Monadic Second-Order Logic and Transitive Closure Logics over Trees

Hans-Jörg Tiede
Department of Mathematics and Computer Science
Illinois Wesleyan University
Bloomington, IL, USA

Stephan Kepser
Collaborative Research Centre 441
University of Tübingen
Tübingen, Germany

Abstract

Model theoretic syntax is concerned with studying the descriptive complexity of grammar formalisms for natural languages by defining their derivation trees in suitable logical formalisms. The central tool for model theoretic syntax has been monadic second-order logic (MSO). Much of the recent research in this area has been concerned with finding more expressive logics to capture the derivation trees of grammar formalisms that generate non-context-free languages. The motivation behind this search for more expressive logics is to describe formally certain mildly context-sensitive phenomena of natural languages. Several extensions to MSO have been proposed, most of which no longer define the derivation trees of grammar formalisms directly, while others introduce logically odd restrictions. We therefore propose to consider first-order transitive closure logic. In this logic, derivation trees can be defined in a direct way. Our main result is that transitive closure logic, even deterministic transitive closure logic, is more expressive in defining classes of tree languages than MSO. (Deterministic) transitive closure logics are capable of defining non-regular tree languages that are of interest to linguistics.

1 Introduction

Model theoretic syntax is a research program in mathematical linguistics introduced by Rogers [22]. It is concerned with studying the descriptive complexity of grammar formalisms for natural languages by defining their derivation trees in suitable logical formalisms. The central tool for model theoretic syntax has been monadic second-order logic (MSO), interest in which is motivated by its relationship to context-free grammars: the yields of MSO-definable tree languages are context-free string languages (see [27]).

Much of the recent research in model theoretic syntax has been concerned with finding more expressive logics to capture the derivation trees of grammar formalisms that generate non-context-free languages. The motivation for this is the desire to capture all phenomena of natural languages by model theoretic means. While the morphology and syntax of most natural languages are known to be describable by context-free string languages, there are a few phenomena that transcend this framework. Among these "mildly context-sensitive" phenomena are cross-serial dependencies in verb clusters of Dutch and Swiss German [9, 25] and parts of the morphology of Bambara [5].

Even in the context of the model theoretic description of mildly context-sensitive phenomena, MSO has played a central role. For example, Rogers [23] extends MSO to n-dimensional trees, Kolb et al. [13] encode non-regular tree language in regular tree languages, and Langholm [14] extends MSO by adding quantification over certain functions, following Lautemann et al. [15] who characterized the context-free string languages in a similar fashion.

The main constraints placed on logics for model theoretic syntax are that they should be decidable, be natural from a logical point of view, and correspond to automata theoretical complexity measures of tree or string languages. These constraints to some extent conflict with each other. For instance, finding logics that correspond to particular formal language classes may result in a logic that is somewhat unnatural from a logical point of view or undecidable, due to closure or decision properties of the corresponding language class. Furthermore, some of the extensions of MSO discussed above have in common that they no longer define the derivation trees of grammar formalisms directly.

In this paper, we consider a different approach to extending the definability of MSO: first-order transitive closure logic (FO(TC)), which was introduced by Immerman (see the references in [10]) to capture the complexity class NLOGSPACE descriptively. The main motivation for this approach is that derivation trees can be defined in a direct fashion. On the other hand, the expressive power of FO(TC) is large enough to describe the known mildly context-sensitive phenomena of natural language. Indeed, we show here that the classes of tree languages definable by FO(TC) strictly extend the classes of tree languages definable by MSO. This is true even for *deterministic* transitive closure logic. These results may look somewhat surprising, because Moschovakis [19] showed that the transitive closure of an MSO-definable binary relation is MSO definable. This let to a wide spread common belief in computational linguistics, spelled out explicitely in [3], that MSO should be the more expressive logic as compared to FO((D)TC). The higher expressive power of FO(DTC) is based on the capability of taking transitive closures over relations on tuples of nodes instead of individual nodes. Indeed it is possible to define a non-regular tree language using a deterministic transitive closure over a relation on pairs of nodes. While FO(TC) can define derivation trees of grammars directly, as well as describe non-context-free phenomena, it does not retain decidability, as we will demonstrate below.

2 Preliminaries

We consider finite labelled ordered ranked trees. A tree is ordered, if for each node in the tree its set of children is totally ordered. A tree is ranked, if the number of children of a node is a function of the label.

A signature Σ consists of a set of function symbols S and an arity function $\rho : S \to \mathbb{N}$ that assigns each function symbol its arity. Function symbols of arity 0 are called constants. We will only consider finite signatures. The set T_Σ of trees (or terms) over a signature Σ are inductively defined as follows: $\{c \mid c \in S, \rho(c) = 0\} \subseteq T_\Sigma$ and if $f \in S$ with $\rho(f) = n > 0$ and $t_1, \ldots, t_n \in T_\Sigma$ then $f(t_1, \ldots, t_n) \in T_\Sigma$. For a given signature Σ, a *tree language* is just a subset of T_Σ.

The *yield* of a tree is the sequence of labels of leaves of a tree, i.e., $yield : T_\Sigma \to S^*$ such that $yield(c) = c$ for every constant (leaf) c and $yield(f(t_1, \ldots, t_n)) = yield(t_1) \circ \cdots \circ yield(t_n)$, where $\circ$ denotes string concatenation.

To describe trees by means of logics we regard them as relational structures. In this view, function symbols from S turn into node labels, i.e., unary predicates. For a finite signature of function symbols Σ there is an $r \in \mathbb{N}$ such that r is the maximal arity of a function symbol in Σ. We define r successor relations $S_1, \ldots, S_r$. A pair of nodes (x, y) stands in the relation $S_i(x, y)$ if x has a label with arity of at least i and y is the i-th child of x.

A tree language is *regular* iff it is definable by an MSO-sentence. The yield language of a regular tree language is a context-free string language. Every context-free string language is the yield of some regular tree language.

Certainly not every relational structure is a tree. And it is also well known, that the class of structures that are finite trees or trees is *not* first-order logic axiomatizable, but MSO-axiomatizable.

For a thorough discussion of these issues the reader is kindly referred to [22].

3 Transitive Closure Logic

A fundamental restriction in the expressive power of first-order logic is the lack of any type of recursion mechanism. One of the simplest and most fundamental queries that are not first-order expressible is the *transitive closure*, denoted TC. It assigns to a given binary relation E on a universe U its reflexive transitive closure, i.e., the set of all pairs $(x, y) \in U \times U$ such that there exist $z_0, \ldots, z_r \in U$ with $z_0 = x, z_r = y$ and $E(z_i, z_{i+1})$ for all $i < r$. It was first shown in [8] that TC is not expressible in FO.

Let M be a set and $R \subseteq M \times M$ a binary relation over M. The *transitive closure* $TC(R)$ of R is the smallest set containing R and for all $x, y, z \in M$ such that $(x, y) \in TC(R)$ and $(y, z) \in TC(R)$ we have $(x, z) \in TC(R)$, i.e.,

$$TC(R) := \bigcap \{W \mid R \subseteq W \subseteq M \times M, \forall x, y, z \in M : (x, y), (y, z) \in W \implies (x, z) \in W\}.$$

This notion can be extended to relations over tuples. Let $k \in \mathbb{N}$ and R a binary relation over k-tuples ($R \subseteq M^k \times M^k$). Then

$$TC(R) := \bigcap \{W \mid R \subseteq W \subseteq M^k \times M^k, \forall \bar{x}, \bar{y}, \bar{z} \in M^k : (\bar{x}, \bar{y}), (\bar{y}, \bar{z}) \in W \implies (\bar{x}, \bar{z}) \in W\}.$$

Deterministic transitive closure is the transitive closure of a deterministic, i.e., functional relation. For an arbitrary binary relation R over k-tuples we define its *deterministic reduct* by

$$R_D := \{(\bar{x},\bar{y}) \in R \mid \forall \bar{z} : (\bar{x},\bar{z}) \in R \implies \bar{y} = \bar{z}\}.$$

Now

$$DTC(R) := TC(R_D).$$

Since neither the transitive closure of a relation nor its deterministic counterpart are definable in FO, as was shown in [8], it makes sense to add these operators to first-order logic to extend its expressive power in moderate and controlled way.

Definition 1 The formulae of FO(TC) are defined by adding to first-order logic the transitive closure operator (TC):

> If φ is an FO(TC) formula, $\bar{x} = x_1,\ldots,x_n$, $\bar{y} = y_1,\ldots,y_n$ are a subset of the free variables of φ such that $\forall i,j, x_i \neq y_j$, and $\bar{s} = s_1,\ldots,s_n, \bar{t} = t_1,\ldots,t_n$ are terms, then $[\mathrm{TC}_{\bar{x},\bar{y}}\ \varphi]\bar{s},\bar{t}$ is an FO(TC) formula. For FO(DTC) we add the deterministic transitive closure operator. If φ is an FO(DTC) formula, then $[\mathrm{DTC}_{\bar{x},\bar{y}}\ \varphi]\bar{s},\bar{t}$ is an FO(DTC) formula.

We also consider logics in which the length of the tuples are restricted to be of length at most k, which are denoted by FO(TC^k) or FO(DTC^k). Of special interest are the cases where $k = 1$, i.e., monadic transitive closure logic, and where $k = 2$, i.e., transitive closures are taken over binary relations on pairs. It is clear from the definition that if $m \leq n$, FO(TC^m) is included in FO(TC^n) and similarly for their deterministic counterparts.

A predicate of the form $[\mathrm{TC}_{\bar{x},\bar{y}}\ \varphi]$ ($[\mathrm{DTC}_{\bar{x},\bar{y}}\ \varphi]$) is supposed to denote the (deterministic) transitive closure of the relation defined by φ.

Definition 2 We define $\mathfrak{M} \models \varphi$ for FO(TC) or FO(DTC) in the usual way. To evaluate predicates defined with the transitive closure operator, we define

$$\mathfrak{M} \models [\mathrm{TC}_{\bar{x},\bar{y}}\ \varphi]\bar{s},\bar{t}$$

iff

$$(\bar{s}^{\mathfrak{M}},\bar{t}^{\mathfrak{M}}) \in TC\{(\bar{a},\bar{b}) \mid \mathfrak{M} \models \varphi[\bar{a},\bar{b}]\}.$$

And

$$\mathfrak{M} \models [\mathrm{DTC}_{\bar{x},\bar{y}}\ \varphi]\bar{s},\bar{t}$$

iff

$$(\bar{s}^{\mathfrak{M}},\bar{t}^{\mathfrak{M}}) \in DTC\{(\bar{a},\bar{b}) \mid \mathfrak{M} \models \varphi[\bar{a},\bar{b}]\}.$$

We just mention the following in passing. For every formula in FO(DTC) there exists an equivalent formula in FO(TC) (see, e.g., [10]). Secondly, Engelfriet and Hoogeboom [6] provided an automaton model for FO(DTC). For every k, the logic FO(DTC^k) corresponds to a particular type of deterministic pebble tree walking automata with k heads. Whether there exists also an automaton model for FO(TC) is an open question.

To provide some intuition on the use and expressive power of FO(TC) and FO(DTC) we show how to define a linear order on the nodes of a tree. A detailed presentation of this property of FO(DTC) can be found in [11]. We assume a given finite signature $\Sigma = (S, \rho)$ and maximal arity r. The dominance relation in a tree can be defined by an FO(TC)-formula in the following way

$$dom(x,y) := [\mathrm{TC}_{x,y}\, S_1(x,y) \vee S_2(x,y) \vee \cdots \vee S_r(x,y)](x,y).$$

More interestingly, the dominance relation is already definable in FO(DTC[1]) by the following formula

$$dom(x,y) := [\mathrm{DTC}_{y,x}\, S_1(x,y) \vee S_2(x,y) \vee \cdots \vee S_r(x,y)](y,x).$$

In order to be able to use a deterministic transitive closure here one has to look at trees from the leaves to the root. The "child-of" relation is non-deterministic, but each node has a unique parent. Hence we define dominance via the deterministic transitive closure of the parent relation. This idea goes back to Etessami and Immerman [7]. That two nodes x and y are siblings and x is to the left of y can be expressed in FO by a disjunction as follows

$$LSib(x,y) := \exists z \bigvee_{\substack{i=1,\ldots,r-1 \\ j=i+1,\ldots,r}} S_i(z,x) \wedge S_j(z,y).$$

The order on the nodes we define now is depth-first left-to-right tree traversal, also known as preorder. A node x is smaller than y according to this order if x dominates y or x is to be found in a subtree to the left of the subtree where y is to be found. Formally, $x < y$ is defined as

$$dom(x,y) \vee \exists z,w : LSib(z,w) \wedge (z = x \vee dom(z,x)) \wedge (w = y \vee dom(w,y)).$$

It is simple to define an immediate successor y of a node x according to the order by setting

$$Succ(x,y) := x < y \wedge \neg \exists z : x < z < y.$$

The definition of a minimum and maximum of the order are even simpler. This example shows that trees are ordered structures in the logics FO(DTC) and FO(TC).

The following part presents the main insight we want to convey, namely that transitive closure logics are more powerful than MSO on finite trees. The results to follow are known to be true for string languages. But our emphasis lies on trees as the data structures underlying model theoretic syntax. The next theorem is an extension to the one given in [28].

Theorem 3 *Every regular tree language is definable in FO(DTC).*

Proof This result goes back to results by Mehlhorn [18] and Lynch [17], who independently showed that parentheses languages are LOGSPACE-recognizable. Their algorithms extend immediately to regular tree languages. And on ordered structures LOGSPACE and FO(DTC) define the same classes of structures [10]. □

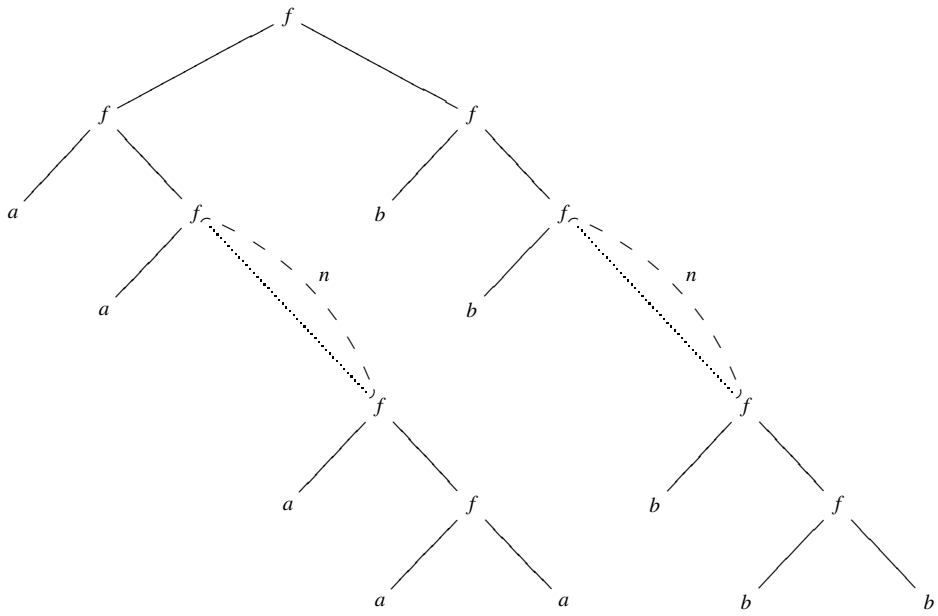

Figure 1: Trees with crossed dependencies as defined by Formula 1.

A fortiori, every regular tree language is definable in FO(TC). We are here interested in the observation that there are non-regular tree languages that can be defined using FO(TC). Again, we present a strengthening of that statement.

Proposition 4 *There exists a non-regular tree language that is FO(DTC^2)-definable.*

This proposition can be derived from known results on string languages. It is easy to see that the non-regular string language $\{a^n b^n \mid n \in \mathbb{N}\}$ is definable in FO(DTC^2).

But in the spirit of model theoretic syntax we would like to present another proof, which exemplifies which types of non-regular tree structures can be defined using FO(DTC^2).

Proof We consider labelled binary trees. We will stepwise construct a formula which defines the tree language sketched in Figure 1. There are two successor relations S_1, S_2. We have the following labels: f, a, b where a and b are labels of leaves and f is the label of internal nodes. There is a constant, r, for the root node of a tree. Now consider the following predicate P:

$$[\mathrm{DTC}_{(y_1,y_3),(y_2,y_4)}\ S_2(y_1,y_2) \wedge S_2(y_3,y_4)]$$

which states that y_2 is at the same distance from y_1 on a right branch as y_4 from y_3. Let $Leaf(x)$ denote that x is a leaf, i.e.,

$$Leaf(x) := \neg\exists y S_1(x,y) \vee S_2(x,y).$$

For perspicuity, let $\hat{a}(x)$ be $a(x) \wedge Leaf(x)$ and let $\hat{b}(x)$ be $b(x) \wedge Leaf(x)$. Let $\varphi(x_1,x_2)$ be the formula

$$\begin{aligned}\forall y_1,y_2,y_3,y_4 \;\; & P(x_1,x_2,y_1,y_2) \;\rightarrow \\ & ((\hat{a}(y_1) \wedge \hat{b}(y_2)) \;\vee \\ & (S_1(y_1,y_3) \wedge S_1(y_2,y_4) \wedge f(y_1) \wedge f(y_2) \wedge \hat{a}(y_3) \wedge \hat{b}(y_4))).\end{aligned}$$

Then

$$\begin{aligned}\exists x_1,x_2 \;\; & (f(r) \wedge S_1(r,x_1) \wedge S_2(r,x_2) \wedge \\ & ((\hat{a}(x_1) \wedge \hat{b}(x_2)) \;\vee \\ & (f(x_1) \wedge f(x_2) \wedge \exists y_1,y_2\,(S_1(x_1,y_1) \wedge S_1(x_2,y_2) \wedge \hat{a}(y_1) \wedge \hat{b}(y_2)) \wedge \\ & \varphi(x_1,x_2))))\end{aligned} \tag{1}$$

defines the trees that are isomorphic to derivation trees for the context-free string language $\{a^n b^n \mid n \geq 1\}$ with crossed dependencies, which is not a regular tree language. Apart from the trivial trees $f(a,b), f(f(a,a),f(b,b))$ trees have the shape depicted in Figure 1.

The two subtrees below the root node are isomorphic. This fact cannot be expressed by any MSO formula, a fact that follows from the pumping lemma for regular tree languages. □

The following corollary can be obtained from this result.

Corollary 5 *The expressive power of FO(DTC) as a language to define classes of ordered trees is strictly higher than that of MSO.*

The proof of Theorem 3 is based on the fact that every regular tree language is LOGSPACE-recognizable. Indeed a stronger result is known. Lohrey [16] showed that regular tree languages are even ALogTime-recognizable. One may now ask whether all ALogTime-recognizable tree languages are regular or whether there are ALogTime-recognizable tree languages which are not regular. The latter is indeed the case. There are actually quite simple non-regular tree languages which are ALogTime-recognizable. An example is a variant of the tree language in the proof above. In this example we take the algebraic view on trees and consider the following signature $\Sigma = (\{f,g,a\},\{(f,2),(g,1),(a,0)\})$ where f is binary, g unary, and a a constant. The tree language $\{f(g^n(a), g^n(a)) \mid n \geq 1\}$ is clearly not regular. But it is ALogTime-recognizable.

For a given finite signature $\Sigma = (S,\rho)$ the formula

$$idlab(x,y) := \bigvee_{f \in S} f(x) \wedge f(y)$$

expresses that x and y carry the same label. We will use this formula to show that subtree isomorphism is definable in FO(TC).

Proposition 6 *Let Σ be a finite signature with maximal arity r. The subtree isomorphism relation $Iso(x,y)$, which holds of two nodes x and y if the subtrees rooted in x and y are isomorphic, is FO(DTC^2) definable.*

Proof Let P be the predicate

$$[\mathrm{DTC}_{(x_3,x_4)(x_1,x_2)} \bigvee_{i=1}^{r} S_i(x_1,x_3) \wedge S_i(x_2,x_4)]$$

which states that the path from x_1 to x_3 is isomorphic to the path from x_2 to x_4 (not considering labels). Then

$$\begin{aligned} Iso(x,y) = \quad & \forall z \exists w(dom(x,z) \rightarrow P(z,w,x,y) \wedge idlab(z,w)) \\ & \wedge \forall z \exists w(dom(y,z) \rightarrow P(w,z,x,y) \wedge idlab(w,z)) \end{aligned}$$

states the usual back and forth conditions of isomorphism. □

This is another example of a relation that is *not* MSO-definable, as was shown in [22]. Its definability in FO(DTC) has the following consequence.

Proposition 7 *The logic FO(DTC^2) is undecidable on finite ordered trees.*

Actually, it is known that FO(DTC^2) is undecidable even on strings. Every string language recognizable by a deterministic two-way two-head automaton is FO(DTC^2)-definable, as shown in [1]. And the emptyness problem for deterministic two-way two-head automata is undecidable [24].

The following proof shows that Proposition 7 follows from Proposition 6. We present it as an example demonstrating the expressive power of FO(DTC^2) over trees.

Proof The proof is based on Rogers' [22, p. 48ff] proof of the *un*definability of subtree isomorphism in MSO. For any set $\mathcal{D}$ of tiles, the origin-constrained tiling problem is definable in FO where the tiles of $\mathcal{D}$ are unary predicates, as shown by Rogers. Therefore the FO theory of grids is undecidable. Rogers furthermore proves that the theory of finite grids can be encoded in the theory of finite trees provided there is a predicate available expressing subtree isomorphism. The grid is basically encoded by demanding that each pair of paths that end in the same node on the grid have to end in isomorphic nodes in the tree. Since subtree isomorphism is definable in FO(DTC^2) and origin-constrained tiling problems are definable in FO(DTC^2), it follows that FO(DTC^2) is undecidable over finite ordered trees. □

The proposition implies that both FO(DTC) and FO(TC) are undecidable on finite ordered trees.

Monadic Transitive Closure Logics

All of the above results use (deterministic) transitive closures of tuples of width at least 2. If we restrict the transitive closure operators to be applied to binary relations only (denoted as FO((D)TC^1) and called *monadic* transitive closure logic), the situation changes. It is an old result that goes back at least to Moschovakis [19] that the transitive closure of every MSO-definable binary relation is also MSO-definable. Let R be an MSO-definable binary relation. Then

$$\forall X(\forall z,w(z \in X \wedge R(z,w) \implies w \in X) \wedge \forall z(R(x,z) \implies z \in X)) \implies y \in X$$

is a formula with free variables x and y that defines the transitive closure of R (see [4] for details). It follows that every tree language definable in FO(TC^1) can be defined in MSO. The same is true for string languages. But we observe an interesting difference in expressive power between string languages and tree languages when we ask whether every MSO definable language is also FO(TC^1) definable. On the one hand, Bargury and Makowsky [1] showed that MSO and monadic transitive closure logic are equally expressive on string languages. It should be noted that this result follows immediately from Kleene's theorem on regular languages [12], since regular expressions are practically a special kind of FO(TC^1) formulae. Translations from finite word automata to more restricted fragments of FO(TC^1), thus yielding a stronger normal form, can be found in [20]. On the other hand, ten Cate and Segoufin [26] recently demonstrated the existence of MSO definable tree languages that can *not* be defined in FO(TC^1).

It is clear that FO(DTC^1) extends the expressive power of FO, because it is capable of expressing some second-order properties. This is of interest for model theoretic syntax, because it is possible to define over arbitrary structures the classes of trees and finite trees in FO(DTC^1), as was shown by Kepser [11]. The defining formulae are derived from the ones given by Rogers [22], but it is shown that the MSO-formulae can be translated into the weaker logic FO(DTC^1).

4 A TC Logic Account of Cross-Serial Dependencies

An important motivation for this paper is to propose a logic for the logical description of natural language providing a direct definition of derivation trees. The aim of this section is to show that MSO is insufficient for this task while transitive closure logics suffice. The underlying reason is that there are natural languages the sentences of which cannot be described by context-free (string) languages. The discussion about the status of natural language started very early after the definition of the Chomsky hierarchy. But many early arguments in favour of the non-contextfreeness of natural languages were simply incorrect (see [21]). Finally, Huybregts [9] provided data for Swiss German and Dutch that show that neither of these languages can be context-free. Shortly after Shieber [25] independently provided the same data for Swiss German. These data exhibit cross-serial dependencies in the verbal complex. Consider the following example:

wil	de Karl	d'Maria	em Peter	de Hans	laat	hälffe	lärne	schwüme
because	Charles	Mary_1	Peter_2	John_3	lets_1	help-inf_2	teach-inf_3	swim-inf

'because Charles lets Mary help Peter to teach John to swim'

The main observations here are the following: Swiss German has overt case marking for Dative and Accusative case. Verbs like *laat* and *lärne* take their objects in Accusative case, verbs like *hälffe* take their objects in dative case. When we consider the sequence of objects and the sequence of verbs to which they belong, we observe the following pattern of cross-serial dependencies:
$NP_1\ NP_2\ NP_3\ V_1\ V_2\ V_3$
It appears that there are no limits on the length of such constructions in grammatical sentences of Swiss German.

In order to show that Swiss German in *toto*, and not just the above fragment, is not context-free Shieber argues as follows. Firstly, there are subordinate clauses where all Vs follow all NPs. Secondly, sentences where all dative NPs precede all accusative NPs and all verbs subcategorizing for dative NPs precede all verbs subcategorizing for Accusative NPs are grammatical. Thirdly, the number of verbs and their corresponding objects must agree. And lastly, an arbitrary number of verbs can occur in such clauses. The argument is now completed using the well-known fact that context-free languages are closed under intersection with regular languages. Shieber defines the following regular language:

Karl säit das mer (d'chind)* (em Hans)* es huus händ wele (laa)* (hälfe)* aastriiche.
Charles said that we (the children)* (John)* the house wanted to (let)* (help)* paint.

Intersecting Swiss German with this regular language results in the following language:

Karl säit das mer (d'chind)^n (em Hans)^m es huus händ wele (laa)^n (hälfe)^m aastriiche.

which is known not to be context-free.

We will now provide an FO(DTC) formula defining a tree language with $a^n b^m c^n d^m$ as yield language using a method similar to the one in Proposition 4. Labels of leaf nodes are a,b,c,d, binary internal nodes are labelled with f, the root node has four children and is labelled with rt. Again consider the predicate P:

$$[\mathrm{DTC}_{(y_1,y_3),(y_2,y_4)}\ S_2(y_1,y_2) \wedge S_2(y_3,y_4)]$$

which still states that y_2 is at the same distance from y_1 on a right branch as y_4 from y_3. For simplicity, we use $\hat{i}(x)$ for $i(x) \wedge Leaf(x)$ where $i \in \{a,b,c,d\}$. Let $\varphi_1(x_1,x_2)$ be the formula

$$\begin{array}{ll} \forall y_1,y_2,y_3,y_4 & P(x_1,x_2,y_1,y_2) \ \rightarrow \\ & ((\hat{a}(y_1) \wedge \hat{c}(y_2)) \ \vee \\ & (S_1(y_1,y_3) \wedge S_1(y_2,y_4) \wedge f(y_1) \wedge f(y_2) \wedge \hat{a}(y_3) \wedge \hat{c}(y_4))). \end{array}$$

Let $\varphi_2(x_1,x_2)$ be the result of replacing label a by b and c by d in formula $\varphi_1(x_1,x_2)$, i.e.,

$$\begin{array}{ll} \forall y_1,y_2,y_3,y_4 & P(x_1,x_2,y_1,y_2) \ \rightarrow \\ & ((\hat{b}(y_1) \wedge \hat{d}(y_2)) \ \vee \\ & (S_1(y_1,y_3) \wedge S_1(y_2,y_4) \wedge f(y_1) \wedge f(y_2) \wedge \hat{b}(y_3) \wedge \hat{d}(y_4))). \end{array}$$

Then

$$\begin{array}{lll} \exists x_1,x_2,x_3,x_4 & rt(r) \wedge S_1(r,x_1) \wedge S_2(r,x_2) \wedge S_3(r,x_3) \wedge S_4(r,x_4) \wedge & (2) \\ & f(x_1) \wedge f(x_2) \wedge f(x_3) \wedge f(x_4) \wedge & \\ & \exists y_1,y_2,y_3,y_4 \ \ S_1(x_1,y_1) \wedge S_1(x_2,y_2) \wedge S_1(x_3,y_3) \wedge S_1(x_4,y_4) \wedge & \\ & \qquad \hat{a}(y_1) \wedge \hat{b}(y_2) \wedge \hat{c}(y_3) \wedge \hat{d}(y_4) \wedge & \\ & \varphi_1(x_1,x_3) \wedge \varphi_2(x_2,x_4) & \end{array}$$

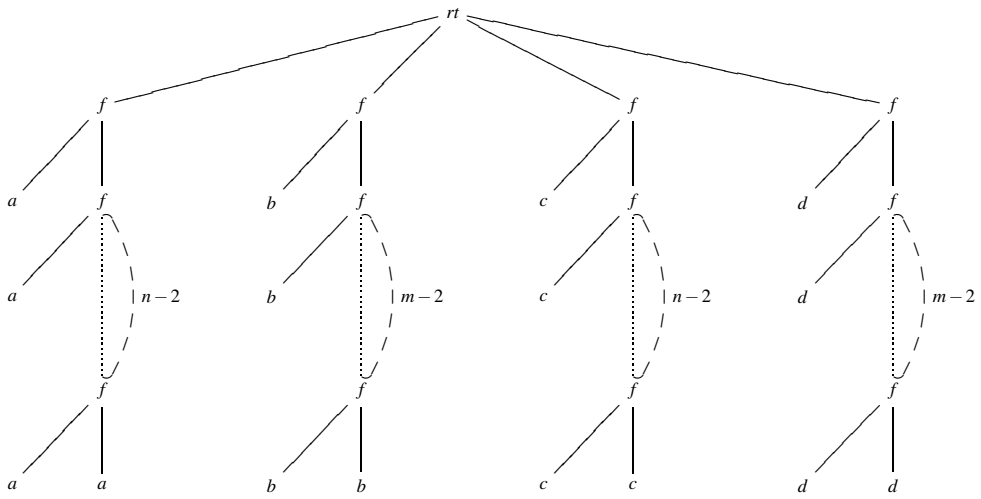

Figure 2: Shapes of trees defined by Formula 2.

defines a tree language with yield language $\{a^n b^m c^n d^m \mid n,m > 1\}$.[1] The shape of the trees is sketched in Figure 2. It should be noted that this form of analysis of cross-serial dependencies is supported by findings in the linguistic literature (see, e.g.,[2]).

There exists another view onto this problem. If one only considers the pattern of cross-serial dependency $\mathrm{NP}_1\ \mathrm{NP}_2\ \mathrm{NP}_3\ \ldots\ \mathrm{V}_1\ \mathrm{V}_2\ \mathrm{V}_3\ \ldots$ then the resulting string language $\{\mathrm{NP}^n\,\mathrm{V}^n\}$ is obviously context-free and abstracts from the inherent dependencies. The derivation trees on the other hand should certainly capture these dependencies. That this can be done in FO(DTC) was already shown in the previous section. Please reconsider Formula 1 and the corresponding Figure 1. By a closer look onto them it is simple to see that Formula 1 exactly defines the type of cross-serial dependencies we discuss in the present section.

There are two issues we would like to point out here. Firstly, it is sufficient to use the deterministic transitive closure of relations on pairs. This logic can be seen as a minimal extension over MSO. Secondly, the logic does not just define the desired string language. Rather it captures the notion of a cross-serial dependency in a direct fashion in the derivation trees.

5 Conclusion

We have given some indications that FO(TC) and FO(DTC) are useful formalisms for model theoretic syntax. We showed that the classes of tree languages that can be

[1] The lack of definitions for trees with yield language $abcd$, a^2bc^2d, and ab^2cd^2 is immaterial to the point we make here. Adding these definitions would be trivial, but make formulas only harder to read.

defined by both languages properly extend the classes of tree languages that can be defined with MSO. We also provided an indication that the known non-contextfree phenomena in natural languages can be rendered using FO(TC) or even FO(DTC).

There are some interesting open problems regarding the relationship between MSO and FO(TC), particularly whether MSO is strictly weaker than FO(TC^2). In fact, it is possible that MSO is incomparable to FO(TC^k) for each $k > 1$.

From a perspective of model-theoretic syntax there is an interesting tension to be observed here. The use of transitive closure logics for the description of derivation trees that we advocate here has the advantage of being capable to express properties of derivation trees in a direct fashion. The price to be paid for this advantage seems to be the undecidability of the logics. As stated in the introduction, there are competing extension to MSO for model-theoretic syntax. They frequently lack the capability of expressing derivation tree properties in a direct way and encode them rather indirect. As a "compensation" for this disadvantage, at least the approach described in [13] remains decidable over finite trees. It seems that linguists have to make a choice of which property they regard as more important: direct perspicuous encoding or decidability.

Acknowledgements

We would like to thank Sam Buss for an interesting discussion on ALogTime-recognizable tree languages. We would also like to express our gratitude to 7 anonymous referees for their comments which helped significantly improving this paper.

References

[1] Yaniv Bargury and Johann A. Makowsky. The expressive power of transitive closue and 2-way multihead automata. In Egon Börger, Gerhard Jäger, Hans Kleine Büning, and Michael M. Richter, editors, *Computer Science Logic, CSL '91*, LNCS 626, pages 1–14, Berlin, 1992. Springer.

[2] Joan Bresnan, Ron Kaplan, Stanley Peters, and Annie Zaenen. Cross-serial dependencies in Dutch. In Walter J. Savitch, Emmon Bach, William Marsh, and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*, pages 286–319. Reidel, Dordrecht, 1987.

[3] Tom Cornelle and James Rogers. Model theoretic syntax. In Lisa Cheng and Rint Sybesma, editors, *The First GLOT International State-of-the Article Book*, SGG 48, Berlin, 2000. Mouton de Gruyter.

[4] Bruno Courcelle. Graph rewriting: an algebraic and logic approach. In *Handbook of theoretical computer science, Vol. B*. Elsevier, Amsterdam, 1990.

[5] Christopher Culy. The complexity of the vocabulary of Bambara. In Walter Savitch, Emmon Bach, William Marsch, and Gila Safran-Naveh, editors, *The Formal Complexity of Natural Language*, pages 345–351. Reidel, 1987.

[6] Joost Engelfriet and Hendrik Jan Hoogeboom. Nested pebbles and transitive closure. In Bruno Durand and Wolfgang Thomas, editors, *STACS 2006*, volume LNCS 3884, pages 477–488, Berlin, 2006. Springer.

[7] Kousha Etessami and Neil Immerman. Reachability and the power of local ordering. *Theoretical Computer Science*, 148(2):261–279, 1995.

[8] Ronald Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.

[9] Riny Huybregts. The weak inadequacy of context–free phrase structure grammars. In Ger J. de Haan, Mieke Trommelen, and Wim Zonneveld, editors, *Van Periferie naar Kern*, pages 81–99. Foris, Dordrecht, 1984.

[10] Neil Immerman. *Descriptive Complexity*. Springer, Berlin, 1999.

[11] Stephan Kepser. Properties of binary transitive closure logic over trees. In Paola Monachesi, Gerald Penn, Giorgio Satta, and Shuly Wintner, editors, *Formal Grammar 2006*, pages 77–89, 2006.

[12] Stephen. C. Kleene. Representation of events in nerve nets and finite automata. In *Automata studies*, pages 3–41. Princeton University Press, Princeton, N. J., 1956.

[13] Hans-Peter Kolb, Jens Michaelis, Uwe Mönnich, and Frank Morawietz. An operational and denotational approach to non-context-freeness. *Theoretical Computer Science*, 293(2):261–289, 2003.

[14] Tore Langholm. A descriptive characterisation of indexed grammars. *Grammars*, 4(3):205–262, 2001.

[15] Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic CSL 1994*. Springer, Berlin, 1995.

[16] Markus Lohrey. On the parallel complexity of tree automata. In Aart Middeldorp, editor, *Rewriting Techniques and Applications, RTA 2001*, LNCS 2051, pages 201–215, Berlin, 2001. Springer.

[17] Nancy Lynch. Log space recognition and translation of parenthesis languages. *Journal of the ACM*, 24:583–590, 1977.

[18] Kurt Mehlhorn. Bracket languages are recognizable in logarithmic space. *Information Processing Letter*, 5:168–170, 1976.

[19] Yiannis Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland Publishing Company, 1974.

[20] Andreas Potthoff. *Logische Klassifizierung regulärer Baumsprachen*. PhD thesis, Christian-Albrechts-Universität zu Kiel, 1994.

[21] Geoffrey Pullum and Gerald Gazdar. Natural languages and context-free languages. *Linguistics and Philosophy*, 4:471–504, 1982.

[22] James Rogers. *A Descriptive Approach to Language Theoretic Complexity*. CSLI Publications, Stanford, CA, 1998.

[23] James Rogers. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1(3-4):265–305, 2003.

[24] Arnold Rosenberg. On multi-head finite automata. *IBM Journal of Research and Development*, 10:388–394, 1966.

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

[26] Balder ten Cate and Luc Segoufin. XPath, transitive closure logic, and nested tree walking automata. In *Proceedings of ACM SIGMOD/PODS Conference: Vancouver*, 2008.

[27] James W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[28] Hans-Jörg Tiede. Model theoretic syntax and transitive closure logic. In Leonid Libkin and Gerald Penn, editors, *LICS Workshop on Logic and Computational Linguistics*, 2003.