IRCS Workshop Linguistic Databases 2001
Steven Bird, Peter Buneman, and Mark Liberman (eds.)

A Regular Query for Context-Sensitive Relations

Uwe Mönnich, Frank Morawietz, and Stephan Kepser

Seminar für Sprachwissenschaft, Universität Tübingen, Germany
{um,frank,kepser}@sfs.uni-tuebingen.de

Abstract

One of the fundamental problems when defining a query language for databases consists in finding a balance between the desiderata of a sufficiently large expressive power on the one hand and an adequate computability of queries on the other. This problem occurs of course also with linguistic treebanks, the prototype of non-relational semistructured databases. There are many linguistic phenomena which can be adequately annotated by using trees, and for which there exists a powerful yet decidable query language, namely monadic second order logic (MSO). But on the other hand there exist linguistic phenomena such as cross-serial dependencies in Swiss German which cannot be described with context-free means and for which therefore MSO is not expressive enough as a query language. Instead of going over to a more expressive query language and losing decidability on the way we propose to employ a two-level approach, which has proven successful in handling mildly context-sensitive phenomena before.

The two-level approach consists of a lifting step in which the (grammar of) the treebank and the MSO query is lifted to the free Lawvere-algebra where a coding of mildly context-sensitive relations within the realm of MSO logic is possible. This step allows to filter out all undesirable query results and retrieve only the relevant ones. In the second step, the returned answer trees are retranslated into the original trees of the treebank. By using techniques from automata theory in both steps we can ensure that the query language remains decidable.

1 Introduction

The present paper tries to exploit the recent convergence of studies in the fields of database technology and computational linguistics. Within the field of database technology the challenges resulting from the new data model of so-called semistructured or self-describing data have led to a wide spectrum of research activities that are concerned with the problems of data integration, Web technology, and the design of new query languages. Within linguistics where the semistructured data models have a long tradition, the availability of powerful storage media has brought the divergence of different traditions of descriptive schemes and annotation systems into sharp focus. Complicating the task of integrating data types from diverse sources of linguistic documentation is the fact that natural languages exhibit a property that makes it impossible for them to be accommodated within the limits of core XML, the standard format for data exchange on the Web. As has been noticed since the beginning of formal language theory certain grammatical phenomena like morphological congruences (e.g., Bambara) and cross-serial dependencies between case markings (Swiss German) are outside the realm of context-free languages and need for their descriptive analysis a (limited) amount of contextual information. Due to this character of context-sensitivity that comes with a range of grammatical constructions, even monadic second-order logic (MSO), considered as a powerful query language, is too weak to capture these phenomena.

Taking our inspiration from universal algebra we regard grammatical categories as basic constants of a many-sorted algebra with a distinguished set of composition and projection symbols. Through the explicit introduction of these operation symbols it becomes possible to turn the data models of contemporary syntactic theories into a kind of labeled tree structures that can either be generated by regular tree grammars or are identifiable with collections of finite trees specifiable by formulae of MSO logic.

In the particular case of the verbal complex of Swiss German mentioned above it is easy to describe in MSO terms the two verbal and nominal clusters, respectively. What is problematic from

the point of view of regularity is the set of fixed syntactic and semantic relations between the verbal elements and their case-marked arguments. In other words, an MSO specification of these bipartite structures would return – regarding the MSO specification as a yes/no query – structures that do not satisfy the particular set of cross-serial dependencies characteristic of the instance of context-sensitivity under discussion. Despite this lack of expressive power of the chosen query language the algebraic approach adumbrated above is remarkably effective in filtering out the syntactic "noise" from the query result. Since the explicit algebraic structures are elements of a regular family of trees it is again easy to produce an MSO formula that characterizes exactly these cross-serial dependencies among the explicit structures that were out of the reach of the query language, on the intended linguistic level.

It takes then only linear time to project those explicit structures that comply with the set of syntactic and semantic relations onto the data model where the original query was formulated. It turns out that this method of regularizing queries of context-sensitive structures can be adopted to all grammatical phenomena that fall within the reach of current linguistic theories (Kolb et al., 2000a; Kolb et al., 2000b; Michaelis et al., 2001; Morawietz and Mönnich, 2001).

The idea of using composition and projection as operations on trees is a special case of a general approach developed by Mezei and Wright (1967) in which regular tree languages denote subsets of arbitrary algebras. Of particular relevance for the present application to context-sensitive query problems have been the contributions of Courcelle (1990) to the interaction between graph operations and MSO. Courcelle has devised a primitive set of operations such that any finite graph can be considered as the value of a term that is constructed from (symbols for) these primitive operations. These operations exhibit a feature that relates the logical description of graph properties to its translation in terms of the corresponding tree expressions in a very strong sense: For every MSO formula φ expressing a graph property there is a "lifted" MSO formula φ^L over trees that is satisfied by exactly those structures that denote the sets of graphs modeled by the original formula.

Courcelle's ideas have been adapted to the context of semistructured data. Buneman, Fernandez, and Suciu (Buneman et al., 2000) define within the framework of their data model nine constructors that constitute a directed extension of Courcelle's proposal.

Since our data model is firmly entrenched in the linguistic tradition where trees with a limited amount of cross-serial dependencies play a prominent role, we are able to restrict our attention to two constructors denoting the familiar operations of composition and projection. This advantage which is provided by the considerable reduction of the set of primitive constructors does not lead directly to a family of canonical expressions that suits our purposes. As was noted above a query whose expressive power does not go beyond MSO is too weak to specify an answer set displaying the sort of dependencies so characteristic of natural language structure. It is therefore necessary to translate the result of the first step of the query into a family of trees that can be checked by a suitable constraint formula for the intended dependencies. We will show that this translation of the first step is tightly controlled by the constraint formula. Using the constraint formula as a template for the translation process allows us to avoid the problem of context-sensitive parsing without being forced to consider the unbounded set of "lifted" expressions denoting the same tree.

2 Preliminaries

Recall that for a given set of sorts $\mathcal{S}$, a *many-sorted alphabet* Σ *(over* $\mathcal{S}$*)* is an indexed family $\langle \Sigma_{w,s} \mid w \in \mathcal{S}^*, s \in \mathcal{S} \rangle$ of disjoint sets. A symbol $\sigma \in \Sigma_{w,s}$ is an *operator of type* $\langle w,s \rangle$, *arity* w, *sort* s and *rank* $|w|$. The elements of $\Sigma_{\epsilon,s}$ are also called constants (of sort s).

In case $\mathcal{S}$ is a singleton set $\{s\}$, i.e., in case Σ is a *single-sorted or ranked alphabet (over sort s)*, we usually write Σ_n to denote the (unique) set of operators of rank $n \in \mathbb{N}$.

In later sections of the paper we will mainly use the single-sorted case of alphabets. We will indicate the need for many-sorted alphabets where necessary.

For such a ranked alphabet Σ, we denote by $T(\Sigma)$ the set of *trees* over Σ. $T(\Sigma)$ is inductively defined with base case $\Sigma_0 \subseteq T(\Sigma)$ and recursive step $f(t_1,\ldots,t_n) \in T(\Sigma)$ if $f \in \Sigma_n$ and $t_i \in T(\Sigma)$ for $i = 1,\ldots,n$.

We fix an indexed set $X = \{x_1, x_2, \ldots\}$ of *variables* and denote by X_n the subset $\{x_1,\ldots,x_n\}$. Variables are considered to be constants, i.e., opera-

tors of rank 0. For a ranked alphabet Σ the family $T(\Sigma,X)$ is defined to be $T(\Sigma(X))$, where $\Sigma(X)$ is the ranked alphabet with $\Sigma(X)_0 = \Sigma_0 \cup X$ and $\Sigma(X)_n = \Sigma_n$ for every $n \neq 0$. A subset L of $T(\Sigma)$ is called a *tree language* over Σ.

Having described the tree terms, it remains to specify the central notion of an algebra and to give a precise definition of the way in which the operator symbols induce operations on an algebra.

Suppose that Σ is a ranked alphabet. A Σ-*algebra* $\mathbb{A}$ is a pair $\mathbb{A} = (A, (f_\mathbb{A})_{f\in\Sigma})$ where the set A is the *carrier* of the algebra and for each operator $f \in \Sigma_n$, $f_\mathbb{A} : A^n \to A$ is an operation of arity n on A.

Different algebras, defined over the same operator domain, are related to each other if there exists a mapping between their carriers that is compatible with the basic structural operations.

A Σ-*homomorphism* of Σ-algebras $h : \mathbb{A} \longrightarrow \mathbb{B}$ is a function $h : A \longrightarrow B$, such that $h(f_\mathbb{A}(a_1,\ldots,a_n)) = f_\mathbb{B}(h(a_1),\ldots,h(a_n))$ for every operator f of rank n and for every n-tuple $(a_1,\ldots,a_n) \in A^n$.

The set of trees $T(\Sigma,X)$ can be made into a Σ-algebra $\mathbb{T}(\Sigma,X)$ by defining the operations in the following way. For every f in Σ_n, for every $(t_1,\ldots,t_n)$ in $T(\Sigma,X)^n$: $f_{\mathbb{T}(\Sigma,X)}(t_1,\ldots,t_n) = f(t_1,\ldots,t_n)$.

Every variable-free tree $t \in T(\Sigma)$ has a value in every Σ-algebra $\mathbb{A}$. It is the value at t of the unique homomorphism $h : \mathbb{T}(\Sigma) \to \mathbb{A}$.

The existence of a unique homomorphism from the Σ-algebra of trees into an arbitrary Σ-algebra $\mathbb{A}$ provides also the basis for the view that regards the elements of $T(\Sigma,X_n)$ as *derived operations*. Each tree $t \in T(\Sigma,X_n)$ induces an n-ary function $t_\mathbb{A} : A^n \to A$.

The meaning of this function $t_\mathbb{A}$ is defined in the following way. For every $(a_1,\ldots,a_n) \in A^n$: $t_\mathbb{A}(a_1,\ldots,a_n) = \hat{a}(t)$, where $\hat{a} : \mathbb{T}(\Sigma,X_n) \to \mathbb{A}$ is the unique homomorphism with $\hat{a}(x_i) = a_i$.

In the particular case where $\mathbb{A}$ is the Σ-algebra $\mathbb{T}(\Sigma,X_m)$ of trees over Σ that contain at most variables from $X_m = \{x_1,\ldots,x_m\}$ at their leaves the unique homomorphism extending the assignment of a tree $t_i \in T(\Sigma,X_m)$ to the variable x_i in X_n acts as a substitution $t_{\mathbb{T}(\Sigma,X_m)}(t_1,\ldots,t_n) = t[t_1,\ldots,t_n]$ where the right hand side indicates the result of substituting t_i for x_i in t.

3 Queries

The aim of the paper is to provide a way to query mildly context-sensitive relations in a treebank. A treebank in our sense is just a set of labeled trees. We use monadic second order logic (MSO) as query language. A query is therefore an MSO-sentence, and the answer to a query is the set of all those trees in the treebank for which this formula is true. MSO is quite a powerful query language, indeed every regular set of trees can be characterized by an MSO-formula. On the other hand, MSO is known to be decidable on trees (Thatcher and Wright, 1968). This makes MSO an appealing query language. But MSO is restricted to context-free phenomena. A linguist, on the other hand, may be interested in certain non-context-free relations. An example of such a phenomenon are cross-serial dependencies in Swiss-German. In the sentence fragment

(... wil) mer de maa em chind lönd hälffe schwüme

a block of accusative objects (*de maa*) is followed by a block of dative objects (*em chind*). Then follow the verbs taking complements in the accusative (*lönd*) followed by the verbs taking complements in the dative (*hälffe schwüme)*. In our case there is only a single accusative and dative object, but in principle there could be several of them. Examples of this type are discussed at length by Shieber (1985). A language describing such phenomena is of the form $a^n b^m c^n d^m$, which is known to be non-context-free. If a linguist wishes to search a treebank for these mildly context-sensitive relations, MSO is insufficient as a query language. The straight-forward solution of expanding the expressive power of the query language will almost inevitably end in an undecidable language. Therefore we propose a solution of a different kind, namely a two-level approach.

In this approach, MSO is still used as the query language. Consequently a query, if posed correctly, will not only return the desired trees exhibiting the context-sensitive relation, but also certain others. To filter out these undesired trees we use a grammar. That is to say the linguist has to specify a grammar that generates the trees he is interested in. This requires the grammar formalism to be more expressive than context-free (string) grammars. The largest class of grammars suitable for our approach are context-free tree grammars.

Definition 1 [Context-Free Tree Grammar] Let $\mathcal{S}$

be a singleton set of sorts. Then a *context-free tree grammar (CFTG)* for $\mathcal{S}$ is a 5-tuple $\Gamma = \langle \Sigma, \mathsf{F}, S, \mathsf{X}, \mathsf{P} \rangle$, where Σ and F are ranked alphabets of *inoperatives* and *operatives* over $\mathcal{S}$, respectively. $S \in \mathsf{F}$ is the start symbol, X is a countable set of variables, and P is a set of productions. Each $p \in \mathsf{P}$ is of the form $F(x_1, \cdots, x_n) \longrightarrow t$ for some $n \in \mathbb{N}$, where $F \in \mathsf{F}_n$, $x_1, \cdots, x_n \in \mathsf{X}$, and $t \in T(\Sigma \cup \mathsf{F}, \{x_1, \cdots, x_n\})$.

Intuitively, an application of a rule of the form $F(x_1, \ldots, x_n) \rightarrow t$ "rewrites" a tree rooted in F as the tree t with its respective variables substituted by F's daughters.

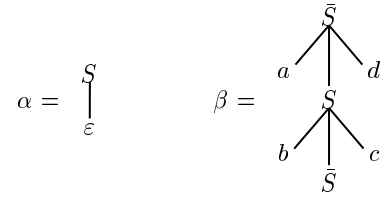
A CFTG $\Gamma = \langle \Sigma, \mathsf{F}, S, \mathsf{X}, \mathsf{P} \rangle$ with $\mathsf{F}_n = \emptyset$ for $n \neq 0$ is called a *regular tree grammar (RTG)*. Since RTGs always just substitute some tree for a leaf-node, it is easy to see that they can only generate recognizable sets of trees, *a forteriori* context-free string languages (Mezei and Wright, 1967). If F_n is nonempty for some $n \neq 0$, that is, if we allow the *operatives* to be parameterized by variables, however, the situation changes. CFTGs in general are capable of generating sets of structures, the *yields* of which belong to the subclass of context-sensitive languages known as the *indexed* languages.

An example of such a grammar formalism used in linguistics that can express certain mildly context-sensitive relations is Tree Adjoining Grammar (Joshi and Schabes, 1997; Vijay-Shanker and Weir, 1994). TAG is known to be weakly equivalent to monadic context-free tree grammars, as was shown independently by Mönnich (1997) and Fujiyoshi and Kasai (2000). Another example are minimalist grammars in the sense of Stabler (Stabler, 2001), which are equivalent to certain types of context-free tree grammars (Michaelis et al., 2001).

In order to be able later on to find the desired context-sensitive relations, it is necessary that the actual grammar given is such that it generates all those trees which embody non-context-free relations. It need not be a grammar for a single query. But it should usually not be a general grammar for the whole treebank either, because it will be used as a filter.

Let us illustrate the above by means of an example. We simplify the language for Swiss-German by setting $n = m$, which results in the context-sensitive language $a^n b^n c^n d^n$. A Tree Adjoining Grammar generating this language is given below.

Example 2 Let $G_{TAG} = \langle \{S\}, \{a,b,c,d\}, S, \{\alpha\}, \{\beta\} \rangle$ be a TAG. The only initial tree α and the only auxiliary tree β are given as follows:



A derivation yielding *aabbccdd* has only two steps, both adjoin the auxiliary tree in the only possible position, see Figure 1.

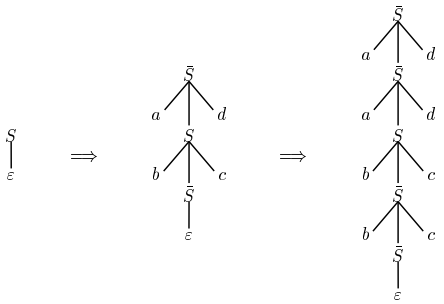


Figure 1: An example derivation of the TAG G_{TAG} given in Example 2

The corresponding monadic context-free tree grammar for the TAG grammar looks like this:

Example 3 Consider the CFTG $\Gamma_{TAG} = \langle \{a,b,c,d, \varepsilon, S_t, S_t^0\}, \{S, S', \overline{S}_1, \overline{S}_2, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}, S', \{x\}, \mathsf{P} \rangle$ resulting from a translation of the TAG G_{TAG} with P given as follows

$$
\begin{array}{ll}
S' \longrightarrow S(\varepsilon) & \overline{a} \longrightarrow a \\
S(x) \longrightarrow \overline{S}_1(S(\overline{S}_2(x))) & \overline{b} \longrightarrow b \\
S(x) \longrightarrow S_t^0(x) & \overline{c} \longrightarrow c \\
\overline{S}_1(x) \longrightarrow S_t(\overline{a}, x, \overline{d}) & \overline{d} \longrightarrow d \\
\overline{S}_2(x) \longrightarrow S_t(\overline{b}, x, \overline{c}) &
\end{array}
$$

A derivation of the string *aabbccdd* is shown in Figure 2. The example derivation is somewhat longer than the one given for the almost identical TAG grammar generating the same language. This is due to the fact that we need nonterminals to introduce each branching of the resulting tree separately. In the first step, we simply rewrite the start symbol. In the second one, the symbol S with the term $\overline{S}_1(S(\overline{S}_2(x)))$ where the (degenerate) tree ε is simply appended to the only argument position x of $\overline{S}_2$. This step is repeated before we terminate with an

application of the rule rewriting S as S_t^0. We simplified the presentation in the sense that in this last step we also applied the rules for the "barred" operatives, i.e., we replaced each $\overline{S}_i$, $i \in \{1,2\}$ with the corresponding term and each $\overline{s} \in \{\overline{a},\overline{b},\overline{c},\overline{d}\}$ with s.

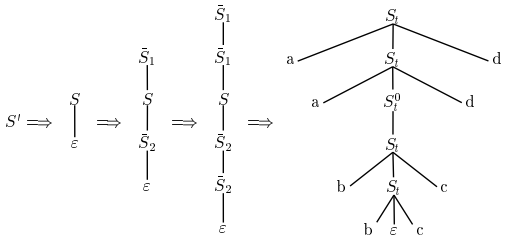


Figure 2: An example derivation of the CFTG Γ_{TAG} from Example 3

4 Lifting

The MSO-query on the treebank produced a set of candidate trees out of which we would like to filter the desired trees by means of the grammar. In principle, this could be done by starting a context-sensitive parsing process on the set of candidate trees. Rather than choosing this inefficient approach, we show in this section that the desired trees can be filtered out by purely regular means. To do so, we LIFT both the grammar and the set of candidate trees. The intuition here is that the basic assumptions about the operations of a tree grammar, namely tree substitution and argument insertion, are made explicit. We make them visible by inserting the "control" information which allows us to code the resulting structures with regular means, i.e., regular tree grammars or finite-state tree automata and therefore with MSO logic. The intuition behind the LIFTing process is that each term compactly encodes information such as composition and concatenation.

In the following, we will briefly describe LIFTing on a more formal level. All technical details, in particular concerning many-sorted signatures, can be found in a paper by Mönnich (1999). Any *context-free* tree grammar Γ for a singleton set of sorts $\mathcal{S}$ can be transformed into a *regular* tree grammar Γ^L for the set of sorts $\mathcal{S}^*$, which characterizes a (necessarily recognizable) set of trees encoding the instructions necessary to convert them by means of a unique homomorphism h into the ones the original grammar generates (Maibaum, 1974). This "LIFTing" is achieved by constructing for a given single-sorted signature Σ a new, derived alphabet (an $\mathcal{S}^*$-sorted signature) Σ^L, and by translating the terms over the original signature into terms of the derived one via a primitive recursive procedure. The LIFT-operation takes a term in $T(\Sigma,X_k)$ and transforms it into one in $T(\Sigma^L,k)$. Intuitively, the LIFTing eliminates variables and composes functions with their arguments explicitly, e.g., a term $f(a,b) = f(x_1,x_2) \circ (a,b)$ is lifted to the term $c(c(f,\pi_1,\pi_2),a,b)$. The old function symbol f now becomes a constant, the variables are replaced with appropriate projection symbols and the only remaining non-nullary alphabet symbols are the explicit composition symbols c.

Definition 4 [LIFT] Let Σ be a ranked alphabet of sort $\mathcal{S}$ and $X_k = \{x_1,\ldots,x_k\}$, $k \in \mathbb{N}$, a finite set of variables. The *derived* $\mathcal{S}^*$-sorted alphabet Σ^L is defined as follows: For each $n \geq 0$, $\Sigma'_{\varepsilon,n} = \{f' \mid f \in \Sigma_n\}$ is a new set of symbols of type $\langle \varepsilon, n\rangle$; for each $n \geq 1$ and each $i, 1 \leq i \leq n$, π_i^n is a new symbol, the ith *projection symbol* of type $\langle \varepsilon, n\rangle$; for each $n,k \geq 0$ the new symbol $c_{(n,k)}$ is the (n,k)th *composition symbol* of type $\langle nk_1\cdots k_n, k\rangle$ with $k_1 = \cdots = k_n = k$.

$$
\begin{aligned}
\Sigma^L_{\varepsilon,0} &= \Sigma'_{\varepsilon,0} \\
\Sigma^L_{\varepsilon,n} &= \Sigma'_{\varepsilon,n} \cup \{\pi_i^n \mid 1 \leq i \leq n\} \text{ for } n \geq 1 \\
\Sigma^L_{nk_1\cdots k_n,k} &= \{c_{(n,k)}\} \text{ for } n,k \geq 0 \text{ and } k_i = k \text{ for } 1 \leq i \leq k \\
\Sigma^L_{w,s} &= \emptyset \text{ otherwise}
\end{aligned}
$$

For $k \geq 0$, $\text{LIFT}^\Sigma_k : T(\Sigma,X_k) \to T(\Sigma^L,k)$ is defined as follows:

$$
\begin{aligned}
\text{LIFT}^\Sigma_k(x_i) &= \pi_i^k \\
\text{LIFT}^\Sigma_k(f) &= c_{(0,k)}(f') \text{ for } f \in \Sigma_0 \\
\text{LIFT}^\Sigma_k(f(t_1,\ldots,t_n)) &= \\
& c_{(n,k)}(f', \text{LIFT}^\Sigma_k(t_1),\ldots,\text{LIFT}^\Sigma_k(t_n)) \\
& \text{for } n \geq 1, f \in \Sigma_n \text{ and } t_1,\ldots,t_n \in T(\Sigma,X_k)
\end{aligned}
$$

Note that this very general procedure allows the translation of any term over the original signature. The left hand side as well as the right hand side (RHS) of a rule of a CFTG $\Gamma = \langle \Sigma, F, X, S, P\rangle$ is just a term belonging to $T(\Sigma \cup F, X)$, but so is, e.g., any structure *generated* by Γ.

Further remarks on the observation that the result of LIFTing a CFTG is always an RTG can be also found in the paper by Mönnich (1999). To further

illustrate the techniques, we present the continuation of Example 3. Note that for better readability, we omit all the 0- and 1-place composition symbols.

Example 5 Let $\Gamma^L_{TAG} = \langle \{a,b,c,d,\varepsilon,S_t,S^0_t\}, \{S,S', \overline{S}_1,\overline{S}_2,\overline{a},\overline{b},\overline{c},\overline{d}\}, S', \mathrm{P}\rangle$ with P given as follows

$$
\begin{aligned}
S' &\longrightarrow c_{(1,0)}(S,\varepsilon)\\
S &\longrightarrow c_{(1,1)}(\overline{S}_1, c_{(1,1)}(S, c_{(1,1)}(\overline{S}_2,\pi^1_1)))\\
S &\longrightarrow c_{(1,1)}(S^0_t,\pi^1_1)\\
\overline{S}_1 &\longrightarrow c_{(3,1)}(S_t,a,\pi^1_1,d)\\
\overline{S}_2 &\longrightarrow c_{(3,1)}(S_t,b,\pi^1_1,c)
\end{aligned}
$$

Note that we now have only nullary operatives but extra composition and projection symbols.

For lifting the set of candidate trees, it is unfortunately not possible to directly apply the lifting definition above. Since the candidate trees have no variables, their lifts would have no projection symbols at all. For example, a candidate tree of the form $f(a,b)$ would be lifted to $c(f,a,b)$. On the other hand, almost all trees generated by the lifted grammar contain projection symbols. Consequently the lifted grammar cannot generate a single tree of the shape of the simple-minded lifted candidate trees and can therefore not be used to single out the correct, lifted trees from the lifted candidate trees. We therefore lift the candidate trees by generating for each tree a set of lifted trees with projection symbols. This set is finite and can be made quite small for each tree on the basis of the following assumptions. We assume the lifted grammar to be in Greibach normal form. If it is not, we can easily convert it thereinto. The fact that every rule application of the lifted grammar in Greibach normal form produces an inoperative symbol together with the fact that every symbol (terminal or inner) of the candidate tree will be an inoperative symbol after lifting gives a depth bound on the lifted tree.

An example of lifting a tree, namely the rightmost tree in Figure 2 can be found in Figure 3. The additional arcs in this figure will be explained in the next section.

The lifted grammar is now applied as the filter on the set of lifted candidate trees. Practically this can be done in the following way. Since the lifted grammar is regular, it can be represented by a tree automaton. This tree automaton representing the grammar is run on the set of lifted candidate trees accepting only those ones that are compatible with the grammar. Now we have the lifted solution set containing only the desired trees representing the context-sensitive relations.

5 Reconstruction

Unfortunately the trees of the solution set are – since lifted – not in the shape of trees in the treebank. In fact, they do not seem to have much in common with the structures linguists want to talk about, i.e., the ones in Figure 2. However, in some sense to be made operational, the LIFTed structures contain the intended structures. As mentioned before, there is a mapping h from these explicit structures onto structures interpreting the compositions (the c's) and the projections (the π's) the way the names we have given them suggest, *viz.* as compositions and projections, respectively, which are, in fact, exactly the intended structures.

On the denotational side, we can implement the mapping h with an MSO definable tree transduction (as defined in Courcelle (1997)) and on the operational side with both tree-walking automata (FSTWA, see (Bloem and Engelfriet, 1997)) and Macro Tree Transducer (MTT, see (Engelfriet and Maneth, 1999)) to transform the LIFTed structures into the intended ones. In this paper, we will focus on the logical transduction.

Let us restate our goal then: Rogers (1998) has shown the suitability of an MSO description language $L^2_{K,P}$ for linguistics which is based upon the primitive relations of immediate ($\lhd$), proper ($\lhd^+$) and reflexive ($\lhd^*$) dominance and proper precedence ($\prec$). We will show how to define these relations with an MSO transduction built upon finite-state tree-walking automata thereby implementing the unique homomorphism mapping the terms into elements of the corresponding context-free tree language, i.e., the trees linguists want to talk about.

Put differently, it should be possible to define a set of relations $R^I = \{\blacktriangleleft, \blacktriangleleft^+, \blacktriangleleft^*$ *(dominance)*, $\lhd$ *(precedence)*$, \ldots\}$ holding between the nodes of the explicit or LIFTed tree which carry a *"linguistic"* label L in such a way, that when interpreting $\blacktriangleleft^* \in R^I$ as a tree order on the set of *"linguistic"* nodes and $\lhd \in R^I$ as the precedence relation on the resulting structure, we have a "new" description language on the intended structures.

As mentioned before, we will use an MSO definable tree transduction to transform the LIFTed struc-

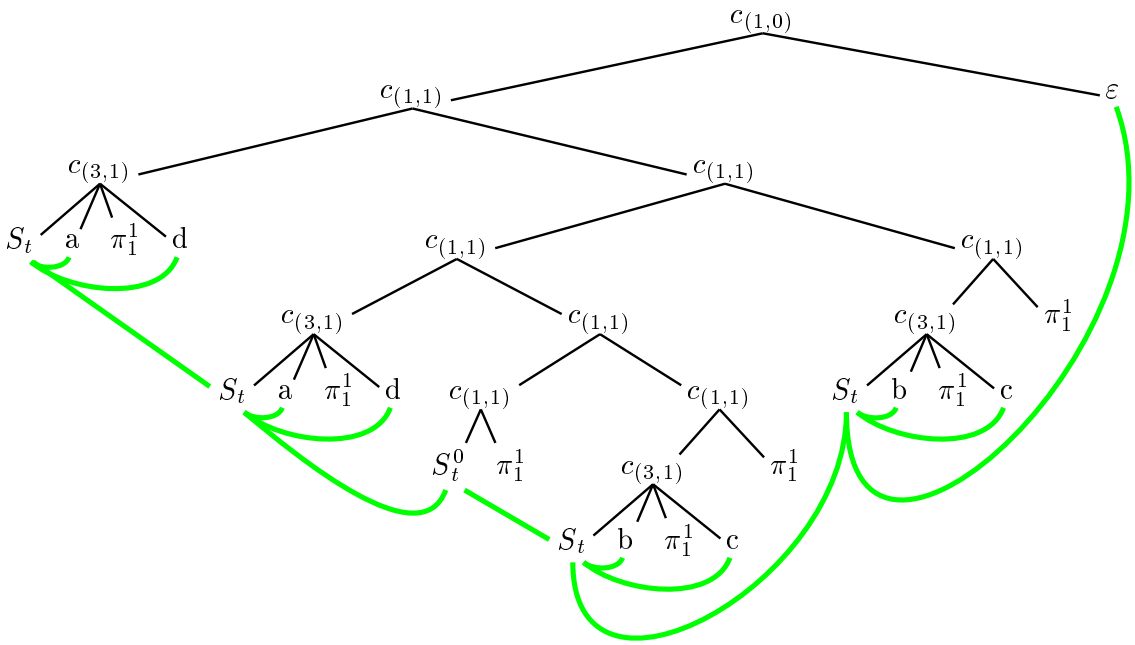


Figure 3: Intended relations on a LIFTed tree

tures into the intended ones. Let R be a finite set of relation symbols with the corresponding arity for each $r \in R$ given by $\rho(r)$. A relational structure $\mathcal{R} = \langle D_\mathcal{R}, (r_\mathcal{R})_{r\in\mathcal{R}} \rangle$ consists of the domain $D_\mathcal{R}$ and the $\rho(r)$-ary relations $r_\mathcal{R} \subseteq D_\mathcal{R}^{\rho(r)}$. We can code trees as relational structures by taking a tree domain as the domain $D_{\mathcal{R}_{w,A}}$ of the structure and defining *suc* as the corresponding tree order.

The classical technique of interpreting a relational structure within another one forms the basis for MSO transductions. Intuitively, the output tree is interpreted on the input tree. E.g., suppose that we want to transduce the input tree t_1 into the output tree t_2. The nodes of the output tree t_2 will be a subset of the nodes from t_1 specified with a unary MSO relation ranging over the nodes of t_1. The daughter relation will be specified with a binary MSO relation with free variables x and y ranging over the nodes from t_1.

Definition 6 [MSO transduction] Let R and Q be two finite sets of ranked relation symbols. A (non-copying) MSO transduction of a relational structure $\mathcal{R}$ (with set of relation symbols R) into another one $\mathcal{Q}$ (with set of relation symbols Q) is defined to be a tuple $(\varphi, \psi, (\theta_q)_{q\in Q})$ consisting of an MSO formula φ defining the domain of the transduction in $\mathcal{R}$, an MSO formula ψ defining the resulting domain of $\mathcal{Q}$, and a family of MSO formulas θ_q defining the new relations Q using only definable formulas from the "old" structure $\mathcal{R}$, i.e., for α a variable assignment,

$$D_\mathcal{Q} = \{d \in D_\mathcal{R} \mid (\mathcal{R}, d) \models \psi[\alpha]\}$$

and for each $q \in Q$
$q_\mathcal{Q} = \{(d_1, \ldots, d_n) \in D_\mathcal{Q}^n \mid (\mathcal{R}, d_1, \ldots, d_n) \models \theta_q[\alpha]\}$
where $n = \rho(q)$

Note that the transduction is only defined if φ holds.

The specific MSO transduction we need to transform the LIFTed structures into the intended ones simply looks as follows:

$$(\varphi, \psi, (\theta_q)_{q\in Q})$$
$$Q = \{\blacktriangleleft, \blacktriangleleft^*, \blacktriangleleft^+, \lhd, \ldots\}$$

$$\begin{array}{rcl}
\varphi & \equiv & \varphi_\mathcal{A} \\
\psi & \equiv & \mathrm{L}(x) \\
\theta_\blacktriangleleft(x,y) & \equiv & x \blacktriangleleft y \\
\theta_{\blacktriangleleft^*}(x,y) & \equiv & (\forall X)[\ \blacktriangleleft\text{-closed}(X) \wedge \\
 & & \qquad x \in X \rightarrow y \in X\ ] \\
\theta_{\blacktriangleleft^+}(x,y) & \equiv & x \blacktriangleleft^* y \vee x \not\approx y \\
\theta_\lhd(x,y) & \equiv & x \lhd y \\
\theta_{\text{labels}} & \equiv & \text{taken over from } R
\end{array}$$

In the particular situation of an MSO query, the domain of the transduction is given by the answer set $\mathcal{A}$, as described in Section 4. The set of nodes

of the intended tree is characterized by the formula ψ which identifies the nodes via the "linguistic" labels. Building on it, we define the other primitives of our description language analogous to $L^2_{K,P}$ by means of finite state tree walking automata. As explained in full detail in the paper by Morawietz and Mönnich (2001), the intended immediate dominance relation $\blacktriangleleft$ and the intended precedence relation $\lhd$ can be defined using a tree walking automaton, which itself can be translated into an MSO formula. Note that for a relation R, R-closed is the reflexive, transitive closure of R, which is known to be (weakly) MSO-definable (Courcelle, 1990).

An example of a reconstruction is given in Figure 3. The gray arcs show how the intended tree can be read of the LIFTed one.

6 Conclusion

In this paper, we presented an approach to querying context-sensitive relations with purely regular means. At the heart of this two-level approach lies the insight that lifting a context-free tree grammar results in a regular tree grammar, which, since it is regular, can again be handled by monadic second order logic and its associated automata theory. The seeming contradiction of using regular means to query mildly context-sensitive relations gets resolved by the old result (see, e.g., Courcelle (1990, 1997)) that the application of MSO-definable transductions on MSO-definable structures results in structures that may no longer be MSO-expressible.

It should be noted that both LIFTing the grammar and the reconstruction via MSO-transductions are of linear complexity. Restricting the generation of LIFTed candidate trees by means of the lifted grammar is a topic of further research. We expect the number of lifted trees for a given candidate tree to be polynomially bound by the size of the candidate tree. The problem of the complexity of translating an MSO-formula into a tree automaton is addressed by Neven and Schwentick (2000), who provide the definition of an MSO-fragment which has the same expressive power as full MSO on trees, but for which the translation into an automaton is only exponential.

Due to space restrictions we were not able to describe all the methods presented here in full mathematical detail. The interested reader is asked to consult the papers (Kolb et al., 2000a; Kolb et al., 2000b; Michaelis et al., 2001) and in particular (Morawietz and Mönnich, 2001).

References

Roderick Bloem and Joost Engelfriet. 1997. Characterization of properties and relations defined in Monadic Second Order logic on the nodes of trees. Technical Report 97-03, Dept. of Computer Science, Leiden University.

Peter Buneman, Mary Fernandez, and Dan Suciu. 2000. Unql: A query language and algebra for semistructured data based on structural recursion. *VLDB Journal*, 9(1):76–110.

Bruno Courcelle. 1990. Graph rewriting, an algebraic and logic approach. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 193–242. Elsevier.

Bruno Courcelle. 1997. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. I: Foundations*, pages 313–400. World Scientific.

Joost Engelfriet and Sebastian Maneth. 1999. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, 154:34–91.

Akio Fujiyoshi and Takumi Kasai. 2000. Spinal-formed context-free tree grammars. *MST: Mathematical Systems Theory*, 33.

Aravind Joshi and Yves Schabes. 1997. Tree adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3: Beyond Words of *Handbook of Formal Languages*, pages 69–123. Springer, Berlin.

Hans-Peter Kolb, Jens Michaelis, Uwe Mönnich, and Frank Morawietz. 2000a. An operational and denotational approach to non-context-freeness. To appear in: *Theoretical Computer Science*, Elesevier Science.

Hans-Peter Kolb, Uwe Mönnich, and Frank Morawietz. 2000b. Descriptions of cross-serial dependencies. *Grammars*, 3(2/3):189–216.

Thomas Maibaum. 1974. A generalized approach to formal languages. *J. Comput. System Sci.*, 88:409–439.

J. Mezei and Jesse Wright. 1967. Algebraic automata and contextfree sets. *Information and Control*, 11:3–29.

Jens Michaelis, Uwe Mönnich, and Frank Morawietz. 2001. On minimalist attribute grammars and macro tree transducers. In *(Rohrer et al., 2001)*.

Uwe Mönnich. 1997. Adjunction as substitution.

In G-J. M. Kruijff, G.V. Morill, and R.T. Oehrle, editors, *Formal Grammar '97*, pages 169–178.

Uwe Mönnich. 1999. On cloning contextfreeness. In Hans-Peter Kolb and Uwe Mönnich, editors, *The Mathematics of Syntactic Structure*, number 44 in Studies in Generative Grammar, pages 195–229. Mouton de Gruyter.

Frank Morawietz and Uwe Mönnich. 2001. A model-theoretic description of tree adjoining grammars. *ENTCS*, 53.

Frank Neven and Thomas Schwentick. 2000. Expressive and efficient pattern languages for tree-structured data. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 145–156. ACM.

James Rogers. 1998. *A Descriptive Approach to Language-Theoretic Complexity*. CSLI Publications and FoLLI.

Chistian Rohrer, Antje Roßdeutscher, and Hans Kamp, editors. 2001. *Linguistic Form and its Computation*. University of Chicago Press.

Stuart Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Edward Stabler. 2001. Minimalist grammars and recognition. In *(Rohrer et al., 2001)*.

James Thatcher and Jesse Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81.

Krishnamurti Vijay-Shanker and David Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.