AN EARLEY ALGORITHM FOR GENERIC ATTRIBUTE AUGMENTED GRAMMARS AND APPLICATIONS

Frédéric Tendeau*
INRIA-Rocquencourt†, BP 105, 78153 Le Chesnay CEDEX, France
E-mail: Frederic.Tendeau@inria.fr

Abstract

We describe an extension of Earley's algorithm which computes the decoration of a shared forest in a generic domain. Attribute computations are defined by a morphism from leftmost derivations to the generic domain, which leaves the computations independent from (even if guided by) the parsing strategy. The approach is illustrated by the example of a definite clause grammar, seen as CF-grammars decorated by attributes.

1 Introduction

Computational linguistics offers a variety of grammatical formalisms, such as DCGs (Pereira and Warren [13]), LFGs (Kaplan and Bresnan [9]), FUGs (Kay [11]), PATR (Shieber [14]), ... They often rely upon a context-free backbone, on which contextual information is grafted. Such information amounts to a collection of attributes of different types: first-order terms, numerical data, probabilities, feature structures, etc. Hence a parser has to perform context-free and attribute operations, but essentially we think that the algorithmics is not radically different accross linguistic formalisms. Of course the computation details differ but we show that it is possible to abstract sufficiently to see many attribute domains in a unified framework.

In a non-deterministic parser, Lang [12] showed the independence between the strategy (expressed by a non-deterministic pushdown automaton) and the dynamic programming interpretation, which handles non-determinism. The present paper can be seen as a proposal to consider the attribute decoration independently from the two previous aspects.

1. Provided the attribute domain respects some algebraic structure (namely a semiring),

2. given both a strategy for attribute computation, and a dynamic programming interpretation,

the decoration of the parse forest can be computed abstractly, i.e. relying entirely on the algebraic structure.

Our goal is modularity: Earley's algorithm is used to show that a parser can be designed independently from the attribute domain and can be applicable to several linguistic formalisms.

Our extension of Earley's algorithm computes the abstract decoration of a parse forest, and we show that abstract computations are applicable to a variety of interpretations: from probabilities to unification grammar formalisms.

Dynamic programming techniques were introduced by Bellman [3] to reduce the complexity of operational research problems from exponential to polynomial (e.g. the knapsack problem), Cocke, Kasami [10], Younger [17] as well as Earley [8] applied them to non-deterministic context-free parsing. They are applicable on recursive problems for which each sub-problem can be identified by an index and they consist in tabulating the computations, ensuring the solution of each sub-problem is computed only once.

In [15], we presented computations of stochastic information in parallel with the parsing process, taking advantage of dynamic programming techniques. We attempt to apply them to attribute augmented context-free grammars in such a way that the following algorithm appears as a generalization of the stochastic Earley's algorithm in [15]. In this paper, probabilities are defined with respect to leftmost derivations and the so-called *recognition probability* corresponds to the

*presently working at Rank Xerox Research Centre, Grenoble, France.

†this work was partially supported by the grant 95-B030 from the Centre National d'Études des Télécommunications.

classical *inside probability* (Baker [2]). We propose to abstract away from probabilities and to compute a *recognition decoration*, defined with respect to the (abstract) decoration of leftmost derivations, in the same way the recognition probability is in [15].

We described in [16] a formalism for generic decoration, generalizing the probabilities. The sum and the product between positive real numbers are replaced by an abstract sum and product. The sum models non-determinism, or ambiguity, i.e. the fact that a set of leftmost derivations can be associated with a single sentence. The order in which the set of possible leftmost derivations is built is not relevant, neither is the order of evaluation of the associated attributes; therefore, the abstract sum must be commutative.

The product models the construction of a single leftmost derivation from two others (by plugging the second one into the yield of the first one, which is equivalent to parse-tree construction). When considering a sequence of three derivations, the associated semantics must be the same with a bottom-up and a top-down construction, meaning that the abstract product must be associative. The same holds for the sum.

A last condition expected for the abstract operations is distributivity of the product w.r.t. the sum, in order to factorize and expand whenever needed.

As a consequence, the abstract domain must have a semiring structure, which justifies the use of the algebraic power series formalism in [16], where we specify in which conditions and how dynamic programming techniques can be used to compute the abstract decoration of a shared forest, from the definition of an abstract decoration on the grammar.

The theory of algebraic power series, introduced by Schützenberger and Chomsky [6], allows not only to generalize many attribute domains with the semiring structure, but also the context-free structure itself; indeed, the computed values can be probabilities, or first order terms (i.e. classical attributes), but also leftmost derivations or shared forests. Therefore, the decoration of a syntactic construction by attributes can be seen as a morphism from the context-free backbone to the attribute domain: a semiring morphism. So, any parsing algorithm can be extended by a morphism in order to compute attribute decorations.

In many cases however, the semiring structure is too strong: the product might be a partial function rather than an application. For instance, we describe leftmost derivations in an algebraic manner and naturally, composing (multiplying) two derivations is possible only if the second one *can actually* be applied after the first one: consider for example $A \Longrightarrow \alpha \textcircled{\ell} B \Longrightarrow \beta$ (where $\textcircled{\ell}$ denotes the composition of leftmost derivations) then such a product is defined only if $\alpha = wB\gamma$, in which case the composition yields $A \Longrightarrow \alpha \Longrightarrow w\beta\gamma$.

Hence the *partial* semiring of leftmost derivations is defined and attribute computations are defined as a partial semiring morphism from derivations to the attribute domain. As parse trees are equivalent to leftmost derivations, any syntactic algorithm only has to apply the decoration morphism to compute the decoration.

The sequel is organized as follows: section 2 defines the algebra of leftmost derivations, the attribute algebra, and the decoration (which is a semiring morphism) of the leftmost derivations in the attribute domain. Then, after the presentation of shared forests, section 2 introduces the adaptation of Earley items in order to carry attributes. Section 3 describes the algorithm with attribute computations. Section 4 applies our approach to DCGs, which are presented as CFGs decorated in a semiring. Section 5 runs the algorithm on an example of DCG.

2 Analysis material

2.1 Grammar and decoration

- Consider: a context-free grammar $G = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, the vocabulary $\mathcal{V} = \Sigma \cup \mathcal{N}$, the relation *leftmost derive* is $\underset{\ell}{\Longrightarrow} = \{(xA\omega,\ x\eta\omega) \mid xA\omega \Longrightarrow x\eta\omega\}$. From this relation, a *leftmost derivation* is defined as a sequence s.t. each element derives the following one, more formally: $(\alpha_i)_{1 \leq i \leq l} \in \mathcal{V}^{\mathbb{N}}$ s.t. $l \geq 0$ and if $l > 1$ then $\forall i \geq 1$: $\alpha_i \underset{\ell}{\Longrightarrow} \alpha_{i+1}$ —for $l = 0$, the empty sequence corresponds to the empty derivation denoted by (ϵ). For example, the derivation denoted by $A \Longrightarrow \alpha \Longrightarrow w\beta\gamma$ in the introduction corresponds to the sequence $(A, \alpha, w\beta\gamma)$.

  The set of all leftmost derivations of α into β is denoted by $\alpha \overset{\ell}{\rightsquigarrow} \beta$, and for $n \in \mathbb{N}$, the subset of the latter, restricted to derivations of length n, is $\alpha \overset{n}{\underset{\ell}{\rightsquigarrow}} \beta = \{d \in \alpha \overset{\ell}{\rightsquigarrow} \beta \mid d \text{ is a sequence of length } n+1\}$. Note that for $n = 0$: $\alpha = \beta$ and the derivation is (α).

- Attribute decoration of G in an abstract domain $\mathcal{A}$
  - First, we define a *partial* monoid (this unusual algebraic structure has already been introduced by Arbib and Manes [1]): $(\mathcal{A}, \times_{\mathcal{A}}, 1_{\mathcal{A}})$ is a partial monoid iff $\times_{\mathcal{A}}$ is a partial function over $\mathcal{A} \times \mathcal{A}$, $1_{\mathcal{A}}$ is a neutral element for $\times_{\mathcal{A}}$ ($\forall a \in \mathcal{A}$:

$a \times_{\mathcal{A}} 1_{\mathcal{A}} = 1_{\mathcal{A}} \times_{\mathcal{A}} a = a$), and $\times_{\mathcal{A}}$ is partially associative ($\forall a, b, c \in \mathcal{A}$: $a \times_{\mathcal{A}} b$ and $b \times_{\mathcal{A}} c$ are defined implies $(a \times_{\mathcal{A}} b) \times_{\mathcal{A}} c$ and $a \times_{\mathcal{A}} (b \times_{\mathcal{A}} c)$ are defined and equal). Naturally, if $\times_{\mathcal{A}}$ is a mapping then $(\mathcal{A}, \times_{\mathcal{A}}, 1_{\mathcal{A}})$ is a monoid, and in addition, if $\times_{\mathcal{A}}$ is commutative then so is the monoid.

- Then $(\mathcal{A}, +_{\mathcal{A}}, \times_{\mathcal{A}}, 0_{\mathcal{A}}, 1_{\mathcal{A}})$ is a *partial* semiring iff $(\mathcal{A}, \times_{\mathcal{A}}, 1_{\mathcal{A}})$ is a partial monoid, $(\mathcal{A}, +_{\mathcal{A}}, 0_{\mathcal{A}})$ is a commutative monoid, $0_{\mathcal{A}}$ is absorbent for $\times_{\mathcal{A}}$ (i.e. $\forall c \in \mathcal{A}$: $c \times_{\mathcal{A}} 0_{\mathcal{A}} = 0_{\mathcal{A}} = 0_{\mathcal{A}} \times_{\mathcal{A}} c$), and $\times_{\mathcal{A}}$ is distributive with respect to $+_{\mathcal{A}}$.
- The decoration of G in the partial semiring $(\mathcal{A}, +_{\mathcal{A}}, \times_{\mathcal{A}}, 0_{\mathcal{A}}, 1_{\mathcal{A}})$ is represented by the mapping $\phi_{\mathcal{A}}: \mathcal{R} \to \mathcal{A}$.

We want to see the syntactic constructions from an algebraic point of view, in order to present the decoration as a (partial) semiring morphism. Before the operations, the ground set must be defined.

Definition 1. The *set of leftmost derivations* is

$$\Delta_\ell = (\epsilon) \cup \bigcup_{n \in \mathbb{N}} \alpha \overset{n}{\underset{\ell}{\rightsquigarrow}} \beta$$

For short, the composition between leftmost derivations $d' \textcircled{\ell} d''$ is defined if the first element of d'' occurs in the last element of the d', and it consists in continuing the derivation d' by using d''.

Definition 2. The *composition of leftmost derivations* is the function $\textcircled{\ell}$: $\Delta_\ell \times \Delta_\ell \longrightarrow \Delta_\ell$ such that $\forall d', d'' \in \Delta_\ell$:

$$d' \textcircled{\ell} d'' = \begin{cases} d' & \text{if } d'' = (\epsilon) \\ d'' & \text{if } d' = (\epsilon) \\ d \in \alpha \overset{l'+l''}{\underset{\ell}{\rightsquigarrow}} x\gamma\delta & \text{if } d' \in \alpha \overset{l'}{\underset{\ell}{\rightsquigarrow}} x\beta\delta, \text{ and } d'' \in \beta \overset{l''}{\underset{\ell}{\rightsquigarrow}} \gamma \text{ such that } \begin{cases} \forall i \leq l' \colon d_i = d'_i \\ \forall i > l' \colon d_i = x d''_{i-l'} \delta \end{cases} \\ \text{undefined} & \text{otherwise} \end{cases}$$

where d_i denotes the i-th element of the sequence d.

For instance, the example of composition in the introduction can be rewritten: $(A, \alpha) \textcircled{\ell} (B, \beta) = (A, \alpha, w\beta\gamma)$.

In order to deal with non-determinism, sets of leftmost derivations are considered and the function $\textcircled{\ell}$ is naturally extended on $\wp(\Delta_\ell)$ (the power set of Δ_ℓ) in such a way that $\forall e \in \wp(\Delta_\ell)$: $\emptyset \textcircled{\ell} e = \emptyset = e \textcircled{\ell} \emptyset$. Then one easily checks that $(\wp(\Delta_\ell), \cup, \textcircled{\ell}, \emptyset, (\epsilon))$ is a partial semiring.

We consider derivations starting from the axiom, or more generally, from a non-terminal. At parse time, dealing with a $\mathcal{V}^*$ string always comes from the derivation of a non-terminal. Therefore, consider the set $\Delta_\ell^{\mathcal{N}}$ of leftmost derivations of which first element is a non-terminal. Clearly, $(\wp(\Delta_\ell^{\mathcal{N}}), \cup, \textcircled{\ell}, \emptyset, (\epsilon))$ is a partial semiring. This partial semiring is going to be the starting point to compute the abstract decoration.

Definition 3. The function $\phi_{\mathcal{A}}$ is extended as a partial semiring morphism of $\wp(\Delta_\ell^{\mathcal{N}})$ to $\mathcal{A}$, i.e., $\phi_{\mathcal{A}}(\emptyset) = 0_{\mathcal{A}}$, $\phi_{\mathcal{A}}((\epsilon)) = 1_{\mathcal{A}}$, $\phi_{\mathcal{A}}(e_1 \cup e_2) = \phi_{\mathcal{A}}(e_1) +_{\mathcal{A}} \phi_{\mathcal{A}}(e_2)$, and $\phi_{\mathcal{A}}(e_1 \textcircled{\ell} e_2) = \phi_{\mathcal{A}}(e_1) \times_{\mathcal{A}} \phi_{\mathcal{A}}(e_2)$, for all $e_1, e_2 \in \wp(\Delta_\ell^{\mathcal{N}})$.

Grammar rules can be considered as elementary leftmost derivations. If a set s of leftmost derivations is represented with the help of an algebraic expression involving only $\cup$ and $\textcircled{\ell}$as only operations, and with $\mathcal{R}$ as only constants, then it is possible to associate any decoration in $\mathcal{A}$ with s: only by substituting $\cup$ by $+_{\mathcal{A}}$, $\textcircled{\ell}$ by $\times_{\mathcal{A}}$ and elements of $\mathcal{R}$ respectiveley by elements of $\phi_{\mathcal{A}}(\mathcal{R})$, i.e. by a semiring morphism.

In fact, our algorithm computes such a symbolic expression.

2.2 Shared forest

The following notation is used to describe substrings of any $w \in \Sigma^*$: for $0 \leq i \leq j \leq |w|$: $w_{]i,j]}$ stands for the substring of w comprised within the index interval $]i, j]$ (if $i = j$, the empty string is intended).

Billot and Lang [4] represent the set of all parse trees for x with respect to G by mean of a context-free grammar $\boldsymbol{g}_x$. This grammar is called the shared forest of x with respect to G and has a polynomial size (cubic if G is in Chomsky normal form). Classically, the nodes of a parse-tree are instances of G, which justifies the grammatical form of $\boldsymbol{g}_x$. In addition, a suitable choice of the names of these instances leads to the polynomial complexity. Indeed, new non-terminals are introduced: $A^{i,j}$ is such that $A \in \mathcal{N}$ and $A \overset{*}{\Longrightarrow} x_{]i,j]}$. The (finite) set of such non-terminals is $\overline{\mathcal{N}} = \mathcal{N} \times \{0, \cdots, |x|\} \times \{0, \cdots, |x|\}$. A labelling mapping is defined from instances to original symbols of G: consider *lab*: $\overline{\mathcal{N}} \to \mathcal{N}$ such that $\forall A^{i,j} \in \overline{\mathcal{N}}$: $lab(A^{i,j}) = A$; consider $\overline{\mathcal{V}} = \overline{\mathcal{N}} \cup \Sigma$, then *lab* is extended first as the identity mapping on Σ and naturally as a morphism of $\overline{\mathcal{V}}^*$ to $\mathcal{V}^*$. A new (finite) set of rules is defined: $\overline{\mathcal{R}}$ is the biggest subset of $\overline{\mathcal{N}} \times \overline{\mathcal{V}}^*$ such that *lab* can be extended as a morphism of $\overline{\mathcal{R}}$ to $\mathcal{R}$.

Definition 4. The *shared forest* g_x for x with respect to $(\Sigma, \mathcal{N}, \mathcal{R}, S)$ is the context-free grammar obtained after reduction of $(\Sigma, \overline{\mathcal{N}}, \overline{\mathcal{R}}, S^{0,n})$.

Note that $\phi_{\mathcal{A}}(A \overset{}{\underset{\ell}{\rightsquigarrow}} x_{]i,\,j]}) = \phi_{\mathcal{A}}(A^{i,j} \overset{}{\underset{\ell}{\rightsquigarrow}} x_{]i,\,j]})$ (the first derivation uses G, the latter uses g_x).

Exemple 1 Let G be defined by the rules $S \rightarrow SS$ and $S \rightarrow a$. The sentence aaa has two parse-trees which can be represented in a grammatical form in the following way:

$$\begin{array}{lcll} S_1 & \rightarrow & S_2 & S_3 \\ S_2 & \rightarrow & a & \\ S_3 & \rightarrow & S_4 & S_5 \\ S_4 & \rightarrow & a & \\ S_5 & \rightarrow & a & \end{array} \qquad \text{and} \qquad \begin{array}{lcll} S'_1 & \rightarrow & S'_2 & S'_3 \\ S'_2 & \rightarrow & S'_4 & S'_5 \\ S'_3 & \rightarrow & a & \\ S'_4 & \rightarrow & a & \\ S'_5 & \rightarrow & a & \end{array}$$

but with a more suitable choice of the instance names, these can be represented in a single grammar: the shared forest

$$\begin{array}{lcll} S^{0,3} & \rightarrow & S^{0,1} & S^{1,3} \\ S^{0,3} & \rightarrow & S^{0,2} & S^{2,3} \\ S^{0,2} & \rightarrow & S^{0,1} & S^{1,2} \\ S^{1,3} & \rightarrow & S^{1,2} & S^{2,3} \\ S^{0,1} & \rightarrow & a & \\ S^{1,2} & \rightarrow & a & \\ S^{2,3} & \rightarrow & a & \end{array}$$

□

2.3 Earley items and decoration

At parse-time, we rename the non-terminals of the items incrementally in the following manner: when a non-terminal $N \in \mathcal{N}$ is shifted in a completion step of Earley's algorithm, it is renamed by some $N^{i,j} \in \overline{\mathcal{N}}$.

Definition 5. An *Earley item* is an element of $\mathcal{I}(G,x) = \mathcal{N} \times \overline{\mathcal{V}}^* \times \mathcal{V}^* \times \{0,\cdots,|x|\} \times \{0,\cdots,|x|\}$, denoted by $[A \rightarrow \overline{\omega} \bullet \delta,\ i,\ j]$ and satisfying: $A \rightarrow \omega\delta \in \mathcal{R}$, $\overline{\omega} \overset{*}{\underset{\ell}{\Longrightarrow}} x_{]i,\,j]}$, and $\overline{\omega} = \overline{\alpha} B^{p,q} \overline{\beta}$ implies $B^{p,q} \overset{*}{\underset{\ell}{\Longrightarrow}} x_{]p,\,q]}$.

Definition 6. The 4-tuple $< A \rightarrow \overline{\alpha} \bullet \beta, i, j, \mathbf{R} >$ is a *decorated item* iff $I = [A \rightarrow \overline{\alpha} \bullet \beta,\ i,\ j]$ is an Earley item such that $\phi^{\mathbf{r}}_{\mathcal{A}}(I) = \mathbf{R}$, where $\phi^{\mathbf{r}}_{\mathcal{A}}$ is defined below.

The recognition decoration corresponds, for an item $[A \rightarrow \overline{\alpha} \bullet \beta,\ i,\ j]$, to the decoration of the recognized part: $\overline{\alpha}$. In general, it is a vector because the respective decoration of $\overline{\alpha}$ elements might not be composable, i.e., their product in $\mathcal{A}$ may be undefined ($\mathcal{A}$ is a partial semiring). This explains that the yield of $\phi^{\mathbf{r}}_{\mathcal{A}}$ is $\mathcal{A}^*$. Note that if $\mathcal{A}$ is a semiring then the recognition decoration is a scalar: the product ($\times_{\mathcal{A}}$) of all components of the vector.

Definition 7. The *recognition decoration* is the function $\phi^{\mathbf{r}}_{\mathcal{A}}$: $\mathcal{I}(G,x) \rightarrow \mathcal{A}^*$ such that $\forall \overline{\alpha} = w_0 A^{p_1,q_1} w_1 \cdots w_{m-1} A^{p_m,q_m} w_m$:

$$\begin{array}{rcl} \phi^{\mathbf{r}}_{\mathcal{A}}([A \rightarrow \overline{\alpha} \bullet \beta,\ i,\ j]) & = & v \in \mathcal{A}^m \\ \text{such that } \forall k \in \{1,\cdots,m\} : v_k & = & \phi_{\mathcal{A}}(A^{p_k,q_k} \overset{}{\underset{\ell}{\rightsquigarrow}} x_{]p_k,\,q_k]}) \end{array}$$

Note that the recognition decoration is a bottom-up information.

When a right-hand side is entirely recognized, e.g. $[A \rightarrow \overline{\alpha} \bullet,\ i,\ j]$, the whole rule is recognized and the decoration of $A \rightarrow \alpha$ must be composed to the decoration of $\overline{\alpha}$. Hence, the product $\times_{\mathcal{A}}$ is extended on $\mathcal{A} \times \mathcal{A}^*$.

Definition 8. The function $\times_{\mathcal{A}}: \mathcal{A} \times \mathcal{A}^* \rightarrow \mathcal{A}$ is such that $\forall a \in \mathcal{A}, v \in \mathcal{A}^m$:

$$a \times_{\mathcal{A}} v = \begin{cases} a & \text{if } m = 0 \\ (\cdots(a \times_{\mathcal{A}} v_1) \times_{\mathcal{A}} \cdots) \times_{\mathcal{A}} v_m & \text{otherwise} \end{cases}$$

i.e. the product from left to right of a and elements of v.

3 Earley-like algorithm with attribute valuation

Our dynamic programming interpretation consists in destroying the stack of the pushdown automaton: all stack elements (i.e. decorated items) are stored in a set $\mathcal{E}$. The name of these items ensures the correctness of (context-free) computations.

Because of non-determinism, attribute computation cannot be performed immediately: symbols are introduced at each level and they are evaluated after each level is complete.

3.1 Symbolic decoration: recognition of A between i and j

$\phi_{\mathcal{A}}(A^{i,j})$ is introduced because a subtree rooted in $A^{i,j}$ and spanning $x_{]i,j]}$ may be produced more than once. This symbol represents the decoration of all derivations starting from A and ending with $x_{]i,j]}$. The index j is assumed fixed, there is one such symbol for each pair (A, i):

$$\phi_{\mathcal{A}}(A^{i,j}) = \phi_{\mathcal{A}}(A \overset{*}{\underset{\ell}{\rightsquigarrow}} x_{]i,j]}) \tag{1}$$

$$= \sum_{\substack{r=A\longrightarrow w_0A_1w_1\cdots w_{p-1}A_pw_p \\ s.t.\ x_{]i,j]}=w_0y_1w_1\cdots w_{p-1}y_pw_p}} \phi_{\mathcal{A}}(r) \times_{\mathcal{A}} \phi_{\mathcal{A}}(A_1 \overset{*}{\underset{\ell}{\rightsquigarrow}} y_1)\times_{\mathcal{A}}\cdots\times_{\mathcal{A}}\phi_{\mathcal{A}}(A_p \overset{*}{\underset{\ell}{\rightsquigarrow}} y_p) \tag{2}$$

hence $\phi_{\mathcal{A}}(A^{i,j}) =$

$$\sum_{\substack{r=A\longrightarrow w_0A_1w_1\cdots w_{p-1}A_pw_p \\ s.t.\ x_{]i,j]}=w_0y_1w_1\cdots w_{p-1}y_pw_p}} \phi_{\mathcal{A}}(r) \times_{\mathcal{A}} \phi_{\mathcal{A}}({A_1}^{i+|w_0|, i+|w_0y_1|})\times_{\mathcal{A}}\cdots\times_{\mathcal{A}}\phi_{\mathcal{A}}({A_p}^{i+|w_0\cdots w_{p-1}|, j-|w_p|}) \tag{3}$$

The symbols $\phi_{\mathcal{A}}(A^{j,j})$ form an independant equation system. Let us assume that for all A, $\phi_{\mathcal{A}}(A^{j,j}) = \phi_{\mathcal{A}}(A \overset{*}{\underset{\ell}{\rightsquigarrow}} \epsilon)$ is known. Note that it can be statically computed.

Now, consider $i \neq j$ and assume that for all $k_1 \leq k_2 < j$: $\phi_{\mathcal{A}}(A^{k_1,k_2})$ is known. The system is then reduced to the variables $\phi_{\mathcal{A}}(A^{k,j})$ s.t. $k < j$. Moreover, in the equation (3), and in each term of the right-hand side, there is at most one variable $\phi_{\mathcal{A}}(A^{k,j})$ ($i \leq k < j$), hence the system is linear.

Let us solve this system by substitution.

Applying the substitution method to solve the system leads to cyclic (linear) equations: the same variable occurs in both the right-hand side and the left-hand side. Indeed, some cycles appear in equations of the form

$$\phi_{\mathcal{A}}(A^{k,j}) = a\times_{\mathcal{A}}\phi_{\mathcal{A}}(A^{k,j})+_{\mathcal{A}}b \tag{4}$$

with a and b independent from $\phi_{\mathcal{A}}(A^{k,j})$.

Let the Kleene star *: $\mathcal{A} \to \mathcal{A}$ be defined by $\forall a \in \mathcal{A}$:

$$a^* = \lim_{k\to\infty} \sum_{i\in\{0,\cdots,k\}}^{\mathcal{A}} a^i$$

then the solution of (4) is a^*b, if this limit exists in $\mathcal{A}$. More generally, cycles look like

$$\phi_{\mathcal{A}}(A^{k,j}) = a_1\times_{\mathcal{A}}\phi_{\mathcal{A}}(A^{k,j})\times_{\mathcal{A}}a_2 +_{\mathcal{A}} b \tag{5}$$

with a_1, a_2 and b independent from $\phi_{\mathcal{A}}(A^{k,j})$. This amounts to the equation (4) only if $\mathcal{A}$ is commutative (then $a = a_1\times_{\mathcal{A}}a_2$). Otherwise let the limit function be defined by $lim_{\mathcal{A}}$: $\mathcal{A}\times\mathcal{A}\times\mathcal{A}\to\mathcal{A}$, s.t. $\forall a, b, c \in \mathcal{A}$:

$$lim_{\mathcal{A}}(a,\ b,\ c) = \lim_{k\to\infty} \sum_{i\in\{0,\cdots,k\}}^{\mathcal{A}} (a^i\times_{\mathcal{A}}b\times_{\mathcal{A}}c^i)$$

then the solution of 5 is $lim_{\mathcal{A}}(a_1, b, a_2)$, provided that this limit exists.

The definition domains of these functions are such that the limits above exist in $\mathcal{A}$ with respect to an appropriately chosen notion of convergence in $\mathcal{A}$.

Note that in non-commutative semirings, $lim_{\mathcal{A}}$ cannot be defined from a^*. But $a^* = lim_{\mathcal{A}}(a, 1_{\mathcal{A}}, 1_{\mathcal{A}})$ and if $\mathcal{A}$ is commutative, then the limit function amounts to the Kleene star: $lim_{\mathcal{A}}(a_1, b, a_2) = (a_1\times_{\mathcal{A}}a_2)^*\times_{\mathcal{A}}b$.

Therefore, for a given j, the system $\phi_{\mathcal{A}}(A^{i,j})$ has a computable solution assuming

1. $\phi_{\mathcal{A}}(A^{j,j})$ is known;

2. $lim_{\mathcal{A}}$ is well defined and always used in its definition domain; the Kleene star is enough if $\mathcal{A}$ is commutative.

3.2 Algorithm

Vectors of $\mathcal{A}^*$ are denoted between $\langle$ and $\rangle$. A concatenation, denoted by $\diamond$, is defined on $\mathcal{A}^*$. The empty vector is denoted by $\langle\rangle$. The mapping $\overline{\phi_{\mathcal{A}}}$: $\overline{\mathcal{R}} \rightarrow \mathcal{A}$ is a decoration mapping introduced during parsing because it offers a decoration of the shared forest, hence more precise than $\phi_{\mathcal{A}}$: $\mathcal{R} \rightarrow \mathcal{A}$.

$\mathcal{E}$:= $\{<S' \rightarrow \bullet S\$,\ 0,\ 0,\ 1_{\mathcal{A}}>\}$;
`for` j := 0 `to` $|x|$ `loop`
  `loop` **(intra-level)**
    (Prediction)
    `if` $<A \rightarrow \overline{\alpha} \bullet B\delta,\ i,\ j,\ \mathbf{R}> \in \mathcal{E}$ `then add` $<B \rightarrow \bullet\eta,\ j,\ j,\ \langle\rangle>$ `to` $\mathcal{E}$; `end if`;
    (Completion)
    `if` $<A \rightarrow \overline{\alpha}\bullet,\ i,\ j,\ \mathbf{R}'> \in \mathcal{E}$ `and` $<B \rightarrow \overline{\eta} \bullet A\beta,\ k,\ i,\ \mathbf{R}> \in \mathcal{E}$ `then`
      `add` $<B \rightarrow \overline{\eta} A^{i,j} \bullet \beta,\ k,\ j,\ \mathbf{R} \diamond \langle \phi_{\mathcal{A}}(A^{i,j}) \rangle>$ `to` $\mathcal{E}$;
      `output` $(A^{i,j} \rightarrow \overline{\alpha},\ \overline{\phi_{\mathcal{A}}}(A^{i,j} \rightarrow \overline{\alpha}) \times_{\mathcal{A}} \mathbf{R}')$;
    `end if`;
  `until` no more item is added to $\mathcal{E}$;
  `loop` **(Scan)**
    `if` $<A \rightarrow \overline{\alpha} \bullet x_j \beta,\ i,\ j,\ \mathbf{R}> \in \mathcal{E}$ `then add` $<A \rightarrow \overline{\alpha} x_j \bullet \beta,\ i,\ j+1,\ \mathbf{R}>$ `to` $\mathcal{E}$; `end if`;
  `until` no more item is added to $\mathcal{E}$;
`end loop`;

The complexity depends on the semiring under consideration. If $\mathcal{A}$ is so that the cost of each operation is 1 then the time and space complexities are the same as the original syntactic parser, i.e. $\mathcal{O}(n^{L+1})$, where L is the size of the longest right-hand side.

For example, the semiring $(\mathbb{R}^+, +, \times, 0, 1)$ respects such a constraint. If $\phi_{\mathcal{A}}$ is defined so that to describe a probabilistic grammar, then we get a probabilistic parser which computes at each node N of the shared forest, the probability of the whole shared forest dominated by N (see [15]). Another interesting computation is the selection of the best tree, which is achieved by taking the semiring $(\mathbb{R}^+, \max, \times, 0, 1)$ and memorizing only the nodes corresponding to the best probability.

4 Application to DCGs

Naturally, DCGs have a context-free backbone. We just have to find a semiring structure which describes computations with first order terms. This is done in two steps: first, section 4.1 presents Herbrand's domain in a (sup-semi-) lattice framework (which is very close to a semiring) and then, section 4.2 puts first order terms together with the CF information, to form the (partial) semiring of DCGs.

4.1 Herbrand's domain

Consider: a set of variables *Var*, a set of function symbols *Fonc*, a set of (first order) terms $\mathcal{T}$, constructed over *Var* and *Fonc* (by definition $0_{\mathcal{T}}$ is a term called inconsistent term), a set *Subst* of substitutions. The application of a substitution σ to a term t is conventionally denoted by $t\sigma$. By definition 0_{Subst} is a substitution called inconsistent substitution and verifies $\forall t \in \mathcal{T}$: $t0_{Subst} = 0_{\mathcal{T}}$.

Substitutions are used to define a partial pre-order over terms, for which *small* means *precise*.

Definition 9. $\forall t, r \in \mathcal{T}$: $t \sqsubseteq r$ iff there exists a substitution σ verifying $t = r\sigma$.

Note that $0_{\mathcal{T}}$ is the least element of $\mathcal{T}$ w.r.t. $\sqsubseteq$.

To get a partial order, terms which are alphabetical variants are collapsed in a single term. From the (partial) pre-order, an equivalence relation between terms is defined: $\forall s, t \in \mathcal{T}$: $s \equiv t$ iff $s \sqsubseteq t$ and $t \sqsubseteq s$.

Consequently, two quotient terms in $\mathcal{T}/_{\equiv}$ always have a greatest lower bound $\sqcap$ in $\mathcal{T}/_{\equiv}$: $(\mathcal{T}/_{\equiv}, \sqcap, 0_{\mathcal{T}}, \sqsubseteq/_{\equiv})$ is a complete sup-semi lattice.

The unification between two terms is then defined as their greatest lower bound.

Definition 10. For two (quotient) terms s and t, the unification of s and t is $s \sqcap t$.

The unification is extended elementwise to vectors of terms —the only vector with which the empty vector (the vector with no components: $\vec{\epsilon}$) is unifiable is $\vec{\epsilon}$, the result is the substitution 1_{Subst}.

4.2 The partial semiring

Definite clause grammars

A definite clause is seen as a pair composed by a context-free rule $r = A_0 \rightarrow x_0 A_1 x_1 ... A_k x_k$ (in which terminal and non terminal symbols respectively correspond to extensional and intentional predicates) and a mapping $\{0, \cdots, k\} \rightarrow \mathcal{T}^*$ (where $\mathcal{T}^*$ is the set of finite vectors over $\mathcal{T}$) that associates each non-terminal of r with a vector of first order terms. The collection of definite clauses over $\mathcal{R}$ is $\mathcal{C}(\mathcal{R}) = \mathcal{R} \times (\mathcal{T}^*)^{\mathbb{N}}$.

A definite clause grammar is a set of definite clauses.

Domain

The domain $\mathcal{D}$ is the subset of $(\overline{\mathcal{N}} \times (\overline{\mathcal{N}} \cup \Sigma)^*) \times (\mathcal{T}^*)^{\mathbb{N}} \cup \{0_\mathcal{D}, 1_\mathcal{D}\}$ such that for any $d = ((\overline{A}, \overline{\alpha}), \Theta) \in \mathcal{D}$: $(\overline{A}, \overline{\alpha})$ verifies $\overline{A} \underset{\ell}{\overset{*}{\Longrightarrow}} \overline{\alpha}$, and $\mathrm{Dom}(\Theta) = \{0, \cdots, k\}$ iff $\overline{\alpha}$ contains k non-terminals.

The domain of the semiring is the power set of $\mathcal{D}$: $\wp(\mathcal{D})$, denoted by $\mathcal{H}$. The sum $+_\mathcal{H}$ is the union in $\mathcal{H}$. Now we need a product.

Product First, we define $\times_\mathcal{D}$: $\mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$

- $\forall d \in \mathcal{D}$: $1_\mathcal{D} \times_\mathcal{D} d = d = d \times_\mathcal{D} 1_\mathcal{D}$
- $\forall d \in \mathcal{D}$: $0_\mathcal{D} \times_\mathcal{D} d = 0_\mathcal{D} = d \times_\mathcal{D} 0_\mathcal{D}$
- $\forall d_1 = ((\overline{A}, w\overline{A'}\overline{\alpha}), \Theta, \sigma) \in \mathcal{D}$, $d_2 = ((\overline{B}, \overline{\beta}), \Omega, \rho) \in \mathcal{D}$: let $k_\Theta = \max(\mathrm{Dom}(\Theta))$, $k_\Omega = \max(\mathrm{Dom}(\Omega))$. Then

$$d_1 \times_\mathcal{D} d_2 = \begin{cases} \text{undefined} & \text{if } \overline{A'} \neq \overline{B} \text{ (context-free mismatch)} \\ 0_\mathcal{D} & \text{if } \Theta(1) \sqcap \Omega(0) = 0_\mathcal{T} \text{ (unification failure)} \\ ((\overline{A}, w\overline{\beta}\overline{\alpha}), \Theta \cup \Omega) & \text{otherwise (the product succeeds)} \end{cases}$$

where $\Theta \cup \Omega$: $\mathbb{N} \rightarrow \mathcal{T}^*$ s.t. $\mathrm{Dom}(\Theta \cup \Omega) = \{0, \cdots, k_\Theta - 1 + k_\Omega\}$ and

$$\Theta \cup \Omega(v) = \begin{cases} \Theta(1) \sqcap \Omega(0) & \text{if } v = 0 \text{ (unification at grafting point)} \\ \Omega(v) & \text{if } 0 < v \leq k_\Omega \\ \Theta(v - k_\Omega + 1) & \text{if } v > k_\Omega \end{cases}$$

Note that this operation is not commutative, and *partially* associative: take three elements d_1, d_2, d_3 of $\mathcal{H}$ s.t. their respective first component is $A \rightarrow \alpha B \beta C \gamma$, $B \rightarrow \delta$, $C \rightarrow \omega$. Then $(d_1 \times_\mathcal{H} d_2) \times_\mathcal{H} d_3$ is not necessarily undefined whereas $d_2 \times_\mathcal{D} d_3$ is.

The product $\times_\mathcal{D}$ is extended as a function $\mathcal{H} \times \mathcal{H} \rightarrow \mathcal{H}$ being the natural extension of $\times_\mathcal{D}$ in the usual (distributive) way. Then $1_\mathcal{H} = \{1_\mathcal{D}\}$ and $0_\mathcal{H} = \emptyset$. Finally, $(\mathcal{H}, +_\mathcal{H}, \times_\mathcal{H}, 0_\mathcal{H}, 1_\mathcal{H})$ is a partial semiring, because of $\times_\mathcal{H}$ partial associativity.

The decoration mapping

Consider the CF-grammar G and a definite clause grammar DG.

$$\overline{\phi_\mathcal{H}}(\overline{A} \rightarrow \overline{\alpha}) = \{((\overline{A}, \overline{\alpha}), \Theta) \mid (A \rightarrow \alpha, \Theta) \in DG\}$$

5 Example

Notations are simplified to improve readability: clauses are represented in the usual way, without splitting the context-free backbone from the terms. The vectors of terms are noted between square brackets: e.g., $[c, f(X), g(a, f(Y))]$.

This example shows both how our algorithm works and how operations are performed in the definite clause semiring. The following grammar DG recognizes trivial sentences, verifying that the number of the subject matches the number of the verb. We have $\mathcal{F}onc = \{sg, pl\}$, both functions are constants so $\mathcal{T}$ is reduced to $\mathcal{F}onc$ and $\mathcal{V}ar$.

$$\begin{array}{rcl}
S & \rightarrow & NP[Nb]VP[Nb] \\
NP[Nb_1] & \rightarrow & DetN[Nb_1] \\
NP[Nb_2] & \rightarrow & Pron[Nb_2] \\
VP[Nb_3] & \rightarrow & V[Nb_3] \\
VP[Nb_4] & \rightarrow & V[Nb_4]NP[X] \\
Det & \rightarrow & the \\
N[sg] & \rightarrow & man \mid apple \\
N[pl] & \rightarrow & men \mid apples \\
V[sg] & \rightarrow & eats \mid sings \\
V[pl] & \rightarrow & eat \mid sing \\
Pron[pl] & \rightarrow & you
\end{array}$$

It corresponds to the context-free grammar G:

$$\begin{array}{lll}
S \rightarrow NPVP & NP \rightarrow DetN \mid Pron & VP \rightarrow VNP \mid V \\
Det \rightarrow the & N \rightarrow man \mid men \mid apple \mid apples & V \rightarrow eats \mid sings \mid eat \mid sing
\end{array}$$

with the decoration mapping ϕ:

$$\begin{array}{rcl}
\phi(S \rightarrow NPVP) & = & \{ (S, NP[Z]VP[Z]) \} \\
\phi(NP \rightarrow DetN) & = & \{ (NP[X_2], DetN[X_2]) \} \\
\phi(NP \rightarrow Pron) & = & \{ (NP[Y_2], Pron[Y_2]) \} \\
\phi(VP \rightarrow VNP) & = & \{ (VP[X_3], V[X_3]NP[Y]) \} \\
\phi(VP \rightarrow V) & = & \{ (VP[Z_3], V[Z_3]) \} \\
\phi(Det \rightarrow the) & = & \{ (Det, the) \} \\
\phi(N \rightarrow man) & = & \{ (N[sg], man) \} \\
\phi(N \rightarrow men) & = & \{ (N[pl], men) \} \\
\phi(N \rightarrow apple) & = & \{ (N[sg], apple) \} \\
\phi(N \rightarrow apples) & = & \{ (N[pl], apples) \} \\
\phi(V \rightarrow eats) & = & \{ (V[sg], eats) \} \\
\phi(V \rightarrow eat) & = & \{ (V[pl], eat) \} \\
\phi(V \rightarrow sing) & = & \{ (V[pl], sing) \} \\
\phi(V \rightarrow sings) & = & \{ (V[sg], sings) \}
\end{array}$$

We parse the sentence "the man eats the apples".

The decorated items $<r, i, j, \mathbf{R}>$ computed by the algorithm are given below, ranked with respect to the index (level) j.

Level 0

$$\begin{array}{rll}
<S' \rightarrow \bullet S\$, & 0, 0, & \langle\rangle> \\
<S \rightarrow \bullet NPVP, & 0, 0, & \langle\rangle> \\
<NP \rightarrow \bullet DetN, & 0, 0, & \langle\rangle> \\
<Det \rightarrow \bullet the, & 0, 0, & \langle\rangle>
\end{array}$$

Level 1

$$\begin{array}{rll} <Det{\rightarrow}the\bullet, & 0,1, & \{\,(S,\ Det\ N[Z]\ VP[Z])\,\}> \\ <NP{\rightarrow}Det^{0,1}\bullet N, & 0,1, & \langle\rangle\diamond\langle\phi(Det^{0,1})\rangle> \\ <N{\rightarrow}\bullet man, & 1,1, & \langle\rangle> \\ <N{\rightarrow}\bullet apple, & 1,1, & \langle\rangle> \\ <N{\rightarrow}\bullet men, & 1,1, & \langle\rangle> \\ <N{\rightarrow}\bullet apples, & 1,1, & \langle\rangle> \end{array}$$

Recognition symbol:

$$\begin{array}{rcl} \phi(Det^{0,1}) & = & \overline{\phi}(Det^{0,1}{\rightarrow}the) \\ & = & \{\,(Det^{0,1}, the)\,\} \end{array}$$

Level 2

$$\begin{array}{rll} <N{\rightarrow}man\bullet, & 1,2, & \{\,(S,\ the\ man\ VP[sg])\,\}> \\ <NP{\rightarrow}Det^{0,1}N^{1,2}\bullet, & 0,2, & \langle\phi(Det^{0,1})\rangle\diamond\langle\phi(N^{1,2})\rangle> \\ <S{\rightarrow}NP^{0,2}\bullet VP, & 0,2, & \langle\rangle\diamond\langle\phi(NP^{0,2})\rangle> \\ <VP{\rightarrow}\bullet VNP, & 2,2, & \langle\rangle> \\ <V{\rightarrow}\bullet eats, & 2,2, & \langle\rangle> \\ <V{\rightarrow}\bullet sings, & 2,2, & \langle\rangle> \\ <V{\rightarrow}\bullet eat, & 2,2, & \langle\rangle> \\ <V{\rightarrow}\bullet sing, & 2,2, & \langle\rangle> \end{array}$$

Recognition symbols:

$$\begin{array}{rcl} \phi(N^{1,2}) & = & \overline{\phi}(N^{1,2}{\rightarrow}man) \\ & = & \{\,(N^{1,2}[sg], man)\,\} \\ \phi(NP^{0,2}) & = & \overline{\phi}(NP^{0,2}{\rightarrow}Det^{0,1}N^{1,2}){\times_{\mathcal{H}}}\overline{\phi}(Det^{0,1}){\times_{\mathcal{H}}}\overline{\phi}(N^{1,2}) \\ & = & \{\,(NP^{0,2}[X_1], Det^{0,1}N^{1,2}[X_1])\,\}{\times_{\mathcal{H}}}\{\,(Det^{0,1}, the)\,\}{\times_{\mathcal{H}}}\overline{\phi}(N^{1,2}) \\ & = & \{\,(NP^{0,2}[X_1],\ the\ N^{1,2}[X_1])\,\}{\times_{\mathcal{H}}}\{\,(N^{1,2}[sg], man)\,\} \\ & = & \{\,(NP^{0,2}[sg],\ the\ man)\,\} \end{array}$$

As a consequence, the 2 last items disappear while computing their prediction decoration: the unification of sg and pl fails.

Level 3

$$\begin{array}{rll} <V{\rightarrow}eats\bullet, & 2,3, & \{\,(S,\ the\ man\ eatNP[Y])\,\}> \\ <VP{\rightarrow}V^{2,3}\bullet NP, & 2,3, & \langle\rangle\diamond\langle\phi(V^{2,3})\rangle> \\ <NP{\rightarrow}\bullet Det\ N, & 3,3, & \langle\rangle> \\ <Det{\rightarrow}\bullet the, & 3,3, & \langle\rangle> \end{array}$$

Recognition symbol:

$$\begin{array}{rcl} \phi(V^{2,3}) & = & \overline{\phi}(V^{2,3}{\rightarrow}eats) \\ & = & \{\,(V^{2,3}[sg], eats)\,\} \end{array}$$

Level 4

$$\begin{array}{rll} <Det{\rightarrow}the\bullet, & 3,4, & \{\,(S,\ the\ man\ eats\ the\ N[Y])\,\}> \\ <NP{\rightarrow}Det^{3,4}\bullet N, & 3,4, & \langle\rangle\diamond\langle\phi(Det^{3,4})\rangle> \\ <N{\rightarrow}\bullet man, & 4,4, & \langle\rangle> \\ <N{\rightarrow}\bullet apple, & 4,4, & \langle\rangle> \\ <N{\rightarrow}\bullet men, & 4,4, & \langle\rangle> \\ <N{\rightarrow}\bullet apples, & 4,4, & \langle\rangle> \end{array}$$

Recognition symbol:

$$\begin{array}{rcl} \phi(Det^{3,4}) & = & \overline{\phi}(Det^{3,4}{\rightarrow}the) \\ & = & \{\,(Det^{3,4}, the)\,\} \end{array}$$

Level 5

$$\begin{array}{lll} <N \rightarrow apples\bullet, & 4,5, & \{(S, the\ man\ eats\ the\ apples)\}> \\ <NP \rightarrow Det^{3,4}N^{4,5}\bullet, & 3,5, & \langle\phi(Det^{3,4})\rangle\diamond\langle\phi(N^{4,5})\rangle> \\ <VP \rightarrow V^{2,3}NP^{3,5}\bullet, & 2,5, & \langle\phi(V^{2,3})\rangle\diamond\langle\phi(NP^{3,5})\rangle> \\ <S \rightarrow NP^{0,2}VP^{2,5}\bullet, & 0,5, & \langle\phi(NP^{0,2})\rangle\diamond\langle\phi(VP^{2,5})\rangle> \end{array}$$

Recognition symbols:

$$\begin{array}{rcl} \phi(N^{4,5}) & = & \overline{\phi}(N^{4,5} \rightarrow apples) \\ & = & \{(N^{4,5}[pl], apples)\} \\ \phi(NP^{3,5}) & = & \overline{\phi}(NP^{3,5} \rightarrow Det^{3,4}N^{4,5}) \times_{\mathcal{H}} \overline{\phi}(Det^{3,4}) \times_{\mathcal{H}} \overline{\phi}(N^{4,5}) \\ & = & \{(NP^{3,5}[X_4], Det^{3,4}N^{4,5}[X_4])\} \times_{\mathcal{H}} \{(Det^{3,4}, the)\} \times_{\mathcal{H}} \overline{\phi}(N^{4,5}) \\ & = & \{(NP^{3,5}[X_4],\ the\ N^{4,5}[X_4])\} \times_{\mathcal{H}} \{(N^{4,5}[pl], apples)\} \\ & = & \{(NP^{3,5}[pl],\ the\ apples)\} \\ \phi(VP^{2,5}) & = & \overline{\phi}(VP^{2,5} \rightarrow V^{2,3}NP^{3,5}) \times_{\mathcal{H}} \phi(V^{2,3}) \times_{\mathcal{H}} \phi(NP^{3,5}) \end{array}$$

$$\overline{\phi}(VP^{2,5} \rightarrow V^{2,3}NP^{3,5}) = \{(VP^{2,5}[X_3], V^{2,3}[X_3]NP^{3,5}[Y])\}$$

$$\{(VP^{2,5}[X_3], V^{2,3}[X_3]NP^{3,5}[Y])\} \times_{\mathcal{H}} \{(V^{2,3}[sg], eats)\} = \{(VP^{2,5}[sg], eats\ NP^{3,5}[Y])\}$$

then

$$\begin{array}{rcl} \phi(VP^{2,5}) & = & \{(VP^{2,5}[sg], eats\ NP^{3,5}[Y])\} \times_{\mathcal{H}} \{(NP^{3,5}[pl],\ the\ apples) \\ & = & \{(VP^{2,5}[sg], eats\ the\ apples)\} \\ \phi(S^{0,5})) & = & \overline{\phi}(S^{0,5} \rightarrow NP^{0,2}VP^{2,5}) \times_{\mathcal{H}} \phi(NP^{0,2}) \times_{\mathcal{H}} \phi(VP^{2,5}) \end{array}$$

$$\overline{\phi}(S^{0,5} \rightarrow NP^{0,2}VP^{2,5}) = \{(S^{0,5}, NP^{0,2}[Z]VP^{2,5}[Z])\}$$

$$\{(S^{0,5}, NP^{0,2}[Z]\ VP^{2,5}[Z])\} \times_{\mathcal{H}} \{(NP^{0,2}[sg],\ the\ man)\} = \{(S^{0,5}, the\ man\ VP^{2,5}[sg])\}$$

then

$$\begin{array}{rcl} \phi(S^{0,5}) & = & \{(S^{0,5}, the\ man\ VP^{2,5}[sg])\} \times_{\mathcal{H}} \{(VP^{2,5}[sg], eats\ the\ apples)\} \\ & = & \{(S^{0,5}, the\ man\ eats\ the\ apples)\} \end{array}$$

One remarks that the last substitution is the identity. It can be any substitution because there are no free variables. If the axiom was associated with the variable Z then the substitution would be $(Z \mapsto sg)$.

6 Conclusion

The presentation of first-order terms in a semi-lattice frame suggests the extension to feature structures, especially for hierarchies of types which are distributive lattices (see Carpenter [5]) because they are equivalent to a semiring structure.

In case of cyclic grammars and if limit computations cannot be provided to solve the encountered equation systems, one can think of approximating the solution with the help of abstract interpretation (Cousot and Cousot [7]): instead of $(\mathcal{A}, +_{\mathcal{A}}, \times_{\mathcal{A}}, 0_{\mathcal{A}}, 1_{\mathcal{A}})$, one can consider $(\mathcal{A}^\sharp, +_{\mathcal{A}^\sharp}, \times_{\mathcal{A}^\sharp}, 0_{\mathcal{A}^\sharp}, 1_{\mathcal{A}^\sharp})$ with an abstraction function Θ: $\mathcal{A} \rightarrow \mathcal{A}^\sharp$ which must be a semiring morphism. Naturally, $\mathcal{A}^\sharp$ must be chosen so that limit computations are possible in $\mathcal{A}^\sharp$. Recall that cyclic grammars are usually proscribed by formalisms from computational linguistics, however introducing cycles may be an elegant manner to cope with ill-formed sentences, and then, an approximation just injures the error recovery.

Earley's algorithm is essentially bottom-up (the final information is synthesized) but the search space is reduced by a top-down phase (prediction), and we claim that each syntactic operation can be performed (by morphism) in the decoration domain. In Left Corner, these predictions are statically performed. This suggests that prediction computations could also be performed in the decoration domain, allowing to prune some wrong parses earlier, and possibly statically.

References

[1] M.A. Arbib and E.G. Manes, Partially-additive categories and flow-diagram semantics, *Journal of Algebra* **62** (1980) 203–227.

[2] J.K. Baker, Trainable grammars for speach recognition, *Speech Comm. Papers for 97th Meet. of the Acoustical Society of America, J.J. Wolf and D.H. Klatt Eds.* (1979) 547–550.

[3] R.E. Bellman, *Dynamic Programming* (1957), Princeton Univ. Press.

[4] S. Billot, B. Lang, The structure of shared forest in ambiguous parsing, *Proc. 27th Ann. Meet. of the Assoc. for Comput. Ling.* (1989) .

[5] B. Carpenter, *The Logic of Typed Features Structures* (1992), Cambridge Univ. Press (Cambridge Tracts in Theoretical Computer Science), Cambridge, England.

[6] N. Chomsky, M.P. Schützenberger, The algebraic theory of context-free languages, in: P. Braffort and D. Hirschberg, *Computer Programming and Formal Systems* (1963) 118–161, North-Holland, Amsterdam.

[7] P. Cousot, R. Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *Proc. 4th ACM Symp. on Principles of Programming Languages* (1977) 238–252.

[8] J. Earley, An efficient context-free parsing algorithm, *Comm. Assoc. Comp. Mach.* **13** (1970) 94–102.

[9] R. Kaplan, J. Bresnan, Lexical-functional grammar: a formal system for grammatical representation, *J. Bresnan Ed., The mental Representation of Grammatical Relations, MIT Press, Cambridge, MA* (1982) 173–281.

[10] T. Kasami, *An Efficient Recognition and Syntax Analysis Algorithm for Context-Free Languages* (1965), Scientific Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

[11] M. Kay, Functional Unification Grammars: a formalism for machine translation, *Proc. 10th Int. Conf. on Comput. Ling., Stanford* (1984) 75–78.

[12] B. Lang, Deterministic techniques for efficient non-deterministic parsers, in J. Loeckx (ed.), *Proc. 2nd Colloquium on Automata, Languages and Programming, Lect. Notes in Comp. Sci.* **14** (1974) 255–269, Springer, Berlin, etc.

[13] F.C.N. Pereira, D.H.D. Warren, Definite Clause Grammars for language analysis – survey of the formalism and comparision with augmented transition networks, *Artificial Intel.* **13:3** (1980) 231–278.

[14] S.M. Shieber, The design of a computer language for linguistic information, *Proc. 10th Int. Conf. on Comput. Ling., Stanford* (1984) 362–366.

[15] F. Tendeau, Stochastic parse-tree recognition by a pushdown automaton, *Proc. 4th International Workshop on Parsing Technology,* Prague / Karlovy Vary (1995) 234–249.

[16] F. Tendeau, Computing abstract decorations of parse forests using dynamic programming and algebraic power series, *To appear in Theor. Comp. Sci.* (1997) .

[17] D.H. Younger, Recognition and parsing of context-free languages in time n^3, *Inform. Contr.* **10** (1967) 189–208.