STUDIA UNIV. "BABEŞ–BOLYAI", INFORMATICA, Volume **XLIII**, Number 1, 1998

INDEXED GRAMMARS AND UNIFICATION GRAMMARS

DOINA TĂTAR AND MIHAELA LUPEA

Abstract. Feature structures (typed or untyped) are used extensively for the specification of linguistic information in the unification based phrase structure grammars (UBPSG's). On the other hand, another quite different grammars, indexed grammars are introduced as an extension of context -free grammars and since now they are used in many different contexts. The two sort of grammars have in common the possibility to share information between different parts of a sentential form. In this paper we show how we can generate the same language $\{a^n\ b^n\ c^n\}$ with an indexed and an unification grammar.

1. Introduction

The Chomsky hierarchy contains: grammars of type 0, context- sensitive grammars, context-free grammars and regular grammars. Aho introduced in a paper from 1969 a new type of grammars which is situated in the Chomsky hierarchy between context-sensitive grammars and context-free grammars and are called **indexed grammars**. An indexed grammar generates an **indexed language**. This class of languages is a generalization of the class of context-free languages and preserves the closure properties and decidability results from the last one.

Definition 1. *An* **indexed grammar***is a 5-tuple $G = (N, T, I, P, S)$, where:*

- *- N is a finite nonempty set of symbols called the* **nonterminal alphabet**
- *- T is a finite set of symbols called the* **terminal alphabet**
- *- I is a finite set of* **indices**
- *- $S \in N$ is the* **start symbol**
- *- P is a finite set of* **productions** *of three forms: (1) $A \to \alpha$,*
  (2) $A \to Bf$,
  (3) $Af \to \alpha$
  where $A, B \in N$, $f \in I$ and $\alpha \in (N \cup T)^*$.

1991 *Mathematics Subject Classification.* 68Q50, 03D05.
1991 *CR Categories and Descriptors.* F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems – *Grammar types* F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages – *Classes defined by grammars or automata*, *Operations on languages* .

The **indices** may follow a nonterminal and were introduced to model a context for derivation, context more particular than in the case of context-sensitive grammars. Many parts of a sentential form are related to which other by sharing a common index string. Terminals may not be followed by indices.

Definition 2. *A string of symbols of the form $A\zeta$, where $A \in N$ and $\zeta \in I^*$ is called an* **indexed nonterminal**. *A* **sentential form** *is a string in $(NI^* \cup T)^*$.*

Definition 3. *A* **derivation relation** *($\Longrightarrow$), defined on sentential forms depends on the type of the production which is applied as follows:*

- *(1) If $A \rightarrow X_1X_2...X_n$ is a production of type (1) then:*

$$\beta A\delta\gamma \Longrightarrow \beta X_1\delta_1X_2\delta_2 \ldots X_k\delta_k\gamma]$$

This type of derivation assures that the index string $\delta \in I^*$ distributes over all the nonterminal symbols on the right side of the production, but not over the terminals. If $X_i \in N, \delta_i = \delta$ and if $X_i \in T, \delta_i = \epsilon$.

- *(2) If $A \rightarrow Bf$ is the applied production, then the derivation is:*

$$\beta A\delta\gamma \Longrightarrow \beta Bf\delta\gamma$$

The index string which follows B is composed of f concatenated with the index string of A.

- *(3) If the production $Af \rightarrow X_1X_2...X_n$ is applied, then:*

$$\beta Af\delta\gamma \Longrightarrow \beta X_1\delta_1X_2\delta_2 \ldots X_k\delta_k\gamma$$

.

The first index f is consumed by the nonterminal A, and the remaining indices γ of A distribute over nonterminals as in (1).

The reflexive and transitive closure of the relation $\Longrightarrow$ on $(NI^* \cup T)^*$, denoted $\overset{*}{\Longrightarrow}$, is defined as follows:

1. $\gamma \overset{k}{\Longrightarrow} \delta$, where $k > 0$ if and only if there is a sequence of sentential forms $\alpha_1, \ldots, \alpha_{k-1}$ such that

$$\gamma \Longrightarrow \alpha_1 \Longrightarrow \cdots \Longrightarrow \alpha_{k-1} \Longrightarrow \delta$$

2. $\gamma \overset{+}{\Longrightarrow} \delta$ if and only if there is k¿0 such that $\gamma \overset{k}{\Longrightarrow} \delta$
3. $\gamma \overset{*}{\Longrightarrow} \delta$ if and only if $\gamma = \delta$ or $\gamma \overset{+}{\Longrightarrow} \delta$.

The **indexed language** generated by the indexed grammar $G = (N, T, F, P, S)$ is defined as $L(G) = \{w \in T^* | S \overset{*}{\Longrightarrow} w\}$.

The grammars G_1 and G_2 are said to be **equivalent** if $L(G_1) = L(G_2)$.

A context-free grammar can be transformed into an equivalent grammar in Chomsky normal form. Similary, for an indexed grammar we can construct an equivalent **reduced form** grammar.

Definition 4. *An indexed grammar $G = (N, T, I, P, S)$ is said to be in* **reduced form** *if each production in P is of one of the forms:*

- *(1)* $A \to BC$
- *(2)* $A \to Bf$
- *(3)* $A \to a$
- *(4)* $Af \to B$ *where* $A, B, C \in N$, $f \in I$ *and* $a \in T \cup \{\epsilon\}$.

Theorem 1. *Given an indexed grammar $G = (N, T, F, P, S)$, an equivalent reduced form indexed grammar $G' = (N', T, F, P', S)$ can be constructed.*

Construction:

1. A production of type (1) of definition is replaced by a set of productions specific to the Chomsky normal form, by adding new nonterminals.
2. A production of type (2): $Af \to X_1 \dots X_k$ is replaced by the productions: $Af \to A'$, where A' is a new nonterminal not in N, and the set of productions in Chomsky normal form coresponding to the production: $A' \to X_1 \dots X_k$.

We can observe that in this process of construction are not introduced new indices. This reduced form for an indexed grammar is very useful for processing this type of grammars. The closure properties make indexed languages seem to more closely resemble context-free languages than context-sensitive languages. An indexed language has also a new type of specification by a new automaton model, called a **nested stack automaton**, which is a generalization of push-down automaton. A set is recognized by a nested stack automaton if and only if the set is an indexed language.

2. Unification Based Phrase Structure Grammars

Over the years, computational linguistics adopted many different grammar formalisms which are in use in various NLP projects. Many of current ones are declarative formalisms on the unification paradigm. The unification grammars are phrase structure grammars in which non-terminal and terminals symbols are replaced by feature structures.

Intuitively a feature structure (FS) is a description of some linguistic object, specifying some or all of the information that is asserted to be true of it [2, 5]. We will present shortly the two definitions of (untyped) feature structures and the connection between them following [1].

Definition 5. *A feature structure over a signature Types and Feats is a labeled rooted directed graph represented by the tuple:*

$$F = < Q, \bar{q}, \theta, \delta >$$

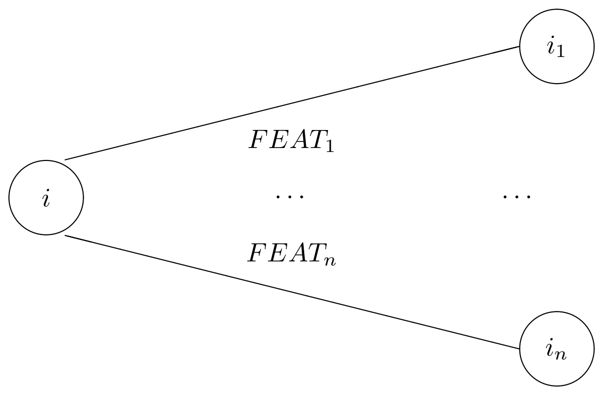
where: • Q *is the finite set of nodes of the graph.*

• $\bar{q} \in Q$ *is the root node*

• $\theta : Q \longrightarrow$ **Type** *is a partial node typing function.*

• δ : **Feat** $\times Q \longrightarrow Q$ *is a partial value function, which associates with a node* i *the nodes* $i_1, \cdots, i_n$ *if* $\delta(FEAT_1, i) = i_1, \cdots, \delta(FEAT_n, i) = i_n$

See figure bellow where the edges are labeled with the feature names and the nodes are labeled with the type names:



As we will use in this paper another formal description for FS's, attribute value matrices (AVM's) let us observe that the correspondence with the graph formalism can be displayed in the following figure:

$$ {}_{i}\begin{bmatrix} FEAT_1 : {}_{i_1}\begin{bmatrix}\cdots\end{bmatrix} \\ \cdots \\ FEAT_n : {}_{i_n}\begin{bmatrix}\cdots\end{bmatrix} \end{bmatrix} $$

In the rewriting relations two notions about FS's are important: subsumption relation and unification operation.

Definition 6. *A feature structure* F *subsumes another feature structure* G *or* $F \sqsubseteq G$ *iff:*

• *if a feature $f \in$ **Feat** is defined in F then f is also defined in G and its value in F subsumes the value in G.*

• *if the values of two paths are shared in F , then they are also shared in G.*

Thus $F \sqsubseteq G$ if G contains more information than F or F is *more general* than G.

The notion of subsumes can be used to define the notion of unification, the main information combining operation in unification based grammars. Unification joins the information in two feature structures into a single result if they are consistent and detects an inconsistency otherwise.

Definition 7. *The result of the unification of two FS's F and F' is an other FS (if it exists), denoted $F \sqcup F'$ which is the most general FS (in the sense of relation $\sqsubseteq$) subsumed by both input FS's.*

Thus, $F \sqcup F'$ is the l. u. b of F and F', if it exists, on the ordering relation $\sqsubseteq$.

Definition 8. *(UBPSG) For a set $Type$ of types, a set $Feat$ of features, a set Lex of terminals (lexical entries), a UBPSG is:*

- *A set $Rule$ of rewriting rules:*

$$E_0 \to E_1 \dots E_n,$$

where each E_i is either a feature structure or a terminal (and in this case $n = 1$ and the rule is named "terminal" or "lexical").

The interpretation of such a rule is: the category E_0 can consist of an expression of category E_1, followed by the category E_2, etc.

- *An initial FS, A_s*

There are many rewriting relation for a UBPSG. We will present now unification in context rewriting [2].

Let σ be a sequence of FS's, which can share information by the tags. This kind of sequence we will call "multi-feature structure " or " multi -FS". So, σ is a sequence $(\sigma, 1), \cdots, (\sigma, n)$, where (σ, i) is the FS of index i from the multi-FS σ. Let ρ be a rewriting rule of UBPSG considered also as a multi-FS where $(\rho, 1)$ is the left side and $(\rho, 2), \cdots, (\rho, k)$ represent the right side of the rule. If (σ, i) and the $(\rho, 1)$ can be unified, let the unificator be a FS $(\sigma', i) = (\sigma, i) \sqcup (\rho, 1)$. As the FS's from σ (and also from ρ) can share information, after unification these FS's are modified. Let denote $\sigma' = (\sigma', 1), ..., (\sigma', n)$, where (σ', i) is the defined as above, the multi-FS σ after unification of the i-th component with the first component of ρ. Analogously, let denote $\rho' = (\rho', 1), ..., (\rho', k)$, the multi-FS ρ after the same unification.

Definition 9. *If (σ, ρ) and (σ', ρ') have the above significance, then we denote $(\sigma, \rho) \xrightarrow{i} (\sigma', \rho')$ and call this relation the " unification in context".*

Definition 10. *If σ_1 and σ_2 are two multi-FS, then the relation of rewriting $\Rightarrow$ between them is defined as following:*

$$\sigma_1 \Rightarrow \sigma_2$$

if:

- *1. There exists a rewriting rule* ρ *such that* $(\sigma_1, \rho) \xrightarrow{i} (\sigma', \rho')$
- *2.* σ_2 *is the multi-FS* $(\sigma', 1), \cdots, (\sigma', i-1), (\rho', 2), (\rho', 3), \cdots, (\rho', k), (\sigma', i+1), \cdots, (\sigma', n)$.

In the following we denote by *Lex* the set of lexical entries of a UBPSG. The accepted sentences are from Kleene closure of *Lex*, denoted *Lex**. If a sentence $w = w_1 \cdots w_j$ is from *Lex** then by $PT_w(1, \cdots, j)$ we denote the sequence of FS's which are in the LHS's of the terminal rules which have in the RHS's the words $w_1, \cdots, w_j$ (in this order).

We denote, like usually, by $\overset{*}{\Rightarrow}$ the transitive reflexive closure of this rewriting.

We can now define the language generated by a UBPSG G:

Definition 11. *If G is a UBPSG, then the language generated by G,* $L(G)$ *is defined as:*

$$L(G) = \{w | w \in Les*, w = w_1 \cdots w_j, \text{there exists}\, \sigma \text{ s.t.} \mathrm{A_s} \overset{*}{\Rightarrow}$$
$$\sigma,\, \sigma\, is\, unifiable\, with\, PT_w(1, \cdots, j)\}$$

3. An indexed language generated by an unification grammar

In this section we will describe how the indexed language $\{a^n\ b^n\ c^n\}$ can be obtained by an unifcation grammar.

As indexed language, it can be obtained from the following indexed grammar:

$$G = (\{S, T, A, B, C\}, \{a, b, c\}, \{f, g\}, P, S)$$

where the production are adnotated by the type (1,2 or 3), as in above definition:

1.
$T \longrightarrow A\ B\ C$
2.
$S \longrightarrow T\ g$
$T \longrightarrow T\ f$
3.
$A\ f \longrightarrow a\ A$
$A\ g \longrightarrow a$
$B\ f \longrightarrow b\ B$
$B\ g \longrightarrow b$
$C\ f \longrightarrow c\ C$
$C\ g \longrightarrow c$

The generation of the word $a^2\ b^s\ c^2$ is the following:

$$S \xrightarrow{2} T\ g \xrightarrow{2} T\ f\ g \xrightarrow{1} A\ f\ g\ B\ f\ g\ C\ f\ g$$

$$\xRightarrow{3} a\ A\ g\ B\ f\ g\ C\ f\ g \xRightarrow{3,3} a\ A\ g\ b\ B\ g\ c\ C\ g \xRightarrow{3,3,3} a^2\ b^2\ c^2$$

The unification grammar for the same language $\{a^n\ b^n\ c^n\}$ is the following:

The set $Feats$ is $\{CAT, LG\}$, the set $Types$ is $\{s, a, b, c, at, bt, ct\}$, the $A_s = [CAT : s]$ and the set Lex is $\{a, b, c\}$.

The production rules are:

$$r1 : [CAT : s] \to \begin{bmatrix}\text{CAT:a}\\ \text{LG:X}\end{bmatrix} \begin{bmatrix}\text{CAT:b}\\ \text{LG:X}\end{bmatrix} \begin{bmatrix}\text{CAT:c}\\ \text{LG:X}\end{bmatrix}$$

$$r2 : \begin{bmatrix}\text{CAT:a}\\ \text{LG: }\begin{bmatrix}\text{LG: Y}\end{bmatrix}\end{bmatrix} \to [CAT : at] \begin{bmatrix}\text{CAT:a}\\ \text{LG:Y}\end{bmatrix} \qquad r3 : \begin{bmatrix}\text{CAT:a}\\ \text{LG:1}\end{bmatrix} \to [CAT : at]$$

$$r4 : \begin{bmatrix}\text{CAT:b}\\ \text{LG: }\begin{bmatrix}\text{LG: Y}\end{bmatrix}\end{bmatrix} \to [CAT : bt] \begin{bmatrix}\text{CAT:b}\\ \text{LG:Y}\end{bmatrix} \qquad r5 : \begin{bmatrix}\text{CAT:b}\\ \text{LG:1}\end{bmatrix} \to [CAT : bt]$$

$$r6 : \begin{bmatrix}\text{CAT:c}\\ \text{LG: }\begin{bmatrix}\text{LG: Y}\end{bmatrix}\end{bmatrix} \to [CAT : bt] \begin{bmatrix}\text{CAT:c}\\ \text{LG:Y}\end{bmatrix} \qquad r7 : \begin{bmatrix}\text{CAT:c}\\ \text{LG:1}\end{bmatrix} \to [CAT : ct]$$

$$r8 : [cat; at] \to a \qquad r9 : [cat; bt] \to b \qquad r10 : [cat; ct] \to c$$

The derivation of the word $w = a^2\ b^2\ c^2$ is the following:

$$[CAT : s] \overset{r1}{\Rightarrow} \begin{bmatrix}\text{CAT:a}\\ \text{LG:X}\end{bmatrix} \begin{bmatrix}\text{CAT:b}\\ \text{LG:X}\end{bmatrix} \overset{r2,r4,r6}{\Rightarrow} [CAT : at] \begin{bmatrix}\text{CAT:a}\\ \text{LG:Y}\end{bmatrix} [CAT : bt]$$

$$\begin{bmatrix}\text{CAT:b}\\ \text{LG:Y}\end{bmatrix} [CAT : ct] \begin{bmatrix}\text{CAT:c}\\ \text{LG:Y}\end{bmatrix}$$

$$\overset{r3,r5,r7}{\Rightarrow} [CAT : at]\ [CAT : at]\ [CAT : bt]\ [CAT : bt]\ [CAT : ct]\ [CAT : ct]$$

In the first rewriting, by the rule $r1$, the unification was as bellow:

$$\begin{bmatrix}\text{CAT:a}\\ \text{LG:X}\end{bmatrix} \sqcup \begin{bmatrix}\text{CAT:a}\\ \text{LG: }\begin{bmatrix}\text{LG: Y}\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\text{CAT:a}\\ \text{LG: }\begin{bmatrix}\text{LG: Y}\end{bmatrix}\end{bmatrix}$$

As the first AVM shares information (by Y) with the second and the third, X is replaced in these AVM's too. The obtained AVM's are exactely the LHS of the rules $r2, r4, r6$ and thus the rewriting is by these rules. The next unification is:

$$\begin{bmatrix}\text{CAT:a}\\ \text{LG:Y}\end{bmatrix} \sqcup \begin{bmatrix}\text{CAT:a}\\ \text{LG:1}\end{bmatrix} = \begin{bmatrix}\text{CAT:a}\\ \text{LG:1}\end{bmatrix}$$

. As Y is made 1 an all the AVM's, the rules $r3, r5, r7$ can be applied.

For the word $w = a^2\ b^2\ c^2$, $PT_w(1, \cdots, 6)$ is $[CAT : at]\ [CAT : at]\ [CAT : bt]\ [CAT : bt]\ [CAT : ct]\ [CAT : ct]$ which is exactely the form obtained from A_s. So, $w = a^2\ b^2\ c^2$ is in $L(G)$.

References

[1] B. Carpenter, *The logic of typed feature structures*, Cambridge University Press, 1992.

[2] N. Francez, S. Wintner, *Feature structure based linguistic formalisms*, draft 1998, http.

[3] G. Gazdar, C. Mellish: *NLP in Prolog. An introduction to CL*, Addison Wesley, New York, 1989.

[4] M. Johnson, *Attribute-value logic and the theory of grammar*, CSLI, 1988.

[5] M. Shieber, *Introduction to unification-based approaches to grammars*, CSLI, 1986.

[6] D. Tatar, *Compiling definite clause grammars*, Studia Univ. "Babes-Bolyai", Informatica, vol I, nr.1, 1996, pp 45-56.

Babeş-Bolyai University, Faculty of Mathematics and Informatics,
RO 3400 Cluj-Napoca, str. Kogălniceanu 1, Romania

E-mail address: {dtatar,lupea}@cs.ubbcluj.ro