Automata and Logic (CSC2428)
Lecture 11

Prof. Leonid Libkin

March 31, 2005

1 String Automatic Structures

In this lecture we will study definability issues for automatic structures.

Let Σ be a finite alphabet and Ω a set of constants, relations and functional symbols. We define the following strucutre $\mathcal{M} = (\Sigma^*, \Omega)$.

Obs. 1 *If every relation, function and constant in Ω is definable by an automaton, then for every first-order formula $\varphi(\bar{x})$ over $\mathcal{M}$, the set of all the tuples $\{\bar{s} | \mathcal{M} \models \varphi(\bar{s})\}$ is definable by an automaton.*

In order to see whether the above claim is true, we first have to define what it means for an automaton to accept a tuple (see Figure 1). Let $\bar{s} = (s_1, ..., s_n)$ be a tuple such that $s_i \in \Sigma^*$ for every $i = \overline{1, n}$. We represent the tuple $\bar{s}$ as a string over the alphabet $(\Sigma \cup \{\perp\})^n$. The symbol $\perp$ is a padding symbol not belonging to Σ which is appended to some of the words s_i to make sure that all components have the same length. The automaton works synchronously on all n components of $\bar{s}$. Since the automata are closed under projection, intersection, union, complementation and cartesin product, the above obeservation is true.

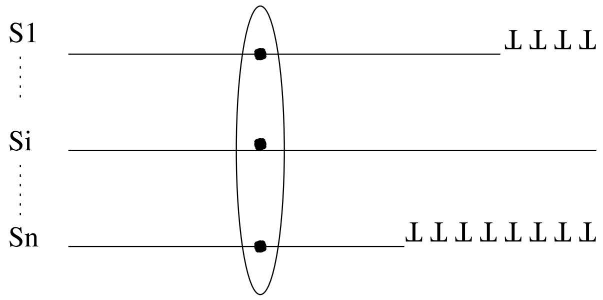

Figure 1: A run of a n-letter automaton working on tuples

The *FO-theory* of $\mathcal{M}$ is defined as the set of all first-order sentences that are true in structure $\mathcal{M}$ and is denoted as $Th_{FO}(\mathcal{M})$.

Claim 1 *Let $\mathcal{M}$ be a structure where each symbol in Ω is effectively given by an automaton. Then $Th_{FO}(\mathcal{M})$ is decidable.*

Proof. Any FO-sentence can be translated into an input-free automaton. Also, the operations on automata are effective. $\square$

This approach is used to prove decidability of theories.

Definition 1 *The structure $\mathcal{M} = (\Sigma^*, \Omega)$ where each realtion in the vocabulary Ω is effectively given by an automaton is an* automatic structure.

Definition 2 *Let $\mathcal{M}' = (\mathcal{U}, \Omega')$ be an arbitrary structure where $\mathcal{U}$ is a non-empty universe and Ω' is a set of m relation, functions and constants symbols. We say that $\mathcal{M}'$ has an* automatic presentation *if there exists a regular language L over Σ and for each n-ary relation (function, constant) R in Ω' there exists a regular language L_R over $(\Sigma \cup \{\perp\})^n$ such that*

$$\mathcal{M}' \cong (L, L_{R_1}, ..., L_{R_m})$$

Claim 2 *Let $\mathcal{M}'$ be a structure that has an automatic presentation. Then $Th_{FO}(\mathcal{M}')$ is decidable.*

Example 1 The *dense linear ordering* $(\mathbb{Q}, <)$ has an automatic presentation

$$(\mathbb{Q}, <) \cong (\{0, 1\}^*1, <_{lex})$$

and $Th_{F0}(\mathbb{Q}, <)$ is decidable (see lecture ♯9).

Example 2 The *Presburger arithmetic* $(\mathbb{N}, +)$ is an automatic structure. Every natural number is represented as its binary encoding (starting with the most significant digit), i.e. $\mathbb{N} \rightsquigarrow 1\{0, 1\}^* \cup \{0\}$, and the addition relation $+$ can be represented by an automaton (see lecture ♯3). Thus, $Th_{FO}(\mathbb{N}, +)$ is decidable.

```
    0........................................................01.........1
n   ___________________________________________________________

    0......................................................01
m   _______________________________________________________
```

Figure 2: The binary encoding of n and m such that n is a power of 2 dividing m

Example 3 The *Buchi arithmetic* $(\mathbb{N}, +, |_p)$, where p is a prime number, is the expansion of the Presburger arithmetic by the realtion

$$n|_p m : \text{iff } n \text{ is a power of } p \text{ dividing } m$$

The *Buchi arithmetic* has an automatic presentation. Let us consider the case when $p = 2$ and let n, m be two natural numbers. We represent n and m by their binary encoding starting with the least significant digit, i.e. $\mathbb{N} \rightsquigarrow \{0, 1\}^*1 \cup \{0\}$. The $|_p$ can be defined by an automaton that checks synchronously for 0s in both strings. It rejects the tuple if it reads a 1 in n and a 0 in m, but accepts otherwise (see Figure 2).
Note that we can define the *Buchi arithmetic* by considering the structure: $(\mathbb{N}, +, V_p)$, where $V_p : \mathbb{N} \to \mathbb{N}$ and

$$V_p(n) \text{ is the largest power of } p \text{ that divides } n$$

Thus, $Th_{FO}(\mathbb{N}, +, V_p)$ is decidable for every prime number p, since we can construct an automaton similar with the one above.

The question is : *Is there a vocabulary Ω such that the definable sets in the structure $\mathcal{M} = (\Sigma^*, \Omega)$ are all the regular languages ?*

Definition 3 *The structure $\mathcal{M} = (\Sigma^*, \Omega)$ is a* universal automatic structure *if for every regular relation R, where $R \subseteq (\Sigma^*)^m$, $m > 0$, there is a formula $\varphi_R(\bar{x})$ over Ω such that $R = \varphi_R(\mathcal{M}) = \{\bar{s} | \mathcal{M} \models \varphi_R(\bar{s})\}$.*

There exists a vocabulary Ω such that $\mathcal{M} = (\Sigma^*, \Omega)$ is a *universal automatic structure*. The vocabulary contains the following relations:

- $s \prec s'$ - s is a prefix of s'
- $L_a(s)$ - for each $a \in \Sigma$, $L_a(s)$ is true iff the last symbol of s is a
- $el(s, s')$ - the equal length predicate is true iff $|s| = |s'|$

Also, we will consider that the alphabet Σ has at least two symbols.

We are going to show that the realtions definable in a structure $\mathcal{M} = (\Sigma^*, \Omega)$, where Ω is the vocabulary defined above are precisely the *regular relations*.

Claim 3 *The relations $\{\prec, el, L_a\}$ are definable by an automaton.*

Proof. Let $s = (s_1, ..., s_k)$ and $s' = (s'_1, ..., s'_l)$ be two words over Σ and let $k \leq l$. We padd the word s with the symbol $\perp$ such that $|s| = |s'|$.

In order to see whether $s \prec s'$, the automaton will check that $s_i = s'_i$ and when it reads a pair of the form $(\perp, s'_j)$ it enters in a final state and accepts the input.

The automaton defining the relation el has to check that the input contains no padding symbols $\perp$. $\square$

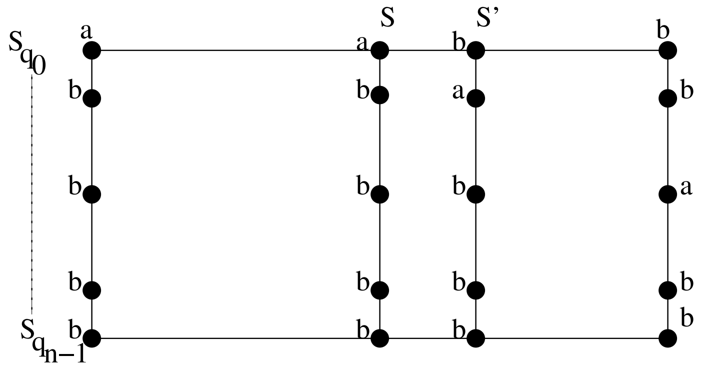

Figure 3: Illustration to the proof of Lemma 4

Lemma 4 *The structure $\mathcal{S} = (\Sigma^*, \prec, (L_a)_{a \in \Sigma}, el)$ is a universal automatic structure.*

Proof. We will prove the result on sets, since the proof for relations is similar.

We can assume that the alphabet Σ consists of two symbols a and b. Let $L \subseteq \Sigma^*$ be a regular language. Then the language L is definable by a DFA $\mathcal{A} = (Q = \{q_0, ..., q_{n-1}\}, q_0, \delta : Q \times \Sigma \to Q, F)$. The formula has to simulate the accepting run of the automaton. That means that it has to check that the first symbol of the string is in q_0, the last symbol is in q_{n-1}, the set forms a partition and the transition function is respected.

Therefore, each state q_i in the set $Q = \{q_0, ..., q_{n-1}\}$ is encoded as a string s_{q_i} over $\{a, b\}^*$ such that the position j is a iff the automaton is in state q_i and is b iff the automaton is not in that state (see Figure 3).

The formula $\varphi_L(\bar{x})$ says that $\exists x_0 ... \exists x_{n-1} \wedge_{i,j} (el(x_i, x_j) \wedge el(x_i, x))$ and they partition the set of positions and they encode an accepting run, i.e. if s is a prefix of s' such that there is no other prefix between them and the possible configuration presented in Figure 3 has to be a valid transition.

$\square$

Example 4 **(Definability over $\mathcal{S}$)** The following relations can be defined over $\mathcal{S}$:

- $|s| = |s'| + 1$ - *it is defined by* $\exists w (w \prec s' \wedge \neg \exists l (w \prec l \wedge l \prec s') \wedge el(w, s))$

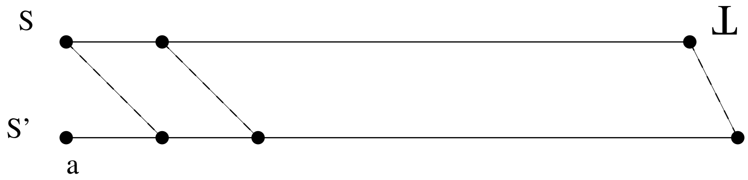

Figure 4: A run of the automaton checking that $s' = as$

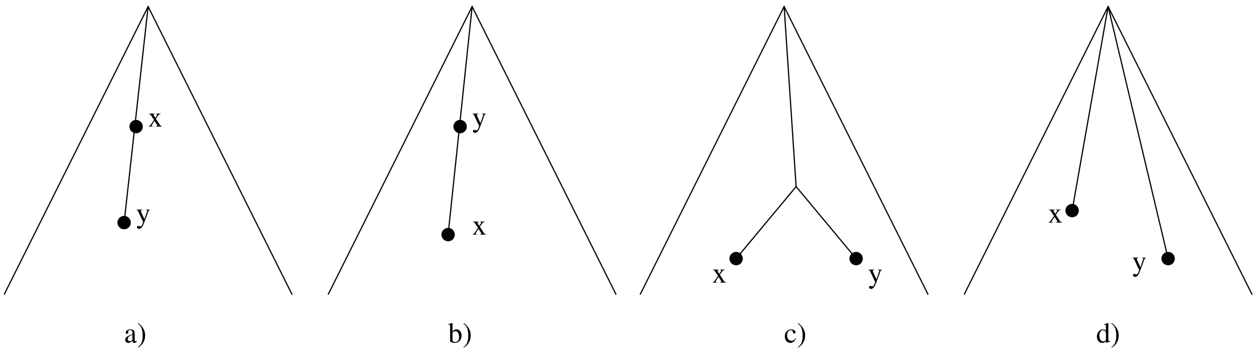

Figure 5: Positions of x and y in a tree

- $s \sqcap s'$ *- the largest common prefix of s and* s'
- $s <_{lex} s'$ *- the lexicographic ordering*
- $l_a : s \to sa$ *- adds last character*
- $f_a : s \to as$ *- adds first character*

The $<_{lex}$ is defined as follows. Assume that there is given an ordering $a_1 < ... < a_n$ on the elements of Σ. Then

$$x <_{lex} y \equiv x \prec y \vee \exists z(z \prec x \wedge z \prec y \wedge \bigvee_{i<j}((l_{a_i}(z) \prec x) \wedge (l_{a_j}(z) \prec y))$$

The function f_a, where $a \in \Sigma$, can be defined by an automaton. Given two words s and s', the automaton will read pairs of two symbols (s_i, s'_i) over the alphabet $\Sigma \cup \{\perp\}$. First the automaton checks that s' starts with an a and stores the last symbol in s. Then it checks that $s_i = s'_{i+1}$, it again stores the symbol s_{i+1} and it will proceed like this until the automaton reads the symbol $\perp$. If it reads more than one $\perp$ it will reject the input (see Figure 4).

Now, let us consider the following structure $\mathcal{S}_p = (\Sigma^*, \prec, (L_a)_{a\in\Sigma})$.

Example 5 (Definability over $\mathcal{S}_p$) The following relations can be defined over $\mathcal{S}_p$ (the equal-length predicate is not needed to define them)

- $s \sqcap s'$
- $s <_{lex} s'$
- $l_a : s \to sa$

The function f_a is not included in the above list since the formula that defines it uses the equal-length predicate.

Informally, the $\mathcal{S}_p$ is a restriction of **S2S** to **FO**-definability. Let us consider that $\Sigma = \{0, 1\}$. Then $\prec$ becomes the descendent-relation and for every point it is possbile to check its left or right succesor by using L_0 and L_1.

Theorem 5 *The subsets of Σ^* definable in the structure $\mathcal{S}_p = (\Sigma^*, \prec, (L_a)_{a\in\Sigma})$ are precisely the $*$-free languages.*

First, let us define the following predicate. For each $L \subseteq \Sigma*$, let $P_L(s, s')$ be true iff $s \prec s'$ and $s' - s \in L$ (i.e. $s' - s$ is the string v such that $sv = s'$).

In order to prove the theorem we need to have the following lemma.

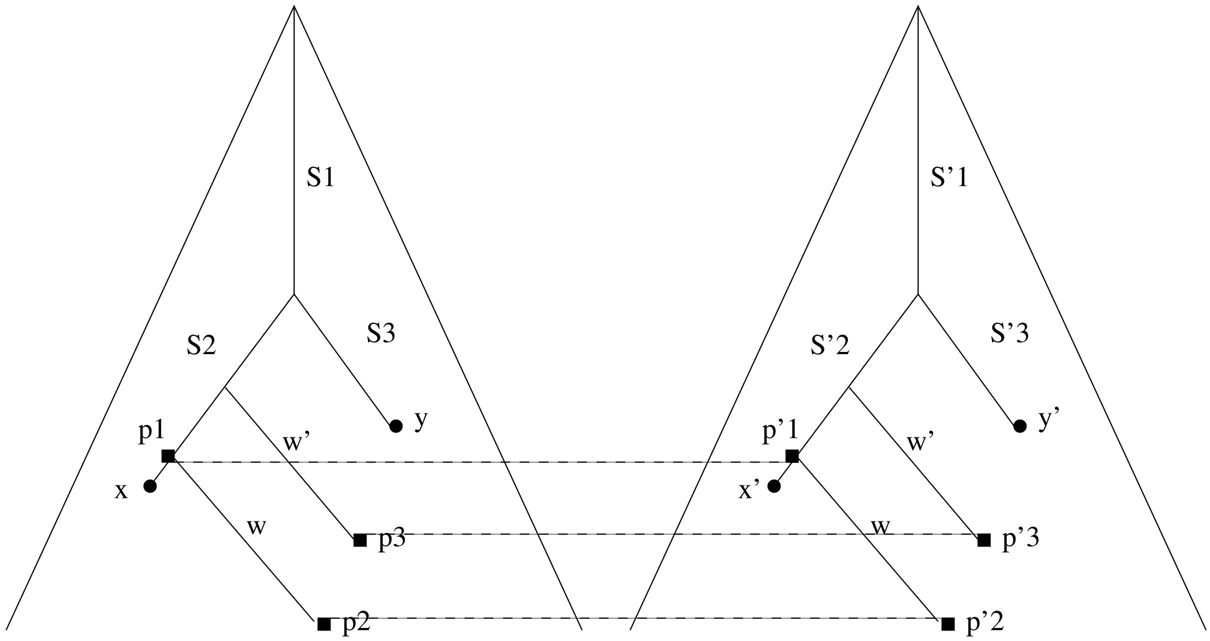

Figure 6: Illustration to the proof of Claim 7

Lemma 6 *$\mathcal{S}_p$ has quantifier elimination in the vocabulary that involves the following*

$$(\prec, (L_a)_{a\in\Sigma}, \epsilon, \sqcap, (P_L)_{L \text{ star-free}})$$

, where ϵ is the constant representing the empty string.

Proof. We use Ehrenfeucht-Fraisse games. Note that there is also a constructive proof of this lemma.

Let us consider binary predicates, since the same arguments will work for k-ary predicates. Let $\varphi(x, y)$ be a formula of quantifier-rank k. Since Σ^+ can be viewed as a tree, there are four possible way in which x and y can be positioned in the tree (see Figure 5).

For example, the arragement in Figure 5 c) can be defined by the formula: $\neg(x \prec y) \wedge \neg(y \prec x) \wedge \neg(x \sqcap y = \epsilon)$. For each case, we can express $\varphi(x, y)$ without using quantifiers. Then, we take the disjunction of all such quantifier-free formula.

Let us consider the case in Figure 6. The distinguished points x, y are at the same level in the tree as the distinguished points x', y'. Since each string is a structure, we can use in our proof the following claim.

Claim 7 *If $s_1 \equiv_{k+2} s'_1$ and $s_2 \equiv_{k+2} s'_2$ and $s_3 \equiv_{k+2} s'_3$ then $(\mathcal{S}_p, x, y) \equiv_k (\mathcal{S}_p, x', y')$.*

In other words, the duplicator wins the game in k rounds by playing in each of the three structure s_1, s_2, s_3 for $k+2$ rounds.

Why it is true the above claim? We have to show that the duplicator wins the game. The proof is shown in Figure 6. Suppose the spoiler plays the position p_1 in the first tree somewhere on the string s_2, then the duplicator can respond by choosing p'_1 such that the isomorphism between the two structures can be kept. Next, let us assume the spoiler chooses to play position p_2. The duplicator responds by playing p'_2 such that in both trees the isomorphism defined by the game preserves the string w.

Now we can show the lemma. From the claim, the formula $\varphi(x, y)$ is a union of equivalence classes with respect to the relations $\equiv_{k+2}$ and these equivalences are $*$-free definable. Thus, there is a set of triples $(L_{i_1}, L_{i_2}, L_{i_3})$ of $*$-free languages such that $\varphi(x, y)$ is true in the above tree configuration iff for some i, $s_1 \in L_{i_1}$ and $s_2 \in L_{i_2}$ and $s_3 \in L_{i_3}$. The fact that $s_1 \in L_{i_1}$ can be expressed as $P_{L_{i_1}}(\epsilon, x \sqcap y)$, $s_2 \in L_{i_2}$ is expressed as $P_{L_{i_2}}(x \sqcap y, x)$ and $s_3 \in L_{i_3}$ as $P_{L_{i_3}}(x \sqcap y, y)$. □

Remark 1 *The equal-length relation cannot be expressed in $\mathcal{S}_p$ since there are languages not $*$-free.*

Now, let us consider the following automatic structure $\mathcal{S}_p^{reg} = (\Sigma^*, \prec, (L_a)_{a\in\Sigma}, \epsilon, \sqcap, (P_L)_{L \text{ regular}})$.

Proposition 8 *$\mathcal{S}_p^{reg}$ has quantifier elimination.*

Theorem 9 *$MSO(\mathcal{S}_p^{reg})$=$S2S$ with only FO free variables, and $MSO(\mathcal{S}_p^{reg})$=$FO(\mathcal{S}_p^{reg})$.*

Corollary 10 *$S2S$ collapsses to $wS2S$ for formulas with only FO free varibales, while in general, $S2S$≠$wS2S$.*

Proof. Each relation from $\mathcal{S}_p^{reg}$ can be expressed in $wS2S$ and the structure has quantifier elimination. □

Example 6 (Structures that are decidable but without automatic presentation)

- *$\mathcal{M}' = (U, \Omega')$ such that it defines a pairing function, i.e. $\pi : U \times U \to U$ is an injection*
- *Skolem arithmetic* $(\mathbb{N}, \cdot)$
- *Random Graphs*

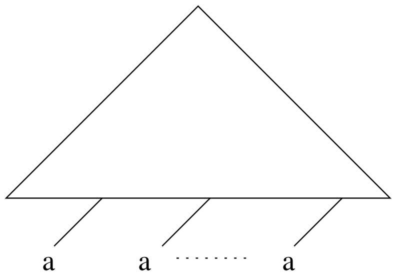

Figure 7: Illustration of the relation $succ_0^a(T)$

2 Tree Automatic Structures

A tree is represented as the structure $T = (D, \lambda)$, where the domain D is a prefix-closed subset $\{0,1\}^*$ and $\lambda : D \to \Sigma$ is the labeling function.

Let $\mathcal{T} = (Tree(\Sigma), \prec, succ_0^{a\in\Sigma}, succ_1^{a\in\Sigma}, \approx_{dom})$ be a structure over binary trees such that

- Σ - the alphabet has at least two symbols
- $Tree(\Sigma)$ - all the binary Σ-labeled trees (we allow unary branching)
- $T_1 \prec T_2$ - if $T_1 = (D_1, \lambda_1)$, $T_2 = (D_2, \lambda_2)$ then $T_1 \prec T_2$ is true iff $D_1 \subset D_2$ and $\lambda_2 = \lambda_1$ over the domain D_1
- $T_1 \approx_{dom} T_2$ - if $T_1 = (D_1, \lambda_1), T_2 = (D_2, \lambda_2)$ then $T_1 \approx_{dom} T_2$ is true iff $D_1 = D_2$
- $succ_0^a(T)$ - for each leaf of the tree T a left child is added and labeled a (see Figure 7)
- $succ_1^a(T)$ - for each leaf of the tree T a right child is added and labeled a

Theorem 11 *$\mathcal{T}$ is a universal tree-automatic structure.*

We consider the structure $\mathcal{T}_p = (Tree(\Sigma), \prec, succ_0^{a\in\Sigma}, succ_1^{a\in\Sigma})$.

Theorem 12

- *$\approx_{dom}$ is not definable in $\mathcal{T}_p$.*
- *Subsets of $Tree(\Sigma)$ definable in $\mathcal{T}_p$ are precisely the regular tree-languages.*

Claim 13 *$\mathcal{T}$ does not have an autoamtic presentation.*
Proof (Sketch). $\mathcal{T}$ defines a pairing function.