CS 2428 - Logic and Automata

Lecture #3: 27 January 2005

Lecturer: Leonid Libkin

Scribe Notes by: Solmaz Kolahi

1 *MSO* on trees and regular tree languages

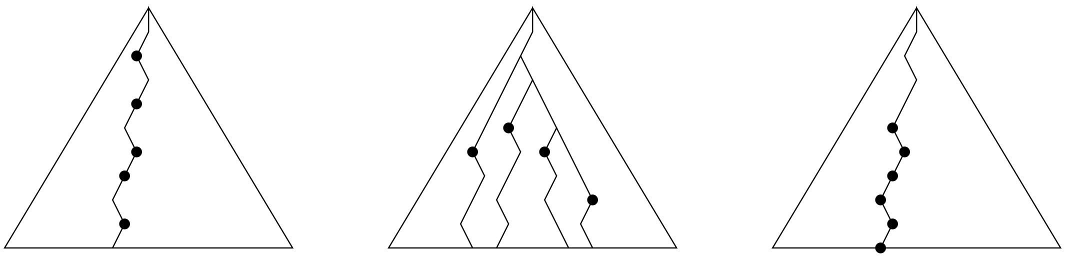
Theorem 1 (Thatcher, Wright, Doner 1968-1970) *A tree language is regular iff it is definable in MSO.*

The above theorem states that like regular string languages, regular tree languages can be characterized by formulae in monadic second-order logic. The proof is very similar to that of Büchi's theorem for strings. As a corollary of this theorem, we have:

Corollary 2 *Over trees, MSO = EMSO.*
(*EMSO* = Existential *MSO*: all formulae are of the form $\exists X_1 \dots \exists X_n\ \varphi$, where $\varphi \in FO$)

Definition Given a tree domain D, we define the following concepts:

- ***Chain*** is a subset $S \subseteq D$ such that $\forall x\ \forall y\ x, y \in S \rightarrow x \prec y\ \vee\ y \prec x$, where $\prec$ is the *prefix* relation.
- ***Antichain*** is a subset $S \subseteq D$ such that $\forall x\ \forall y\ x, y \in S \rightarrow x \not\prec y\ \wedge\ y \not\prec x$.
- ***Path*** is a chain S such that if $x \prec y \prec z$ and $x, z \in S$, then $y \in S$. Moreover, if $x \in S$ is not a leaf, then either $x0$ or $x1$ is in S.



Chain Antichain Path

By restricting the quantification over particular sets in a tree, we can define different fragments of *MSO*.

Definition *Monadic chain, antichain, path logic* is *MSO*, in which second-order quantifiers range only over chains, antichains, and paths respectively.

Observation Monadic chain, antichain, path logic $\subseteq$ *MSO*.

This is because we have formulae $\varphi_*(X)$ such that $\varphi_*(X)$ is true iff X is a $*$, where $*$ is chain, antichain, or path. Moreover, the prefix relation $\prec$ is also *MSO*-definable (this is left as an exercise).

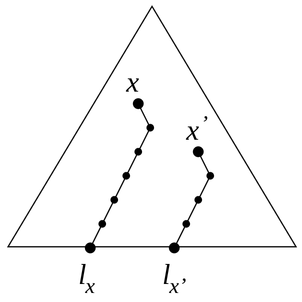
Lemma 3 *Over finite trees, monadic path logic = FO in vocabulary* $(<_0, <_1, \prec, (P_a)_{a\in\Sigma})$.

Proof sketch. It suffices to show that every formula in monadic path logic can be translated into a first-order formula. The idea is that each path p is uniquely determined by its start and end points. Therefore, given a formula in monadic path logic, we replace every $\exists X$ by ($\exists s_X \exists e_X$ e_X is a leaf) and every occurrence of $X(y)$ by $(s_X \preceq y \wedge y \preceq e_X)$.

Theorem 4 *Over trees, monadic antichain logic = MSO.*

Proof sketch. Since we have already observed that monadic antichain $\subseteq$ *MSO*, it suffices to show that every formula in *MSO* can be translated into a formula in monadic antichain logic. The idea is that every subset X of tree domain D can be encoded into a pair (X_1, X_2), such that X_1, X_2 are antichains, and both the encoding $X \rightsquigarrow (X_1, X_2)$ and the decoding $(X_1, X_2) \rightsquigarrow X$ are *MSO*-definable.

Given $X \subseteq D$, we construct the pair (X_1, X_2) as follows. For each $x \in X$, such that x is not a leaf, X_1 contains l_x, where l_x is a leaf obtained by first moving from x to its right child and then keep moving to left until we reach the leaf l_x (see the figure below). Note that the decoding is possible since each l_x uniquely represents a non-leaf x when we disallow unary branching. The other set, X_2, simply contains all of the leaves in X.



Theorem 5 *FO* $\subsetneq$ *monadic chain logic* $\subsetneq$ *monadic antichain logic = MSO.*

In the proof sketch that we presented for Theorem 4, we observed that the argument would not be correct if we allowed unary branching in the tree. However, we still need to investigate the validity of the result for the case of trees with unary branching.

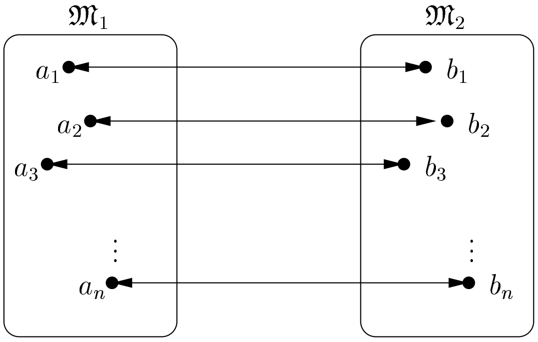
We can consider strings as a class of trees that only allow unary branching. Since in such trees there is no antichain of size greater than one, a monadic antichain formula can only quantify over sets containing a single tree position and hence can be rewritten as a first-order formula. Therefore, if the result of Theorem 4 was valid for this case, we would have *FO* = *MSO* over strings.

To compare the expressiveness of *FO* and *MSO* we use *Ehrenfeucht-Fraïssé games*, which are briefly introduced in the next section. For more details refer to [1], Chapter 3.

2 Ehrenfeucht-Fraïssé games

Ehrenfeucht-Fraïssé games provide us with a nice tool for describing expressiveness of logics over finite (or infinite) models. In each game, there are two players called the *spoiler* and the *duplicator* (or simply player I and player II). The board of the game consists of two structures $\mathfrak{M}_1$ and $\mathfrak{M}_2$ of vocabulary σ. The goal of the spoiler is to show that these two structures are different, while the goal of the duplicator is to show that they are the same.

At each round of the game, the spoiler makes a move by picking an element from either of the structures $\mathfrak{M}_1$ or $\mathfrak{M}_2$. Then the duplicator responds by picking an element in the other structure.



Let $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ be the moves after n rounds of an Ehrenfeucht-Fraïssé game. We define partial function $f : \mathfrak{M}_1|_{\{a_1,\ldots,a_n\}} \to \mathfrak{M}_2|_{\{b_1,\ldots,b_n\}}$, such that for each $1 \leq i \leq n$ we have $f(a_i) = b_i$. We say that $(\vec{a}, \vec{b})$ is a *winning position for the duplicator* if f is a partial isomorphism.

If the duplicator can play in a way that guarantees a winning position after n rounds, no matter how the spoiler plays, we say that the duplicator has an *n-round winning strategy*, written as $\mathfrak{M}_1 \equiv_n \mathfrak{M}_2$.

Example 1 Let $\sigma = \emptyset$, and $|m_1|, |m_2| \geq n$, i.e. the structures are two simple sets of size greater than n. We present a winning strategy for the duplicator to show $\mathfrak{M}_1 \equiv_n \mathfrak{M}_2$. Suppose after playing i rounds, the position is $((a_1, \ldots, a_n), (b_1, \ldots, b_n))$, and the spoiler picks $a_{i+1} \in m_1$ at the next round. If $a_{i+1} = a_j$ for some $j \leq i$, the duplicator responds with $b_{i+1} = b_j$; otherwise he responds with any fresh $b_{j+1} \in m_2$ that has not been played so far. Since the sizes of the two structures are at least n, the duplicator can keep the winning position until after round n.

Example 2 In the next canonical example, we let $\sigma = \{<\}$. Suppose $\mathfrak{M}_1, \mathfrak{M}_2$ are two linear orders of size at least 2^n, i.e. they are structures of the form $(\{1, \ldots, m\}, <), m \geq 2^n$. We want to show that $\mathfrak{M}_1 \equiv_n \mathfrak{M}_2$.

Let $d(x, y)$ denote the distance of elements x, y in the linear order $\mathfrak{M}_1$ or $\mathfrak{M}_2$. The idea is that the duplicator can play in such a way that the following hold after round i for each $j, l \leq i$:

1. if $d(a_j, a_l) < 2^{n-i}$, then $d(b_j, b_l) = d(a_j, a_l)$.

2. if $d(a_j, a_l) \geq 2^{n-i}$, then $d(b_j, b_l) \geq 2^{n-i}$.

3. $a_j \leq a_l \Longleftrightarrow b_j \leq b_l$.

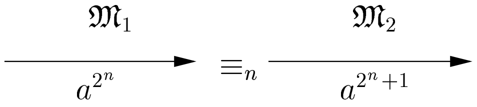
This can be shown by induction on i. Refer to [1], Chapter 3 for more details.

Theorem 6 *Let $\mathfrak{M}_1$ and $\mathfrak{M}_2$ be two structures in a relational vocabulary. The following are equivalent:*

1. $\mathfrak{M}_1 \equiv_n \mathfrak{M}_2$.
2. $\mathfrak{M}_1$ *and* $\mathfrak{M}_2$ *agree on all FO sentences of* $qr \leq n$.
3. $tp_n^{FO}(\mathfrak{M}_1) = tp_n^{FO}(\mathfrak{M}_2)$. $(tp_n^{FO}(\mathfrak{M}) = \{\varphi \in FO \mid qr(\varphi) \leq n,\ \mathfrak{M} \models \varphi\})$.

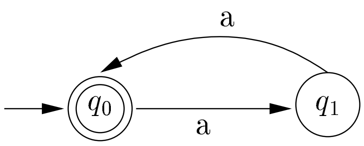
Corollary 7 *Language $(aa)^*$ is not FO-definable.*

Proof. Assume there is a formula $\varphi \in FO$ defining the language $(aa)^*$. Let $n = qr(\varphi)$ and $\mathfrak{M}_1, \mathfrak{M}_2$ be two string structures over vocabulary $\{\prec\}$ (interpreted as prefix relation) of size 2^n and $2^n + 1$ respectively:



The two structures that we have defined look very similar to those of Example 2, so likewise, we can show that $\mathfrak{M}_1 \equiv_n \mathfrak{M}_2$. By Theorem 6, we have $\mathfrak{M}_1 \models \varphi \iff \mathfrak{M}_2 \models \varphi$. This is obviously not true since $a^{2^n} \in L((aa)^*)$ and $a^{2^n+1} \notin L((aa)^*)$. Therefore, the language $(aa)^*$ cannot be defined by any FO formula.

algebraic proof. Let $\mathcal{A} = (Q, q_0, F, \delta)$ be the minimal automaton accepting the language $(aa)^*$:



For each string $s \in \Sigma^*$, we define a function $f_s : Q \to Q$ such that $f_s(q) = q'$ if when starting from q with input s, we arrive at state q'. Note that in this case, f_s has only two alternatives: $f_s^1(q_0) = q_1, f_s^1(q_1) = q_0$ or $f_s^2(q_i) = q_i$ for $i \in \{0, 1\}$.

The monoid of automaton $\mathcal{A}$, defined as $\mathcal{M}(\mathcal{A}) = ((f_s)_{s \in \Sigma^*}, f_\epsilon, \circ)$, is a field of two elements and therefore contains a group. Hence $(aa)^*$ is not $*$-free, which means it cannot be defined by any FO formula.

Since we know that the language $(aa)^*$ is MSO-definable, immediately we have:

Corollary 8 $FO \subsetneq MSO$.

3 Top-down tree automata

Definition A non-deterministic top-down tree automaton is a tuple $\mathcal{A} = (Q, Q_0, F, \delta : Q \times \Sigma \to 2^{Q \times Q})$, where Q is a finite set of states, $Q_0 \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final (accepting) states, and δ is the transition function.

Given a tree $T = (D, \lambda)$, a *run* of $\mathcal{A}$ on T is a function $\rho : D \to Q$ such that:

1. if $s \in D$ is the root labeled a, then $\rho(s) \in \delta(q_0, a)$, $q_0 \in Q_0$.

2. if $s \in D$, $s0, s1 \in D$, and $\lambda(s) = a$, then $(\rho(s0), \rho(s1)) \in \delta(\rho(s), a)$.

A run ρ is *accepting* iff for all $s \in D$ such that s is a leaf, $\delta(\rho(s), \lambda(s)) \in F$.

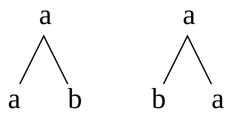
Theorem 9 *Bottom-up and top-down non-deterministic tree automata are equivalent.*

Intuitively, bottom-up and top-down tree automata are the same thing. We just reverse the notion of initial and accepting states.

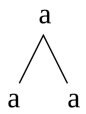
A top-down tree automaton is deterministic if the transition function is of the form $\delta : Q \times \Sigma \to Q \times Q$. In previous lecture, we saw that non-deterministic bottom-up automata are equivalent to deterministic bottom-up automata. However, here we show that this is not the case for top-down tree automata.

Lemma 10 *Deterministic top-down tree automata $\neq$ non-deterministic top-down tree automata.*

Proof. Consider tree language L containing only the following trees:



Assume there is a deterministic top-down automaton $\mathcal{A}$ that accepts L. It is easy to see that $\mathcal{A}$ also accepts the following tree:



Therefore, there is no deterministic top-down automaton $\mathcal{A}$ such that $L(\mathcal{A}) = L$. Since this language can be accepted by a non-deterministic top-down automaton, we have shown that deterministic automata are weaker.

4 Satisfiability of *MSO* formulae

The satisfiability problem of *MSO* formulae on strings (or trees) is defined as follows:

Problem 1 *Given an MSO sentence φ, does it have a string (or tree) model, i.e. $\exists s\, \mathfrak{M}_s \models \varphi$?*

The above problem is decidable. Given an *MSO*-formula φ, we can construct NFA (or NTA) $\mathcal{A}_\varphi$ such that $s \in L(\mathcal{A})$ iff $\mathfrak{M}_s \models \varphi$. Then the satisfiability problem is equivalent to deciding whether $L(\mathcal{A})$ is non-empty, which is NLOG-complete for strings and PTIME-complete for trees. The following table summarizes the complexity of some decision problems for NFAs and NTAs.

	Emptiness $L(\mathcal{A}) \stackrel{?}{=} \emptyset$	**Universality** $L(\mathcal{A}) \stackrel{?}{=} \Sigma^*$	**Intersection** $s \stackrel{?}{\in} \bigcap_i L(\mathcal{A}_i)$	**Containment** $L(\mathcal{A}_1) \stackrel{?}{\subseteq} L(\mathcal{A}_2)$	**Equivalence** $L(\mathcal{A}_1) \stackrel{?}{=} L(\mathcal{A}_2)$
NFA	NLOG-Complete	PSPACE-Complete	PSPACE-Complete	PSPACE-Complete	PSPACE-Complete
NTA	PTIME-Complete	EXPTIME-Complete	EXPTIME-Complete	EXPTIME-Complete	EXPTIME-Complete

Note that while Problem 1 is decidable, the following satisfiability problems are not. In fact, Problem 3 is not even recursively enumerable.

Problem 2 *Given an FO sentence φ, does it have a model?*

Problem 3 *Given an FO sentence φ, does it have a finite model?*

5 Decidability of WS1S and Presburger Arithmetic

Definition The *weak second-order theory of one successor (WS1S)* consists of the structure $(\mathbb{N}, succ)$, where *succ* is the successor function interpreted as $succ(n) = n + 1$, and *MSO* formulae in which the second-order quantifiers range only over finite sets ($\exists X\ \varphi$ is interpreted as "for some finite X, $\varphi(X)$").

Problem 4 *Given a sentence φ of WS1S, is φ true (i.e. $(\mathbb{N}, succ) \models \varphi$)?*

The above problem is decidable. In fact, we will see in the next lecture that given $\varphi \in$ WS1S, one can construct a string automaton $\mathcal{A}_\varphi$ such that φ is true iff $\epsilon \in L(\mathcal{A}_\varphi)$.

Definition *Presburger Arithmetic* is the first-order theory of natural numbers containing addition but no multiplication. That is, it consists of the structure $(\mathbb{N}, +)$ and *FO* formulae.

Problem 5 *Given a sentence φ of Presburger Arithmetic, is φ true (i.e. $(\mathbb{N}, +) \models \varphi$)?*

We show that Problem 5 is decidable by translating an *FO* formulae of Presburger Arithmetic into an *MSO* formula of WS1S. However, it is well-known that if we add multiplication, the problem will be highly undecidable (Gödel).

If we represent numbers in binary notation, we can have a function g that maps every natural number n to a finite subset of $\mathbb{N}$, such that $g(n) =$ set of positions that are set to one in the binary representation of n:

$$13 \to \ 1 \quad 1 \quad 0 \quad 1 \ \to \begin{array}{cccc} 1 & 0 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{array} \to \{0, 2, 3\}$$

Let $\varphi(x_1, \ldots, x_k)$ be a Presburger Arithmetic formula. Then we can construct a WS1S formula $\varphi'(X_1, \ldots, X_k)$ such that $\varphi(n_1, \ldots, n_k)$ is true iff $\varphi'(g(n_1), \ldots, g(n_k))$ is true. The idea is that any addition of two natural numbers x_1 and x_2 in φ can be translated into a second-order formula $\exists X_{carry} X_{result}\ \psi(X_1, X_2, X_{carry}, X_{result})$ such that $X_1, X_2, X_{carry}, X_{result}$ respectively denote the set representation of $x_1, x_2, x_1 + x_2$, and the positions of carry bits during addition.

References

[1] Leonid Libkin. *Elements of Finite Model Theory.* Springer-Verlag, 2004.