)}$$

The translation to FO[$\downarrow^\star, \rightarrow^\star$] is as above, with

$$\begin{aligned}
\mathrm{ST}_{x,y}(\downarrow) &\overset{\text{def}}{=} x \downarrow^\star y \wedge x \neq y \wedge \forall z. x \downarrow^\star z \wedge x \neq z \supset y \downarrow^\star z\\
\mathrm{ST}_{x,y}(\downarrow^*) &\overset{\text{def}}{=} x \downarrow^\star y\\
\mathrm{ST}_{x,y}((\alpha;\varphi?)^*) &\overset{\text{def}}{=} \forall z.(\mathrm{ST}_{x,z}(\alpha^*) \wedge \mathrm{ST}_{z,y}(\alpha^*)) \supset \mathrm{ST}_z(\varphi)\ .
\end{aligned}$$

An example of a PDL formula that is *not* first-order definable, and thus not definable in cPDL, is $[(\downarrow;\downarrow)^*]a$, which ensures that all the nodes situated at an even distance from the root are labelled by a.

Exercise 4.8. Express the formulæ (4.12)–(4.21) in cPDL. **(∗)**

4.3 Parsing as Intersection

See Boral and Schmitz (2013) for the complexity of PDL parsing when the shape and labels of trees is constrained by a CFG.

The parsing as intersection framework readily applies to model-theoretic syntax. Indeed, in both the wMSO and the PDL cases, given a formula φ, we can effectively construct a non-deterministic tree automaton $\mathcal{A}_\varphi$ that recognises the exactly closed trees that satisfy φ. Given a sentence w to parse, it remains to intersect this tree language $L(\mathcal{A}_\varphi)$ with the set of closed binary trees with w as yield to recover the set of parses of w:

(∗) **Exercise 4.9.** Fix a finite word w and a finite alphabet Γ of internal nodes. Define a non-deterministic tree automaton that recognises the set of closed binary trees with w as yield—the yield should here be understood with the '#' symbols ignored.

Chapter 5

Mildly Context-Sensitive Syntax

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, for the purposes of constituent analysis of syntax, one would like to use string- and tree-generating formalisms with greater expressive power than context-free grammars. The rationale is twofold:

See Pullum (1986).

- 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.

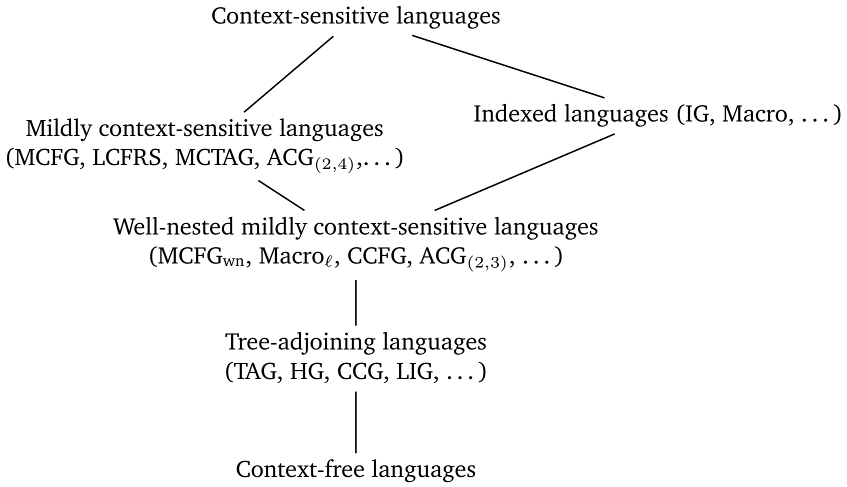

Context-sensitive languages
Mildly context-sensitive languages
(MCFG, LCFRS, MCTAG, $\mathrm{ACG}_{(2,4)}$,. . .)
Indexed languages (IG, Macro, . . .)
Well-nested mildly context-sensitive languages
($\mathrm{MCFG}_{\mathrm{wn}}$, Macro_{ℓ}, CCFG, $\mathrm{ACG}_{(2,3)}$, . . .)
Tree-adjoining languages
(TAG, HG, CCG, LIG, . . .)
Context-free languages

Figure 5.1: Hierarchies between context-free and full context-sensitive languages.

We will however concentrate on two strict subclasses: tree adjoining languages (TALs, Section 5.1) and well-nested MCSLs (wnMCSLs, Section 5.2); Figure 5.1 illustrates the relationship between these classes. As in Section 3.1.1 our main focus will be on the corresponding tree languages, representing linguistic constituency analyses and sentence composition.

5.1 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 5.1 (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* alphabet, Σ a finite *terminal* alphabet 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

- $N{\downarrow} \overset{\text{def}}{=} \{A{\downarrow} \mid A \in N\}$ the ranked alphabet of **substitution** labels, all with arity 0,
- $N^{\mathrm{na}} \overset{\text{def}}{=} \{A^{\mathrm{na}} \mid A \in N\}$ the unranked alphabet of **null adjunction** labels,
- $N_\star \overset{\text{def}}{=} \{A_\star \mid A \in N \cup N^{\mathrm{na}}\}$ the ranked alphabet of **foot** variables, all with arity 0.

In order to work on ranked trees, we confuse N with $N_{>0}$, Σ with Σ_0, and N^{na} with $N^{\mathrm{na}}_{>0}$ in the following. Then the set $T_\alpha \cup T_\beta$ of elementary trees is a set of trees of height at least one. They always have a root labeled by a symbol in $N \cup N^{\mathrm{na}}$, and we define accordingly $\mathrm{rl}(t)$ of a tree t as its *unranked* root label modulo $^{\mathrm{na}}$: $\mathrm{rl}(t) \overset{\text{def}}{=} A$ if there exists m in $\mathbb{N}_{>0}$, $t(\varepsilon) = A^{(m)}$ or $t(\varepsilon) = A^{\mathrm{na}(m)}$. Then

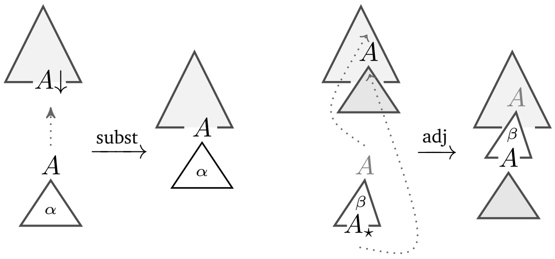


Figure 5.2: Schematics for the substitution and adjunction operations.

- $T_\alpha \subseteq T(N \cup N{\downarrow} \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\})$ 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;
- $T_\beta \subseteq T(N \cup N{\downarrow} \cup N^{\text{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\}, N_\star)$ trees $\beta[A_\star]$ are defined similarly, except for the additional condition that they should have *exactly* one leaf, called the **foot node**, labeled by a variable $A_\star$, which has to match the root label $A = \text{rl}(\beta)$. The foot node $A_\star$ acts as a hole, and the auxiliary tree is basically a context.

The semantics of a TAG is that of a finite term rewrite system with rules (see Figure 5.2)

$$\begin{aligned} R_\mathcal{G} \overset{\text{def}}{=} & \{A{\downarrow} \to \alpha \mid \alpha \in T_\alpha \wedge \text{rl}(\alpha) = A\} && \text{(substitution)}\\ \cup\ & \{A^{(m)}(x_1,\ldots,x_m) \to \beta[A^{(m)}(x_1,\ldots,x_m)] \mid m \in \mathbb{N}_{>0}, A^{(m)} \in N_m, \beta[A_\star] \in T_\beta\}\\ \cup\ & \{A^{(m)}(x_1,\ldots,x_m) \to \beta[A^{\text{na}(m)}(x_1,\ldots,x_m)] \mid m \in \mathbb{N}_{>0}, A^{(m)} \in N_m, \beta[A^{\text{na}}_\star] \in T_\beta\}\ .\\ & && \text{(adjunction)}\end{aligned}$$

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

$$L_T(\mathcal{G}) \overset{\text{def}}{=} \{h(t) \mid \exists t \in T(N \cup \Sigma \cup \{\varepsilon^{(0)}\}), \exists \alpha \in T_\alpha, \text{rl}(\alpha) = S \wedge \alpha \overset{R_\mathcal{G}}{\Longrightarrow}{}^\star\ t\}$$

is the **tree language** of $\mathcal{G}$, where the $^{\text{na}}$ annotations are disposed of, thanks to an alphabetic tree homomorphism h generated by $h(A^{\text{na}(m)}) \overset{\text{def}}{=} A^{(m)}$ for all $A^{\text{na}(m)}$ of N^{na}, and $h(X) \overset{\text{def}}{=} X$ for all X in $N \cup \Sigma \cup \{\varepsilon^{(0)}\}$. The **string language** of $\mathcal{G}$ is

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

the set of yields of all its trees.

Example 5.2. Figure 5.3 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

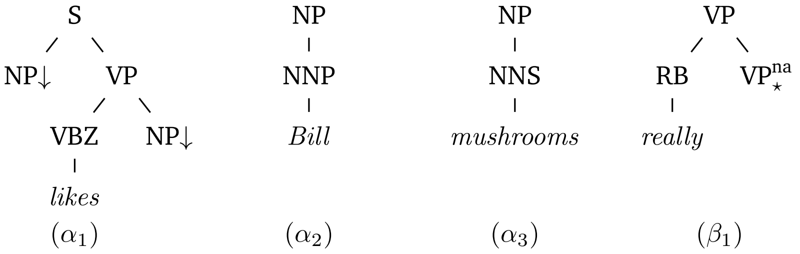


Figure 5.3: A tree adjoining grammar.

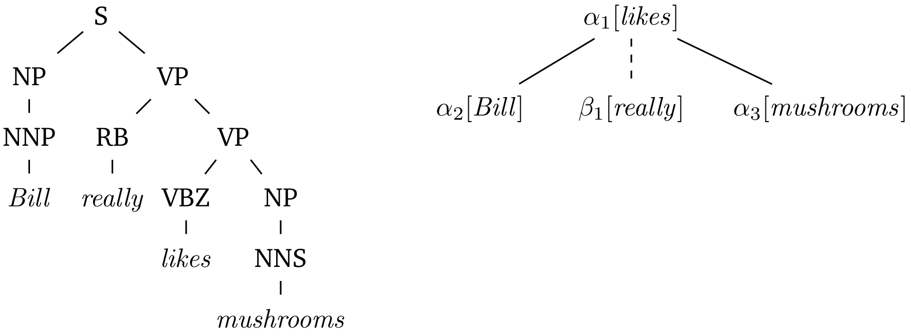


Figure 5.4: A derived tree and the corresponding derivation tree for the TAG of Example 5.2.

VP node of α_1 also yields *Bill really likes mushrooms* (see Figure 5.4) or *mushrooms really really really likes Bill*. In the TAG literature, a tree in $T(N \cup N^{\mathrm{na}} \cup \Sigma \cup \{\varepsilon^{(0)}\})$ obtained through the substitution and adjunction operations is called a **derived tree**, while a **derivation tree** records how the rewrites took place (see Figure 5.4 for an example; children of an elementary tree are shown in addressing order, with plain lines for substitutions and dashed lines for adjunctions).

Example 5.3 (Copy Language)**.** The **copy language** $L_{\text{copy}} \stackrel{\text{def}}{=} \{ww \mid w \in \{a,b\}^*\}$ is generated by the TAG of Figure 5.5 with $N = \{S\}$, $\Sigma = \{a,b\}$, $T_\alpha = \{\alpha_\varepsilon\}$, and $T_\beta = \{\beta_a, \beta_b\}$.

(∗) **Exercise 5.1.** Give a TAG for the language $\{a^n b^m c^n d^m \mid n,m \geq 0\}$.

5.1.1 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

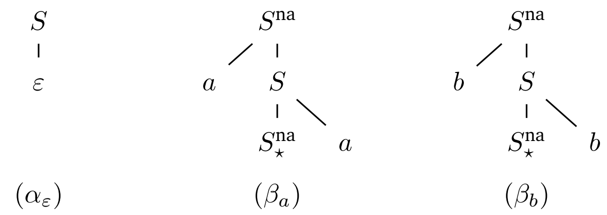


Figure 5.5: A TAG for L_{copy}.

coanchors. An anchor serves as head word for at least a part of the elementary tree, as *likes* for α_1 in Figure 5.3. 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 5.6.

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 5.3; similarly, *to think* takes a clausal S complement, as instantiated by β_2 in Figure 5.6. 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 5.4. Extend the TAG of Figure 5.3 with the trees of Figure 5.6. 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.

Syntactic Lexicon In practice, elementary trees as the ones of Figure 5.3 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 5.3 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, . . .).

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). See also (Kallmeyer and Kuhlmann, 2012) on the extraction of dependency analyses from TAG derivations.

Long-Distance Dependencies

Let us focus on α_4 in Figure 5.6. 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 5.7. We leave the question of syntax/semantics interfaces using derivation trees to later chapters.

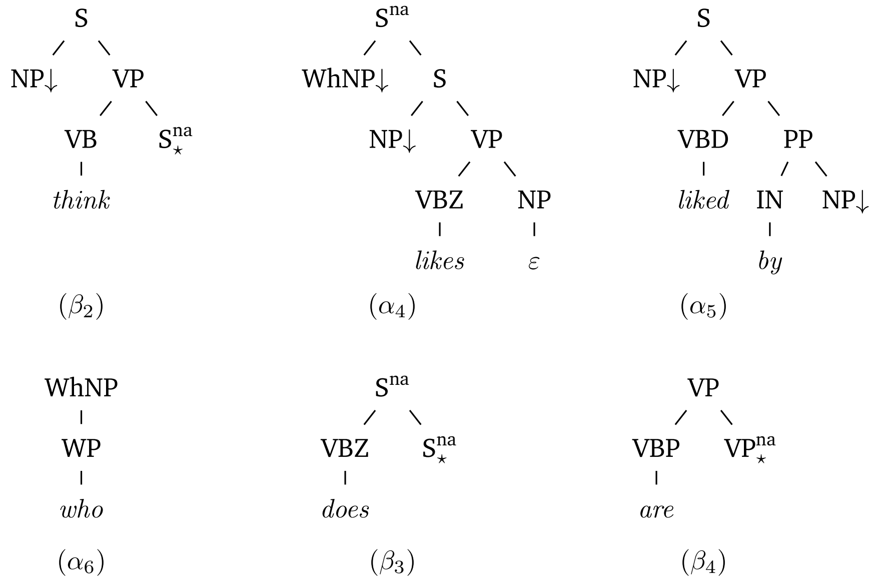


Figure 5.6: More elementary trees for the tree adjoining grammar of Example 5.2.

5.1.2 *Background:* Context-Free Tree Grammars

Context-free tree languages are an extension of regular tree languages proposed by Rounds (1970):

See Gécseg and Steinby (1997, Section 15) and Comon et al. *(2007, Section 2.5). Regarding string languages, the set* yield$(L(\mathcal{G}))$ *of CFTGs characterizes the class of* ***indexed languages*** *(Aho, 1968; Fischer, 1968). Context-free tree languages are also defined through* ***top-down pushdown tree automata*** *(Guessarian, 1983).*

Definition 5.5 (Context-Free Tree Grammars). A **context-free tree grammar** (CFTG) is a tuple $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$ consisting of a ranked *nonterminal* alphabet N, a ranked *terminal* alphabet $\mathcal{F}$, an *axiom* $S^{(0)}$ in N_0, and a finite set of rules R of form $A^{(n)}(y_1, \ldots, y_n) \to e$ with $e \in T(N \cup \mathcal{F}, \mathcal{Y}_n)$ where $\mathcal{Y}$ is an infinite countable set of *parameters*. The *language* of $\mathcal{G}$ is defined as

$$L(\mathcal{G}) \stackrel{\text{def}}{=} \{t \in T(\mathcal{F}) \mid S^{(0)} \xRightarrow{R}{}^{\star}\, t\}.$$

Observe that a **regular tree grammar** is simply a CFTG where every nonterminal is of arity 0.

Example 5.6 (Squares). The CFTG with rules

$$\begin{aligned} S &\to A(a, f(a, f(a, a))) \\ A(y_1, y_2) &\to A(f(y_1, y_2), f(y_2, f(a, a))) \mid y_1 \end{aligned}$$

has $\{a^{n^2} \mid n \geq 1\}$ for $\mathrm{yield}(L(\mathcal{G}))$: Note that

$$\sum_{i=0}^{n-1} 2i + 1 = n + 2\sum_{i=0}^{n-1} i = n^2 \tag{5.1}$$

and that if $S \Rightarrow^n A(t_1, t_2)$, then $\mathrm{yield}(t_1) = a^{n^2}$ and $\mathrm{yield}(t_2) = a^{2n+1}$.

Example 5.7 (Non-primes). The CFTG with rules

$$\begin{aligned} S &\to A(f(a, a)) \\ A(y) &\to A(f(y, a)) \mid B(y) \\ B(y) &\to f(y, B(y)) \mid f(y, y) \end{aligned}$$

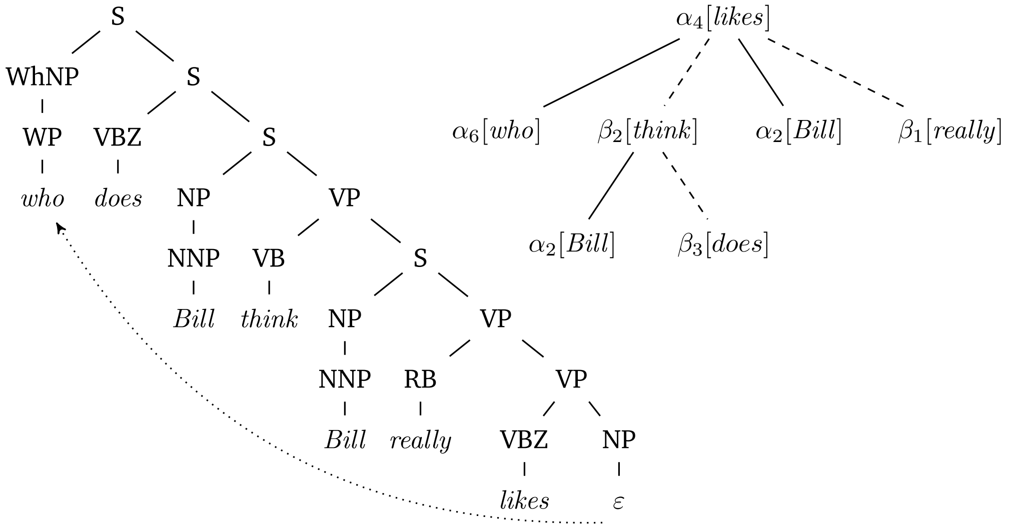


Figure 5.7: Derived and derivation trees for *Who does Bill think Bill really likes?* using the TAG of Figures 5.3 and 5.6.

has $\{a^n \mid n \geq 2 \text{ is not a prime}\}$ for $\text{yield}(L(\mathcal{G}))$: in a derivation

$$S \Rightarrow A(f(a,a)) \Rightarrow^m A(t) \Rightarrow B(t) \Rightarrow^n C[B(t)] \Rightarrow t'$$

with t' in $T(\mathcal{F})$, we have $\text{yield}(t) = a^{2+m}$, $\text{yield}(C[B(t)]) = a^{(2+m)n}$, and finally $\text{yield}(t') = a^{(2+m)(n+1)}$.

Exercise 5.2 (Powers of 2). Give a CFTG with $\text{yield}(L(\mathcal{G})) = \{a^n b a^{2^n} \mid n \geq 1\}$. **(∗)**

Exercise 5.3 (Normal Form). Show that any CFTG can be put in a normal form where every rule in R is either of form $A^{(n)}(y_1, \ldots, y_n) \to a^{(n)}(y_1, \ldots, y_n)$ with a in $\mathcal{F}_n$ or of form $A^{(n)}(y_1, \ldots, y_n) \to e$ with e in $T(N, \mathcal{Y}_n)$. **(∗)**

IO and OI Derivations

If we see derivations in a CFTG as evaluation in a recursive program with nonterminals are functions, a natural way to define the semantics of a nonterminal $A^{(n)}$ is for them to take fully derived trees in $T(\mathcal{F})$ as parameters, i.e. to use *call-by-value* semantics, or equivalently inside-out (**IO**) evaluation of the rewrite rules, i.e. evaluation starting from the innermost nonterminals. The dual possibility is to consider outside-in (**OI**) evaluation, which corresponds to *call-by-name* semantics. Formally, for a set of rewrite rules R,

See Fischer (1968).

$$\xRightarrow{\text{IO}} \stackrel{\text{def}}{=} \xRightarrow{R} \cap \{(C[A^{(n)}(t_1, \ldots, t_n)], C[t]) \mid C \in \mathcal{C}(N \cup \mathcal{F}), A^{(n)} \in N_n, t_1, \ldots, t_n \in T(\mathcal{F})\}$$

$$\xRightarrow{\text{OI}} \stackrel{\text{def}}{=} \xRightarrow{R} \cap \{(C[A^{(n)}(t_1, \ldots, t_n), t_{n+1}, t_{n+m-1}], C[t, t_{n+1}, \ldots, t_{n+m-1}])$$
$$\mid m \geq 1, C \in \mathcal{C}^m(\mathcal{F}), A^{(n)} \in N_n, t_1, \ldots t_{n+m-1} \in T(N \cup \mathcal{F})\}\,.$$

Example 5.8 (IO vs. OI). Consider the CFTG with rules

$$S \to A(B) \qquad A(y) \to f(y,y)$$
$$B \to g(B) \qquad B \to a\,.$$

Then OI derivations are all of form

$$S \overset{\text{OI}}{\Longrightarrow} A(B) \underset{\text{OI}}{\overset{f}{\Longrightarrow}} (B,B) \overset{\text{OI}}{\Longrightarrow}{}^{n+m} f(g^m(a), g^n(a))$$

for some m, n in $\mathbb{N}$, whereas the IO derivations are all of form

$$S \overset{\text{IO}}{\Longrightarrow} A(B) \overset{\text{IO}}{\Longrightarrow}{}^{n} A(g^n(a)) \overset{\text{IO}}{\Longrightarrow} f(g^n(a), g^n(a)) \;.$$

The two modes of derivation give rise to two tree languages $L_{\text{OI}}(\mathcal{G})$ and $L_{\text{IO}}(\mathcal{G})$, both obviously included in $L(\mathcal{G})$.

Theorem 5.9 (Fischer, 1968). *For any CFTG $\mathcal{G}$, $L_{\text{IO}}(\mathcal{G}) \subseteq L_{\text{OI}}(\mathcal{G}) = L(\mathcal{G})$.*

As seen with Example 5.8, the case $L_{\text{IO}}(\mathcal{G}) \subsetneq L_{\text{OI}}(\mathcal{G})$ can occur. Theorem 5.9 shows that can assume OI derivations whenever it suits us; for instance, a basic observation is that OI derivations on different subtrees are independent:

Lemma 5.10. *Let $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$. If $t_1, \ldots, t_n$ are trees in $T(N \cup \mathcal{F})$, C is a context in $\mathcal{C}^n(\mathcal{F})$, and $t = C[t_1, \ldots, t_n] \overset{R}{\Longrightarrow}{}^m t'$ for some m, then there exist $m_1, \ldots, m_n$ in $\mathbb{N}$ and $t'_1, \ldots, t'_n$ in $T(N \cup \mathcal{F})$ s.t. $t_i \overset{R}{\Longrightarrow}{}^{m_i} t'_i$, $m = m_1 + \cdots + m_n$, and $t' = C[t'_1, \ldots, t'_n]$.*

Proof. Let us proceed by induction on m. For the base case, the lemma holds immediately for $m = 0$ by choosing $m_i = 0$ and $t'_i = t_i$ for each $1 \leq i \leq n$. For the induction step, consider a derivation $t = C[t_1, \ldots, t_n] \overset{R}{\Longrightarrow}{}^m t' \overset{R}{\Longrightarrow} t''$. By induction hypothesis, we find $m_1, \ldots, m_n$ and $t'_1, \ldots, t'_n$ with $t_i \overset{R}{\Longrightarrow}{}^{m_i} t'_i$, $m = \sum_{i=1}^n m_i$, and $t' = C[t'_1, \ldots, t'_n] \overset{R}{\Longrightarrow} t''$. Since $C \in \mathcal{C}^n(\mathcal{F})$ is a linear term devoid of nonterminal symbols, the latter derivation step stems from a rewrite occurring in some t'_i subtree. Thus $t_i \overset{R}{\Longrightarrow}{}^{m_i+1} t''_i$ for some t''_i s.t. $t'' = C[t'_1, \ldots, t''_i, \ldots, t'_n]$. □

In contrast with Theorem 5.9, if we consider the *classes* of tree languages that can be described by CFTGs using IO and OI derivations, we obtain incomparable classes (Fischer, 1968).

5.1.3 TAGs as Context-Free Tree Grammars

Tree adjoining grammars can be seen as a special case of **context-free tree grammars** with a few restrictions on the form of its rewrite rules. This is a folklore result, which was stated (at least) by Mönnich (1997), Fujiyoshi and Kasai (2000), and Kepser and Rogers (2011), and which is made even more obvious with the "rewriting"-flavoured definition we gave for TAGs.

Translation from TAGs to CFTGs Given a TAG $\mathcal{G} = \langle N, \Sigma, T_\alpha, T_\beta, S\rangle$, we construct a CFTG $\mathcal{G}' = \langle N', \mathcal{F}, S{\downarrow}, R \cup R'\rangle$ with

$$\begin{aligned}
N' &\overset{\text{def}}{=} N{\downarrow} \cup \{\bar{A}^{(1)} \mid A \in N\} \\
\mathcal{F} &\overset{\text{def}}{=} \Sigma_0 \cup \{\varepsilon^{(0)}\} \cup N_{>0} \\
R &\overset{\text{def}}{=} \{A{\downarrow} \to \tau(\alpha) \mid \alpha \in T_\alpha \wedge \text{rl}(\alpha) = A\} \\
&\quad \cup \{\bar{A}^{(1)}(y) \to \tau(\beta)[\bar{A}^{(1)}(y)] \mid \beta[A_\star] \in T_\beta\} \\
&\quad \cup \{\bar{A}^{(1)}(y) \to \tau(\beta)[y] \mid \beta[A_\star^{\text{na}}] \in T_\beta\} \\
R' &\overset{\text{def}}{=} \{\bar{A}^{(1)}(y) \to y \mid \bar{A}^{(1)} \in \bar{N}\}
\end{aligned}$$

where $\tau : T(\Delta \cup \{\square\}) \to T(\Delta' \cup \{\square\})$ for $\Delta \overset{\text{def}}{=} N{\downarrow} \cup N^{\text{na}}_{>0} \cup N \cup \Sigma_0$ and $\Delta' \overset{\text{def}}{=} N' \cup \mathcal{F}$ is a tree homomorphism generated by

$$\tau(A^{(m)}(x_1, \ldots, x_m)) \overset{\text{def}}{=} \bar{A}^{(1)}(A^{(m)}(x_1, \ldots, x_m))$$
$$\tau(A^{\text{na}(m)}) \overset{\text{def}}{=} A^{(m)}(x_1, \ldots, x_m)$$

and the identity for the other cases (i.e. for symbols in $N{\downarrow} \cup \Sigma_0 \cup \{\varepsilon, \square\}$).

Example 5.11. Consider again the TAG of Figure 5.5 for the copy language: we obtain $\mathcal{G}' = \langle N', \mathcal{F}, S{\downarrow}, R \cup R' \rangle$ with $N' = \{S{\downarrow}, \bar{S}\}$, $\mathcal{F} = \{S, a, b, \varepsilon\}$, and rules

$$\begin{aligned} R = \{S{\downarrow} &\to \bar{S}(S(\varepsilon)), && \text{(corresponding to } \alpha_\varepsilon)\\ \bar{S}(y) &\to S(a, \bar{S}(S(y, a))), && \text{(corresponding to } \beta_a)\\ \bar{S}(y) &\to S(b, \bar{S}(S(y, b)))\} && \text{(corresponding to } \beta_b)\\ R' = \{\bar{S}(y) &\to y\} \ . \end{aligned}$$

Proposition 5.12. $L_T(\mathcal{G}) = L(\mathcal{G}')$.

Proof of $L_T(\mathcal{G}) \subseteq L(\mathcal{G}')$. We first prove by induction on the length of derivations:

Claim 5.12.1. For all trees t in $T(\Delta)$, $t \overset{R_\mathcal{G}}{\Longrightarrow}{}^\star t'$ implies t' is in $T(\Delta)$ and $\tau(t) \overset{R}{\Longrightarrow}{}^\star \tau(t')$.

Proof of Claim 5.12.1. That $T(\Delta)$ is closed under $R_\mathcal{G}$ is immediate. For the second part of the claim, we only need to consider the case of a single derivation step:

For a substitution $C[A{\downarrow}] \overset{R_\mathcal{G}}{\Longrightarrow} C[\alpha]$ occurs iff α is in T_α with $\text{rl}(\alpha) = A$, which implies $\tau(C[A{\downarrow}]) = \tau(C)[\tau(A{\downarrow})] = \tau(C)[A{\downarrow}] \overset{R}{\Rightarrow} \tau(C)[\tau(\alpha)] = \tau(C[\alpha])$.

For an adjunction $C[A^{(m)}(t_1, \ldots, t_m)] \overset{R_\mathcal{G}}{\Longrightarrow} C[\beta[A^{(m)}(t_1, \ldots, t_m)]]$ occurs iff $\beta[A_\star]$ is in T_β, implying

$$\begin{aligned} \tau(C[A^{(m)}(t_1, \ldots, t_m)]) &= \tau(C)[\bar{A}^{(1)}(A^{(m)}(\tau(t_1), \ldots, \tau(t_m)))]\\ &\overset{R}{\Rightarrow} \tau(C)[\tau(\beta)[\bar{A}^{(1)}(A^{(m)}(\tau(t_1), \ldots, \tau(t_m)))]]\\ &= \tau(C[\beta[A^{(m)}(t_1, \ldots, t_m)]]) \ . \end{aligned}$$

The case of a tree $\beta[A^{\text{na}}_\star]$ is similar. [5.12.1]

Claim 5.12.2. If t is a tree in $T(N^{\text{na}} \cup \mathcal{F})$, then there exists a derivation $\tau(t) \overset{R'}{\Longrightarrow}{}^\star h(t)$ in $\mathcal{G}'$.

Proof of Claim 5.12.2. We proceed by induction on t:

For a tree rooted by $A^{(m)}$:

$$\begin{aligned} \tau(A^{(m)}(t_1, \ldots, t_m)) &= \bar{A}^{(1)}(A^{(m)}(\tau(t_1), \ldots, \tau(t_m)))\\ &\overset{R'}{\Longrightarrow} A^{(m)}(\tau(t_1), \ldots, \tau(t_m))\\ &\overset{R'}{\Longrightarrow}{}^\star A^{(m)}(h(t_1), \ldots, h(t_m)) && \text{(by ind. hyp.)}\\ &= h(A^{(m)}(t_1, \ldots, t_m)) \ . \end{aligned}$$

For a tree rooted by $A^{\mathrm{na}(m)}$:

$$\begin{aligned}\tau(A^{\mathrm{na}(m)}(t_1,\ldots,t_m)) &= A^{(m)}(\tau(t_1),\ldots,\tau(t_m))\\ &\overset{R'}{\Longrightarrow}{}^{\star}\ A^{(m)}(h(t_1),\ldots,h(t_m)) && \text{(by ind. hyp.)}\\ &= h(A^{\mathrm{na}(m)}(t_1,\ldots,t_m))\ .\end{aligned}$$

The case of a tree rooted by a in $\Sigma \cup \{\varepsilon\}$ is trivial. [5.12.2]

For the main proof: Let t be a tree in $L_T(\mathcal{G})$; there exist t' in $T(N^{\mathrm{na}} \cup \mathcal{F})$ and α in T_α with $\mathrm{rl}(\alpha) = S$ s.t. $\alpha \overset{R_\mathcal{G}}{\Longrightarrow}{}^{\star}\ t'$ and $t = h(t')$. Then $S{\downarrow} \overset{R}{\Rightarrow} \tau(\alpha) \overset{R}{\Rightarrow}{}^{\star}\ \tau(t')$ according to Claim 5.12.1, and then $\tau(t') \overset{R'}{\Longrightarrow}{}^{\star}\ t$ removes all its nonterminals according to Claim 5.12.2. □

Proof of $L(\mathcal{G}') \subseteq L_T(\mathcal{G})$. We proceed similarly for the converse proof. We first need to restrict ourselves to *well-formed* trees (and contexts): we define the set $L \subseteq T(\Delta' \cup \{\square\})$ as the language of all trees and contexts where every node labeled $\bar{A}^{(1)}$ in $\bar{N}$ has $A^{(m)}$ in N as the label of its daughter—L is defined formally in the proof of the following claim:

Claim 5.12.3. The homomorphism τ is a bijection from $T(\Delta \cup \{\square\})$ to L.

Proof of Claim 5.12.3. It should be clear that τ is injective and has a range included in L. We can define τ^{-1} as a deterministic top-down tree transduction from $T(\Delta' \cup \{\square\})$ into $T(\Delta \cup \{\square\})$ with L for domain, thus proving surjectivity: Let $\mathcal{T} = \langle \{q\} \cup \{q_A \mid A \in N\}, \Delta' \cup \{\square\}, \Delta \cup \{\square\}, \rho, \{q\}\rangle$ with rules

$$\begin{aligned}\rho = {} & \{q(A^{(1)}(x)) \rightarrow q_A(x) \mid \bar{A}^{(1)} \in \bar{N}\}\\ & \cup \{q_A(A^{(m)}(x_1,\ldots,x_m)) \rightarrow A^{(m)}(q(x_1),\ldots,q(x_m)) \mid A^{(m)} \in N\}\\ & \cup \{q(A^{(m)}(x_1,\ldots,x_m)) \rightarrow A^{\mathrm{na}(m)}(q(x_1),\ldots,q(x_m)) \mid A^{(m)} \in N\}\\ & \cup \{q(a^{(m)}(x_1,\ldots,x_m) \rightarrow a^{(m)}(q(x_1),\ldots,q(x_m)) \mid a^{(m)} \in N{\downarrow} \cup \Sigma \cup \{\varepsilon^{(0)}, \square^{(0)}\}\}\ .\end{aligned}$$

We see immediately that $[\![\mathcal{T}]\!](t) = \tau^{-1}(t)$ for all t in L. [5.12.3]

Thanks to Claim 5.12.3, we can use τ^{-1} in our proofs. We obtain claims mirroring Claim 5.12.1 and Claim 5.12.2 using the same types of arguments:

Claim 5.12.4. For all trees t in L, $t \overset{R}{\Rightarrow}{}^{\star}\ t'$ implies t' in L and $\tau^{-1}(t) \overset{R_\mathcal{G}}{\Longrightarrow}{}^{\star}\ \tau^{-1}(t')$.

Claim 5.12.5. If t is a tree in $L \cap T(\bar{N} \cup \mathcal{F})$, t' a tree in $T(\mathcal{F})$, and $t \overset{R'}{\Longrightarrow}{}^{\star}\ t'$, then $h(\tau^{-1}(t')) = \tau^{-1}(t)$.

For the main proof, consider a derivation $S{\downarrow} \overset{R}{\Rightarrow}{}^{\star}\ t$ with $t \in T(\mathcal{F})$ of $\mathcal{G}$. We can reorder this derivation so that $S{\downarrow} \overset{R}{\Rightarrow} \tau(\alpha) \overset{R}{\Rightarrow}{}^{\star}\ \tau(t') \overset{R'}{\Longrightarrow}{}^{\star}\ t$ for some α in T_α with $\mathrm{rl}(\alpha) = S$ and t' in $L \cap T(\bar{N} \cup \mathcal{F})$ (i.e. t' does not contain any symbol from $N{\downarrow}$). By Claim 5.12.4, $\alpha \overset{R_\mathcal{G}}{\Longrightarrow}{}^{\star}\ t'$ and by Claim 5.12.5 $h(t') = \tau^{-1}(t)$. Since t belongs to $T(\mathcal{F})$, $\tau^{-1}(t) = t$, which shows that t belongs to $L_T(\mathcal{G})$. □

From CFTGs to TAGs The converse direction is more involved, because TAGs as usually defined have *locality* restrictions (in a sense comparable to that of CFGs generating only *local* tree languages) caused by their label-based selection mechanisms for the substitution and adjunction rules. This prompted the definition of **non-strict** definitions for TAGs, where root and foot labels of auxiliary trees do not have to match, where tree selection for substitution and adjunction is made through *selection lists* attached to each substitution node or adjunction site, and where elementary trees can be reduced to a leaf or a foot node (which does not make much sense for strict TAGs due to the selection mechanism); see Kepser and Rogers (2011).

Putting these considerations aside, the essential fact to remember is that TAGs are "almost" equivalent to **linear**, **monadic** CFTGs as far as tree languages are concerned, and *exactly* for string languages: a CFTG is called

- **linear** if, for every rule $A^{(n)}(y_1, \ldots, y_n) \to e$ in R, the right-hand side e is linear,

- **monadic** if the maximal rank of a non-terminal is 1.

Exercise 5.4 (Non-Strict TAGs)**.** Definition 5.1 is a **strict** definition of TAGs. **(***)**

1. Read the definition of non-strict TAGs given by Kepser and Rogers (2011). Show that strict and non-strict TAGs derive the same string languages.

2. Give a non-strict TAG for the regular tree language
$$S((A(a,\square))^* \cdot b, (A(\square,a))^* \cdot b)\ . \tag{5.2}$$

3. Can you give a strict TAG for it? There are more trivial tree languages lying beyond the reach of strict TAGs: prove that the two following finite languages are not TAG tree languages:
$$\{A(a), B(a)\} \tag{5.3}$$
$$\{a\} \tag{5.4}$$
Note that allowing distinct foot and root labels in auxiliary trees is useless for these examples.

5.2 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 5.1. This provides characterizations both in terms of

See (Kuhlmann, 2013) for related definitions in terms of dependency syntax.

- **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).

We concentrate on this second view.

5.2.1 Linear CFTGs

As already seen with tree adjoining grammars, the case of **linear** CFTGs is of particular interest. Intuitively, the relevance of linearity for linguistic modeling is that *arguments* in a subcategorization frame have a linear behaviour: they should appear exactly the stated number of times (by contrast, *modifiers* can be added freely).

See Kepser and Mönnich (2006).

Linear CFTGs enjoy a number of properties. For instance, unlike the general case, for linear CFTGs the distinction between IO and OI derivations is irrelevant:

Proposition 5.13. *Let $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$ be a linear CFTG. Then $L_{\mathrm{IO}}(\mathcal{G}) = L_{\mathrm{OI}}(\mathcal{G})$.*

Proof. Consider a derivation $S \overset{R}{\Longrightarrow}^\star t$ in a linear CFTG. Thanks to Theorem 5.9, we can assume this derivation to be OI. Let us pick the last non-IO step within this OI derivation:

$$\begin{aligned} S &\overset{\mathrm{OI}}{\Longrightarrow}^\star C[A^{(n)}(e_1, \dots, e_n)] \\ &\overset{r_A}{\Longrightarrow} C[e_A\{y_1 \leftarrow e_1, \dots, y_n \leftarrow e_n\}] \\ &\overset{\mathrm{IO}}{\Longrightarrow}^\star t \end{aligned}$$

using some rule $r_A : A^{(n)}(y_1, \dots, y_n) \to e_A$, where an e_i contains a nonterminal. By Lemma 5.10, we can "pull" all the independent rewrites occurring after this $\overset{r_A}{\Longrightarrow}$ so that they occur before the $\overset{r_A}{\Longrightarrow}$ rewrite, so that the next rewrite occurs within the context C. Since everything after this $\overset{r_A}{\Longrightarrow}$ is IO, this rewrite has to involve an innermost nonterminal, thus a nonterminal that was not introduced in e_A, but one that already appeared in some e_i: in the context C:

$$\begin{aligned} &e_A\{y_1 \leftarrow e_1, \dots, y_i \leftarrow C'[B^{(m)}(e'_1, \dots, e'_m)], \dots, y_n \leftarrow e_n\} \\ &\overset{r_B}{\Longrightarrow} e_A\{y_1 \leftarrow e_1, \dots, y_i \leftarrow C'[e_B\{x_1 \leftarrow e'_1, \dots, x_m \leftarrow e'_m\}], \dots, y_n \leftarrow e_n\} \end{aligned}$$

which is possible *thanks to linearity*: in general, there is no way to force the various copies of e_i to use the same rewrite for $B^{(m)}$. Now this sequence is easily swapped: in the context C:

$$\begin{aligned} &A^{(n)}(e_1, \dots, C'[B^{(m)}(e'_1, \dots, e'_m)], \dots, e_n) \\ &\overset{r_B}{\Longrightarrow} A^{(n)}(e_1, \dots, C'[e_B\{x_1 \leftarrow e'_1, \dots, x_m \leftarrow e'_m\}], \dots, e_n) \\ &\overset{r_A}{\Longrightarrow} e_A\{y_1 \leftarrow e_1, \dots, y_i \leftarrow C'[e_B\{x_1 \leftarrow e'_1, \dots, x_m \leftarrow e'_m\}], \dots, y_n \leftarrow e_n\}\ . \end{aligned}$$

Repeating this operation for every nonterminal that occurred in the e_i's yields a derivation of the same length for $S \overset{R}{\Longrightarrow}^\star t$ with a shorter OI prefix and a longer IO suffix. Repeating the argument at this level yields a full IO derivation. □

Proposition 5.13 allows to apply several results pertaining to IO derivations to linear CFTGs. A simple one is an alternative semantics for IO derivations in a CFTG $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$: the semantics of a nonterminal $A^{(n)}$ can be recast as a subset of the relation $[\![A^{(n)}]\!] \subseteq (T(\mathcal{F}))^{n+1}$:

$$[\![A^{(n)}]\!](t_1, \dots, t_n) \overset{\mathrm{def}}{=} \bigcup_{(A^{(n)}(y_1,\dots,y_n)\to e)\in R} [\![e]\!](t_1, \dots, t_n)$$

where $[\![e]\!] \subseteq (T(\mathcal{F}))^{n+1}$ is defined inductively for all subterms e in rule right-hand sides—with n variables in the corresponding *full* term—by

$$[\![a^{(m)}(e_1,\dots,e_m)]\!](t_1,\dots,t_n) \stackrel{\text{def}}{=} \{a^{(m)}(t'_1,\dots,t'_m) \mid \forall 1 \leq i \leq m. t'_i \in [\![e_i]\!](t_1,\dots,t_n)\}$$
$$[\![B^{(m)}(e_1,\dots,e_m)]\!](t_1,\dots,t_n) \stackrel{\text{def}}{=} \{[\![B^{(m)}]\!](t'_1,\dots,t'_m) \mid \forall 1 \leq i \leq m. t'_i \in [\![e_i]\!](t_1,\dots,t_n)\}$$
$$[\![y_i]\!](t_1,\dots,t_n) \stackrel{\text{def}}{=} \{t_i\} \ .$$

The consequence of this definition is

$$L_{\text{IO}}(\mathcal{G}) = [\![S^{(0)}]\!] \ .$$

This semantics will be easier to employ in the following proofs concerned with IO derivations (and thus applicable to linear CFTGs).

5.2.2 Parsing as Intersection

This section relies heavily on Maneth et al. *(2007).*

Let us look into more algorithmic issues and consider the parsing problem for linear CFTGs. In order to apply the parsing as intersection paradigm, we need two main ingredients: the first is emptiness testing (Proposition 5.14), the second is closure under intersection with regular sets (Proposition 5.15). We actually prove these results for IO derivations in CFTGs rather than for linear CFTGs solely.

Proposition 5.14 (Emptiness). *Given a CFTG $\mathcal{G}$, one can decide whether $L_{\text{IO}}(\mathcal{G}) = \emptyset$ in $O(|\mathcal{G}|)$.*

Proof sketch. Given $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$, we construct a context-free grammar $\mathcal{G}' = \langle N', \emptyset, P, S\rangle$ s.t. $L_{\text{IO}}(\mathcal{G}) = \emptyset$ iff $L(\mathcal{G}') = \emptyset$ and $|\mathcal{G}'| = O(|\mathcal{G}|)$. Since emptiness of CFGs can be tested in linear time, this will yield the result. We define for this

$$N' \stackrel{\text{def}}{=} N \cup \bigcup_{A^{(m)}(y_1,\dots,y_m)\to e \in R} \text{Sub}(e) \ ,$$

i.e. we consider both nonterminals and positions inside rule right hand sides as nonterminals of $\mathcal{G}'$, and

$$\begin{aligned} P' \stackrel{\text{def}}{=} & \{A \to e \mid A^{(m)}(y_1,\dots,y_m) \to e \in R\} && \text{(rules)} \\ & \cup \{a^{(m)}(e_1,\dots,e_m) \to e_1 \cdots e_m \mid a \in \mathcal{F} \cup \mathcal{Y}\} && \text{($\mathcal{F}$- or $\mathcal{Y}$-labeled positions)} \\ & \cup \{A^{(m)}(e_1,\dots,e_m) \to A e_1 \cdots e_m\} \ . && \text{(N-labeled positions)} \end{aligned}$$

We note N-labeled positions with arity information and nonterminal symbols without in order to be able to distinguish them. Note that terminal- or variable-labeled positions with arity 0 give rise to empty rules, whereas for nonterminal-labeled positions of arity 0 we obtain unit rules.

The constructed grammar is clearly of linear size; we leave the fixpoint induction proof of $X \stackrel{\mathcal{G}'}{\Longrightarrow}{}^\star \varepsilon$ iff $[\![X]\!] \neq \emptyset$ to the reader. □

Proposition 5.15 (Closure under Intersection with Regular Tree Languages). *Let $\mathcal{G}$ be a (linear) CFTG with maximal nonterminal rank M and maximal number of nonterminals in a right-hand side D, and $\mathcal{A}$ a DTA with $|Q|$ states. Then we can construct a (linear) CFTG $\mathcal{G}'$ with $L_{\text{IO}}(\mathcal{G}') = L_{\text{IO}}(\mathcal{G}) \cap L$ and $|\mathcal{G}'| = O(|\mathcal{G}| \cdot |Q|^{M+D+1})$.*

Proof. Let $\mathcal{G} = \langle N, \mathcal{F}, S, R\rangle$ and $\mathcal{A} = \langle Q, \mathcal{F}, \delta, F\rangle$. We define $\mathcal{G}' = \langle N', \mathcal{F}, S', R'\rangle$ where

$$N' \stackrel{\text{def}}{=} \{S'\} \cup \bigcup_{m \leq M} N_m \times Q^{m+1},$$

i.e. we add a new axiom and otherwise consider tuples of form $\langle A^{(m)}, q_0, q_1, \ldots, q_m\rangle$ as nonterminals of rank m,

$$\begin{aligned} R' &\overset{\text{def}}{=} \{S' \to \langle S, q_f\rangle \mid q_f \in F\} \\ &\cup \{\langle A, q_0, \ldots, q_m\rangle^{(m)}(y_1, \ldots, y_m) \to e' \\ &\qquad \mid A^{(m)}(y_1, \ldots, y_m) \to e \in R \wedge e' \in \theta_{q_0q_1\cdots q_m}(e)\}, \end{aligned}$$

where each $\theta_{q_0q_1\cdots q_m}$ is a nondeterministic translation of right-hand sides, under the understanding that variable y_i should hold a tree recognized by state q_i and the root should be recognized by q_0:

$$\begin{aligned} \theta_{q_0q_1\cdots q_m}(a^{(m)}(e_1, \ldots, e_m)) &\overset{\text{def}}{=} \{a^{(m)}(e'_1, \ldots, e'_m) \mid \exists(q_0, a, q'_1, \ldots, q'_m) \in \delta, \\ &\qquad \forall 1 \leq i \leq m, e'_i \in \theta_{q'_iq_1\cdots q_m}(e_i)\} \\ \theta_{q_0q_1\cdots q_m}(B^{(m)}(e_1, \ldots, e_m)) &\overset{\text{def}}{=} \{\langle B, q_0, q'_1, \ldots, q'_m\rangle(e'_1, \ldots, e'_m) \mid \forall 1 \leq i \leq m, \\ &\qquad q'_i \in Q \wedge e'_i \in \theta_{q'_iq_1\cdots q_m}(e_i)\} \\ \theta_{q_iq_1\cdots q_m}(y_i) &\overset{\text{def}}{=} \{y_i\}\ . \end{aligned}$$

The intuition behind this definition is that $\mathcal{G}'$ guesses that the trees passed as y_i parameters will be recognized by state q_i of $\mathcal{A}$, leading to a tree generated by $A^{(m)}$ and recognized by q_0. A computationally expensive point is the translation of nonterminals in the right-hand side, where we actually guess an assignment of states for its parameters.

We can already check that $\mathcal{G}'$ is constructed through at most $|R| \cdot |Q|^{M+1}$ calls to θ translations, each allowing at most $|Q|^D$ choices for the nonterminals in the argument right-hand side. In fine, each rule of $\mathcal{G}$ is duplicated at most $|Q|^{M+D+1}$ times.

For a tuple of states $q_1, \ldots, q_m$ in Q^m, let us define the relation $[\![q_1 \cdots q_m]\!] \subseteq (T(\mathcal{F}))^m$ as the cartesian product of the sets $[\![q_i]\!] \overset{\text{def}}{=} \{t \in T(\mathcal{F}) \mid q_i \overset{R_T}{\Longrightarrow}{}^\star\ t\}$. We can check that, for all $m \leq M$, all states $q_0, q_1, \ldots, q_m$ of Q, and all nonterminals $A^{(m)}$ of N,

$$[\![\langle A, q_0, q_1, \ldots, q_m\rangle]\!]([\![q_1 \cdots q_m]\!]) = [\![A^{(m)}]\!] \cap [\![q_0]\!]\ .$$

This last equality proves the correctness of the construction. □

In order to use these results for *string* parsing, we merely need to construct, given a string w and a ranked alphabet $\mathcal{F}$, the "universal" DTA with w as yield—it has $O(|w|^2)$ states, thus we can obtain an $O(|\mathcal{G}| \cdot |w|^{2(M+D+1)})$ upper bound for IO parsing with CFTGs, *even in the non linear case*.

Chapter 6

Probabilistic Syntax

Probabilistic approaches to syntax and 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 3.2, 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.

Smoothing and Hidden Variables The relevance of statistical models of syntax has been a subject of heated discussion: Chomsky (1957) famously wrote

See Pereira (2000).

> (1) Colorless green ideas sleep furiously.
> (2) Furiously sleep ideas green colorless.
>
> . . . It is fair to assume that neither sentence (1) nor (2) (nor indeed any parts of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be rules out on identical grounds as equally 'remote' from English. Yet (1), though nonsensical, is grammatical, while (2) is not.

The main issue with this statement is the 'in any statistical model' bit, which actually assumes a rather impoverished statistical model, unable to assign a non-null probability to unseen events. The current statistical models are quite capable of handling them, mainly through two techniques:

smoothing which consists in assigning some weight to unseen events (and renormalizing probabilities). A very basic smoothing technique is called **Laplace smoothing**, and simply adds 1 to the counts of occurrence of any unseen event. Using such a technique over the *Google books corpus* from 1800 to 1954, Norvig trains a model where (1) is about 10^4 times more probable than (2).

hidden variables where the model assumes the existence of **hidden variables** responsible for the observations. Pereira trains a model using the **expectation maximization** method on newspaper text, where (1) is about 2.10^5 times more probable than (2).

We will *not* go much into the details of learning algorithms (which is the subject of another course at MPRI), but rather look at the algorithmics of weighted models.

6.1 Weighted and Probabilistic CFGs

The models we consider are actually *weighted* models defined over semirings, for which probabilities are only one particular case.

6.1.1 *Background:* Semirings and Formal Power Series

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 $\mathbb{K}$.

A semiring is **commutative** if $\langle \mathbb{K}, \odot, 1_\mathbb{K} \rangle$ is a commutative monoid.

Among the main semirings of interest are the

boolean semiring $\langle \mathbb{B}, \vee, \wedge, 0, 1 \rangle$ where $\mathbb{B} = \{0, 1\}$,

probabilistic semiring $\langle \mathbb{R}_{\geq 0}, +, \cdot, 0, 1 \rangle$ where $\mathbb{R}_{\geq 0} = [0, +\infty)$ is the set of non-negative reals (sometimes restricted to $[0, 1]$ when in presence of a probability distribution),

tropical semiring $\langle \mathbb{R}_{\geq 0} \uplus \{+\infty\}, \min, +, +\infty, 0 \rangle$,

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. This is the only non-commutative example here.

Weighted Automata

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

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

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

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 6.1 (Hadamard Product). 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 $s \odot s'$? **(∗∗)**

6.1.2 Weighted Grammars

Definition 6.1 (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 $\alpha \overset{\pi}{\underset{\mathrm{lm}}{\Longrightarrow}}{}^\star \beta$ 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 $S \overset{\pi}{\underset{\mathrm{lm}}{\Longrightarrow}}{}^\star \gamma$, i.e.

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

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

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

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 $S \overset{\pi_1}{\underset{\mathrm{lm}}{\Longrightarrow}}{}^\star uA\gamma \overset{\pi_2}{\underset{\mathrm{lm}}{\Longrightarrow}}{}^\star uv$ and $\rho(\pi_1\pi_2) \neq 0_{\mathbb{K}}$.

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

Exercise 6.2. 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. **(∗∗)**

6.1.3 Probabilistic Grammars

Definition 6.1 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,

$$\sum_{p = A \to \alpha \in P} \rho(p) = 1 \ , \tag{6.1}$$

i.e. ρ can be seen as a mapping from N to $\mathrm{Disc}(\{p \in P \mid p = A \to \alpha\})$, where $\mathrm{Disc}(S)$ denotes the set of discrete distributions over S, i.e. $\{p : S \to [0,1] \mid \sum_{e \in S} p(e) = 1\}$.

Partition Functions

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

$$Z(A) = \sum_{w \in \Sigma^*, A \overset{\pi}{\underset{\mathrm{lm}}{\Longrightarrow}}{}^{*} w} \rho(\pi) \;. \tag{6.2}$$

A PCFG is **convergent** if

$$Z(S) < \infty \;; \tag{6.3}$$

in particular, it is **consistent** if

$$Z(S) = 1 \;, \tag{6.4}$$

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 (6.2) can be decomposed using commutativity of multiplication into

$$Z(A) = \sum_{p = A \to \alpha \in P} \rho(p) \cdot Z(\alpha) \qquad \text{for all } A \text{ in } N \tag{6.5}$$

$$Z(a) = 1 \qquad \text{for all } a \text{ in } \Sigma \uplus \{\varepsilon\} \tag{6.6}$$

$$Z(X\beta) = Z(X) \cdot Z(\beta) \qquad \text{for all } (X, \beta) \text{ in } V \times V^*. \tag{6.7}$$

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

Example 6.2. Properness and consistency are two distinct notions. For instance, the PCFG

$$S \xrightarrow{q} S\,S$$
$$S \xrightarrow{1-q} a$$

is proper for all $0 \leq q \leq 1$, but the equation $x = qx^2 + 1 - q$ has two roots 1 and $\frac{1-q}{q}$, and thus if $q \leq \frac{1}{2}$ the grammar is consistent with $Z(S) = 1$, but otherwise $Z(S) = \frac{1-q}{q} < 1$. Conversely,

$$S \xrightarrow{q/(1-q)} A$$
$$A \xrightarrow{q} A\,A$$
$$A \xrightarrow{1-q} a$$

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

$$S \xrightarrow{1/6} S\,S\,S\,S\,S$$
$$S \xrightarrow{1/2} a\ .$$

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

$$\rho'(p = A \to \alpha) = \frac{\rho(p)Z(\alpha)}{Z(A)}\ . \tag{6.8}$$

Exercise 6.3. 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 (6.8) is well-defined.) **(∗)**

Exercise 6.4. Show that $\mathcal{G}'$ is a proper PCFG. **(∗)**

Proposition 6.4. *The grammar* $\mathcal{G}'$ *defined by* (6.8) *is consistent if* $\mathcal{G}$ *is reduced and convergent.*

Proof. We rely for the proof on the following claim:
Claim 6.4.1. For all Y in V, π in P^*, and w in Σ^* with $Y \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^\star\, w$,

$$\rho'(\pi) = \frac{\rho(\pi)}{Z(Y)}\ . \tag{6.9}$$

Proof of Claim 6.4.1. Note that, because $\mathcal{G}$ is reduced, $Z(Y) > 0$ for all Y in V, so all the divisions we perform are well-defined.

We prove the claim by induction over the derivation π. For the base case, in an empty derivation $\pi = \varepsilon$, $\rho'(\varepsilon) = \rho(\varepsilon) = 1$ and $Z(Y) = 1$ since Y is necessarily a terminal, hence the claim holds. For the induction step, consider a derivation $p\pi$ for some production $p = A \to X_1 \cdots X_m$: $A \underset{\text{lm}}{\overset{p}{\Longrightarrow}} X_1 \cdots X_m \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^\star\, w$. This derivation can be decomposed using a derivation $X_i \underset{\text{lm}}{\overset{\pi_i}{\Longrightarrow}}{}^\star\, w_i$ for each i, such that $\pi = \pi_1 \cdots \pi_n$ and $w = w_1 \cdots w_n$. By induction hypothesis, $\rho'(\pi_i) = \rho(\pi_i)/Z(X_i)$. Hence

$$\begin{aligned}
\rho'(p\pi) &= \rho'(p) \cdot \prod_{i=1}^{m} \rho'(\pi_i) \\
&= \frac{\rho(p)Z(X_1 \cdots X_m)}{Z(A)} \cdot \prod_{i=1}^{m} \rho'(\pi_i) && \text{(by (6.8))} \\
&= \frac{\rho(p)}{Z(A)} \cdot \prod_{i=1}^{m} Z(X_i) \cdot \prod_{i=1}^{m} \frac{\rho(\pi_i)}{Z(X_i)} && \text{(by ind. hyp.)} \\
&= \frac{\rho(p)}{Z(A)} \cdot \prod_{i=1}^{m} \rho(\pi_i) \\
&= \frac{\rho(p\pi)}{Z(A)}\ .
\end{aligned}$$

[6.4.1]

Claim 6.4.1 shows that $\mathcal{G}'$ is consistent, since

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

Remark 6.5. Note that Claim 6.4.1 also yields for all w in Σ^*

$$\rho'(w) = \sum_{S\overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}^\star w} \rho'(\pi) = \sum_{S\overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}^\star w} \frac{\rho(\pi)}{Z(S)} = \frac{\rho(w)}{Z(S)}\ , \tag{6.10}$$

thus the ratios between derivation weights are preserved by the normalization procedure.

Example 6.6. Considering again the first grammar of Example 6.2, 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.

6.2 Learning PCFGs

We rely on an annotated corpus for **supervised** learning. We consider for this 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)

$$\rho(p = A \to \alpha) = \frac{C(p, D)}{C(A, D)}\ . \tag{6.11}$$

(∗∗) **Exercise 6.5.** Show that the obtained PCFG is proper and consistent.

*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—they *overfit* the available data.

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 numerator of (6.11) and normalize, called **Laplace smoothing**), or more involved ones that take into account the probability of observations with a single occurrence (**Good-Turing discounting**).

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 allows to capture some *hidden state* information from the treebank.

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:

[$_{\text{NP}}$ He] [$_{\text{VP}}$[$_{\text{VP}}$ hurled [$_{\text{NP}}$ the ball]] [$_{\text{PP}}$ into the basket]].
[$_{\text{NP}}$ She] [$_{\text{VP}}$ took [$_{\text{NP}}$[$_{\text{NP}}$ a sip] [$_{\text{PP}}$ 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 6.1 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

$$\text{VP}[\text{S}, \mathit{hurled}, \text{VBD}] \to \text{VP}[\text{VP}, \mathit{hurled}, \text{VBD}]\ \text{PP}[\text{VP}, \mathit{into}, \text{IN}]\ ,$$

allowing to give it a higher probability than that of

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

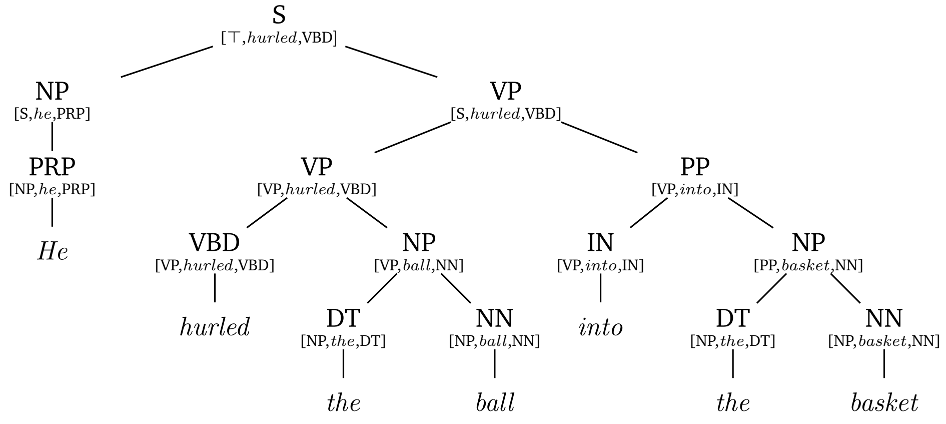


Figure 6.1: A derivation tree refined with lexical and parent information.

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

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

for sentence

This mostly happens because we [$_{\text{VP}}$ go [$_{\text{PP}}$ from football] [$_{\text{PP}}$ in the fall] [$_{\text{PP}}$ to lifting] [$_{\text{PP}}$ in the winter] [$_{\text{PP}}$ to football] [$_{\text{ADVP}}$ again] [$_{\text{PP}}$ 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

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

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.

6.3 Probabilistic Parsing as Intersection

We generalize in this section the intersective approach of Theorem 3.7. 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.

6.3.1 Weighted Product

We generalize here Theorem 3.7 to the weighted case. Observe that it also answers Exercise 6.1 since $\mathbb{K}$-automata are equivalent to right-linear $\mathbb{K}$-CFGs according to Exercise 6.2.

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

Theorem 6.7. *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*

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

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

See Maletti and Satta (2009) for a version of Theorem 6.7 that works on weighted tree automata instead of CFGs.

As with Theorem 3.7, the construction of Theorem 6.7 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 6.6. Modify the quadratic form construction of Lemma 3.8 for the weighted case. **(∗)**

6.3.2 Most Probable Parse

The weighted CFG $\mathcal{G}'$ constructed by Theorem 6.7 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 6.7 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 $j = 1, \ldots, |N'|$

$$M(A_j) = \max_{A \xrightarrow{k} X_1 \cdots X_m} k \cdot M(X_1) \cdots M(X_m) \tag{6.12}$$

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 when we use (6.12) to compute $M(A_j)$.

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): see Algorithm 6.1.

```
Data: $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$
foreach $a \in \Sigma$ do
    $M(a) = 1$
$D \longleftarrow \Sigma$
while $D \neq V$ do
    foreach $A \in V \backslash D$ do
        $\nu(A) \longleftarrow \max_{A \xrightarrow{k} X_1 \cdots X_m \text{ s.t. } X_1, \ldots, X_m \in D} k \cdot M(X_1) \cdots M(X_m)$
    $A \longleftarrow \mathrm{argmax}_{V \backslash D}\, \nu(A)$
    $M(A) \longleftarrow \nu(A)$
    $D \longleftarrow D \uplus \{A\}$
return $M(S)$
```

Algorithm 6.1: Most probable derivation.

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 7, and tracking which productions to consider for the computation of $\nu(A)$ at line 6, 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 8; assuming the opposite, there must exist a shortest derivation $B \overset{\pi}{\underset{\text{lm}}{\Longrightarrow}}{}^{\star}\, w$ with $\rho(\pi) > \nu(A)$ for some $B \notin D$. We can split this derivation into $B \overset{p}{\underset{\text{lm}}{\Longrightarrow}}{}^{\star}\, X_1 \cdots X_m$ and $X_i \overset{\pi_i}{\underset{\text{lm}}{\Longrightarrow}}{}^{\star}\, w_i$ 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 6, and finally $\rho(\pi) \leq \nu(B) \leq \nu(A)$ by line 8—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.

6.3.3 Most Probable String

Section inspired by de la Higuera and Oncina (2011, 2013).

We have just seen that the algorithms for the Boolean case are rather easy to extend in order to handle general (commutative) semirings, including the probabilistic semiring. Let us finish with an example showing that *some* problems become hard.

Consider the following decision problems:

Most Probable String (MPS**)**

input a PCFG $\mathcal{G}$ over Σ with rational weights (coded in binary) and a rational p in $[0, 1]$ (also in binary);

question is there a string w in Σ^* s.t. $\langle [\![\mathcal{G}]\!], w\rangle \geq p$?

Bounded Most Probable String (BMPS**)**

input a PCFG $\mathcal{G}$ over Σ with rational weights (coded in binary), a length b (in unary), and a rational p in $[0, 1]$ (in binary);

question is there a string w in $\Sigma^{\leq b}$ s.t. $\langle [\![\mathcal{G}]\!], w\rangle \geq p$?

Example 6.8. Consider the following right-linear proper consistent PCFG:

$$\begin{aligned} S &\xrightarrow{9/10} aA & \qquad S &\xrightarrow{1/10} b \\ A &\xrightarrow{2/3} aA & A &\xrightarrow{1/3} aB \\ B &\xrightarrow{2/3} aB & B &\xrightarrow{1/3} a\ . \end{aligned}$$

The most probable derivations are for the strings b and aaa, with probability $1/10$. The most probable strings are actually $aaaa$ and $aaaaa$, with probability $(4/3) \cdot (1/10)$.

Hardness

We show here that both problems are already hard for convergent right-linear PCFGs. Note that, in the *non convergent* right-linear case, MPS is known as the Threshold Problem for Rabin probabilistic automata and is undecidable (e.g. Blondel and Canterini, 2003).

Theorem 6.9 (Casacuberta and de la Higuera, 2000). *MPS and BMPS for convergent right-linear PCFGs are NP-hard.*

See also the work of Sima'an (2002) for similar bounds.

Proof. The proof reduces from SAT. Let $\varphi = \bigwedge_{i=1}^{k} C_k$ be a propositional formula in conjunctive normal form, where each clause C_i is a non-empty disjunction of literals over the set of variables $\{x_1, \dots, x_n\}$. Without loss of generality, we assume that each variable appears at most once in each clause, be it positively or negatively.

We construct in polynomial time an instance $\langle \mathcal{G}, p\rangle$ of MPS or $\langle \mathcal{G}, b, p\rangle$ of BMPS such that φ is satisfiable if and only if there exists w in Σ^* such that $\langle [\![\mathcal{G}]\!], w\rangle \geq p$. We define for this $\mathcal{G} \overset{\text{def}}{=} \langle N, \Sigma, P, S\rangle$ where

$$\begin{aligned}
N &\overset{\text{def}}{=} \{S\} \uplus \{A_{i,j} \mid 1 \leq i \leq k \wedge 0 \leq j \leq n\} \uplus \{B_j \mid 1 \leq j \leq n\}\\
\Sigma &\overset{\text{def}}{=} \{0, 1, \$\} ,\\
P &\overset{\text{def}}{=} \{S \xrightarrow{1/k} \$A_{i,0} \mid 1 \leq i \leq k\}\\
&\quad \cup \{A_{i,j-1} \xrightarrow{1/2} vB_j, A_{i,j-1} \xrightarrow{1/2} (1-v)A_{i,j} \mid v \in \{0,1\} \wedge x_j \mapsto v \models C_i\\
&\qquad\qquad \wedge 1 \leq i \leq k \wedge 1 \leq j \leq n\}\\
&\quad \cup \{A_{i,j-1} \xrightarrow{1/2} 1A_{i,j}, A_{i,j-1} \xrightarrow{1/2} 0A_{i,j} \mid x_j \notin C_i \wedge 1 \leq i \leq k \wedge 1 \leq j \leq n\}\\
&\quad \cup \{A_{i,n} \xrightarrow{0} \$ \mid 1 \leq i \leq k\}\\
&\quad \cup \{B_{j-1} \xrightarrow{1/2} 0B_i, B_{j-1} \xrightarrow{1/2} 1B_i \mid 2 \leq j \leq n\}\\
&\quad \cup \{B_n \xrightarrow{1} \$\}
\end{aligned}$$

and fix

$$\begin{aligned}
b &\overset{\text{def}}{=} n + 2 ,\\
p &\overset{\text{def}}{=} 1/2^n .
\end{aligned}$$

First note that the construction can indeed be carried in polynomial time—remember that p is encoded in binary. Second, $\mathcal{G}$ is visibly right-linear by construction, and also convergent because every derivation is of length $n + 2$ and every string has finitely many derivations.

It remains to show that φ is satisfiable if and only if there exists w in Σ^* such that $\langle [\![\mathcal{G}]\!], w\rangle \geq p$. Note that any string w with $\langle [\![\mathcal{G}]\!], w\rangle > 0$ is necessarily of form $\$v_1 \cdots v_n\$$ with each v_j in $\{0, 1\}$, i.e. describes a valuation for φ.

Observe that, for each clause C_i and each string $w = \$v_1 \cdots v_n\$$, w describes a valuation $V_w\colon x_j \mapsto v_j$ that

- either satisfies C_i, and then the corresponding string w has a single derivation π_w (the one that uses $A_{i,j-1} \xrightarrow{1/2} v_j B_j$ for the lowest index j such that $x_j \mapsto v_j \models C_i$); this derivation has probability $\rho(\pi_w) = 1/(k2^n)$,

- or does not satisfies C_i, and there is a single derivation, which must use the production $A_{i,n} \xrightarrow{0} \$$, and is thus of probability 0.

Therefore, if φ is satisfiable, i.e. if there exists V that satisfies all the clauses, then the corresponding string w_V has probability $\sum_{i=1}^{k} 1/(k2^n) = p$. Conversely, if φ is not satisfiable, then any w with $\langle [\![\mathcal{G}]\!], w\rangle > 0$ is of form $\$v_1 \cdots v_n\$$ and describes an assignment $V_w\colon x_j \mapsto v_j$ that does not satisfy at least one of the clauses, thus has a total probability $\rho(w) < p$. □

Corollary 6.10. *MPS and BMPS for proper and consistent right-linear PCFGs are NP-hard.*

Proof. If suffices to reduce and normalize the PCFG constructed in Theorem 6.9. Because every derivation is of bounded length, the computation of the partition function for $\mathcal{G}$ converges in polynomial time, and the grammar can be normalized in polynomial time.

Let us nevertheless perform those computations by hand as an exercise. For instance, for all $1 \leq j \leq n$,

$$Z(B_j) = 1 \; , \tag{6.13}$$

$$Z(S) = \frac{1}{k} \sum_{i=1}^{k} Z(A_{i,0}) \; . \tag{6.14}$$

We need to introduce some notation in order to handle the computation of $Z(A_{i,j})$. For each clause C_i, and each $0 \leq j \leq n$, let $q_{i,j}$ be the number of variables x_ℓ with $j < \ell \leq n$ that occur (positively or negatively) in C_i:

$$q_{i,j} \stackrel{\text{def}}{=} |\{x_\ell \in C_i \mid j < \ell \leq n\}| \; . \tag{6.15}$$

Claim 6.10.1. For all $1 \leq i \leq k$ and $0 \leq j \leq n$,

$$Z(A_{i,j}) = 1 - \frac{1}{2^{q_{i,j}}} \; .$$

Proof of Claim 6.10.1. Fix some $1 \leq i \leq k$; we proceed by induction over $n - j$. For the base case, $Z(A_{i,n}) = 0$ since the only production available is $A_{i,n} \xrightarrow{0} \$$. For the induction step, two cases arise:

1. $q_{i,j} = q_{i,j+1}$, i.e. when x_{j+1} does not appear in C_i. Then $Z(A_{i,j}) = 1/2 \cdot Z(A_{i,j+1}) + 1/2 \cdot Z(A_{i,j+1})$ by (6.5), and thus $Z(A_{i,j}) = Z(A_{i,j+1}) = 1 - 1/2^{q_{i,j+1}} = 1 - 1/2^{q_{i,j}}$ by induction hypothesis.

2. $q_{i,j} = 1 + q_{i,j+1}$, i.e. when x_{j+1} appears in C_i. Then $Z(A_{i,j}) = 1/2 \cdot Z(A_{i,j+1}) + 1/2 \cdot Z(B_{j+1})$, hence by (6.13) and the induction hypothesis, $Z(A_{i,j}) = 1/2 - 1/2^{q_{i,j+1}+1} + 1/2 = 1 - 1/2^{q_{i,j}}$. [6.10.1]

In particular, if we reduce from a 3SAT instance instead of any SAT instance, then $Z(A_{i,0}) = 7/8$ for all i, and thus $Z(S) = 7/8$.

Any nonterminal with probability mass 0 can be disposed of during the reduction phase, which can be performed in polynomial time. We use next (6.8) to normalize the grammar of Theorem 6.9, thereby obtaining a proper and consistent right-linear PCFG $\mathcal{G}'$ in polynomial time.

There remains the issue of computing an appropriate bound p' for this new grammar. By Remark 6.5, for any word w in Σ^*, $\langle [\![G]\!], w\rangle \geq p$ if and only if $\langle [\![G']\!], w\rangle \geq p/Z(S)$: we define therefore

$$p' \stackrel{\text{def}}{=} \frac{p}{Z(S)} \; . \tag{6.16}$$

□

Upper Bounds

Bounded Case Deciding BMPS is mostly straightforward: guess a string w in $\Sigma^{\leq b}$, compute the PCFG $\mathcal{G}'$ for w using Theorem 6.7 in polynomial time, and compute the partition function Z for $\mathcal{G}'$—which is non-recursive since $\mathcal{G}$ is acyclic—, which can be performed in polynomial time: then $\langle [\![\mathcal{G}]\!], w\rangle = Z(S')$. Hence:

Proposition 6.11. *BMPS is* NP-*complete.*

Right-Linear Case The case of MPS is more involved: there is no reason for the most probable string to be short.

Example 6.12 (Long Strings). De la Higuera and Oncina (2011) exhibit a right-linear grammar, for which the most probable string is of exponential length. Let m be a natural number and q a rational in $(0, 1)$, then the right-linear grammar with axiom $A_{q,m,0}$ and productions

$$A_{q,m,0} \xrightarrow{q} \varepsilon \qquad A_{q,m,0} \xrightarrow{1-q} aA_{q,m,1} \qquad A_{q,m,i} \xrightarrow{1} aA_{q,m,i+1 \bmod m}$$

for all $1 \leq i < m$ assigns a probability $\rho(a^{km}) = q(1-q)^k$ to a string of a's whose length is a multiple of m, and probability 0 to any other string. Consider now a set of primes $\{m_1, \dots, m_n\}$ and add the production

$$S \xrightarrow{1/n} A_{q,m_j,0}$$

for each m_j. This grammar has a size in $O(\sum_{j=1}^{n} m_j)$.

On the one hand, the probability of the string a^M of length $M \overset{\text{def}}{=} \prod_{i=j}^{n} m_j$ (which is exponential in $\sum_{j=1}^{n} m_j$) is

$$\begin{aligned}\rho(a^M) &= \sum_{j=1}^{n} \frac{1}{n} q(1-q)^{\frac{M}{m_j}} \\ &\geq \frac{q}{n} \sum_{j=1}^{n} (1-q)^M && \text{(since } \frac{M}{m_j} \leq M) \\ &= q(1-q)^M \;.\end{aligned}$$

On the other hand, a string a^ℓ of length $\ell < M$ is not accepted by at least one of the subgrammars, therefore its probability is at most

$$\rho(a^\ell) \leq q \frac{n-1}{n} \;.$$

Hence, a choice of q such that

$$q \leq 1 - \sqrt[M]{\frac{n-1}{n}}$$

ensures that no shorter string can have probability higher than $\rho(a^M)$.

Fortunately, we are also provided with a probability p in an instance of MPS. When taking this threshold into account, de la Higuera and Oncina (2013) can then provide a polynomial bound on the length of the most probable strings. The following proposition uses a normal form on right-linear PCFGs:

Definition 6.13. A right-linear PCFG $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$ is in **ε-free form** if $P \subseteq N \times (\Sigma \cup \Sigma N)$.

Exercise 6.7. Show that any (acyclic) right linear convergent PCFG can be put in ε-free form in polynomial time. **(∗∗)**

Proposition 6.14 (Probable Strings are Short). *Let $\mathcal{G} = \langle N, \Sigma, P, S, \rho \rangle$ be a right-linear reduced convergent PCFG in ε-free form and w be a sequence in Σ^* with $\rho(w) \geq p$. Then $|w| \leq \frac{Z(S)|N|^2}{p} + |N|$.*

Proof. Let $w = a_1 \cdots a_\ell$ be a string of length ℓ with $\rho(w) \geq p$ (with a_i in Σ for every i). Any derivation for w in the ε-free grammar $\mathcal{G}$ is necessarily of the form

$$S = A_1 \underset{\text{lm}}{\overset{p_1}{\Longrightarrow}} a_1 A_2 \underset{\text{lm}}{\overset{p_2}{\Longrightarrow}} a_1 a_2 A_3 \underset{\text{lm}}{\overset{p_3}{\Longrightarrow}} \cdots \underset{\text{lm}}{\overset{p_\ell}{\Longrightarrow}} a_1 \cdots a_\ell \tag{6.17}$$

using the productions $p_i = A_i \to a_i A_{i+1}$ for $1 \leq i < \ell - 1$ and $p_\ell = A_\ell \to a_\ell$. Define

$$D_w \overset{\text{def}}{=} \{\pi \in P^* \mid S \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}}{}^\star\, w\} \tag{6.18}$$

the set of derivations of w. Assuming some total ordering $\prec$ over the nonterminals in N, we write D_w^A for the subset of D_w where A is the nonterminal that occurs as left-hand side the most often, using $\prec$ to choose between ties. Then

$$D_w = \biguplus_{A \in N} D_w^A\ , \qquad\qquad \rho(w) = \sum_{\pi \in D_w} \rho(\pi) \geq p\ , \tag{6.19}$$

hence there exists A in N such that

$$\sum_{\pi \in D_w^A} \rho(\pi) \geq \frac{p}{|N|}\ . \tag{6.20}$$

In any derivation $\pi = p_1 \cdots p_\ell$ in D_w^A, A appears as left-hand side at least $\ell/|N|$ times. By removing a subderivation between two such occurrences, we obtain a derivation for a shorter sequence with at least the same probability. We call such shorter sequences *alternatives* for π; there are at least $\ell/|N| - 1$ alternatives, that we gather in a set $\mathrm{Alt}(\pi, A)$. Hence

$$\sum_{\pi' \in \mathrm{Alt}(\pi, A)} \rho(\pi') \geq \left(\frac{\ell}{|N|} - 1\right) \rho(\pi)\ . \tag{6.21}$$

We want to sum the probability mass of alternatives over all π in D_w^A; however, there might be common alternatives for different derivations π_1 and π_2. This is not an issue, as shown by the following claim:

Claim 6.14.1. Let π_1 and π_2 be two different derivations in D_w^A, and let π be a derivation in $\mathrm{Alt}(\pi_1, A) \cap \mathrm{Alt}(\pi_2, A)$. Then $\rho(\pi) \geq \rho(\pi_1) + \rho(\pi_2)$.

Hence

$$\sum_{\pi \in D_w^A} \sum_{\pi' \in \mathrm{Alt}(\pi, A)} \rho(\pi') \geq \sum_{\pi \in D_w^A} \left(\frac{\ell}{|N|} - 1\right) \rho(\pi) \geq \left(\frac{\ell}{|N|} - 1\right) \frac{p}{|N|}\ . \tag{6.22}$$

The probability mass on the left side of the previous inequality is contributed by strings different from w; hence, summing with the probability of w, we obtain

$$\left(\frac{\ell}{|N|} - 1\right) \frac{p}{|N|} + p \leq Z(S)$$

thus

$$\ell \leq \frac{(Z(S) - p)|N|^2}{p} + |N|\ ,$$

from which we deduce the desired bound since $Z(S) \geq \rho(w) \geq p$. □

Proof of Claim 6.14.1. [6.14.1]

Chapter 7

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

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

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^+:

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

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 7.2). Thus we are considering a **product-free** fragment. (See also Morrill, 1994; Steedman, 2000; Moortgat, 1997, for further extended sets of operators.)*

Definition 7.1. 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 7.1 and 7.2.

7.1 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

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

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

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

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 5.1 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.).

7.1.1 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

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

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

7.1.2 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

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

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

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

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

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

Exercise 7.1. Fill out the missing details of the proof of equivalence between AB categorial grammars and context-free grammars. **(∗∗)**

Exercise 7.2. The Dyck language D_n over n pairs of parentheses $a_i, \bar{a}_i$ is generated by the CFG with productions $\{S \rightarrow a_i S \bar{a}_i \mid 1 \leq i \leq n\} \cup \{S \rightarrow SS\} \cup \{S \rightarrow \varepsilon\}$. Give an AB categorial grammar for the language $D_n\$$ where \$ is an endmarker distinct from all the $a_i, \bar{a}_i$. **(∗∗)**

7.1.3 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.

7.2 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 7.1.1.

7.2.1 *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 ($\rightarrow$I) and ($\rightarrow$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 7.1.1. What we have just defined is a natural deduction version of the Lambek calculus.

Example 7.3. Here is a derivation of a type raising rule from Section 7.1.3:

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

Exercise 7.3. Show that the composition and Geach rules from Section 7.1.3 are also derivable in this natural deduction version of the Lambek calculus. **(∗)**

7.2.2 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.

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

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.

(∗) **Exercise 7.4.** Show that the decision procedure sketched above is in NP.

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).

7.2.3 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 7.4 (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 7.5.** Prove using Theorem 7.4 the equivalence of $\mathcal{C}$ and $\mathcal{G}$ as defined above.

Chapter 8

First-Order Semantics

In this chapter and the next two chapters, we survey a few aspects of computational semantics. Many formalisms can be used to define **meaning representations** of linguistic expressions. Here we focus on **first-order** representations, along with a few related ones.

See Chapter 17 of Jurafsky and Martin (2009) for more examples of meaning representations.

8.1 Formal Semantics

Concrete applications of computational semantics include for instance weeding out syntactic representations that map to unsatisfiable sentences, checking whether some form of **entailment** holds between two sentences (for instance for **summarisation** tasks), or **querying** databases with natural language interfaces (think airline reservation or weather forecasts), etc. The algorithmic aspects of these applications turn around the usual decision problems in model-theoretic aspects of logic: satisfiability, model-checking (i.e. satisfiability in presence of a database), and querying (an existing database).

Here by "database" we simply mean a (not necessarily finite) relational structure $\mathfrak{M} = \langle W, (R_i)_i \rangle$ where W is a *domain* of the various possible **entities**, and $(R_i^{(k_i)})_i$ is a *vocabulary*, where each $R_i^{(k_i)}$ is *interpreted* as a k_i-ary relation R_i over W, $k_i > 0$. We also allow for constants and denote them using nullary symbols like $R^{(0)}$; they are interpreted as single points in W. The first-order language thus allows to reason about **truths** regarding entities and their relations.

Example 8.1. For instance, assume our vocabulary includes $\mathit{John}^{(0)}$ as a constant denoting *John*, along with $\mathit{apple}^{(1)}$, $\mathit{red}^{(1)}$, and $\mathit{eat}^{(2)}$, we can associate the sentence

$$\exists x.\mathit{apple}^{(1)}(x) \wedge \mathit{red}^{(1)}(x) \wedge \mathit{eat}^{(2)}(\mathit{John}^{(0)}, x) \tag{8.1}$$

to the sentence *John eats a red apple*. Our interpretation might be s.t.

$$\begin{aligned} &a, j \in W && a \in \mathit{red} && a \in \mathit{apple} \\ &j = \mathit{John} && (j, a) \in \mathit{eat}\ , \end{aligned}$$

in which case the sentence is satisfiable using the assignment $\{x \mapsto a\}$.

An interesting consequence of this analysis is that paraphrases are typically associated with the same semantics: (8.1) could for instance be the formalisation of

John eats a red apple.
A red apple is eaten by John.
An apple that John eats is red.

8.1.1 Event Semantics

The kind of modelling that underlies Example 8.1 is a rather straightforward one: named entities (e.g. *John*, or *the President*) are interpreted as constants, properties (e.g. *red*, *apple*) as unary relations, and verbs as relations with an arity equal to the number of arguments present in their **subcategorisation frames**.

This however leads to some issues when determining the number of arguments for a particular instance of a verb, and drawing the appropriate inferences from our representations. Consider for instance the sentences

> John eats.
> John eats a red apple.
> John eats an apple in a park.
> John eats in a park.
> John slowly eats a red apple in a park.

Using the approach of Example 8.1, we need to introduce several relations $eat^{(i)}$ largely beyond the simple choice between the intransitive $eat_1^{(1)}$ and transitive $eat_2^{(2)}$ forms of *eat*:

$$eat_1^{(1)}(John^{(0)}) \tag{8.2}$$

$$\exists x.eat_2^{(2)}(John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \tag{8.3}$$

$$\exists xy.eat_3^{(3)}(John^{(0)}, x, y) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{8.4}$$

$$\exists y.eat_4^{(2)}(John^{(0)}, y) \wedge park^{(1)}(y) \tag{8.5}$$

$$\exists xy.eat_5^{(4)}(John^{(0)}, x, y, slowly^{(0)}) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{8.6}$$

where basically any extra **modifier** also necessitates a new variant of *eat*.

How can we relate all the variations of *eat* so that e.g. (8.6) entails each of (8.2–8.5)? One possibility is to add explicit meaning postulates like

$$\forall jxy.eat_3^{(3)}(j, x, y) \supset eat_2^{(2)}(j, x) \tag{8.7}$$

$$\forall jx.eat_2^{(2)}(j, x, y) \supset eat_1^{(1)}(j) \tag{8.8}$$

$$\dots \tag{8.9}$$

Similarly, we could treat *slowly* and the locative *in* as modal operators and rewrite (8.6) as

$$\exists xy.in^{(2)}(slowly^{(1)}(eat_2^{(2)}(John^{(0)}, x)), y) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{8.10}$$

along with the schemata

$$\forall Py.in^{(2)}(P, y) \supset P \tag{8.11}$$

$$\forall P.slowly^{(1)}(P) \supset P \tag{8.12}$$

where P ranges over formulæ. Of course there is no particular reason not to choose

$$\exists xy.slowly^{(1)}(location^{(2)}(eat_2^{(2)}(John^{(0)}, x), y)) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{8.13}$$

instead, and proving the equivalence of (8.10) and (8.13) would require yet more machinery. (We will however return to modal operators later in Section 8.3.)

As we can see, this solution scales rather poorly. Another possibility is to pick a very general version of *eat*, like eat_5, and express the simpler versions with existentially quantified arguments:

$$eat_1^{(1)}(j) \overset{\text{def}}{=} \exists xya.eat_5^{(4)}(j,x,y,a) \tag{8.14}$$

$$eat_2^{(2)}(j,x) \overset{\text{def}}{=} \exists ya.eat_5^{(4)}(j,x,y,a) \tag{8.15}$$

$$eat_3^{(3)}(j,x,y) \overset{\text{def}}{=} \exists a.eat_5^{(4)}(j,x,y,a) \tag{8.16}$$

$$eat_4^{(2)}(j,y) \overset{\text{def}}{=} \exists ya.eat_5^{(4)}(j,x,y,a)\ . \tag{8.17}$$

However, while it seems reasonable that the event denoted by *John eats* has an implicit object and location, there is no particular reason for it to be performed *slowly* or *quickly*, and it could also occur *at noon* or *at dawn*, necessitating yet another argument slot.

A solution is to use a two-sorted domain that differentiates between **events** and **entities**, and to add an explicit event argument to verbs:

$$\exists e.eat_1^{(2)}(e, John^{(0)}) \tag{8.18}$$

$$\exists ex.eat_2^{(3)}(e, John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \tag{8.19}$$

$$\exists exy.eat_2^{(3)}(e, John^{(0)}, x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y) \tag{8.20}$$

$$\exists ey.eat_1^{(2)}(e, John^{(0)}) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y) \tag{8.21}$$

$$\begin{aligned}\exists exy.eat_2^{(3)}(e, John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y)\\ \wedge\ slowly^{(1)}(e)\end{aligned} \tag{8.22}$$

This **Davidsonian** analysis succeeds in reducing the variations to the two main forms of *eat*. It also yields a rather more natural way of handling time and aspects modifiers like *slowly*. Note that the distinction between intransitive and transitive forms of verbs are better motivated than the ones between say (8.2) and (8.5): contrast for instance

See Davidson (1967).

I sank the Bismark.
I sank.

where the transitive usage does *not* imply the intransitive one.

8.1.2 Thematic Roles

The Davidsonian analysis can be further refined by employing **thematic roles**: instead of seeing the intransitive form $eat_1^{(2)}$ and the transitive one $eat_2^{(3)}$ as two wholly different relations, we can further refine them using a fixed set of thematic relations between events and entities:

*This is known as a **neo-Davidsonian** analysis (Parsons, 1990).*

$$\exists e.eat^{(1)}(e) \wedge agent^{(2)}(e, John^{(0)}) \tag{8.23}$$

$$\exists ex.eat^{(1)}(e) \wedge agent^{(2)}(e, John^{(0)}) \wedge patient^{(2)}(e,x) \wedge apple^{(1)}(x) \tag{8.24}$$

correspond to the two sentences *John eats* and *John eats an apple* respectively. The earlier issue with *sank* is avoided by changing the nature of the relation between

Role	Typical use
agent	*John eats*
patient	*John eats an apple.*
experiencer	*John regrets his actions.*
	The crisis worries John.
cause	*The crisis worries John.*
	John regrets his behaviour.
theme	*John asks a question.*
	John gives Mary a kiss.
beneficiary	*John gives Mary a kiss.*

Table 8.1: A basic set of thematic roles.

the subject and the verb:

$$\exists e.\mathit{sink}^{(1)}(e) \wedge \mathit{agent}^{(2)}(e, I^{(0)}) \wedge \mathit{patient}^{(2)}(e, \mathit{Bismark}^{(0)}) \tag{8.25}$$

$$\exists e.\mathit{sink}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, I^{(0)}) \tag{8.26}$$

The definition of a fixed set of thematic roles and how to classify the different uses are of course problematic; Table 8.1 proposes a very simple account.

For the sake of simplicity, we will not explicitly use event semantics and thematic roles in the remainder of the notes; the reader might convince herself that it is always possible.

8.2 A Dip into Description Logics

Section based on (Baader et al.*, 2007).*

Let us make a short detour through a family of logics primarily developed for knowledge representation. Basic **description logics**, similarly to the modal logics we will see in Section 8.3, can be translated into first-order logic, so their use does not yield any additional expressive power. Their interest is rather that they force us into well-behaved fragments of FO, where we are able to draw inferences and reason automatically.

8.2.1 A Basic Description Logic

See Schmidt-Schauß and Smolka (1991).

We will confine our interest to one of the most basic logics: $\mathcal{ALC}$ the "attributive concept language with complements." We describe the models of $\mathcal{ALC}$ as structures $\mathfrak{M} = \langle W, A, R\rangle$ where W is a *domain*, A is a finite set of atomic *concepts* $a \subseteq W$, and R is a finite set of *roles* $r \subseteq W^2$.

An $\boldsymbol{\mathcal{ALC}}$ **concept definition** C is defined by the syntax

$$C ::= \top \mid a \mid C \sqcap C \mid \neg C \mid \exists r.C$$

where a ranges over A and r over R. This syntax can be enriched by $\bot \overset{\text{def}}{=} \neg\top$, $C \sqcup D \overset{\text{def}}{=} \neg(\neg C \sqcap \neg D)$, and $\forall r_j.C \overset{\text{def}}{=} \neg\exists r_j.\neg C$. A concept defines a subset $[\![C]\!]^{\mathfrak{M}}$ of a model $\mathfrak{M}$:

$$[\![\top]\!]^{\mathfrak{M}} \overset{\text{def}}{=} W \qquad [\![a]\!]^{\mathfrak{M}} \overset{\text{def}}{=} a$$

$$[\![C \sqcap D]\!]^{\mathfrak{M}} \overset{\text{def}}{=} [\![C]\!]^{\mathfrak{M}} \cap [\![D]\!]^{\mathfrak{M}} \qquad [\![\neg C]\!]^{\mathfrak{M}} \overset{\text{def}}{=} W \setminus [\![C]\!]^{\mathfrak{M}}$$

$$[\![\exists r.C]\!]^{\mathfrak{M}} \overset{\text{def}}{=} r^{-1}([\![C]\!]^{\mathfrak{M}})\ .$$

The basic questions one might ask on concepts are **consistency** ones, i.e. whether there exists a model $\mathfrak{M}$ such that $[\![C]\!]^{\mathfrak{M}}$ is non-empty. An especially useful case is that of an **inclusion** $C \sqsubseteq D$, i.e. the inconsistency of $C \sqcap \neg D$.

Examples Consider the sentence *Every man loves a woman.* Its most common semantic reading can be formalised in first-order logic as

$$\forall y.man^{(1)}(y) \supset \exists x.woman^{(1)}(x) \wedge love^{(2)}(y,x) \tag{8.27}$$

It can also be formalised as a consistency question in $\mathcal{ALC}$:

$$\mathsf{Man} \sqsubseteq \exists\mathsf{love}.\mathsf{Woman} \tag{8.28}$$

where the binary relation $love^{(2)}$ is translated as a role, and the unary predicates $man^{(1)}$ and $woman^{(1)}$ as atomic concepts. The sentence *A man eats an apple* is captured by the consistency of

$$\mathsf{Man} \sqcap \exists\mathsf{eat}.\mathsf{Apple} \tag{8.29}$$

Extensions There are many extensions of $\mathcal{ALC}$ in the literature. For instance, description logics often allow for names in the form of **nominals** i, which are atomic concepts interpreted as singleton sets in the model. The syntax of concept definitions is then extended to allow $\{i\}$.

For instance, the sentence *John eats a red apple* can be checked by

$$\{John\} \sqsubseteq \exists\mathsf{eat}.(\mathsf{Apple} \sqcap \mathsf{Red}) \tag{8.30}$$

and the sentence *Helen of Troy is loved by every man in Greece* by

$$(\mathsf{Man} \sqcap \exists\mathsf{inhabit}.\{\mathit{Greece}\}) \sqsubseteq \exists\mathsf{love}.\{\mathit{Helen\ of\ Troy}\} \tag{8.31}$$

8.2.2 Translation into First-Order Logic

As hinted by the first-order and $\mathcal{ALC}$ formalisations in (8.27)–(8.28), there is a translation of $\mathcal{ALC}$ into first-order logic. Every nominal i is associated with a constant symbol $i^{(0)}$, every atomic concept a with a unary predicate $a^{(1)}$, and every role r with a binary relation $r^{(2)}$. Then, a concept definition C is translated into a first-order formula $\mathrm{ST}_x(C)$ with a single free variable x:

$$\begin{aligned} \mathrm{ST}_x(\top) &\stackrel{\text{def}}{=} x = x & \mathrm{ST}_x(a) &\stackrel{\text{def}}{=} a^{(1)}(x)\\ \mathrm{ST}_x(\{i\}) &\stackrel{\text{def}}{=} x = i^{(0)} & \mathrm{ST}_x(\neg C) &\stackrel{\text{def}}{=} \neg\mathrm{ST}_x(C)\\ \mathrm{ST}_x(C \sqcap D) &\stackrel{\text{def}}{=} \mathrm{ST}_x(C) \wedge \mathrm{ST}_x(D) & \mathrm{ST}_x(\exists r.C) &\stackrel{\text{def}}{=} \exists y.r^{(2)}(x,y) \wedge \mathrm{ST}_y(C) \end{aligned}$$

This satisfies $[\![C]\!]^{\mathfrak{M}} = \{w \in W \mid \mathfrak{M} \models_{x\mapsto w} \mathrm{ST}_x(C)\}$. Consistency questions are then translated into first-order sentences:

$$\mathrm{ST}(C) \stackrel{\text{def}}{=} \exists x.\mathrm{ST}_x(C) \qquad \mathrm{ST}(C \sqsubseteq D) \stackrel{\text{def}}{=} \forall x.\mathrm{ST}_x(C) \supset \mathrm{ST}_x(D)$$

These definitions result for instance in the following first-order semantics for (8.31):

$$\forall y.(man^{(1)}(y) \wedge inhabit^{(2)}(y, \mathit{Greece}^{(0)})) \supset love^{(2)}(y, \mathit{Helen\ of\ Troy}^{(0)}) \tag{8.32}$$

Two important remarks can be made regarding this translation:

1. it only requires two distinct variables, and
2. every first-order quantifier is *guarded* by a binary relation symbol (corresponding to the $\mathcal{ALC}$ role).

Each of these conditions is enough to yield decidability of $\mathcal{ALC}$; see Section 8.4.

8.3 Modal Semantics

Modalities are a means of qualifying truth judgements. Modal operators capture the linguistic concepts of **tense**, **mood**, and **aspect**, and more generally modifiers: in

John is ____ happy.

we can insert instead of the blank any of *necessarily*, *possibly*, *known by me to be*, *now*, *then*, ... Modal logic offer a unified framework to study such modifiers.

8.3.1 *Background:* Modal Logic

See (Blackburn et al.*, 2001).*

A **frame** is a couple $\mathfrak{F} = \langle W, R\rangle$ where W is a non-empty set of *worlds* and R a binary relation over W. A **model** is a couple $\mathfrak{M} = \langle \mathfrak{F}, V\rangle = \langle W, R, V\rangle$ where $\mathfrak{F}$ is a frame and V is a valuation from a set of *atomic propositions* A to subsets of W.

Basic Modal Language Given a set A of atomic propositions, a **(basic) modal formula** φ is defined by the syntax

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond\varphi$$

where p ranges over A. The $\Box$ modality is defined as the dual of $\Diamond$:

$$\Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi\ .$$

A formula *satisfies* a model $\mathfrak{M}$ in a world w of W, written $\mathfrak{M}, w \models \varphi$, in the following inductive cases:

$$\begin{array}{ll} \mathfrak{M}, w \models \top & \text{always} \\ \mathfrak{M}, w \models p & \text{iff } w \in V(p) \\ \mathfrak{M}, w \models \neg\varphi & \text{iff } \mathfrak{M}, w \not\models \varphi \\ \mathfrak{M}, w \models \varphi \vee \varphi' & \text{iff } \mathfrak{M}, w \models \varphi \text{ or } \mathfrak{M}, w \models \varphi' \\ \mathfrak{M}, w \models \Diamond\varphi & \text{iff } \exists w', w\, R\, w' \text{ and } \ M, w' \models \varphi\ . \end{array}$$

Logics The diamond $\Diamond$ and box $\Box$ modalities can take many different interpretations. For instance,

- in **alethic logic**, we reason about possible truths: $\Diamond\varphi$ denotes that "*possibly* φ" and $\Box\varphi$ "*necessarily* φ". If we follow Leibniz and imagine multiple "possible worlds" in an universe W, something "possible" is one holding in at least one possible world, and something "necessary" holds in all possible worlds. In order to obtain such semantics, we should work on **total frames** where $w\, R\, w'$ for all w, w' in W.

- In **epistemic logic**, we reason about knowledge of agents (mind the difference with beliefs): instead of writing $\Box\varphi$ to denote the fact that "the agent *knows* φ", we write $K\varphi$. Epistemic logic is typically interpreted over transitive, symmetric, and reflexive frames, i.e. where R is an equivalence relation. If the knowledge of several agents is to be modelled, we can introduce multiple relations R_a and modalities K_a, one for each agent a.

- In the **basic temporal logic**, $\Diamond\varphi$ denotes that "at some *future* point, φ holds", written $F\varphi$. Its dual $G\varphi$ means that in all future points, φ holds. Its **converse** P allows to reason about the past, and is defined by $\mathfrak{M}, w \models P\varphi$ iff there exists $w' \mathrel{R} w$ s.t. $\mathfrak{M}, w' \models \varphi$, with dual H. One expects R to be a transitive, irreflexive relation. An important distinction arises between **linear time** and **branching time** frames: in the first case, there is a unique possible future, while in the second case there exist multiple different futures.

In branching frames, the $\Diamond$ modality becomes similar to the EF modality of CTL (thus $\Box$ is similar to AG). A similar distinction between linear past and branching past can be made (Kupfermana et al.*, 2012).*

Exercise 8.1 (Basic Axiom)**.** Show that $\mathbf{K} : \Box(\varphi \supset \psi) \supset (\Box\varphi \supset \Box\psi)$ is **valid**, i.e. for any model $\mathfrak{M}$ and any world w of W, $\mathfrak{M}, w \models \mathbf{K}$. **(∗)**

Exercise 8.2 (Transitive Frames)**.** Show that, if R is transitive, then $\mathbf{4} : \Diamond\Diamond\varphi \supset \Diamond\varphi$ is valid. **(∗)**

Exercise 8.3 (Epistemic Frames)**.** Prove the following implications for all modal formulæ φ when R is an equivalence relation: **(∗)**

T : $\Box\varphi \supset \varphi$—in epistemic logic, if indeed an agent *really* knows something, then it must be true—,

4 : $\Box\varphi \supset \Box\Box\varphi$—in epistemic logic again, an agent has *introspection* about its own knowledge—,

B : $\varphi \supset \Box\Diamond\varphi$—in epistemic logic again, a truth is known by the agent as possibility compatible with her knowledge.

Modal Languages As seen with our examples, the basic modal language can be extended to multiple modalities and underlying relations; in particular PDL defined in Section 4.2 is a modal language with an unbounded number of binary relations. A **modal similarity type** O is a ranked alphabet of modal operators $\triangle$ of arity $r(\triangle)$. A **modal formula** is then defined as

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \triangle(\varphi_1, \ldots, \varphi_{r(\triangle)})$$

where p ranges over A and $\triangle$ over O. Its semantics are defined over ***O*-frames** $\mathfrak{F} = \langle W, (R_\triangle)_{\triangle\in O}\rangle$ where each $R_\triangle$ relation is of arity $r(\triangle)+1$, by

$$\begin{aligned}\mathfrak{M}, w \models \triangle(\varphi_1, \ldots, \varphi_{r(\triangle)}) \quad & \text{iff } \exists w_1, \ldots, w_{r(\triangle)} \in W.(w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle \\ & \text{and } \forall 1 \leq i \leq r(\triangle).\mathfrak{M}, w_i \models \varphi_i\ .\end{aligned}$$

Exercise 8.4 ($\mathcal{ALC}$ as a Modal Language)**.** Provide a consistency-preserving translation from $\mathcal{ALC}$ concepts into modal formulæ. **(∗)**

Standard Translation Modal languages have a **standard translation** into first-order logic over the vocabulary $\langle (R_\triangle)_{\triangle\in O}, (P_p)_{p\in A}\rangle$ where $P_p = V(p)$:

$$\begin{aligned}\mathrm{ST}_x(p) &\stackrel{\text{def}}{=} P_p(x)\\ \mathrm{ST}_x(\top) &\stackrel{\text{def}}{=} (x = x)\\ \mathrm{ST}_x(\neg\varphi) &\stackrel{\text{def}}{=} \neg\mathrm{ST}_x(\varphi)\\ \mathrm{ST}_x(\varphi \vee \varphi') &\stackrel{\text{def}}{=} \mathrm{ST}_x(\varphi) \vee \mathrm{ST}_x(\varphi')\\ \mathrm{ST}_x(\triangle(\varphi_1, \ldots, \varphi_{r(\triangle)})) &\stackrel{\text{def}}{=} \exists x_1 \ldots x_{r(\triangle)}.R_\triangle(x, x_1, \ldots, x_{r(\triangle)}) \wedge \bigwedge_{i=1}^{r(\triangle)} \mathrm{ST}_{x_i}(\varphi_i)\end{aligned}$$

is a FO formula with a free variable x equivalent to φ: $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M} \models_{x \mapsto w} \mathrm{ST}_x(\varphi)$. By reusing variables in the standard translation, we can use only $(n+1)$ first-order variables if $\max_{\triangle \in O}(r(\triangle)) = n$.

See Blackburn et al. *(2001, Chapter 2).*

Bisimulations and Modal Invariance

Definition 8.2 (Bisimulations). Let O be a modal similarity type and let $\mathfrak{M} = \langle W, (R_\triangle)_{\triangle \in O}, V \rangle$ and $\mathfrak{M}' = \langle W, (R'_\triangle)_{\triangle \in O}, V' \rangle$ be two O-models. A non-empty relation $Z \subseteq W \times W'$ is a **bisimulation** between $\mathfrak{M}$ and $\mathfrak{M}'$ if for all w, w' s.t. $w \, Z \, w'$,

1. $\{p \in A \mid w \in V(p)\} = \{p' \in A \mid w' \in V'(p')\}$,
2. if $(w, w_1, \dots, w_{r(\triangle)}) \in R_\triangle$, then there are $w'_1, \dots, w'_{r(\triangle)}$ in W' s.t. $w_i \, Z \, w'_i$ for all $1 \le i \le r(\triangle)$ and $(w', w'_1, \dots, w'_{r(\triangle)}) \in R'_\triangle$, and
3. if $(w', w'_1, \dots, w'_{r(\triangle)}) \in R'_\triangle$, then there are $w_1, \dots, w_{r(\triangle)}$ in W s.t. $w_i \, Z \, w'_i$ for all $1 \le i \le r(\triangle)$ and $(w, w_1, \dots, w_{r(\triangle)}) \in R_\triangle$.

We say that w and w' are **bisimilar**, noted $w \underline{\leftrightarrow} w'$, if there exists a bisimulation Z s.t. $w \, Z \, w'$.

Proposition 8.3 (Invariance for Bisimulation). *Let O be a modal similarity type, and $\mathfrak{M}$ and $\mathfrak{M}'$ be O-models. Then, for every w in W and w' in W' with $w \underline{\leftrightarrow} w'$, and every modal formula φ, $\mathfrak{M}, w \models \varphi$ iff $\mathfrak{M}, w' \models \varphi$.*

Proof. The proof proceeds by induction on φ. The case where φ is an atomic proposition is a consequence of (1) in Definition 8.2, the case where φ is $\top$ is trivial, and the cases of Boolean connectives follow from the induction hypothesis. For a formula of form $\triangle(\varphi_1, \dots, \varphi_{r(\triangle)})$:

$$\mathfrak{M}, w \models \triangle(\varphi_1, \dots, \varphi_{r(\triangle)})$$

implies $\exists w_1, \dots, w_{r(\triangle)} \in W. (w, w_1, \dots, w_{r(\triangle)}) \in R_\triangle \wedge \forall 1 \le i \le r(\triangle). \mathfrak{M}, w_i \models \varphi_i$

implies $\exists w'_1, \dots, w'_{r(\triangle)} \in W'. (w', w'_1, \dots, w'_{r(\triangle)}) \in R_\triangle \wedge \forall 1 \le i \le r(\triangle). \mathfrak{M}', w'_i \models \varphi_i$

(by ind. hyp. and (2))

implies $\mathfrak{M}', w' \models \triangle(\varphi_1, \dots, \varphi_{r(\triangle)})$,

and the converse implication holds symmetrically thanks to (3) and the induction hypothesis. □

It is worth mentioning that the converse does not hold in general: there exist models which are undistinguishable by modal formulæ but not bisimilar. In the case of models with **finite image** however, where for every $R_\triangle$ and w

$$\{(w_1, \dots, w_{r(\triangle)}) \mid (w, w_1, \dots, w_{r(\triangle)}) \in R_\triangle\}$$

is finite, the converse holds: let us define the **modal equivalence** relation $w \leftrightsquigarrow w'$ as holding iff w and w' are indistinguishable, i.e.

$$\{\varphi \mid \mathfrak{M}, w \models \varphi\} = \{\varphi' \mid \mathfrak{M}', w' \models \varphi'\} \,.$$

Theorem 8.4 (Hennessy-Milner Theorem). *Let O be a modal similarity type, and $\mathfrak{M}$ and $\mathfrak{M}'$ be O-models with finite image. If $w \leftrightsquigarrow w'$, then $w \underline{\leftrightarrow} w'$.*

Proof. Let us prove that modal equivalence is a bisimulation relation. Condition (1) holds since a difference in labelling would be witnessed by propositional formulæ. For condition (2), assume $w \leftrightsquigarrow w'$ and $(w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle$, and assume that there do not exist $w'_1, \ldots, w'_{r(\triangle)}$ satisfying (2). The image set $S' = \{(w'_1, \ldots, w'_{r(\triangle)}) \mid (w', w'_1, \ldots, w'_{r(\triangle)}) \in R'_\triangle\}$ is finite, and non empty since otherwise $\mathfrak{M}, w \models \triangle(\top, \ldots, \top)$ but $\mathfrak{M}', w' \not\models \triangle(\top, \ldots, \top)$. Thus S' is a finite set $\{(w'_{1,1}, \ldots, w'_{1,r(\triangle)}), \ldots, (w'_{n,1}, \ldots, w'_{n,r(\triangle)})\}$ where, by assumption, for every $1 \leq j \leq n$, there exists $1 \leq i \leq r(\triangle)$ s.t. $w_i \not\leftrightsquigarrow w'_{j,i}$, i.e. there exists a formula $\varphi_{j,i}$ s.t. $\mathfrak{M}, w_i \models \varphi_{j,i}$ but $\mathfrak{M}', w'_{j,i} \not\models \varphi_{j,i}$. But then

$$\mathfrak{M}, w \models \triangle\left(\bigwedge_{1\leq j\leq n} \varphi_{j,1}, \ldots, \bigwedge_{1\leq j\leq n} \varphi_{j,r(\triangle)}\right)$$

$$\mathfrak{M}', w' \not\models \triangle\left(\bigwedge_{1\leq j\leq n} \varphi_{j,1}, \ldots, \bigwedge_{1\leq j\leq n} \varphi_{j,r(\triangle)}\right),$$

in contradiction with $w \leftrightsquigarrow w'$. The argument for condition (3) is symmetric. □

The van Benthem Characterisation Theorem We saw earlier that any modal formula has a standard translation into first-order. A converse statement holds for a semantically restricted class of first-order formulæ.

Let us say that a first-order formula $\psi(x)$ in FO$((R_\triangle)_{\triangle\in O}, (P_p)_{p\in A})$ with one free variable x is **invariant for bisimulation** if for all models $\mathfrak{M}$ and $\mathfrak{M}'$, all states w in $\mathfrak{M}$ and w' in $\mathfrak{M}'$ in bisimulation, we have $\mathfrak{M} \models_{x\mapsto w} \psi(x)$ iff $\mathfrak{M} \models_{x\mapsto w'} \psi(x)$.

Theorem 8.5 (van Benthem Characterisation Theorem). *Let $\psi(x)$ be a first-order formula in FO$((R_\triangle)_{\triangle\in O}, (P_p)_{p\in A})$ with one free variable x. Then $\psi(x)$ is invariant for bisimulation iff it is equivalent to the standard translation of a modal formula.*

See Otto (2004).

Decision Problems Many classes of frames yield modal logics with decidable satisfiability and model-checking problems, even when the corresponding first-order theory is undecidable, or suffers from much larger decision complexities. Many logics have NP-complete satisfaction problems, while the basic modal language is PSPACE-complete. Model-checking of finite models is usually P-complete.

See Blackburn et al. *(2001, Chapter 6).*

8.3.2 First-Order Modal Logic

In order to work with both modal operators and first-order semantics as in Section 8.1, we introduce a mixed logic, **first-order modal logic** (FOML). For simplicity we give the definitions for the basic modal operator and not the fully general modal logic. The syntax of the logic over a vocabulary $\langle (R_i)_i \rangle$ of k_i-ary symbols is

$$\varphi ::= x = y \mid R_i(x_1, \ldots, x_{k_i}) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \exists x.\varphi$$

with $x, x_1, \ldots, x_{k_i}, y$ ranging over an infinite countable set of variables $\mathcal{X}$.

We consider structures $\mathfrak{M} = \langle W, R, D, I \rangle$ where $\langle W, R \rangle$ is a *frame*, D is a *domain* function from W to non-empty sets, and I is an *interpretation* function mapping each R_i with arity $k_i > 0$ and world w from W into a k_i-ary relation $I(R_i)(w)$ over $D(w)$ (constants are handled similarly). The **domain** of the model is $\mathfrak{D} = \bigcup_{w\in W} D(w)$. A *valuation* is a partial mapping from variables in $\mathcal{X}$ to the domain

$\mathfrak{D}$. The satisfaction of a formula by a model $\mathfrak{M}$ at a world w for a valuation ν is defined inductively by

$$\begin{array}{ll}
\mathfrak{M}, w \models_\nu x = y & \text{iff } \nu(x) = \nu(y) \\
\mathfrak{M}, w \models_\nu R_i(x_1, \ldots, x_{k_i}) & \text{iff } (\nu(x_1), \ldots, \nu(x_n)) \in I(R_i)(w) \\
\mathfrak{M}, w \models_\nu \neg\varphi & \text{iff } \mathfrak{M}, w \not\models_\nu \varphi \\
\mathfrak{M}, w \models_\nu \varphi \wedge \varphi' & \text{iff } \mathfrak{M}, w \models_\nu \varphi \text{ and } \mathfrak{M}, w \models_\nu \varphi' \\
\mathfrak{M}, w \models_\nu \Diamond\varphi & \text{iff } \exists w' \in W. w\, R\, w' \text{ and } \mathfrak{M}, w' \models_\nu \varphi \\
\mathfrak{M}, w \models_\nu \exists x.\varphi & \text{iff } \exists e \in D(w). \mathfrak{M}, w \models_{\nu[x \leftarrow e]} \varphi\ .
\end{array}$$

See also the entry on actualism in the Stanford Encyclopedia of Philosophy.

The domain $D(w)$ denotes the set of objects in the world w; this set is allowed to vary from world to world, i.e. the semantics allows a **varying domain**. Because we restrict the domain of quantified variables to the current domain, we take an **actualist quantification**. A **constant domain** semantics instead considers $D(w) = \mathfrak{D}$ for all w in W; the resulting semantics is also called **possibilist quantification**.

Unlike the domain, valuations are **rigid** in this semantics: the value of a variable does not depend on the current world. In the case of varying domains, it can potentially refer to an object from another world but not existing in the current one (but cannot do much with it). In the following we will use constant domains.

Example 8.6 (First-order temporal logic)**.** Let us consider some very simple examples in the temporal extension of first-order logic: we can model the meaning of the following sentence

John will eat an apple.

as

$$\exists a. apple^{(1)}(a) \wedge F(eat_2^{(2)}(John^{(0)}, a))\ . \tag{8.33}$$

Observe however that, in an actualist view, this reading implies the existence of the apple John will eventually eat in the current instant; the formula might not be satisfied by the model if no appropriate object a on which $apple(a)$ holds can be found. Another reading would be

$$F(\exists a. apple^{(1)}(a) \wedge eat_2^{(2)}(John^{(0)}, a))\ . \tag{8.34}$$

8.4 Decidability

See Börger et al. *(1997).*

In moderns terms, the **Entscheidungsproblem** or **classical decision problem** of Hilbert asks, given a first-order formula ψ, whether it is satisfiable. Church and Turing famously proved in the 1930s that the problem is undecidable, and a long line of research has established the decidability status of many fragments of first-order logic. Notably, the decidability status is known for all the *prefix classes* for formulæ in prenex normal form.

For instance, the semantic reading

$$\exists x. woman^{(1)}(x) \wedge \forall y. man^{(1)}(y) \supset love^{(2)}(y, x) \tag{8.35}$$

for *Every man loves a woman*—to be contrasted with (8.27)—belongs to the $\exists^*\forall^*$ class shown decidable by Bernays and Schönfinkel and NExpTime-complete by Lewis (1980). It also belongs to the two-variable fragment FO^2, which was shown decidable by Mortimer and NExpTime-complete by Grädel, Kolaitis, and Vardi (1997). The standard translations of $\mathcal{ALC}$ and of basic modal logic also yield FO^2 formulæ, and they are therefore decidable (they are actually PSpace-complete).

8.4.1 The Guarded Fragment

We are going to look more closely at one of the decidable fragments of first-order logic, called the k-variable **guarded fragment** (GFO^k). The satisfiability problem in GFO^k is ExpTime-complete (Grädel and Walukiewicz, 1999); in fact this complexity also holds for the fixed-point extension of GFO^k.

The section follows Grädel (2002). The guarded fragment has been advanced by Andréka, van Benthem, and Németi (1998) as an explanation for the good model- and complexity-theoretic properties of modal logics.

Let $\mathcal{X} \overset{\text{def}}{=} \{x_1, \ldots, x_k\}$ be the set of variables. A *guarded formula* over a vocabulary $(R_i^{(k_i)})_i$ is defined syntactically by

$$\psi ::= x = y \mid R_i^{(k_i)}(\mathbf{z}) \mid \neg\psi \mid \psi \wedge \psi \mid \exists \mathbf{y}.\alpha(\mathbf{x}, \mathbf{y}).\psi(\mathbf{y})$$

where x, y are variables in $\mathcal{X}$, $R_i^{(k_i)}$ is a relation symbol of arity k_i, $\mathbf{z}$ is a k_i-tuple of variables in $\mathcal{X}$, and $\mathbf{x}, \mathbf{y}$ denote tuples of variables in $\mathcal{X}$, $\alpha(\mathbf{x}, \mathbf{y})$ a positive atomic formula, and $\psi(\mathbf{x}, \mathbf{y})$ a GFO^k formula with FV$(\psi) \subseteq$ FV$(\alpha) = \mathbf{x} \cup \mathbf{y}$. Guarded universal quantification $\forall \mathbf{y}.\alpha(\mathbf{x}, \mathbf{y}) \supset \psi(\mathbf{x}, \mathbf{y})$ is defined by duality.

For example, the formula (8.27) is in GFO^2: $man^{(1)}(y)$ guards the universal quantification and $love^{(2)}(y, x)$ guards the existential quantification. By contrast, (8.35) is not in GFO^2: the universal quantification $\forall y.man^{(1)}(y) \supset love^{(2)}(y, x)$ is not guarded. Observe more generally that the standard translations of $\mathcal{ALC}$ or basic modal formulæ are in GFO^2.

Guarded Bisimulations

Let $\mathfrak{M} = \langle W, (R_i)_i \rangle$ be a relational structure. A set $X = \{w_1, \ldots, w_n\} \subseteq W$ is **guarded** in $\mathfrak{M}$ if there exists a positive atomic formula $\alpha(x_1, \ldots, x_n)$ such that $\mathfrak{M} \models_{x_1 \mapsto w_1, \ldots, x_n \mapsto w_n} \alpha(x_1, \ldots, x_n)$. In particular, every singleton $\{w\}$ is guarded by $x = x$ and every hyperedge $\langle w_1, \ldots, w_{k_i} \rangle$ in the relation R_i is guarded by $R_i^{(k_i)}(x_1, \ldots, x_{k_i})$.

A **guarded-k-bisimulation** between two structures $\mathfrak{M}$ and $\mathfrak{M}'$ is a non-empty set I of partial isomorphisms $f: X \to X'$ from $\mathfrak{M}$ to $\mathfrak{M}'$, where $X \subseteq W$ and $X' \subseteq W'$ are guarded sets of cardinal at most k, such that the following condition is satisfied: for every $f: X \to X'$ in I,

1. for every guarded set $Y \subseteq W$ in $\mathfrak{M}$ of size at most k, there exists $g: Y \to Y'$ in I such that f and g agree on $X \cap Y$, and
2. for every guarded set $Y' \subseteq W'$ in $\mathfrak{M}'$ of size at most k, there exists $g: Y \to Y'$ in I such that f^{-1} and g^{-1} agree on $X' \cap Y'$.

As in the modal case, we write $\mathfrak{M} \underline{\leftrightarrow}_k \mathfrak{M}'$ if there exists a guarded-k-bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$. We also write $\mathfrak{M} \leftrightsquigarrow_k \mathfrak{M}'$ if for all GFO^k sentences ψ, $\mathfrak{M} \models \psi$ iff $\mathfrak{M}' \models \psi$. Proposition 8.3 can be extended to the case of guarded-k-bisimilarity:

Proposition 8.7. *Let $\mathfrak{M}$ and $\mathfrak{M}'$ be two relational structures over the vocabulary $(R_i)_i$. If $\mathfrak{M} \underline{\leftrightarrow}_k \mathfrak{M}'$, then $\mathfrak{M} \leftrightsquigarrow_k \mathfrak{M}'$.*

Proof. Let I be a guarded-k-bisimulation between $\mathfrak{M}$ and $\mathfrak{M}'$. We show by induction on ψ in GFO^k that, if $\psi(\mathbf{x})$ has n free variables and there exist two n-tuples $\mathbf{a}$ in $\mathfrak{M}$ and $\mathbf{a}'$ in $\mathfrak{M}'$ such that $\mathfrak{M} \models_{\mathbf{x} \mapsto \mathbf{a}} \psi(\mathbf{x})$ but $\mathfrak{M}' \not\models_{\mathbf{x} \mapsto \mathbf{a}'} \psi(\mathbf{x})$, then there is no partial isomorphism f in I with $f: \mathbf{a} \mapsto \mathbf{a}'$. This will entail that I is empty when $n = 0$, i.e. in the case of a sentence ψ in GFO^k, thus contradicting $\mathfrak{M} \underline{\leftrightarrow}_k \mathfrak{M}'$.

For an atomic formula $\psi(\mathbf{x}) = \alpha(\mathbf{x})$ where $\mathfrak{M} \models_{\mathbf{x} \mapsto \mathbf{a}} \alpha(\mathbf{x})$ but $\mathfrak{M}' \not\models_{\mathbf{x} \mapsto \mathbf{a}'} \alpha(\mathbf{x})$, assume that there exists f in I mapping $\mathbf{a}$ to $\mathbf{a}'$. Then by condition (1), there must exist g in I with domain $\mathbf{a}$ that agrees with f on $\mathbf{a}$, i.e. $g\colon \mathbf{a} \mapsto \mathbf{a}'$. This would entail $\mathfrak{M}' \not\models_{\mathbf{x}\mapsto\mathbf{a}'} \alpha(\mathbf{x})$, a contradiction.

For a conjunction $\psi(\mathbf{x}_1, \mathbf{x}_2) = \psi_1(\mathbf{x}_1) \wedge \psi_2(\mathbf{x}_2)$ where $\mathfrak{M} \models_{\mathbf{x}_1\mapsto\mathbf{a}_1,\mathbf{x}_2\mapsto\mathbf{a}'_1} \psi(\mathbf{x}_1, \mathbf{x}_2)$ but $\mathfrak{M}' \not\models_{\mathbf{x}_1\mapsto\mathbf{a}'_1,\mathbf{x}_2\mapsto\mathbf{a}'_2} \psi(\mathbf{x}_1, \mathbf{x}_2)$, for some j in $\{1,2\}$, $\mathfrak{M}' \not\models_{\mathbf{x}_j\mapsto\mathbf{a}'_j} \psi_j(\mathbf{x}_j)$ and by induction hypothesis there is no f_j in I that maps $\mathbf{a}_j$ to $\mathbf{a}'_j$, and therefore no f in I that maps $\mathbf{a}_j$ to $\mathbf{a}'_j$ for all $j \in \{1,2\}$. The case of a negated formula is similarly immediate by induction hypothesis.

The interesting case is that of an existential quantification $\psi(\mathbf{x}) = \exists \mathbf{y}.\alpha(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}, \mathbf{y})$. Since $\mathfrak{M} \models_{\mathbf{x}\mapsto\mathbf{a}} \psi(\mathbf{x})$, there exists $\mathbf{b}$ in $\mathfrak{M}$ such that $\mathfrak{M} \models_{\mathbf{x}\mapsto\mathbf{a},\mathbf{y}\mapsto\mathbf{b}} \alpha(\mathbf{x}, \mathbf{y}) \wedge \varphi(\mathbf{x}, \mathbf{y})$. Suppose toward a contradiction that there exists f in I that maps $\mathbf{a}$ to $\mathbf{a}'$. By condition (1), since $\mathbf{a} \cup \mathbf{b}$ is guarded by $\alpha(\mathbf{x}, \mathbf{y})$, there exists g in I that maps $\mathbf{a}$ to $\mathbf{a}'$ and $\mathbf{b}$ to $\mathbf{b}'$. Then $\mathfrak{M}' \models_{\mathbf{x}\mapsto\mathbf{a}',\mathbf{y}\mapsto\mathbf{b}'} \alpha(\mathbf{x}, \mathbf{y})$ since g is a partial isomorphism, which entails that $\mathfrak{M}' \not\models_{\mathbf{x}\mapsto\mathbf{a}',\mathbf{y}\mapsto\mathbf{b}'} \varphi(\mathbf{x}, \mathbf{y})$, which together with the existence of g contradicts the induction hypothesis on φ. □

Models of Bounded Treewidth

An important model-theoretic property of $\mathcal{ALC}$ and the basic modal language is that they enjoy the *tree model property*: if a formula is satisfiable, then it has a tree model. In the case of GFO^k, we can generalise this idea to models of *treewidth* bounded by $k-1$, see Proposition 8.8. In the case where $k = 2$ (which is the case of $\mathcal{ALC}$ and the basic modal logic), we find again the tree model property.

See Robertson and Seymour (1986).

On an intuitive level, the treewidth of a structure tells how close to a tree the structure looks like. Trees and forests have treewidth 1, cycles have treewidth 2, etc. An example of a class of structures with unbounded treewidth is the class of $n \times n$ grids, each with treewidth n. Formally, the **treewidth** of a structure $\mathbf{M} = \langle W, (R_i^{(k_i)})_i \rangle$ is the minimal k such that there exists a tree t labelled by *bags* in $\{X \subseteq W \mid |X| \leq k+1\}$, such that

1. for every guarded set X in $\mathfrak{M}$ there exists a position u in $\operatorname{dom} t$ with $X \subseteq t(u)$, and

2. for every element a in $\mathfrak{M}$, the set of nodes $\{u \in \operatorname{dom} t \mid b \in t(u)\}$ is connected in t using the child relation $\downarrow$.

For each u in $\operatorname{dom} t$, $t(u)$ induces a substructure $\mathfrak{T}(u) \subseteq \mathfrak{M}$ of cardinality at most $k+1$. The tree t is called a *tree decomposition* of $\mathfrak{M} = \bigcup_{u\in\operatorname{dom} t} \mathfrak{T}(u)$.

Consider a structure $\mathfrak{M}$. We are going to construct a guarded-k-bisimilar **unravelling** $\mathfrak{M}'$ with treewidth at most $k-1$. We construct for this two trees t and t' with the same domain $\operatorname{dom} t = \operatorname{dom} t'$ such that for each position u, $t(u)$ induces a guarded substructure $\mathfrak{T}(u) \subseteq \mathfrak{M}$ and $t'(u)$ a substructure $\mathfrak{T}'(u) \subseteq \mathfrak{M}'$ isomorphic to $\mathfrak{T}(u)$; then t' will be a tree decomposition of $\mathfrak{M}'$.

The root ε is labelled $\emptyset$ in both t and t'. Inductively, given a position u with $t(u) = \{a_1, \dots, a_r\}$ and $t'_u = \{a'_1, \dots, a'_r\}$, we create for every guarded set $\{b_1, \dots, b_s\}$ of size $s \leq k$ in $\mathfrak{M}$ a child node v of u such that $t(v) = \{b_1, \dots, b_s\}$ and $t'(v) = \{b'_1, \dots, b'_s\}$ defined for all $1 \leq i \leq s$ by $b'_i = a'_j$ if $b_i = a_j$ for some $1 \leq j \leq r$ and b'_i is a fresh element otherwise. Define accordingly the induced substructure $\mathfrak{T}'(v)$ to be isomorphic to the induced substructure $\mathfrak{T}(v)$, giving rise to a partial isomorphism $f_v\colon t(v) \to t'(v)$ when setting $f_v(b_i) \overset{\text{def}}{=} b'_i$. Finally, let $\mathfrak{M}' \overset{\text{def}}{=} \bigcup_{u\in\operatorname{dom} t'} \mathfrak{T}'(u)$.

Observe that the tree t' is a tree decomposition of $\mathfrak{M}'$. This entails that $\mathfrak{M}'$ has treewidth at most $k-1$. Furthermore, $\{f_u \mid u \in \operatorname{dom} t\}$ is a non-empty (note that the root ε gives rise to the empty isomorphism) set of partial isomorphisms

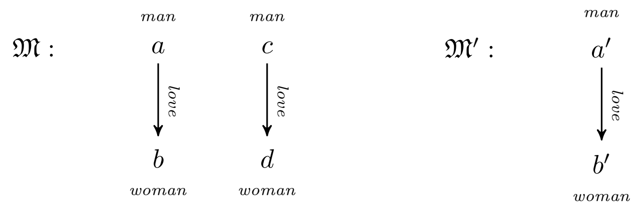


Figure 8.1: Two structures which are not guarded-2-bisimilar over the vocabulary $man^{(1)}, woman^{(1)}, love^{(2)}$.

between $\mathfrak{M}$ and $\mathfrak{M}'$, which satisfies the conditions of a guarded-k-bisimulation: $\mathfrak{M} \underline{\leftrightarrow}_k \mathfrak{M}'$. Hence, by Proposition 8.7:

Proposition 8.8. *If a sentence ψ in GFO^k has a model, then it has a model of treewidth at most $k-1$.*

Proposition 8.8 is instrumental in the proof of Grädel and Walukiewicz (1999) that the satisfiability problem for GFO^k is in ExpTime. More precisely, the idea is to reduce the problem to a modal μ-calculus satisfiability question over (infinite, countable) trees: given a GFO^k formula ψ, one can construct a modal μ-calculus formula φ which describes a tree decomposition of a model of ψ of treewidth at most $k-1$. The complexity then follows by adapting the results of Vardi (1998) on the emptiness problem for 2ATAs over infinite trees.

Limitations & Extensions

Although the guarded fragment includes many formulæ of interest in formal semantics, it is not comprehensive: (8.35) is an example of an unguarded formula. We can furthermore show that there is no equivalent formula in GFO. Observe that the two structures depicted in Figure 8.1 are guarded-2-bisimilar for the following set I of partial isomorphisms

*There are extensions on the guarded fragment that retain most of its model- and complexity-theoretic properties, e.g. the **guarded negation** fragment of Bárány, ten Cate, and Segoufin (2011). Those extensions do not solve the issues pointed here.*

$$f_{ab}\colon a \mapsto a', b \mapsto b' \qquad f_a\colon a \mapsto a' \qquad f_b\colon b \mapsto b'$$
$$f_{cd}\colon c \mapsto a', d \mapsto b' \qquad f_c\colon c \mapsto a' \qquad f_d\colon d \mapsto b'$$

Because every guarded set in $\mathfrak{M}$ is in the domain of one of the partial isomorphisms in I, every guarded set in $\mathfrak{M}'$ is in the range of at least one of the partial isomorphisms in I, and all the partial isomorphisms in I agree, this is indeed a guarded-2-bisimulation. Therefore by Proposition 8.7 $\mathfrak{M}$ and $\mathfrak{M}'$ are undistinguishable through guarded formulæ over the vocabulary $\{man^{(1)}, woman^{(1)}, love^{(2)}\}$. However, $\mathfrak{M} \not\models \exists x.woman^{(1)}(x) \wedge (\forall y.man^{(1)}(y) \supset love^{(2)}(y,x))$ but $\mathfrak{M}' \models \exists x.woman^{(1)}(x) \wedge (\forall y.man^{(1)}(y) \supset love^{(2)}(y,x))$. In particular, no $\mathcal{ALC}$ formula can express (8.35).

Another issue with the guarded fragment is that the axiom for transitivity of a binary relation R, which can be expressed by

$$\forall xyz.R^{(2)}(x,y) \wedge R^{(2)}(y,z) \supset R^{(2)}(x,z) \tag{8.36}$$

or by

$$\forall xy.R^{(2)}(x,y) \supset (\forall z.R^{(2)}(y,z) \supset R^{(2)}(x,z)) \tag{8.37}$$

is not guarded—nor in FO^2. In fact, the two-variable guarded fragment without equality and only a handful of transitive relations is already undecidable

(Ganzinger et al., 1999). This is an issue when considering epistemic or temporal modal logics, where transitivity is assumed; thankfully, decidability can be recovered when restricting transitive relations to occur solely in guards (e.g. Ganzinger et al., 1999; Michaliszyn, 2009).

Chapter 9

Tree Patterns

In this chapter, we consider formulæ called **patterns** from severely restricted fragments of first-order logic over trees. These provide concise means to define tree languages while avoiding the non-elementary complexity of full first-order logic over finite trees (e.g. Reinhardt, 2002). More precisely, we use patterns to define *finite* tree languages, which are then used as elementary trees in a grammar (Section 9.2) or as possible semantic readings in ambiguous sentences (Section 9.3).

9.1 *Background:* Existential First-Order Logic

When describing finite structures, existential sentences of first-order logic pop-up naturally: given a structure $\mathfrak{M} = \langle W, (R_i)_i \rangle$ over a finite domain $W = \{w_1, \ldots, w_n\}$ and a finite relational vocabulary $(R_i)_i$ (with no constants), the **canonical sentence** associated with $\mathfrak{M}$ is

$$\varphi_{\mathfrak{M}} \overset{\text{def}}{=} \exists x_1 \ldots x_n . \chi^+_{\mathfrak{M}}(x_1, \ldots, x_n) \tag{9.1}$$

where the formula $\chi^+_{\mathfrak{M}}$ is its **positive diagram** and consists of the conjunction of all the positive relational atomic formulæ true of $\mathfrak{M}$:

$$\chi^+_{\mathfrak{M}}(x_1, \ldots, x_n) \overset{\text{def}}{=} \bigwedge_i \bigwedge_{(w_{i_1}, \ldots, w_{i_k}) \in R_i} R^{(i_k)}(x_{i_1}, \ldots, x_{i_k}) \; . \tag{9.2}$$

Observe that $\mathfrak{M} \models \varphi_{\mathfrak{M}}$, and more precisely $\mathfrak{M} \models_\nu \chi^+_{\mathfrak{M}}(x_1, \ldots, x_n)$ using the valuation $\nu \colon x_i \mapsto w_i$. The canonical sentence $\varphi_{\mathfrak{M}}$ only uses existential quantification and conjunction.

EFO and its Fragments More generally, existential first-order logic (EFO) over a vocabulary σ is defined syntactically by

$$\begin{aligned} \alpha &::= x = y \mid R^{(k)}(x_1, \ldots, x_k) && \text{(atomic formulæ)} \\ \varphi &::= \alpha \mid \neg\alpha \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x . \varphi && \text{(existential formulæ)} \end{aligned}$$

where $x, y, x_1, \ldots, x_k$ range over $\mathcal{X}$ the set of variables, and R over the vocabulary σ.

- If both negated atoms $\neg\varphi$ and disjunctions $\varphi \vee \varphi$ are forbidden, we obtain **primitive positive** formulæ (E$^+$CFO), which are equivalent to **conjunctive queries** used in the database literature.

- If negated atoms $\neg\varphi$ are forbidden, we obtain **existential positive** formulæ ($\mathrm{E}^+\mathrm{FO}$), which are equivalent to **unions of conjunctive queries** used in the database literature.
- Finally, if disjunctions $\varphi\vee\varphi$ are forbidden, we obtain **existential conjunctive** formulæ (ECFO).

Normal Forms When putting an existential formula φ in disjunctive normal form, we see that it is equivalent to a finite disjunction of existential conjunctive formulæ ψ_i

$$\varphi \equiv \bigvee_i \psi_i \tag{9.3}$$

where in turn each existential conjunctive formula ψ can be put in prenex form

$$\psi \equiv \exists \mathbf{x}. \bigwedge_j \beta_j(\mathbf{x}_j) \tag{9.4}$$

where the β_j's are atoms or negated atoms and $\mathbf{x}_j$ is a subvector of $\mathbf{x}$. (If additionally φ was positive, then each ψ_i is primitive positive and the β_j's are atoms.) Observe finally that any atom of the form $x = y$ in some ψ can be eliminated by identifying the two variables x and y in ψ:

$$\exists x_1 x_2 \ldots x_n.\chi \wedge x_1 = x_2 \equiv \exists x_2 \ldots x_n.\chi\{x_1 \leftarrow x_2\} \tag{9.5}$$

so that the β_j's are necessarily relational or of the form $x \neq y$.

Small Models Given an existential conjunctive sentence $\psi = \exists x_1 \ldots x_n.\chi$, we can look at its models with at most n elements:

$$\mathrm{Mod}_{\leq n}(\psi) \stackrel{\text{def}}{=} \{\mathfrak{M} = \langle W, \sigma\rangle \mid |W| \leq n \wedge \mathfrak{M} \models \psi\} \,. \tag{9.6}$$

If ψ is positive, and positive equality atoms of the form $x = y$ have been eliminated as explained just before (thus only positive relational atoms appear in ψ), then ψ has a **canonical model** $\mathfrak{M}_\psi$ with domain $\{w_1, \ldots, w_n\}$ and a tuple $(w_{i_1}, \ldots, w_{i_k})$ in a k-ary relation R iff $R^{(k)}(x_{i_1}, \ldots, x_{i_k})$ is an atom in ψ. Clearly, $\mathfrak{M}_\psi \models \psi$, and furthermore the canonical sentence associated with $\mathfrak{M}_\psi$ is ψ itself.

(∗) **Exercise 9.1** (Canonical Model). Given an existential conjunctive sentence ψ without positive equality atoms (but possibly with some negated atom of the form $\neg R^{(k)}(x_{i_1}, \ldots, x_{i_k})$ or $x_i \neq x_j$), we distinguish its **positive part** ψ^+, which contains only the positive relational atoms of ψ. Show that, if ψ is satisfiable, then $\mathfrak{M}_{\psi^+} \models \psi$.

9.1.1 Characterisations over Finite Models

Fix some finite vocabulary $\sigma = (R_i)_i$. Given two structures $\mathfrak{M} = \langle W, (R_i)_i\rangle$ and $\mathfrak{M}' = \langle W', (R'_i)_i\rangle$, $\mathfrak{M}$ is an **induced substructure** of $\mathfrak{M}'$ if $W \subseteq W'$ and $R_i = R'_i \cap W^{k_i}$ for each k_i-ary relation. In that case, we also say that $\mathfrak{M}'$ is an **extension** of $\mathfrak{M}$ and write $\mathfrak{M} \subseteq_i \mathfrak{M}'$. A sentence φ in FO is **preserved under extensions** if $\mathfrak{M} \models \varphi$ and $\mathfrak{M} \subseteq_i \mathfrak{M}'$ together imply $\mathfrak{M}' \models \varphi$.

The Łoś-Tarski theorem fails over the class of all finite *structures (e.g. Ebbinghaus and Flum, 1999, Section 3.5). See Asterias* et al. *(2008) for classes of finite structures where it holds.*

The **Łoś-Tarski theorem** states that a first-order sentence is preserved under extensions over the class of all (finite and infinite) structures if and only if it is equivalent to an existential sentence. If we work on a particular class of structures, the theorem might fail, but one direction remains correct:

Proposition 9.1. *Let $\mathcal{C}$ be a class of structures. If φ is equivalent to an existential sentence over $\mathcal{C}$, then it is preserved under extensions over $\mathcal{C}$.*

Proof. Let $\mathfrak{M} = \langle W, (R_i)_i \rangle$ and $\mathfrak{M}' = \langle W', (R'_i)_i \rangle$ be two structures in $\mathcal{C}$ with $\mathfrak{M} \models \varphi$ and $\mathfrak{M} \subseteq_i \mathfrak{M}'$. Write φ as a finite disjunction of ECFO sentences as in (9.3): there exists a disjunct ψ such that $\mathfrak{M} \models \psi$. More precisely, ψ can be put in prenex normal form as $\psi \equiv \exists x_1 \dots x_n.\chi$ where χ is a conjunction of atoms and negated atoms, and $\mathfrak{M} \models_\nu \chi$ for some valuation $\nu\colon \{x_1, \dots, x_n\} \to W$. Consider the substructure $\mathfrak{M}_\nu$ (not necessarily in $\mathcal{C}$) induced by the subset $\nu(\{x_1, \dots, x_n\}) \subseteq W$ in $\mathfrak{M}$: then $\mathfrak{M}_\nu \models_\nu \chi$ and $\mathfrak{M}_\nu \subseteq_i \mathfrak{M} \subseteq_i \mathfrak{M}'$. We can easily check that $\mathfrak{M}' \models_\nu \chi$ and the result follows. □

In our applications, we will be especially interested in the (induced-)**minimal** models of existential sentences: given a class $\mathcal{C}$ of structures and a first-order sentence φ, $\mathfrak{M}$ in $\mathcal{C}$ is a minimal model of φ if $\mathfrak{M} \models \varphi$ and, if $\mathfrak{M}' \subsetneq_i \mathfrak{M}$, then $\mathfrak{M}' \not\models \varphi$.

Lemma 9.2. *Let $\mathcal{C}$ be a class of finite structures closed under induced substructures. If φ is equivalent to an existential sentence over $\mathcal{C}$, then φ has finitely many minimal models in $\mathcal{C}$.*

Proof. Using again the disjunctive normal form equivalent to φ, it suffices to show that there are finitely many minimal models for a disjunct ψ in ECFO. Let $\psi \equiv \exists x_1 \dots x_n.\chi$, $\mathfrak{M}$ be a minimal model of ψ and ν be a valuation such that $\mathfrak{M} \models_\nu \chi$. Then ν induces as in the proof of Proposition 9.1 a substructure $\mathfrak{M}_\nu \subseteq_i \mathfrak{M}$ with $\mathfrak{M}_\nu \models_\nu \chi$. Because $\mathcal{C}$ is closed under induced substructures, $\mathfrak{M}_\nu$ also belongs to $\mathcal{C}$, and because $\mathfrak{M}$ was assumed minimal, this in turn entails that $\mathfrak{M}_\nu$ and $\mathfrak{M}$ are isomorphic, and thus that $\mathfrak{M}$ has at most n elements.

In other words, if $\mathfrak{M}$ is a minimal model in $\mathcal{C}$, then

$$\mathfrak{M} \in \mathrm{Mod}_{\leq n}(\psi)\ . \tag{9.7}$$

(Note that this is not directly implied by Exercise 9.1, because $\mathfrak{M}_{\psi^+}$ might not be in $\mathcal{C}$.) We conclude by noting that $\mathrm{Mod}_{\leq n}(\psi)$ is finite for every n, and that n itself is bounded by the quantifier depth of φ. □

Exercise 9.2 (Diagrams). Let $\mathfrak{M} = \langle W, (R_i)_i \rangle$ be a finite structure with $W = \{w_1, \dots, w_n\}$. We define its **diagram** as the conjunction of the atomic and negated atomic formulæ it satisfies under the valuation $\nu\colon x_j \mapsto w_j$: **(∗)**

$$\chi_{\mathfrak{M}} \overset{\text{def}}{=} \bigwedge_{1 \leq j \leq k \leq n} x_j \neq x_k \wedge \bigwedge_i \Big(\bigwedge_{(w_{i_1}, \dots w_{i_k}) \in R_i} R_i^{(i_k)}(x_{i_1}, \dots, x_{i_k}) \wedge \bigwedge_{(w_{i_1}, \dots w_{i_k}) \notin R_i} \neg R_i^{(i_k)}(x_{i_1}, \dots, x_{i_k}) \Big) \tag{9.8}$$

Show that, for any structure $\mathfrak{M}'$, $\mathfrak{M}' \models \exists x_1 \dots x_n.\chi_{\mathfrak{M}}$ iff $\mathfrak{M} \subseteq_i \mathfrak{M}'$ (up to isomorphism).

Exercise 9.3 (Converse of Lemma 9.2). Let $\mathcal{C}$ be a class of finite structures and let φ be a first-order sentence preserved under extensions on $\mathcal{C}$. Show that, if φ has finitely many minimal models in $\mathcal{C}$, then it is equivalent to an existential sentence over $\mathcal{C}$. **(∗)**

Somewhat similar ideas can be worked out for existential positive sentences (instead of existential sentences) and homomorphisms between structures (instead of induced substructures), see Asterias et al. (2006); Rossman (2008); Dawar (2010).

9.1.2 Tree Models

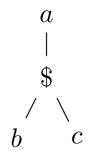
See Hidders (2004); Björklund et al. *(2011) for related results on tree patterns and XPath fragments.*

Unranked Trees Let us consider finite ordered unranked trees, with labels taken from some finite set A; note that for our applications we assume that each tree position is labelled by a single symbol from A. Because first-order logic cannot express transitive closures, we explicitly add the transitive reflexive closures $\downarrow^*$ of $\downarrow$ and $\rightarrow^*$ of $\rightarrow$ to our signature. In other words, we work over the relational signature $\langle \downarrow, \downarrow^*, \rightarrow, \rightarrow^*, (P_a)_{a \in A} \rangle$, and our class of models is restricted to trees, where the interpretation of $\downarrow^*$ (resp. $\rightarrow^*$) must coincide with the transitive reflexive closure of the interpretation of $\downarrow$ (resp. $\rightarrow$).

An issue with the class of trees is that it is not closed under induced substructures. For instance, the proof of Lemma 9.2 is incorrect for trees, e.g. the sentence

$$\exists xyz.P_a(x) \land P_b(y) \land P_c(z) \land x \downarrow^* y \land x \not\downarrow y \land x \downarrow^* z \land x \not\downarrow z \tag{9.9}$$

has minimal models of size 4 of the following form, for any label \$ in A:



Ranked Trees Another vocabulary of interest is $\langle (\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a \in A} \rangle$ where A is a finite ranked alphabet and k is the maximal arity in A. Again, the class of ranked trees is not closed under induced substructures.

Theorem 9.3 (Koller et al., 2001). *Satisfiability of ECFO*$((\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a \in A})$ *sentences is* NP*-complete.*

9.2 Meta-Grammars

In order to cope with the difficulty of hand-writing grammars with an adequate coverage of a natural language, it turns out to be quite convenient to see the grammar itself as the result of a compilation from a higher-level formalism. There exist many ways to define such a **meta-grammar**. Here we will focus on a simple formalism where the low-level grammar is the set of *minimal* models of an existential first-order formula on trees.

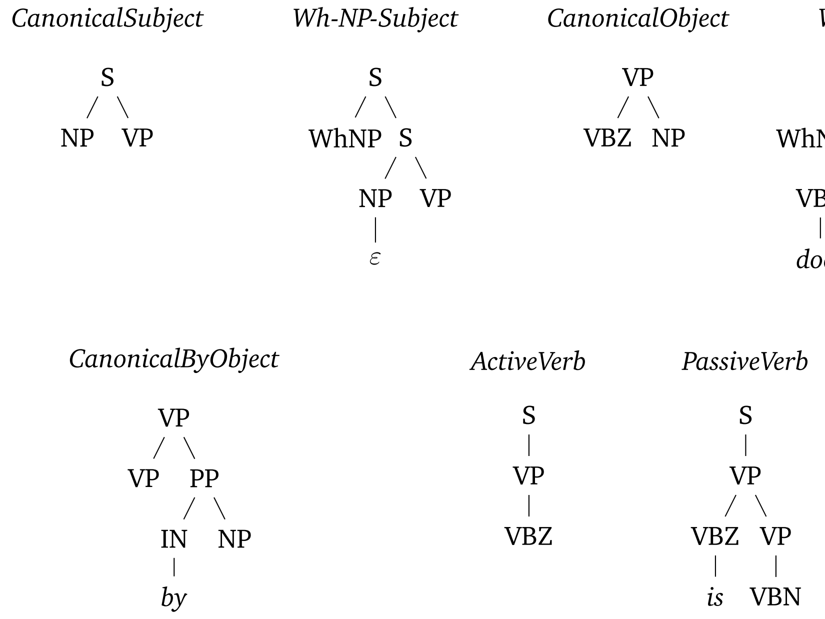
9.2.1 Diathesis Alternation

Section based on Crabbé et al. *(2013).*

One of the difficulties in competence grammars is to account for the many possible subcategorisation frames each lemma might allow. For instance, a transitive verb like *eat* allows for the sentences

John eats an apple.
Who eats an apple?
What does John eat?
An apple is eaten by John.

This not only leads to an explosion in the number of elementary tree structures in a context-free or tree-adjoining grammar, but also makes the semantic mapping (with adequate thematic roles) more cumbersome.



CanonicalSubject *Wh-NP-Subject* *CanonicalObject* *Wh-NP-Object*

CanonicalByObject *ActiveVerb* *PassiveVerb*

Figure 9.1: Basic tree fragments.

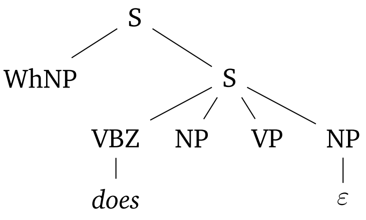


Figure 9.2: A minimal model of (9.10).

By allowing to factor some common patterns in elementary trees, we gain in succinctness. Moreover, by identifying linguistically-motivated atomic constructions, we obtain a more readable, easier to maintain description of the syntax. For instance, various elementary trees for transitive verbs can be described by the formulæ (number agreement could be handled through feature structures):

$$\begin{aligned} \mathit{TransitiveVerb} &\stackrel{\text{def}}{=} \mathit{ActiveTransitiveVerb} \vee \mathit{PassiveTransitiveVerb} \\ \mathit{ActiveTransitiveVerb} &\stackrel{\text{def}}{=} \mathit{Subject} \wedge \mathit{ActiveVerb} \wedge (\mathit{CanonicalObject} \vee \textit{Wh-NP-Object}) \\ \mathit{PassiveTransitiveVerb} &\stackrel{\text{def}}{=} \mathit{CanonicalSubject} \wedge \mathit{PassiveVerb} \wedge \mathit{CanonicalByObject} \\ \mathit{Subject} &\stackrel{\text{def}}{=} \mathit{CanonicalSubject} \vee \textit{Wh-NP-Subject} \end{aligned}$$

where each of the basic formulæ *ActiveVerb*, *PassiveVerb*, etc. is the canonical positive primitive formula of the corresponding tree in Figure 9.1. For instance, the conjunction

$$\mathit{CanonicalSubject} \wedge \mathit{ActiveVerb} \wedge \textit{Wh-NP-Object} \tag{9.10}$$

gives rise to the unique minimal model of Figure 9.2.

9.2.2 Complexity

Observe that we only used the ↓, →, and $\rightarrow^+$ axes in our examples in Section 9.2. One might hope that this fragment of $\mathrm{E}^+\mathrm{FO}$ would have a polynomial-time sat-

Section based on Björklund et al. *(2011).*

isfiability problem, but it turns out to be NP-hard already for primitive positive sentences with only $\to$ and $\to^+$:

Proposition 9.4. *Satisfiability of $E^+CFO(\to, \to^+, (P_a)_a)$ sentences is* NP*-complete.*

Proof. By **??**, satisfiability is in NP, thus we only need to prove hardness.

We reduce for this from the Shortest Common Supersequence Problem (SSSP), c.f. (Räihä and Ukkonen, 1981). An instance of SSSP is an integer k in unary and a set of strings $S = \{s_i = a_{i_1} \cdots a_{i_{\ell_i}}\}_{1 \leq i \leq p}$ over some finite alphabet Σ. The instance is positive if there exists a string s of length at most k, which is simultaneously a supersequence of every string in S, i.e. for every i, there exist strings $s'_0, \ldots, s'_{i_\ell+1}$ s.t. $s = s'_0 a_{i_1} s'_1 a_{i_2} \cdots a_{i_\ell} s'_{i_\ell+1}$. Importantly, if s is such a witness, then any supersequence of s over some alphabet that includes Σ and of length *exactly* k is also a witness.

Given an instance $\langle k, S\rangle$ of SSSP, we build an existential positive sentence φ, which is satisfiable iff the instance is positive. The idea is to find a sequence of children that spells out a witness s for the SSSP instance. In order to isolate this sequence, we add a fresh symbol # to Σ and make sure that we work between two nodes labelled with #:

$$\varphi \overset{\text{def}}{=} \exists z z'.P_\#(z) \wedge P_\#(z') \wedge \varphi_{=k}(z, z') \wedge \varphi_S(z, z')$$

On the one hand, our intention is for $\varphi_{=k}$ to make sure that the segment between z and z' is of length exactly k:

$$\varphi_{=k}(z, z') \overset{\text{def}}{=} \exists x_1 \ldots x_k.z \to x_1 \wedge x_k \to z' \wedge \big(\bigwedge_{1 \leq j < k} x_j \to x_{j+1}\big) .$$

On the other hand, $\varphi_S(z, z')$ should ensure that the segment between z and z' is indeed a supersequence of every $s_i = a_{i_1} \cdots a_{i_{\ell_i}}$:

$$\varphi_S(z, z') \overset{\text{def}}{=} \bigwedge_{1 \leq i \leq p} \exists y_1 \ldots y_{\ell_i}.z \to^+ y_1 \wedge y_{\ell_i} \to^+ z' \wedge \big(\bigwedge_{1 \leq j \leq \ell_i} P_{a_{i_j}}(y_j)\big) \\ \wedge \big(\bigwedge_{1 \leq r < \ell_i} y_r \to^+ y_{j+1}\big) . \qquad \square$$

9.3 Underspecified Semantics

9.3.1 Scope Ambiguities

An pervasive issue in semantic representations is related to **scope ambiguities**. Linguistic expressions are often semantically ambiguous (i.e. they have several possible readings that are mapped to different meaning representations) but fail to reflect this ambiguity syntactically (e.g. they have a single syntactic analysis). For instance, the sentence *Every man loves a woman* accepts two readings

$$\exists y.woman(y) \wedge \forall x.man(x) \supset \exists e.love(e) \wedge agent(e, x) \wedge patient(e, y) \tag{9.11}$$

$$\forall x.man(x) \supset \exists y.woman(y) \wedge \exists e.love(e) \wedge agent(e, x) \wedge patient(e, y) \tag{9.12}$$

depending on whether we are talking about one single woman or not; there is no clear reason why we should provide the sentence with different syntactic analyses.

Assuming we view meaning construction as a relation from one syntactic representation to several semantic ones, the number of readings can grow exponentially with the number of scope-bearing operators (quantifiers, modal operators, etc.), and simply enumerating the possible readings quickly turns impossible.

For instance, the sentence

> A politician can fool most voters on most issues most of the time, but no politician can fool every voter on every issue all of the time.
>
> (Poesio, 1994)

is reputed as having several thousand readings. Arguably, not all these readings are born equal: some might be implied by others (just like (9.11) implies (9.12)), and some downright impossible. However there can still remain a considerable number of incomparable readings. A naive approach to counting the number of possible readings is to consider all the permutations of quantifiers in a sentence: for a sentence with n quantifiers this will yield $n!$ different readings. Hobbs and Shieber (1987) for instance refine this approach and show how the sentence

> Every representative of a company saw most samples.

has actually 5 distinct readings instead of $3! = 6$: they argue that the reading where "for each representative there is a group of most samples which he saw, and furthermore, for each sample he saw, there was a company he was a representative of" is impossible.

A broadly adopted solution to the problems raised by scope ambiguities is to employ **underspecified representations** for semantics, which allow to represent several readings with a single representation. One might think such a trick, while computationally useful, defeats the very purpose of compositionality, but it does not if we view the underspecified representation as *the actual meaning* of the sentence…

There exist several such formalisms (e.g. Bos, 1996; Egg et al., 2001; Althaus et al., 2003; Copestake et al., 2005) but we will focus on one in particular: the **hole semantics** of Bos. The idea of hole semantics is to take as a semantic representation language (SRL) the logic we use for semantic representation (in our case FO) and build on top of it an underspecified representation language (URL), which describes the set of desired SRLs. As the latter are terms, the URL can be a formula s.t. the SRLs are its ranked tree models, i.e. we can reuse classical model-theoretic methods.

9.3.2 Hole Semantics

The syntax of **hole formulæ** is a restricted fragment of ECFO$((\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a\in A})$. We distinguish between two sorts of variables: **labels** l in $\mathcal{L}$ and **holes** h in $\mathcal{H}$ so that dominance relations $\downarrow^*$ can only go from holes to labels, and holes can only appear as unlabeled leaves; furthermore, immediate children relations and labelling predicates P_a are combined in a construct $l : a^{(r)}(x_1, \dots, x_r)$ that enforces the correct arity of a:

Our ECFO presentation of hole semantics follows Blackburn and Bos (2005, Chapter 3) rather than the original definition of Bos (1996).

$$\gamma ::= l : a^{(r)}(x_1, \dots, x_r) \mid h \downarrow^* l \mid \gamma \wedge \gamma \mid \exists x.\gamma \qquad \text{(hole formulæ)}$$

where l ranges over $\mathcal{L}$, $a^{(r)}$ over A_r, $x, x_1, \dots, x_r$ over $\mathcal{L} \uplus \mathcal{H}$, and h over $\mathcal{H}$. As with ECFO formulæ, hole formulæ γ can be put in prenex normal form

$$\gamma \equiv \exists l_1 \dots l_n h_1 \dots h_m. \bigwedge_p \gamma_p \;. \qquad (9.13)$$

Hole formulæ γ are *interpreted* in ECFO$((\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a\in A})$ by associating a formula $[\gamma]$

$$[\gamma] = \exists l_1 \dots l_n h_1 \dots h_m. \bigwedge_{1\leq i<j\leq n} l_i \neq l_j \wedge \bigwedge_p \gamma_p \tag{9.14}$$

where we interpret

$$l : a^{(r)}(x_1, \dots, x_r) \stackrel{\text{def}}{=} P_a(l) \wedge \bigwedge_{i=1}^{r} l \downarrow_{i-1} x_i \,. \tag{9.15}$$

A variable x in a hole formula is a **root** if there does not exist $x_0, \dots, x_r$ and $a^{(r)}$ s.t. $x_0 : a^{(r)}(x_1, \dots, x_r)$ is a subformula of γ where $x = x_j$ for some $1 \leq j \leq r$. A hole formula is **normal** if

1. in every $h \downarrow^* l$ subformula, l is a root of γ,
2. every hole appears exactly once as a child of a $l : a^{(r)}(x_1, \dots, x_r)$ subformula, and thus cannot be a root,
3. every label should appear at most once as a parent and at most once as a child in a $l : a^{(r)}(x_1, \dots, x_r)$ subformula. This excludes for instance $l' : f^{(2)}(l, l)$, $l : f^{(2)}(l_1, l_2) \wedge l : f^{(2)}(l'_1, l'_2)$, or $l_1 : g^{(1)}(l) \wedge l_2 : g^{(1)}(l)$.

Normal hole formulæ with this interpretation into ECFO give rise to **normal dominance constraints**, which are known to be efficiently testable for satisfiability:

Theorem 9.5 (Althaus et al., 2003). *Satisfiability of normal hole formulæ is in* P.

Constructive Satisfiability

The issue with our interpretation of hole formulæ into ECFO is that not every model $\mathfrak{M}$ over A is suitable as a SRL formula. For instance, there could be extra points in the model not constrained by γ, or conversely several labels could be mapped to a single node. An alternative notion of model is needed in practice.

Consider a hole formula in prenex conjunctive normal form as in (9.13). Then a **plugging** P is an injective function from holes $\{h_1, \dots, h_m\}$ to labels $\{l_1, \dots, l_n\}$. A model $\mathfrak{M} = \langle \mathrm{dom}(t), (\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a\in A}\rangle$ of γ is a **plugged model** for a plugging P if its domain is in bijection with the set of labels (we write $\mathrm{dom}(t) = \{\hat{l}_1, \dots, \hat{l}_n\}$) and $\mathfrak{M} \models_\nu \gamma$ where the valuation ν is defined by

$$\nu(x) \stackrel{\text{def}}{=} \begin{cases} \hat{x} & \text{if } x \in \mathcal{L} \\ \widehat{P(x)} & \text{if } x \in \mathcal{H}\,. \end{cases} \tag{9.16}$$

The structure $\mathfrak{M}$ is a **constructive model** for γ if there exists a plugging P s.t. it is a plugged model for P.

Example 9.6. Let us extend the syntax of hole formulæ by allowing larger tree segments:

$$\gamma ::= l : a^{(r)}(\theta_1, \dots, \theta_r) \mid h \downarrow^* l \mid \gamma \wedge \gamma \mid \exists x.\gamma \qquad \text{(hole formulæ)}$$
$$\theta ::= a^{(r)}(\theta_1, \dots, \theta_r) \mid h \qquad \text{(tree formulæ)}$$

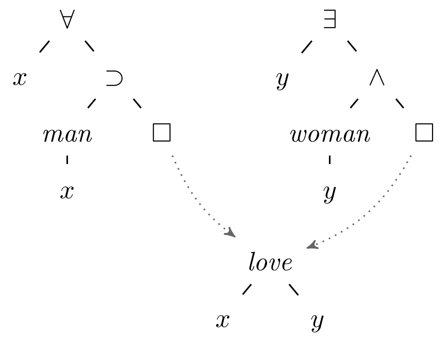


Figure 9.3: Underspecified formula for (9.11) and (9.12). Dominance relations are indicated through dotted arrows and holes by boxes.

and translating back into hole formulæ by defining

$$x_\theta \overset{\text{def}}{=} \begin{cases} h & \text{if } \theta = h \\ l_\theta \in \mathcal{L} & \text{a fresh label for each } \theta \text{ otherwise} \end{cases}$$

$$l : a^{(r)}(\theta_1, \ldots, \theta_r) \overset{\text{def}}{=} l : a^{(r)}(x_{\theta_1}, \ldots, x_{\theta_r})$$

$$a^{(r)}(\theta_1, \ldots, \theta_r) \overset{\text{def}}{=} \exists l_\theta . l_\theta : a^{(r)}(x_{\theta_1}, \ldots, x_{\theta_r}) \; .$$

A hole semantic formula that models the two readings (9.11) and (9.12) is the following (see also Figure 9.3):

$$\exists l_1 l_2 l_3 h_1 h_2 . l_1 : \forall^{(2)}(x^{(0)}, \mathit{man}^{(1)}(x^{(0)}) \supset^{(2)} h_1) \wedge l_2 : \exists^{(2)}(y^{(0)}, \mathit{woman}^{(1)}(y^{(0)}) \wedge^{(2)} h_2)$$
$$\wedge l_3 : \mathit{love}^{(2)}(x^{(0)}, y^{(0)}) \wedge h_1 \downarrow^* l_3 \wedge h_2 \downarrow^* l_3 \; .$$

Constructive satisfiability puts a higher toll on computations than basic satisfiability:

Polynomial-time processing can be recovered if we further restrict hole formulæ; see Koller et al. *(2003).*

Theorem 9.7. *Constructive satisfiability of normal hole formulæ is* NP*-complete.*

Proof. For the NP upper bound, deciding whether a formula γ has a constructive model can be checked by

1. guessing both a plugging P and the corresponding model
$$\mathfrak{M} = \langle \{\hat{l}_1, \ldots, \hat{l}_n\}, (\downarrow_i)_{i<k}, (P_a)_{a \in A} \rangle \; ; \qquad (9.17)$$
this model is of polynomial size in $|\gamma|$,

2. computing the dominance relation $(\bigcup_{i<k} \downarrow_i)^\star$ over $\mathfrak{M}$ (this is in P) to obtain a model
$$\mathfrak{M}' = \langle \{\hat{l}_1, \ldots, \hat{l}_n\}, (\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a \in A} \rangle \qquad (9.18)$$
still of polynomial size, and

3. verifying that $\mathfrak{M}'$ is a model of the existentially conjunctive formula $[\gamma]$ for the assignment ν defined in (9.16) (this is in P).

For the NP lower bound, we exhibit a reduction from the 3-Partition Problem. An instance of this problem is given by a finite multiset $A = \{a_1, \ldots, a_{3m}\}$ of integers and a bound B, all in $\mathbb{N}$ and encoded in unary, such that $\frac{B}{4} < a_i < \frac{B}{2}$ for all i and $\sum_{i=1}^{3m} a_i = mB$. The instance is positive if there exists a partition $A_1 \uplus A_2 \uplus \cdots \uplus A_m$

of A s.t. for all j, $|A_j| = 3$ and $\sum_{a\in A_j} a = B$. We can assume $B > 0$ (or $a_i = 0$ for all i).

This hardness proof from the 3-partition problem is taken from (Althaus et al.*, 2003, Theorem 10.1).*

We construct from an instance $\langle A, B\rangle$ a hole formula over the ranked alphabet $\{\$^{(1)}, f_i^{(a_i+1)}, g^{(m)}, b^{(0)} \mid 1 \leq i \leq 3m\}$:

$$\begin{aligned}&\exists l_\$ l_{f_1} \ldots l_{f_{3m}} l_g l_b^{1,1} \ldots l_b^{1,B+1} l_b^{2,1} \ldots l_b^{m,B+1} h_\$ h_g^1 \ldots h_g^m h_{f_1}^1 \ldots h_{f_1}^{a_1+1} h_{f_2}^1 \ldots h_{f_{3m}}^{a_{3m}+1} .\\
&\quad \$^{(1)}(h_\$) \wedge \bigwedge_{1\leq i\leq 3m} l_{f_i} : f_i^{(a_i+1)}(h_{f_1}^1, \ldots, h_{f_1}^{a_i+1})\\
&\quad \wedge l_g : g^{(m)}(h_g^1, \ldots, h_g^m) \wedge \bigwedge_{1\leq j\leq m} \bigwedge_{1\leq k\leq B+1} l_b^{j,k} : b^{(0)}\\
&\quad \wedge \bigwedge_{1\leq i\leq 3m} h_\$ \downarrow^* l_{f_i} \wedge h_\$ \downarrow^* l_g \wedge \bigwedge_{1\leq j\leq m} \bigwedge_{1\leq k\leq B+1} h_g^j \downarrow^* l_b^{j,k} .\end{aligned}$$

Assume first that there exists a partition $A_1 \uplus \cdots \uplus A_m$ of A: we plug $h_\$$ with l_g, and for each class $A_j = \{a_x, a_y, a_z\}$ with $a_x + a_y + a_z = B$, we plug h_g^j with l_{f_x}, $h_{f_x}^{a_x+1}$ with l_{f_y}, and $h_{f_y}^{a_y+1}$ with l_{f_z}, and the remaining $B + 1$ holes $h_{f_x}^1, \ldots, h_{f_x}^{a_x}, h_{f_y}^1, \ldots, h_{f_y}^{a_y}, h_{f_z}^1, \ldots, h_{f_z}^{a_z+1}$ by the labels $l_b^{j,k}$ for $1 \leq k \leq B + 1$.

Conversely, assume there is a plugging P from holes to labels and let $\mathfrak{M}$ be the corresponding plugged model using valuation ν. For every $1 \leq j \leq m$, consider the set A_j of integers a_i such that f_i-rooted fragments are plugged below h_g^j, i.e. $A_j \overset{\text{def}}{=} \{a_i \mid \mathfrak{M} \models_\nu h_g^j \downarrow^* l_{f_i}\}$. Note that $A_1 \uplus \cdots \uplus A_m$ forms a partition of A. Because a plugging is injective from holes to labels, each f_i-rooted fragment requires $a_i + 1$ labels, h_g^j requires one, and $|A_j| + B + 1$ are available using the f_i- and b-rooted fragments, we get that $1 + |A_j| + \sum_{a\in A_j} a_i \leq |A_j| + B + 1$, hence $\sum_{a\in A_j} \leq B$ for every $1 \leq j \leq m$. Because $\sum_{a\in A} a = mB$, there is no choice and $\sum_{a\in A_j} a = B$.

Furthermore, $|A_j| \geq 3$:

- $|A_j| \neq 0$ since $B > 0$, and $B + 1$ fragments rooted by b must be plugged somewhere below the single hole h_g;
- $|A_j| \neq 1$ since a single f_i-rooted fragment provides $a_i + 1 < \frac{B}{2} < B + 1$ holes,
- $|A_j| \neq 2$ since a pair $\{a_x, a_y\}$ provides $a_x + a_y + 2 - 1 < B + 1$ holes.

Thus every A_j is of cardinality at least 3, and because $3m$ f_i-rooted fragments are available in total, this means that $|A_j| = 3$ for all j. □

(∗∗∗) **Exercise 9.4** (Tree Automata for Hole Formulæ)**.** The set of constructive models of a constraint is clearly a regular tree language. Provide a construction for a regular tree automaton $\mathcal{A}_\gamma$ that recognizes exactly the constructive models of a normal hole formula γ.

Hint: I would use $2^{\{l_1,\ldots,l_n\}} \times \{l_1, \ldots, l_n\} \times 2^{\{h_1,\ldots,h_m\}}$ as state set, although there certainly are better ways; see for instance Koller et al. *(2008).*

The size of the automaton constructed in Exercise 9.4 is exponential in the size of the formula. This is unavoidable, as there exist normal formulæ γ_n of size $O(n)$ s.t. any automaton recognizing the set of plugged models of γ_n requires at least 2^n states: let

$$A_n \overset{\text{def}}{=} \{a^{(0)}, g_1^{(1)}, \ldots, g_n^{(1)}\} \tag{9.19}$$

$$\gamma_n \overset{\text{def}}{=} \exists l l_1 \ldots l_n h_1 \ldots h_n . l : a^{(0)} \wedge \bigwedge_{i=1}^{n} l_i : g_i^{(1)}(h_i) \wedge h_i \downarrow^* l \,. \tag{9.20}$$

The normal formula γ_n has $n!$ different models, corresponding to the possible orderings of its n components $g_i(\Box)$: its set of plugged models is

$$L_n = \{g_{\pi(1)}(\Box) \cdot g_{\pi(2)}(\Box) \cdots g_{\pi(n)}(a) \mid \pi \text{ a permutation of } \{1,\dots,n\}\}\ . \tag{9.21}$$

Lemma 9.8. *Any finite tree automaton for L_n requires at least 2^n states.*

Proof. Define for every subset $K = \{i_1,\dots,i_{|K|}\}$ of $\{1,\dots,n\}$ (where $i_j < i_{j+1}$) the context

$$C_K \overset{\text{def}}{=} g_{i_1}(\Box) \cdots g_{i_{|K|}}(\Box) \tag{9.22}$$

and let $\bar{K} = \{1,\dots,n\}\backslash K$. Then the tree

$$t_K \overset{\text{def}}{=} C_{\bar{K}} \cdot C_K \cdot a \tag{9.23}$$

is in L_n.

Let Q_K be the set of states q of an automaton $\mathcal{A}_n$ for L_n s.t.

$$C_{\bar{K}} \cdot C_K \cdot a \Rightarrow^\star C_{\bar{K}} \cdot q \Rightarrow^\star q_f \tag{9.24}$$

for some final state q_f. Since t_K is in L_n, $Q_K \neq \emptyset$. Suppose there exist $K \neq K'$ s.t. $Q_K \cap Q_{K'} \neq \emptyset$, i.e. there exists i in $K\backslash K'$ and $q \in Q_K \cap Q_{K'}$. Then i belongs to $\bar{K}'$ and

$$C_{\bar{K}'} \cdot C_K \cdot a \Rightarrow^\star C_{\bar{K}'} \cdot q \Rightarrow^\star q_f \tag{9.25}$$

recognizes a tree not in L_n (the pattern $g_i(\Box)$ appears twice). Hence the non-empty sets Q_K must be disjoint for different sets K, thus $\mathcal{A}_n$ has at least 2^n states. □

Note that the tree automaton $\langle 2^{\{1,\dots,n\}}, A, \delta, \{\emptyset\}\rangle$ with $\delta = \{(q\backslash\{i\}, g_i, q) \mid i \in q\} \cup \{(\{1,\dots,n\}, b)\}$ recognizes L_n, so this bound is optimal.

Lemma 9.8 shows that there might be exponential succinctness gains from the use of hole formulæ rather than tree automata for the description of semantic representations. One might object that the classes of tree languages obtained at the output of the linear higher-order tree functions of Section 10.1.4 are *context-free* tree languages and not necessarily regular ones, with potential exponential gains in succinctness. However, note that L_n is basically a string language, and the exponential lower bounds on the size of any context-free string grammar for permutation languages (see e.g. Filmus, 2011) also apply to CFTGs for L_n.

Chapter 10

Higher-Order Semantics

In this last chapter, we consider the use of higher-order functions in natual language semantics. We first motivate the need for such functions in Section 10.1 in order to define the interface between syntax and semantics. We then observe that, more generally, 'increasing the order' allows for elegant solutions to some difficulties like intensionality phenomena and many-world semantics.

10.1 Compositional Semantics

We have presented several possible first-order analyses for simple sentences in the previous chapters, but we have not touched yet the subject of *how* to obtain such semantic representations from syntactic analyses. A key concept in this regard is that of **compositionality**:

See Janssen (1997) and the compositionality article of the Stanford Encyclopedia of Philosophy *for extensive discussions of compositionality.*

> The meaning of a compound expression is a function of the meanings of its parts and of the syntactic rule by which they are combined.
>
> (Partee et al., 1990, Chapter 13)

Let us illustrate this principle on Example 8.1: by associating a semantic representation to each meaningful word in the sentence, i.e. if we define $[\![\textit{John}]\!]$, $[\![\textit{eats}]\!]$ and so on, then the semantics of each intermediate structure like *a red apple* or *John eats a red apple* can be systematically computed as a function of its parts, based on the syntactic structures. Note that these structures play a crucial role, as otherwise *John loves Mary* and *Mary loves John* would not be distinguishable as naive 'functions of their parts.'

You are probably familiar with this principle from programming language semantics. Typical arguments in favour of this principle for natural language hinge on **productivity** and **systematicity** of semantic construction: we are able to understand new linguistic expressions, and to understand similar expressions built from the same blocks and syntactic processes.

Leaving these questions aside and adopting a modelling viewpoint, compositionality is a rather strenuous requirement: for instance, assuming $[\![\textit{John}]\!] = \mathit{John}^{(0)}$ and $[\![\textit{a red apple}]\!] = \exists x.\mathit{apple}^{(1)}(x) \wedge \mathit{red}^{(1)}(x)$, it is not so clear how one should combine everything and obtain (8.1) or more involved representations like (8.24). Moreover any solution will be dependent on the specific syntactic analysis.

10.1.1 *Background:* Simply Typed Lambda Calculus

See e.g. Hindley (1997).

One of the best-studied ways to implement compositional semantics for natural languages is to use lambda expressions as semantic values associated with each component (Montague, 1970, 1973). As Church's simple theory of types provides an elegant setting for model-theoretic higher-order semantics (see Section 10.3), we favour a presentation that uses the **simply typed λ-calculus** over the untyped one.

Lambda Terms Given an infinite countable set $\mathcal{X}$ of *variables*, and C a countable set of *constants*, the set $\Lambda(C)$ of **λ-terms** is defined by

$$L ::= c \mid x \mid LL \mid \lambda x.L$$

where c is a constant in C and x a variable in $\mathcal{X}$.

The λ operator is a *binding* with the usual associated notion of free variables. We draw a distinction between **closed** terms, which have no free variables, and **ground** terms, which have no variables at all.

A λ-term L is a **λI-term** if in every subterm $\lambda x.M$, $x \in \mathrm{FV}(M)$. If furthermore x appears free in M exactly once, and each free variable y of L has at most one free occurrence in L, then L is a **linear λ-term**; we let $\Lambda_\ell(C)$ denote the set of linear λ-terms over C. We write by convention $\lambda xy.L$ for $\lambda x.\lambda y.L$ and LMN for $(LM)N$ (i.e. we treat application as left associative).

We assume the usual definitions for α, β, and η reductions:

$$\begin{aligned} \lambda x.L &\to_\alpha \lambda y.(L\{x \leftarrow y\}) \\ (\lambda x.L)M &\to_\beta L\{x \leftarrow M\} \\ \lambda x.(Lx) &\to_\eta L \end{aligned}$$

(where substitutions have to avoid name clashes and $x \notin \mathrm{FV}(L)$ for η-reductions), and recall that $\beta\eta$-reductions are **Church-Rosser**: if $L \Rightarrow^\star_{\beta\eta} M$ and $L \Rightarrow^\star_{\beta\eta} N$, then there exists L' s.t. $M \Rightarrow^\star_{\beta\eta} L'$ and $N \Rightarrow^\star_{\beta\eta} L'$, which implies that $\beta\eta$ reductions define unique **normal forms**, noted $\Downarrow_{\beta\eta}L$.

Types Assume we are provided with some non-empty countable set of *atomic* types A; then **types** in $\mathcal{T}_A$ are terms defined inductively by

$$\tau ::= a \mid \tau \to \tau$$

where a ranges over A. By convention we consider $\to$ to be right-associative, i.e. we write $\rho \to \sigma \to \tau$ for $\rho \to (\sigma \to \tau)$. The **order** of a type τ is defined inductively as

$$\mathrm{ord}(a) = 1 \qquad\qquad \mathrm{ord}(\sigma \to \tau) = \max(\mathrm{ord}(\sigma) + 1, \mathrm{ord}(\tau)) \;.$$

A **higher-order signature** is a triple $\Sigma = \langle A, C, \tau \rangle$ where A is a set of atomic types, C a countable set of *constants* and $\tau : C \to \mathcal{T}_A$ a *typing* of the constants. Given a higher-order signature, each λI-term of $\Lambda(C)$ can be assigned a type in $\mathcal{T}_A$ by the deduction rules

$$\frac{}{\vdash_\Sigma c : \tau(c)}\;(\mathsf{Cons}) \qquad \frac{}{x : \tau \vdash_\Sigma x : \tau}\;(\mathsf{Var}) \qquad \frac{\Gamma, x : \sigma \vdash_\Sigma L : \tau}{\Gamma \vdash_\Sigma \lambda x.L : \sigma \to \tau}\;(\to\mathsf{I})$$

$$\frac{\Gamma \vdash_\Sigma L : \sigma \to \tau \quad \Delta \vdash_\Sigma M : \sigma \quad \Gamma, \Delta \text{ compatible}}{\Gamma, \Delta \vdash_\Sigma LM : \tau}\;(\to\mathsf{E})$$

where the **type contexts** Γ, Δ are type assignments from free variables to $\mathcal{T}_A$; in ($\to$E) the two assignments have to be *compatible*, i.e. assign the same types to common variables. For *linear* lambda terms, this compatibility requirement is useless as $\mathrm{FV}(L) \cap \mathrm{FV}(M) = \emptyset$. We can extend the typing system to any λ-term instead of λI-terms if we additionally allow ($\to$I) to work on the premise $\Gamma \vdash_\Sigma L : \tau$ where x is not among $\mathrm{FV}(L)$ nor in the domain of Γ.

Example 10.1 (**B** combinator)**.** Define $\mathbf{B} \overset{\text{def}}{=} \lambda xyz.x(yz)$. It can be typed by:

$$\dfrac{\dfrac{\dfrac{\dfrac{x : a \to b \vdash_\Sigma x : a \to b \qquad \dfrac{y : c \to a \vdash_\Sigma y : c \to a \qquad z : c \vdash_\Sigma z : c}{y : c \to a, z : c \vdash_\Sigma yz : a}}{x : a \to b, y : c \to a, z : c \vdash_\Sigma x(yz) : b}}{x : a \to b, y : c \to a \vdash_\Sigma \lambda z.x(yz) : c \to b}}{x : a \to b \vdash_\Sigma \lambda yz.x(yz) : (c \to a) \to c \to b}}{\vdash_\Sigma \lambda xyz.x(yz) : (a \to b) \to (c \to a) \to c \to b}$$

Properties Let us end this quick survey with a few important properties of the simply typed λ calculus: The first two show that types are preserved by reductions:

See e.g. (Hindley, 1997, Chapter 2).

Proposition 10.2 (Subject Reduction)**.** *If* $\Gamma \vdash_\Sigma L : \tau$ *and* $L \Rightarrow^\star_{\beta\eta} M$ *then* $\Gamma \vdash_\Sigma M : \tau$.

The converse holds for *linear* terms (and more generally for reductions that do not exercise non linear variables):

Proposition 10.3 (Subject Expansion)**.** *If* τ *is a linear* λ*-term,* $\Gamma \vdash_\Sigma L : \tau$*, and* $M \Rightarrow^\star_\beta L$*, then* $\Gamma \vdash_\Sigma M : \tau$.

Exercise 10.1. Prove Proposition 10.2 and Proposition 10.3. (∗)

The second main result about typed λ-terms is that reduction is **strongly normalising**: every sequence of rewrites eventually terminates to a term in normal form:

The length of $\beta\eta$ reductions can be non elementary in the size of the starting term (see Statman, 1979a; Schwichtenberg, 1991).

Theorem 10.4 (Strong Normalisation)**.** *If* L *is a typable* λ*-term, then every* $\beta\eta$*-reduction starting at* L *is finite.*

Remember that not every λ-term is typable; the typical example of a non-typable term being $\lambda x.xx$. However, every linear λ-term *is* typable. A related question is the **type inhabitation** problem: given a simple type τ, does there exist a closed λ-term L with type τ? This is usually formulated over an empty set of constants $C = \emptyset$. By the Curry-Howard isomorphism (see e.g. Hindley, 1997, Chapter 6), the type inhabitation problem is the same as provability in intuitionistic propositional logic:

The type inhabitation problem becomes 2EXPTIME*-complete for* λI*-terms (Schmitz, 2014).*

Theorem 10.5 (Statman, 1979b)**.** *Simple type inhabitation is* PSPACE*-complete.*

10.1.2 Ground Terms over Second-Order Signatures

Because we are typically interested in tree structures, it is worth investigating how they can be represented in the simply-typed λ-calculus. To this end, we restrict ourselves to second-order signatures $\Sigma = \langle A, C, \tau \rangle$, i.e. signatures such that the type of any constant c is of form

$$\tau(c) = a_1 \to \cdots \to a_n \to a_0$$

for atomic a_i's in A.

(∗∗) **Exercise 10.2** (Normalised Typing System)**.** Consider the normalised typing system with a single rule

$$\frac{\tau(c) = a_1 \to \cdots \to a_n \to a_0 \quad \vdash'_\Sigma t_1 : a_1 \ \ldots\ \vdash'_\Sigma t_n : a_n}{\vdash'_\Sigma c\, t_1 \cdots t_n : a_0} \text{ (App)}$$

We want to show that, for all ground terms t and atomic types a, $\vdash_\Sigma t : a$ if and only if $\vdash'_\Sigma t : a$.

1. Show that, if $\tau(c) = a_1 \to \cdots \to a_n \to a_0$, $0 \leq i \leq n$, and $\vdash_\Sigma t_j : a_j$ for all $0 < j \leq i$, then $\vdash_\Sigma c\, t_1 \cdots t_i : a_{i+1} \to \cdots \to a_n \to a_0$. Deduce that $\vdash'_\Sigma t : a$ implies $\vdash_\Sigma t : a$ if t is ground and a atomic.

2. Show that, if $\vdash_\Sigma t : \alpha$ for a ground term t and type α, then $t = c\, t_1 \cdots t_i$ for some constant c with $\tau(c) = a_1 \to \cdots \to a_n \to a_0$, some $0 \leq i \leq n$, and some ground terms $t_1, \ldots, t_i$ such that $\alpha = a_{i+1} \to \cdots \to a_n \to a_0$ and $\vdash_\Sigma t_j : a_j$ for $0 < j \leq i$ for some atomic types a_j's.

3. Deduce that $\vdash_\Sigma t : a$ implies $\vdash'_\Sigma t : a$ whenever t is a ground term and a an atomic type.

For a second-order constant c with type $\tau(c) = a_1 \to \cdots \to a_n \to a_0$, we call n its *arity* (and thus can see C as a ranked alphabet) and associate to the *ground* lambda term $t = c\, t_1 \cdots t_n$ with atomic type a_0 the unique tree $\bar{t} = c^{(n)}(\bar{t}_1, \ldots, \bar{t}_n)$. Given a second-order signature Σ and a distinguished atomic type s, we define the **ground tree language**

$$\mathscr{G}(\Sigma, s) \overset{\text{def}}{=} \{\bar{t} \in T(C) \mid \vdash_\Sigma t : s \text{ where } t \text{ is ground}\} .$$

Example 10.6. Consider the second-order signature Σ_0 with atomic types $A_0 = \{np, s, c\}$, constants $C_0 = \{\mathit{Alice}, \mathit{believe}, \mathit{left}, \mathit{someone}, \mathit{that}\}$, and typing

$$\begin{aligned} \tau_0(\mathit{Alice}) &= np & \tau_0(\mathit{believe}) &= c \to np \to s \\ \tau_0(\mathit{left}) &= np \to s & \tau_0(\mathit{someone}) &= np \\ \tau_0(\mathit{that}) &= s \to c \end{aligned}$$

The corresponding ranked alphabet is $\mathcal{F}_0 = \{\mathit{Alice}^{(0)}, \mathit{believe}^{(2)}, \mathit{left}^{(1)}, \mathit{someone}^{(0)}, \mathit{that}^{(1)}\}$. Then the set of trees in $\mathscr{G}(\Sigma_0, s)$ is recognised by a tree automaton $\mathcal{A} = \langle Q, \mathcal{F}_0, \delta, I\rangle$ with $Q = A_0$, $I = \{s\}$, and rules

$$\begin{aligned} \delta = \{&(np, \mathit{Alice}^{(0)}), \\ &(s, \mathit{believe}^{(2)}, c, np) \\ &(s, \mathit{left}^{(1)}, np) \\ &(np, \mathit{someone}^{(0)}) \\ &(c, \mathit{that}^{(1)}, s)\} . \end{aligned}$$

(∗∗) **Exercise 10.3** (Local Tree Automata)**.** Let $\mathcal{F}$ be a ranked alphabet. A deterministic top-down tree automaton $\mathcal{A} = \langle Q, \mathcal{F}, \delta, \{q_0\}\rangle$ is **local** if there exists a function $\ell{:}\mathcal{F} \to Q$ such that the rules in δ are all of the form $(\ell(f^{(n)}), f^{(n)}, q_1, \ldots, q_n)$.

1. Show that, if L is recognized by a local deterministic top-down tree automaton, then there is a second order signature Σ and a distinguished atomic type s such that $L = \mathscr{G}(\Sigma, s)$.

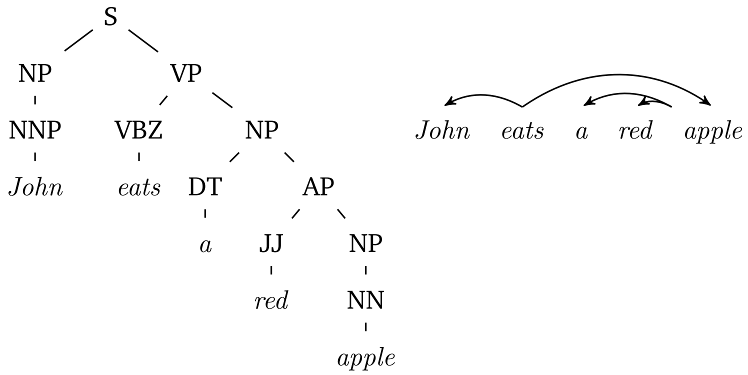


Figure 10.1: Constituent and dependency analyses for *John eats a red apple*.

2. Show that, conversely, given a second-order signature Σ and a distinguished atomic type s, there exists a local top-down deterministic tree automaton $\mathcal{A}$ such that $L(\mathcal{A}) = \mathscr{G}(\Sigma, s)$.

By the previous exercise, not every regular tree language can be expressed as the ground tree language of a second-order signature, e.g. the language $L = \{f(g(a), g(b))\}$ is not local.

10.1.3 Higher-Order Homomorphisms

One of the main legacies of Richard Montague's work is the idea that semantic representations can be obtained through the application of a homomorphism on the syntactic structure. However tree homomorphisms are clearly too weak for the kind of tree transductions we want to define; following Montague we use instead **higher-order homomorphisms**. The idea of these homomorphisms is to translate a syntactic tree representation (e.g. a derivation tree or a dependency tree), seen as a typed λ-term over the input signature, into a λ-term over the output signature and then to $\beta\eta$-reduce it to a λ-term in normal form.

This idea is now pretty common, and lies at the heart of (second-order) ***abstract categorial grammars*** *(ACG de Groote, 2001); see also the* ***context-free λ-term grammar*** *(CFLG) formulation of Kanazawa (2007) or the simple presentation of Blackburn and Bos (2005, Chapter 2).*

Definition 10.7 (Higher-Order Homomorphism)**.** A **higher-order homomorphism** from a set of constants C to a set of constants C' is generated by a function $[\![.]\!]$ mapping constants in C to closed λ-terms in $\Lambda(C')$. We lift $[\![.]\!]$ to a homomorphism from $\Lambda(C)$ to $\Lambda(C')$ by $[\![x]\!] = x$, $[\![LM]\!] = [\![L]\!][\![M]\!]$, and $[\![\lambda x.L]\!] = \lambda x.[\![L]\!]$.

Example 10.8. Continuing with Example 8.1, Figure 10.1 presents two syntactic analyses (the dependency one could for instance be obtained from the constituent one through head percolation analysis or as the derivation tree of a TAG). For the constituent analysis of Figure 10.1, we have

$$C = \{\textit{John}^{(0)}, \textit{apple}^{(0)}, \ldots, \mathrm{AP}^{(2)}, \mathrm{NP}^{(2)}, \mathrm{JJ}^{(1)}, \ldots, \mathrm{S}^{(2)}\}$$

and

$$C' = \{\textit{John}^{(0)}, \wedge^{(2)}, \exists^{(2)}, \ldots\}\ .$$

We assign the semantics

$$
\begin{aligned}
[\![\mathit{John}^{(0)}]\!] &= \lambda x.x\;John^{(0)}\\
[\![\mathit{apple}^{(0)}]\!] &= \lambda x.apple^{(1)}\,x\\
[\![\mathit{red}^{(0)}]\!] &= \lambda x.red^{(1)}\,x\\
[\![\mathrm{AP}^{(2)}]\!] &= \lambda x_1 x_2 x.(x_1\,x) \wedge (x_2\,x)\\
[\![\mathit{a}^{(0)}]\!] &= \lambda xy.\exists u.(x\,u) \wedge (y\,u)\\
[\![\mathrm{NP}^{(2)}]\!] &= \lambda x_1 x_2 x.x_1\,x_2\,x\\
[\![\mathit{eats}^{(0)}]\!] &= \lambda xy.\exists e.(eat^{(1)}\,e) \wedge x(\lambda a.agent^{(2)}\,e\,a)\\
&\quad \wedge y(\lambda p.patient^{(2)}\,e\,p)
\end{aligned}
$$

$$
\begin{aligned}
[\![\mathrm{VP}^{(2)}]\!] &= \lambda x_1 x_2 x.x_1\,x\,x_2\\
[\![\mathrm{S}^{(2)}]\!] &= \lambda x_1 x_2.x_2\,x_1
\end{aligned}
$$

(ignoring tree nodes with a single child, for which we set e.g. $[\![\mathrm{NN}^{(1)}]\!] = \lambda x_1.x_1$). The first-order variables u and e could be considered as constants of arity 0 in C', but this causes some naming issues; an alternative would be treat $\exists x.\varphi$ as $\exists\lambda x.\varphi$. This definition results successively in

$$
\begin{aligned}
[\![\mathrm{AP}\,\mathit{red}\;\mathit{apple}]\!] &\Rightarrow^{\star}_{\beta} \lambda x.(red^{(1)}\,x) \wedge (apple^{(1)}\,x)\\
[\![\mathrm{NP}\,\mathit{a}\,\mathrm{AP}\,\mathit{red}\;\mathit{apple}]\!] &\Rightarrow^{\star}_{\beta} \lambda x.\exists u.(red^{(1)}\,u) \wedge (apple^{(1)}\,u) \wedge (x\,u)\\
[\![\mathrm{VP}\,\mathit{eats}\,\mathrm{NP}\,\mathit{a}\,\mathrm{AP}\,\mathit{red}\;\mathit{apple}]\!] &\Rightarrow^{\star}_{\beta} \lambda x.\exists e.(eat^{(1)}\,e) \wedge x(\lambda a.agent^{(2)}\,e\,a)\\
&\qquad \wedge \exists u.(red^{(1)}\,u) \wedge (apple^{(1)}\,u) \wedge (patient^{(2)}\,e\,u)\\
[\![\mathrm{S}\ldots]\!] &\Rightarrow^{\star}_{\beta} \exists e.(eat^{(1)}\,e) \wedge (agent^{(2)}\,e\;John^{(0)})\\
&\qquad \wedge \exists u.(red^{(1)}\,u) \wedge (apple^{(1)}\,u) \wedge (patient^{(2)}\,e\,u)\;,
\end{aligned}
$$

which is the λ-term encoding of (8.24).

(∗) **Exercise 10.4.** Propose similarly a higher-order homomorphism from the dependency structure of Figure 10.1 into its semantics.

10.1.4 Tree Transductions

The definition we provided for higher-order homomorphisms does not use types explicitly; this is easy to remedy:

Definition 10.9 (Typed Homomorphism)**.** A **typed homomorphism** between two signatures $\Sigma = \langle A, C, \tau\rangle$ and $\Sigma' = \langle A', C', \tau'\rangle$ extends a higher-order homomorphism $[\![.]\!]$ between C and C' by mapping each atomic type of A into a type of $\mathcal{T}_{A'}$ s.t. $\vdash_{\Sigma'} [\![c]\!] : [\![\tau(c)]\!]$ is a valid typing judgement for all c in C.

Example 10.10 (Higher-Order Tree Functions)**.** Let us see how this definition can be exercised to define tree transductions. We define the generic **tree signature** over a ranked alphabet $\mathcal{F}$ as $\Sigma_{\mathcal{F}} \overset{\text{def}}{=} \langle\{o\}, \mathcal{F}, \tau_{\mathcal{F}}\rangle$ where for every $f^{(n)}$ in $\mathcal{F}$, $\tau_{\mathcal{F}}(f^{(n)}) \overset{\text{def}}{=} \underbrace{o \to \cdots \to o}_{n\text{ times}} \to o = o^n \to o$.

Let Σ_C and $\Sigma_{C'}$ be two generic tree signatures over the ranked alphabets C and C', and let $[\![.]\!]$ be a typed homomorphism between Σ_C and $\Sigma_{C'}$, and $s \in A$

be a distinguished input atomic type with $[\![s]\!] = o$. We define the corresponding (partial) higher-order tree function $\mathcal{T}\colon T(C) \to T(C')$ by

$$\mathcal{T}(\bar{t}_1) = \bar{t}_2 \text{ iff } \vdash_\Sigma t_1 : s \wedge [\![t_1]\!] \Rightarrow^\star_{\beta\eta} t_2 \;. \tag{10.1}$$

Note that in this definition, because the bijection $\bar{.}$ between λ-terms and trees is only defined for ground λ-terms, t_2 must be in $\beta\eta$-normal form.

The semantic construction of Example 10.8 is a higher-order tree function when setting Σ_C and $\Sigma_{C'}$ as input and output signatures and if we consider e and v as nullary constants in C'.

Linear Higher-Order Tree Functions As often in linguistic applications, a case of particular interest is the *linear* one: a higher-order homomorphism between C and C' is **linear** if $[\![c]\!]$ is a linear term for every c in C.

See de Groote (2001).

Definition 10.11 (Abstract Categorial Grammar)**.** An **abstract categorial grammar** (ACG) is a tuple $\mathcal{G} = \langle \Sigma, \Sigma', [\![.]\!], s\rangle$ where $\Sigma = \langle A, C, \tau\rangle$ and $\Sigma' = \langle A', C', \tau'\rangle$ are two signatures, $[\![.]\!]$ is a *linear* typed homomorphism, and s in A is a distinguished atomic type. The **abstract language** $\mathscr{A}(\mathcal{G})$ of $\mathcal{G}$ is

$$\mathscr{A}(\mathcal{G}) \stackrel{\text{def}}{=} \{L \in \Lambda_\ell(C) \mid \vdash_\Sigma L : s\}$$

the set of closed linear λ-terms typed by s in the input signature, while its **object language** $\mathscr{O}(\mathcal{G})$ is

$$\mathscr{O}(\mathcal{G}) \stackrel{\text{def}}{=} [\![\mathscr{A}(\mathcal{G})]\!]$$

the set of linear λ-terms obtained through the application of the homomorphism $[\![.]\!]$ to abstract terms.

A **second-order** ACG is an ACG with a second-order abstract signature Σ. Such ACGs are arguably the most relevant for the linguistic applications. Note that our objects of interest are usually the *normal forms* found in the object language: these turn out to be exactly the normal forms of the images of the *ground* terms in $\mathscr{A}(\mathcal{G})$:

$$\Downarrow_{\beta\eta}\mathscr{O}(\mathcal{G}) = \Downarrow_{\beta\eta}\{[\![t]\!] \in \Lambda_\ell(C') \mid t \text{ ground} \in \mathscr{A}(\mathcal{G})\} \;. \tag{10.2}$$

This follows from $\Downarrow_{\beta\eta}[\![L]\!] = \Downarrow_{\beta\eta}[\![\Downarrow_{\beta\eta}L]\!]$ since $[\![.]\!]$ is a higher-order homomorphism, and the fact that a closed term L in normal form is of atomic type s iff it is ground (on a second-order signature).

Therefore, if the object signature is a generic tree signature $\Sigma_{C'}$, then a second-order ACG can be understood as defining a linear higher-order tree function from a local tree language (its abstract language) into the set of trees over C' (its object language). The following exercise examines the simplest such situation, where the homomorphic images of atomic types in the abstract signature are mapped to tree types o in the object signature:

Exercise 10.5 (Tree Languages of ACG$_{2,1}$)**.** Given an ACG $\mathcal{G} = \langle \Sigma, \Sigma_\mathcal{F}, [\![.]\!], s\rangle$ with a second-order abstract signature $\Sigma = \langle A, C, \tau\rangle$ and a generic tree signature $\Sigma_\mathcal{F}$ over some ranked alphabet $\mathcal{F}$ as object signature, we define its **tree language** as **(**)**

$$\mathscr{T}(\mathcal{G}) \stackrel{\text{def}}{=} \{\bar{t} \in T(\mathcal{F}) \mid t \text{ ground} \in \mathscr{O}(\mathcal{G})\} \;. \tag{10.3}$$

Assume that $\max_{a\in A} \operatorname{ord}([\![a]\!]) = 1$. Show that such ACGs generate exactly the set of regular tree languages.

More generally, the expressiveness of second-order ACGs has been studied by Kanazawa (2010): their object languages correspond to the tree languages of **context-free hyperedge replacement grammars**, which are also equivalent to **attributed context-free grammars** (Engelfriet and Heyker, 1992) and outputs of restricted forms of MTTs (Engelfriet and Maneth, 2000). This means that we could also implement the tree transformations defined by second-order ACGs using more classical tree transductions. However, this would be at the expense of the ability to view the translation as one into higher-order semantics, as we will do in Section 10.3. In that situation, we will no longer work with ground object terms.

10.2 Intensionality

This section is based on (Fitting, 2004) and the entry on intensional logic in the Stanford Encyclopedia of Philosophy.

Intensional Phenomena deal with the difference between a meaning and its denotation. A classical example given by Frege is concerned about equality in mathematics: if a and b designate the same object, and equality is about objects and not about their names, then there is no difference between "$a = b$" and "$a = a$". There is however a difference in informational content: the truth of these assertions depends on the context, and there exist contexts that differentiate between the two, namely those where a and b do not denote the same object.

Considering an example with more linguistic content, the sentence *John knows that the morning star is the evening star* might have different truth values depending on the extent of the knowledge of *John*, but if *morning star* and *evening star* are always mapped to the same object, namely Venus, we cannot model the case where John is not aware of their identity. Similar intensional phenomena can occur in relation with temporal modalities instead of epistemic ones: *The King of England was the head of the Church of England* holds true after King Henry VIII separated the Church from Rome in 1534, thus in worlds after 1534 where *the King of England* denotes Henry VIII or one of his successors; again an intensional reading should be preferred. A last classical example of Montague contrasts *John finds a unicorn* with *John seeks a unicorn*. These are structurally similar, but the first one implies that there exists a unicorn, while the second allows both readings: the so-called **de dicto** reading which does not imply the existence of unicorns, and the **de re** reading from which existence of unicorns follows. These two readings could be modelled using different scopes for the modal *seeks*.

Intensional Logic This reveals an issue with FOML: there is no way to map variables to different objects depending on the world under consideration. The solution adopted in **first-order intensional logic** (FOIL) is to use two sorts of variables, intensional and extensional ones. Intensions might denote different objects in different worlds: for instance if f is an intension and w is a world, then $f(w)$ would be the **extension** of f in w.

There is an issue with this account of intensionality. If f is an intension and P a unary predicate, then $P(f)$ could mean that the extension of f verifies P (*de re* reading), or that the intension f itself verifies P (*de dicto* reading). For instance, *The morning star is the evening star* would use a de re reading, but *The morning star is the last star seen in the morning* would be true regardless of the actual object denoted by *the morning star*. If we consider alethic modalities, $\Diamond P(f)$ might either mean that in some possible world w, $P(f(w))$ holds, or that in some possible world w', $P(f)$ holds. In order to distinguish between these alternatives, the de re reading is noted $[\lambda x.\Diamond P(x)](f)$ and the de dicto one $\Diamond[\lambda x.P(x)](f)$.

Given an infinite countable set of object variables $\mathcal{O}$ and an infinite countable set of intension variables $\mathcal{I}$, FOIL formulæ follow the syntax

$$\varphi ::= x = x' \mid R_i(y_1, \ldots, y_{k_i}) \mid [\lambda x.\varphi](f) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi \mid \exists y.\varphi$$

where x, x' range over $\mathcal{O}$, f over $\mathcal{I}$, $y, y_1, \ldots, y_{k_i}$ over $\mathcal{I}\uplus\mathcal{O}$, R_i is a k_i-ary relational symbol, and φ is a formula with a free object variable x, so that $[\lambda x.\varphi](f)$ denotes $\varphi\{x \leftarrow f\}$. We write $[\lambda xx'.\varphi](f, f')$ for $[\lambda x.[\lambda x'.\varphi](g)](f)$. This last construction is a form of *abstraction* limited to first-order.

Fitting (2004) also adds a typing discipline to the relations R_i to better differentiate between intensional and extensional arguments.

Intensional models for FOIL are of form $\mathfrak{M} = \langle W, R, D_{\mathcal{O}}, D_{\mathcal{I}}, I\rangle$ where a distinction is drawn between the *object domain* $D_{\mathcal{O}}$, which is a non-empty set in our constant semantics, and the *intension domain* $D_{\mathcal{I}}$, which is a non-empty set of functions from W to $D_{\mathcal{O}}$, and I maps a relational symbols R_i with arity k_i to a mapping $I(R_i)$ from W to relations over $(D_{\mathcal{O}} \cup D_{\mathcal{I}})^{k_i}$. A *valuation* is now a mapping assigning members of $D_{\mathcal{O}}$ to object variables and members of $D_{\mathcal{I}}$ to intension variables. The satisfiability relation is similar to that of FOML, with

$$\begin{aligned} \mathfrak{M}, w &\models_\nu \exists f.\varphi && \text{iff } \exists i \in D_{\mathcal{I}}(w).\mathfrak{M}, w \models_{\nu[f\leftarrow i]} \varphi \\ \mathfrak{M}, w &\models_\nu [\lambda x.\varphi](f) && \text{iff } \mathfrak{M}, w \models_{\nu[x\leftarrow\nu(f)(w)]} \varphi\ . \end{aligned}$$

Example 10.12 (Morning Star)**.** Let us consider again the sentence *The morning star is the evening star* and associate f to the intension *the morning star* and g to the intension *the evening star*. Then $[\lambda xx'.x = x'](f, g)$ is correct in the real word w, where f and g are associated to the same object $\nu(f)(w) = \nu(g)(w)$ in $D_{\mathcal{O}}$, namely Venus. In an epistemic setting, the de dicto reading $K[\lambda xx'.x = x'](f, g)$ can be falsified if we find another state of knowledge w' compatible with the real world w where this information is missing, i.e. where $\nu(f)(w') \neq \nu(g)(w')$—this could be the case in the sentence *John knows that the morning star is the evening star* if *John* is unaware of their both being Venus. By contrast, the de re reading $[\lambda xx'.K(x = x')](f, g)$ is always satisfied in w because in any state of knowledge compatible with the real world, f and g have received the same extension $\nu(f)(w) = \nu(g)(w)$.

Example 10.13 (King of England)**.** The treatment of the sentence *The King of England was the head of the Church of England* is similar: consider the intensions f for *the King of England*, g for *the head of the Church of England*, and a point in time w. Then $P[\lambda xx'.x = x'](f, g)$ could be invalidated if there is no past time $w' < w$ where the denotations $\nu(f)(w')$ and $\nu(g)(w')$ were the same—i.e. before the 1538 secession from the Roman Church—, but is valid in time points w after the secession. The de re reading does not make any sense: $[\lambda xx'.P(x = x')](f, g)$ holds iff $\nu(f)(w) = \nu(g)(w)$ at the time of interest, regardless of past times where equality is evaluated.

Total Intensionality Let $D(f, x)$ stand for $[\lambda x'.x = x'](f)$ where x and x' are distinct object variables. Then $\mathfrak{M}, w \models_\nu D(f, x)$ holds iff $\nu(f)(w) = \nu(x)$.

The formula $\forall f\exists x.D(f, x)$ is valid in intensional models as defined so far, since $\nu(f)$ is a total function from W to $D_{\mathcal{O}}$. There is however no requirement for every object to be designated by some intension, i.e. for

$$\forall x.\exists f.D(f, x) \tag{10.4}$$

to hold. This is however a reasonable restriction; let us check for instance the following equivalence under the hypothesis of (10.4):

$$\exists x.\varphi \equiv \exists f.[\lambda x.\varphi](f)\ . \tag{10.5}$$

Indeed, for all $\mathfrak{M}$, w, ν and φ,

$$\begin{aligned}
&\mathfrak{M}, w \models_\nu \exists f.[\lambda x.\varphi](f)\\
\text{iff }&\exists i \in D_{\mathcal{I}}.\mathfrak{M}, w \models_{\nu[f\leftarrow i]} [\lambda x.\varphi](f)\\
\text{iff }&\exists i \in D_{\mathcal{I}}.\mathfrak{M}, w \models_{\nu[f\leftarrow i, x\leftarrow i(w)]} \varphi\\
\text{iff }&\exists e \in D_{\mathcal{O}}.\mathfrak{M}, w \models_{\nu[x\leftarrow e]} \varphi && \text{(by (10.4) when choosing } i(w)=e\text{)}\\
\text{iff }&\mathfrak{M}, w \models_\nu \exists x.\varphi\ .
\end{aligned}$$

(∗) **Exercise 10.6.** Show the following equivalence when (10.4) holds:

$$\exists f.\Diamond[\lambda x.\varphi](f) \equiv \Diamond(\exists x.\varphi)\ . \tag{10.6}$$

Example 10.14 (Unicorn)**.** The sentence *John finds a unicorn* could be associated with the semantics

$$\exists ex.\mathit{find}^{(1)}(e) \wedge \mathit{agent}^{(2)}(e, \mathit{John}^{(0)}) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x) \tag{10.7}$$

but it is better to treat *unicorn* as an intension in the formula

$$\exists u.[\lambda x.\exists e.\mathit{find}^{(1)}(e) \wedge \mathit{agent}^{(2)}(e, \mathit{John}^{(0)}) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u)\ , \tag{10.8}$$

equivalent to (10.7) in totally intensional models according to (10.5). Then we better see the connection with the sentence *John seeks a unicorn*: its de dicto semantics would be

$$\begin{aligned}
&\exists u.\textsc{try}(\mathit{John}^{(0)}, [\lambda x.\exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u)) && (10.9)\\
&\equiv \textsc{try}(\mathit{John}^{(0)}, \exists ex.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)) && \text{(by (10.6))}
\end{aligned}$$

and its de re semantics

$$\begin{aligned}
&\exists u.[\lambda x.\textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u) && (10.10)\\
&\equiv \exists x.\textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)) && \text{(by (10.5))}
\end{aligned}$$

and if the interpretation of $\mathit{unicorn}^{(1)}$ is the same in all worlds accessible through the TRY modality,

$$\equiv \exists x.\mathit{unicorn}^{(1)}(x) \wedge \textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x))\ .$$

10.3 Higher-Order Logic

Most of the discussion on semantic representations can be recast in the framework of higher-order logic. This allows in particular to view the higher-order operations of Section 10.1 not as a technical means to generate λ-terms viewed as trees (which in turn can be interpreted in some logic), but instead to interpret these terms directly in the higher-order logic. They become the semantics of the sentences under consideration, with associated models.

10.3.1 *Background:* Church's Simple Theory of Types

See Church (1940) and the entry in the Stanford Encyclopedia of Philosophy.

Higher-order semantics are typically expressed in simply typed lambda calculus as defined in Section 10.1.1. As we want not just to manipulate typed λ-terms, but also to be able to infer truths, we need to introduce a set of **logical constants** and the associated **logical rules**.

Higher-Order Signature In Church's simple theory of types, we use a signature $\Sigma = \langle A, C, t \rangle$ where $A = \{\iota, o\}$ is set of atomic types, where ι denotes *entities* and o *truths*. The logical constants are $C = \{\bot, \supset, (\forall_\tau)_{\tau \in \mathcal{T}(A)}\}$ with types $t(\bot) = o$, $t(\supset) = o \to o \to o$, and $(\forall_\tau) = (\tau \to o) \to o$ for each type τ in $\mathcal{T}(A)$.

We write as usual $L \supset M$ for $\supset L\,M$ and $\forall_\tau x.L$ for $\forall_\tau(\lambda x.L)$. The other logical connectives are defined as usual: $\neg L \overset{\text{def}}{=} L \supset \bot$, $L \vee M \overset{\text{def}}{=} (\neg L) \supset M$, $L \wedge M \overset{\text{def}}{=} \neg((\neg L) \vee (\neg M))$, etc. Equality is defined in the Leibnizian way as $L = M \overset{\text{def}}{=} \forall x.x\,L \supset x\,M$, i.e. equality is defined as having L and M agree on all possible properties x.

Logical and Conversion Rules The formal system needs two types of rules: logical rules for the logical constants, and conversion rules for the λ-terms. In natural deduction sequent style,

$$\frac{}{\Gamma, L \Vdash L}\,(\mathsf{Ax}) \qquad \frac{\Gamma, \neg L \Vdash \bot}{\Gamma \Vdash L}\,(\bot\mathsf{E})$$

$$\frac{\Gamma, L \Vdash M}{\Gamma \Vdash L \supset M}\,(\supset\mathsf{I}) \qquad \frac{\Gamma \Vdash L \supset M \quad \Gamma \Vdash L}{\Gamma \Vdash M}\,(\supset\mathsf{E})$$

$$\frac{\Gamma \Vdash L \quad x \notin \mathrm{FV}(\Gamma)}{\Gamma \Vdash \forall_\tau x.L}\,(\forall\mathsf{I}) \qquad \frac{\Gamma \Vdash \forall_\tau L \quad \Delta \vdash_\Sigma M : \tau}{\Gamma \Vdash L\,M}\,(\forall\mathsf{E})$$

$$\frac{\Gamma \Vdash L \quad L =_\beta M}{\Gamma \Vdash M}\,(\beta)$$

The deduction system also often includes the **extensionality axioms**:

$$\frac{}{\Gamma \Vdash (\forall_\tau x.L\,x = M\,x) \supset (L = M)}\,(\lambda\mathsf{X}) \qquad \frac{}{\Gamma \Vdash (L \equiv M) \supset (L = M)}\,(\equiv\mathsf{X})$$

As their name indicates, the extensionality axioms make the simple theory of types unable to deal with intensional phenomena directly; a solution we will see in Section 10.3.2 will be to introduce an new atomic type s ranging over *worlds*.

More axioms are used in the simple theory of types; see Church (1940).

Higher-order logic can express a form of set theory: view the set comprehension $\{x \mid P\}$ as $\lambda x.P$, or $e \in E$ as $E\,e$. In fact, Church (1940) shows how to implement Peano's arithmetic in the simple theory of types, from which we can deduce the incompleteness of higher-order logic.

Standard Models Higher-order logic comes with a very natural model theory. For each τ in $\mathcal{T}(A)$, let D_τ be the domain of expressions of type τ. Let $D_o = \{\top, \bot\}$ and D_ι be some set of entities; then $D_{\tau\to\rho}$ denotes the set of functions from D_τ to D_ρ, so that e.g. $D_{\iota\to o}$ is the type of first-order predicates.

See also Henkin (1950).

10.3.2 Type-Logical Semantics

Many classical modellings of natural language semantics in higher-order logic posit an additional type s of **worlds** in order to account for modalities and intensionality phenomena. The idea is to always treat truth values (of type o) as relativized with respect to a possible world of evaluation. Thus we will consider a

We follow Muskens (2011) for this section, itself based on Gallin (1975). See also the entry on Montague semantics in the Stanford Encyclopedia of Philosophy.

syntactic category	examples	type
intransitive verbs	walk, talk, eat$_1$, ...	$\iota \to s \to o$
transitive verbs	eat$_2$, love, ...	$\iota \to \iota \to s \to o$
common nouns	apple, man, woman, ...	$\iota \to s \to o$
adjectives	red, ...	$\iota \to s \to o$
determiners	every, a, the, no, ...	$(\iota \to s \to o) \to (\iota \to s \to o) \to s \to o$
proper nouns	John, Mary, ...	ι
modal adverbs	necessarily, possibly, ...	$(s \to o) \to s \to o$
modal verbs	know, believe, ...	$(s \to o) \to \iota \to s \to o$
negation	not	$(s \to o) \to s \to o$

Table 10.1: Some constants and their possible types.

$$\begin{aligned}
[\![\text{walk}]\!] &= walk_{\iota\to s\to o}\\
[\![\text{eat}_2]\!] &= eat2_{\iota\to\iota\to s\to o}\\
[\![\text{apple}]\!] &= apple_{\iota\to s\to o}\\
[\![\text{red}]\!] &= \lambda P_{\iota\to s\to o} x_\iota w_s . red_{\iota\to s\to o}\, x\, w \land P\, x\, w\\
[\![\text{every}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \forall_\iota x . (P\, x\, w \supset P'\, x\, w)\\
[\![\text{a}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \exists_\iota x . (P\, x\, w \land P'\, x\, w)\\
[\![\text{no}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \forall_\iota x . (P\, x\, w \supset \neg P'\, x\, w)\\
[\![\text{the}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \exists_\iota x . (P'\, x\, w \land \forall_\iota y . (P\, x\, w \equiv x = y))\\
[\![\text{John}]\!] &= John_\iota\\
[\![\text{necessarily}]\!] &= \lambda p_{s\to o} w_s . \forall_s w' . (R_{s\to s\to o}\, w\, w') \supset p\, w'\\
[\![\text{possibly}]\!] &= \lambda p_{s\to o} w_s . \exists_s w' . (R_{s\to s\to o}\, w\, w') \land p\, w'\\
[\![\text{know}]\!] &= \lambda p_{s\to o} x_\iota w_s . \forall_s w' . (K_{\iota\to s\to s\to o}\, x\, w\, w') \supset p\, w'\\
[\![\text{believe}]\!] &= \lambda p_{s\to o} x_\iota w_s . \forall_s w' . (B_{\iota\to s\to s\to o}\, x\, w\, w') \supset p\, w'\\
[\![\text{not}]\!] &= \lambda p_{s\to o} w_s . \neg\, p\, w
\end{aligned}$$

Table 10.2: Examples of semantics associated with lexical elements.

higher-order signature $\Sigma = \langle A, \{\bot, \supset, (\forall_\tau)_{\tau\in\mathcal{T}(A)}\} \cup C, t\rangle$ as in the simple theory of types, where $A = \{s, \iota, o\}$ and C denotes additional non-logical constants. To simplify matters, we avoid explicit events from Section 8.1.2.

Due to the relativisation wrt. worlds, a simple sentence like *John walks* is expected to be of type $s \to o$ and to be associated to a logical representation like

$$walks\ John\ . \tag{10.11}$$

In order to obtain the appropriate type, a possibility is to set $t(walks) = \iota \to s \to o$ and $t(John) = \iota$. Looking at more complex examples (for instance Example 10.8), we arrive at the types of Table 10.1. The semantics of a sentence can then be computed by a higher-order homomorphism as in Section 10.1, but there will be no need to translate back from λ-terms to first-order terms in order to reason about the semantics: the λ-term is a meaning representation with full-fledged model theory. See Table 10.2 for some examples of semantic values.

Observe that we introduced an explicit type for worlds in the logic: this can be avoided if we use ***intensional models*** *as in (Muskens, 2007). Recall that Church's simple type theory verifies the extensionality axioms!*

In this table, the semantics of alethic and epistemic modal logics have been implemented directly using the R, K, and B constants with types $s \to s \to o$,

$\iota \to s \to s \to o$, and $\iota \to s \to s \to o$ respectively. The desired properties of these relations can also be enforced; for instance $\forall_s ww'.\ R\,ww'$ forces R to be total.

Chapter 11

References

Afanasiev, L., Blackburn, P., Dimitriou, I., Gaiffe, B., Goris, E., Marx, M., and de Rijke, M., 2005. PDL for ordered trees. *Journal of Applied Non-Classical Logic*, 15(2):115–135. doi:10.3166/jancl.15.115-135. Cited on pages 48, 55.

Aho, A.V., 1968. Indexed grammars—An extension of context-free grammars. *Journal of the ACM*, 15(4):647–671. doi:10.1145/321479.321488. Cited on page 68.

Ajdukiewicz, K., 1935. Die syntaktische Konnexität. *Studia Philisophica*, 1:1–27. Cited on pages 91, 92.

Althaus, E., Duchier, D., Koller, A., Mehlhorn, K., Niehren, J., and Thiel, S., 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48(1):194–219. doi:10.1016/S0196-6774(03)00050-6. Cited on pages 119, 120, 122.

Andréka, H., van Benthem, J., and Németi, I., 1998. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274. doi:10.1023/A:1004275029985. Cited on page 109.

Asterias, A., Dawar, A., and Kolaitis, P.G., 2006. On preservation under homomorpisms and unions of conjunctive queries. *Journal of the ACM*, 53(2):208–237. doi:10.1145/1131342.1131344. Cited on page 115.

Asterias, A., Dawar, A., and Grohe, M., 2008. Preservation under extensions on well-behaved finite structures. *SIAM Journal on Computing*, 38(4):1364–1381. doi:1.1137/060658709. Cited on page 114.

Baader, F., Horrocks, I., and Sattler, U., 2007. *Description Logics*, volume 3 of *Foundations of Artificial Intelligence*, chapter 3, pages 135–179. Elsevier. doi:10.1016/S1574-6526(07)03003-9. Cited on page 102.

Backus, J.W., 1959. The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM Conference. In *IFIP Congress*, pages 125–131. Cited on page 10.

Bar-Hillel, Y., 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29 (1):47–58. doi:10.2307/410452. Cited on pages 12, 91, 92.

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 93.

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 40, 43.

Bárány, V., ten Cate, B., and Segoufin, L., 2011. Guarded negation. In Aceto, L., Henzinger, M., and Sgall, J., editors, *ICALP 2011, 38th International Colloquium on Automata, Languages and Programming*, volume 6756 of *Lecture Notes in Computer Science*, pages 356–367. Springer. doi:10.1007/978-3-642-22012-8_28. Cited on page 111.

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 19, 21.

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 19.

Billot, S. and Lang, B., 1989. The structure of shared forests in ambiguous parsing. In *ACL'89, 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151. ACL Press. doi:10.3115/981623.981641. Cited on page 40.

Björklund, H., Martens, W., and Schwentick, T., 2011. Conjunctive query containment over trees. *Journal of Computer and System Sciences*, 77(3):450–472. doi:10.1016/j.jcss.2010.04.005. Cited on pages 116, 117.

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, Fourth Workshop on Speech and Natural Language*, pages 306–311. ACL Press. doi:10.3115/112405.112467. Cited on page 84.

Blackburn, P., Gardent, C., and Meyer-Viol, W., 1993. Talking about trees. In *EACL '93, Sixth Meeting of the European Chapter of the Association for Computational Linguistics*, pages 21–29. ACL Press. doi:10.3115/976744.976748. Cited on page 55.

Blackburn, P., Meyer-Viol, W., and Rijke, M.d., 1996. A proof system for finite trees. In Kleine Büning, H., editor, *CSL '95, 9th International Workshop on Computer Science Logic*, volume 1092 of *Lecture Notes in Computer Science*, pages 86–105. Springer. doi:10.1007/3-540-61377-3_33. Cited on page 55.

Blackburn, P., de Rijke, M., and Venema, Y., 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press. Cited on pages 56, 104, 106, 107.

Blackburn, P. and Bos, J., 2005. *Representation and Inference for Natural Language: A First Course in Computational Semantics*. CSLI Studies in Computational Linguistics. CSLI Publications. ISBN 1-57586-496-7. Cited on pages 119, 129.

Blondel, V.D. and Canterini, V., 2003. Undecidable problems for probabilistic automata of fixed dimension. 36(3):231–245. doi:10.1007/s00224-003-1061-2. Cited on page 86.

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 79, 80.

Boral, A. and Schmitz, S., 2013. Model checking parse trees. In *LICS 2013, Twenty-Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 153–162. IEEE Press. doi:10.1109/LICS.2013.21. Cited on page 62.

Börger, E., Grädel, E., and Gurevich, Y., 1997. *The Classical Decision Problem*. Perspectives Mathematical Logic. Springer. Cited on page 108.

Bos, J., 1996. Predicate logic unplugged. In Dekker, P. and Stokhof, M., editors, *AC '96, Tenth Amsterdam Colloquium*, pages 133–143. ILLC/Department of Philosophy, University of Amsterdam. Cited on page 119.

Brants, T., 2000. TnT – a statistical part-of-speech tagger. In *ANLP 2000, 6th Conference on Applied Natural Language Processing*, pages 224–231. doi:10.3115/974147.974178. Cited on page 26.

Brill, E., 1992. A simple rule-based part of speech tagger. In *ANLP '92, third Conference on Applied Natural Language Processing*, pages 152–155. ACL Press. doi:10.3115/974499.974526. Cited on pages 26, 27, 28.

Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M., 2009. An automata-theoretic approach to Regular XPath. In Gardner, P. and Geerts, F., editors, *DBPL 2009, 12th International Symposium on Database Programming Languages*, volume 5708 of *Lecture Notes in Computer Science*, pages 18–35. Springer. doi:10.1007/978-3-642-03793-1_2. Cited on page 58.

Casacuberta, F. and de la Higuera, C., 2000. Computational complexity of problems on probabilistic grammars and transducers. In Oliveira, A.L., editor, *ICGI 2000, 5th International Conference on Grammatical Inference: Algorithms and Applications*, volume 1891 of *Lecture Notes in Artificial Intelligence*, pages 15–24. Springer. doi:10.1007/978-3-540-45257-7_2. Cited on page 87.

Charniak, E., 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI '97/IAAI '97*, pages 598–603. AAAI Press. Cited on page 12.

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 82.

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 pages 10, 39.

Chomsky, N., 1957. *Syntactic Structures*. Mouton de Gruyter. Cited on page 77.

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 39.

Chomsky, N. and Halle, M., 1968. *The Sound Pattern of English*. Harper and Row. Cited on page 23.

Church, A., 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68. doi:10.2307/2266170. Cited on pages 126, 134, 135.

Cocke, J. and Schwartz, J.T., 1970. *Programming languages and their compilers*. Courant Institute of Mathematical Sciences, New York University. Cited on page 40.

Collins, M., 1999. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania. http://www.cs.columbia.edu/~mcollins/papers/thesis.ps. Cited on page 52.

Collins, M., 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589–637. doi:10.1162/089120103322753356. Cited on page 12.

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 4, 10, 41, 49, 54, 68.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I., 2005. Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(2):281–332. doi:10.1007/s11168-006-6327-9. Cited on page 119.

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 45.

Crabbé, B., 2005. Grammatical development with XMG. In Blache, P., Stabler, E., Busquets, J., and Moot, R., editors, *LACL 2005, 5th International Conference on Logical Aspects of Computational Linguistics*, 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 67.

Crabbé, B., Duchier, D., Gardent, C., Le Roux, J., and Parmentier, Y., 2013. XMG: eXtensible MetaGrammar. *Computational Linguistics*, 39(3):591–629. doi:10.1162/COLI_a_00144. Cited on page 116.

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 28.

Davidson, D., 1967. The logical form of action sentences. In Rescher, N., editor, *The Logic of Decision and Action*. University of Pittsburgh Press. doi:10.1093/0199246270.001.0001. Cited on page 101.

Dawar, A., 2010. Homomorphism preservation on quasi-wide classes. *Journal of Computer and System Sciences*, 76(5):324–332. doi:10.1016/j.jcss.2009.10.005. Cited on page 115.

de Groote, P., 2001. Towards abstract categorial grammars. In *ACL 2001, 39th Annual Meeting of the Association for Computational Linguistics*, pages 252–259. ACL Press. doi:10.3115/1073012.1073045. Cited on pages 67, 129, 131.

de la Higuera, C. and Oncina, J., 2011. Finding the most probable string and the consensus string: an algorithmic study. In *IWPT 2011, 12th International Workshop on Parsing Technologies*, pages 26–36. ACL Press. http://www.aclweb.org/anthology/W11-2904. Cited on pages 86, 89.

de la Higuera, C. and Oncina, J., 2013. Computing the most probable string with a probabilistic finite state machine. In *FSMNLP 2013, 11th International Conference on Finite State Methods and Natural Language Processing*, pages 1–8. ACL Press. http://www.aclweb.org/anthology/W13-1801. Cited on pages 86, 89.

Doner, J., 1970. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451. doi:10.1016/S0022-0000(70)80041-1. Cited on page 54.

Duchier, D. and Debusmann, R., 2001. Topological dependency trees: a constraint-based account of linear precedence. In *ACL 2001, 39th Annual Meeting of the Association for Computational Linguistics*, pages 180–187. Annual Meeting of the Association for Computational Linguistics. doi:10.3115/1073012.1073036. Cited on page 12.

Duchier, D., Prost, J.P., and Dao, T.B.H., 2009. A model-theoretic framework for grammaticality judgements. In *FG 2009, 14th International Conference on Formal Grammar*. http://hal.archives-ouvertes.fr/hal-00458937/. Cited on page 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 40, 45.

Ebbinghaus, H.D. and Flum, J., 1999. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer. Cited on page 114.

Egg, M., Koller, A., and Niehren, J., 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, 10(4):457–485. doi:10.1023/A:1017964622902. Cited on page 119.

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 73.

Engelfriet, J. and Heyker, L., 1992. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29(2):161–210. doi:10.1007/BF01178504. Cited on page 132.

Engelfriet, J. and Maneth, S., 2000. Tree languages generated by context-free graph grammars. In Ehrig, H., Engels, G., Kreowski, H.J., and Rozenberg, G., editors, *TAGT '98, 6th International Workshop on Theory and Application of Graph Transformations*, volume 1764 of *Lecture Notes in Computer Science*, pages 15–29. Springer. doi:10.1007/978-3-540-46464-8_2. Cited on page 132.

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 pages 80, 81.

Filmus, Y., 2011. Lower bounds for context-free grammars. *Information Processing Letters*, 111(18):895–898. doi:10.1016/j.ipl.2011.06.006. Cited on page 123.

Fischer, M.J., 1968. Grammars with macro-like productions. In *SWAT '68, 9th Annual Symposium on Switching and Automata Theory*, pages 131–142. IEEE Computer Society. doi:10.1109/SWAT.1968.12. Cited on pages 68, 69, 70, 73.

Fischer, M.J. and Ladner, R.E., 1979. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211. doi:10.1016/0022-0000(79)90046-1. Cited on pages 55, 56.

Fitting, M., 2004. First-order intensional logic. *Annals of Pure and Applied Logic*, 127 (1–3):173–193. doi:10.1016/j.apal.2003.11.014. Cited on pages 132, 133.

Fujiyoshi, A. and Kasai, T., 2000. Spinal-formed context-free tree grammars. *Theory of Computing Systems*, 33(1):59–83. doi:10.1007/s002249910004. Cited on page 70.

Gaifman, H., 1965. Dependency systems and phrase-structure systems. *Information and Control*, 8(3):304–337. doi:10.1016/S0019-9958(65)90232-9. Cited on page 12.

Gallin, D., 1975. *Intensional and Higher-Order Modal Logic*, volume 19 of *Mathematic Studies*. Elsevier. ISBN 0-444-11002-X. Cited on page 135.

Ganzinger, H., Meyer, C., and Veanes, M., 1999. The two-variable guarded fragment with transitive relations. In *LICS '99, 14th Annual IEEE Symposium on Logic in Computer Science*, pages 24–34. IEEE Computer Society. doi:10.1109/LICS.1999.782582. Cited on page 112.

Gardent, C. and Kallmeyer, L., 2003. Semantic construction in feature-based TAG. In *EACL 2003, Tenth Meeting of the European Chapter of the Association for Computational Linguistics*, pages 123–130. ACL Press. ISBN 1-333-56789-0. doi:10.3115/1067807.1067825. Cited on page 67.

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 80.

Gécseg, F. and Steinby, M., 1997. Tree languages. In Rozenberg, G. and Salomaa, A., editors, *Hanbook of Formal Languages*, volume 3: Beyond Words, chapter 1. Springer. ISBN 3-540-60649-1. Cited on page 68.

Ginsburg, S. and Rice, H.G., 1962. Two families of languages related to ALGOL. *Journal of the ACM*, 9(3):350–371. doi:10.1145/321127.321132. Cited on page 10.

Girard, J.Y., 1987. Linear logic. *Theoretical Computer Science*, 50(1):1–101. doi:10.1016/0304-3975(87)90045-4. Cited on page 95.

Grädel, E., Kolaitis, P.G., and Vardi, M.Y., 1997. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69. doi:10.2307/421196. Cited on page 108.

Grädel, E. and Walukiewicz, I., 1999. Guarded fixed-point logic. In *LICS '99, 14th Annual IEEE Symposium on Logic in Computer Science*, pages 45–54. IEEE Computer Society. doi:10.1109/LICS.1999.782585. Cited on pages 109, 111.

Grädel, E., 2002. Guarded fixed point logics and the monadic theory of countable trees. *Theoretical Computer Science*, 288(1):129–152. doi:10.1016/S0304-3975(01)00151-7. Cited on page 109.

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 40.

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 93.

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 40.

Guessarian, I., 1983. Pushdown tree automata. 16(1):237–263. doi:10.1007/BF01744582. Cited on page 68.

Harel, D., Kozen, D., and Tiuryn, J., 2000. *Dynamic Logic*. Foundations of Computing. MIT Press. Cited on page 56.

Hays, D.G., 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525. http://www.jstor.org/stable/411934. Cited on page 12.

Henkin, L., 1950. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2): 81–91. doi:http://dx.doi.org/10.2307/2266967. Cited on page 135.

Hidders, J., 2004. Satisfiability of XPath expressions. In Lausen, G. and Suciu, D., editors, *DBPL 2003, 9th International Conference on Database Programming Languages*, volume 2921 of *Lecture Notes in Computer Science*, pages 21–36. Springer. doi:10.1007/978-3-540-24607-7_3. Cited on page 116.

Hindley, J.R., 1997. *Basic Simple Type Theory*, volume 42 of *Cambride Tracts in Theoretical Computer Science*. Cambridge University Press. ISBN 0-521-46518-4. doi: 10.1017/CBO9780511608865. Cited on pages 126, 127.

Hobbs, J.R. and Shieber, S.M., 1987. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13(1–2):47–63. http://aclweb.org/anthology/J87-1005.pdf. Cited on page 119.

Janssen, T.M., 1997. Compositionality. In Benthem, J.F. and ter Meulen, A., editors, *Handbook of Logic and Language*, chapter 7, pages 417–473. Elsevier. ISBN 0-444-81714-3. doi:10.1016/B978-044481714-3/50011-4. Cited on page 125.

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 40.

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 64.

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 63.

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 63.

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://www.seas.upenn.edu/~joshi/joshi-schabes-tag-97.pdf. Cited on page 64.

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 13, 16, 34, 82, 99.

Kallmeyer, L. and Romero, M., 2004. LTAG semantics with semantic unification. In Rambow, O. and Stone, M., editors, *TAG+7, Seventh International Workshop on Tree-Adjoining Grammars and Related Formalisms*, pages 155–162. http://www.cs.rutgers.edu/TAG+7/papers/kallmeyer-c.pdf. Cited on page 67.

Kallmeyer, L. and Kuhlmann, M., 2012. A formal model for plausible dependencies in lexicalized tree adjoining grammar. In *TAG+11, 11th International Workshop on Tree-Adjoining Grammars and Related Formalisms*, pages 108–116. http://user.phil-fak.uni-duesseldorf.de/~kallmeyer/papers/KallmeyerKuhlmann-TAG+11.pdf. Cited on page 67.

Kanazawa, M., 2007. Parsing and generation as Datalog queries. In *ACL 2007, 45th Annual Meeting of the Association for Computational Linguistics*, pages 176–183. Annual Meeting of the Association for Computational Linguistics. http://www.aclweb.org/anthology/P07-1023. Cited on page 129.

Kanazawa, M., 2009. The pumping lemma for well-nested multiple context-free languages. In Diekert, V. and Nowotka, D., editors, *DLT 2009, 13th International Conference on Developments in Language Theory*, volume 5583 of *Lecture Notes in Computer Science*, pages 312–325. Springer. doi:10.1007/978-3-642-02737-6_25. Cited on page 73.

Kanazawa, M., 2010. Second-order abstract categorial grammars as hyperedge replacement grammars. *Journal of Logic, Language, and Information*, 19(2):137–161. doi:10.1007/s10849-009-9109-6. Cited on page 132.

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 25.

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 22.

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 16.

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 40.

Kepser, S. and Mönnich, U., 2006. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*, 354(1):82–97. doi:10.1016/j.tcs.2005.11.024. Cited on page 74.

Kepser, S., 2004. Querying linguistic treebanks with monadic second-order logic in linear time. *Journal of Logic, Language, and Information*, 13(4):457–470. doi:10.1007/s10849-004-2116-8. Cited on page 51.

Kepser, S. and Rogers, J., 2011. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. *Journal of Logic, Language, and Information*, 20(3):361–384. doi:10.1007/s10849-011-9134-0. Cited on pages 70, 73.

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 40.

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 84, 85.

Koller, A., Niehren, J., and Treinen, R., 2001. Dominance constraints: Algorithms and complexity. In Moortgat, M., editor, *LACL 1998, Third International Conference on Logical Aspects of Computational Linguistics*, volume 2014 of *Lecture Notes in Computer Science*, pages 106–125. doi:10.1007/3-540-45738-0_7. Cited on page 116.

Koller, A., Niehren, J., and Thater, S., 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *EACL 2003, 10th Meeting of the European Chapter of the Association for Computational Linguistics*, pages 195–202. ACL Press. doi:10.3115/1067807.1067834. Cited on page 121.

Koller, A., Regneri, M., and Thater, S., 2008. Regular tree grammars as a formalism for scope underspecification. In *ACL 2008:HLT, 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 218–226. ACL Press. http://www.aclweb.org/anthology/P08-1026. Cited on page 122.

Koskenniemi, K. and Church, K.W., 1988. Complexity, two-level morphology and Finnish. In *CoLing '88, 12th International Conference on Computational Linguistics*, pages 335–340. ACL Press. doi:10.3115/991635.991704. Cited on page 23.

Kracht, M., 1995. Syntactic codes and grammar refinement. *Journal of Logic, Language and Information*, 4(1):41–60. doi:10.1007/BF01048404. Cited on page 55.

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 66.

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 63.

Kuhlmann, M., 2013. Mildly non-projective dependency grammar. 39(2):355–387. doi:10.1162/COLI_a_00125. Cited on page 73.

Kupfermana, O., Pnueli, A., and Vardi, M.Y., 2012. Once and for all. *Journal of Computer and System Sciences*, 78(3):981–996. doi:10.1016/j.jcss.2011.08.006. Cited on page 105.

Kurki-Suonio, R., 1969. Notes on top-down languages. *BIT Numerical Mathematics*, 9 (3):225–238. doi:10.1007/BF01946814. Cited on page 40.

Lai, C. and Bird, S., 2010. Querying linguistic trees. *Journal of Logic, Language, and Information*, 19(1):53–73. doi:10.1007/s10849-009-9086-9. Cited on page 56.

Lambek, J., 1958. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170. doi:10.2307/2310058. Cited on pages 13, 94, 95.

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 95.

Lang, B., 1974. Deterministic techniques for efficient non-deterministic parsers. In Loeckx, J., editor, *ICALP'74, 2nd International Colloquium on Automata, Languages and Programming*, volume 14 of *Lecture Notes in Computer Science*, pages 255–269. Springer. doi:10.1007/3-540-06841-4_65. Cited on page 40.

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 43.

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 40.

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 46.

Lewis, H.R., 1980. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21(3):317–353. doi:10.1016/0022-0000(80)90027-6. Cited on page 108.

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 37.

Maletti, A. and Satta, G., 2009. Parsing algorithms based on tree automata. In *IWPT 2009, 11th International Workshop on Parsing Technologies*, pages 1–12. ACL Press. http://www.aclweb.org/anthology/W09-3801.pdf. Cited on page 84.

Maneth, S., Perst, T., and Seidl, H., 2007. Exact XML type checking in polynomial time. In Schwentick, T. and Suciu, D., editors, *ICDT 2007, 11th International Conference on Database Theory*, volume 4353 of *Lecture Notes in Computer Science*, pages 254–268. Springer. doi:10.1007/11965893_18. Cited on page 75.

Manning, C.D. and Schütze, H., 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. ISBN 978-0-262-13360-9. Cited on pages 10, 13, 34, 82.

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 17, 18, 26, 82.

Martin, W.A., Church, K.W., and Patil, R.S., 1987. Preliminary analysis of a breadth-first parsing algorithm: Theoretical and experimental results. In Bolc, L., editor, *Natural Language Parsing Systems*, Symbolic Computation, pages 267–328. Springer. doi:10.1007/978-3-642-83030-3_8. Cited on page 9.

Marx, M., 2005. Conditional XPath. *ACM Transactions on Database Systems*, 30(4):929–959. doi:10.1145/1114244.1114247. Cited on pages 55, 61.

Marx, M. and de Rijke, M., 2005. Semantic characterizations of navigational XPath. *SIGMOD Record*, 34(2):41–46. doi:10.1145/1083784.1083792. Cited on page 55.

Maryns, H. and Kepser, S., 2009. MonaSearch — a tool for querying linguistic treebanks. In Van Eynde, F., Frank, A., De Smedt, K., and van Noord, G., editors, *TLT 7, 7th International Workshop on Treebanks and Linguistic Theories*, pages 29–40. http://lotos.library.uu.nl/publish/articles/000260/bookpart.pdf. Cited on page 51.

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 24.

McCarthy, J.J., 1982. Prosodic structure and expletive infixation. *Language*, 58(3):574–590. doi:10.2307/413849. Cited on page 16.

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 24.

Mel'čuk, I.A., 1988. *Dependency syntax: Theory and practice*. SUNY Press. Cited on page 11.

Meyer, A., 1975. Weak monadic second order theory of successor is not elementary-recursive. In Parikh, R., editor, *Logic Colloquium '75*, volume 453 of *Lecture Notes in Mathematics*, pages 132–154. Springer. doi:10.1007/BFb0064872. Cited on pages 48, 54.

Michaliszyn, J., 2009. Decidability of the guarded fragment with the transitive closure. In Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., and Thomas, W., editors, *ICALP 2009, 36th International Colloquium on Automata, Languages and Programming*, volume 5556 of *Lecture Notes in Computer Science*, pages 261–272. Springer. doi:10.1007/978-3-642-02930-1_22. Cited on page 112.

Mohri, M. and Sproat, R., 1996. An efficient compiler for weighted rewrite rules. In *ACL '96, 34th Annual Meeting of the Association for Computational Linguistics*, pages 231–238. ACL Press. doi:10.3115/981863.981894. Cited on page 25.

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 37.

Mönnich, U., 1997. Adjunction as substitution: An algebraic formulation of regular, context-free and tree adjoining languages. In *FG '97, Second Conference on Formal Grammar*. arXiv:cmp-lg/9707012. Cited on page 70.

Montague, R., 1970. Universal grammar. *Theoria*, 36(3):373–398. doi:10.1111/j.1755-2567.1970.tb00434.x. Cited on page 126.

Montague, R., 1973. The proper treatment of quantification in ordinary English. In Hintikka, J., Moravcsik, J., and Suppes, P., editors, *Approaches to Natural Language*, pages 221–242. Reidel. https://www.blackwellpublishing.com/content/BPL_Images/Content_store/Sample_chapter/0631215417/Portner.pdf. Cited on pages 126, 129.

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 pages 9, 41.

Moortgat, M., 1997. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3–4):349–385. doi:10.1007/BF00159344. Cited on page 91.

Morrill, G.V., 1994. *Type Logical Grammar*. Kluwer Academic Publishers. ISBN 0-7923-3095-1. Cited on page 91.

Muskens, R., 2007. Intensional models for the theory of types. *Journal of Symbolic Logic*, 72(1):98–118. doi:10.2178/jsl/1174668386. Cited on page 136.

Muskens, R., 2011. Type-logical semantics. In Craig, E., editor, *Routledge Encyclopedia of Philosophy Online*. Routledge. http://let.uvt.nl/general/people/rmuskens/pubs/rep.pdf. (to appear). Cited on page 135.

Nederhof, M.J. and Satta, G., 2004. Tabular parsing. In MartÃ■n-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 43.

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 79.

Otto, M., 2004. Modal and guarded characterisation theorems over finite transition systems. *Annals of Pure and Applied Logic*, 130(1–3):173–205. doi:10.1016/j.apal.2004.04.003. Cited on page 107.

Palm, A., 1999. Propositional tense logic of finite trees. In *MOL 6, 6th Biennial Conference on Mathematics of Language*. http://www.phil.uni-passau.de/linguistik/palm/papers/mol99.pdf. Cited on page 55.

Parikh, R.J., 1966. On context-free languages. *Journal of the ACM*, 13(4):570–581. doi:10.1145/321356.321364. Cited on page 63.

Parsons, T., 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*, volume 19 of *Current Studies in Linguistics*. MIT Press. ISBN 0-262016120-6. http://www.humnet.ucla.edu/humnet/phil/faculty/tparsons/EventSemantics/download.htm. Cited on page 101.

Partee, B.H., ter Meulen, A.G., and Wall, R.E., 1990. *Mathematical Methods in Linguistics*, volume 30 of *Studies in Linguistics and Philosophy*. Springer. Cited on page 125.

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 97.

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 96.

Pereira, F.C.N. and Warren, D.H.D., 1983. Parsing as deduction. In *ACL '83, 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144. ACL Press. doi:10.3115/981311.981338. Cited on page 45.

Pereira, F., 2000. Formal grammar and information theory: together again? *Philosophical Transactions of the Royal Society A*, 358(1769):1239–1253. doi:10.1098/rsta.2000.0583. Cited on pages 10, 77.

Pesetsky, D., 1985. Morphology and logical form. *Linguistic Inquiry*, 16(2):193–246. http://www.jstor.org/stable/4178430. Cited on page 22.

Poesio, M., 1994. Ambiguity, underspecification and discourse interpretation. In *IWCS-1, First International Workshop on Computational Semantics*. Cited on page 119.

Pullum, G.K., 1986. Footloose and context-free. *Natural Language & Linguistic Theory*, 4 (3):409–414. doi:10.1007/BF00133376. Cited on page 63.

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, 4th International Conference on Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Computer Science*, pages 17–43. Springer. doi:10.1007/3-540-48199-0_2. Cited on page 47.

Pullum, G.K., 2007. The evolution of model-theoretic frameworks in linguistics. In *Model-Theoretic Syntax at 10*, pages 1–10. http://www.lel.ed.ac.uk/~gpullum/EvolutionOfMTS.pdf. Cited on page 11.

Rabin, M.O., 1969. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35. doi:10.2307/1995086. Cited on page 54.

Räihä, K.J. and Ukkonen, E., 1981. The shortest common supersequence problem over binary alphabet is NP-complete. *Theoretical Computer Science*, 16(2):187–198. doi:10.1016/0304-3975(81)90075-X. Cited on page 118.

Raney, G.N., 1958. Sequential functions. *Journal of the ACM*, 5(2):177–180. doi: 10.1145/320924.320930. Cited on page 20.

Reinhardt, K., 2002. The complexity of translating logic to finite automata. In Grädel, E., Thomas, W., and Wilke, T., editors, *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, chapter 13, pages 231–238. Springer. doi: 10.1007/3-540-36387-4_13. Cited on pages 48, 113.

Retoré, C., 2005. The logic of categorial grammars: Lecture notes. Technical Report RR-5703, INRIA. http://hal.inria.fr/inria-00070313/. Cited on pages 91, 97.

Robertson, N. and Seymour, P., 1986. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322. doi:10.1016/0196-6774(86)90023-4. Cited on page 110.

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 26, 27, 28.

Rogers, J., 1996. A model-theoretic framework for theories of syntax. In *ACL '96, 34th Annual Meeting of the Association for Computational Linguistics*, pages 10–16. ACL Press. doi:10.3115/981863.981865. Cited on page 54.

Rogers, J., 1998. *A Descriptive Approach to Language-Based Complexity*. Studies in Logic, Language, and Information. CSLI Publications. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.912&rep=rep1&type=pdf. Cited on page 51.

Rogers, J., 2003. wMSO theories as grammar formalisms. *Theoretical Computer Science*, 293(2):291–320. doi:10.1016/S0304-3975(01)00349-8. Cited on page 54.

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 40.

Rossman, B., 2008. Homomorphism preservation theorems. *Journal of the ACM*, 55(3): 15:1–15:53. doi:10.1145/1379759.1379763. Cited on page 115.

Rounds, W.C., 1970. Mappings and grammars on trees. 4(3):257–287. doi:10.1007/BF01695769. Cited on pages 68, 73.

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 19, 20, 21.

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 pages 17, 40.

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 67.

Schmidt-Schauß, M. and Smolka, G., 1991. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26. doi:10.1016/0004-3702(91)90078-X. Cited on page 102.

Schmitz, S., 2014. Implicational relevance logic is 2-ExpTime-complete. In Dowek, G., editor, *RTA-TLCA 2014, Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications*, volume 8560 of *Lecture Notes in Computer Science*, pages 395–409. Springer. doi:10.1007/978-3-319-08918-8_27. Cited on page 127.

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 pages 21, 79.

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 20.

Schwichtenberg, H., 1991. An upper bound for reduction sequences in the typed λ-calculus. *Archive for Mathematical Logic*, 30(5–6):405–408. doi:10.1007/BF01621476. Cited on page 127.

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 pages 11, 63.

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 73.

Sénizergues, G., 1996. On the termination problem for one-rule semi-Thue system. In Ganzinger, H., editor, *RTA '96, 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 302–316. Springer. doi:10.1007/3-540-61464-8_61. Cited on page 24.

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 63.

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 45.

Sima'an, K., 2002. Computational complexity of probabilistic disambiguation. 5(2): 125–151. doi:10.1023/A:1016340700671. Cited on page 87.

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 28.

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 22.

Statman, R., 1979a. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9(1):73–81. doi:10.1016/0304-3975(79)90007-0. Cited on page 127.

Statman, R., 1979b. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9(1):67–72. doi:10.1016/0304-3975(79)90006-9. Cited on page 127.

Steedman, M., 2000. *The Syntactic Process*. MIT Press. ISBN 0-262-69268-6. Cited on page 91.

Steedman, M., 2011. Romantics and revolutionaries. *Linguistic Issues in Language Technology*, 6. http://elanguage.net/journals/lilt/article/view/2587. Cited on page 10.

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 40.

ten Cate, B. and Segoufin, L., 2010. Transitive closure logic, nested tree walking automata, and XPath. *Journal of the ACM*, 57(3):18:1–18:41. doi:10.1145/1706591.1706598. Cited on pages 60, 61.

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 40.

Thatcher, J.W. and Wright, J.B., 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Theory of Computing Systems*, 2(1): 57–81. doi:10.1007/BF01691346. Cited on page 54.

Tomita, M., 1986. *Efficient Parsing for Natural Language*. Kluwer Academic Publishers. ISBN 0-89838-202-5. Cited on page 40.

Troelstra, A.S., 1992. *Lectures on Linear Logic*, volume 29 of *CSLI Lecture Notes*. CSLI Publications. http://standish.stanford.edu/bin/detail?fileID=1846861073. Cited on pages 13, 95.

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 40.

Vardi, M., 1998. Reasoning about the past with two-way automata. In Larsen, K.G., Skyum, S., and Winskel, G., editors, *ICALP '98, 25th International Colloquium on Automata, Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer. doi:10.1007/BFb0055090. Cited on pages 59, 111.

Weir, D.J., 1992. Linear context-free rewriting systems and deterministic tree-walking transducers. In *ACL '92, 30th Annual Meeting of the Association for Computational Linguistics*, pages 136–143. ACL Press. doi:10.3115/981967.981985. Cited on page 63.

Weyer, M., 2002. Decidability of S1S and S2S. In Grädel, E., Thomas, W., and Wilke, T., editors, *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*, chapter 12, pages 207–230. Springer. doi:10.1007/3-540-36387-4_12. Cited on page 54.

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 41.

Williams, V., 2012. Multiplying matrices faster than Coppersmith-Winograd. In *STOC 2012, 44th Symposium on Theory of Computing*, pages 887–898. ACM Press. doi:10.1145/2213977.2214056. Cited on page 40.

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 66.

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 40.

Zimmer, K., 1964. *Affixal negation in English and other languages*. William Clowes. Supplement to *Word* 20:2, monograph 5. Cited on page 22.

Zwicky, A.M., 1985. Clitics and particles. *Language*, 61(2):283–305. doi:10.2307/414146. Cited on page 16.

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, Thirteen Annual Meeting of the Berkeley Linguistics Society*, pages 330–340. http://www.ling.ed.ac.uk/~gpullum/bls_1987.pdf. Cited on page 17.