Logics For Context–Free Languages

Clemens Lautemann[1] Thomas Schwentick[1]
Denis Thérien[2]*

[1] Johannes Gutenberg–Universität Mainz
[2] McGill University Montreal

Abstract. We define *matchings*, and show that they capture the essence of context–freeness. More precisely, we show that the class of context–free languages coincides with the class of those sets of strings which can be defined by sentences of the form $\exists b\varphi$, where φ is first order, b is a binary predicate symbol, and the range of the second order quantifier is restricted to the class of matchings. Several variations and extensions are discussed.

1 Introduction

In descriptive complexity theory, we try to find logical characterisations of language classes which are relevant for Computer Science. The first such characterisation was given by Büchi [3], who showed that a set of strings is a regular language if, and only if, it is the set of finite models of a sentence of monadic second order logic. Since then, many important subclasses of regular languages have been characterised by means of syntactical restrictions, or extensions of first order logic, c.f. [2].

In this paper, we initiate a similar programme for context–free languages. In order to define any non–regular languages, we have to go beyond monadic second order logic. On the other hand, existential quantification over a single binary relation is enough to express all context–free languages – but also some languages which are not context–free. We therefore have to find a logic whose expressive power lies between that of monadic second order logic on the one, and of first order logic with one binary existential second order quantifier, on the other hand. We choose a *semantical* approach in which we restrict the second order quantifier to range only over a specified class of binary relations. To be more precise, let σ be a signature, B a binary predicate symbol, and a let $\mathcal{B}$ be a class of binary relations (such as, e.g., linear order relations, or equivalence relations). We define the class $\exists \mathcal{B}\, f.o.(\sigma)$ to consist of all those sets L of σ–structures for which there is a first order sentence over $\sigma<b>$, such that, for every σ–structure G,

$$G \in L \iff \text{there is a binary relation } B \in \mathcal{B} \text{ on } G, \text{with } <G, B> \models \varphi.$$

* Work done while on sabbatical leave at the Universidad Politecnica di Catalunya

In the following sections, we will use string logic. We consider strings over some alphabet $A = \{\alpha_1, \ldots, \alpha_r\}$ as structures over the signature $<A, <> := <Q_{\alpha_1}, \ldots, Q_{\alpha_r}, <>$ in the usual way: a string $w = w_1 \cdots w_n \in A^+$ is identified with the structure[3] $<\{1, \ldots, n\}, Q_{\alpha_1}, \ldots, Q_{\alpha_r}, <>$, where $i \in Q_{\alpha_j}$ iff $w_i = \alpha_j$. For the sake of clarity we will also use obvious abbreviations such as min and max (denoting the smallest and largest element, repectively). With *CFL(A)* we denote the class of context–free languages over the alphabet A.

2 Matchings

The essence of context–free languages is, in a sense, contained in the *Dyck* languages D_k, which consist of strings of properly matched parentheses with k different kinds of parentheses, c.f. [10]. Given some means of expressing that two positions in a string belong together, it is easy to describe the elements of D_k in first order logic. The binary predicates which provide this means are called *matchings*, and are defined as follows (c.f. Figure 1).

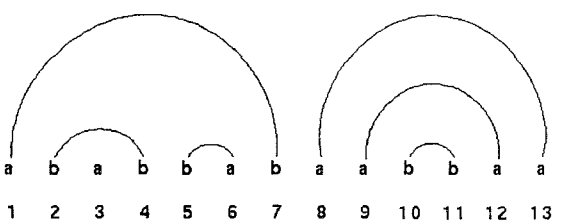


Fig. 1. Example of a matching.

A binary relation $B \subseteq \{1, \ldots, n\}^2$ is called a *pairing* relation, if, for all $i, j, k \in \{1, \ldots, n\}$ it satisfies

1. $(i,j) \in B \Longrightarrow i < j$ (B is compatible with $<$);
2. if $(i,j) \in B$ and $k \notin \{i,j\}$ then $(i,k), (k,i), (k,j), (j,k) \notin B$ (elements belong to at most one pair).

A pairing relation is called a *matching*, if it is *noncrossing*, i.e., if for all $i, j, k, l \in \{1, \ldots, n\}$,

1. $(i,j) \in B, (k,l) \in B, i < k < j \Longrightarrow i < l < j$.

If M is a matching, we call every pair $(i,j) \in M$ an *arch* of M.
Let Match denote the class of all matchings.

[3] Whenever appropriate, we use the same letters to denote predicate symbols and their interpretations in the structure.

With quantification over matchings, the Dyck language D_k over the alphabet $A_k := \{\alpha_1, \beta_1, \alpha_2, \ldots, \beta_k\}$ can be defined as follows:

$$\exists M \forall x \forall y \exists z : \Big(M(x,z) \vee M(z,x)\Big) \wedge \Big(M(x,y) \rightarrow \bigvee_{i=1}^{k} \Big(Q_{\alpha_i}(x) \wedge Q_{\beta_i}(y)\Big)\Big).$$

We will now show that first order logic plus existential quantification over matchings is enough to define every context–free language, and, moreover, no other languages can be defined that way.

2.1 Theorem. *CFL*(A) = $\exists$ *Match f.o.*$(A, <)$.

Proof of CFL$(A) \subseteq \exists$ Match *f.o.*$(A, <)$:
Here we have to show that, for every context–free language $L \in A^+$, there is a first order sentence φ over $<A, <, M>$, with the property that for all $w \in A^+$: $w \in L \iff$ there is a matching M such that $<w, M> \models \varphi$.
As a first step, we derive a normal form for context–free productions which is particularly convenient for our purposes. As a starting point for our normalisation, we take the following normal form (c.f. [10],ch.VI, exercise 4).

2.1.1 Lemma. *Every context–free language $L \subseteq A^+$ is generated by a context–free grammar $\mathcal{G} = (A, N, S, P)$, in which every production is of one of the following forms:*

1. $X \longrightarrow \alpha$, $X \in N$, $\alpha \in A$;
2. $X \longrightarrow \alpha u \beta$, *with* $X \in N$, $\alpha, \beta \in A$, *and* $u \in (N \cup A)^*$.

Condition 2 ensures that every right–hand side which allows further derivations begins and ends with a terminal symbol. It is these symbols which we are going to match. In order to avoid some unpleasant technicalities, we need a slightly stronger normal form. Let $p \equiv X_0 \longrightarrow v_0 X_1 v_1 \cdots v_{s-1} X_s v_s$ be a context–free production, where $X_0, \ldots, X_s$ are nonterminals, and $v_0, \ldots, v_s$ are (possibly empty) terminal strings. If $s > 0$, we call p a *nonterminal production.* Let $|$ be a new symbol. We call the string $v_0 | \cdots v_{s-1} | v_s$ the *pattern* of the production p. We want the left–hand side of a nonterminal production to be uniquely determined by its pattern. This can be achieved in the following way, starting from a grammar as in 2.1.1:

1. Eliminate all productions of the form $X \longrightarrow \alpha$, $X \in N \backslash \{S\}$, $\alpha \in A$, introduce a new production $Y \longrightarrow u\alpha v$, for every production $Y \longrightarrow uXv$.
2. Enumerate all nonterminal symbols $X_1, \ldots, X_r$. Starting with i=2, do the following for every i: as long as there is a nonterminal production $p \equiv X_i \longrightarrow v$ whose pattern also appears as the pattern of a production with left–hand side X_j, $j < i$, replace p by all productions which can be obtained from it by substituting one of the nonterminals in v in all possible ways.

This process will eventually terminate, since the substitution in 2 will either make the production terminal, or it will increase its length. Thus we conclude:

2.1.2 Lemma. *Every context-free language $L \subseteq A^+$ is generated by a context-free grammar $\mathcal{G} = (A, N, S, P)$ which satisfies the following conditions:*

- *All productions are of one of the following forms*
  1. $S \longrightarrow \alpha$, $\alpha \in A$, *or*
  2. $X \longrightarrow \alpha u \beta$, $X \in N$, $\alpha, \beta \in A$, $u \in (A \cup N)^*$.
- *If any two nonterminal productions have the same pattern, they have the same left-hand side.*

Let L be generated by a grammar as in 2.1.2 and consider a derivation tree T of a string $w \in L$. The leftmost and the rightmost child of each internal node of T are leaves, labeled with terminal symbols. Define a pairing M_T on w by (c.f. Figure 2):

$(i,j) \in M_T \iff$ i corresponds to the leftmost, j to the rightmost child of the same internal node of T.

Clearly, M_T is a matching.

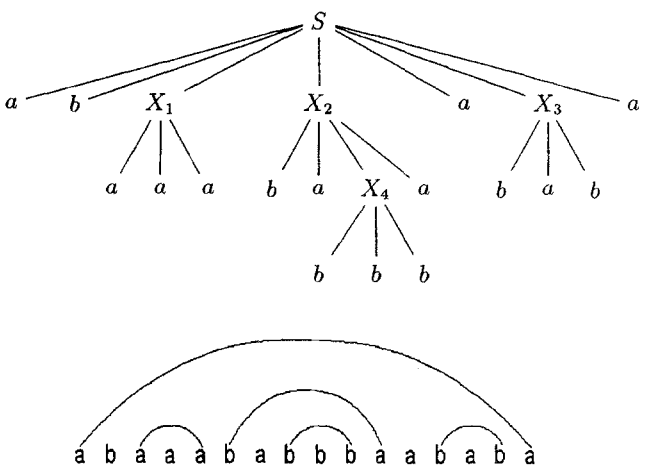


Fig. 2. The matching constructed from a derivation tree.

We show now that there is a formula φ over $<A, <, M>$, which holds for a string w with matching M, if, and only if, there is a $\mathcal{G}$-derivation tree T of w such that $M = M_T$. It follows that there is a matching M on w with $<w, M> \models \varphi$ iff w can be derived in $\mathcal{G}$.

Every arch $(i,j) \in M$ defines a substring, $w_i, \ldots, w_j$. We say that an arch $(k,l) \in M$ $(i < k < l < j)$ lies at the *surface* of (i,j), if there is no other arch between it and (i,j), i.e., if there is no arch (r,s) with $i < r < k < s$. Similarly, a position k $(i \leq k \leq j)$ which is not the endpoint of an arch other than (i,j) lies at the *surface* of (i,j), if there is no other arch between it and (i,j), i.e., if there

is no arch (r,s) with $i<r<k<s$.The string of surface symbols, with surface arches replaced by the symbol | is called the *pattern* of (i,j), c.f. Figure 3.

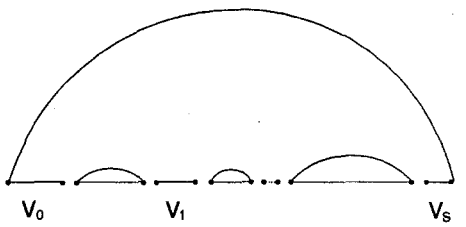


Fig. 3. The surface of an arch, with pattern $v_0|v_1|\ldots|v_s$.

We say that an arch (i,j) *corresponds to* a production p, if their patterns are identical. This property of a pair of positions can easily be expressed by a first order formula $\chi_p(x,y)$, for every production p:
Let, for every string v, $\psi_v(x,y)$ be a formula which expresses that the string strictly between positions x and y is v, i.e,
$<w,i,j> \models \psi_v(x,y) \iff w_{i+1}\cdots w_{j-1}=v$.
Then, for $p \equiv X_0 \longrightarrow \alpha v_0 X_1 v_1 \cdots v_{s-1} X_s v_s \beta$, $\chi_p(x,y)$ can be formed as:

$$\chi_p(x,y) \equiv Q_\alpha(x) \wedge Q_\beta(y) \wedge \exists x_1 \exists y_1 \ldots \exists x_s \exists y_s : \big(x<x_1<y_1<x_2<\ldots<y_s<y\big) \wedge$$
$$\wedge\Big(\psi_{v_0}(x,x_1) \wedge \psi_{v_1}(y_1,x_2) \wedge \ldots \wedge \psi_{v_s}(y_s,y)\Big) \wedge$$
$$\wedge\Big(M(x_1,y_1) \wedge \ldots \wedge M(x_s,y_s)\Big).$$

For every $X \in N$, let $\tilde{\chi}_X(x,y)$ be the disjunction of all those $\chi_q(x,y)$, for which X is the left–hand side of the production q. Then, for p as above, the following formula $\hat{\chi}_p(x,y)$ expresses that (x,y) corresponds to p *and* that the surface arches $(x_1,y_1),\ldots,(x_s,y_s)$ correspond to productions with left–hand sides $X_1,\ldots,X_s$, respectively.

$$\hat{\chi}_p(x,y) \equiv Q_\alpha(x) \wedge Q_\beta(y) \wedge \exists x_1 \exists y_1 \ldots \exists x_s \exists y_s : \big(x<x_1<y_1<x_2<\ldots<y_s<y\big) \wedge$$
$$\wedge\Big(\psi_{v_0}(x,x_1) \wedge \psi_{v_1}(y_1,x_2) \wedge \ldots \wedge \psi_{v_s}(y_s,y)\Big) \wedge$$
$$\wedge\Big(M(x_1,y_1) \wedge \ldots \wedge M(x_s,y_s)\Big) \wedge \Big(\tilde{\chi}_{X_1}(x_1,y_1) \wedge \ldots \wedge \tilde{\chi}_{X_s}(x_s,y_s)\Big).$$

Now the formula φ with the property that w has a matching M such that $<w,M> \models \varphi$ if and only if $w \in L$, is formed as follows[4]:

$$\bigvee_{S\to u\in P} \tilde{\varphi}_u \vee \Big[\Big(\forall x \forall y : M(x,y) \to \bigvee_{p\in P} \hat{\chi}_p(x,y)\Big) \wedge \Big(M(\min,\max) \wedge \tilde{\chi}_s(\min,\max)\Big)\Big].$$

φ expresses that either $S \longrightarrow w$ or

[4] Here $\tilde{\varphi}_u$ is a formula which holds for w iff $w=u$.

1. for every arch (i,j) of M:
   - (i,j) corresponds to a production $X_0 \longrightarrow \alpha v_0 X_1 v_1 \cdots X_s v_s \beta$, and
   - the surface arches $(i_1,j_1),\ldots,(i_s,j_s)$ correspond to productions $p_1,\ldots,p_s$ with left–hand sides $X_1,\ldots,X_s$, respectively;
2. $(1,n)$ is in M and corresponds to a production with left-hand side S.

Since every production is uniquely determined by its pattern, the two conditions under 1 must be consistent, and we can see inductively, that $(i,j)\in M$ implies that $w_i \cdots w_j$ can be derived from some nonterminal X. □

Proof of $\exists$Match $f.o.(A,<) \subseteq$ CFL(A):
In order to prove this direction, we need some notation for trees. For our purposes, trees are rooted and ordered (in a leftmost depth–first way), and have labeled nodes, where each node label has an *arity* which corresponds to the out-degree of the node. A *tree language* is a set of trees over some finite label set. The set A of 0–ary labels is called the *leaf alphabet.* The leaf labels of a tree T with leaf alphabet A, concatenated according to the order relation (i.e., from left to right) form a string over A, called the *yield* of T.
As a relational structure, the universe of a tree is the set of its nodes; there is a unary relation Q_α, for every label α, there is a binary relation C, with $(i,j) \in C \iff i$ is a child of j, and, finally, there is the order relation $<$. With predicate symbols for these relations, we can easily express the following properties in first order logic:
$Lf(i)$: i is a leaf;
$Lc(i,j)$: i is the leftmost child of j;
$Rc(i,j)$: i is the rightmost child of j;
and in monadic second order logic[5]
$An(i,j)$: i is an ancestor of j;
$Pt(U,i,j)$: $An(i,j)$ and $Lf(j)$ and the set U contains precisely the nodes on the path from i to j.

In what follows, we will make use of two characterisations, of context–free languages, and of recognisable tree languages, respectively.[6]

2.1.3 Lemma. *[9] A language L over A is context–free if, and only if, there is a recognisable tree language $\mathcal{T}$, with leaf alphabet A, such that every $w \in A^+$ belongs to L iff it is the yield of some tree $T \in \mathcal{T}$.*

2.1.4 Lemma. *[5, 13] A tree language is recognisable iff it is definable in monadic second order logic.*

Combining these two results, we can proceed as follows.
Given a first order sentence ψ over $<A,<,m>$ we will construct a monadic second order sentence Φ over some tree signature, in such a way that a string

[5] This is first order logic with additional (existential and universal) quantification over unary predicates.
[6] For the notion of a recognisable tree language, see e.g. [8].

w has a matching M with $<w, M> \models \psi$, if and only if w is the yield of a tree which satisfies Φ.

Φ will be of the form $\Psi \wedge \Upsilon$, where Υ describes a class of trees which can be obtained from strings with matchings in a certain way that will be explained below. The formula Ψ corresponds to ψ, and holds for those trees which are obtained from strings with matchings that satisfy ψ.

Let $w = w_1 \cdots w_n$ be a string, and M a matching on w. We construct a tree $T_{w,M}$ in two stages as follows:

1. Define a tree $\tilde{T}_{w,M}$ the nodes of which are the positions of w and the arches of M:
   - for every $i \leq n$, i is a leaf, labeled w_i;
   - every arch $(i, j) \in M$ is an internal node, labeled $\otimes$, its children are all those positions k and arches (k, l) which are at the surface of (i, j), in the order in which they appear in $<w, M>$.

   If $n > 1$, and $(1, n) \notin M$, add a root, labeled $\odot$, whose children are all those positions and arches which are not underneath any arch.
2. Whenever an internal node has more than two children, we distribute them over binary subtrees,[7] using additional nodes with label $\odot$. This results in a tree $T_{w,M}$, with yield w and with internal nodes labeled $\odot$ or $\otimes$.

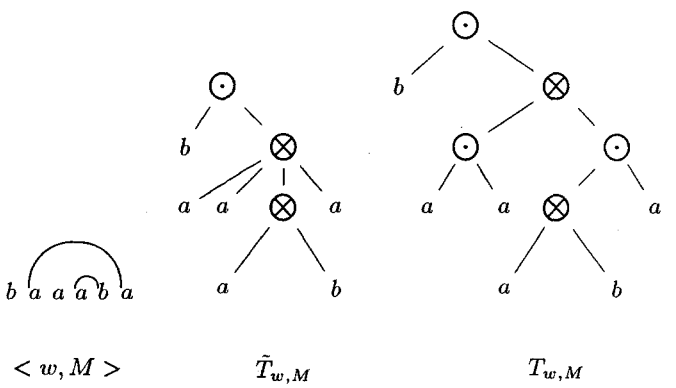


Fig. 4. The construction of the tree.

Thus our trees have leaf alphabet A and two binary labels, $\odot$ and $\otimes$. Among all such trees the ones that can be obtained in the way described above are characterised by the following property:

> Both, the leftmost and the rightmost path out of of every $\otimes$–labeled node lead to leaves, without meeting any $\otimes$–labeled nodes.

[7] We don't need this construction to be deterministic – any tree $T_{w,M}$ obtained this way will do.

This property can be expressed in a monadic second order formula over the tree signature $<Q_{\alpha_1}, \ldots Q_{\alpha_r}, Q_\otimes, Q_\odot, C, <>$ as $\Upsilon = \Upsilon_L \wedge \Upsilon_R$, where Υ_L deals with the leftmost, Υ_R with the rightmost path, respectively. For Υ_L, e.g., we can write:

$$\forall x \Big[Q_\otimes(x) \rightarrow \exists y \exists X \forall s \forall t : Pt(X,x,y) \wedge \Big(\big(X(s) \wedge X(t) \wedge C(s,t)\big) \rightarrow Lc(s,t) \Big) \wedge$$
$$\wedge \Big(\big(X(s) \wedge Q_\otimes(s)\big) \rightarrow s{=}x \Big) \Big].$$

If M is a matching on w, there is a tree T with yield w, which satisfies Υ, namely $T_{w,M}$. On the other hand, if T satisfies Υ, then a matching M_T on the yield of T can be constructed by defining:

$(i,j) \in M_T$, iff c, the common ancestor of i and j in T, has label $\otimes$, and i lies on a leftmost, j on a rightmost path from c.

We want to restrict T so that $<w, M_T> \models \psi$. This is done by the formula Ψ which is derived from ψ by

- restricting all quantifiers to range over leaves only; to this end, we replace every subformula of the form $\exists x \chi$ by $\exists x : Lf(x) \wedge \chi$, and every occurence of $\forall x \chi$ by $\forall x : Lf(x) \rightarrow \chi$;
- replacing $x < y$ by the monadic second order formula

$$\exists n \exists l \exists r : Lc(l,n) \wedge Rc(r,n) \wedge \Big(l{=}x \vee An(l,x) \Big) \wedge \Big(r{=}y \vee An(r,y) \Big)$$

  which expresses that there is a node n whose left child is x or an ancestor of x, and whose right child is y or an ancestor of y;
- replacing $m(x,y)$ by a monadic second order formula (similar to Υ) which expresses that there is a node n with label $\otimes$, such that the leftmost path from n leads to x and the rightmost path from n leads to y.

Then there is a matching M on w such that $<w, M> \models \psi$, if, and only if, $w = yield(T)$ for some T such that $T \models \Upsilon \wedge \Psi$. Hence the set of those w for which there is such a matching is context–free. □

The construction in the second half of the proof of Theorem 2.1 still yields a monadic second order tree formula Ψ if the string formula ψ is monadic second order rather than first order. Therefore, the theorem remains true with f.o. replaced by m.s.o.

2.1.1 Corollary. $CFL(A) = \exists\, Match\, m.s.o.(A, <)$. □

3 Variations

One motivation for this research was an attempt to find logical characterisations for interesting subclasses of context–free languages. With our approach, this is not difficult for the class of *k–linear* languages, i.e., languages generated by grammars with rules of the form

- $S \longrightarrow X_1 \cdots X_k$, $X_1, \ldots, X_k \in N \backslash \{S\}$.
- $X \longrightarrow uYv$, $u, v \in A^*$, $X \in N$, $Y \in N \backslash \{S\}$;

These languages can be characterised by concatenations of at most k strictly nested matchings, where we call a matching M on $\{1, \ldots, n\}$ *strictly nested* if there are no two pairs $(i,j),(k,l) \in M$ such that $i < j < k < l$.

There are other classes of binary relations which can be used to characterise context–free languages. E.g., the proof of Theorem 2.1 can easily be modified to show $\mathrm{CFL}(A) = \exists \mathcal{B}\, f.o.(A, <)$, where the class $\mathcal{B}$ consist of those binary relations B that satisfy the following noncrossing condition:

$$\text{if } j,k \in \{i{+}1, \ldots, l{-}1\}, \text{ and } (i,j),(k,l) \in B \text{ then } j < k.$$

Here we allow several arches to share the same left or right endpoint; note, however, that no position can serve both as a left and a right endpoint.

We can also employ special order relations to define context–freeness. Call a linear order relation $\sqsubset$ on $\{1, \ldots, n\}$ *tree definable* if there is a binary tree T such that

- the leaves of T are, from left to right, the numbers $1, \ldots, n$;
- the internal nodes are labeled either $\swarrow$ or $\searrow$;
- $i \sqsubset j$ iff i is visited before j in the depth–first traversal of T in which, at every node with label $\swarrow$, first the left, and at every node with label $\searrow$, first the right child is visited.

As there are only $2^{O(n)}$ binary trees with n leaves, not all linear orders can have this property. Let TDO denote the class of tree definable linear orderings.

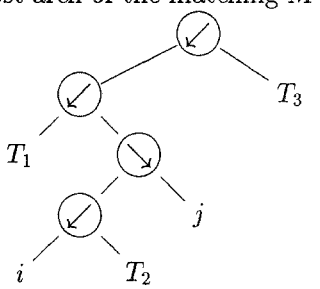
3.1 Theorem. $CFL(A) = \exists TDO\, f.o.(A, <)$[8].

Proof (sketch):
Let $<w, \sqsubset> \models \varphi$, and let T be a tree defining the order relation $\sqsubset$ on $\{1, \ldots, n\}$. We can express $x \sqsubset y$ as $\exists z : lca(z,x,y) \wedge \Big(\big(x{<}y \wedge Q_{\swarrow}(z) \big) \vee \big(y{<}x \wedge Q_{\searrow}(z) \big) \Big)$, where the mso formula $lca(z,x,y)$ expresses that z is the least common ancestor of x and y.
As in the proof of the second half of Theorem 2.1 we can construct an mso tree formula Φ such that for the tree T' obtained from T by labeling the leaves with the symbols of w it holds that $T' \models \Phi$, and that for every tree T with $T \models \Phi$, the order relation $\sqsubset$ defined by T satisfies $<yield(T), \sqsubset> \models \varphi$. This shows that $\exists TDO\, f.o.(A, <) \subseteq \mathrm{CFL}(A)$.
For the other direction, let $L \in \mathrm{CFL}(A)$, and let L be defined by $\exists m\, \varphi(m)$, according to Theorem 2.1. From a matching M on $\{1, \ldots, n\}$ we derive a successor relation $\boxdot$ as follows:

[8] Note that formulas of this logic refer to *two* linear order relations: the given positional ordering $<$, and the additional (tree definable) order.

- If j is the right endpoint of an arch (i,j) then $j \boxdot i$;
- otherwise let k be the smallest number $> j$ which is not the right endpoint of an arch.
  - If k is the left endpoint of an arch (k,l) then $j \boxdot l$;
  - otherwise $j \boxdot k$.

Then (i,j) is an arch of M iff $i<j \wedge j \sqsubset\!\!\cdot\, i$. Since the defining properties of matchings are first–order definable, we can derive a f.o. formula $\tilde{\varphi}(\sqsubset)$ from φ, such that $<w,M> \models \varphi \iff <w,\sqsubset> \models \tilde{\varphi}$. Here $\sqsubset$ is the order relation induced by $\boxdot$.
It remains to show that $\sqsubset$ is tree definable. This can be shown by induction on n. For the induction step let (i,j) be an outermost arch of the matching M on $\{1,\ldots,n\}$. Then the following tree defines M.



Here T_1, T_2, T_3 are trees defining the order relations derived from the restrictions of M to $\{1,\ldots,i-1\}, \{i+1,\ldots,j-1\}$, and $\{j+1,\ldots,n\}$, respectively. □

Theorem 3.1 implies that all context–free languages are contained in the class $\exists\, LO\, f.o.(A,<)$, where LO stands for the class of *all* linear order relations, since the property of a linear order relation to be tree definable can be expressed in first order logic. However, this is not immediately obvious, and we will not prove it here. But it is not difficult to show directly that linear order relations can express all context–free languages.

3.2 Proposition. $CFL(A) \subseteq \exists\, LO\, f.o.(A,<)$. □

This inclusion is strict, since the set $L = \{vv \,/\, v \in A^+\}$, a well–known example of a non–context free language, is contained in $\exists\, LO\, f.o.(A,<)$. To see this, let $w = w_1 \cdots w_{2n}$ be equipped with the additional order relation $\prec$ induced by $1 \prec n{+}1 \prec 2 \prec n{+}2 \prec \cdots \prec n \prec 2n$. With this order it holds that $w \in L \iff <w,\prec> \models \forall x \forall y : \Big(x < y \wedge y = suc_{\prec}(x)\Big) \rightarrow \bigwedge_{a \in A} \Big(Q_a(x) \leftrightarrow Q_a(y)\Big)$.[9]
In combination with the following formula, which uniquely determines $\prec$, this formula characterises L:

$$\Big(\min_{\prec} = \min_{<}\Big) \wedge \forall x \forall y : \Big(y = suc_{\prec}(suc_{\prec}(x)) \rightarrow y = suc_{<}(x)\Big).$$

[9] It is only for the sake of conciseness that we use a function symbol $suc_{\prec}$ here to express the direct successor in the order relation $\prec$. Of course, the successor can easliy be expressed in first order logic without this symbol.

4 Discussion

Of course, our main result does not teach us anything new about context–free languages. However, it supports, in a rather satisfying mathematical way, the intuition that the essence of context–freeness is contained in the notion of a matching relation. And since context–free languages are well understood, it tells us something about our logic $\exists$Match *f.o.*, e.g., that it is not closed under negation. We believe that semantically restricted logics can be used to characterise not only the class of context–free languages, and some of its subclasses, but also many other interesting classes. We are convinced that a more general study of these logics will prove worthwhile also in the context of general finite structures, instead of strings. Here, as opposed to the string case, the class monNP defined by existential monadic second order logic is not closed under complementation. In fact, the set of connected graphs, although expressible by means of a *universal* monadic second order sentence, is not contained in monNP [6], a result which has since been refined and extended in a number of ways [4, 1, 7, 11, 12]. On the other hand, the expressive power of sentences with one *binary* existential second order quantifier is not well understood. We believe that studying semantically restricted versions of this latter logic, will help us understand the limitations, and the expressive power of binary existential second order quantification.

References

1. M. Ajtai and R. Fagin. Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic*, 55(1):113–150, 1990.
2. D. A. M. Barrington, K. Compton, H. Straubing, and D. Thérien. Regular languages in NC^1. *Journal of Computer and System Sciences*, 44:478–499, 1992.
3. J. R. Büchi. Weak second order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
4. M. de Rougemont. Second-order and inductive definability on finite structures. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 33:47–63, 1987.
5. J. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.
6. R. Fagin. Monadic generalized spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 21:89–96, 1975.
7. R. Fagin, L. J. Stockmeyer, and M. Y. Vardi. On monadic NP vs. monadic Co-NP. In *Proc. 8th Annual Conference Structure in Complexity Theory*, pages 19–30, 1993. To appear in *Information and Computation*.
8. F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, 1984.
9. J. Mezei and J. B. Wright. Algebraic automata and context–free sets. *Information and Control*, 11:3–29, 1967.
10. A. Salomaa. *Formal Languages*. Academic Press, 1987.
11. T. Schwentick. Graph connectivity and monadic NP. In *Proc. 35st IEEE Symp. on Foundations of Computer Science*, pages 614–622, 1994.
12. T. Schwentick. Graph connectivity, monadic NP and built–in relations of moderate degree. In *Proc. 22nd International Colloq. on Automata, Languages, and Programming*, 1995. to appear.

13. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second–order–logic. *Mathematical System Theory*, 2:57–81, 1968.