Contents

6 Multi-component grammars **2**
6.1 Pair-wrapping grammars . . . 2
6.1.1 Strings with a gap . . . 2
6.1.2 Definition by deduction . . . 3
6.1.3 Definition by rewriting . . . 4
6.1.4 Definition by language equations . . . 4
6.1.5 Equivalence of the three definitions . . . 4
6.1.6 Examples . . . 5
6.1.7 Pumping lemma . . . 5
6.1.8 Normal forms . . . 6
6.1.9 Closure properties . . . 6

7 First-order theory of grammars **7**
7.1 First-order logic over positions in the input . . . 7
7.1.1 Definition by equations . . . 9
7.1.2 Definition by logical derivation . . . 10
7.1.3 Equivalence of the two definitions . . . 11
7.4 Decision procedure . . . 11
7.4.1 Basic algorithm . . . 11

Bibliography **13**

Name index **14**

Chapter 6

Multi-component grammars

6.1 Pair-wrapping grammars

Joshi, Levy and Takahashi [3] introduced *tree-adjoining grammars*, defined by tree rewriting. Later, Pollard [4] defined a model called a "head grammar", which was proved equivalent to tree-adjoining grammars in the papers by Vijay-Shanker and Joshi [8] and by Weir, Vijay-Shanker and Joshi. The first definition of this model involved heavy notation; its essense was distilled in an improved definition by Rounds [6, Sect. 5], which is adopted in this text with minor adjustments. The model is given a new name, *pair-wrapping grammars*, which better explains its meaning.

6.1.1 Strings with a gap

Strings with a gap, or simply *pairs:* $u : v$ with $u, v \in \Sigma^*$. For any two languages $K, L \subseteq \Sigma^*$, denote by $K : L$ the set of strings with a gap $\{\, u : v \mid u \in K,\ v \in L \}$. The set of all strings with a gap: $\Sigma^* : \Sigma^*$. *Wrapping operation* on strings with a gap: $(u : v)(x : y) = ux : yv$. This operation is used instead of the concatenation, with $\varepsilon : \varepsilon$ as an identity. Another operation of *flattening a string with a gap,* that is, filling the gap with the empty string and pairing this concatenation with the empty string: $(u : v) : \varepsilon = uv : \varepsilon$ and symmetrically, $\varepsilon : (u : v) = \varepsilon : uv$.

Definition 6.1. *A pair-wrapping grammar is a quadruple* (Σ, N, R, S)*, where the rules in* R *are of the form*

$$A \to X_1 \dots X_\ell,$$

for $\ell \geqslant 0$ *and* $X_i \in N \cup \{\, Y : \varepsilon \mid Y \in \Sigma \cup N \} \cup \{\, \varepsilon : Z \mid Z \in \Sigma \cup N \}$*. An empty sequence* $X_1 \dots X_\ell$ *with* $\ell = 0$ *is denoted by* $\varepsilon : \varepsilon$*.*

In the following, a more general syntax for rules in pair-wrapping grammars shall be adopted: a rule of the form $A \to X_1 \dots X_\ell$ may have any X_i either of the form $B \in N$, of the form $Y_1 \dots Y_m : Z_1 \dots Z_n$, where $m, n \geqslant 0$ and $Y_j, Z_j \in \Sigma \cup N$. Such a symbol X_i shall be regarded as an abbreviation for flattening all Y_j and Z_j and wrapping the resulting pairs as follows.

$$Y_1 \dots Y_m : Z_1 \dots Z_n = (Y_1 : \varepsilon) \dots (Y_m : \varepsilon)(\varepsilon : Z_n) \dots (\varepsilon : Z_1).$$

Each symbol $A \in N$ defines a language $L_G(A) \subseteq \Sigma^* \times \Sigma^*$ of strings with a gap. Then the language of standard strings generated by the grammar is defined by filling the gap in all elements of $L_G(S)$ with the empty string: $L(G) = \{\, uv \mid u : v \in L_G(S) \}$.

A standard string has two sides; a string with a gap has four, and a pair-wrapping grammar

can independently append substrings from each of the four sides.

$$\begin{aligned}(a:\varepsilon)\cdot(u:v) &= au:v\\ (u:v)\cdot(b:\varepsilon) &= ub:v\\ (u:v)\cdot(\varepsilon:c) &= u:cv\\ (\varepsilon:d)\cdot(u:v) &= u:vd\end{aligned}$$

Used in the following example.

Example 6.1. *Consider the pair-wrapping grammar* $G = (\{a,b,c,d\},\{S\},R,S)$*, where* R *contains the following two rules*

$$S \to (a:d)S(b:c) \mid \varepsilon:\varepsilon$$

The symbol S *defines the set* $\{\, a^n b^n : c^n d^n \mid n \geqslant 0\}$*, and accordingly, the grammar defines the string language* $\{\, a^n b^n c^n d^n \mid n \geqslant 0\}$*.*

6.1.2 Definition by deduction

Definition by deduction of items $A(u:v)$. Axioms: $u:v(u:v)$. Filling the gap: $A(u:v) \vdash A:\varepsilon(uv:\varepsilon)$, $A(u:v) \vdash \varepsilon:A(\varepsilon:uv)$. Grammar rule: $X_1(u_1:v_1),\ldots,X_n(u_n:v_n) \vdash A(u_1\ldots u_n:v_n\ldots v_1)$.

Definition 6.1(D). *For a pair-wrapping grammar* $G = (\Sigma,N,R,S)$*, consider elementary propositions of the form "a string with a gap* $u:v$ *has a property* X*", with* $w \in \Sigma^*$ *and* $X \in N \cup \{\, Y:\varepsilon \mid Y \in \Sigma\cup N\}\cup\{\,\varepsilon:Z \mid Z\in\Sigma\cup N\}$*, denoted by* $X(u:v)$*. The deduction system uses the following axioms:*

$$\begin{aligned}&\vdash a:\varepsilon(a:\varepsilon) && \textit{(for all } a\in\Sigma\textit{)},\\ &\vdash \varepsilon:a(\varepsilon:a) && \textit{(for all } a\in\Sigma\textit{)}.\end{aligned}$$

Each rule $A \to X_1\ldots X_\ell$ *is regarded as the following schema for deduction rules:*

$$X_1(u_1:v_1),\ldots,X_\ell(u_\ell:v_\ell) \vdash A(u_1\ldots u_\ell:v_\ell\ldots v_1) \qquad \textit{for all } u_1,v_1,\ldots,u_\ell,v_\ell\in\Sigma^*.$$

Furthermore, there are the following flattening rules:

$$\begin{aligned}&A(u:v)\vdash A:\varepsilon(uv:\varepsilon) && \textit{(for all } A\in N \textit{ and } u,v\in\Sigma^*\textit{)},\\ &A(u:v)\vdash \varepsilon:A(\varepsilon:uv) && \textit{(for all } A\in N \textit{ and } u,v\in\Sigma^*\textit{)}.\end{aligned}$$

Whenever an item $X(u:v)$ *can be deduced from the above axioms by the given deduction rules, this is denoted by* $\vdash X(u:v)$*. Define* $L_G(X) = \{\, u:v \mid \vdash X(u:v)\}$ *and* $L(G) = \{\, uv \mid \vdash S(u:v)\}$*.*

Example 6.1(D). *For the grammar in Example 6.1.*

$$\begin{aligned}\vdash S(\varepsilon:\varepsilon) &&& (S\to\varepsilon:\varepsilon)\\ \vdash a:d(a:d) &&& (axiom)\\ \vdash b:c(b:c) &&& (axiom)\\ a:d(a:d),S(\varepsilon:\varepsilon),b:c(b:c)\vdash S(ab:cd) &&& (S\to(a:d)S(b:c))\\ a:d(a:d),S(ab:cd),b:c(b:c)\vdash S(aabb:ccdd) &&& (S\to(a:d)S(b:c))\end{aligned}$$

6.1.3 Definition by rewriting

Definition 6.1(R). *For a pair-wrapping grammar $G = (\Sigma, N, R, S)$, consider terms over the operations of* wrapping *XY (an associative binary operator) and* pair composition *$X : Y$ (a non-associative binary operator).*

The relation $\Longrightarrow$ of one-step rewriting on the set of terms is defined as follows:

- *Using a rule $A \to X_1 \dots X_\ell \in R$, any atomic subterm A of any term can be rewritten by the subterm $X_1 \dots X_\ell$:*
$$\dots A \dots \Longrightarrow \dots X_1 \dots X_\ell \dots$$
- *A subterm of the form $(u : v) : w$ may be rewritten with $uv : w$.*
- *A subterm of the form $w : (u : v)$ may be rewritten with $w : uv$.*

Define $L_G(X) = \{ u : v \mid S \Longrightarrow^* u : v \}$ and $L(G) = \{ uv \mid u : v \in L_G(S) \}$.

Example 6.1(R). *For the grammar in Example 6.1.*

$S \Longrightarrow (a : d)S(b : c) \Longrightarrow (a : d)(a : d)S(b : c)(b : c) = (aa : dd)S(bb : cc) \Longrightarrow (aa : dd)(\varepsilon : \varepsilon)(bb : cc)$
$= aabb : ccdd$

6.1.4 Definition by language equations

Unknown languages of strings with gaps, $L \subseteq \Sigma^* \times \Sigma^*$. Their wrapping: $K \cdot L = \{ uu' : v'v \mid u : v \in K,\ u' : v' \in L \}$. Filling the gap: $K : L = \{ uv : u'v' \mid u : v \in K,\ u' : v' \in L \}$. These operations are monotone and continuous.

Definition 6.1(E). *System corresponding to a grammar:*

$$A = \bigcup_{A \to X_1 \dots X_\ell \in R} \bigcap_{i=1}^{m} X_1 \cdot \ldots \cdot X_\ell \quad (\textit{for all } A \in N)$$

Each $X_i \in \Sigma : \varepsilon \cup \varepsilon : \Sigma$ in the equation represents a constant language $\{a : \varepsilon\}$ or $\{\varepsilon : a\}$, and a rule $A \to \varepsilon : \varepsilon$ is represented by a constant $\{\varepsilon : \varepsilon\}$. Let $(\dots, L_A, \dots)_{A \in N}$ with $L_A \subseteq \Sigma^* : \Sigma^*$ be the least solution of this system. Then $L_G(A)$ is defined as L_A for each $A \in N$.

Example 6.1(D). *For the grammar in Example 6.1, the corresponding language equation is*

$$S = \{a : d\} \cdot S \cdot \{b : c\} \cup \{\varepsilon : \varepsilon\}$$

6.1.5 Equivalence of the three definitions

To see that Definitions 6.1(R), 6.1(D) and 6.1(E) are equivalent.

Theorem 6.1. *Let $G = (\Sigma, N, R, S)$ be a pair-wrapping grammar, as in Definition 6.1. For every $X \in \Sigma : \varepsilon \cup \varepsilon : \Sigma \cup N$ and $u, v \in \Sigma^*$, the following three statements are equivalent:*

(R). $X \Longrightarrow^* u : v$,

(D). $\vdash X(u : v)$,

(E). $u : v \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot) \right]_X$.

Proof. $\boxed{\text{(R)} \Rightarrow \text{(D)}}$ Induction on the number of steps in the rewriting of X to $u : v$.

Basis: $X \Longrightarrow^* u : v$ in zero steps. This means that $X = u : v = a : \varepsilon$ or $X = u : v = \varepsilon : a$, and in both cases $\vdash X(u : v)$ is an axiom.

Induction step. Let $X \Longrightarrow^k w$ with $k \geqslant 1$. Then $X = A \in N$ and the rewriting begins by applying a rule $A \to X_1 \dots X_\ell$. Each X_i belongs to $N \cup \{ Y : \varepsilon \mid Y \in \Sigma \cup N \} \cup \{ \varepsilon : Z \mid Z \in \Sigma \cup N \}$, and is rewritten to a string with a gap $u_i : v_i \in \Sigma^*$ in less than k steps, where $u = u_1 \dots u_\ell$ and $v = v_\ell \dots v_1$. By the induction hypothesis, $\vdash X_i(u_i : v_i)$ for each i. and the desired item $A(u : v)$ can be deduced as

$$X_1(u_1 : v_1), \dots, X_\ell(u_\ell : v_\ell) \vdash A(u : v) \qquad \text{by the rule } A \to X_1 \dots X_\ell.$$

other cases TBW □

6.1.6 Examples

Example 6.2. *The following pair-wrapping grammar generates the language* $\{ ww \mid w \in \{a, b\}^* \}$*:*

$$S \to (a : \varepsilon) S (\varepsilon : a) \mid (b : \varepsilon) S (\varepsilon : b) \mid \varepsilon : \varepsilon$$

The language of strings with gaps defined by the symbol S *is* $\{ w : w \mid w \in \{a, b\}^* \}$*.*

6.1.7 Pumping lemma

Pumping from all four sides of a string with a gap.

Lemma 6.1 (Pumping lemma for pair-wrapping grammars)**.** *For every pair-wrapping language* $L \subseteq \Sigma^*$ *there exists a constant* $p \geqslant 1$*, such that for every string* $w \in L$ *with* $|w| \geqslant p$ *there exists a partition* $w = x_0\, u_1 x_1 u_2\, x_2\, u_3 x_3 u_4\, x_4$*, where* $|u_1| + |u_2| + |u_3| + |u_4| > 0$ *and* $|u_1 x_1 u_2| + |u_3 x_3 u_4| \leqslant p$*, such that* $x_0\, u_1^i x_1 u_2^i\, x_2\, u_3^i x_3 u_4^i\, x_4 \in L$ *for all* $i \geqslant 0$*.*

Example 6.3. *The language* $\{ a^n b^n c^n d^n e^n \mid n \geqslant 0 \}$ *is not defined by any pair-wrapping grammar.*

Proof. Suppose it is defined by one. Let p be the constant given by the pumping lemma, and consider the string $w = a^p b^p c^p d^p e^p$. The pumping lemma gives a partition $w = x_0\, u_1 x_1 u_2\, x_2\, u_3 x_3 u_4\, x_4$. If any of u_1, u_2, u_3, u_4 spans over any border between the blocks (that is, if it contains symbols of different types), then a single pumping produces a string not in L.

Therefore, each u_i is a substring of a^p, b^p, c^p, d^p or e^p. Since there are five blocks in w and only four strings u_i in the partition. at least one of the blocks is disjoint with u_1, u_2, u_3, u_4, and is not subject to pumping. On the other hand, since $|u_1 u_2 u_3 u_4| > 0$, at least one of the blocks contains some symbols from u_1, u_2, u_3 or u_4, and therefore the pumped string $w' = x_0\, u_1^2 x_1 u_2^i\, x_2\, u_3^2 x_3 u_4^i\, x_4$ contains more than p symbols in that block, but only p symbols in the block that is not pumped. Accordingly, w' is not in L, which contradicts the assertion of the pumping lemma. □

Example 6.4. *The language* $\{ www \mid w \in \{a, b\}^* \}$ *is not defined by any pair-wrapping grammar.*

Kanazawa and Salvati proved that the language $\{ w \mid w \in \{a, b, c\}^*,\ |w|_a = |w|_b = |w|_c \}$ is not a pair-wrapping language.

Example 6.5 (Radzinski [5])**.** *The language* $\{ ab^{k_1} ab^{k_2} \dots ab^{k_n} \mid k_1 > k_2 > \dots > k_n > 0 \}$ *is not defined by any pair-wrapping grammar.*

6.1.8 Normal forms

Normal form, with all rules of the form

$$A \to BC,$$
$$A \to B : \varepsilon,$$
$$A \to \varepsilon : C,$$
$$A \to a : \varepsilon,$$
$$A \to \varepsilon : a$$

where $B, C \in N$ and $a \in \Sigma^*$.

Theorem 6.2. *For every pair-wrapping grammar there exists and can be effectively constructed a pair-wrapping grammar in the normal form generating the same language.*

6.1.9 Closure properties

Obviously closed under union and concatenation.

Theorem 6.3 (Vijay-Shanker and Joshi [8])**.** *Not closed under intersection and complementation.*

Proof. The language $\{ a^n b^n c^n d^n e^n \mid n \geqslant 0 \}$ is an intersection of two linear ordinary languages, as well as a complement of a linear ordinary language. □

Closed under homomorphisms and inverse homomorphisms.

Exercises

6.1.1. Construct a pair-wrapping grammar for the language $\{ a^n b^n c^n \mid n \geqslant 0 \}$.

6.1.2. Construct a pair-wrapping grammar for the language $\{ wxw \mid w \in \{a, b\}^+, \ x \in \{a, b\}^* \}$.

6.1.3. Let $D \subseteq \{a, b\}^*$ be the Dyck language. Construct a pair-wrapping grammar for the language $\{ wc^{|w|} \mid w \in D \} = \{\varepsilon, abcc, aabbcccc, ababcccc, aaabbbcccccc, \ldots\}$.

Chapter 7

First-order theory of grammars

7.1 First-order logic over positions in the input

First-order logic with least fixpoint was first defined by Chandra and Harel [1] as a programming language for expressing queries to relational databases. It was further studied by Immerman [2] and by Vardi [7]. It was applied in the context of formal grammars by Rounds [6].

Intuitive interpretation of a grammar as a logical formula. Each category symbol A is regarded as a binary predicate $A(x, y)$, where the variables x and y range over positions in a string. Positions in a string $w = a_1 \dots a_n$, are numbered from 0 to n. A pair of positions (i, j) refers to a substring $a_{i+1} \dots a_j$.

Example 7.1. *Consider the following ordinary grammar for the Dyck language.*

$$S \to SS \mid aSb \mid \varepsilon$$

It is expressed as the following definition of a predicate $S(x, y)$ by an iterated first-order formula: a substring from position x to position y belongs to the Dyck language if and only if one of the following three conditions holds.

1. *The substring from x to y is a concatenation of two substrings from the Dyck language, one from x to some position z, and the other from z to y:*

$$(\exists z)(S(x, z) \wedge S(z, y))$$

2. *The position pointed by x (to be exact, this is position $x + 1$) contains a symbol a, the position pointed by y contains b, and the substring between positions $x+1$ and $y-1$ belongs to the Dyck language.*

$$a(x + 1) \wedge S(x + 1, y - 1) \wedge y(j)$$

3. *This is an empty substring, which begins and ends in the same position.*

$$x = y$$

Now, $S(x, y)$ can be defined as a disjunction of these three conditions, which formally transcribe the logic within the given formal grammar describing this language.

The membership of a string $w \in \{a, b\}^*$ in the Dyck language is then expressed by the statement $S(0, |w|)$, which may be true or false.

Terms are arithmetical expressions that evaluate to positions.

Definition 7.1. *Consider any set of variables. The set of terms over these variables is defined inductively as follows.*

- *any variable is an term;*
- *the symbols* $\underline{\text{begin}}$ *and* $\underline{\text{end}}$ *are constant terms referring to the first and the last positions in the string;*
- *if* t *is a term, then so are* $t+1$ *and* $t-1$ *(increment and decrement operations).*

A predicate has finitely many arguments (positions), and is true or false for any given assignment of positions to its arguments. Pre-defined predicates for reading symbols of the input string: $a(x)$, with $a \in \Sigma$, asserts that the symbol in position x is a. Pre-defined arithmetic predicates for comparing positions: $x < y$ and $x = y$. Formulae are constructed from predicates using conjunction, disjunction and first-order quantification.

Definition 7.2. *Let* Σ *be an alphabet, let* N *be a finite set of predicate symbols, with each* $A \in N$ *having a finite rank, denoted by* $\operatorname{rank} A$.

- *if* $A \in N$ *is a predicate symbol with* $\operatorname{rank} A = k$ *and* $t_1, \ldots, t_k$ *are terms, then* $A(t_1, \ldots, t_k)$ *is a formula (basic predicate);*
- *if* $a \in \Sigma$ *is a symbol of the alphabet and* t *is an term, then* $\varphi = a(t)$ *is a formula;*
- *if* t *and* t' *are terms, then* $t < t'$ *and* $t = t'$ *are formulae (arithmetical predicates);*
- *if* φ *and* ψ *are formulae, then so are* $\varphi \vee \psi$ *and* $\varphi \wedge \psi$*;*
- *if* φ *is a formula and* x *is a free variable in* φ*, then* $(\exists x)\varphi$ *and* $(\forall x)\varphi$ *are formulae as well.*

(note: one could allow more sophisticated terms and arithmetical predicates)
Recursive definition: each predicate $A(x_1, \ldots, x_n)$ is defined by a formula $\varphi_A(x_1, \ldots, x_n)$.

Definition 7.3. *A first-order grammar is a quintuple* $G = (\Sigma, N, \operatorname{rank}, \langle \varphi_A \rangle_{A \in N}, \sigma)$*, where*

- Σ *is a finite alphabet,*
- N *is a finite set of predicate symbols,*
- $\operatorname{rank} \colon N \to \mathbb{N}$ *defines the rank of each predicate symbol,*
- *each* φ_A *is a formula with* $\operatorname{rank} A$ *free variables, which defines the predicate* A*, and*
- σ *is a formula with no free variables, which defines the condition of being a syntactically well-formed sentence.*

Such a grammar can be written down as a collection of definitions $A(x_1, \ldots, x_{\operatorname{rank} A}) = \varphi_A(x_1, \ldots, x_{\operatorname{rank} A})$*, with* σ *specified separately.*

The iterative first-order definition of the Dyck language given in Example 7.1 is presented in the notation of first-order grammars as follows.

Example 7.2. *Consider the first-order grammar* $G = (\Sigma, \{S\}, \operatorname{rank}, \langle \varphi_S \rangle, \sigma)$*, where* $\operatorname{rank} S = 2$*, and the predicate* S *is defined as follows.*

$$S(x,y) = \underbrace{\big[(\exists z)(S(x,z) \wedge S(z,y))\big] \vee (a(x+1) \wedge S(x+1, y-1) \wedge b(y)) \vee x = y}_{\varphi_S}$$

The condition of being a well-formed sentence is stated in the formula $\sigma = S(\underline{\text{begin}}, \underline{\text{end}})$*. This grammar defines the Dyck language.*

7.1.1 Definition by equations

The value of a term: evaluates to a position.

Definition 7.1(E). *Let $w \in \Sigma^*$ be a string. Let t be a term using variables $x_1, \ldots, x_n$, and let $x_1 = i_1, \ldots, x_n = i_n$ be an assignment of values to variables. Then the value $t(i_1, \ldots, i_n)$ of the term is defined inductively on its structure as follows:*

- *if $t = x_j$, then $t(i_1, \ldots, i_n) = i_j$;*
- *if $t = \underline{\text{begin}}$, then $t(i_1, \ldots, i_n) = 0$, and if $t = \underline{\text{end}}$, then $t(i_1, \ldots, i_n) = |w|$;*
- *for any term t, define $(t-1)(i_1, \ldots, i_n) = t(i_1, \ldots, i_n) - 1$ if $t(i_1, \ldots, i_n) > 0$; otherwise, the value of $t - 1$ is undefined; similarly, define $(t+1)(i_1, \ldots, i_n) = t(i_1, \ldots, i_n) + 1$, as long as the result is at most $|w|$.*

The truth value of a formula.

Definition 7.2(E). *Let Σ be an alphabet, let N be a finite set of predicate symbols, each of a certain rank. Given a string $w \in \Sigma^*$, consider the set of all elementary statements on the range of positions in this string.*

$$\widehat{I_w} = \{\, A(i_1, \ldots, i_k) \mid A \in N,\ k = \operatorname{rank} A,\ i_1, \ldots, i_k \in \{0, \ldots, |w|\}\}$$

Any subset $I \subseteq \widehat{I_w}$ of statements assumed to be true is called an interpretation. Then, for any formula $\varphi(x_1, \ldots, x_n)$ with n free variables, and for any arguments $i_1, \ldots, i_n \in \{0, \ldots, |w|\}$, the statement $\varphi(i_1, \ldots, i_n)$ is said to be true on the string $w \in \Sigma^*$ under the interpretation I, denoted by $I \models_w \varphi(i_1, \ldots, i_n)$, if it satisfies the following inductive definition.

- *If $\varphi(x_1, \ldots, x_n) = A(t_1, \ldots, t_k)$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if $A(t_1(i_1, \ldots, i_n), \ldots, t_k(i_1, \ldots, i_n)) \in I$.*
- *If $\varphi(x_1, \ldots, x_n) = a(t)$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if the symbol in the position $t(i_1, \ldots, t_n)$ of the string w is a.*
- *If $\varphi(x_1, \ldots, x_n) = (t < t')$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if the number $t(i_1, \ldots, i_n)$ is less than $t'(i_1, \ldots, i_n)$.*
- *Similarly, if $\varphi(x_1, \ldots, x_n) = (t = t')$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if the numbers $t(i_1, \ldots, i_n)$ and $t'(i_1, \ldots, i_n)$ are equal.*
- *If $\varphi(x_1, \ldots, x_n) = \psi \vee \xi$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if $I \models_w \psi(i_1, \ldots, i_n)$ or $I \models_w \xi(i_1, \ldots, i_n)$.*
- *Similarly, if $\varphi(x_1, \ldots, x_n) = \psi \wedge \xi$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if $I \models_w \psi(i_1, \ldots, i_n)$ and $I \models_w \xi(i_1, \ldots, i_n)$.*
- *If $\varphi(x_1, \ldots, x_n) = (\exists x)\psi(x_1, \ldots, x_n, x)$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if there exists such a number $i \in \{0, \ldots, |w|\}$, that $I \models_w \psi(i_1, \ldots, i_n, i)$.*
- *Similarly, if $\varphi(x_1, \ldots, x_n) = (\forall x)\psi(x_1, \ldots, x_n, x)$, then $I \models_w \varphi(i_1, \ldots, i_n)$ if for every number $i \in \{0, \ldots, |w|\}$, $I \models_w \psi(i_1, \ldots, i_n, i)$.*

Definition 7.3(E). *Let $G = (\Sigma, N, \operatorname{rank}, \langle \varphi_A \rangle_{A \in N}, \sigma)$ be a first-order grammar. Let $w \in \Sigma^*$ be a string. An interpretation $I \subseteq \widehat{I_w}$ is called a model, if, for every $A \in N$, $A(i_1, \ldots, i_{\operatorname{rank} A}) \in I$ if and only if $I \models_w \varphi_A(i_1, \ldots, i_{\operatorname{rank} A})$. A statement $\psi(x_1, \ldots, x_k)$ is said to be true on a string w, denoted by $\models_w \psi(i_1, \ldots, i_k)$, if $I \models_w \psi(i_1, \ldots, i_k)$ for every model I. Then the language generated by the grammar is $L(G) = \{\, w \in \Sigma^* \mid \models_w \sigma\}$.*

There is always a least model.

Lemma 7.1. *Let $G = (\Sigma, N, \mathrm{rank}, \langle \varphi_A \rangle_{A \in N}, \sigma)$ be a first-order grammar. For every string $w \in \Sigma^*$, define an operator $\Phi \colon 2^{\widehat{P_w}} \to 2^{\widehat{P_w}}$ mapping an interpretation to an interpretation as*

$$\Phi(I) = \{\, A(i_1, \ldots, i_{\mathrm{rank}\, A}) \mid I \models_w \varphi_A(i_1, \ldots, i_{\mathrm{rank}\, A}) \}.$$

The operator Φ is monotone, in the sense that $I \subseteq I'$ implies $\Phi(I) \subseteq \Phi(I')$. Define

$$I_w = \bigcup_{\ell \geqslant 0} \Phi^\ell(\varnothing).$$

Then I_w is a model. Furthermore, I_w is the least model, in the sense that every model I is a superset of I_w.

7.1.2 Definition by logical derivation

Definition 7.3(D). *Let $G = (\Sigma, N, \mathrm{rank}, \langle \varphi_A \rangle_{A \in N}, \sigma)$ be a first-order grammar. For every string $w \in \Sigma^*$, let $\varphi(x_1, \ldots, x_n)$ be a formula, where $x_1, \ldots, x_n$ are its free variables. Then, for every substitution $(x_1, \ldots, x_n) = (i_1, \ldots, i_n)$, there are the following derivation rules.*

$$\begin{aligned}
&\vdash_w a(t(i_1, \ldots, i_n)) && \text{(if the symbol in position } t(i_1, \ldots, i_n) \text{ in } w \text{ is } a\text{)} \\
&\vdash_w (t < t')(i_1, \ldots, i_n) && \text{(if } t(i_1, \ldots, i_n) \text{ is less than } t'(i_1, \ldots, i_n)\text{)} \\
&\vdash_w (t = t')(i_1, \ldots, i_n) && \text{(if } t(i_1, \ldots, i_n) \text{ is equal to } t'(i_1, \ldots, i_n)\text{)} \\
\varphi_A(i_1, \ldots, i_{\mathrm{rank}\, A}) &\vdash_w A(i_1, \ldots, i_{\mathrm{rank}\, A}) \\
\varphi, \psi &\vdash_w \varphi \wedge \psi \\
\varphi &\vdash_w \varphi \vee \psi \\
\varphi(i_1, \ldots, i_k, i) &\vdash_w (\exists x) \varphi(i_1, \ldots, i_k, x) \\
\big\{ \varphi(i_1, \ldots, i_k, i) \big\}_{i \in \{0, \ldots, |w|\}} &\vdash_w (\forall x) \varphi(i_1, \ldots, i_k, x)
\end{aligned}$$

Then the language generated by the grammar is $L(G) = \{\, w \in \Sigma^* \mid \vdash_w \sigma \}$.

Example 7.2(D). *Consider the first-order grammar $G = (\Sigma, \{S\}, \mathrm{rank}, \langle \varphi_S \rangle, \sigma)$ for the Dyck language given in Example 7.2. Let $w = abaabb$ be a string.*

$$\vdash_w 1 = 1 \qquad \textit{(axiom)}$$
$$1 = 1 \vdash_w \big[(\exists z)(S(1,z) \wedge S(z,1))\big] \vee \big(a(2) \wedge S(2,0) \wedge b(1)\big) \vee 1 = 1 \qquad \textit{(disjunction)}$$
$$\big[(\exists z)(S(1,z) \wedge S(z,1))\big] \vee \big(a(2) \wedge S(2,0) \wedge b(1)\big) \vee 1 = 1 \vdash_w S(1,1) \qquad \textit{(definition of } S\textit{)}$$
$$\vdash_w a(1) \qquad \textit{(input)}$$
$$\vdash_w b(2) \qquad \textit{(input)}$$
$$a(1), S(1,1), b(2) \vdash_w a(1) \wedge S(1,1) \wedge b(2) \qquad \textit{(conjunction)}$$
$$a(1) \wedge S(1,1) \wedge b(2) \vdash_w \big[(\exists z)(S(0,z) \wedge S(z,2))\big] \vee \big(a(1) \wedge S(1,1) \wedge b(2)\big) \vee 0 = 2 \qquad \textit{(disjunction)}$$
$$\big[(\exists z)(S(0,z) \wedge S(z,2))\big] \vee \big(a(1) \wedge S(1,1) \wedge b(2)\big) \vee 0 = 2 \vdash_w S(0,2) \qquad \textit{(definition of } S\textit{)}$$
$$\vdots$$
$$\ldots \vdash_v S(3,5) \qquad \textit{(similarly)}$$
$$a(3) \wedge S(3,5) \wedge b(6) \vdash_w \big[(\exists z)(S(2,z) \wedge S(z,6))\big] \vee \big(a(3) \wedge S(3,5) \wedge b(6)\big) \vee 2 = 6 \qquad \textit{(disjunction)}$$
$$\big[(\exists z)(S(2,z) \wedge S(z,6))\big] \vee \big(a(3) \wedge S(3,5) \wedge b(6)\big) \vee 2 = 6 \vdash_w S(2,6) \qquad \textit{(definition of } S\textit{)}$$
$$S(0,2)S(2,6) \vdash_w (S(0,2) \wedge S(2,6)) \qquad \textit{(conjunction)}$$
$$(S(0,2) \wedge S(2,6)) \vdash_w (\exists z)(S(0,z) \wedge S(z,6)) \qquad \textit{(quantification)}$$
$$(\exists z)(S(0,z) \wedge S(z,6)) \vdash_w \big[(\exists z)(S(0,z) \wedge S(z,6))\big] \vee \big(a(1) \wedge S(1,5) \wedge b(6)\big) \vee 0 = 6 \qquad \textit{(disjunction)}$$
$$\big[(\exists z)(S(0,z) \wedge S(z,6))\big] \vee \big(a(1) \wedge S(1,5) \wedge b(6)\big) \vee 0 = 6 \vdash_w S(0,6) \qquad \textit{(definition of } S\textit{)}$$

7.1.3 Equivalence of the two definitions

Theorem 7.1. *Let $G = (\Sigma, N, \mathrm{rank}, \langle\varphi_A\rangle_{A\in N}, \sigma)$ be a first-order grammar. Then, for every string $w \in \Sigma^*$ for every formula $\varphi(x_1, \ldots, x_k)$ and for all positions $i_1, \ldots, i_k \in \{0, 1, \ldots, |w|\}$,*

$$\vdash_w \varphi(i_1, \ldots, i_k) \quad \textit{if and only if} \quad \models_w \varphi(i_1, \ldots, i_k).$$

7.4 Decision procedure

Decision procedure for FO(LFP). May be regarded as a parsing algorithm. Actually, many actual parsing algorithms for various families of grammars are nothing but implementations of this decision procedure, specialized for the grammar family in question.

7.4.1 Basic algorithm

Theorem 7.2. *Let $G = (\Sigma, N, \mathrm{rank}, \langle\varphi_A\rangle_{A\in N}, \sigma)$ be a first-order grammar, let k be the largest rank of a predicate, let m be the largest number of nested quantifiers in a definition of a predicate. Then there exists an algorithm, which, given an input string $w \in \Sigma^*$, determines whether $w \in L(G)$, and does so in time $O(n^{2k+m})$, using space $O(n^k)$.*

Proof. The algorithm calculates the least model by gradually proving all true elementary statements. There are $O(n^k)$ statements in total. At each step, the algorithm cannot know, which statement it is already able to prove, so it tries proving each of $O(n^k)$ statements. Each nested quantifier requires considering n possibilities for the bounded variables, and thus an attempted proof of each statement requires $O(n^m)$ steps. □

For ordinary grammars, as well as for conjunctive grammars, all predicates are binary ($k = 2$), quantifiers are used, but never nested ($m = 1$), which leads to a decision procedure working in time $O(n^5)$ using space $O(n^2)$. For tree-adjoining grammars, predicates are of degree $k = 4$, and

they use $m = 2$ nested existential quantifiers, which leads to running time $O(n^{10})$ using space $O(n^4)$.

If quantifier elimination were applied to a conjunctive grammar, this would lead to ternary predicates ($k = 3$) and no quantifiers ($m = 0$), and the running time would accordingly be increased to $O(n^6)$.

Bibliography

[1] A. K. Chandra, D. Harel, "Computable queries for relational data bases", *Journal of Computer and System Sciences*, 21:2 (1980), 156–178.

[2] N. Immerman, "Relational queries computable in polynomial time", *Information and Control*, 68:1–3 (1986), 86–104.

[3] A. K. Joshi, L. S. Levy, M. Takahashi, "Tree adjunct grammars", *Journal of Computer and System Sciences*, 10:1 (1975), 136–163.

[4] C. J. Pollard, *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*, Ph. D. thesis, Stanford University, 1984.

[5] D. Radzinski, "Chinese number-names, tree adjoining languages, and mild context-sensitivity", *Computational Linguistics*, 17:3 (1991), 277–299.

[6] W. C. Rounds, "LFP: A logic for linguistic descriptions and an analysis of its complexity", *Computational Linguistics*, 14:4 (1988), 1–9.

[7] M. Y. Vardi, "The complexity of relational query languages", *STOC 1982*, 137–146.

[8] K. Vijay-Shanker, A. K. Joshi, "Some computational properties of tree adjoining grammars", *23rd Meeting of the Association for Computational Linguistics* (Chicago, USA, 8–12 July 1985), 82–93.

[9] K. Vijay-Shanker, D. J. Weir, "The equivalence of four extensions of context-free grammars", *Mathematical Systems Theory* 27:6 (1994), 511–546.

Index

Chandra, Ashok K. (b. 1948), 7

Harel, David (b. 1950), 7

Immerman, Neil (b. 1953), 7

Joshi, Aravind Krishna (b. 1929), 2, 6

Kanazawa, Makoto, 5

Levy, Leon Sholom (1930–2003), 2

Pollard, Carl Jesse (b. 1947), 2

Radzinski, D., 5
Rounds, William Chesley, 2, 7

Salvati, Sylvain, 5

Takahashi, Masako, 2

Vardi, Moshe Ya'akov (b. 1954), 7
Vijay-Shanker, K., 2, 6

Weir, David J., 2