Topics in DB: Foundations of XML (CSC2538)
Lecture 1

Prof. Leonid Libkin
notes by Zheng Zhang

Sept. 23rd, 2005

1 Introduction

In this lecture, we will learn automata on finite strings and trees (both ranked and unranked) and their relationships with First Order Logic (FO) and Monadic Second Order Logic (MSO).

1.1 Course prerequisites

Basic knowledge of databases, propositional logic, complexity, and theory of computation.

2 Automata on finite strings

A *non-deterministic finite state automaton* (NFA) $\mathcal{A}$ is a tuple

$$\mathcal{A} = (\Sigma, Q, Q_0, F, \delta), \text{ where}$$

1. Σ is an alphabet,
   Note: Σ^* is a set of (finite) strings over Σ.
2. Q is a set of states,
3. Q_0 is a set of initial states and $Q_0 \subseteq Q$,
4. F is a set of final states and $F \subseteq Q$,
5. $\delta : Q \times \Sigma \to 2^Q$ is a transition function.

A string is a set of positions and labels over some alphabet. Formally, let a set of positions be $\{0, \ldots, n-1\}$, denoted as $[n-1]$ and a *labeling function* from positions to members of the alphabet Σ be $\lambda : \{0, \ldots, n-1\} \to \Sigma$. For example, if S is the string *abaaba*, then the labeling function labels $\lambda_s(0) = a$, $\lambda_s(1) = b$, etc. Hence a string S of length n is a 2-tuple $([n-1], \lambda_s)$.

A *run* $\rho_{\mathcal{A},S} : \{0, \ldots, n-1\} \to Q$ of automaton $\mathcal{A}$ on a string S is:

$$\rho_{\mathcal{A},S}(0) \in \delta(q_0, \lambda_s(0)), q_0 \in Q_0,$$
$$\rho_{\mathcal{A},S}(i+1) \in \delta(\rho_{\mathcal{A},S}(i), \lambda_s(i+1)).$$

A run $\rho_{\mathcal{A},S}$ is *accepting* iff $\rho_{\mathcal{A},S}(n-1) \in F$.

An automaton $\mathcal{A}$ *accepts* a string S iff there exists an accepting run of $\mathcal{A}$ on S .

The *language* accepted by automaton $\mathcal{A}$ is $L(\mathcal{A}) = \{S \in \Sigma^* \mid \mathcal{A} \text{ accepts } S\}$. Such a language is called a regular language.

An automaton $\mathcal{A}$ is *deterministic* iff

$$\forall q \in Q, \forall a \in \Sigma, |\delta(q, a)| \leq 1,$$

i.e., for each string $S \in \Sigma^*$, there is exactly one run $\rho_{\mathcal{A},S}$. Note that a transition function of a DFA is $\delta : Q \times \Sigma \to Q$.

Theorem 2.1 *: For any non-deterministic automaton $\mathcal{A}$, there is an equivalent deterministic automaton $\mathcal{A}'$ s.t. $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof (Power-set construction):
Suppose $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$. We construct an equivalent DFA $\mathcal{A}'$ as follows:

$$\mathcal{A}' = (\Sigma, 2^Q, \{X \mid X \subseteq Q_0\}, \{Y \mid Y \cap F \neq \emptyset\}, \delta'), \text{ where } \quad \delta' : 2^Q \times \Sigma \to 2^Q \ , \ \delta'(X, a) = \bigcup_{q \in X} \delta(q, a).$$

3 Regular Languages

A regular expression e has the following syntax.

$e = \emptyset \mid a \mid e \cup e' \mid e; e' \mid e^*$ where $a \in \Sigma, e'$ is a regular expression, operator ";" represents concatenation.

Notice sometimes a regular expression can include operator complement: $\bar{e}$. Moreover, $e^+ = e; e^*$ and $e? = e \mid \emptyset$. Denote $L(e)$ the language given by e.

Theorem 3.1 *:*

- *For any regular expression e, there exists a finite automaton $\mathcal{A}_e$ s.t. $L(e) = L(\mathcal{A}_e)$.*
- *For any finite automaton $\mathcal{A}$, there exists a regular expression $e_{\mathcal{A}}$ s.t. $L(\mathcal{A}) = L(e_{\mathcal{A}})$.*

Some complexity results:

Non-emptiness	$L(\mathcal{A}) \neq \emptyset$	NLOGSPACE-complete
Equivalence	$L(\mathcal{A}) = L(\mathcal{A}')$	PSPACE-complete
Containment	$L(\mathcal{A}) \subseteq L(\mathcal{A}')$	PSPACE-complete
Universality	$L(\mathcal{A}) = \Sigma^*$	PSPACE-complete

Notice: DLOGSPACE $\subseteq$ NLOGSPACE $\subseteq$ PTIME $\subseteq$ NP $\subseteq$ PSPACE.
The *closure property* of regular languages: the union or intersection of two regular languages is still regular; the complement of a regular language is regular.

4 Logic background

4.1 First-order predicate calculus (FO)

Note: Syntax and Semantics were omitted during the lecture.

4.1.1 Syntax

A *vocabulary* σ has:

1. function symbols: $f_1, f_2, \ldots$ of non-negative arities $m_1, m_2, \ldots$,
2. constant symbols (zero-ary function symbols): $c_1, c_2, \ldots$,
3. predicate symbols: $P_1, P_2, \ldots$ of positive arities $n_1, n_2, \ldots$.

FO *terms* and FO *formulae*:
(assume an infinite supply of variables)

1. Each variable x is a term.
   Free variables (FV) : x.

2. If $t_1, \ldots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \ldots, t_k)$ is a term.
   FV : $\bigcup_i \mathrm{FV}(t_i)$.

3. Each constant symbol c_i is a term.
   FV : none.

4. If $t_1, \ldots, t_k$ are terms and P is a k-ary predicate symbol, then $P(t_1, \ldots, t_k)$ is an *atomic* formula.
   FV : $\bigcup_i \mathrm{FV}(t_i)$.

5. If t and t' are terms, then $t = t'$ is a formula.
   FV : $\mathrm{FV}(t) \cup \mathrm{FV}(t')$.

6. If ϕ and ψ are formulae and x is a variable, then $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\exists x\phi$, and $\forall x\phi$ are formulae.
   FV : $\mathrm{FV}(\phi) \cup \mathrm{FV}(\psi)$, $\mathrm{FV}(\phi) \cup \mathrm{FV}(\psi)$, $\mathrm{FV}(\phi)$, $\mathrm{FV}(\phi)$ - x, and $\mathrm{FV}(\phi)$ - x, respectively.

4.1.2 Semantics (Tarski semantics)

An *interpretation* (or a *structure*) $\mathcal{M}$ is:

$$\mathcal{M} = (U, f_1^{\mathcal{M}}, f_2^{\mathcal{M}}, \ldots, c_1^{\mathcal{M}}, c_2^{\mathcal{M}}, \ldots, P_1^{\mathcal{M}}, P_2^{\mathcal{M}}, \ldots)$$

where

1. U is the *universe*: a non-empty set,
2. $f_i^{\mathcal{M}} : U^{n_i} \to U$, where n_i is the arity of the function symbol f_i,
3. $c_i^{\mathcal{M}} \in U$,
4. $P_i^{\mathcal{M}}$ of arity n_i is a subset of U^{n_i}.

Each term $t(\overline{x})$, where $|\overline{x}| = m$, is assigned an element $t^{\mathcal{M}}(\overline{a}) \in U$, where $\overline{a} \in U^m$ is defined by a corresponding object assignment.

For a formula ψ, $\mathcal{M} \models \psi$, meaning $\mathcal{M}$ *satisfies* ψ (under some object assignment) is defined by:

1. $\mathcal{M} \models t_1(\overline{a}) = t_2(\overline{a})$ iff $t_1^{\mathcal{M}}(\overline{a}) = t_2^{\mathcal{M}}(\overline{a})$,
2. $\mathcal{M} \models P(t_1(\overline{a}), \ldots, t_k(\overline{a}))$ iff $(t_1^{\mathcal{M}}(\overline{a}), \ldots, t_k^{\mathcal{M}}(\overline{a})) \in P^{\mathcal{M}}$,
3. $\mathcal{M} \models \neg\psi$ iff $\mathcal{M} \not\models \psi$,
4. $\mathcal{M} \models \psi \wedge \phi$ iff $\mathcal{M} \models \psi$ and $\mathcal{M} \models \phi$,
5. $\mathcal{M} \models \psi \vee \phi$ iff $\mathcal{M} \models \psi$ or $\mathcal{M} \models \phi$,
6. $\mathcal{M} \models \exists x\, \psi(\overline{a}, x)$ iff $\mathcal{M} \models \psi(\overline{a}, b)$ for some $b \in U$,
7. $\mathcal{M} \models \forall x\, \psi(\overline{a}, x)$ iff $\mathcal{M} \models \psi(\overline{a}, b)$ for every $b \in U$.

4.2 Representing strings as structures

To represent a string by a structure, we define a vocabulary:

1. a binary predicate symbol $<$, and
2. predicate symbols $P_a, P_b, \ldots$ for every $a, b, \ldots \in \Sigma$.

A string $s \in \Sigma^*$, $|s| = n$, is then represented by the following structure $\mathcal{M}_s$:

$$\mathcal{M}_s = (U, <^{\mathcal{M}_s}, P_a^{\mathcal{M}_s}, P_b^{\mathcal{M}_s}, \ldots), \text{ where } U = \{0, \ldots, n-1\},$$
$$<^{\mathcal{M}_s} \text{ is a binary } < \text{ on numbers, and}$$
$$P_a^{\mathcal{M}_s} = \{i | a \text{ occurs at position } i \text{ in } s, i \leq n-1.\}$$

For example, consider a string *ababaab*. The structure $\mathcal{M}$ corresponding to this string is:

$$(\{0,1,2,3,4,5,6\}, <, P_a, P_b), \text{ where } P_a = \{0,2,4,5\} \text{ and}$$
$$P_b = \{1,3,6\}.$$

Consider another example: a string $\Sigma^* a \Sigma^* b \Sigma^*$ where "a occurs before b" can be represented by the following FO formula:

$$\exists x, \exists y, x < y \wedge P_a(x) \wedge P_b(y).$$

4.3 Monadic second-order logic (MSO)

Syntax and Semantics were omitted in the lecture. Monadic second order logic adds quantification over sets to first-order calculus.

4.3.1 Syntax

We add *second-order variables*, *i.e.*, variables that range over subsets of the universe, to the vocabulary of FO. We denote second-order variables by capital letters.

MSO terms and MSO formulae:

1. all FO terms and all FO formulae,
2. if X is a second-order variable and t is a term, then $X(t)$ is a formula,
3. if ψ is a formula, then $\exists X \psi$ and $\forall X \psi$ are also formulae.

4.3.2 Semantics

We have an interpretation (or a structure) $\mathcal{M}$, as before.

Each term $t(\overline{x})$, where $|\overline{x}| = m$ is assigned an element $t^{\mathcal{M}}(\overline{a}) \in U$, where $\overline{a} \in U^m$ is defined by a corresponding object assignment.

Each formula $\psi(\overline{x}, \overline{X})$, where $|\overline{X}| = m$, is assigned an element $\psi(\overline{a}, \overline{S})$, where $\overline{S} = (S_1, \ldots, S_m)$, $S_i \subseteq U$ is defined by a corresponding object assignment.

For a formula ψ, $\mathcal{M} \models \psi$, meaning $\mathcal{M}$ *satisfies* ψ (under some object assignment) is defined by:

1. all rules from FO,
2. for $\psi(\overline{x}, X) = X(t(\overline{x}))$, $\mathcal{M} \models \psi(\overline{a}, S)$ iff $t^{\mathcal{M}}(\overline{a}) \in S$,
3. $\mathcal{M} \models \exists X\ \psi(X, \ldots)$ iff there exists some $X \subseteq U$ such that $\mathcal{M} \models \psi(S, \ldots)$,
4. $\mathcal{M} \models \forall X\ \psi(X, \ldots)$ iff for every $X \subseteq U$, $\mathcal{M} \models \psi(S, \ldots)$.

Consider the following example:
Given a regular expression a^*b^*, find a sentence $\psi \in MSO$, such that $s \in L(a^*b^*)$ iff $\mathcal{M}_s \models \psi$.

$$\psi = \exists X, \exists Y, ((\forall x \in X, \forall y \in Y, x < y) \bigwedge (\forall x, X(x) \vee Y(x)) \bigwedge (\neg \exists x, X(x) \wedge Y(x)) \bigwedge$$
$$(\forall x \in X, P_a(x) \wedge \forall y \in Y, P_b(y)))$$

5 Büchi 1959 results

A language $L \subseteq \Sigma^*$ is *definable* in a logic $\mathcal{L}$ (FO or MSO) if there is a sentence ψ of $\mathcal{L}$, such that

$$L = \{s \in \Sigma^* \mid \mathcal{M}_s \models \psi\} = L(\psi),\ \textit{i.e.}, s \in L \Leftrightarrow \mathcal{M}_s \models \psi.$$

For example, if ψ is the formula from the above example, then $L(\psi) = L(a^*b^*)$.

Theorem 5.1 (Büchi 1960) *A language is regular iff it is definable in MSO.*

Theorem 5.2 (McNaughton/Pappert 1970) *A language is definable in FO iff it is *-free.*

Notice a regular expression e is *-free if it is generated by $\emptyset \mid a \mid e \cup e' \mid e; e' \mid \bar{e}$. The Kleene star can often be simulated by other operators. For example, $\Sigma^* = \bar{\emptyset}$, hence $\Sigma^* a \Sigma^* b \Sigma^* = \bar{\emptyset} a \bar{\emptyset} b \bar{\emptyset}$, whose complement means no a is proceeding b.

Proof of Büchi Theorem ($\Rightarrow$):

Here we prove that a regular language is definable in MSO: given a regular language $L \subseteq \Sigma^*$, we need to construct a sentence $\psi \in MSO$, such that $L = L(\psi)$.

Recall that a language is regular iff it is accepted by some deterministic finite automaton.
Let $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ be a deterministic finite automaton, such that $L(\mathcal{A}) = L$.
We now produce $\psi_{\mathcal{A}}$, such that $\mathcal{M}_s \models \psi_{\mathcal{A}}$ iff a (unique) run of $\mathcal{A}$ on string s ends in an accepting state, *i.e.*, $\mathcal{A}$ accepts s. Assume that $Q = \{q_0, \ldots, q_{k-1}\}$. Then

$$\psi_{\mathcal{A}} = = \exists X_0, \ldots, X_{k-1}$$

$$\forall x \left(\bigvee_{i=0}^{k-1} X_i(x) \wedge \neg \bigvee_{i \neq j} \exists (X_i(x) \wedge X_j(x)) \right)$$

(X_i's partition the universe)

$$\wedge \ \forall x \left((\forall y \ x \leq y) \longrightarrow \bigwedge_{a \in \Sigma} (P_a(x) \longrightarrow X_i(x)) \right) \text{ , where } q_i = \delta(q_0, a)$$

(position 0 is in X_i, where $q_i = \delta(q_0, a)$, where a is the first symbol)

$$\wedge \ \forall x \forall y \left((x < y \wedge \neg \exists z (x < z \wedge z < y)) \longrightarrow \bigwedge_{\substack{a \in \Sigma \\ i \leq k}} (X_i(x) \wedge P_a(y) \longrightarrow X_j(y)) \right)$$

, where $q_j = \sigma(q_i, a)$

(if position m is in X_i, the symbol in position $m+1$ is a, then position $m+1$ is in X_j, where $q_j = \delta(q_i, a)$)

$$\wedge \ \forall x \left(\forall y (y \leq x) \longrightarrow \bigvee_{q_i \in F} X_i(x) \right)$$

(if m is the last position, then $m \in X_i$, where $q_i \in F$)

Proof of $\Leftarrow$: a language definable in MSO is regular. Notice this part was omitted in the lecture.

Corollary 5.3 *MSO* $= \exists$ *MSO.*

The proof is directly from the Büchi Theorem since MSO $\Rightarrow$ regular language $\Rightarrow \exists$ MSO.

So far we have used $\bigvee$ for union, $\bigwedge$ for intersection, $\neg$ for complement and $\exists$ for non-deterministic guess, which leads to powerset construction. Each alternation of $\forall$ and $\exists$ causes exponential blowup.

Given $\varphi \in MSO$, construct an automaton $\mathcal{A}_\varphi$ s.t. $L(\mathcal{A}_\varphi) = \{s | \mathcal{M}_s \models \varphi\}$. Then the size of $\mathcal{A}_\varphi$ is greater than or equal to 2 to the Kth power of $|\varphi|$, where K is the number of quantifier alternations in φ, denoted as $Tower(|\varphi|, K)$. Notice if there is no quantifier alternations, then $K = 0$. The size is $2^{|\varphi|}$. If $K = 1$, the size is 2 to the power of $2^{|\varphi|}$. $Tower(|\varphi|, K)$ is

$$\left. 2^{2^{\cdot^{\cdot^{2^{|\varphi|}}}}} \right\} K \text{ times}.$$

A function $f : \mathcal{N} \to \mathcal{N}$ is elementary if there exists a K s.t. $\forall n, f(n) \leq Tower(n, K)$.

Theorem 5.4 (Meyer, 1979/Stock, 1980) *Suppose there is a function $f : \mathcal{N} \to \mathcal{N}$ s.t. $\forall \varphi, \exists \mathcal{A}_\varphi$, s.t. $| \mathcal{A}_\varphi | \leq f(| \varphi |)$. Then f is not elementary, i.e., there is no elementary function to bound the size of $\mathcal{A}_\varphi$.*

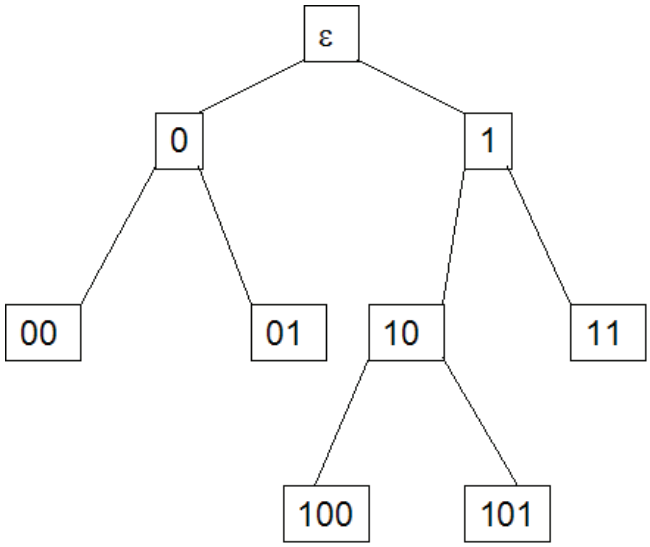

Figure 1: A document tree with a prefix-closed subset of $\{0,1\}^*$.

6 Trees

A tree can be ranked or unranked. A ranked tree is a k-ary tree whose node is either a leaf or has k children. When k=2, we call the tree a binary tree. There are no restrictions over an unranked tree. An XML document can be represented as an unranked tree.

In a binary tree, the tree domain can be represented as a prefix-closed subset of $\{0,1\}^*$, *i.e.*, Figure 1. A tree $T = (D, \lambda_T)$ where D is the tree domain and $\lambda_T : D \to \Sigma$ is a function that maps each position to a symbol in the alphabet. A string is 1-ary tree whose domain is the prefix-closed subset of 0^*, *i.e.*, $\{\varepsilon, 0, 00, \ldots\}$.

6.1 Bottom-Up Tree Automata

A non-deterministic bottom-up tree automaton can be represented as a tuple $\mathcal{A} = (Q, Q_0, F, \delta)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q$ is the set of final (accepting) states and $\delta : Q \times Q \times \Sigma \to 2^Q$ is the transition function. For example, say a node a has two leaves. Suppose the automaton is in states q_1 and q_2 at the leaves, respectively. Then a state q at a is in $\delta(q_1, q_2, a)$ (Figure 2).

A run of a tree automaton $\mathcal{A}$ over a tree T is $\rho_{\mathcal{A},T} : D \to Q$ s.t.

1. if $s, s0, s1 \in D$, then $\rho_{\mathcal{A},T}(s) \in \delta(\rho_{\mathcal{A},T}(s0), \rho_{\mathcal{A},T}(s1), \lambda_T(s))$.

2. if s is a leaf, then $\rho_{\mathcal{A},T}(s) \in \delta(q_0, q_0, \lambda_T(s))$ for some $q_0 \in Q_0$ (Figure 2).

A run is successful if $\rho_{\mathcal{A},T}(\varepsilon) \in F$. A tree T is accepted by $\mathcal{A}$ if there is a successful run $\rho_{\mathcal{A},T}$ on T. $L(\mathcal{A}) = \{T \mid T \text{ accepted by } \mathcal{A}\}$.

Automaton $\mathcal{A}$ is deterministic if $\mid \delta(q, q', a) \mid \leq 1$ for all q, q', a. Each bottom-up tree automaton is equivalent to a deterministic one (proof by powerset construction).

6.2 Top-Down Tree Automata

A non-deterministic top-down tree automaton can be represented as a tuple $\mathcal{A} = (Q, Q_0, F, \delta)$, where Q is a finite set of states, $Q_0 \subseteq Q$ is a set of initial states, $F \subseteq Q \times \Sigma$ is the set of pairs of final (accepting)

Figure 2: Demo for δ.

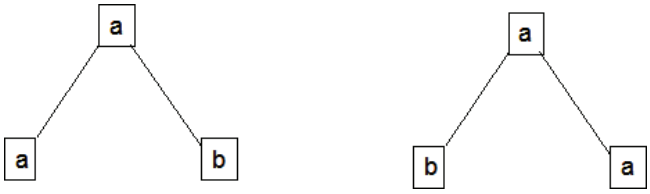

Figure 3: The two trees in L.

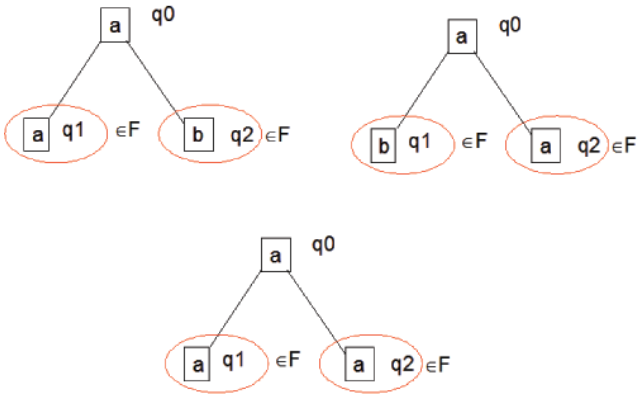

Figure 4: The two trees in L with states and the 3rd tree that should be in L.

states and symbols in Σ, and $\delta : Q \times \Sigma \to 2^{Q \times Q}$ is the transition function.

For every non-deterministic top-down tree automaton, there is an equivalent non-deterministic bottom-up tree automaton, and vice versa. However, the set of deterministic top-down tree automatons is properly contained in the set of non-deterministic top-down tree automatons.

Proof: let a language L contain only two trees as in Figure 3. Suppose there is a deterministic tree automaton $\mathcal{A}$ whose language is L. The states of the trees are shown in Figure 4. Notice the pairs of states with leaf symbols are all in F since L is the language of $\mathcal{A}$. Now that the pairs (q_1, a) and (q_2, a) are in F, the third tree in Figure 4 should also be in L as all pairs of its states with leaf symbols are in F. In other words, this tree is accepted by $\mathcal{A}$ but not in L. Contradiction! Hence there does not exist such an $\mathcal{A}$.

6.3 Logics for Trees

We represent a tree $T = (D, \lambda_T)$ as a first-order structure

$$\mathcal{M}_T = (D, <_0, <_1, (P_a)_{a \in \Sigma})$$

of vocabulary σ_Σ expanded with two binary relations $<_0$ and $<_1$. Relation $<_0$ represents left successor, *i.e.*, $s <_0 s0$ and $<_1$ represents right successor, *i.e.*, $s <_1 s1$. P_a is interpreted as $\{s \in D \mid \lambda_T(s) = a\}$,

i.e., the set of nodes labeled a.

In MSO, to express "y is a descendant of x", we use $\varphi(x, y)$ as follows.

$$\begin{aligned}\exists X, \quad & (X(x) \wedge X(y) \wedge \\ & (\forall z, z \text{ is the parent of } x \rightarrow \neg X(z)) \wedge \\ & (\forall u \in X, u \text{ is a leaf} \vee (\text{exactly one of } u0, u1 \in X, \text{ where } u <_0 u0, u <_1 u1)).\end{aligned}$$

Notice if we would like to use FO to describe similar statements on trees, we need $<_0, <_1$ and $<_*$ as descendant successor.

We call X a cut of the tree if every path from the tree root to a leaf intersects X.

Given $\varphi(X)$, represent it as a circuit with X as a set of leaves and $\vee, \wedge, \neg$ as internal nodes. Then $\varphi(X)$ is true iff the circuit evaluates to be 1.

Definition 6.1 *A tree language L (set of binary trees) is regular if it is accepted by a (bottom-up) tree automaton.*

Definition 6.2 *A tree language L is $\mathcal{L}$-definable if there is a sentence φ of $\mathcal{L}$ s.t. $\forall T \in L \Leftrightarrow T \models \varphi$.*

Theorem 6.1 (Thatcher/Wright, 1970) *A tree language is regular iff it is MSO-definable/$\exists$MSO-definable.*

There is no elementary function to bound the complexity of translation of MSO to automaton. The reason is the same as the previous one in Section 5.

Some complexity results:

Non-emptiness	$L(\mathcal{A}) \neq \emptyset$	PTIME-complete
Equivalence	$L(\mathcal{A}) = L(\mathcal{A}')$	EXPTIME-complete
Containment	$L(\mathcal{A}) \subseteq L(\mathcal{A}')$	EXPTIME-complete
Universality	$L(\mathcal{A})$ is the set of all trees	EXPTIME-complete