Automata theory and its applications

Lecture 12 -14: Automata over ranked finite trees

Zhilin Wu

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences

January 16, 2013

Ranked and unranked alphabets

Ranked alphabet: A finite set (Σ, arity) with $\text{arity} : \Sigma \to \mathbb{N}$.

Every node labeled by σ have $\text{arity}(\sigma)$ children

Motivation: Terms, e.g. $\times(5, +(3, 4))$

A *tree domain* D: A subset of $\mathbb{N}^*$ satisfying

- D is prefix-closed, i.e. $\forall xi \in D$, it holds $x \in D$,
- $\forall xi \in D$, $xj \in D$ for every $j : 0 \leqslant j \leqslant i$.

A tree t over a ranked alphabet (Σ, arity) is a tupe (D, L) where

- D is a tree domain,
- $L : D \to \Sigma$ s.t. $\forall x \in D$, the number of children of x is $\text{arity}(L(x))$.

Let T_Σ denote the set of ranked trees over the alphabet Σ.

Unranked alphabet: A finite set Σ

Every node labeled by σ can have an arbitrary number of children

Motivation: XML documents

Outline

Bottom-up finite tree automata (BUTA)

A BUTA $\mathcal{A}$ is a tuple (Q, Σ, δ, F), where δ includes the tuples of the following form

$$(q_1, \ldots, q_k, a, q) \text{ s.t. } k = \text{arity}(a).$$

In particular, if $\text{arity}(a) = 0$, then δ includes the tuples (a, q).

Runs of BUTA:

Let $t = (D, L)$ be a tree and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a BUTA,
a run of $\mathcal{A}$ over t is a tree $r_{\mathcal{A},t} = (D, L')$, where $L' : D \to Q$ s.t.
$\forall x \in D$ with children $x_1, \ldots, x_k$, $(q(x_1), \ldots, q(x_k), a, q(x)) \in \delta$.
A run $r_{\mathcal{A},t} = (D, L')$ is *accepting* if $L'(\varepsilon) \in F$.

Deterministic BUTA (DBUTA):

for every $(q_1, \ldots, q_k, a, q), (q_1, \ldots, q_k, a, q') \in \delta$, it holds $q = q'$.

Example: Boolean expressions that evaluate to "true".

$\mathcal{A} = (\{q_0, q_1\}, \{\wedge, \vee, 0, 1\}, \delta, \{q_1\})$ where δ is defined as follows,

- $(0, q_0), (1, q_1) \in \delta$,
- $(q_0, q_0, \vee, q_0), (q_1, q_0, \vee, q_1), (q_0, q_1, \vee, q_1), (q_1, q_1, \vee, q_1) \in \delta$,
- $(q_0, q_0, \wedge, q_0), (q_1, q_0, \wedge, q_0), (q_0, q_1, \wedge, q_0), (q_1, q_1, \wedge, q_1) \in \delta$.

Top-down finite tree automata (TDTA)

A TDTA $\mathcal{A}$ is a tuple (Q, Σ, δ, I), where δ includes the tuples of the following form

$$(q, a, q_1, \ldots, q_k) \text{ s.t. } k = \text{arity}(a).$$

In particular, if $\text{arity}(a) = 0$, then δ includes the tuples (q, a).

Runs of TDTA:

Similar to those of BUTA. A run (D, L') is accepting if $L'(\varepsilon) \in I$.

Deterministic TDTA (DTDTA):

- I is a singleton,
- for every $(q, a, q_1, \ldots, q_k), (q, a, q'_1, \ldots, q'_k) \in \delta$, it holds $q_1 = q'_1, \ldots, q_k = q'_k$.

Proposition. BUTA $\equiv$ TDTA.

$$(q_1, \ldots, q_k, a, q) \Longleftrightarrow (q, a, q_1, \ldots, q_k)$$

Regular languages over ranked trees: Tree languages defined by BUTAs.

Outline

BUTA ≡ DBUTA

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be BUTA.
Construct an equivalent DBUTA $\mathcal{A}' = (Q', \Sigma, \delta', F')$.

Idea: Extension of subset construction.

- $Q' = 2^Q$,
- $(S_1, \ldots, S_k, a, S) \in \delta'$, where

$$S = \{q \mid \exists q_1, \ldots, q_k.(q_1, \ldots, q_k, a, q) \in \delta, \forall i.q_i \in S_i\},$$

- $F' = \{S \mid S \cap F \neq \varnothing\}$.

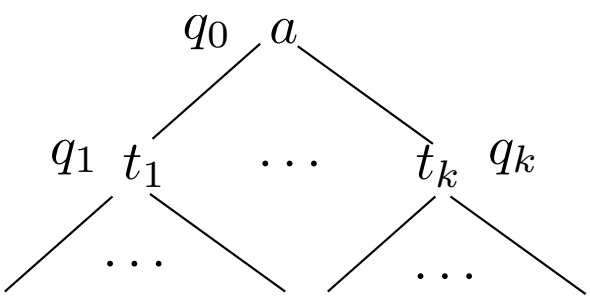
TDTA > DTDTA

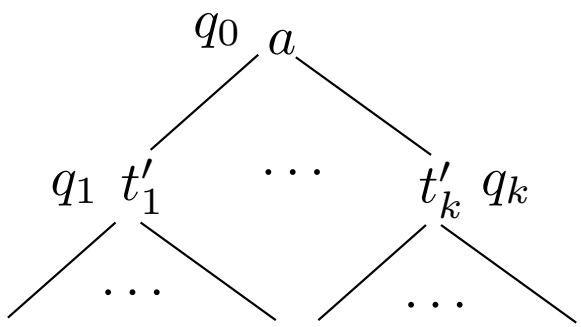
Proposition. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DTDTA.
Suppose $t = a(t_1, \ldots, t_k), t' = a(t'_1, \ldots, t'_k) \in \mathcal{L}(\mathcal{A})$, then

$$t'' = a(t''_1, \ldots, t''_k) \in \mathcal{L}(\mathcal{A}), \text{ where } \forall i,\ t''_i = t_i \text{ or } t''_i = t'_i.$$

Proof.

Let x (resp. x') be the root of t (resp. t'), $x_1, \ldots, x_k$ (resp. $x'_1, \ldots, x'_k$) be the children of x (resp. x').
$\mathcal{A}$ is deterministic and the run $r_{\mathcal{A},t}, r_{\mathcal{A},t'}$ are accepting $\Rightarrow$
$r_{\mathcal{A},t}(x) = r_{\mathcal{A},t'}(x') = q_0 \in I$, $r_{\mathcal{A},t'}(x_i) = r_{\mathcal{A},t'}(x'_i)$.
Thus, $r_{\mathcal{A},t''}$, a composition of $r_{\mathcal{A},t}|_{D(t_i)}$'s or $r_{\mathcal{A},t'}|_{D(t'_i)}$'s, is also accepting. □





TDTA > DTDTA

Proposition. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a DTDTA.
Suppose $t = a(t_1, \ldots, t_k), t' = a(t'_1, \ldots, t'_k) \in \mathcal{L}(\mathcal{A})$, then

$$t'' = a(t''_1, \ldots, t''_k) \in \mathcal{L}(\mathcal{A}), \text{ where } \forall i,\ t''_i = t_i \text{ or } t''_i = t'_i.$$

Corollary. TDTA > DTDTA.

Proof.

Let L: The Boolean expressions that evaluate to true.

Claim. *L cannot be defined by DTDTA.*

To the contrary, suppose that L is recognized by a DTDTA.
Then $\vee(0,1), \vee(1,0) \in L$ implies that $\vee(0,0) \in L$, a contradiction. □

Closure properties

Theorem. Regular tree languages are closed under union, intersection and complementation.

- Union:
  Suppose $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, F_2)$.
  Then $\mathcal{A} = (Q_1 \cup Q_2, \Sigma, \delta_1 \cup \delta_2, F_1 \cup F_2)$ defines $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.
- Intersection:
  Suppose $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, F_2)$.
  Then $\mathcal{A} = (Q_1 \times Q_2, \Sigma, \delta, F_1 \times F_2)$ defines $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$, where

  $$((q_1, q_1'), \ldots, (q_k, q_k'), a, (q, q')) \in \delta \ \textit{iff}$$
  $$(q_1, \ldots, q_k, a, q) \in \delta_1 \ \textit{and} \ (q_1', \ldots, q_k', a, q') \in \delta_2.$$

- Complementation:
  Let $L \subseteq T_\Sigma$ be defined by a DBUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$.
  Then $(Q, \Sigma, \delta, Q \backslash F)$ defines $T_\Sigma \backslash L$.

Closure properties

Proposition. Regular tree languages are closed under letter projection ($\theta : \Sigma \to \Sigma'$).

Proof.

Let $L \subseteq T_\Sigma$ be defined by a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$.
Then $\mathcal{A}' = (Q, \Sigma', \delta', F)$ s.t.

$$(q_1, \ldots, q_k, a', q) \in \delta' \ \textit{iff}\ \exists a \in \Sigma.(q_1, \ldots, q_k, a, q) \in \delta \ \textit{and}\ \theta(a) = a'.$$

Outline

Pumping lemma

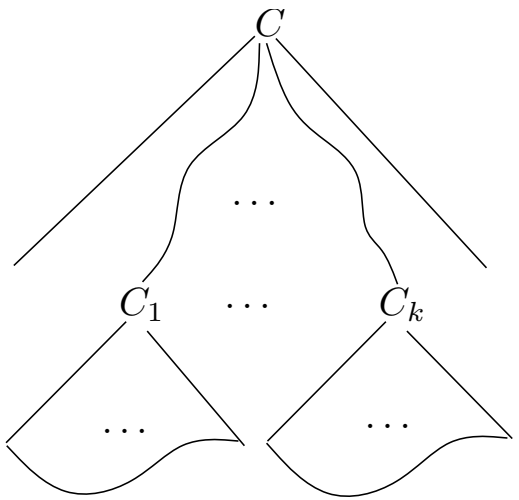
Contexts over Σ:

A ranked tree over the alphabet $\Sigma \cup \{\Box\}$ s.t. the arity of $\Box$ is zero.

Let $\mathcal{C}_{\Sigma,n}$ denote the set of all contexts over Σ with n occurrences of $\Box$.
By convention, $\mathcal{C}_{\Sigma,0} = T_\Sigma$. We also use $\mathcal{C}_\Sigma$ to denote $\mathcal{C}_{\Sigma,1}$.

Let $C \in \mathcal{C}_{\Sigma,n}$ and $C_1 \in \mathcal{C}_{k_1}, \ldots, C_n \in \mathcal{C}_{k_n}$.
Then $C[C_1, \ldots, C_n]$ is the context obtained from C by

replacing the i-th (from left to right) occurrence of $\Box$ with C_i for every $i : 1 \leqslant i \leqslant n$.



Pumping lemma

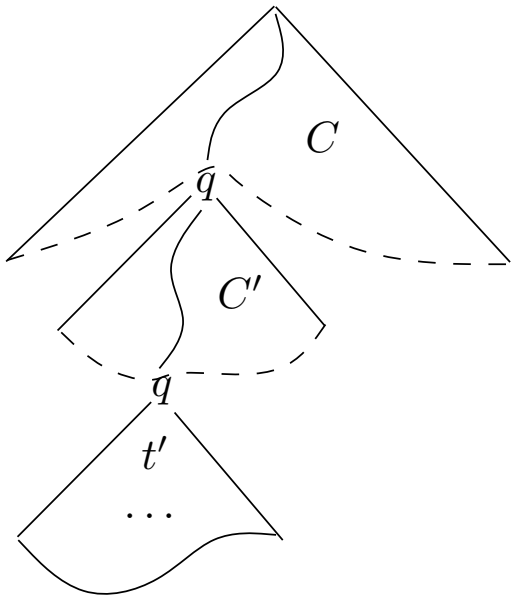
Pumping lemma. Let $L \subseteq T_\Sigma$ be a regular tree language.

Then there exists $k \in \mathbb{N}$ s.t.

if the depth of a tree $t \in L$ is $\geqslant k$ (the root is of depth 0),
then $\exists C, C' \in \mathcal{C}_\Sigma$ and $t' \in T_\Sigma$ s.t.
$t = C[C'[t']]$ *and for all* $n \in \mathbb{N}$, $C[C'^n[t']] \in L$.

Proof.

Let L defined by a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ and $k = |Q|$. □



Pumping lemma

Pumping lemma. Let $L \subseteq T_\Sigma$ be a regular tree language.

Then there exists $k \in \mathbb{N}$ s.t.

if the depth of a tree $t \in L$ *is* $\geqslant k$ *(the root is of depth 0),*
then $\exists C, C' \in \mathcal{C}_\Sigma$ *and* $t' \in T_\Sigma$ *s.t.*
$t = C[C'[t']]$ *and for all* $n \in \mathbb{N}$, $C[C'^n[t']] \in L$.

Proof.

Let L defined by a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$ and $k = |Q|$. □

Application of pumping lemma:

The tree language $\{f(g^i(a), g^i(a)) \mid i \in \mathbb{N}\}$ is not regular.

Outline

Regular tree grammar (RTG)

A regular tree grammar: $G = (\mathcal{N}, \Sigma, S, \mathcal{R})$, where

- $\mathcal{N}$ nonterminals,
- Σ: terminals,
- $S \in \mathcal{N}$: Start symbol,
- $\mathcal{R}$ contains rules of the form $A \to \alpha$, where $\alpha \in T_{\Sigma \cup \mathcal{N}}$

  Symbols in $\mathcal{N}$ has arity zero.

Derivation relation $s \to_G t$:

$\exists C \in \mathcal{C}_{\Sigma \cup \mathcal{N}}, A \in \mathcal{N},$ *and* $A \to \alpha \in \mathcal{R}$ *s.t.* $s = C[A], t = C[\alpha]$.

The *language* generated by G (denoted $\mathcal{L}(G)$): $\mathcal{L}(G) = \{t \in T_\Sigma \mid S \to_G^* t\}$.

Example:

$$S \to \wedge(S, S), S \to \vee(S, S), S \to 0, S \to 1.$$

BUTA ≡ RTG

RTG ⇒ Reduced RTG

Reachable nonterminals ($R(G)$): $N \in \mathcal{N}$ is reachable from S iff

$$\exists t \in T_{\Sigma \cup \mathcal{N}},\ S \to_G^* t \text{ and } N \text{ occurs in } t.$$

Productive nonterminals ($P(G)$):

$$N \in \mathcal{N} \text{ is productive iff } \exists t \in T_\Sigma \text{ s.t. } N \to_G^* t.$$

Reduced regular tree grammars:

A RTG whose nonterminals are all reachable and productive.

Computation of $R(G)$ (resp. $P(G)$): Computation of a fixpoint.

- $R_0(G) = \{S\}$ (resp. $P_0(G) = \{N \in \mathcal{N} \mid \exists \alpha \in T_\Sigma.\ N \to \alpha\}$),
- $R_{i+1}(G) = R_i(G) \cup \{N \in \mathcal{N} \mid \exists N' \in R_i(G).\ N' \to \alpha, N \text{ occurs in } \alpha\}$ (resp. $P_{i+1}(G) = P_i(G) \cup \{N \in \mathcal{N} \mid \exists \alpha \in T_{\Sigma \cup P_i(G)}.\ N \to \alpha\}$).

BUTA ≡ RTG

Reduced RTG ⇒ Normalized RTG

Normalized RTG:

Reduced RTG whose rules are of the form
$A \to a(A_1, \ldots, A_k)$ *or* $A \to a$ *(where* $a \in \Sigma, A_1, \ldots, A_k \in \mathcal{N}$*).*

Repeat the following rule until a normalized RTG is obtained:

Select a rule $A \to \alpha$ *not of the desired form, let* $\alpha = a(\alpha_1, \ldots, \alpha_k)$*,*
introduce new nonterminals $A_1, \ldots, A_k$*,*
replace $A \to \alpha$ *by* $A \to a(A_1, \ldots, A_k)$*,* $A_1 \to \alpha_1, \ldots, A_k \to \alpha_k$*.*

BUTA ≡ RTG

Reduced RTG ⇒ Normalized RTG

Normalized RTG:

$$\textit{Reduced RTG whose rules are of the form}$$
$$A \to a(A_1, \ldots, A_k) \textit{ or } A \to a \textit{ (where } a \in \Sigma, A_1, \ldots, A_k \in \mathcal{N}\textit{)}.$$

Repeat the following rule until a normalized RTG is obtained:

Select a rule $A \to \alpha$ *not of the desired form, let* $\alpha = a(\alpha_1, \ldots, \alpha_k)$,
introduce new nonterminals $A_1, \ldots, A_k$,
replace $A \to \alpha$ *by* $A \to a(A_1, \ldots, A_k)$, $A_1 \to \alpha_1, \ldots, A_k \to \alpha_k$.

Normalized RTG ⇒ TDTA

Idea: Nonterminals as states,
$$A \to a(A_1, \ldots, A_k) \Rightarrow (A, a, A_1, \ldots, A_k) \in \delta,$$
$$A \to a \Rightarrow (A, a) \in \delta.$$

TDTA ⇒ RTG

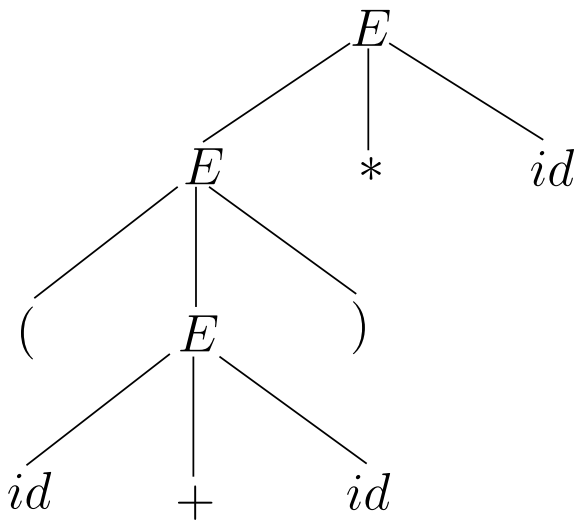
Idea: States as nonterminals,
$$(q, a, q_1, \ldots, q_k) \in \delta \Rightarrow q \to a(q_1, \ldots, q_k).$$

Outline

Yield operator

Compute a word from a tree
by concatenating the leaves of the tree from the left to the right.

- $\mathrm{Yield}(a) = a$ if $a \in \Sigma$ and $\mathrm{arity}(a) = 0$,
- $\mathrm{Yield}(a(t_1, \ldots, t_n)) = \mathrm{Yield}(t_1) \ldots \mathrm{Yield}(t_n)$.



$$\mathrm{Yield}(t) = (id + id) * id.$$

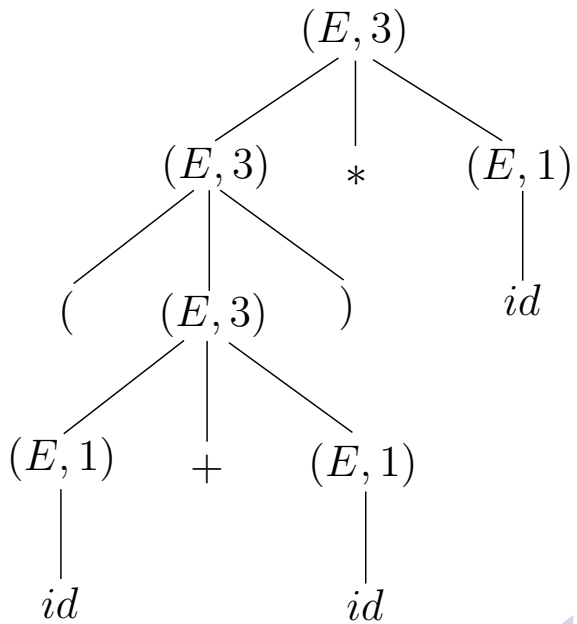
Ranked derivation trees of CFG

Let $G = (\mathcal{N}, \Sigma, S, \mathcal{R})$ be a CFG.
For each $A \in \mathcal{N}$ s.t. there is a rule $A \to \alpha \in \mathcal{R}$ with $|\alpha| = k$,
introduce a new nonterminal (A, k).
A ranked derivation tree is obtained from a derivation tree as follows:

For every node x labeled by A with k-children, replace A by (A, k).

Let $DT_R(G)$ denote the set of ranked derivation trees of words in $L(G)$.

$$E := id \mid E * E \mid E + E \mid (E)$$



CFL and Regular tree languages

Theorem. The following fact hold.

1. For each CFG G, $DT_R(G)$ is a regular tree language.
2. For each regular tree language $L \subseteq T_\Sigma$, Yield(L) is a CFL.
3. There exists a regular tree language
   which is not the set of derivation trees of any CFG.

Proof.

1. CFG $G = (\mathcal{N}, \Sigma, \mathcal{P}, S) \Rightarrow$ RTG $G' = (\mathcal{N}, \Sigma', S, \mathcal{R})$
   - $\Sigma' = \Sigma \cup \{\varepsilon\} \cup \{(A, p) \mid \exists A \to \alpha \text{ s.t. } |\alpha| = p\}$.
   - $\mathcal{R}$: $A \to a_1 \dots a_p \Rightarrow A \to (A, p)(a_1, \dots, a_p)$,
     $A \to \varepsilon \Rightarrow A \to (A, 0)(\varepsilon)$.
2. Let L generated by a normalized RTG $G = (\mathcal{N}, \Sigma, S, \mathcal{R})$.
   Then the CFG $(\mathcal{N}, \Sigma, \mathcal{P}, S)$ generates Yield(L), where
   $$\mathcal{P} : A \to a(A_1, \dots, A_k) \Rightarrow A \to A_1 \dots A_k.$$
3. The regular tree language $L = \{f(g(a), g(b))\}$
   is not the derivation tree of any CFG. □

Outline

Syntax of regular tree expressions

Regular expressions:

$$r := a \mid r_1 \cup r_2 \mid r_1 \cdot r_2 \mid r_1^*$$

Regular tree expressions:

How to define concatenation and star operator over ranked trees ?

Main idea:

Use a finite set of ports $\mathcal{P} = \{\Box_1, \ldots, \Box_k\}$
to denote the positions of trees where concatenations happen

Syntax of regular tree expressions

Regular expressions:

$$r := a \mid r_1 \cup r_2 \mid r_1 \cdot r_2 \mid r_1^*$$

Regular tree expressions:

How to define concatenation and star operator over ranked trees ?

Main idea:

Use a finite set of ports $\mathcal{P} = \{\Box_1, \ldots, \Box_k\}$
to denote the positions of trees where concatenations happen

Regular tree expressions over Σ and $\mathcal{P}$ (denoted by $\mathrm{Reg}(\Sigma, \mathcal{P})$):

$$r := a \mid f(r_1, \ldots, r_n) \mid r_1 \cup r_2 \mid r_1 \cdot_{\Box_i} r_2 \mid (r_1)^{*,\Box_i},$$

where

- $a \in \Sigma \cup \mathcal{P}$ such that $\mathrm{arity}(a) = 0$ (where ports in $\mathcal{P}$ have arity 0),
- $f \in \Sigma$ such that $\mathrm{arity}(f) = n$,
- $i : 1 \leqslant i \leqslant k$ and $\Box_i \in \mathcal{P}$.

Semantics of regular tree expressions

$$r := a \mid f(r_1, \ldots, r_n) \mid r_1 \cup r_2 \mid r_1 \cdot_{\Box_i} r_2 \mid (r_1)^{*,\Box_i}$$

- $[\![a]\!] = \{a\}$, $[\![f(r_1, \ldots, r_n)]\!] = \{f(s_1, \ldots, s_n) \mid s_1 \in [\![r_1]\!], \ldots, s_n \in [\![r_n]\!]\}$,
- $[\![r_1 \cup r_2]\!] = [\![r_1]\!] \cup [\![r_2]\!]$, $[\![r_1 \cdot_{\Box_i} r_2]\!] = [\![r_1]\!] \cdot_{\Box_i} [\![r_2]\!]$, $[\![(r_1)^{*,\Box_i}]\!] = [\![r_1]\!]^{*,\Box_i}$.

Suppose $L, L' \subseteq T_{\Sigma \cup \mathcal{P}}$, then $L \cdot_{\Box_i} L' = \bigcup_{t \in L} t[\Box_i \leftarrow L']$, where

$t[\Box_i \leftarrow L']$ is defined as follows:

Suppose $\Box_i$ occurs n times in t,
then $t[\Box_i \leftarrow L']$ is the set of trees obtained from t by
choosing n trees $s_1, \ldots, s_n \in L'$
and replacing the j-th (left-to-right) occurrence of $\Box_i$ by s_j.

Let $L \subseteq T_{\Sigma \cup \mathcal{P}}$, then

$$L^{0,\Box_i} = \{\Box_i\}, \text{ and } L^{n+1,\Box_i} = L \cdot_{\Box_i} L^{n,\Box_i}.$$

In addition, let $L^{*,\Box_i} = \bigcup_{n \in \mathbb{N}} L^{n,\Box_i}$.

Example: $(f(\Box_1, 0))^{*,\Box_1} \cdot_{\Box_1} 1$.

BUTA ≡ Regular tree expressions

From regular tree expressions to BUTA

By induction on the structure of regular tree expressions, show that for every regular tree expression r, $[\![r]\!]$ is a regular tree language.

Proposition. If L and L' are regular tree languages, then $L \cdot_{\Box_i} L'$ and $L^{*,\Box_i}$ are also regular tree languages.

$L \cdot_{\Box_i} L'$:

Suppose $\mathcal{A} = (Q, \Sigma \cup \mathcal{P}, \delta, F)$ and $\mathcal{A}' = (Q', \Sigma \cup \mathcal{P}, \delta', F')$ are two BUTAs defining L and L' such that $Q \cap Q' = \varnothing$.

Then $L \cdot_{\Box_i} L'$ is defined by $\mathcal{B} = (Q'', \Sigma \cup \mathcal{P}, \delta'', F'')$, where

- $Q'' = Q \cup Q'$,
- $F'' = F$,
- $\delta'' = \delta \cup \delta' \cup \{(q'_1, \ldots, q'_n, \sigma, q) \mid (\Box_i, q) \in \delta, \exists q' \in F'.(q'_1, \ldots, q'_n, \sigma, q') \in \delta'\}$.

BUTA ≡ Regular tree expressions

From regular tree expressions to BUTA

By induction on the structure of regular tree expressions, show that for every regular tree expression r, $[\![r]\!]$ is a regular tree language.

Proposition. If L and L' are regular tree languages, then $L \cdot_{\Box_i} L'$ and $L^{*,\Box_i}$ are also regular tree languages.

$L^{*,\Box_i}$:

Suppose $\mathcal{A} = (Q, \Sigma \cup \mathcal{P}, \delta, F)$ is a BUTA defining L such that

if $(\Box_i, q) \in \delta$, *then there are no transitions* $(q_1, \ldots, q_n, \sigma, q)$ *with* $\sigma \in \Sigma$.

The BUTA $\mathcal{A}' = (Q, \Sigma \cup \mathcal{P}, \delta', F')$ defines $L^{*,\Box_i}$, where

- $\delta' = \delta \cup \{(q_1, \ldots, q_n, \sigma, q) \mid (\Box_i, q) \in \delta, \exists q' \in F.\ (q_1, \ldots, q_n, \sigma, q') \in \delta\}$,
- $F' = \{q \mid (\Box_i, q) \in \delta\}$.

BUTA ≡ Regular tree expressions

From BUTA to regular tree expressions

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a BUTA such that $Q = \{q_1, \ldots, q_n\}$.

The goal:

Define a regular tree expression $r_{\mathcal{A}}$ over Σ and Q for $\mathcal{L}(\mathcal{A})$.

The idea: Notation $T(i, j, K)$ s.t. $1 \leqslant i \leqslant n$, $0 \leqslant j \leqslant n$, and $K \subseteq Q$.

The set of trees $t = (D, L)$ satisfying that $\exists$ a run of $\mathcal{A}$ over t, say $r_{\mathcal{A},t}$, s.t.

- $r_{\mathcal{A},t}(\varepsilon) = q_i$,
- *for every leaf x in t, $L(x) \in K$ or $L(x) = a \in \Sigma$ s.t.* $\mathrm{arity}(a) = 0$,
- *and for all the non-leaf nodes $x \neq \varepsilon$ in t, $r_{\mathcal{A},t}(x) \in \{q_1, \ldots, q_j\}$.*

$$\mathcal{L}(\mathcal{A}) = \bigcup_{q_i \in F} T(i, n, \varnothing).$$

BUTA ≡ Regular tree expressions

From BUTA to regular tree expressions

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a BUTA such that $Q = \{q_1, \ldots, q_n\}$.

The goal:

Define a regular tree expression $r_{\mathcal{A}}$ over Σ and Q for $\mathcal{L}(\mathcal{A})$.

The idea: Notation $T(i, j, K)$ s.t. $1 \leqslant i \leqslant n$, $0 \leqslant j \leqslant n$, and $K \subseteq Q$.

Induction on j.

$j = 0$:

$T(i, 0, K)$ is the set of trees $q_i \in K$, or $\sigma \in \Sigma$ s.t. $\text{arity}(\sigma) = 0$ and $(\sigma, q_i) \in \delta$, or trees $\sigma(\theta_1, \ldots, \theta_k)$ s.t.

- $\text{arity}(\sigma) = k$, $\forall j : 1 \leqslant j \leqslant k$, $\theta_j \in K$ or $\theta_j \in \Sigma$ s.t. $\text{arity}(\theta_j) = 0$,
- there are $q'_1, \ldots, q'_k \in Q$ s.t. $\forall j : 1 \leqslant j \leqslant k$, either $q'_j = \theta_j$ or $(\theta_j, q'_j) \in \delta$, and $(q'_1, \ldots, q'_k, \sigma, q_i) \in \delta$.

$j > 0$:

$$T(i, j, K) = \begin{array}{c} T(i, j-1, K) \cup \\ T(i, j-1, K \cup \{q_j\}) \cdot_{q_j} T(j, j-1, K \cup \{q_j\})^{*, q_j} \cdot_{q_j} T(j, j-1, K) \end{array}.$$

From BUTA to regular tree expressions: An example

Example: Boolean expressions that evaluate to "true".

BUTA $\mathcal{A} = (\{q_1, q_2\}, \{\wedge, \vee, 0, 1\}, \delta, \{q_2\})$, where δ is defined as follows,

- $(0, q_1), (q_1, q_1, \vee, q_1), (q_1, q_1, \wedge, q_1), (q_2, q_1, \wedge, q_1), (q_1, q_2, \wedge, q_1) \in \delta$,
- $(1, q_2), (q_2, q_1, \vee, q_2), (q_1, q_2, \vee, q_2), (q_2, q_2, \vee, q_2), (q_2, q_2, \wedge, q_2) \in \delta$.

From BUTA to regular tree expressions: An example

Example: Boolean expressions that evaluate to "true".

BUTA $\mathcal{A} = (\{q_1, q_2\}, \{\wedge, \vee, 0, 1\}, \delta, \{q_2\})$, where δ is defined as follows,

- $(0, q_1), (q_1, q_1, \vee, q_1), (q_1, q_1, \wedge, q_1), (q_2, q_1, \wedge, q_1), (q_1, q_2, \wedge, q_1) \in \delta$,
- $(1, q_2), (q_2, q_1, \vee, q_2), (q_1, q_2, \vee, q_2), (q_2, q_2, \vee, q_2), (q_2, q_2, \wedge, q_2) \in \delta$.

Then

- $T(2, 2, \varnothing) = T(2, 1, \varnothing) \cup T(2, 1, 2) \cdot_{q_2} T(2, 1, 2)^{*, q_2} \cdot_{q_2} T(2, 1, \varnothing)$.
- $T(2, 1, \varnothing) = T(2, 0, \varnothing) \cup T(2, 0, 1) \cdot_{q_1} T(1, 0, 1)^{*, q_1} \cdot_{q_1} T(1, 0, \varnothing)$.
- $T(2, 1, 2) = T(2, 0, 2) \cup T(2, 0, 1) \cdot_{q_1} T(1, 0, \{1, 2\})^{*, q_1} \cdot_{q_1} T(1, 0, 2)$.
- $T(2, 0, \varnothing) = 1 \cup \vee(0, 1) \cup \vee(1, 0) \cup \vee(1, 1) \cup \wedge(1, 1)$,
- $T(1, 0, \varnothing) = 0 \cup \vee(0, 0) \cup \wedge(0, 0) \cup \wedge(1, 0) \cup \wedge(0, 1)$,
- $T(2, 0, 1) = T(2, 0, \varnothing) \cup \vee(1, q_1) \cup \vee(q_1, 1)$,
- $T(1, 0, 1) = T(1, 0, \varnothing) \cup \vee(q_1, 0) \cup \vee(0, q_1) \cup \vee(q_1, q_1) \cup \wedge(0, q_1) \cup \wedge(q_1, 0) \cup \wedge(q_1, 1) \cup \wedge(1, q_1) \cup \wedge(q_1, q_1)$,
- $T(2, 0, 2) = T(2, 0, \varnothing) \cup \vee(0, q_2) \cup \vee(q_2, 0) \cup \vee(q_2, q_2) \cup \wedge(q_2, 1) \cup \wedge(1, q_2) \cup \wedge(q_2, q_2)$,
- $T(1, 0, 2) = T(1, 0, \varnothing) \cup \wedge(0, q_2) \cup \wedge(q_2, 0)$,
- $T(1, 0, \{1, 2\}) = T(1, 0, 1) \cup T(1, 0, 2) \cup \wedge(q_1, q_2) \cup \wedge(q_2, q_1)$.

Outline

MSO over ranked trees

Let Σ be a ranked alphabet and $k = \max\{\text{arity}(\sigma) \mid \sigma \in \Sigma\}$.

Syntax of $MSO[(P_\sigma)_{\sigma\in\Sigma}, (\text{suc}_i)_{1\leqslant i\leqslant k}]$

$$\varphi := P_\sigma(x) \mid x = y \mid suc_i(x,y) \mid x < y \mid X(x) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists x\varphi_1 \mid \exists X\varphi_1$$

Semantics

Let $t \in T_\Sigma$, then $t = (D, L)$ can be seen as
a relational structure $(D, (P_\sigma)_{\sigma\in\Sigma}, (\text{suc}_i)_{1\leqslant i\leqslant k}, <)$ s.t.
$P_\sigma = \{x \in D \mid L(x) = \sigma\}$, $\text{suc}_i = \{(x, x(i-1)) \mid x, x(i-1) \in D\}$
and $<$ is the transitive closure of $\bigcup_i \text{suc}_i$.

Then MSO formulas are interpreted over $t \in T_\Sigma$ in a natural way.

Example: There is a path of even length in the tree

$\exists X_1X_2(\exists x(\varphi_\varepsilon(x) \wedge X_1(x)) \wedge \varphi_{path}(X_1, X_2) \wedge \varphi_{even}(X_1, X_2))$, where

- $\varphi_{path} := \forall x\forall y((X_1(x) \vee X_2(x)) \wedge (X_1(y) \vee X_2(y)) \rightarrow (x < y \vee x = y \vee y < x))$,
- $\varphi_{even} := \bigwedge_{r=1,2} \forall x\exists y((X_r(x) \wedge \neg\varphi_{leaf}(x)) \rightarrow \vee_i \text{suc}_i(x,y) \wedge X_{3-r}(y)) \wedge \exists x(\varphi_{leaf}(x) \wedge X_2(x))$,
- $\varphi_\varepsilon(x) := \forall y(x < y \vee x = y)$, $\varphi_{leaf}(x) := \neg\exists y(x < y)$.

MSO ≡ BUTA

From MSO to BUTA:

Normal form for $MSO[(P_\sigma)_{\sigma\in\Sigma}, (\mathrm{suc}_i)_{1\leqslant i\leqslant k}]$ formulas:

$$\varphi := X \subseteq P_\sigma \mid X \subseteq Y \mid Sing(X) \mid suc_i(X,Y) \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi_1 \mid \exists X \varphi_1$$

Induction on the structure of MSO formulas,

utilizing the closure of BUTAs under projection and Boolean operations.

From BUTA to MSO:

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a BUTA such that $Q = \{q_1, \ldots, q_n\}$. Then

$$\varphi := \exists q_1 \ldots q_n (\varphi_{init} \wedge \varphi_{trans} \wedge \varphi_{acc}),$$

where

- $\varphi_{init} := \bigwedge_{\mathrm{arity}(\sigma)=0} \forall x((\varphi_{leaf}(x) \wedge P_\sigma(x)) \rightarrow \bigvee_{(\sigma,q)\in\delta} q(x))$,
- $\varphi_{trans} := \bigwedge_{(q_1,\ldots,q_r,\sigma)} \forall x \forall y_1 \ldots y_r \left(\begin{array}{c} (P_\sigma(x) \wedge \wedge_i \mathrm{suc}_i(x,y_i) \wedge \wedge_i q_i(y_i)) \\ \rightarrow \bigvee_{(q_1,\ldots,q_r,\sigma,q)\in\delta} q(x) \end{array} \right)$,
- $\varphi_{acc} = \exists x(\varphi_\varepsilon(x) \wedge \bigvee_{q\in F} q(x))$.

Outline

Myhill-Nerode theorem for tree languages

Let $L \subseteq T_\Sigma$. Define a congruence $\sim_L$ on T_Σ as follows:

$$t \sim_L t' \text{ iff for all contexts } C,\ C[t] \in L \Leftrightarrow C[t'] \in L.$$

Proposition. $\sim_L$ is a congruence, that is,

if $t \sim_L t'$, then for all contexts C, $C[t] \sim_L C[t']$.

Proof.

For all contexts C', if $C'[C[t]] \in L$, then $[C'[C]][t] \in L$.
From the fact that $t \sim_L t'$, we have $[C'[C]][t'] \in L$, that is, $C'[C[t]] \in L$. □

Myhill-Nerode theorem for tree languages

Let $L \subseteq T_\Sigma$. Define a congruence $\sim_L$ on T_Σ as follows:

$$t \sim_L t' \text{ iff for all contexts } C,\ C[t] \in L \Leftrightarrow C[t'] \in L.$$

Theorem. Let $L \subseteq T_\Sigma$. Then L is regular iff $\sim_L$ is of finite index.

Proof.

"Only if" direction:
Suppose L is recognized by a deterministic BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$.
Define $\sim_\mathcal{A}$ as follows: $t \sim_\mathcal{A} t'$ iff $r_{\mathcal{A},t}(\varepsilon) = r_{\mathcal{A},t'}(\varepsilon)$.
Evidently, $\sim_\mathcal{A}$ is of finite index, since Q is finite. It remains to show $\sim_\mathcal{A} \subseteq \sim_L$.
If $t \sim_\mathcal{A} t'$, then $r_{\mathcal{A},t}(\varepsilon) = r_{\mathcal{A},t'}(\varepsilon)$. So $\forall$ context C, $r_{\mathcal{A},C[t]}(\varepsilon) = r_{\mathcal{A},C[t']}(\varepsilon)$.
Thus, for every context C, $C[t] \in L$ iff $C[t'] \in L$, we have $t \sim_L t'$.

"If" direction:
Suppose $\sim_L$ is of finite index. Define $\mathcal{A}_L = (Q_L, \Sigma, \delta_L, F_L)$ as follows.

- Q_L is the set of equivalence classes of $\sim_L$, $F_L = \{[t] \mid t \in L\}$,
- for every $\sigma \in \Sigma$ s.t. $\text{arity}(\sigma) = k$ and $[t_1], \ldots, [t_k] \in Q$,
  $([t_1], \ldots, [t_k], \sigma, [\sigma(t_1, \ldots, t_k)]) \in \delta_L$.

Myhill-Nerode theorem for tree languages

Let $L \subseteq T_\Sigma$. Define a congruence $\sim_L$ on T_Σ as follows:

$$t \sim_L t' \text{ iff for all contexts } C,\ C[t] \in L \Leftrightarrow C[t'] \in L.$$

Theorem. Let $L \subseteq T_\Sigma$. Then L is regular iff $\sim_L$ is of finite index.

Corollary. For every regular tree language $L \subseteq T_\Sigma$,
there is a unique deterministic BUTA of the minimum size defining L.

Minimization of deterministic tree automata

Let $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a deterministic BUTA defining L.
Compute inductively an equivalence relation $\approx_{\mathcal{A}}$ over Q as follows, until $\approx^{i}_{\mathcal{A}} = \approx^{i+1}_{\mathcal{A}}$.

- $q \approx^{0}_{\mathcal{A}} q'$ iff $q \in F \Leftrightarrow q' \in F$,
- $q \approx^{i+1}_{\mathcal{A}} q'$ iff $q \approx^{i}_{\mathcal{A}} q'$ and
  for every $\sigma \in \Sigma$ s.t. $\text{arity}(\sigma) = k$, $q_1, \ldots, q_{j-1}, q_{j+1}, \ldots, q_k \in Q$,
  $\delta(q_1, \ldots, q_{j-1}, q, q_{j+1}, \ldots, q_k, \sigma) \approx^{i}_{\mathcal{A}} \delta(q_1, \ldots, q_{j-1}, q', q_{j+1}, \ldots, q_k, \sigma)$.

Theorem. $\forall t, t' \in T_\Sigma$, $t \sim_L t'$ iff $r_{\mathcal{A},t}(\varepsilon) \approx_{\mathcal{A}} r_{\mathcal{A},t'}(\varepsilon)$.

Lemma. The following two facts hold.

1. For every $t, t' \in T_\Sigma$ and context C,
   $r_{\mathcal{A},t}(\varepsilon) \approx_{\mathcal{A}} r_{\mathcal{A},t'}(\varepsilon)$ implies $r_{\mathcal{A},C[t]}(\varepsilon) \approx_{\mathcal{A}} r_{\mathcal{A},C[t']}(\varepsilon)$.
2. $t \sim_L t'$ implies $r_{\mathcal{A},t}(\varepsilon) \approx_{\mathcal{A}} r_{\mathcal{A},t'}(\varepsilon)$.

Proof.

1. Induction on the depth of $\Box$ in C.
2. By an induction on i, prove that $t \sim_L t'$ implies $r_{\mathcal{A},t}(\varepsilon) \approx^{i}_{\mathcal{A}} r_{\mathcal{A},t'}(\varepsilon)$. $\Box$

Outline

Membership and emptiness

Membership problem: Given $t \in T_\Sigma$ and a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$, is $t \in \mathcal{L}(\mathcal{A})$?

Nonemptiness problem: Given a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$, is $\mathcal{L}(\mathcal{A}) \neq \varnothing$?

Theorem. The membership and nonemptiness problem can be solved in polynomial time.

Proof.

Membership: A bottom-up computation of the reachable states $(R_x)_{x \in t}$.
Let $t = (D, L) \in T_\Sigma$ and $\mathcal{A} = (Q, \Sigma, \delta, F)$ be a BUTA.

- for every leaf x of t, $R_x = \{q \mid (L(x), q) \in \delta\}$,
- for every non-leaf node x labeled by $L(x) = \sigma$ of arity k,
  $R_x = \{q \mid \exists q_1 \in R_{x0}, \ldots, q_k \in R_{x(k-1)}.\ (q_1, \ldots, q_k, \sigma, q) \in \delta\}$.

$t \in \mathcal{L}(\mathcal{A})$ iff $R_\varepsilon \cap F \neq \varnothing$

Nonemptiness: Bottom-up computation of the set of reachable states R.

- $R_0 = \{q \mid (\sigma, q) \in \delta\}$,
- $R_{i+1} = R_i \cup \{q \mid \exists \sigma \in \Sigma, \exists q_1, \ldots, q_k \in R_i.\ (q_1, \ldots, q_k, \sigma, q) \in \delta\}$.

$\mathcal{L}(\mathcal{A}) \neq \varnothing$ iff $R \cap F \neq \varnothing$ □

Language inclusion

Language inclusion: Given two BUTAs $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, F_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, F_2)$, is $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$?

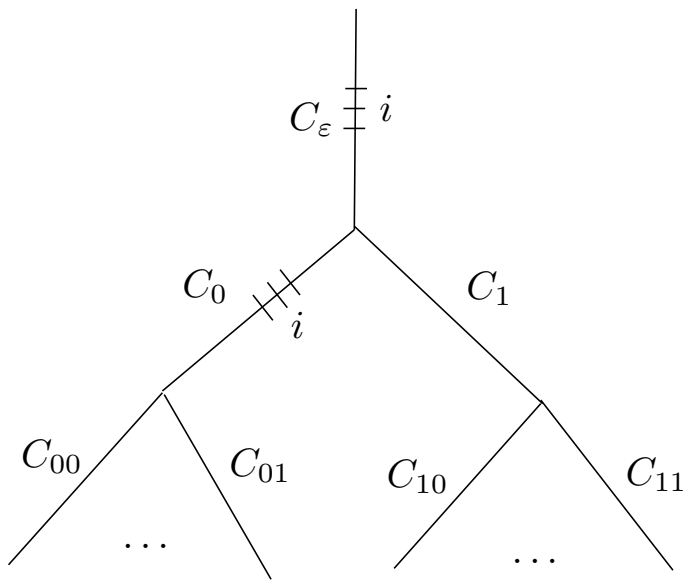
Universality: Given a BUTA $\mathcal{A} = (Q, \Sigma, \delta, F)$, is $\mathcal{L}(\mathcal{A}) = T_\Sigma$?

Theorem. The language inclusion and universality problem are EXPTIME-complete.

Proof.

Upper bound:
Construct a BUTA $\mathcal{A}_2'$ defining the complement of $\mathcal{L}(\mathcal{A}_2)$, decide whether $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2') = \varnothing$.

Lower bound (Universality):
Reduction from polynomial space alternating Turing machines.
Use a BUTA to describe the unsuccessful computations. □



Outline

Definition

We restrict our attention to binary trees.
Let $Types = \{r, 0, 1\} \times \{l, i\}$.
Let $t = (D, L)$ be a binary tree and $x \in D$, then the *type* of x (denoted by $type(x)$), is an element of $Types$ s.t.

- the first component of $type(x)$:
  r: the root, 0: the left-child, 1: the right-child,
- the second component of $type(x)$: l: a leaf, i: an internal node.

A tree walking automaton (TWA) $\mathcal{A}$ is a tuple $(Q, \Sigma, I, F, \delta)$, where

- Q: the set of states,
- I : the set of initial states, F: the set of accepting states,
- $\delta \subseteq Q \times Types \times \Sigma \times \{\uparrow, 0, 1\} \times Q$ s.t.

$$for\ every\ (q, (b, l), \sigma, act, q') \in \delta,\ we\ have\ act = \uparrow.$$

A *deterministic* TWA is a TWA $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ such that

- I is a singleton,
- $\forall (q, \tau, \sigma) \in Q \times Types \times \Sigma$, $\delta(q, \tau, \sigma)$ is a singleton.

Definition: continued

Let $\mathcal{A}$ be a TWA and t be a binary tree.

- A *configuration* of $\mathcal{A}$ over t is a pair (q, x) s.t. $q \in Q$ and x is a node in t.
- An *initial configuration* is a configuration (q, ε) with $q \in I$.
- A *run* of $\mathcal{A}$ over t is a sequence $(q_0, x_0)(q_1, x_1) \dots (q_n, x_n)$ such that
  - (q_0, x_0) is an initial configuration,
  - $\forall i : 0 \leqslant j < n$, $\exists act$ s.t. $(q_j, type(x_j), L(x_j), act, q_{j+1}) \in \delta$, and
    - if $act = \uparrow$ and $type(x_j) \in \{(0, l), (0, i)\}$, then $x_j = x_{j+1}0$,
    - if $act = \uparrow$ and $type(x_j) \in \{(1, l), (1, i)\}$, then $x_j = x_{j+1}1$,
    - if $act = 0$ and $type(x_j) \in \{r, 0, 1\} \times \{i\}$, then $x_{j+1} = x_j 0$,
    - if $act = 1$ and $type(x_j) \in \{r, 0, 1\} \times \{i\}$, then $x_{j+1} = x_j 1$.

  A run $(q_0, x_0)(q_1, x_1) \dots (q_n, x_n)$ is accepting iff $q_n \in F$.

  A tree t is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ over t.

TWA: Example

Depth-first search of trees

Three states $\{q, q_{left}, q_{right}\}$

- $(q, (b, i), a, 0, q) \in \delta$, where $b = r, 0, 1$,
  (q, x) s.t. x is not a leaf
  $\Rightarrow$ goes to the left child of x and the state is changed to q.
- $(q, (0, l), a, \uparrow, q_{left}) \in \delta$ (resp. $(q, (1, l), a, \uparrow, q_{right}) \in \delta$),
  (q, x) s.t. x is a leaf and the left (resp. right) child of its parent
  $\Rightarrow$ goes to the parent of x and the state is changed to q_{left} (resp. q_{right}).

TWA: Example

Depth-first search of trees

Three states $\{q, q_{left}, q_{right}\}$

- $(q, (b, i), a, 0, q) \in \delta$, where $b = r, 0, 1$,
  (q, x) s.t. x is not a leaf
  $\Rightarrow$ goes to the left child of x and the state is changed to q.
- $(q, (0, l), a, \uparrow, q_{left}) \in \delta$ (resp. $(q, (1, l), a, \uparrow, q_{right}) \in \delta$),
  (q, x) s.t. x is a leaf and the left (resp. right) child of its parent
  $\Rightarrow$ goes to the parent of x and the state is changed to q_{left} (resp. q_{right}).
- $(q_{left}, (b, i), a, 1, q) \in \delta$,
  (q_{left}, x) s.t. x is a not a leaf
  $\Rightarrow$ goes to the right child of x and the state is changed to q.
- $(q_{right}, (0, l), a, \uparrow, q_{left}), (q_{right}, (0, i), a, \uparrow, q_{left}) \in \delta$,
  (q_{right}, x) s.t. x is the left child of its parent
  $\Rightarrow$ goes to the parent of x and the state is changed to q_{left}.
- $(q_{right}, (1, l), a, \uparrow, q_{right}), (q_{right}, (1, i), a, \uparrow, q_{right}) \in \delta$,
  (q_{right}, x) s.t. x is the right child of its parent
  $\Rightarrow$ goes to the parent of x and the state is changed to q_{right}.

TWA $\subseteq$ BUTA

Theorem. From every TWA, an equivalent BUTA can be constructed.

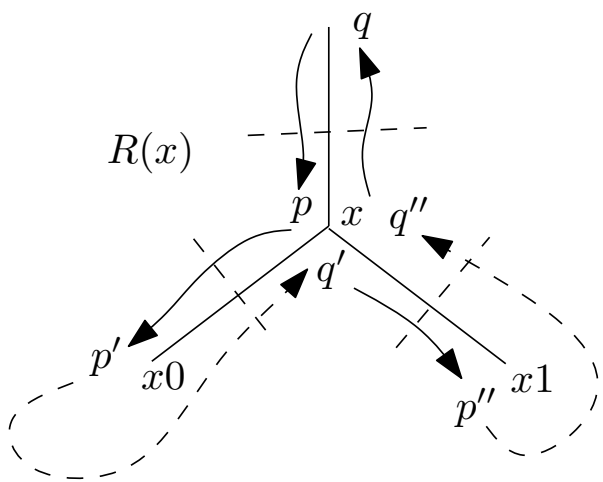
Let $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ be a TWA.
W.l.o.g. assume that for every tree t,
every accepting run of $\mathcal{A}$ over t stops at the root of t.

The idea:

For each non-leaf node x in $t = (D, L)$, define $R(x) \subseteq Q \times Q$ as follows.

$(p, q) \in R(x)$ *iff* $\exists$ *a partial run of* $\mathcal{A}$ *over* t, *say* $(q_1, x_1) \dots (q_n, x_n)$, *s.t.*

- $x_1 = x$, $q_1 = p$,
- $\forall i : 1 \leqslant i < n$, $x_i \in t|_x$,
- x_n *is the parent of* x, $q_n = q$, *and* $(q_{n-1}, type(x), L(x), q_n, \uparrow) \in \delta$.



Observation. $R(x)$ only depends on $t|_x$.

TWA $\subseteq$ BUTA

Theorem. From every TWA, an equivalent BUTA can be constructed.

Let $\mathcal{A} = (Q, \Sigma, I, F, \delta)$ be a TWA.
W.l.o.g. assume that for every tree t,
every accepting run of $\mathcal{A}$ over t stops at the root of t.

For every leaf node x,

$$R(x) = \{(p, q) \mid (p, type(x), L(x), q, \uparrow) \in \delta\}.$$

For every non-leaf node x, $R(x)$ is computed from $R(x0), R(x1)$ as follows.

If there exist $p_1, \ldots, p_k, q_1, \ldots, q_{k+1} \in Q, b_1, \ldots, b_k \in \{0, 1\}$ *s.t.*

- $\forall i : 1 \leqslant i \leqslant k.\ (q_i, type(x), L(x), p_i, b_i) \in \delta,\ (p_i, q_{i+1}) \in R(xb_i)$,
- $p = q_1,\ (q_{k+1}, type(x), L(x), q, \uparrow) \in \delta$,

then $(p, q) \in R(x)$.

There is an accepting run of $\mathcal{A}$ over t iff $R(\varepsilon) \cap (I \times F) \neq \varnothing$.

TWA $\subset$ BUTA

Let $\Sigma = \{a, b, c\}$ such that $\text{arity}(a) = \text{arity}(c) = 0$ and $\text{arity}(b) = 1$.

Theorem. The language $K \subseteq T_\Sigma$ defined in the following cannot be defined by TWAs.

Let $t \in T_\Sigma$ and x be a non-leaf node in t. Then

x is balanced if both $t|_{x0}$ and $t|_{x1}$ contain a leaf labeled by a.

Define $K \subseteq T_\Sigma$ as follows: $t \in K$ iff

for every leaf x labeled by a in t,
there are an even number of balanced proper ancestors of x.

Emptiness of TWA

Theorem. The emptiness of TWAs is EXPTIME-complete.

Proof.

Upper bound: Translation to BUTAs.

Lower bound: Reduction from Polynomial space alternating Turing machines.

Encode the successful computations of PSPACE ATMs by binary trees and check the consistency of adjacent configurations using TWAs.