Automata on Finite Trees

Preliminaries

Trees

A *tree* over Σ is a partial function $t : \mathbb{N}^* \to \Sigma$ such that $dom(t)$ is a prefix-closed set:

- for each $p \in dom(t)$ for all $q \leq p$ we have $q \in dom(t)$.

A word $p \in dom(t)$ is called a *position*.

If $p, q \in dom(t)$ such that $p \cdot n = q$ for some $n \in \mathbb{N}$:

- p is the *parent* of q,
- q is the *n-th child* of p.

Trees

Given a finite tree $t \in \mathcal{T}(\Sigma)$, the *frontier* of t is the set

$$fr(t) = \{p \in dom(t) \mid \text{for all } n \in \mathbb{N} \; pn \notin dom(t)\}$$

A *path* in t is a **maximal subset** π of $dom(t)$ linearly ordered by $\leq$.

Given $p \in dom(t)$, the *subtree* t_p is defined as

$$t_p : \{q \in \mathbb{N}^* \mid pq \in dom(t)\} \to \Sigma$$

such that $t_p(q) = t(pq)$, for all $q \in dom(t_p)$.

Lemma 1 (König) *A finitely branching tree is infinite if and only if it has an infinite path.*

Coding ω-branching trees as binary trees

Let $t : \mathbb{N}^* \to \Sigma$ be a tree of arbitrary (possibly infinite) branching.

Define $t' : \{0,1\}^* \to \Sigma \cup \{\bullet\}$ as follows:

- $t'(\epsilon) = t(\epsilon)$
- for all $n_1 n_2 \ldots n_k \in dom(t)$, with $k > 0$, let

$$t'(01^{n_1}01^{n_2}\ldots 01^{n_k}) = t(n_1 n_2 \ldots n_k)$$

- for all other p let $t'(p) = \bullet$

Tree Concatenation

Let $\sigma \in \Sigma$ and $T, T' \subseteq \mathcal{T}(\Sigma)$.

By $T \cdot_\sigma T'$ we denote the set of trees obtained from some $t \in T$ by replacing each occurrence of σ on $fr(t)$ by a tree in T'.

If $\vec{\sigma} = \langle \sigma_1, \ldots, \sigma_n \rangle$, let $T \cdot_{\vec{\sigma}} \langle T_1, \ldots, T_m \rangle$ be the set of trees obtained from some $t \in T$ by replacing each occurrence of σ_i on $fr(t)$ by a tree in T_i.

We denote by $T \cdot_{\vec{\sigma}} \langle T_1, \ldots, T_m \rangle^{\omega\vec{\sigma}}$ the set of infinite trees obtained by the infinite unfolding of the concatenation operation.

Terms

A *ranked alphabet* $\langle \Sigma, \# \rangle$ is a set of symbols together with a function $\# : \Sigma \to \mathbb{N}$. For $f \in \Sigma$, the value $\#(f)$ is said to be the *arity* of f.

Zero-arity symbols are called *constants*, and denoted by $a, b, c, \ldots$.

A *term* t over Σ is a partial function $t : \mathbb{N}^* \to \Sigma$:

- $dom(t)$ is a finite prefix-closed subset of $\mathbb{N}^*$, and
- for each $p \in dom(t)$, if $\#(t(p)) = n > 0$ then $\{i \mid pi \in dom(t)\} = \{0, \ldots, n-1\}$.

Contexts

Let $X = \{x_1, \ldots, x_n\}$ be a finite set of variables, interpreted over terms.

A term $t \in \mathcal{T}(\Sigma \cup X)$ is said to be *linear* if each variable occurs in t at most once.

A *context* is a linear term $C[x_1, \ldots, x_n]$, and $C[t_1, \ldots, t_n]$ denotes the result of replacing x_i with the term t_i, for all $1 \leq i \leq n$.

A context is said to be *trivial* if it is reduced to a variable, and *non-trivial* otherwise.

Bottom Up Tree Automata

Definition

Let $\Sigma = \{f, g, h, \ldots\}$ be a finite *ranked alphabet*. A *bottom-up tree automaton* is a tuple $A = \langle S, T, F \rangle$ where:

- S is a finite set of *states*,
- T is a set of *transition rules* of the form:

$$f(q_1, \ldots, q_n) \to q$$

  where $f \in \Sigma$, $\#(f) = n$, and $q_1, \ldots q_n, q \in S$.

- $F \subseteq S$ is a set of final states.

Notice that there are no initial states.

If $\#(f) = 0$ we have rules of the form $f \to q$.

Runs

A *run* of A over a tree $t : \mathbb{N}^* \to \Sigma$ is a mapping $\pi : dom(t) \to S$ such that, for each position $p \in dom(t)$, where $q = \pi(p)$:

- if $\#(t(p)) = n$ and $q_i = \pi(pi)$, $1 \leq i \leq n$, then T has a rule

$$t(p)(q_1, \ldots, q_n) \to q$$

A run π is said to be *accepting*, if and only if $\pi(\epsilon) \in F$.

The *language* of A, denoted as $\mathcal{L}(A)$ is the set of all trees over which A has an accepting run.

A set of trees $L \subseteq \mathcal{T}(\Sigma)$ is said to be *recognizable* iff there exists a bottom-up tree automaton A such that $\mathcal{L}(A) = L$.

Examples

1. Let $\Sigma = \{f, g, a\}$, where $\#(f) = 2$, $\#(g) = 1$ and $\#(a) = 0$.

Let $A = \langle S, T, F \rangle$, where:

- $S = \{q_f, q_g, q_a\}$,
- $F = \{q_f\}$,
- $T = \{a \to q_a, g(q_a) \to q_g, g(q_g) \to q_g, f(q_g, q_g) \to q_f\}$

2. Let $\Sigma = \{red, black, nil\}$ with $\#(red) = \#(black) = 2$ and $\#(nil) = 0$.

Let $A_{rb} = \langle \{q_b, q_r\}, T, \{q_b\} \rangle$ with

$$T = \{nil \to q_b, black(q_{b/r}, q_{b/r}) \to q_b, red(q_b, q_b) \to q_r\}$$

Determinism

A tree automaton is said to be *deterministic* iff there are no two transition rules with the same left-hand side.

Proposition 1 *A deterministic tree automaton has at most one run for each input tree.*

A tree automaton is said to be *complete* iff there exists at least one transition rule $f(q_1, \ldots, q_n) \to q$, for each $f \in \Sigma$, $\#(f) = n$ and $q_1, \ldots, q_n \in S$.

Proposition 2 *A complete tree automaton has at least one run for each input tree.*

Determinism

Theorem 1 *Let L be a recognizable tree language. Then there exists a complete deterministic tree automaton A such that $\mathcal{L}(A) = L$.*

We define $A_d = \langle S_d, T_d, F_d \rangle$ where $S_d = 2^S$, $F_d = \{s \subseteq S \mid s \cap F \neq \emptyset\}$ and:

$$\begin{aligned} f(s_1, \ldots, s_n) \to s \quad &\Longleftrightarrow \quad s = \{q \in S \mid \exists q_1 \in s_1, \ldots \exists q_n \in s_n \;.\; f(q_1, \ldots, q_n) \to q\} \\ a \to s \quad &\Longleftrightarrow \quad s = \{q \in S \mid a \to q\} \end{aligned}$$

To prove $\mathcal{L}(A_d) = \mathcal{L}(A)$, we prove:

$$t \xrightarrow[A_d]{*} s \iff s = \{q \in S \mid t \xrightarrow[A]{*} q\}$$

Determinism

By induction on the structure of t.

If $t = a$, by definition we have $a \rightarrow s \iff s = \{q \in S \mid a \rightarrow q\}$

If $t = f(t_1, \dots, t_n)$, by ind. hyp. $t_i \xrightarrow[A_d]{*} s_i \iff s_i = \{q \in S \mid t_i \xrightarrow[A]{*} q\}$

"$\Rightarrow$" if $t \xrightarrow[A_d]{*} f(s_1, \dots, s_n) \xrightarrow[A_d]{} s$ we show :

$$\exists q_i \in s_i \; . \; f(q_1, \dots, q_n) \xrightarrow[A]{} q \iff t \xrightarrow[A]{*} q$$

Determinism

"$\Leftarrow$" Let $s_i = \{q \mid t_i \xrightarrow[A]{} q\}$, $i = 1, \ldots, n$ and

$$s' = \{q \mid \exists q_i \in s_i \;.\; f(q_1, \ldots, q_n) \xrightarrow[A]{} q\}$$

We conclude by showing $s = s'$ $\square$

Closure Properties

Theorem 2 *The class of recognizable tree languages is closed under union, complementation and intersection.*

Union Let $A_i = \langle S_i, T_i, F_i \rangle$ for $i = 1, 2$. Suppose that $S_1 \cap S_2 = \emptyset$. Let $A_\cup = \langle S_1 \cup S_2, T_1 \cup T_2, F_1 \cup F_2 \rangle$.

Complementation Let $A = \langle S, T, F \rangle$ be a complete deterministic tree automaton such that $\mathcal{L}(A) = L$. Define $\bar{A} = \langle S, T, S \setminus F \rangle$.

Intersection We use the fact that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Projection

Let $\Sigma = \Sigma_1 \times \Sigma_2 = \{(\sigma_1, \sigma_2) \mid \sigma_1 \in \Sigma_1,\ \sigma_2 \in \Sigma_2,\ \#(\sigma_1) = \#(\sigma_2)\}$

We define $pr_1(t) : \mathbb{N}^* \to \Sigma_1$, where $pr_1(t)(p) = \sigma_1$ iff there exist $\sigma_2 \in \Sigma_2$ such that $t(p) = \langle \sigma_1, \sigma_2 \rangle$.

$pr_2(t)$ is defined in a similar way.

Theorem 3 *If $L \subseteq \mathcal{T}(\Sigma_1 \times \Sigma_2)$ is a recognizable tree language, then so are the projections $pr_1(L)$ and $pr_2(L)$.*

Minimization

A relation $\equiv \subseteq \mathcal{T}(\Sigma) \times \mathcal{T}(\Sigma)$ is a *congruence* on $\mathcal{T}(\Sigma)$ iff for every context $C[x_1, \ldots x_n]$:

$$\forall 1 \leq i \leq n \;.\; u_i \equiv v_i \Rightarrow C[u_1, \ldots, u_n] \equiv C[v_1, \ldots, v_n]$$

For a given tree language L, we define $\equiv_L$:

$$u \equiv_L v \text{ iff for all contexts } C[x] \text{ we have } C[u] \in L \iff C[v] \in L$$

Exercise 1 *Show that $\equiv_L$ is a congruence.*$\Box$

A Myhill-Nerode Theorem for Tree Languages

Theorem 4 (Myhill-Nerode) *A tree language is recognizable iff the congruence $\equiv_L$ is of finite index.*

"$\Rightarrow$" Let $A = \langle S, T, F \rangle$ be a *complete* TA such that $L = \mathcal{L}(A)$.

Let $u \equiv_A v$ iff $u \xrightarrow{*} q \iff v \xrightarrow{*} q$, for all $q \in S$. We have $u \equiv_A v \Rightarrow u \equiv_L v$.

"$\Leftarrow$" Define $A_{min} = \langle S_{min}, T_{min}, F_{min} \rangle$, where:

- $S_{min} = \{[u]_L \mid u \in \mathcal{T}(\Sigma)\}$
- $T_{min} = \{f([u_1]_L, \ldots, [u_n]_L) = [f(u_1, \ldots, u_n)]_L \mid u_1, \ldots, u_n, u \in \mathcal{T}(\Sigma)\}$
- $F_{min} = \{[u]_L \mid u \in L\}$

Pumping Lemma for Recognizable Tree Languages

Lemma 2 (Pumping) *Let L be a recognizable tree language. Then there exists a constant $N > 0$ such that, for every $t \in L$ with $height(t) > N$, there exists a context C, a non-trivial context D and a tree u such that $C[D[u]] \in L$, and, for all $n \geq 0$ we have $C[D^n[u]] \in L$.*

Corollary 1 *Let $A = \langle S, T, F \rangle$ be a tree automaton.*

1. *$\mathcal{L}(A) \neq \emptyset$ iff there exists $t \in \mathcal{L}(A)$ with $height(t) < \|S\|$,*
2. *$\|\mathcal{L}(A)\| = \omega$ iff there exists $t \in \mathcal{L}(A)$ with $\|S\| < height(t) < 2\|S\|$.*

Pumping Lemma for Recognizable Tree Languages

Exercise 2 *Show that* $\{f(g^n(a), g^n(a)) \mid n \geq 0\}$ *is not recognizable.* $\square$

Exercise 3 (Homework) *Let* L *be a recognizable tree language over the alphabet* $\Sigma = \{f, a, b\}$*, where* $\#(f) = 2$ *and* $\#(a) = \#(b) = 0$*. Let* $L^{ac} \supseteq L$ *be the smallest tree language which is closed by the application of the two rules below:*

- **commutativity**: *for all context* C *and subtrees* t_1, t_2:

$$C[f(t_1, t_2)] \in L^{ac} \iff C[f(t_2, t_1)] \in L^{ac}$$

- **associativity**: *for all context* C *and subtrees* t_1, t_2, t_3:

$$C[f(f(t_1, t_2), t_3)] \in L^{ac} \iff C[f(t_1, f(t_2, t_3))] \in L^{ac}$$

Show that there exists a recognizable tree language L *for which* L^{ac} *is not recognizable.* $\square$

Decidability

- **Emptiness** $\mathcal{L}(A) = \emptyset$?
- **Equality** $\mathcal{L}(A) = \mathcal{L}(B)$?
- **Infinity** $\|\mathcal{L}(A)\| < \infty$?
- **Universality** $\mathcal{L}(A) = \mathcal{T}(\Sigma)$?

Theorem 5 *The emptiness, equality, infinity and universality problems on tree automata are decidable. In particular, emptiness is decidable in time polynomial in the size (number of states) of automata.*

Top Down Tree Automata

Definition

A top-down tree automaton is a tuple $A = \langle S, I, T, F \rangle$ where:

- S is a set of *states*,
- $I \subseteq S$ is a set of *initial states*,
- T is a set of *transition rules* of the form

$$q(f) \to \langle q_1, \ldots, q_n \rangle$$

  where $\#(f) = n > 0$.

- F is a set of *final states*

Notice that, for $\#(f) = 0$ there are no rules in T.

Runs

A *run* of A over a tree $t : \mathbb{N}^* \to \Sigma$ is a mapping $\pi : dom(t) \to S$ such that, for each position $p \in dom(t)$, where $q = \pi(p)$, we have:

- if $p = \epsilon$ then $q \in I$, and
- if $\#(t(p)) = n$ and $q_i = \pi(pi)$, $1 \leq i \leq n$, then T has a rule

$$q(t(p)) \to \langle q_1, \ldots, q_n \rangle$$

A run π is said to be *accepting*, if and only if $\pi(p) \in F$, for all $p \in fr(t)$.

Top Down vs. Bottom Up

Theorem 6 *Bottom up and top down tree automata recognize the same languages.*

A top down tree automaton is said to be *deterministic* if it has one initial state and no two rules with the same left-hand side.

Proposition 3 *A deterministic top down tree automaton has at most one run for each input tree.*

Proposition 4 *There exists a recognizable tree language that is not accepted by any top down deterministic tree automaton.*

Proof: $L = \{f(g(a), h(a)), f(g(a), h(a))\}$ $\square$

Tree Automata and WSkS

MSOL on Trees: (W)SωS

Let $\Sigma = \{a, b, \ldots\}$ be a tree alphabet. The alphabet of (W)SωS is:

- the function symbols $\{s_i \mid i \in \mathbb{N}\}$; $s_i(x)$ denotes the i-th successor of x
- the set constants $\{p_a \mid a \in \Sigma\}$; p_a denotes the set of positions of a
- the first and second order variables and connectives.

Examples

Let us consider binary trees, i.e. the alphabet of WS2S.

- The formula

$$closed(X) \; : \; \forall x \, . \, X(x) \rightarrow X(s_0(x)) \wedge X(s_1(x))$$

  denotes the fact that X is a downward-closed set.

- The prefix ordering on tree positions is defined by

$$x \leq y \; : \; \forall X \, . \, closed(X) \wedge X(x) \rightarrow X(y)$$

Examples

- The formula $path(X)$ denotes the fact that X is a path in the tree:

$$\begin{array}{rcl} total(X) & : & \forall x, y \,.\, X(x) \wedge X(y) \rightarrow x \leq y \vee y \leq x \\ path(X) & : & total(X) \wedge \forall Y \,.\, total(Y) \wedge X \subseteq Y \rightarrow X = Y \end{array}$$

- A leaf is defined by the formula:

$$leaf(x) \;:\; \exists X \,.\, path(X) \wedge X(x) \wedge \forall y \,.\, X(y) \rightarrow y \leq x$$

- A tree is finite iff:

$$\forall X \,.\, path(X) \rightarrow \exists x \,.\, X(x) \wedge \forall y \,.\, X(y) \rightarrow y \leq x$$

From Automata to Formulae

Let $X_1, \ldots X_k, x_{k+1}, \ldots, x_m$, and $\Sigma = \{0, 1\}^m$.

Let $A = \langle S, I, T, F \rangle$ be a non-deterministic top-down tree automaton, where $S = \{s_1, \ldots, s_p\}$.

Coding of Σ

Let $\sigma \in \{0,1\}^m$ and $\vec{X} = \langle X_1, \ldots, X_m \rangle$.

We define the formula $\Phi_\sigma(x, \vec{X})$ as the conjunction of:

- $X_i(x)$, $1 \leq i \leq m$, if $\sigma_i = 1$,
- $\neg X_i(x)$, $1 \leq i \leq m$, if $\sigma_i = 0$.

It follows, that for any $t \in \mathcal{T}(\Sigma)$, we have $t \models \forall x \ . \ \bigvee_{\sigma \in \Sigma} \Phi_\sigma(x, \vec{X})$.

Coding of S

Let $\vec{Y} = \{Y_1, \ldots, Y_p\}$ be set variables.

Intuitivelly, the set variable Y_i, $1 \leq i \leq p$ contains all tree positions labeled by A with state s_i during the run on some tree.

$$\Phi_S(\vec{Y}) \ : \ \forall z \ . \ \bigvee_{1 \leq i \leq p} Y_i(z) \ \wedge \ \bigwedge_{1 \leq i < j \leq p} \neg \exists z \ . \ Y_i(z) \wedge Y_j(z)$$

Coding of I, T and F

Every run starts from an initial state:

$$\Phi_I(\vec{Y}) \ : \ \exists x \forall y \ . \ x \leq y \wedge \bigvee_{s_i \in I} Y_i(x)$$

If A is at position x and $t(x) \in \{0,1\}^m$, A moves on $\langle s_0(x), s_1(x) \rangle$:

$$\Phi_T(\vec{X}, \vec{Y}) \ : \ \bigwedge_{i=1}^{p} \forall x \ . \ Y_i(x) \wedge \bigvee_{\sigma \in \Sigma} \Phi_\sigma(x, \vec{X}) \rightarrow \bigvee_{s_i(\sigma) \longrightarrow \langle s_j, s_k \rangle} Y_j(s_0(x)) \wedge Y_k(s_1(x))$$

If A is at a frontier position it must be in an accepting state:

$$\Phi_F(\vec{X}, \vec{Y}) \ : \ \forall x \ . \ \mathit{leaf}(x) \rightarrow \bigvee_{s_i \in F} Y_i(x)$$

From Formulae to Automata

Let $\varphi \ : \ x_2 \in X_1$.

We define $A_\varphi = \langle \{s_0, s_1\}, s_0, T, \{s_1\} \rangle$, where:

$$
\begin{array}{lcl}
\langle 0,0 \rangle (s_0) & \to & \{\langle s_0, s_1 \rangle, \langle s_1, s_0 \rangle\} \\
\langle 1,0 \rangle (s_0) & \to & \{\langle s_0, s_1 \rangle, \langle s_1, s_0 \rangle\} \\
\langle 1,1 \rangle (s_0) & \to & \langle s_1, s_1 \rangle \\
\langle 0,0 \rangle (s_1) & \to & \langle s_1, s_1 \rangle \\
\langle 1,0 \rangle (s_1) & \to & \langle s_1, s_1 \rangle
\end{array}
$$

From Formulae to Automata

Let $\varphi \ : \ s_0(x_1) = x_2$.

We define $A_\varphi = \langle \{s_0, s_1, s_2\}, T, \{s_0\} \rangle$, where:

$$
\begin{aligned}
\langle 0, 0 \rangle &\quad \rightarrow \quad s_2 \\
\langle 0, 1 \rangle &\quad \rightarrow \quad s_1 \\
\langle 0, 0 \rangle (s_2, s_2) &\quad \rightarrow \quad s_2 \\
\langle 0, 1 \rangle (s_2, s_2) &\quad \rightarrow \quad s_1 \\
\langle 1, 0 \rangle (s_1, s_2) &\quad \rightarrow \quad s_0 \\
\langle 0, 0 \rangle (s_0, s_2) &\quad \rightarrow \quad s_0 \\
\langle 0, 0 \rangle (s_2, s_0) &\quad \rightarrow \quad s_0
\end{aligned}
$$

From Formulae to Automata

As in the case of automata on words, A_Φ can be effectively constructed, for any formula Φ of $WSkS$.

Theorem 7 *Given a ranked alphabet Σ, a tree language $L \subseteq \mathcal{T}(\Sigma)$ is definable in WSkS iff it is recognizable.*

Corollary 2 *The SAT problem for WSkS is decidable.*