Logic and Linguistic Modelling

Sylvain Schmitz
LSV, ENS Cachan & CNRS & INRIA
November 13, 2014 (r5219M)

These notes cover the second part 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.

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 $R^{i+1} = R \mathbin{⨟} R^i$.

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.

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

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 $t_1, \ldots, t_n$ are terms in $T(\Sigma, \mathcal{X})$, then $f(t_1, \ldots, t_n)$ 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 $C[t_1, \ldots, t_n]$ for $t_1, \ldots, t_n$ 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 $C\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$. 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 $\square$ 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 $\xRightarrow{r} = \{(C[t\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}], C[t'\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}]) \mid C \in \mathcal{C}(\Sigma), t_1, \ldots, t_n \in T(\Sigma)\}$. We write $\xRightarrow{r_1 r_2}$ for $\xRightarrow{r_1} \mathbin{⨟} \xRightarrow{r_2}$, and $\xRightarrow{R}$ for $\bigcup_{r \in R} \xRightarrow{r}$.

Chapter 1

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.

1.0.1 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 (1.19) and (1.20)).

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).

1.0.2 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

$$\downarrow \stackrel{\text{def}}{=} \{(w, wi) \mid i \in \mathbb{N} \wedge wi \in W\} \tag{1.1}$$

$$\rightarrow \stackrel{\text{def}}{=} \{(wi, w(i+1)) \mid i \in \mathbb{N} \wedge w(i+1) \in W\}\ . \tag{1.2}$$

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

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 $\{0,1\}^* \rightarrow \Sigma$ with a prefix-closed domain. We define for this $\mathrm{dom}(t) = \mathrm{fcns}(W)$ and $t(w) = \{a \in A \mid P_a(\mathrm{fcns}^{-1}(w))\}$ for all $w \in \mathrm{dom}(t)$, where

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

$$\mathrm{fcns}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon \qquad \mathrm{fcns}(w0) \stackrel{\text{def}}{=} \mathrm{fcns}(w)0 \qquad \mathrm{fcns}(w(i+1)) \stackrel{\text{def}}{=} \mathrm{fcns}(wi)1 \tag{1.3}$$

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

$$\mathrm{fcns}^{-1}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon \quad \mathrm{fcns}^{-1}(w0) \stackrel{\text{def}}{=} \mathrm{fcns}^{-1}(w)0 \qquad \mathrm{fcns}^{-1}(w1) \stackrel{\text{def}}{=} \mathrm{fcns}^{-1}(w) + 1 \tag{1.4}$$

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 $\mathrm{dom}(t) \subseteq 0\{0,1\}^*$.

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

$$\downarrow_0 \stackrel{\text{def}}{=} \{(w, w0) \mid w0 \in \mathrm{dom}(t)\} \tag{1.5}$$

$$\downarrow_1 \stackrel{\text{def}}{=} \{(w, w1) \mid w1 \in \mathrm{dom}(t)\} \tag{1.6}$$

$$P_a \stackrel{\text{def}}{=} \{w \in \mathrm{dom}(t) \mid a \in t(w)\}\ . \tag{1.7}$$

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

$$\mathrm{dom}(\bar{t}) \stackrel{\text{def}}{=} \{\varepsilon, 1\} \cup \{0w \mid w \in \mathrm{dom}(t)\} \cup \{0wi \mid w \in \mathrm{dom}(t) \wedge i \in \{0,1\}\} \tag{1.8}$$

and labels

$$\bar{t}(w) \stackrel{\text{def}}{=} \begin{cases} t(w') & \text{if } w = 0w' \wedge w' \in \mathrm{dom}(t) \\ \{\#\} & \text{otherwise.} \end{cases} \tag{1.9}$$

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.

1.1 Monadic Second-Order Logic

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

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

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

where x, y range over $\mathcal{X}_1$, X over $\mathcal{X}_2$, and a over A. We write $\mathrm{FV}(\psi)$ 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

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

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

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':

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

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

Example 1.1. Here are a few useful wMSO formulæ: To allow any label in a finite set $B \subseteq A$:

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

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

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

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:

$$x\ [\mathrm{TC}_{u,v}\ \psi(u,v)]\ y \stackrel{\text{def}}{=} \forall X.(x \in X \wedge \forall uv.(u \in X \wedge \psi(u,v) \supset v \in X) \supset y \in X) \tag{1.10}$$

For example,

$$x \downarrow^\star y \stackrel{\text{def}}{=} x\ [\mathrm{TC}_{u,v}\ u \downarrow v]\ y$$
$$x \rightarrow^\star y \stackrel{\text{def}}{=} x\ [\mathrm{TC}_{u,v}\ u \rightarrow v]\ y\ .$$

1.1.1 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

$$\mathrm{labels}_{N,\Theta} \stackrel{\text{def}}{=} \forall x.\mathrm{root}(x) \supset P_S(x)\ , \tag{1.11}$$

which forces the root label to be S;

$$\wedge\ \forall x.\mathrm{internal}(x) \supset \bigvee_{a \in N \uplus \Theta} P_a(x) \wedge \bigwedge_{b \in N \uplus \Theta \setminus \{a\}} \neg P_b(x) \tag{1.12}$$

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

$$\wedge\ \forall x.\mathrm{leaf}(x) \supset \neg P_{N \uplus \Theta}(x) \tag{1.13}$$

forbids grammatical labels on leaves;

$$\wedge\ \forall y.\mathrm{leaf}(y) \supset \exists x.x \downarrow y \wedge P_\Theta(x) \tag{1.14}$$

expresses that leaves should have POS-labelled parents;

$$\wedge\ \forall x.\exists y_0 y_1 y_2.x \downarrow^\star y_0 \wedge y_0 \downarrow y_1 \wedge y_1 \downarrow y_2 \wedge \mathrm{leaf}(y_2) \supset P_N(x) \tag{1.15}$$

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 (1.12), and thus cannot have a leaf as a child by (1.13);

$$\wedge\ \forall x.P_\Theta(x) \supset \neg\exists yz.y \neq z \wedge x \downarrow y \wedge x \downarrow z \tag{1.16}$$

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

$$\mathrm{lexicon}_{L,\mathrm{pos}} \stackrel{\text{def}}{=} \forall x. \bigvee_{\ell \in L} \left(P_\ell(x) \supset \mathrm{leaf}(x) \wedge \bigwedge_{\ell' \in L \setminus \{\ell\}} \neg P_{\ell'}(x) \wedge \forall y.y \downarrow x \supset P_{\mathrm{pos}(\ell)}(y) \right) \tag{1.17}$$

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

$$\text{grammar}_{\mathcal{G}} \stackrel{\text{def}}{=} \forall x.(P_\varepsilon(x) \supset \neg P_{N \uplus \Theta \uplus L}(x)) \wedge \bigvee_{B \in N} P_B(x) \supset \bigvee_{B \to \beta \in P} \exists y.x \downarrow_0 y \wedge \text{rule}_\beta(y) \tag{1.18}$$

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

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

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

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

would result in the correct head annotations in Figure 1.1.

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

$$\text{head}(x) \stackrel{\text{def}}{=} \text{leaf}(x) \vee \bigvee_{B \in N} \exists y Y. y \downarrow x \wedge \text{children}(y, Y) \wedge P_B(y) \wedge \text{head}_{h(B)}(x, Y) \tag{1.19}$$

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

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

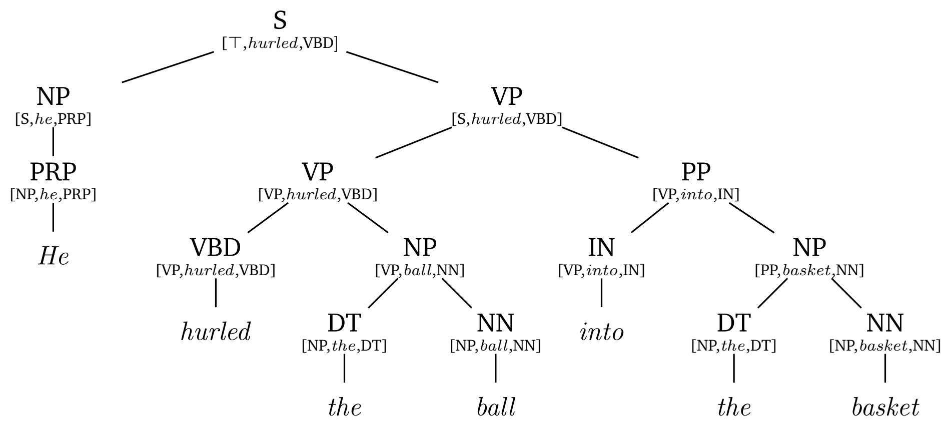


Figure 1.1: A derivation tree refined with lexical and parent information.

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

$$\text{lexicalise}(x, y) \stackrel{\text{def}}{=} \text{leaf}(y) \wedge x \, [\text{TC}_{u,v} \, u \downarrow v \wedge \text{head}(v)] \, y \,. \tag{1.20}$$

This formula recovers the lexical information in Figure 1.1.

Exercise 1.1. Propose wMSO formulæ to recover the parent and lexical POS information in constituent trees, as illustrated in Figure 1.1. **(∗)**

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

$$\begin{aligned}
\text{modify}_{\text{RB}}(x, y) \stackrel{\text{def}}{=} \exists x' y' z. z \downarrow x \wedge P_{\text{RB}}(z) \wedge \text{lexicalise}(x', x) \wedge y' \downarrow x' \\
\wedge \neg\text{lexicalise}(y', x) \wedge \text{lexicalise}(y', y)
\end{aligned} \tag{1.21}$$

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 1.2.

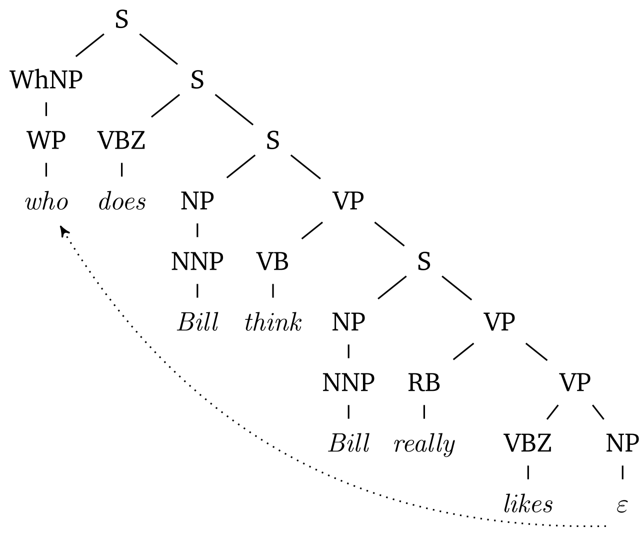


Figure 1.2: Derivation tree for *Who does Bill think Bill really likes?*

(∗) **Exercise 1.2.** Modify (1.21) to make sure that any leaf with a parent tagged by the POS RB modifies either a verb or an adjective.

(∗∗) **Exercise 1.3.** Consider the ε node in Figure 1.2: modify (1.20) to recover that *who* lexicalises the bottommost NP node.

1.1.2 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).*

$$L(\mathcal{A}_\psi) = \{\bar{t} \in T(\Sigma \uplus \{\{\#\}\}) \mid t \text{ finite} \wedge t \models \psi\}\ . \tag{1.22}$$

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

$$\psi' \stackrel{\text{def}}{=} \psi \wedge \exists x.\neg(\exists z.z \downarrow_0 x \vee z \downarrow_1 x) \wedge \neg(\exists y.x \downarrow_1 y) \tag{1.23}$$

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

$$x \downarrow y \stackrel{\text{def}}{=} \exists x_0.x \downarrow_0 x_0 \wedge (x_0\ [\mathrm{TC}_{u,v}\ u \downarrow_1 v]\ y) \tag{1.24}$$

$$x \rightarrow y \stackrel{\text{def}}{=} x \downarrow_1 y\ . \tag{1.25}$$

The conclusion of this construction is

Theorem 1.2. *Satisfiability and model-checking for wMSO are decidable.*

Exercise 1.4 (ω Successors)**.** Show that the weak second-order logic of ω successors (**wS$\boldsymbol{\omega}$S**), i.e. with $\downarrow_i \stackrel{\text{def}}{=} \{(w, wi) \mid wi \in W\}$ defined for every $i \in \mathbb{N}$, has decidable satisfiability and model-checking problems. (∗)

1.2 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

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).*

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

where a ranges over A.

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:

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

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

We define the classical dual operators

$$\bot \stackrel{\text{def}}{=} \neg\top \qquad \varphi_1\wedge\varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1\vee\neg\varphi_2) \qquad [\pi]\varphi \stackrel{\text{def}}{=} \neg\langle\pi\rangle\neg\varphi\;. \tag{1.26}$$

We also define

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

Exercise 1.5 (Converses)**.** Prove the following equivalences: (∗)

$$(\pi_1;\pi_2)^{-1} \equiv \pi_2^{-1};\pi_1^{-1} \tag{1.27}$$
$$(\pi_1+\pi_2)^{-1} \equiv \pi_1^{-1}+\pi_2^{-1} \tag{1.28}$$
$$(\pi^*)^{-1} \equiv (\pi^{-1})^* \tag{1.29}$$
$$(\varphi?)^{-1} \equiv \varphi?\;. \tag{1.30}$$

(∗) **Exercise 1.6** (Reductions)**.** Prove the following equivalences:

$$\langle \pi_1; \pi_2 \rangle \varphi \equiv \langle \pi_1 \rangle \langle \pi_2 \rangle \varphi \tag{1.31}$$
$$\langle \pi_1 + \pi_2 \rangle \varphi \equiv (\langle \pi_1 \rangle \varphi) \vee (\langle \pi_2 \rangle \varphi) \tag{1.32}$$
$$\langle \pi^* \rangle \varphi \equiv \varphi \vee \langle \pi; \pi^* \rangle \varphi \tag{1.33}$$
$$\langle \varphi_1 ? \rangle \varphi_2 \equiv \varphi_1 \wedge \varphi_2 \ . \tag{1.34}$$

1.2.1 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.

1.2.2 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 1.3 (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 1.1.2 we can *interpret* these relations in PDL by

$$\downarrow \stackrel{\text{def}}{=} \downarrow_0; \downarrow_1^* \qquad\qquad \rightarrow \stackrel{\text{def}}{=} \downarrow_1 \tag{1.35}$$

and translate any PDL formula φ into a formula

$$\varphi' \stackrel{\text{def}}{=} \varphi \wedge ([\uparrow^*; \downarrow^*; \downarrow_0] 0 \wedge \neg 1) \wedge ([\uparrow^*; \downarrow^*; \downarrow_1] 1 \wedge \neg 0) \wedge [\uparrow^*; \mathrm{root}?; \downarrow_1] \bot \tag{1.36}$$

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 (1.36) 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 1.5 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 (1.26), 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 1.4. *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 1.6 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 1.5. *Let φ be a PDL formula in normal form. Its Fisher-Ladner closure is of size $|\mathrm{FL}(\varphi)| \leq |\varphi|$.*

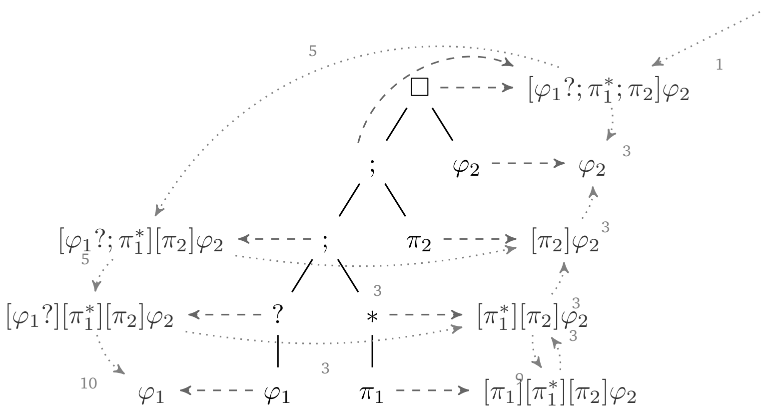


Figure 1.3: The surjection σ from positions in $\varphi \stackrel{\text{def}}{=} [\varphi_1 ?; \pi_1^*; \pi_2]\varphi_2$ to $\mathrm{FL}(\varphi)$ (dashed), and the rules used to construct $\mathrm{FL}(\varphi)$ (dotted).

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 1.3 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 1.6 and their duals, and by putting the formula into disjunctive normal form, i.e.

$$\varphi \equiv \bigvee_i \bigwedge_j \chi_{i,j} \tag{1.37}$$

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 1.6. *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 1.7. 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 $\mathcal{B}_+(\{-1, \ldots, k\} \times Q)$, defined by the abstract syntax

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

where d ranges over $\{-1, \ldots, k\}$ and q over Q. For a set $J \subseteq \{-1, \ldots, k\} \times Q$ and a formula f, we say that J *satisfies* f if assigning $\top$ to elements of J and $\bot$ to those in $\{-1, \ldots, k\} \times Q \backslash J$ 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 $\{1, \ldots, k\}$).

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 $J \subseteq \{-1, \ldots, k\} \times Q$ of form $J = \{(d_0, q_0), \ldots, (d_n, q_n)\}$ 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 1.8 (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 1.8. We assume wlog. that $A \subseteq \mathrm{Sub}(\varphi)$. We define

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

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

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

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 1.9. *Satisfiability of PDL can be decided in* EXPTIME.

Proof sketch. Given a PDL formula φ, by Lemma 1.4 construct an equivalent formula in normal form φ' with $|\varphi'| = O(|\varphi|)$. We then construct $\mathcal{A}_{\varphi'}$ with $O(|\varphi|)$ states by Lemma 1.5 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 1.8 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. ☐

1.2.3 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 $\mathrm{FO}[\mathrm{TC}^1]$ (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 $\mathrm{FO}[\mathrm{TC}^1]$. It is open whether $\mathrm{FO}[\mathrm{TC}^1]$ is strictly more expressive than PDL without this extension.

Exercise 1.7 (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 $\mathsf{W}\varphi$ 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 $\mathrm{FO}[\mathrm{TC}^1]$. **(∗∗)**

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

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 $\mathrm{FO}[\downarrow^\star, \rightarrow^\star]$ 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 1.8. Express the formulæ (1.12)–(1.21) in cPDL. **(∗)**

1.3 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 1.9.** 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 2

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.

2.1 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 2.1. For instance, assume our vocabulary includes $John^{(0)}$ as a constant denoting *John*, along with $apple^{(1)}$, $red^{(1)}$, and $eat^{(2)}$, we can associate the sentence

$$\exists x. apple^{(1)}(x) \wedge red^{(1)}(x) \wedge eat^{(2)}(John^{(0)}, x) \tag{2.1}$$

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

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

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: (2.1) 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.

2.1.1 Event Semantics

The kind of modelling that underlies Example 2.1 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 2.1, 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*:

$$eat_1^{(1)}(John^{(0)}) \tag{2.2}$$
$$\exists x.eat_2^{(2)}(John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \tag{2.3}$$
$$\exists xy.eat_3^{(3)}(John^{(0)}, x, y) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{2.4}$$
$$\exists y.eat_4^{(2)}(John^{(0)}, y) \wedge park^{(1)}(y) \tag{2.5}$$
$$\exists xy.eat_5^{(4)}(John^{(0)}, x, y, slowly^{(0)}) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{2.6}$$

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. (2.6) entails each of (2.2–2.5)? One possibility is to add explicit meaning postulates like

$$\forall jxy.eat_3^{(3)}(j, x, y) \supset eat_2^{(2)}(j, x) \tag{2.7}$$
$$\forall jx.eat_2^{(2)}(j, x, y) \supset eat_1^{(1)}(j) \tag{2.8}$$
$$\ldots \tag{2.9}$$

Similarly, we could treat *slowly* and the locative *in* as modal operators and rewrite (2.6) as

$$\exists xy.in^{(2)}(slowly^{(1)}(eat_2^{(2)}(John^{(0)}, x)), y) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{2.10}$$

along with the schemata

$$\forall Py.in^{(2)}(P, y) \supset P \tag{2.11}$$
$$\forall P.slowly^{(1)}(P) \supset P \tag{2.12}$$

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

$$\exists xy.slowly^{(1)}(location^{(2)}(eat_2^{(2)}(John^{(0)}, x), y)) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \tag{2.13}$$

instead, and proving the equivalence of (2.10) and (2.13) would require yet more machinery. (We will however return to modal operators later in Section 2.3.)

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:

$$eat_1^{(1)}(j) \stackrel{\text{def}}{=} \exists xya.eat_5^{(4)}(j,x,y,a) \tag{2.14}$$

$$eat_2^{(2)}(j,x) \stackrel{\text{def}}{=} \exists ya.eat_5^{(4)}(j,x,y,a) \tag{2.15}$$

$$eat_3^{(3)}(j,x,y) \stackrel{\text{def}}{=} \exists a.eat_5^{(4)}(j,x,y,a) \tag{2.16}$$

$$eat_4^{(2)}(j,y) \stackrel{\text{def}}{=} \exists ya.eat_5^{(4)}(j,x,y,a)\ . \tag{2.17}$$

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:

$$\exists e.eat_1^{(2)}(e, John^{(0)}) \tag{2.18}$$

$$\exists ex.eat_2^{(3)}(e, John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \tag{2.19}$$

$$\exists exy.eat_2^{(3)}(e, John^{(0)}, x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y) \tag{2.20}$$

$$\exists ey.eat_1^{(2)}(e, John^{(0)}) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y) \tag{2.21}$$

$$\exists exy.eat_2^{(3)}(e, John^{(0)}, x) \wedge red^{(1)}(x) \wedge apple^{(1)}(x) \wedge park^{(1)}(y) \wedge location^{(2)}(e,y) \wedge slowly^{(1)}(e) \tag{2.22}$$

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 (2.2) and (2.5): contrast for instance

See Davidson (1967).

> I sank the Bismark.
> I sank.

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

2.1.2 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).*

$$\exists e.eat^{(1)}(e) \wedge agent^{(2)}(e, John^{(0)}) \tag{2.23}$$

$$\exists ex.eat^{(1)}(e) \wedge agent^{(2)}(e, John^{(0)}) \wedge patient^{(2)}(e,x) \wedge apple^{(1)}(x) \tag{2.24}$$

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 2.1: A basic set of thematic roles.

the subject and the verb:

$$\exists e.sink^{(1)}(e) \wedge agent^{(2)}(e, I^{(0)}) \wedge patient^{(2)}(e, Bismark^{(0)}) \tag{2.25}$$

$$\exists e.sink^{(1)}(e) \wedge patient^{(2)}(e, I^{(0)}) \tag{2.26}$$

The definition of a fixed set of thematic roles and how to classify the different uses are of course problematic; Table 2.1 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.

2.2 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 2.3, 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.

2.2.1 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 $\bot \stackrel{\text{def}}{=} \neg\top$, $C \sqcup D \stackrel{\text{def}}{=} \neg(\neg C \sqcap \neg D)$, and $\forall r_j.C \stackrel{\text{def}}{=} \neg\exists r_j.\neg C$. A concept defines a subset $[\![C]\!]^{\mathfrak{M}}$ of a model $\mathfrak{M}$:

$$[\![\top]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} W \qquad [\![a]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} a$$

$$[\![C \sqcap D]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} [\![C]\!]^{\mathfrak{M}} \cap [\![D]\!]^{\mathfrak{M}} \qquad [\![\neg C]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} W \setminus [\![C]\!]^{\mathfrak{M}}$$

$$[\![\exists r.C]\!]^{\mathfrak{M}} \stackrel{\text{def}}{=} r^{-1}([\![C]\!]^{\mathfrak{M}})\ .$$

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

$$\forall y.man^{(1)}(y) \supset \exists x.woman^{(1)}(x) \wedge love^{(2)}(y,x) \tag{2.27}$$

It can also be formalised as a consistency question in $\mathcal{ALC}$:

$$\mathsf{Man} \sqsubseteq \exists \mathsf{love}.\mathsf{Woman} \tag{2.28}$$

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

$$\mathsf{Man} \sqcap \exists \mathsf{eat}.\mathsf{Apple} \tag{2.29}$$

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

$$\{John\} \sqsubseteq \exists \mathsf{eat}.(\mathsf{Apple} \sqcap \mathsf{Red}) \tag{2.30}$$

and the sentence *Helen of Troy is loved by every man in Greece* by

$$(\mathsf{Man} \sqcap \exists \mathsf{inhabit}.\{\mathit{Greece}\}) \sqsubseteq \exists \mathsf{love}.\{\mathit{Helen\ of\ Troy}\} \tag{2.31}$$

2.2.2 Translation into First-Order Logic

As hinted by the first-order and $\mathcal{ALC}$ formalisations in (2.27)–(2.28), 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 (2.31):

$$\forall y.(man^{(1)}(y) \wedge inhabit^{(2)}(y, \mathit{Greece}^{(0)})) \supset love^{(2)}(y, \mathit{Helen\ of\ Troy}^{(0)}) \tag{2.32}$$

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 2.4.

2.3 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.

2.3.1 *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:

$\mathfrak{M}, w \models \top$	always
$\mathfrak{M}, w \models p$	iff $w \in V(p)$
$\mathfrak{M}, w \models \neg\varphi$	iff $\mathfrak{M}, w \not\models \varphi$
$\mathfrak{M}, w \models \varphi \vee \varphi'$	iff $\mathfrak{M}, w \models \varphi$ or $\mathfrak{M}, w \models \varphi'$
$\mathfrak{M}, w \models \Diamond\varphi$	iff $\exists w', w\, R\, w'$ and $M, w' \models \varphi$.

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 2.1 (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 2.2 (Transitive Frames)**.** Show that, if R is transitive, then $\mathbf{4} : \Diamond\Diamond\varphi \supset \Diamond\varphi$ is valid. **(∗)**

Exercise 2.3 (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 1.2 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

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \vee \varphi \mid \triangle(\varphi_1, \dots, \varphi_{r(\triangle)})$$

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

$$\begin{aligned} \mathfrak{M}, w \models \triangle(\varphi_1, \dots, \varphi_{r(\triangle)}) \quad & \text{iff } \exists w_1, \dots, w_{r(\triangle)} \in W.(w, w_1, \dots, w_{r(\triangle)}) \in R_\triangle \\ & \text{and } \forall 1 \le i \le r(\triangle).\mathfrak{M}, w_i \models \varphi_i\ . \end{aligned}$$

Exercise 2.4 ($\mathcal{ALC}$ 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 $P_p = V(p)$:

$$\begin{aligned} \mathrm{ST}_x(p) &\stackrel{\text{def}}{=} P_p(x) \\ \mathrm{ST}_x(\top) &\stackrel{\text{def}}{=} (x = x) \\ \mathrm{ST}_x(\neg\varphi) &\stackrel{\text{def}}{=} \neg\mathrm{ST}_x(\varphi) \\ \mathrm{ST}_x(\varphi \vee \varphi') &\stackrel{\text{def}}{=} \mathrm{ST}_x(\varphi) \vee \mathrm{ST}_x(\varphi') \\ \mathrm{ST}_x(\triangle(\varphi_1, \dots, \varphi_{r(\triangle)})) &\stackrel{\text{def}}{=} \exists x_1 \dots x_{r(\triangle)}.R_\triangle(x, x_1, \dots, x_{r(\triangle)}) \wedge \bigwedge_{i=1}^{r(\triangle)} \mathrm{ST}_{x_i}(\varphi_i) \end{aligned}$$

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 2.2 (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. $w \mathrel{Z} w'$,

1. $\{p \in A \mid w \in V(p)\} = \{p' \in A \mid w' \in V'(p')\}$,
2. if $(w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle$, then there are $w'_1, \ldots, w'_{r(\triangle)}$ in W' s.t. $w_i \mathrel{Z} w'_i$ for all $1 \leq i \leq r(\triangle)$ and $(w', w'_1, \ldots, w'_{r(\triangle)}) \in R'_\triangle$, and
3. if $(w', w'_1, \ldots, w'_{r(\triangle)}) \in R'_\triangle$, then there are $w_1, \ldots, w_{r(\triangle)}$ in W s.t. $w_i \mathrel{Z} w'_i$ for all $1 \leq i \leq r(\triangle)$ and $(w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle$.

We say that w and w' are **bisimilar**, noted $w \mathrel{\underline{\leftrightarrow}} w'$, if there exists a bisimulation Z s.t. $w \mathrel{Z} w'$.

Proposition 2.3 (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 $w \mathrel{\underline{\leftrightarrow}} w'$, 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 2.2, the case where φ is $\top$ is trivial, and the cases of Boolean connectives follow from the induction hypothesis. For a formula of form $\triangle(\varphi_1, \ldots, \varphi_{r(\triangle)})$:

$$\mathfrak{M}, w \models \triangle(\varphi_1, \ldots, \varphi_{r(\triangle)})$$

implies $\exists w_1, \ldots, w_{r(\triangle)} \in W.(w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle \wedge \forall 1 \leq i \leq r(\triangle).\mathfrak{M}, w_i \models \varphi_i$

implies $\exists w'_1, \ldots, w'_{r(\triangle)} \in W'.(w', w'_1, \ldots, w'_{r(\triangle)}) \in R_\triangle \wedge \forall 1 \leq i \leq r(\triangle).\mathfrak{M}', w'_i \models \varphi_i$

(by ind. hyp. and (2))

implies $\mathfrak{M}', w' \models \triangle(\varphi_1, \ldots, \varphi_{r(\triangle)})$,

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

$$\{(w_1, \ldots, w_{r(\triangle)}) \mid (w, w_1, \ldots, w_{r(\triangle)}) \in R_\triangle\}$$

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 2.4 (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 $w \mathrel{\underline{\leftrightarrow}} w'$.*

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 2.5 (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).*

2.3.2 First-Order Modal Logic

In order to work with both modal operators and first-order semantics as in Section 2.1, 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 2.6 (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

$$\exists a.apple^{(1)}(a) \wedge F(eat_2^{(2)}(John^{(0)}, a))\ . \tag{2.33}$$

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

$$F(\exists a.apple^{(1)}(a) \wedge eat_2^{(2)}(John^{(0)}, a))\ . \tag{2.34}$$

2.4 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

$$\exists x.woman^{(1)}(x) \wedge \forall y.man^{(1)}(y) \supset love^{(2)}(y, x) \tag{2.35}$$

for *Every man loves a woman*—to be contrasted with (2.27)—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).

2.4.1 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 $\mathcal{X} \stackrel{\text{def}}{=} \{x_1, \ldots, x_k\}$ 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 (2.27) is in GFO^2: $man^{(1)}(y)$ guards the universal quantification and $love^{(2)}(y, x)$ guards the existential quantification. By contrast, (2.35) 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 2.3 can be extended to the case of guarded-k-bisimilarity:

Proposition 2.7. *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. $\square$

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 2.8. 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 $f_v(b_i) \stackrel{\text{def}}{=} b'_i$. Finally, let $\mathfrak{M}' \stackrel{\text{def}}{=} \bigcup_{u\in\operatorname{dom} t'} \mathfrak{T}'(u)$.

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

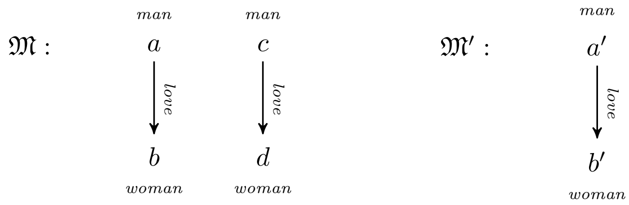


Figure 2.1: Two structures which are not guarded-2-bisimilar over the vocabulary $man^{(1)}, woman^{(1)}, love^{(2)}$.

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 2.7:

Proposition 2.8. *If a sentence ψ in GFO^k has a model, then it has a model of treewidth at most $k-1$.*

Proposition 2.8 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: (2.35) 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 2.1 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 2.7 $\mathfrak{M}$ 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 (2.35).

Another issue with the guarded fragment is that the axiom for transitivity of a binary relation R, which can be expressed by

$$\forall xyz.R^{(2)}(x,y) \wedge R^{(2)}(y,z) \supset R^{(2)}(x,z) \tag{2.36}$$

or by

$$\forall xy.R^{(2)}(x,y) \supset (\forall z.R^{(2)}(y,z) \supset R^{(2)}(x,z)) \tag{2.37}$$

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 3

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 3.2) or as possible semantic readings in ambiguous sentences (Section 3.3).

3.1 *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

$$\varphi_{\mathfrak{M}} \stackrel{\text{def}}{=} \exists x_1 \ldots x_n . \chi^+_{\mathfrak{M}}(x_1, \ldots, x_n) \tag{3.1}$$

where the formula $\chi^+_{\mathfrak{M}}$ is its **positive diagram** and consists of the conjunction of all the positive relational atomic formulæ true of $\mathfrak{M}$:

$$\chi^+_{\mathfrak{M}}(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \bigwedge_i \bigwedge_{(w_{i_1}, \ldots, w_{i_k}) \in R_i} R^{(i_k)}(x_{i_1}, \ldots, x_{i_k}) \ . \tag{3.2}$$

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

$$\alpha ::= x = y \mid R^{(k)}(x_1, \ldots, x_k) \qquad \text{(atomic formulæ)}$$

$$\varphi ::= \alpha \mid \neg\alpha \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x . \varphi \qquad \text{(existential formulæ)}$$

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

$$\varphi \equiv \bigvee_i \psi_i \tag{3.3}$$

where in turn each existential conjunctive formula ψ can be put in prenex form

$$\psi \equiv \exists \mathbf{x}. \bigwedge_j \beta_j(\mathbf{x}_j) \tag{3.4}$$

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:

$$\exists x_1 x_2 \ldots x_n.\chi \wedge x_1 = x_2 \equiv \exists x_2 \ldots x_n.\chi\{x_1 \leftarrow x_2\} \tag{3.5}$$

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:

$$\mathrm{Mod}_{\leq n}(\psi) \stackrel{\text{def}}{=} \{\mathfrak{M} = \langle W, \sigma\rangle \mid |W| \leq n \wedge \mathfrak{M} \models \psi\} . \tag{3.6}$$

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 3.1** (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 $\mathfrak{M}_{\psi^+} \models \psi$.

3.1.1 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 3.1. *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 (3.3): there exists a disjunct ψ such that $\mathfrak{M} \models \psi$. More precisely, ψ can be put in prenex normal form as $\psi \equiv \exists x_1 \ldots x_n.\chi$ where χ is a conjunction of atoms and negated atoms, and $\mathfrak{M} \models_\nu \chi$ for some valuation $\nu\colon \{x_1, \ldots, x_n\} \to W$. Consider the substructure $\mathfrak{M}_\nu$ (not necessarily in $\mathcal{C}$) induced by the subset $\nu(\{x_1, \ldots, x_n\}) \subseteq W$ 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 3.2. *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 $\psi \equiv \exists x_1 \ldots x_n.\chi$, $\mathfrak{M}$ 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 3.1 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

$$\mathfrak{M} \in \mathrm{Mod}_{\leq n}(\psi)\ . \tag{3.7}$$

(Note that this is not directly implied by Exercise 3.1, 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 3.2 (Diagrams). Let $\mathfrak{M} = \langle W, (R_i)_i \rangle$ be a finite structure with $W = \{w_1, \ldots, w_n\}$. We define its **diagram** as the conjunction of the atomic and negated atomic formulæ it satisfies under the valuation $\nu\colon x_j \mapsto w_j$: **(∗)**

$$\chi_{\mathfrak{M}} \stackrel{\text{def}}{=} \bigwedge_{1 \leq j \leq k \leq n} x_j \neq x_k \wedge \bigwedge_i \Big(\bigwedge_{(w_{i_1}, \ldots w_{i_k}) \in R_i} R_i^{(i_k)}(x_{i_1}, \ldots, x_{i_k}) \wedge \bigwedge_{(w_{i_1}, \ldots w_{i_k}) \notin R_i} \neg R_i^{(i_k)}(x_{i_1}, \ldots, x_{i_k}) \Big) \tag{3.8}$$

Show that, for any structure $\mathfrak{M}'$, $\mathfrak{M}' \models \exists x_1 \ldots x_n.\chi_{\mathfrak{M}}$ iff $\mathfrak{M} \subseteq_i \mathfrak{M}'$ (up to isomorphism).

Exercise 3.3 (Converse of Lemma 3.2). 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).

3.1.2 Tree Models

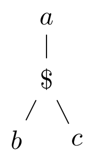
See Hidders (2004); Björklund et al. *(2011) for related results on tree patterns and XPath fragments.*

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 3.2 is incorrect for trees, e.g. the sentence

$$\exists xyz.P_a(x) \wedge P_b(y) \wedge P_c(z) \wedge x \downarrow^* y \wedge x \not\downarrow y \wedge x \downarrow^* z \wedge x \not\downarrow z \tag{3.9}$$

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 3.3 (Koller et al., 2001)**.** *Satisfiability of ECFO*$((\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a \in A})$ *sentences is* NP*-complete.*

3.2 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.

3.2.1 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.

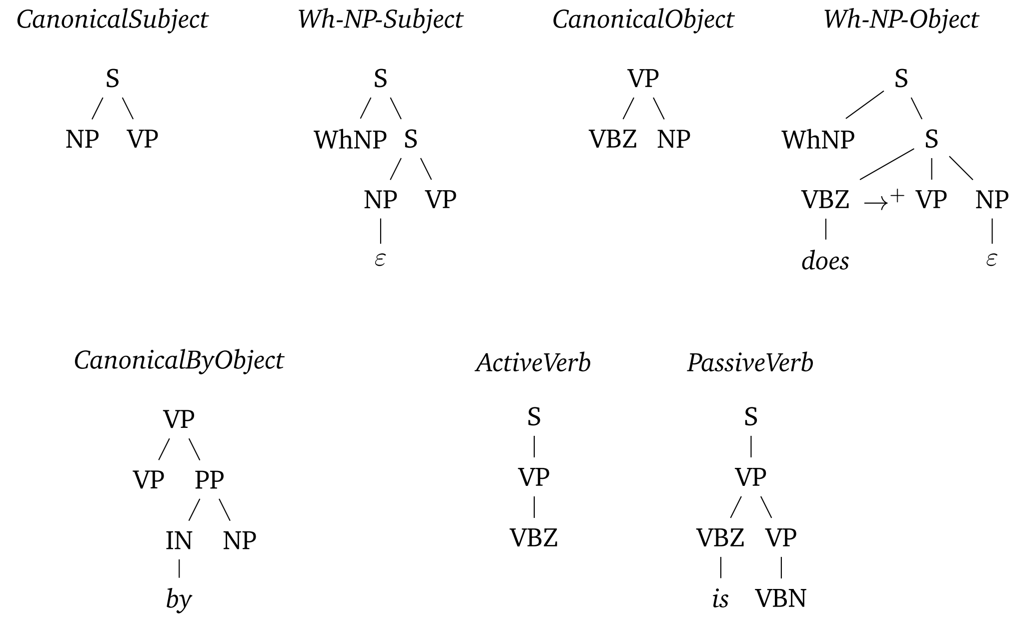


Figure 3.1: Basic tree fragments.

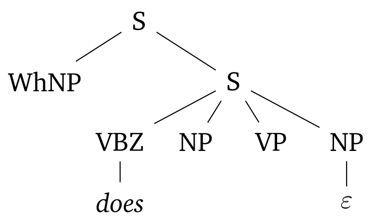


Figure 3.2: A minimal model of (3.10).

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):

$$\begin{aligned}
\textit{TransitiveVerb} &\stackrel{\text{def}}{=} \textit{ActiveTransitiveVerb} \lor \textit{PassiveTransitiveVerb}\\
\textit{ActiveTransitiveVerb} &\stackrel{\text{def}}{=} \textit{Subject} \land \textit{ActiveVerb} \land (\textit{CanonicalObject} \lor \textit{Wh-NP-Object})\\
\textit{PassiveTransitiveVerb} &\stackrel{\text{def}}{=} \textit{CanonicalSubject} \land \textit{PassiveVerb} \land \textit{CanonicalByObject}\\
\textit{Subject} &\stackrel{\text{def}}{=} \textit{CanonicalSubject} \lor \textit{Wh-NP-Subject}
\end{aligned}$$

where each of the basic formulæ *ActiveVerb*, *PassiveVerb*, etc. is the canonical positive primitive formula of the corresponding tree in Figure 3.1. For instance, the conjunction

$$\textit{CanonicalSubject} \land \textit{ActiveVerb} \land \textit{Wh-NP-Object} \tag{3.10}$$

gives rise to the unique minimal model of Figure 3.2.

3.2.2 Complexity

Observe that we only used the $\downarrow$, $\rightarrow$, and $\rightarrow^+$ axes in our examples in Section 3.2. 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 3.4. *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 $\#$:

$$\varphi \stackrel{\text{def}}{=} \exists z z'.P_\#(z) \wedge P_\#(z') \wedge \varphi_{=k}(z, z') \wedge \varphi_S(z, z')$$

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:

$$\varphi_{=k}(z, z') \stackrel{\text{def}}{=} \exists x_1 \ldots x_k.z \to x_1 \wedge x_k \to z' \wedge \Big(\bigwedge_{1 \leq j < k} x_j \to x_{j+1} \Big) \,.$$

On the other hand, $\varphi_S(z, z')$ should ensure that the segment between z and z' is indeed a supersequence of every $s_i = a_{i_1} \cdots a_{i_{\ell_i}}$:

$$\varphi_S(z, z') \stackrel{\text{def}}{=} \bigwedge_{1 \leq i \leq p} \exists y_1 \ldots y_{\ell_i}.z \to^+ y_1 \wedge y_{\ell_i} \to^+ z' \wedge \Big(\bigwedge_{1 \leq j \leq \ell_i} P_{a_{i_j}}(y_j) \Big) \\ \wedge \Big(\bigwedge_{1 \leq r < \ell_i} y_r \to^+ y_{j+1} \Big) \,. \qquad \square$$

3.3 Underspecified Semantics

3.3.1 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

$$\exists y.woman(y) \wedge \forall x.man(x) \supset \exists e.love(e) \wedge agent(e, x) \wedge patient(e, y) \tag{3.11}$$

$$\forall x.man(x) \supset \exists y.woman(y) \wedge \exists e.love(e) \wedge agent(e, x) \wedge patient(e, y) \tag{3.12}$$

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 (3.11) implies (3.12)), 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.

3.3.2 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 $l : a^{(r)}(x_1, \ldots, x_r)$ 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).

$$\gamma ::= l : a^{(r)}(x_1, \ldots, x_r) \mid h \downarrow^* l \mid \gamma \wedge \gamma \mid \exists x.\gamma \qquad \text{(hole formulæ)}$$

where l ranges over $\mathcal{L}$, $a^{(r)}$ over A_r, $x, x_1, \ldots, x_r$ 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

$$\gamma \equiv \exists l_1 \ldots l_n h_1 \ldots h_m. \bigwedge_p \gamma_p \,. \tag{3.13}$$

Hole formulæ γ are *interpreted* in ECFO$((\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a\in A})$ by associating a formula $[\gamma]$

$$[\gamma] = \exists l_1 \dots l_n h_1 \dots h_m. \bigwedge_{1\leq i<j\leq n} l_i \neq l_j \wedge \bigwedge_p \gamma_p \tag{3.14}$$

where we interpret

$$l : a^{(r)}(x_1, \dots, x_r) \stackrel{\text{def}}{=} P_a(l) \wedge \bigwedge_{i=1}^{r} l \downarrow_{i-1} x_i \,. \tag{3.15}$$

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 3.5 (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 (3.13). 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

$$\nu(x) \stackrel{\text{def}}{=} \begin{cases} \hat{x} & \text{if } x \in \mathcal{L} \\ \widehat{P(x)} & \text{if } x \in \mathcal{H}\,. \end{cases} \tag{3.16}$$

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 3.6. Let us extend the syntax of hole formulæ by allowing larger tree segments:

$$\gamma ::= l : a^{(r)}(\theta_1, \dots, \theta_r) \mid h \downarrow^* l \mid \gamma \wedge \gamma \mid \exists x.\gamma \qquad \text{(hole formulæ)}$$
$$\theta ::= a^{(r)}(\theta_1, \dots, \theta_r) \mid h \qquad \text{(tree formulæ)}$$

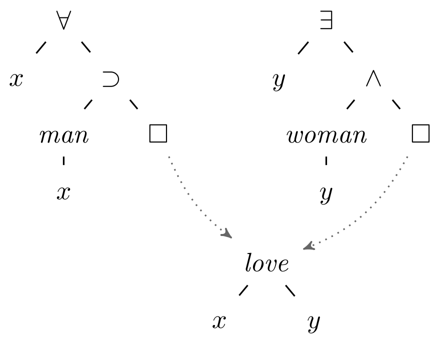


Figure 3.3: Underspecified formula for (3.11) and (3.12). Dominance relations are indicated through dotted arrows and holes by boxes.

and translating back into hole formulæ by defining

$$x_\theta \stackrel{\text{def}}{=} \begin{cases} h & \text{if } \theta = h \\ l_\theta \in \mathcal{L} & \text{a fresh label for each } \theta \text{ otherwise} \end{cases}$$

$$l : a^{(r)}(\theta_1, \ldots, \theta_r) \stackrel{\text{def}}{=} l : a^{(r)}(x_{\theta_1}, \ldots, x_{\theta_r})$$

$$a^{(r)}(\theta_1, \ldots, \theta_r) \stackrel{\text{def}}{=} \exists l_\theta . l_\theta : a^{(r)}(x_{\theta_1}, \ldots, x_{\theta_r}) \ .$$

A hole semantic formula that models the two readings (3.11) and (3.12) is the following (see also Figure 3.3):

$$\exists l_1 l_2 l_3 h_1 h_2 . l_1 : \forall^{(2)}(x^{(0)}, \mathit{man}^{(1)}(x^{(0)}) \supset^{(2)} h_1) \wedge l_2 : \exists^{(2)}(y^{(0)}, \mathit{woman}^{(1)}(y^{(0)}) \wedge^{(2)} h_2)$$
$$\wedge l_3 : \mathit{love}^{(2)}(x^{(0)}, y^{(0)}) \wedge h_1 \downarrow^* l_3 \wedge h_2 \downarrow^* l_3 \ .$$

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 3.7. *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
$$\mathfrak{M} = \langle \{\hat{l}_1, \ldots, \hat{l}_n\}, (\downarrow_i)_{i<k}, (P_a)_{a \in A} \rangle \ ; \tag{3.17}$$
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
$$\mathfrak{M}' = \langle \{\hat{l}_1, \ldots, \hat{l}_n\}, (\downarrow_i)_{i<k}, \downarrow^*, (P_a)_{a \in A} \rangle \tag{3.18}$$
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 (3.16) (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 $\{\$^{(1)}, f_i^{(a_i+1)}, g^{(m)}, b^{(0)} \mid 1 \leq i \leq 3m\}$:

$$\begin{aligned}
&\exists l_\$ l_{f_1} \dots l_{f_{3m}} l_g l_b^{1,1} \dots l_b^{1,B+1} l_b^{2,1} \dots l_b^{m,B+1} h_\$ h_g^1 \dots h_g^m h_{f_1}^1 \dots h_{f_1}^{a_1+1} h_{f_2}^1 \dots h_{f_{3m}}^{a_{3m}+1}.\\
&\quad \$^{(1)}(h_\$) \wedge \bigwedge_{1\leq i\leq 3m} l_{f_i} : f_i^{(a_i+1)}(h_{f_1}^1, \dots, h_{f_1}^{a_i+1})\\
&\quad \wedge l_g : g^{(m)}(h_g^1, \dots, h_g^m) \wedge \bigwedge_{1\leq j\leq m} \bigwedge_{1\leq k\leq B+1} l_b^{j,k} : b^{(0)}\\
&\quad \wedge \bigwedge_{1\leq i\leq 3m} h_\$ \downarrow^* l_{f_i} \wedge h_\$ \downarrow^* l_g \wedge \bigwedge_{1\leq j\leq m} \bigwedge_{1\leq k\leq B+1} h_g^j \downarrow^* l_b^{j,k}\ .
\end{aligned}$$

Assume first that there exists a partition $A_1 \uplus \dots \uplus A_m$ 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 $h_{f_x}^1, \dots, h_{f_x}^{a_x}, h_{f_y}^1, \dots, h_{f_y}^{a_y}, h_{f_z}^1, \dots, h_{f_z}^{a_z+1}$ 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. $A_j \stackrel{\text{def}}{=} \{a_i \mid \mathfrak{M} \models_\nu h_g^j \downarrow^* l_{f_i}\}$. Note that $A_1 \uplus \dots \uplus A_m$ 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 3.4** (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 $2^{\{l_1,\dots,l_n\}} \times \{l_1, \dots, l_n\} \times 2^{\{h_1,\dots,h_m\}}$ as state set, although there certainly are better ways; see for instance Koller et al. *(2008).*

The size of the automaton constructed in Exercise 3.4 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

$$A_n \stackrel{\text{def}}{=} \{a^{(0)}, g_1^{(1)}, \dots, g_n^{(1)}\} \tag{3.19}$$

$$\gamma_n \stackrel{\text{def}}{=} \exists l l_1 \dots l_n h_1 \dots h_n . l : a^{(0)} \wedge \bigwedge_{i=1}^{n} l_i : g_i^{(1)}(h_i) \wedge h_i \downarrow^* l\ . \tag{3.20}$$

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

$$L_n = \{g_{\pi(1)}(\Box) \cdot g_{\pi(2)}(\Box) \cdots g_{\pi(n)}(a) \mid \pi \text{ a permutation of } \{1,\ldots,n\}\}\ . \quad (3.21)$$

Lemma 3.8. *Any finite tree automaton for L_n requires at least 2^n states.*

Proof. Define for every subset $K = \{i_1, \ldots, i_{|K|}\}$ of $\{1, \ldots, n\}$ (where $i_j < i_{j+1}$) the context

$$C_K \stackrel{\text{def}}{=} g_{i_1}(\Box) \cdots g_{i_{|K|}}(\Box) \quad (3.22)$$

and let $\bar{K} = \{1, \ldots, n\} \backslash K$. Then the tree

$$t_K \stackrel{\text{def}}{=} C_{\bar{K}} \cdot C_K \cdot a \quad (3.23)$$

is in L_n.

Let Q_K be the set of states q of an automaton $\mathcal{A}_n$ for L_n s.t.

$$C_{\bar{K}} \cdot C_K \cdot a \Rightarrow^\star C_{\bar{K}} \cdot q \Rightarrow^\star q_f \quad (3.24)$$

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

$$C_{\bar{K}'} \cdot C_K \cdot a \Rightarrow^\star C_{\bar{K}'} \cdot q \Rightarrow^\star q_f \quad (3.25)$$

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. $\Box$

Note that the tree automaton $\langle 2^{\{1,\ldots,n\}}, A, \delta, \{\emptyset\}\rangle$ with $\delta = \{(q \backslash \{i\}, g_i, q) \mid i \in q\} \cup \{(\{1, \ldots, n\}, b)\}$ recognizes L_n, so this bound is optimal.

Lemma 3.8 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 4.1.4 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 4

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 4.1 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.

4.1 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 2.1: 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 $[\![\textit{John}]\!] = John^{(0)}$ and $[\![\textit{a red apple}]\!] = \exists x.apple^{(1)}(x) \wedge red^{(1)}(x)$, it is not so clear how one should combine everything and obtain (2.1) or more involved representations like (2.24). Moreover any solution will be dependent on the specific syntactic analysis.

4.1.1 *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 4.3), 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 **λ*I*-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 4.1 (**B** combinator)**.** Define $\mathbf{B} \stackrel{\text{def}}{=} \lambda xyz.x(yz)$. 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 4.2 (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 4.3 (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 4.1. Prove Proposition 4.2 and Proposition 4.3. **($*$)**

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 4.4 (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 4.5 (Statman, 1979b)**.** *Simple type inhabitation is* PSPACE*-complete.*

4.1.2 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 4.2** (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 $\tau(c) = a_1 \to \cdots \to a_n \to a_0$, $0 \le i \le n$, and $\vdash_\Sigma t_j : a_j$ for all $0 < j \le i$, 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 $0 \le i \le n$, 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 $0 < j \le i$ 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**

$$\mathscr{G}(\Sigma, s) \stackrel{\text{def}}{=} \{\bar{t} \in T(C) \mid \vdash_\Sigma t : s \text{ where } t \text{ is ground}\}\ .$$

Example 4.6. 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 4.3** (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 $L = \mathscr{G}(\Sigma, s)$.

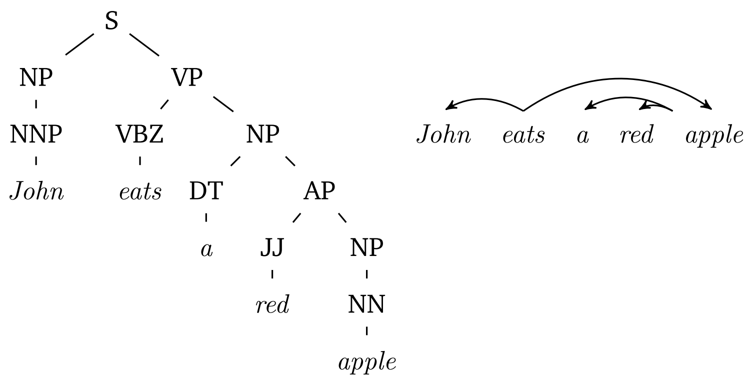


Figure 4.1: Constituent and dependency analyses for *John eats a red apple*.

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.

4.1.3 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 4.7 (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 4.8. Continuing with Example 2.1, Figure 4.1 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 4.1, we have

$$C = \{\mathit{John}^{(0)}, \mathit{apple}^{(0)}, \dots, \mathrm{AP}^{(2)}, \mathrm{NP}^{(2)}, \mathrm{JJ}^{(1)}, \dots, \mathrm{S}^{(2)}\}$$

and

$$C' = \{\mathit{John}^{(0)}, \wedge^{(2)}, \exists^{(2)}, \dots\}\ .$$

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 (2.24).

(∗) **Exercise 4.4.** Propose similarly a higher-order homomorphism from the dependency structure of Figure 4.1 into its semantics.

4.1.4 Tree Transductions

The definition we provided for higher-order homomorphisms does not use types explicitly; this is easy to remedy:

Definition 4.9 (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 4.10 (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 $\Sigma_{\mathcal{F}} \stackrel{\text{def}}{=} \langle\{o\}, \mathcal{F}, \tau_{\mathcal{F}}\rangle$ where for every $f^{(n)}$ in $\mathcal{F}$, $\tau_{\mathcal{F}}(f^{(n)}) \stackrel{\text{def}}{=} \underbrace{o \to \cdots \to o}_{n \text{ times}} \to o = o^n \to o$.

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 $\mathcal{T}: T(C) \to T(C')$ by

$$\mathcal{T}(\bar{t}_1) = \bar{t}_2 \text{ iff } \vdash_\Sigma t_1 : s \wedge [\![t_1]\!] \Rightarrow^\star_{\beta\eta} t_2 \ . \tag{4.1}$$

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 4.8 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.

Definition 4.11 (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

See de Groote (2001).

$$\mathscr{A}(\mathcal{G}) \stackrel{\text{def}}{=} \{L \in \Lambda_\ell(C) \mid \vdash_\Sigma L : s\}$$

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 $\mathscr{A}(\mathcal{G})$:

$$\Downarrow_{\beta\eta}\mathscr{O}(\mathcal{G}) = \Downarrow_{\beta\eta}\{[\![t]\!] \in \Lambda_\ell(C') \mid t \text{ ground} \in \mathscr{A}(\mathcal{G})\} \ . \tag{4.2}$$

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 4.5 (Tree Languages of $\text{ACG}_{2,1}$). 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 **(**)**

$$\mathscr{T}(\mathcal{G}) \stackrel{\text{def}}{=} \{\bar{t} \in T(\mathcal{F}) \mid t \text{ ground} \in \mathscr{O}(\mathcal{G})\} \ . \tag{4.3}$$

Assume that $\max_{a\in A} \text{ord}([\![a]\!]) = 1$. 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 4.3. In that situation, we will no longer work with ground object terms.

4.2 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 $[\lambda x x'.\varphi](f, f')$ 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 4.12 (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 $[\lambda x x'.x = x'](f, g)$ 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 $K[\lambda x x'.x = x'](f, g)$ 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 $[\lambda x x'.K(x = x')](f, g)$ 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 4.13 (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 $P[\lambda x x'.x = x'](f, g)$ 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: $[\lambda x x'.P(x = x')](f, g)$ 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

$$\forall x.\exists f.D(f, x) \tag{4.4}$$

to hold. This is however a reasonable restriction; let us check for instance the following equivalence under the hypothesis of (4.4):

$$\exists x.\varphi \equiv \exists f.[\lambda x.\varphi](f)\ . \tag{4.5}$$

Indeed, for all $\mathfrak{M}$, w, ν and φ,

$$\begin{aligned}
&\mathfrak{M}, w \models_\nu \exists f.[\lambda x.\varphi](f) \\
&\text{iff } \exists i \in D_{\mathcal{I}}.\mathfrak{M}, w \models_{\nu[f\leftarrow i]} [\lambda x.\varphi](f) \\
&\text{iff } \exists i \in D_{\mathcal{I}}.\mathfrak{M}, w \models_{\nu[f\leftarrow i, x\leftarrow i(w)]} \varphi \\
&\text{iff } \exists e \in D_{\mathcal{O}}.\mathfrak{M}, w \models_{\nu[x\leftarrow e]} \varphi && \text{(by (4.4) when choosing } i(w) = e\text{)} \\
&\text{iff } \mathfrak{M}, w \models_\nu \exists x.\varphi\ .
\end{aligned}$$

($*$) **Exercise 4.6.** Show the following equivalence when (4.4) holds:

$$\exists f.\Diamond[\lambda x.\varphi](f) \equiv \Diamond(\exists x.\varphi)\ . \tag{4.6}$$

Example 4.14 (Unicorn)**.** The sentence *John finds a unicorn* could be associated with the semantics

$$\exists ex.\mathit{find}^{(1)}(e) \wedge \mathit{agent}^{(2)}(e, \mathit{John}^{(0)}) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x) \tag{4.7}$$

but it is better to treat *unicorn* as an intension in the formula

$$\exists u.[\lambda x.\exists e.\mathit{find}^{(1)}(e) \wedge \mathit{agent}^{(2)}(e, \mathit{John}^{(0)}) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u)\ , \tag{4.8}$$

equivalent to (4.7) in totally intensional models according to (4.5). Then we better see the connection with the sentence *John seeks a unicorn*: its de dicto semantics would be

$$\begin{aligned}
&\exists u.\textsc{try}(\mathit{John}^{(0)}, [\lambda x.\exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u)) && (4.9) \\
&\equiv \textsc{try}(\mathit{John}^{(0)}, \exists ex.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)) && \text{(by (4.6))}
\end{aligned}$$

and its de re semantics

$$\begin{aligned}
&\exists u.[\lambda x.\textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)](u) && (4.10) \\
&\equiv \exists x.\textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x) \wedge \mathit{unicorn}^{(1)}(x)) && \text{(by (4.5))}
\end{aligned}$$

and if the interpretation of $\mathit{unicorn}^{(1)}$ is the same in all worlds accessible through the TRY modality,

$$\equiv \exists x.\mathit{unicorn}^{(1)}(x) \wedge \textsc{try}(\mathit{John}^{(0)}, \exists e.\mathit{find}^{(1)}(e) \wedge \mathit{patient}^{(2)}(e, x))\ .$$

4.3 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 4.1 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.

4.3.1 *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 4.1.1. 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 4.3.2 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).

4.3.2 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 4.1: Some constants and their possible types.

$$\begin{aligned}
[\![\text{walk}]\!] &= walk_{\iota\to s\to o}\\
[\![\text{eat}_2]\!] &= eat_{2\,\iota\to\iota\to s\to o}\\
[\![\text{apple}]\!] &= apple_{\iota\to s\to o}\\
[\![\text{red}]\!] &= \lambda P_{\iota\to s\to o} x_\iota w_s . red_{\iota\to s\to o}\, x\, w \wedge P\, x\, w\\
[\![\text{every}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \forall_\iota x . (P\, x\, w \supset P'\, x\, w)\\
[\![\text{a}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \exists_\iota x . (P\, x\, w \wedge P'\, x\, w)\\
[\![\text{no}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \forall_\iota x . (P\, x\, w \supset \neg P'\, x\, w)\\
[\![\text{the}]\!] &= \lambda P_{\iota\to s\to o} P'_{\iota\to s\to o} w_s . \exists_\iota x . (P'\, x\, w \wedge \forall_\iota y . (P\, x\, w \equiv x = y))\\
[\![\text{John}]\!] &= John_\iota\\
[\![\text{necessarily}]\!] &= \lambda p_{s\to o} w_s . \forall_s w' . (R_{s\to s\to o}\, w\, w') \supset p\, w'\\
[\![\text{possibly}]\!] &= \lambda p_{s\to o} w_s . \exists_s w' . (R_{s\to s\to o}\, w\, w') \wedge p\, w'\\
[\![\text{know}]\!] &= \lambda p_{s\to o} x_\iota w_s . \forall_s w' . (K_{\iota\to s\to s\to o}\, x\, w\, w') \supset p\, w'\\
[\![\text{believe}]\!] &= \lambda p_{s\to o} x_\iota w_s . \forall_s w' . (B_{\iota\to s\to s\to o}\, x\, w\, w') \supset p\, w'\\
[\![\text{not}]\!] &= \lambda p_{s\to o} w_s . \neg\, p\, w
\end{aligned}$$

Table 4.2: 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 2.1.2.

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

$$walks\ John\ . \tag{4.11}$$

*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 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 4.8), we arrive at the types of Table 4.1. The semantics of a sentence can then be computed by a higher-order homomorphism as in Section 4.1, 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 4.2 for some examples of semantic values.

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 5

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 6, 13.

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 41, 42, 44.

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 31.

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 37.

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 36.

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 24.

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 33.

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 38, 39.

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 13.

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 13.

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 14, 26, 28, 29.

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 41, 51.

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 20.

Börger, E., Grädel, E., and Gurevich, Y., 1997. *The Classical Decision Problem*. Perspectives Mathematical Logic. Springer. Cited on page 30.

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 41.

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 16.

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 48, 56, 57.

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 10.

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 3, 7, 12.

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 41.

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 38.

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 23.

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 37.

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 51, 53.

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 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 5.

Ebbinghaus, H.D. and Flum, J., 1999. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer. Cited on page 36.

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 41.

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 54.

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 54.

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 45.

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 13, 14.

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 54, 55.

Gallin, D., 1975. *Intensional and Higher-Order Modal Logic*, volume 19 of *Mathematic Studies*. Elsevier. ISBN 0-444-11002-X. Cited on page 57.

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 34.

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 30.

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 31, 33.

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 31.

Harel, D., Kozen, D., and Tiuryn, J., 2000. *Dynamic Logic*. Foundations of Computing. MIT Press. Cited on page 14.

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 57.

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 38.

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 48, 49.

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 41.

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 47.

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 page 21.

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 51.

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 54.

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 9.

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 38.

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 43.

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 44.

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 13.

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 27.

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 14.

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 30.

Marx, M., 2005. Conditional XPath. *ACM Transactions on Database Systems*, 30(4):929–959. doi:10.1145/1114244.1114247. Cited on pages 13, 19.

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 13.

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 9.

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 6, 12.

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 34.

Montague, R., 1970. Universal grammar. *Theoria*, 36(3):373–398. doi:10.1111/j.1755-2567.1970.tb00434.x. Cited on page 48.

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 48, 51.

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 58.

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 57.

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 29.

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 13.

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 23.

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 47.

Poesio, M., 1994. Ambiguity, underspecification and discourse interpretation. In *IWCS-1, First International Workshop on Computational Semantics*. Cited on page 41.

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 5.

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 12.

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 40.

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 6, 35.

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 32.

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 12.

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 9.

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 12.

Rossman, B., 2008. Homomorphism preservation theorems. *Journal of the ACM*, 55(3): 15:1–15:53. doi:10.1145/1379759.1379763. Cited on page 37.

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 24.

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 49.

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 49.

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 49.

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 49.

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 18, 19.

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 12.

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 17, 33.

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 12.