Chapter 5

Model Checking Secondary Relations

STEPHAN KEPSER, UWE MÖNNICH, AND FRANK MORAWIETZ

ABSTRACT. In this paper we present an approach to model checking of mildly context-sensitive relations with purely regular means. The approach is based on a generalisation of the classical result that the intersection of a context-free language and a regular one is a context-free language which consists in defining context-freeness in terms of least fixed points of sets of equations over union and concatenation operators. When formalising (linguistic) trees using projection and composition as operators it is possible to characterise structures that are definable by linear context-free tree grammars and can hence be mildly context-sensitive. Since the translation of tree algebras using projection and composition operators into tree algebras familiar to linguists can be defined by monadic second-order transductions, the corresponding automaton theory can be put to use to query mildly context-sensitive secondary relations with purely regular means.

5.1 Introduction

Regarding linguistic data structures as relational structures makes them amenable to the techniques of model checking. The basic question in this area concerns the problem of how to devise efficient procedures that tell structures exhibiting a certain property from those that lack this property. As these properties are expressed by means of logical formulae, one can also regard the problem of model checking as a form of querying relational structures. Of special interested in this connection are formulae in the language of monadic second-order logic (MSO).

In the present paper we try to take advantage of a powerful generalisation of the classical result that the intersection of a context-free language and a regular one is a context-free language. The generalisation consists in defining a family of structures as context-free if it is a component of the least fixed-point of a system of equations over a finite set $\mathcal{F}$ of operations. In the particular case of context-free languages, the equations are expressed with the operations of union and concatenation. Using instead a distinguished set of projection and composition operations it becomes possible to characterise structures by means of appropriate systems of equations that are located on higher levels of the Chomsky hierarchy. Based on the particular set of operation symbols just mentioned the whole family of indexed languages can be accommodated within this framework.

The extension of the classical result concering the intersection of context-free and regular languages depends on an important property the set $T(\mathcal{F})$ of trees

Proceedings of Formal Grammar 2002
G. Jäger, P. Monachesi, G. Penn & S. Wintner (editors).

over $\mathcal{F}$ and the associated evaluation $val_{\mathcal{F}}$ which sends these trees into the intended structures has to satisfy. This property has been called MSO compatibility by Courcelle and Walukiewicz (1998) and requires of a (partial) mapping f from structures $\mathcal{S}(\Sigma)$ over the signature Σ into structures $\mathcal{S}(\Sigma')$ over the signature Σ' that for every MSO-sentence φ one can produce a backwards translation $f^{-1}(\varphi)$ such that

$$S \models f^{-1}(\varphi) \text{ iff } f(S) \models \varphi$$

for every structure S in the domain of f. Given the well-known fact that MSO definability on trees is equivalent to recognisability by finite tree automata the generalisation of the classical result follows immediately once the set of operators $\mathcal{F}$ under consideration is MSO compatible.

In a series of papers (Kolb et al. 2000a,b; Michaelis et al. 2001; Morawietz and Mönnich 2001) Mönnich, Morawietz, Kolb, and Michaelis have shown that the evaluation that interprets trees over projection and composition within the domain of structures familiar to linguists can be expressed as a simple form of MSO transduction ($def_{\Delta}(MSO)$). Such a transduction defines the intended structure, i.e., $val(t)$, for $t \in T(\mathcal{F})$, within the input structure t on the basis of a finite set of MSO formulae written in the signature of the input structure. These defining formulae can then be used for the backwards translation ($def_{\Delta}(MSO)^{-1}$) of a property that is expressed by an MSO formula over the target signature. Succinctly: MSO transductions *are* MSO compatible.

As it turns out, this relationship between the hierarchical structure of trees over composition and projection and their intended interpretation provides the foundation for a very flexible model checking procedure. Suppose one is dealing with a class of structures that exhibit a certain number of secondary relations. For concreteness assume that these relations indicate the sort of context-sensitive dependencies which have been at the focus of attention of linguists. As has been noticed since the beginning of formal language theory certain grammatical phenomena like morphological congruences (in, e.g., Bambara) and cross-serial dependencies between case markings (in Swiss German) are outside the realm of context-free languages and need for their descriptive analysis a (limited) amount of contextual information (Shieber 1985). Due to this character of context-sensitivity that comes with a range of grammatical constructions, even monadic second-order logic, normally considered a powerful query language, is too weak to capture these phenomena.

Taking our inspiration from the concept of MSO compatibility 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 operations 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 general 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 characterises exactly these cross-serial dependencies among the explicit structures that were out of the reach of the query language on the intended linguistic level.

Generalising from the particular problem of cross-serial dependencies in natural languages the impact of the classical result from formal language theory mentioned above can be described as follows. If a set of operations $\mathcal{F}$ and the associated interpretation function $val_{\mathcal{F}}$ satisfy the condition of MSO compatibility, the subset of structures within a context-free language $\mathcal{L}$ (in the general sense) that fulfil a certain MSO formula φ is context-free. In symbols:

$$\{S \mid S \models \varphi \wedge S \in \mathcal{L}\} \in CF$$

Relying again on the fact that MSO definability is equivalent to recognisability by finite tree automata the subset of $\mathcal{L}$ specified by the formula φ can be given an efficient regular description on the level of trees $T(\mathcal{F})$.

It has been shown that this method of regularising queries of context-sensitive structures can be adapted to most grammatical phenomena that fall within the reach of current linguistic theories (see the list of papers cited above). 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 query into a family of trees that can be checked by a suitable constraint formula for the intended dependencies. It can be shown 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. A paper explaining the details and describing an implementation is in preparation.

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. In a recent paper Courcelle and Knapik (2002) prove that the mapping which associates a term t over a complete set of graph operations with its evaluation $val(t)$ is an MSO-transduction (Proposition 2.5).

The method of turning the classical result from formal language theory into a powerful model checking procedure can also be put to use in the context of recent attempts to specify a common logical level for linguistic databases. As has been emphasised by Cotton and Bird (2002) the proliferation of linguistic databases with their bewildering diversity of formats and software tools makes it necessary to integrate them into a general multilayer annotation system. For the special case of treebank formats the authors show how they can be mapped onto the annotation graph model serving as a common logical level. Since the annotation graph model can be regarded as a special type of relational structure it is again easy to verify that the mapping from the entries of a treebank into annotation graphs is an interpretation along the lines of an MSO-transduction.

5.2 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 $\Sigma_{\varepsilon,s}$ 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 $f(t_1,\dots,t_n) \in T(\Sigma)$ if $f \in \Sigma_n$ and $t_i \in T(\Sigma)$ for $i = 1,\dots,n$.

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

5.3 Hierarchical Decomposition and Model Checking

The aim of the paper is to provide a way to check secondary relations in a linguistic database. A database in our sense is just a set of relational structures. As mentioned above, we use monadic second-order logic for querying the database. A query is therefore an MSO-formula, and the answer to a query is the set of all those structures in the database for which this formula is true. But since MSO is restricted to context-free phenomena, we need a device to specify the mildly context-sensitive secondary relations a linguist may be interested in. This device is a grammar. That is to say the linguist has to specify a grammar that generates the structures he is interested in. Obviously, this requires the grammar formalism to be more expressive than context-free (string) grammars. The largest class of grammars suitable for our approach is the class of linear 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 $t \in T(\Sigma \cup \mathsf{F}, \{x_1, \cdots, x_n\})$. The grammar is *linear*, iff each variable occurs at most once on the left hand side and at most once on the right hand side of a production.

Intuitively, an application of a rule of the form $F(x_1, \ldots, x_n) \to t$ "rewrites" a tree rooted in F as the tree t with its respective variables substituted by F's daughters. A context-free tree grammar generates elements of a tree substitution algebra $DT(\Sigma, X)$.

A CFTG $\Gamma = \langle \Sigma, \mathsf{F}, S, \mathsf{X}, \mathsf{P} \rangle$ with $\mathsf{F}_n = \emptyset$ for $n > 0$ 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 recognisable sets of trees, *a forteriori* context-free string languages (Mezei and Wright 1967). If F_n is non-empty for some $n \neq 0$, that is, if we allow the *operatives* to be parameterised 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 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 so-called 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 (2001), which are equivalent to certain types of multiple context-free 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 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 entire database either, because it will be used as a filter.

Let us illustrate the above by means of an example. The following CFTG generates the mildly context-sensitive language $a^n b^n c^n d^n$.

Example 2 Consider the CFTG $\Gamma = \langle\{a,b,c,d,\varepsilon,S_t,S_t^0\},\{S,S',\overline{S}_1,\overline{S}_2,\overline{a},\overline{b},\overline{c},\overline{d}\},S',\{x\},\mathrm{P}\rangle$ with P given as follows

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

An example of a tree generated by this grammar is shown in Figure 1.1.

In order to apply the general approach adumbrated in the introduction to the kind of secondary relations that can be accommodated within the framework of context-free tree grammars we need an appropriate set of operations that subtend the necessary hierarchical decomposition. 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 this LIFTing process is that each term compactly encodes information such as composition and concatenation.

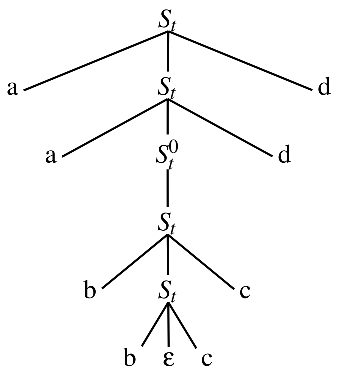

Figure 5.1: Sample tree

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 characterises a (necessarily recognisable) 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 unique homomorphism h is nothing else but the evaluation mapping *val* discussed above. The "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 $T(\Sigma, \mathrm{X}_k)$ 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. The trees over the derived "LIFTed" signature consisting of the old linguistic symbols together with the new projection and composition symbols form the carrier of a free tree algebra T^L.

Definition 3 [LIFT] Let Σ be a ranked alphabet of sort $\mathcal{S}$ and $X_k = \{x_1, \dots, x_k\}$, $k \in \mathbb{N}$, a finite set of variables. The *derived* many-sorted $\mathcal{S}^*$-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^n_i 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 $k_1 = \cdots = k_n = k$.

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

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

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

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 of a rule of a CFTG $\Gamma = \langle \Sigma, F, X, \mathcal{S}, P \rangle$ 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 2. Note that for better readability, we omit all the 0- and 1-place composition symbols.

Example 4 Let $\Gamma^L = \langle \{a, b, c, d, \varepsilon, S_t, S^0_t\}, \{S, S', \overline{S}_1, \overline{S}_2, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}, S', P \rangle$ 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: The linguistic non-terminals have become constants. An example tree generated by this LIFTed grammar is shown in Figure 5.2. It is the LIFTed tree corresponding to the sample tree of Figure 1.1.

Our main result provides a basis for a definition of the linguistically meaningful structures of the tree substitution algebra within the trees of the LIFTed algebra.

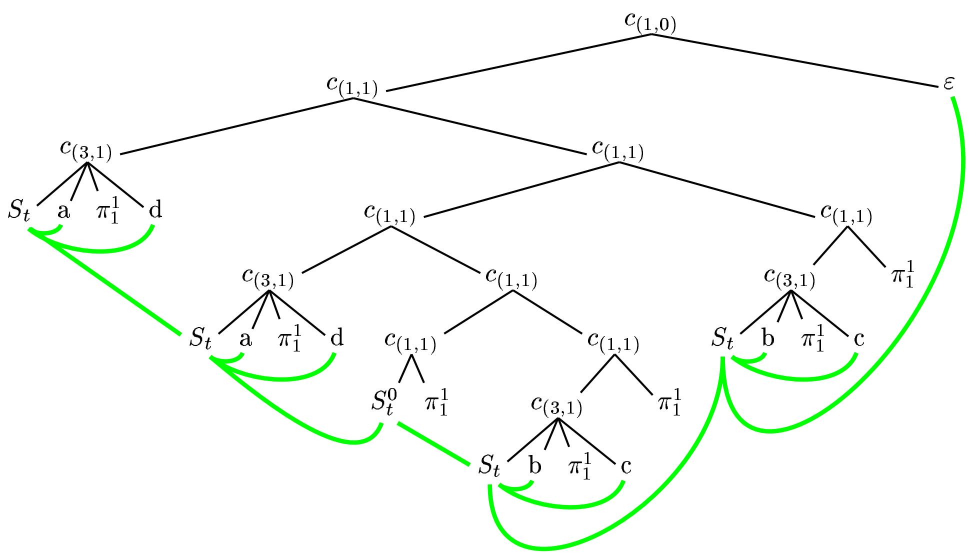

Figure 5.2: Intended relations on a LIFTed tree

Actually, it consists of a variant of the classical technique of interpreting one relational structure inside another one. The particular variant we use is due to Courcelle (1997) and interprets the domain and the relations on the substitution trees by means of suitable MSO formulae written in the signature of the LIFTed algebra.

Proposition 5 *The evaluation val from the free* LIFT*ed algebra T^L into the tree substitution algebra $DT(\Sigma,X)$ is an MSO-transduction*

$$def_\Delta(MSO) : T^L \to DT(\Sigma,X).$$

The idea for the proof of this proposition is due to Kolb (1999). As explained in detail in the paper by Morawietz and Mönnich (2001), it is possible to analyse the elements of T^L in such a way that the mapping from the free LIFTed tree algebra into the tree substitution algebra can be simulated by a tree walking automaton with so-called MSO-tests. The walks on a tree this automaton accepts connect nodes that satisfy the relations on the substitution trees. Bloem and Engelfriet (1997) showed that the relations recognised by tree walking automata with MSO-tests can be defined by MSO-formulae, thereby providing the desired logical definitions of the target relations. In other words, the intended relations of a tree of the tree substitution algebra can be reconstructed in its corresponding lifted tree. An example thereof is given in Figure 5.2 where the grey shaded arcs show the reconstruction paths for obtaining the intended tree from Figure 1.1. As an immediate consequence of the above proposition one has the following

Corollary 6 *The transformation of trees in T^L by means of composing and projecting subterms is MSO-compatible.*

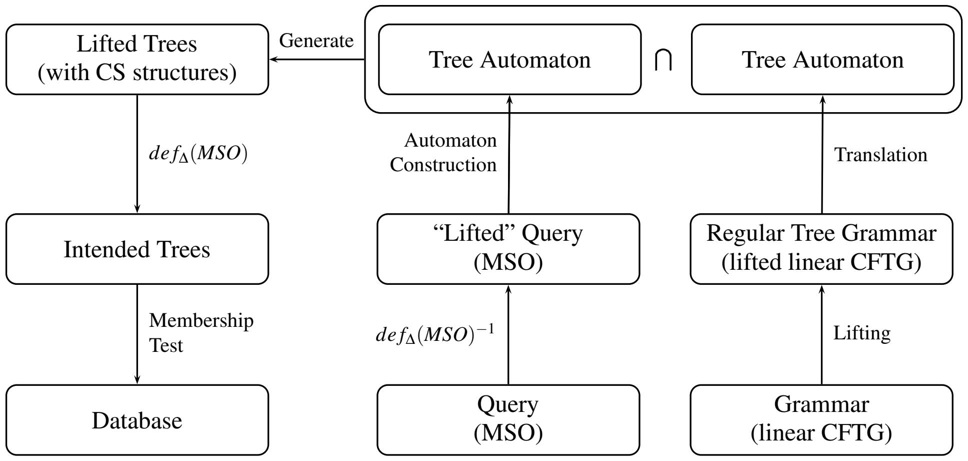

Figure 5.3: Overview of the approach

According to the corollary any MSO query φ addressed at the database can be translated into a query $val^{-1}(\varphi)$, i.e., the result of replacing the relation symbols occurring in φ by their images under $def_\Delta(MSO)^{-1}$, phrased by means of the derived vocabulary incorporating the composition and the projection symbols. By the well-known equivalence between tree automata and MSO formulae $val^{-1}(\varphi)$ has a translation into a corresponding tree automaton. The same transformation applied to the regular tree grammar which is the result of LIFTing the context-free tree grammar in the background of the supposed database produces another tree automaton. Intersection of these two automata produces an automaton A that accepts only structures that are in conformity with both the background grammar and the "lifted" query $val^{-1}(\varphi)$. Using this automaton A in generation yields a set of lifted trees with context-sensitive relations. As explained above, the set of intended trees is gained thereof via the MSO transduction $def_\Delta(MSO)$. Since the linear context-free tree grammar describing the desired context-sensitive secondary relations is supposed to be general in nature, the set of intended trees may contain trees not present in the database. Therefore it is necessary to perform a simple membership test on the database to retrieve the final answer set. Figure 5.3 gives an overview of our approach as we described it above.

5.4 Conclusion

In this paper, we presented an approach to model checking of context-sensitive relations with purely regular means. At the heart of this 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)) that the application of MSO-definable transductions on MSO-definable structures results in structures that may no longer be MSO-expressible.

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 by Kolb et al. (2000a,b); Michaelis et al. (2001) and in particular the one by Morawietz and Mönnich (2001).

Stephan Kepser, Uwe Mönnich, and Frank Morawietz
Theoretical Computational Linguistics Group
University of Tübingen, Germany
{kepser,um,frank}@sfs.uni-tuebingen.de

Bibliography

Bloem, R. and J. 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.

Cotton, S. and S. Bird (2002). An integrated framework for treebanks and multi-layer annotations. In *Proceedings LREC 2002*, pp. 1670–1677.

Courcelle, B. (1990). Graph rewriting, an algebraic and logic approach. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, volume B, pp. 193–242. Elsevier.

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

Courcelle, B. and T. Knapik (2002). The evaluation of first-order substitution is monadic second-order compatible. *Theoretical Computer Science*.

Courcelle, B. and I. Walukiewicz (1998). Monadic second-order logic, graph coverings and unfoldings of transition systems. *Annals of Pure and Applied Logic*, **92**:35–62.

Fujiyoshi, A. and T. Kasai (2000). Spinal-formed context-free tree grammars. *MST: Mathematical Systems Theory*, **33**.

Joshi, A. and Y. Schabes (1997). Tree adjoining grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, volume 3: Beyond Words of *Handbook of Formal Languages*, pp. 69–123. Springer, Berlin.

Kolb, H.-P. (1999). Macros for minimalism? In *Kolb and Mönnich (1999)*, pp. 231–258.

Kolb, H.-P., J. Michaelis, U. Mönnich, and F. Morawietz (2000a). An operational and denotational approach to non-context-freeness. To appear in: *Theoretical Computer Science*.

Kolb, H.-P. and U. Mönnich, eds. (1999). *The Mathematics of Syntactic Structure*. Mouton de Gruyter.

Kolb, H.-P., U. Mönnich, and F. Morawietz (2000b). Descriptions of cross-serial dependencies. *Grammars*, **3**(2/3):189–216.

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

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

Michaelis, J., U. Mönnich, and F. Morawietz (2001). On minimalist attribute grammars and macro tree transducers. In *Rohrer et al. (2001)*, pp. 287–326.

Mönnich, U. (1997). Adjunction as substitution. In G.-J. M. Kruijff, G. Morill, and R. Oehrle, eds., *Formal Grammar '97*, pp. 169–178.

Mönnich, U. (1999). On cloning contextfreeness. In *Kolb and Mönnich (1999)*, pp. 195–229.

Morawietz, F. and U. Mönnich (2001). A model-theoretic description of tree adjoining grammars. *ENTCS*, **53**.

Rohrer, C., A. Roßdeutscher, and H. Kamp, eds. (2001). *Linguistic Form and its Computation*. University of Chicago Press.

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

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

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