Tree Automata and Applications

M1 course, 2023/2024

Organization

Timetable

- Exercises: Thursday 8:30 – 10:30 (Luc Lapointe)
- Course: Thursday 10:45 – 12:45 (Stefan Schwoon)

Exams

- DM or CC (*to be specified by Luc*)
- Final Exam: 2h, 11 January
- First session: DM/CC + Exam (50/50)
- Second session: DM/CC + Repeat Exam (50/50)

Course materials

- Website: lecturer's homepage + Wiki MPRI, course 1-18
  (exercise sheets, slides, former exams)
- Hubert Comon et al.
  Tree Automata Techniques and Applications.
  `http://tata.gforge.inria.fr/`

Motivations

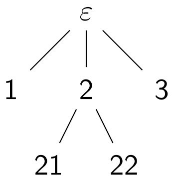
1. Natural extension of formal-language notions (automata, logic, . . .)
2. Treatment of tree-like data structures: parse tree, XML documents (XPath, CSS selectors)
3. Applications e.g. in compiler construction, formal verification

Trees

We consider *finite* *ordered* *ranked* trees.

- *ordered* : internal nodes have children $1 \dots n$
- *ranked* : number of children fixed by node's label

Let N denote the set of positive integers.
Nodes (*positions*) of a tree are associated with elements of N^*:



Definition: Tree

A (finite, ordered) *tree* is a non-empty, finite, prefix-closed set $Pos \subseteq N^*$ such that $w(i+1) \in Pos$ implies $wi \in Pos$ for all $w \in N^*$, $i \in N$.

Ranked Trees

Ranked symbols

Let $\mathcal{F}_0, \mathcal{F}_1, \ldots$ be disjoint sets of symbols of *arity* $0, 1, \ldots$
We note $\mathcal{F} := \bigcup_i \mathcal{F}_i$.

- Notation (example): $\mathcal{F} = \{f(2), g(1), a, b\}$

Let $\mathcal{X}$ denote a set of variables (disjoint from the other symbols).

Definition: Ranked tree

A ranked tree is a mapping $t : Pos \to (\mathcal{F} \cup \mathcal{X})$ satisfying:

- Pos is a tree;
- for all $p \in Pos$, if $t(p) \in \mathcal{F}_n$, $n \geq 1$ then $Pos \cap pN = \{p1, \ldots, pn\}$;
- for all $p \in Pos$, if $t(p) \in \mathcal{X} \cup \mathcal{F}_0$ then $Pos \cap pN = \emptyset$.

Trees and Terms

Definition: Terms

The set of *terms* $T(\mathcal{F}, \mathcal{X})$ is the smallest set satisfying:

- $\mathcal{X} \cup \mathcal{F}_0 \subseteq T(\mathcal{F}, \mathcal{X})$;
- if $t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{X})$ and $f \in \mathcal{F}_n$, then $f(t_1, \ldots, t_n) \in T(\mathcal{F}, \mathcal{X})$.

We note $T(\mathcal{F}) := T(\mathcal{F}, \emptyset)$. A term in $T(\mathcal{F})$ is called *ground term*.
A term of $T(\mathcal{F}, \mathcal{X})$ is *linear* if every variable occurs at most once.

Example: $\mathcal{F} = \{f(2), g(1), a, b\}$, $\mathcal{X} = \{x, y\}$

- $f(g(a), b) \in T(\mathcal{F})$;
- $f(x, f(b, y)) \in T(\mathcal{F}, \mathcal{X})$ is linear;
- $f(x, x) \in T(\mathcal{F}, \mathcal{X})$ is non-linear.

We confuse terms and trees in the obvious manner.

Height and size

Definition

Let $t \in T(\mathcal{F}, \mathcal{X})$. We note $\mathcal{H}(t)$ the *height* of t and $|t|$ the *size* of t.

- if $t \in \mathcal{X}$, then $\mathcal{H}(t) := 0$ and $|t| := 0$; (for notational convenience)
- if $t \in \mathcal{F}_0$, then $\mathcal{H}(t) := 1$ and $|t| := 1$;
- if $t = f(t_1, \ldots, t_n)$, then $\mathcal{H}(t) := 1 + \max\{\mathcal{H}(t_1), \ldots, \mathcal{H}(t_n)\}$ and $|t| := 1 + |t_1| + \cdots + |t_n|$.

Subterms / subtrees

Definition: Subtree

Let $t, u \in T(\mathcal{F}, \mathcal{X})$ and p a position. Then $t|_p : Pos_p \to T(\mathcal{F}, \mathcal{X})$ is the ranked tree defined by

- $Pos_p := \{ q \mid pq \in Pos \}$;
- $t|_p(q) := t(pq)$.

Moreover, $t[u]_p$ is the tree obtained by replacing $t|_p$ by u in t.

$t \trianglerighteq t'$ (resp. $t \triangleright t'$) denotes that t' is a (proper) subtree of t.

Substitutions and Context

Definition: Substitution

- (Ground) substitution σ: mapping from $\mathcal{X}$ to $T(\mathcal{F}, \mathcal{X})$ resp. $T(\mathcal{F})$
- Notation: $\sigma := \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$, with $\sigma(x) := x$ for all $x \in \mathcal{X} \setminus \{x_1, \ldots, x_n\}$
- Extension to terms: for all $f \in \mathcal{F}_m$ and $t'_1, \ldots, t'_m \in T(\mathcal{F}, \mathcal{X})$
$$\sigma(f(t'_1, \ldots, t'_m)) = f(\sigma(t'_1), \ldots, \sigma(t'_m))$$
- Notation: $t\sigma$ for $\sigma(t)$

Definition: Context

A *context* is a linear term $C \in T(\mathcal{F}, \mathcal{X})$ with variables $x_1, \ldots, x_n$.
We note $C[t_1, \ldots, t_n] := C\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$.

$\mathcal{C}^n(\mathcal{F})$ denotes the contexts with n variables and $\mathcal{C}(\mathcal{F}) := \mathcal{C}^1(\mathcal{F})$.
Let $C \in \mathcal{C}(\mathcal{F})$. We note $C^0 := x_1$ and $C^{n+1} = C^n[C]$ for $n \geq 0$.

Tree automata

Basic idea: Extension of finite automata from words to trees
Direct extension of automata theory when words seen as unary terms:

$$abc \mathrel{\widehat{=}} a(b(c(\$)))$$

Finite automaton: labels every prefix of a word with a state.
Tree automaton: labels every position/subtree of a tree with a state.

Two variants: bottom-up vs top-down labelling

Basic results (preview)

- Non-deterministic bottom-up and top-down are equally powerful
- Deterministic bottom-up equally powerful
- Deterministic top-down less powerful

Bottom-up automata

Definition: (Bottom-up tree automata)

A *(finite bottom-up) tree automaton* (NFTA) is a tuple $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$, where:

- Q is a finite set of *states*;
- $\mathcal{F}$ a finite ranked alphabet;
- $G \subseteq Q$ are the *final states*;
- Δ is a finite set of rules of the form
$$f(q_1, \ldots, q_n) \to q$$
for $f \in \mathcal{F}_n$ and $q, q_1, \ldots, q_n \in Q$.

Example: $Q := \{q_0, q_1, q_f\}$, $\mathcal{F} = \{f(2), g(1), a\}$, $G := \{q_f\}$, and rules
$$a \to q_0 \quad g(q_0) \to q_1 \quad g(q_1) \to q_1 \quad f(q_1, q_1) \to q_f$$

Move relation and computation tree

Move relation

Let $t, t' \in T(\mathcal{F}, Q)$. We write $t \rightarrow_{\mathcal{A}} t'$ if the following are satisfied:

- $t = C[f(q_1, \ldots, q_n)]$ for some context C;
- $t' = C[q]$ for some rule $f(q_1, \ldots, q_n) \rightarrow q$ of $\mathcal{A}$.

Idea: successively reduce t to a single state, starting from the leaves.
As usual, we write $\rightarrow_{\mathcal{A}}^{*}$ for the transitive and reflexive closure of $\rightarrow_{\mathcal{A}}$.

Computation

Let $t : Pos \rightarrow \mathcal{F}$ a ground tree. A *run* or *computation* of $\mathcal{A}$ on t is a labelling $t' : Pos \rightarrow Q$ compatible with Δ, i.e.:

- for all $p \in Pos$, if $t(p) = f \in \mathcal{F}_n$, $t'(p) = q$, and $t'(pj) = q_j$ for all $pj \in Pos \cap pN$, then $f(q_1, \ldots, q_n) \rightarrow q \in \Delta$

Regular tree languages

A tree t is *accepted* by $\mathcal{A}$ iff $t \rightarrow^*_{\mathcal{A}} q$ for some $q \in G$.

$\mathcal{L}(\mathcal{A})$ denotes the set of trees accepted by $\mathcal{A}$.

L is *regular/recognizable* iff $L := \mathcal{L}(\mathcal{A})$ for some NFTA $\mathcal{A}$.

Two NFTAs $\mathcal{A}_1$ and $\mathcal{A}_2$ are *equivalent* iff $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

NFTA with ε-moves

Definition:

An ε-NFTA is an NFTA $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$, where Δ can additionally contain rules of the form $q \to q'$, with $q, q' \in Q$.

Semantics: Allow to re-label a position from q to q'.

Equivalence of ε-NFTA

For every ε-NFTA $\mathcal{A}$ there exists an equivalent NFTA $\mathcal{A}'$.

Proof (sketch): Construct the rules of $\mathcal{A}'$ by a saturation procedure.

Deterministic, complete, and reduced NFTA

An NFTA is *deterministic* if no two rules have the same left-hand side.
An NFTA is *complete* if for every $f \in \mathcal{F}_n$ and $q_1, \ldots, q_n \in Q$, there exists at least one rule $f(q_1, \ldots, q_n) \to q \in \Delta$.

As usual, a DFTA has *at most* one run per tree.
A DCFTA as *exactly* one run per tree.

A state q of $\mathcal{A}$ is *accessible* if there exists a tree t s.t. $t \to^*_{\mathcal{A}} q$.
$\mathcal{A}$ is said to be *reduced* if all its states are accessible.

A pumping lemma for tree languages

Lemma

Let L be recognizable. Then there exists a constant k such that for all $t \in L$ with $\mathcal{H}(t) > k$ there exist contexts $C, D \in \mathcal{C}(\mathcal{F})$ and $u \in T(\mathcal{F})$ satisfying:

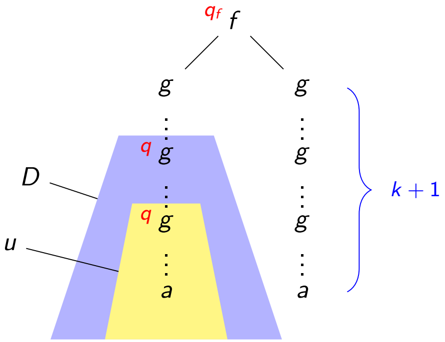
- D is non-trivial (i.e. not just a variable);
- $t = C[D[u]]$;
- for all $n \geq 0$, we have $C[D^n[u]] \in L$.

Proof: Let k be the number of states of an NFTA $\mathcal{A}$ recognizing L. Then an accepting run for t has positions p, pp' ($p' \neq \varepsilon$) labelled with the same state q. Let $C := t[x]_p$, $D := t|_p[x]_{p'}$, and $u := t|_{pp'}$. We have $t = C[D[u]] \in L$, $D[u] \rightarrow^*_{\mathcal{A}} q$, and $u \rightarrow^*_{\mathcal{A}} q$, hence the accepting run of t implies $D[q] \rightarrow^*_{\mathcal{A}} q$ and $C[q] \rightarrow^*_{\mathcal{A}} q_f$, for some final q_f. Therefore, $C[u] \rightarrow^*_{\mathcal{A}} q_f$ and for any $n \geq 0$, (by induction)

$$C[D^{n+1}[u]] \rightarrow^*_{\mathcal{A}} C[D^n[D[q]]] \rightarrow^*_{\mathcal{A}} C[D^n[q]] \rightarrow^*_{\mathcal{A}} C[q] \rightarrow^*_{\mathcal{A}} q_f$$

Illustration of pumping lemma

Let $L = \{ f(g^i(a), g^i(a)) \mid i \geq 0 \}$ for $\mathcal{F} = \{f(2), g(1), a\}$.
Suppose (by contradiction) that L is recognizable by NFTA $\mathcal{A}$ with k states. Let $t = f(g^k(a), g^k(a))$.



Pumping D creates trees outside L $\Rightarrow$ L not recognizable.

Top-down tree automata

Definition

A *top-down tree automaton* (T-NFTA) is a tuple $\mathcal{A} = \langle Q, \mathcal{F}, I, \Delta \rangle$, where $Q, \mathcal{F}$ are as in NFTA, $I \subseteq Q$ is a set of *initial states*, and Δ contains rules of the form

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

for $f \in \mathcal{F}_n$ and $q, q_1, \ldots, q_n \in Q$.

Move relation: $t \to_{\mathcal{A}} t'$ iff

- $t = C[q(f(t_1, \ldots, t_n))]$ for some context C, $f \in \mathcal{F}_n$, and $t_1, \ldots, t_n \in T(\mathcal{F})$;
- $t' = C[f(q_1(t_1), \ldots, q_n(t_n))]$ for some rule $q(f) \to (q_1, \ldots, q_n)$.

t is accepted by $\mathcal{A}$ if $q(t) \to^*_{\mathcal{A}} t$ for some $q \in I$.

From top-down to bottom-up

Theorem (T-NFTA = NFTA)

L is recognizable by an NFTA iff it is recognizable by a T-NFTA.

Claim: L is accepted by NFTA $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ iff it is accepted by T-NFTA $\mathcal{A}' = \langle Q, \mathcal{F}, G, \Delta' \rangle$, with

$$\Delta' := \{\, q(f) \to (q_1, \ldots, q_n) \mid f(q_1, \ldots, q_n) \to q \in \Delta \,\}$$

Proof: Let $t \in T(\mathcal{F})$. We show $t \to^*_{\mathcal{A}} q$ iff $q(t) \to^*_{\mathcal{A}'} t$.

- Base: $t = a$ (for some $a \in \mathcal{F}_0$)

$$t = a \to^*_{\mathcal{A}} q \iff a \to_{\Delta} q \iff q(a) \to_{\Delta'} \varepsilon \iff q(a) \to^*_{\mathcal{A}'} a$$

- Induction: $t = f(t_1, \ldots, t_n)$, hypothesis holds for $t_1, \ldots, t_n$

$$f(t_1, \ldots, t_n) \to^*_{\mathcal{A}} q \iff \exists q_1, \ldots q_n : f(q_1, \ldots, q_n) \to_{\Delta} q \wedge \forall i : t_i \to^*_{\mathcal{A}} q_i$$

$$\iff \exists q_1, \ldots, q_n : q(f) \to_{\Delta'} (q_1, \ldots, q_n) \wedge \forall i : q_i(t_i) \to^*_{\mathcal{A}'} t_i$$

$$\iff q(f(t_1, \ldots, t_n)) \to_{\mathcal{A}'} f(q_1(t_1), \ldots, q_n(t_n)) \to^*_{\mathcal{A}'} f(t_1, \ldots, t_n)$$

From NFTA to DFTA

Theorem (NFTA=DFTA)

If L is recognizable by an NFTA, then it is recognizable by a DFTA.

Claim (subset construct.): Let $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ an NFTA recognizing L. The following DCFTA $\mathcal{A}' = \langle 2^Q, \mathcal{F}, G', \Delta' \rangle$ also recognizes L:

- $G' = \{ S \subseteq Q \mid S \cap G \neq \emptyset \}$
- for every $f \in \mathcal{F}_n$ and $S_1, \ldots, S_n \subseteq Q$, let $f(S_1, \ldots, S_n) \to S \in \Delta'$, where $S = \{ q \in Q \mid \exists q_1 \in S_1, \ldots, q_n \in S_n : f(q_1, \ldots, q_n) \to q \in \Delta \}$

Proof: For $t \in T(\mathcal{F})$, show $t \to^*_{\mathcal{A}'} \{ q \mid t \to^*_{\mathcal{A}} q \}$, by structural induction.

DFTA with accessible states

In practice, the construction of $\mathcal{A}'$ can be restricted to accessible states: Start with transitions $a \to S$, then saturate.

Deterministic top-down are less powerful

E.g., $L = \{f(a, b), f(b, a)\}$ can be recognized by DFTA but not by T-DFTA.

Closure properties

Theorem (Boolean closure)

Recognizable tree languages are closed under Boolean operations.

Negation (invert accepting states)

Let $\langle Q, \mathcal{F}, G, \Delta \rangle$ be a DCFTA recognizing L.
Then $\langle Q, \mathcal{F}, Q \setminus G, \Delta \rangle$ recognizes $T(\mathcal{F}) \setminus L$.

Union (juxtapose)

Let $\langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ be NFTA recognizing L_i, for $i = 1, 2$.
Then $\langle Q_1 \uplus Q_2, \mathcal{F}, G_1 \cup G_2, \Delta_1 \cup \Delta_2 \rangle$ recognizes $L_1 \cup L_2$.

Cross-product construction

Direct intersection

Let $\mathcal{A}_i = \langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ be NFTA recognizing L_i, for $i = 1, 2$.
Then $\mathcal{A} = \langle Q_1 \times Q_2, \mathcal{F}, G_1 \times G_2, \Delta \rangle$ recognizes $L_1 \cap L_2$, where

$$\frac{f(q_1, \ldots, q_n) \to q \in \Delta_1 \quad f(q'_1, \ldots, q'_n) \to q' \in \Delta_2}{f(\langle q_1, q'_1 \rangle, \ldots, \langle q_n, q'_n \rangle) \to \langle q, q' \rangle \in \Delta}$$

Remarks:

- If $\mathcal{A}_1, \mathcal{A}_2$ are D(C)FTA, then so is $\mathcal{A}$.
- If $\mathcal{A}_1, \mathcal{A}_2$ are complete, replace $G_1 \times G_2$ with $(G_1 \times Q_2) \cup (Q_1 \times G_2)$ to recognize $L_1 \cup L_2$.

Tree languages and context-free languages

Front

Let t be a ground tree. Then $fr(t) \in \mathcal{F}_0^*$ denotes the word obtained from reading the leaves from left to right (in increasing lexicographical order of their positions).

Example: $t = f(a, g(b, a), c)$, $fr(t) = abac$

Leaf languages

- Let L be a recognizable tree language. Then $fr(L)$ is context-free.
- Let L be a context-free language that does not contain the empty word. Then there exists an NFTA $\mathcal{A}$ with $L = fr(\mathcal{L}(\mathcal{A}))$.

Proof (idea):

- Given a T-NFTA recognizing L, construct a CFG from it.
- L is generated by a CFG using productions of the form $A \to BC \mid a$ only. Replace $A \to BC$ by $A \to A_2$ and $A_2 \to BC$, construct a T-NFTA from the result.

Visibly pushdown automata

Visibly pushdown automaton

Let $\mathcal{A} = \langle Q, \Sigma, \Gamma, T, q_0 z_0, F \rangle$ be a pushdown automaton.
$\mathcal{A}$ is called *visibly pushdown* (VPA) if there exist $\Sigma_0, \Sigma_1, \Sigma_2$ such that

- $\Sigma = \Sigma_0 \uplus \Sigma_1 \uplus \Sigma_2$
- $T \subseteq \bigcup_{i=0}^{2} (Q \times \Gamma) \times \Sigma_i \times (Q \times \Gamma^i)$

Closure properties

Languages accepted by VPA are closed under boolean operations.

VPA and tree languages

Let $L \subseteq T(\mathcal{F})$ be a recognizable tree language.
Then L, seen as a word language of terms, is accepted by a VPA.

From TA to VPA

Let $\mathcal{A} = \langle Q, \mathcal{F}, I, \Delta \rangle$ be a T-NFTA accepting L.
For convenience, assume $I = \{q_0\}$ is a singleton (closure under union). We construct a single-state VPA $\mathcal{B} = \langle \Sigma, \Gamma, T, q_0 \rangle$ accepting by empty stack and recognizing the terms of L (can be converted into a normal VPA).

- $\Sigma_0 = \mathcal{F}_0 \cup \{\ )\ \}$, $\Sigma_1 = \mathcal{F} \setminus \mathcal{F}_0$, $\Sigma_2 = \{\ ,\ ,\ (\ \}$
- $\Gamma = Q \cup \{\ r_i \mid r \in \Delta,\ \ r = q(f) \to (q_1, \dots, q_n),\ \ n \geq 1,\ \ 0 \leq i \leq n \}$
- $T = \bigcup_{r \in \Delta} T_r$
  - for $r = q(a) \to \varepsilon$, we have $T_r = \{\ \langle q, a, \varepsilon \rangle\ \}$;
  - for $r = q(f) \to (q_1, \dots, q_n)$, $n \geq 1$, we have $T_r = \{\langle q, f, r_0 \rangle, \langle r_0,\ (\ , q_1 r_1 \rangle, \langle r_n,\ )\ , \varepsilon \rangle\}$ $\cup \{\ \langle r_i,\ ,\ , q_{i+1} r_{i+1} \rangle \mid 1 \leq i < n \}$

Idea: $q \xrightarrow{t}{}^{*}_{\mathcal{B}} \varepsilon$ iff $q(t) \to^{*}_{\mathcal{A}} t$

From TA to VPA: Example

Consider a T-NFTA $\langle Q, \mathcal{F}, I, \Delta \rangle$ accepting $L = \{\, f(g^i(a)) \mid i \geq 0 \,\}$:

- $Q = \{q_0, q_1, q_f\}$, $\mathcal{F} = \{f(2), g(1), a\}$, $I = \{q_f\}$;
- $\Delta := \{\alpha : q_0(a) \to \varepsilon, \quad \beta : q_1(g) \to q_0, \quad \gamma : q_1(g) \to q_1, \quad \delta : q_f(f) \to (q_1, q_1)\}$.

We construct the single-state VPA $\langle \Sigma, \Gamma, T, q_f \rangle$, where:

- $\Sigma_0 = \{a, \;)\; \}$, $\Sigma_1 = \{f, g\}$, $\Sigma_2 = \{\; , \; , \; (\; \}$;
- $\Gamma = Q \cup \{\beta_0, \beta_1, \gamma_0, \gamma_1, \delta_0, \delta_1, \delta_2\}$;
- $T_\alpha = \{\langle q_0, a, \varepsilon \rangle\}$;
- $T_\beta = \{\langle q_1, g, \beta_0 \rangle, \langle \beta_0, \; (\; , q_0 \beta_1 \rangle, \langle \beta_1, \;) \; \varepsilon \rangle\}$;
- $T_\gamma = \{\langle q_1, g, \gamma_0 \rangle, \langle \gamma_0, \; (\; , q_1 \gamma_1 \rangle, \langle \gamma_1, \;) \; \varepsilon \rangle\}$;
- $T_\delta = \{\langle q_f, f, \delta_0 \rangle, \langle \delta_0, \; (\; , q_1 \delta_1 \rangle, \langle \delta_1, \; , \; , q_1 \delta_2 \rangle, \langle \delta_2, \;) \; \varepsilon \rangle\}$.

Run on $f(g(a), g(g(a)))$:

$$q_f \xrightarrow{f} \delta_0 \xrightarrow{(} q_1\delta_1 \xrightarrow{g} \beta_0\delta_1 \xrightarrow{(} q_0\beta_1\delta_1 \xrightarrow{a} \beta_1\delta_1 \xrightarrow{)} \delta_1 \xrightarrow{,} q_1\delta_2 \xrightarrow{g} \gamma_0\delta_2 \xrightarrow{(} q_1\gamma_1\delta_2$$

$$\xrightarrow{g} \beta_0\gamma_1\delta_2 \xrightarrow{(} q_0\beta_1\gamma_1\delta_2 \xrightarrow{a} \beta_1\gamma_1\delta_2 \xrightarrow{)} \gamma_1\delta_2 \xrightarrow{)} \delta_2 \xrightarrow{)} \varepsilon$$

Tree homomorphism

Definition

Let $\mathcal{X}_n := \{x_1, \ldots, x_n\}$ and $\mathcal{F}, \mathcal{F}'$ ranked alphabets.
A *tree homomorphism* is a mapping $h : \mathcal{F} \to T(\mathcal{F}', \mathcal{X})$,
with $h(f) \in T(\mathcal{F}, \mathcal{X}_n)$ if $f \in \mathcal{F}_n$.

Extension of h to trees $(T(\mathcal{F}) \to T(\mathcal{F}'))$:

- $h(f(t_1, \ldots, t_n)) = h(f)\{x_1 \leftarrow h(t_1), \ldots, x_n \leftarrow h(t_n)\}$

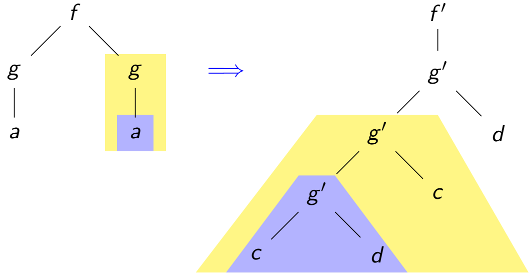
Intuition:

- $h(f)$ "explodes" f-positions into trees
- reorders/copies/deletes subtrees.

Examples

Example

- $\mathcal{F} = \{f(2), g(1), a\}$, $\mathcal{F}' = \{f'(1), g'(2), c, d\}$
- $h(f) = f'(g'(x_2, d))$, $h(g) = g'(x_1, c)$, $h(a) = g'(c, d)$



Example (ternary to binary tree)

- $\mathcal{F} = \{f(3), a, b\}$, $\mathcal{F}' = \{g(2), a, b\}$
- $h_{32}(f) = g(x_1, g(x_2, x_3))$, $h_{32}(a) = a$, $h_{32}(b) = b$

Properties of homomorphisms

A homomorphism h is

- *linear* if $h(f)$ linear for all f;
- *non-erasing* if $\mathcal{H}(h(f)) > 0$ for all f;
- *flat* if $\mathcal{H}(h(f)) = 1$ for all f;
- *complete* if $f \in \mathcal{F}_n$ implies that $h(f)$ contains all of $\mathcal{X}_n$;
- *permuting* if h is complete, linear, and flat;
- *alphabetic* if $h(f)$ has the form $g(x_1, \ldots, x_n)$ for all f.

Example: h_{32} is linear, non-erasing, and complete.

Non-linear homomorphisms do not preserve recognizability

- Example: $h(f) = f'(x_1, x_1)$, $h(g) = g(x_1)$, $h(a) = a$
- $L = \{\, f(g^i(a)) \mid i \geq 0 \,\}$ (recognizable)
- $h(L) = \{\, f'(g^i(a), g^i(a)) \mid i \geq 0 \,\}$ (not recognizable)

Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

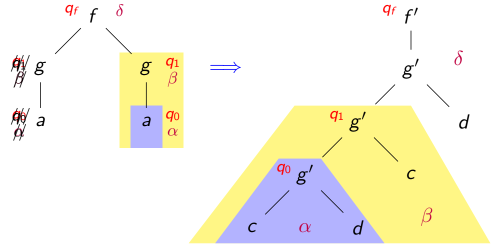
Let $L \subseteq T(\mathcal{F})$ be recognizable and $h : \mathcal{F} \to \mathcal{F}'$ a linear tree homomorphism. Then $h(L)$ is recognizable.

Illustrating example:

- $\mathcal{F} = \{f(2), g(1), a\}$, $\mathcal{F}' = \{f'(1), g'(2), c, d\}$
- $h(f) = f'(g'(x_2, d))$, $h(g) = g'(x_1, c)$, $h(a) = g'(c, d)$
- $L = \{\, f(g^i(a), g^k(a)) \mid i, k \geq 0 \,\}$
- $\mathcal{A} = \langle \{q_0, q_1, q_f\}, \mathcal{F}, \{q_f\}, \Delta \rangle$ recognizes L with
  $\Delta := \{\alpha : a \to q_0, \quad \beta : g(q_0) \to q_1, \quad \gamma : g(q_1) \to q_1, \quad \delta : f(q_1, q_1) \to q_f\}$

Run on $\mathcal{A}$

Rules used to produce states

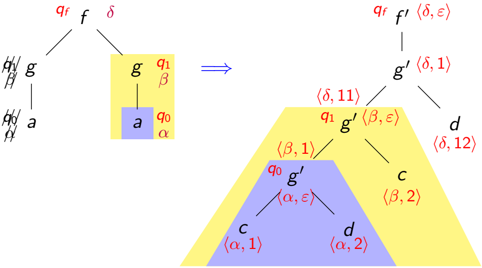


Construct automaton for $h(L)$ preserving state labels from $\mathcal{A}$
$+$
Guess the rules.

Automaton construction for $h(L)$

Given a reduced NFTA $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ for L,
construct NFTA $\mathcal{A}' = \langle Q', \mathcal{F}', G, \Delta' \rangle$ for $h(L)$.

- $Q' := Q \cup \{ \langle r, p \rangle \mid r \in \Delta, \exists f \in \mathcal{F} : r = f(\dots) \to \dots, p \in Pos_{h(f)} \}$;
- Δ' contains, for each transition $r : f(s_1, \dots, s_n) \to s$ in Δ and $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \dots, \langle r, pk \rangle) \to \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'_k$
  - $s_i \to \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \to s$



Correctness

To prove: $\mathcal{A}'$ accepts $h(L)$.

- $h(L) \subseteq \mathcal{L}(\mathcal{A}')$:
  For $t \in T(\mathcal{F})$, prove that $t \to_{\mathcal{A}}^{*} q$ implies $h(t) \to_{\mathcal{A}'}^{*} q$,
  by structural induction over t.

- $h(L) \supseteq \mathcal{L}(\mathcal{A}')$:
  For $t' \in T(\mathcal{F}')$, prove that if $t' \to_{\mathcal{A}'}^{*} q \in Q$,
  then there exists $t \in T(\mathcal{F}) \cap h^{-1}(t')$ with $t \to_{\mathcal{A}}^{*} q$,
  by induction on number of states (of Q) in the computation $t' \to_{\mathcal{A}'}^{*} q$.

Inverse tree homomorphisms

Theorem: Inverse homomorphisms preserve recognizability

Let $L \subseteq T(\mathcal{F}')$ be recognizable and $h : \mathcal{F} \to \mathcal{F}'$ a tree homomorphism (not necessarily linear). Then $h^{-1}(L)$ is recognizable.

Given an NFTA $\mathcal{A}' = \langle Q, \mathcal{F}', G, \Delta' \rangle$ for L, construct NFTA $\mathcal{A} = \langle Q \uplus \{!\}, \mathcal{F}, G, \Delta \rangle$ for $h^{-1}(L)$.

For all $n \geq 0$ and $f \in \mathcal{F}_n$, and $p_1, \ldots, p_n \in Q$,

- add $f(!, \ldots, !) \to\ !$ to Δ;
- if $h(f)\{x_1 \leftarrow p_1, \ldots, x_n \leftarrow p_n\} \to^*_{\mathcal{A}'} q$, add $f(q_1, \ldots, q_n) \to q$ to Δ, with:

$$q_i = \begin{cases} p_i & \text{if } x_i \text{ appears in } h(f) \\ ! & \text{otherwise} \end{cases}$$

Proof: Show $t \to^*_{\mathcal{A}} q$ iff $h(t) \to^*_{\mathcal{A}'} q$, for all $t \in T(\mathcal{F})$.

Intersection problem

Theorem

The following problem is EXPTIME-complete:
Given tree automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, is $\mathcal{L}(\mathcal{A}_1) \cap \cdots \cap \mathcal{L}(\mathcal{A}_n) \neq \emptyset$?

Proof (sketch):

- Hardness: Simulate an linear-space ATM $\mathcal{M}$ with input of length n.
  If $\mathcal{M}$ accepts the input, there is an accepting run.
  Encode the run of $\mathcal{M}$ as a tree.
  Construct $\mathcal{A}_i$, for $i = 1, \ldots, n$, to check:
  1. if $\mathcal{M}$ starts with the correct configuration;
  2. if all configurations in the run are of length n;
  3. if all final configurations are accepting;
  4. if the part of the configurations around the i-th symbol are coherent.
- Membership: Compute the productive tuples of states in $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n$.

Detailed proof: Veanes, 1997

Congruences on trees

Definition: Congruence

Let $\equiv$ be an equivalence relation on $T(\mathcal{F})$.

- $\equiv$ is called a *congruence*
  if for any $n \geq 0$ and $f \in \mathcal{F}_n$, $u_1 \equiv v_1, \ldots, u_n \equiv v_n$ we have
  $$f(u_1, \ldots, u_n) \equiv f(v_1, \ldots, v_n)$$
- $\equiv$ *saturates* L if $u \equiv v$ implies $u \in L \iff v \in L$.

For $L \subseteq T(\mathcal{F})$, write $u \equiv_L v$ if
$$\forall C \in \mathcal{C}(\mathcal{F}) : C[u] \in L \Leftrightarrow C[v] \in L$$

Myhill-Nerode Theorem for trees

The following are equivalent:

1. $L \subseteq T(\mathcal{F})$ is recognizable.
2. L is saturated by some congruence of finite index.
3. $\equiv_L$ is of finite index.

Myhill-Nerode Theorem

Application:

Consider $L = \{ f(g^i(a), g^i(a)) \mid i \geq 0 \}$.
For any pair $i \neq k$, consider $C = f(x, g^i(a))$.
Then $C[g^i(a)] \in L$ but $C[g^k(a)] \notin L \Rightarrow g^i(a) \not\equiv_L g^k(a)$
Therefore $\equiv_L$ is not of finite index, and L is not recognizable.

Proof of the theorem (sketch):

- $1 \to 2$: Let $\mathcal{A}$ be DCFTA and let $u \equiv v$ iff $u \to^*_{\mathcal{A}} q \;{}^*_{\mathcal{A}}\!\leftarrow v$.
  Then $\equiv$ is of finite index and saturates L.
- $2 \to 3$: Let $\equiv$ be a saturating congruence, $u \equiv v$ implies $u \equiv_L v$ (prove $u \equiv v$ implies $C[u] \equiv C[v]$ for all C, by recurrence over height of position of x in C).
- $3 \to 1$: Let $\mathcal{A} = \langle T(\mathcal{F})/\equiv_L, \mathcal{F}, L/\equiv_L, \Delta \rangle$, with
  $$f([u_1], \ldots, [u_n]) \to [f(u_1, \ldots, u_n)]$$
  for all $n \geq 0$, $f \in \mathcal{F}_n$, $u_1, \ldots, u_n \in T(\mathcal{F})$,
  where $[u]$ is the equivalence class of $u \in T(\mathcal{F})$;

Remark: This can be shown to be the canonical minimal DCFTA.

Path languages

Path languages

Let $t \in T(\mathcal{F})$. The *path language* $\pi(t)$ is defined as follows:

- if $t = a \in \mathcal{F}_0$, then $\pi(t) = \{a\}$;
- if $t = f(t_1, \ldots, t_n)$, for $f \in \mathcal{F}_n$, then $\pi(t) = \{\, fiw \mid w \in \pi(t_i) \,\}$.

We write $\pi(L) = \bigcup\{\, \pi(t) \mid t \in L \,\}$ for $L \subseteq T(\mathcal{F})$.

Example: $L = \{f(a, b), f(b, a)\}$, $\pi(L) = \{f1a, f2b, f1b, f2a\}$.

Path closure

Let $L \subseteq T(\mathcal{F})$ be a tree language.

- The *path closure* of L is $pc(L) = \{\, t \mid \pi(t) \subseteq \pi(L) \,\} \supseteq L$.
- L is called *path-closed* if $L = pc(L)$.

Example: $pc(L) = \{f(a, a), f(a, b), f(b, a), f(b, b)\}$, so L is not path-closed.

Path closure and T-NFTA

Lemma

Let $L \subseteq T(\mathcal{F})$ be a recognizable tree language. Then:

- $\pi(L)$ is a recognizable word language.
- $pc(L)$ is a recognizable tree language.

Proof: Let $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ be a reduced T-NFTA for L.

- Construct a finite (word) automaton out of $\mathcal{A}$.
  (Easy, but does require $\mathcal{A}$ to be reduced!)
- Construct $\mathcal{A}' = \langle Q, \mathcal{F}, G, \Delta' \rangle$ for $pc(L)$ as follows:
  for all $a \in \mathcal{F}_0$:
  $$q(a) \rightarrow_\Delta \varepsilon \quad \rightarrow \quad q(a) \rightarrow_{\Delta'} \varepsilon$$
  for all $n \geq 1$, $f \in \mathcal{F}_n$:
  $$\forall i : q(f) \rightarrow_\Delta (q_{i,1}, \ldots, q_{n,1}) \quad \rightarrow \quad q(f) \rightarrow_{\Delta'} (q_{1,1}, \ldots, q_{n,n})$$
  Let $L_q = \mathcal{L}(\langle Q, \mathcal{F}, \{q\}, \Delta \rangle)$ and $L'_q = \mathcal{L}(\langle Q, \mathcal{F}, \{q\}, \Delta' \rangle)$.
  Prove $t \in L'_q \Leftrightarrow \pi(t) \subseteq \pi(L_q)$ for all $q \in Q$, $t \in T(\mathcal{F})$ by induction.

Path closure and T-NFTA

Corollary

It is decidable whether a recognizable tree language is path-closed.

Theorem

Let $L \subseteq T(\mathcal{F})$ be a recognizable tree language.
L is path-closed iff it is recognized by a T-DFTA.

Proof:

- "$\rightarrow$":
  Let $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ be a reduced T-NFTA for L.
  Construct a T-DFTA $\mathcal{A}' = \langle 2^Q, \mathcal{F}, G, \Delta' \rangle$ as follows:
  for all $a \in \mathcal{F}_0$, $S(a) \rightarrow_{\Delta'} \varepsilon$ if $\exists q \in S, q(a) \rightarrow_{\Delta} \varepsilon$;
  for all $n \geq 1, f \in \mathcal{F}_n$, $S(f) \rightarrow_{\Delta'} (S_1, \ldots, S_n)$
  where $S_i = \{\, q_i \mid \exists q \in S, q(f) \rightarrow_{\Delta} (q_1, \ldots, q_n) \,\}$.
- "$\leftarrow$":
  Let $\mathcal{A}$ be a complete T-DFTA for L, define L_q as before.
  Prove that $\pi(t) \subseteq \pi(L_q)$ implies $t \in L_q$, for all $q \in Q, t \in T(\mathcal{F})$.

Logic over trees

Alternative specification for sets of trees

E.g., to describe valid HTML documents:

- A `p` tag may only appear inside a `body` tag.
- A `dl` tag must contain pairs of `dt` and `dd` tags.

Roadmap

- We shall define a logic that defines such properties of trees.
- The sets of trees definable in that language will be recognizable.

Recall: First-/second-order logic

First-order logic (FO)

Let $\sigma = ((R_i)_{1\leq i\leq n})$ be a relation signature and $\mathcal{X}_1 = \{x_1, x_2, \ldots\}$ a set of variables. The first-order formulas $FO(\sigma)$ are:

$$R_i(x_{j_1}, \ldots, x_{j_i}) \mid x = x' \mid \neg\phi \mid \phi \wedge \phi' \mid \exists x.\phi$$

Second-order logic: allow quantifying over relations
Monadic: only quantify over sets

Monadic second-order logic (MSO)

Let σ as before and $\mathcal{X}_1 = \{x_1, x_2, \ldots\}$, $\mathcal{X}_2 = \{X_1, X_2, \ldots\}$ sets of first-/second-order variables. The set of $MSO(\sigma)$ formulae are:

$$R_i(x_{j_1}, \ldots, x_{j_i}) \mid x = x' \mid x \in X \mid \neg\phi \mid \phi \wedge \phi' \mid \exists x.\phi \mid \exists X.\phi$$

Weak second-order: only quantify over *finite* sets

WSkS (weak MSO over with k successors)

$$\text{WS}k\text{S} \;=\; \text{MSO}(<_1, \ldots, <_k)$$

Semantics of MSO

Definition

Let $\mathfrak{M}$ a domain, σ a signature, ν a valuation with

- $\nu(x) \in \mathfrak{M}$ for $x \in \mathcal{X}_1$
- $\nu(X) \subseteq \mathfrak{M}$ for $X \in \mathcal{X}_2$

$$
\begin{array}{lll}
\mathfrak{M}, \sigma, \nu \models R_i(x_{j_1}, \ldots, x_{j_i}) & \text{if} & (\nu(x_{j_1}), \ldots, \nu(x_{j_i})) \in R_i \\
\mathfrak{M}, \sigma, \nu \models x = x' & \text{if} & \nu(x) = \nu(x') \\
\mathfrak{M}, \sigma, \nu \models x \in X & \text{if} & \nu(x) \in \nu(X) \\
\mathfrak{M}, \sigma, \nu \models \neg\phi & \text{if} & \mathfrak{M}, \sigma, \nu \not\models \phi \\
\mathfrak{M}, \sigma, \nu \models \phi \wedge \phi' & \text{if} & \mathfrak{M}, \sigma, \nu \models \phi \ \wedge \ \mathfrak{M}, \sigma, \nu \models \phi' \\
\mathfrak{M}, \sigma, \nu \models \exists x.\phi & \text{if} & \exists m \in \mathfrak{M}.\ \ \mathfrak{M}, \sigma, \nu[x \mapsto m] \models \phi \\
\mathfrak{M}, \sigma, \nu \models \exists X.\phi & \text{if} & \exists M \subseteq \mathfrak{M}.\ \ \mathfrak{M}, \sigma, \nu[X \mapsto M] \models \phi
\end{array}
$$

We omit $\mathfrak{M}, \sigma$ when clear from context.

Recall: Common abbreviations

- $\forall x$, $\forall X$, $\vee$, etc can be expressed in the usual way.
- $X \subseteq Y$:
$$\forall x.(x \in X \to x \in Y)$$
- $Z = X \cup Y$:
$$\forall x.(x \in Z \leftrightarrow x \in X \vee x \in Y)$$
- $Partition(X, X_1, \ldots, X_m)$:
$$\left(\forall x.\left(x \in X \leftrightarrow \bigvee_{i=1}^{m} x \in X_i\right)\right) \wedge \left(\bigwedge_{i=1}^{m} \bigwedge_{j \neq i} \forall x.(x \notin X_i \vee x \notin X_j)\right)$$
- Similarly, $X = \emptyset$, $X = \{x\}$, $X = Y$,...

WSkS and trees

Let $\mathfrak{M} = N^*$, we fix $<_i$ to be the relation $<_i = \{ \langle p, pip' \rangle \mid p, p' \in N^* \}$.
We define $< = \bigcup_{i=1}^{k} <_i$ and $\leq$ as usual, and ε for the minimal element.
We write xi to denote the least q s.t. $\nu(x) <_i q$.

Coding of a tree

Let $t \in T(\mathcal{F})$ and k the maximal arity in $\mathcal{F}$.
As a shorthand, define $S_{\mathcal{F}} := (S_f)_{f \in \mathcal{F}}$.
We note $C(t) := (S, S_{\mathcal{F}})$, where:

- $S = \bigcup_{f \in \mathcal{F}} S_f$;
- for all $f \in \mathcal{F}$, $S_f = \{ p \in Pos_t \mid t(p) = f \}$.

$(S, S_{\mathcal{F}})$ encodes a tree if $Tree(S, S_{\mathcal{F}})$ holds:

$$\begin{aligned} Tree(S, S_{\mathcal{F}}) \quad := \quad & S \neq \emptyset \wedge Partition(S, S_{\mathcal{F}}) \\ & \wedge \forall x. \forall y. (x \in S \wedge y < x) \rightarrow y \in S \\ & \wedge \ \bigwedge_{n=1}^{k} \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^{n} (x \in S_f \rightarrow xi \in S) \\ & \wedge \ \bigwedge_{n=1}^{k} \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=n+1}^{k} (x \in S_f \rightarrow xi \notin S) \end{aligned}$$

Semantics of WSkS on trees

Coded valuation

Let $\mathcal{F}' := \mathcal{F} \times 2^{\mathcal{X}_1 \cup \mathcal{X}_2}$. The arity of (f, τ) is n if $f \in \mathcal{F}_n$.
Let $t \in T(\mathcal{F})$ and ν a valuation. The tuple $\langle t, \nu \rangle$ is *coded* by a tree $t' \in T(\mathcal{F}')$, as follows, for all $p \in Pos$ and $t'(p) = \langle f, \tau \rangle$:

- if $x \in \mathcal{X}_1$ then $\tau(x) = 1$ iff $p = \nu(x)$;
- if $X \in \mathcal{X}_2$ then $\tau(X) = 1$ iff $p \in \nu(X)$.

A tree $t' \in T(\mathcal{F}')$ is *valid* ($t' \in T_v(\mathcal{F}')$) if it codes some $\langle t, \nu \rangle$.

Semantics of WSkS

Let ϕ be a formula of WSkS and $V \subseteq (\mathcal{X}_1 \cup \mathcal{X}_2) \uplus (\{S\} \cup S_{\mathcal{F}})$ its free variables.

$$\mathcal{L}(\phi) := \{ \langle t, \nu \rangle \in T_v(\mathcal{F}') \mid \nu[(S, S_{\mathcal{F}}) \mapsto C(t)] \models \phi \}$$

Examples

- Let $t = f(g(a), a)$.
  Left: $\langle t, \nu \rangle$ with $\nu(x) = \varepsilon$, $\nu(y) = 11$, and $\nu(Z) = \{\varepsilon, 11, 2\}$.
  Right: $\langle t, \nu' \rangle$ with $\nu'(x) = 1$

$$
\begin{array}{ccc}
 & \langle f, 101 \rangle & \\
 \swarrow & & \searrow \\
\langle g, 000 \rangle & & \langle a, 001 \rangle \\
| & & \\
\langle a, 011 \rangle & &
\end{array}
\qquad\qquad
\begin{array}{ccc}
 & \langle f, 0 \rangle & \\
 \swarrow & & \searrow \\
\langle g, 1 \rangle & & \langle a, 0 \rangle \\
| & & \\
\langle a, 0 \rangle & &
\end{array}
$$

- We have $C(t) = (S, S_f, S_g, S_a)$ with $S = \{\varepsilon, 1, 11, 2\}$,
  $S_f = \{\varepsilon\}$, $S_g = \{1\}$, $S_a = \{11, 2\}$.
- $\nu'[(S, S_{\mathcal{F}}) \mapsto C(t)] \models x \in S_g$, thus $\langle t, \nu' \rangle \in \mathcal{L}(x \in S_g)$
- $t \in \mathcal{L}(\exists x. x \in S_g)$

WSkS and recognizability

Theorem

A tree language $L \subseteq T(\mathcal{F})$ is recognizable
iff $L = \mathcal{L}(\phi)$ for some formula $\phi(S, S_{\mathcal{F}})$ of WSkS.

Proof: (sketch)

- DCFTA $\mathcal{A} \to$ WSkS: Construct formula ϕ that
  (i) verifies that the structure is a tree;
  (ii) guesses a computation of $\mathcal{A}$, i.e. partitioning of S onto states;
  (iii) verifies that the computation is locally correct;
  (iv) verifies that the root is labelled by an accepting state.
- WSkS $\phi \to$ NFTA $\mathcal{A}$: Proceed by recurrence on ϕ,
  show that all subformulae of ϕ are recognizable.

Example: DCFTA $\rightarrow$ WSkS

- Let $Q := \{q_0, q_1, q_f\}$, $\mathcal{F} = \{f(2), g(1), a\}$, $G := \{q_f\}$, and rules

$$a \rightarrow q_0 \quad g(q_0) \rightarrow q_1 \quad g(q_1) \rightarrow q_1 \quad f(q_1, q_1) \rightarrow q_f$$

(automate à compléter !)

- Corresponding formula:

$$\begin{array}{rcl} \phi & = & \mathit{Tree}(S, S_{\mathcal{F}}) \\ & & \wedge\ \exists Q_0, Q_1, Q_f.\mathit{Partition}(S, Q_0, Q_1, Q_f) \\ & & \qquad \wedge\ \forall x.(x \in S_a \rightarrow x \in Q_0) \\ & & \qquad \wedge\ \forall x.((x \in S_g \wedge x1 \in Q_0) \rightarrow x \in Q_1) \\ & & \qquad \wedge\ \forall x.((x \in S_g \wedge x1 \in Q_1) \rightarrow x \in Q_1) \\ & & \qquad \wedge\ \forall x.((x \in S_f \wedge x1 \in Q_1 \wedge x2 \in Q_1) \rightarrow x \in Q_f) \\ & & \qquad \wedge\ \cdots \\ & & \qquad \wedge\ \varepsilon \in Q_f \end{array}$$

Example: WSkS $\to$ NFTA

Consider $\mathcal{F} = \{f(2), g(1), a\}$.

- $\phi = x \in S_g$
  $\mathcal{A}_\phi = \langle \{q, q'\}, \mathcal{F} \times 2^{\{x\}}, \{q'\}, \Delta \rangle$ with transitions

  $$\begin{array}{lll} \langle a, 0 \rangle \to q & & \\ \langle g, 1 \rangle(q) \to q' & \langle g, 0 \rangle(q) \to q & \langle g, 0 \rangle(q') \to q' \\ \langle f, 0 \rangle(q, q) \to q & \langle f, 0 \rangle(q, q') \to q' & \langle f, 0 \rangle(q', q) \to q' \end{array}$$

  accepts $\mathcal{L}(x \in S_g)$ (scans for a single g-position with $\tau(x) = 1$).
- $\phi' = \exists x.\phi$
  Obtain $\mathcal{A}_{\phi'}$ from $\mathcal{A}_\phi$ by stripping $\tau(x)$:

  $\mathcal{A}_{\phi'} = \langle \{q, q'\}, \mathcal{F}, \{q'\}, \Delta \rangle$

  $$\begin{array}{lll} a \to q & & \\ g(q) \to q' & g(q) \to q & g(q') \to q' \\ f(q, q) \to q & f(q, q') \to q' & f(q', q) \to q' \end{array}$$

Unranked trees

We now consider *finite ordered unranked* trees.

- *ordered* : internal nodes have children $1 \ldots n$
- *unranked* : nodes may have an arbitrary number of children

Motivation: e.g., XML documents

- "A `html` tag contains an optional `head` and an obligatory `body`."
- "A `div` tag contains an unlimited number of `p`, `ol`, `ul`, ... tags."

Definition: Tree (recall)

A (finite, ordered) *tree* is a non-empty, finite, prefix-closed set $Pos \subseteq N^*$.

Hedge automata

Definition: (Bottom-up) hedge automaton

A *hedge automaton* (NHA) is a tuple $\mathcal{A} = \langle Q, \Sigma, G, \Delta \rangle$, where:

- Q is a finite set of *states*;
- Σ a finite alphabet;
- $G \subseteq Q$ are the *final states*;
- Δ is a finite set of rules of the form
$$a(R) \to q$$
for $a \in \Sigma$, $q \in Q$, and R a regular (word) language over Q.

Example: $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules

$$x(q_h^? q_b) \to q_x \quad h(\varepsilon) \to q_h \quad b(q_p^*) \to q_b \quad p(\varepsilon) \to q_p$$

This accepts trees of the form $x(h, b(p, \ldots, p))$ and $x(b(p, \ldots, p))$.

Semantics of hedge automata

Remark:

- The R in $a(R) \to q$ are called *horizontal languages*.
- They are (finitely) represented by regular expressions or finite automata.

Computation of NHA

Let $t \in T(\Sigma)$ be a tree. A *run* or *computation* of $\mathcal{A}$ on t is a tree $t' \in T(Q)$, i.e. for all $p \in Pos$:

- if $t(p) = a \in \Sigma$, $t'(p) = q \in Q$, and $Pos \cap pN = \{p1, \ldots, pn\}$, there exists $a(R) \to q \in \Delta$ such that $t'(p1) \cdots t'(pn) \in R$.

Acceptance condition: $t'(\varepsilon) \in G$

$L \subseteq T(\Sigma)$ is called *hedge-recognizable* if $L = \mathcal{L}(\mathcal{A})$ for some NHA $\mathcal{A}$.

Complete / normalized / deterministic HA

An NHA is . . .

- *complete* if for all $t \in T(\Sigma)$, $t \rightarrow^*_{\mathcal{A}} q$ for some q;
- *full* if for all $a \in \Sigma$, $q \in Q$, there is some $a(R) \rightarrow q$;
- *reduced* if $a(R_1) \rightarrow q, a(R_2) \rightarrow q \in \Delta$ implies $R_1 = R_2$;
- *deterministic* (DHA) if $a(R_1) \rightarrow q_1, a(R_2) \rightarrow q_2 \in \Delta$ implies $R_1 \cap R_2 = \emptyset$ or $q_1 = q_2$.

Any NHA has an equivalent complete / full / reduced / deterministic NHA.

- complete: add garbage state, as usual
- full: add rules $a(\emptyset) \rightarrow q$ where necessary
- reduced: replace $a(R_1) \rightarrow q$ and $a(R_2) \rightarrow q$ with $a(R_1 \cup R_2) \rightarrow q$ where necessary

Determinization

Determinization of NHA

Let $\mathcal{A} = \langle Q, \Sigma, G, \Delta \rangle$ be a complete, full, reduced NHA. The complete, full, reduced DHA $\mathcal{A}' = \langle 2^Q, \Sigma, G', \Delta' \rangle$ is equivalent to $\mathcal{A}$ where:

- $G' = \{ S \subseteq Q \mid S \cap G \neq \emptyset \}$;
- let $R_{a,q}$ denote the (unique) language s.t. $a(R_{a,q}) \to q \in \Delta$;
- $R'_{a,q} := R_{a,q}[\, q' \to (S \cup \{q'\}) \mid q' \in Q, S \subseteq Q \,]$
- for all $a \in \Sigma$, $S \subseteq Q$, we have $a(R_{a,S}) \to S \in \Delta'$;

$$R_{a,S} := \left(\bigcap_{q \in S} R'_{a,q} \right) \setminus \left(\bigcup_{q \notin S} R'_{a,q} \right)$$

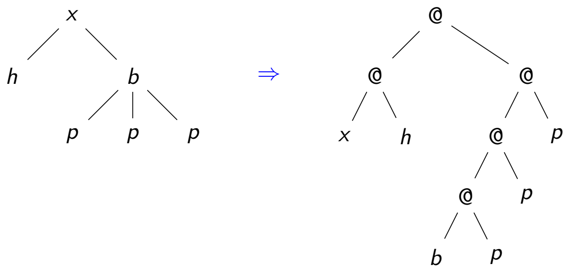
Encoding unranked trees

Bijective encoding of unranked into ranked trees

- Let Σ an alphabet; $\mathcal{F}_\Sigma := \{@(2)\} \cup \{\, a(0) \mid a \in \Sigma \,\}$.
- Define the coding $C_@(t) \in T(\mathcal{F}_\Sigma)$ of $t \in T(\Sigma)$ as

$$C_@(a(t_1, \ldots, t_n)) = \underbrace{@(@(\ldots(@(}_{n}a, C_@(t_1)), C_@(t_2)), \ldots), C_@(t_n))$$

Example:



Recognizing encoded trees

Theorem

A language $L \subseteq T(\Sigma)$ is hedge-recognizable iff $C_{@}(L)$ is recognizable.

- NHA $\rightarrow$ NFTA:
  Let $\mathcal{A} = \langle Q, \Sigma, G, \Delta \rangle$ an NHA; $\Delta = \{a_1(R_1) \rightarrow q_1, \ldots, a_n(R_n) \rightarrow q_n\}$;
  R_i represented by det.compl. FA $\mathcal{A}_i = \langle S_i, Q, s_0^{(i)}, F_i, \delta_i \rangle$.

  Construct NFTA $\mathcal{A}' = \langle Q', \mathcal{F}_\Sigma, G, \Delta' \rangle$, where:
  - $Q' = Q \cup \biguplus_{i=1}^{n} S_i$
  - $\Delta' = \bigcup_{i=1}^{n} (\Delta_1^i \cup \Delta_2^i \cup \Delta_3^i)$

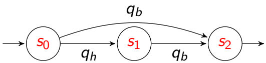
$$
\begin{array}{lcl}
\Delta_1^i & = & \{ a_i \rightarrow s_0^{(i)} \} \\
\Delta_2^i & = & \{ @(s, q) \rightarrow \delta_i(s, q) \mid s \in S_i, q \in Q \} \\
\Delta_3^i & = & \{ s_f \rightarrow q_i \mid s_f \in F_i \}
\end{array}
$$

Example: NHA $\to$ NFTA

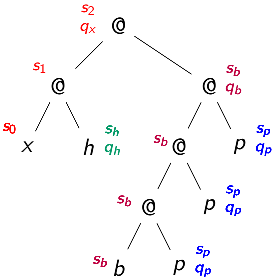
- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules

$$x(q_h^? q_b) \to q_x \quad h(\varepsilon) \to q_h \quad b(q_p^*) \to q_b \quad p(\varepsilon) \to q_p$$

- Automaton for first rule:



- Single-state automata with s_h, s_b, s_p for the other rules



Recognizing encoded trees

Theorem

A language $L \subseteq T(\Sigma)$ is hedge-recognizable iff $C_{@}(L)$ is recognizable.

- NFTA $\rightarrow$ NHA:
  Let $\mathcal{A} = \langle Q, \mathcal{F}_\Sigma, G, \Delta \rangle$ an NFTA (without ε-moves).

  Define $\Delta_R := \{ \langle q_0, q_1, q_2 \rangle \mid @(q_0, q_1) \rightarrow_\Delta q_2 \}$
  and $Out := G \cup \{ q \mid \exists q', q'' : @(q', q) \rightarrow_\Delta q'' \}$.
  For $q \in Q, q' \in Out$, let $A_{q,q'} := \langle Q, Q, q, \{q'\}, \Delta_R \rangle$ a word automaton.

  Construct NHA $\mathcal{A}' := \langle Q, \Sigma, G, \Delta' \rangle$, where
  $$\Delta' = \{ a(\mathcal{L}(\mathcal{A}_{q,q'})) \rightarrow q' \mid a \rightarrow_\Delta q, q' \in Out \}$$

Corollary

Hedge-recognizable languages are closed under boolean operations.

Unranked trees and logic

UTL = weak MSO(*child*,*next*) interpreted over $\mathfrak{M} = N^*$, where

- $child(x, y)$ iff $y = xi$ for some $i \in N$
- $next(x, y)$ iff $\exists z, i : x = zi \wedge y = z(i + 1)$

Further predicates can be defined from this:

- $right(x, y)$ = "y is a right sibling of x"
- $desc(x, y)$ = "y is a descendant of x" = "$x \leq y$"

Notions like $\mathcal{L}(\phi)$ are defined in analogy with WSkS.

Theorem: UTL = NHA

A language $L \subseteq T(\Sigma)$ is hedge-recognizable
iff $L = \mathcal{L}(\phi)$ for some formula $\phi(S, S_\Sigma)$ of UTL.

UTL = NHA: Proof sketch

- UTL $\rightarrow$ NHA:
  Let ϕ be an UTL formula. Define ϕ' of WS2S s.t. $\mathcal{L}(\phi') = C_{@}(\mathcal{L}(\phi))$.

  Define *leftmost*(x, y) as

  $$\begin{aligned} \forall X: \quad & (x \in X \wedge \forall z, z' : (z \in X \wedge z' = z1 \rightarrow z' \in X) \\ & \qquad \wedge \forall z : (z \in X \rightarrow z = x \vee (\exists z' : z' \in X \wedge z = z'1))) \\ & \quad \rightarrow (y \in X \wedge \forall z : z \in X \rightarrow z \leq y) \end{aligned}$$

  ("y is the maximal position in $x1^*$")

  Then *child* and *next* can be translated as follows:

  $$\begin{aligned} child(x, y) & := \exists z : leftmost(z, x) \wedge leftmost(z2, y) \\ next(x, y) & := \exists z : leftmost(z12, x) \wedge leftmost(z2, y) \end{aligned}$$

UTL = NHA: Proof sketch

- NHA $\to$ UTL:
  Let $\mathcal{A}$ be a complete, full, normalized, deterministic NHA.

  Construct formula $\phi(S, S_\Sigma)$ of UTL that
  (i) verifies that the structure is a tree;
  (ii) guesses a computation of $\mathcal{A}$, i.e. partitioning of S onto states;
  (iii) verifies that the computation is locally correct;
  (iv) verifies that the root is labelled by an accepting state.

  The major difference with the NFTA $\to$ WSkS construction is (iii):
  (iii): whenever the computation puts q on an a-labelled position p, guess a run of the automaton for $R_{a,q}$ over p and its children

Tuples of trees

Let $t_1, t_2 \in T(\mathcal{F})$ ranked trees. Add a fresh symbol $-$ to $\mathcal{F}_0$ and let

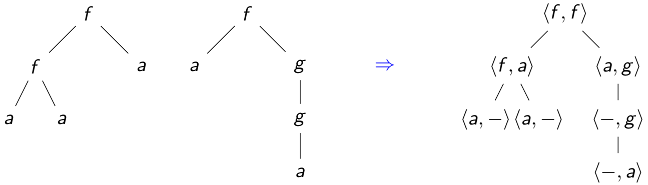
$$\mathcal{F}' := \{ \langle f, g\rangle(k) \mid f \in \mathcal{F}_m, g \in \mathcal{F}_n, k = \max\{m, n\} \}.$$

$\langle t_1, t_2\rangle$ denotes the ranked tree $t \in T(\mathcal{F}')$ as follows:

- $Pos_t = Pos_{t_1} \cup Pos_{t_2}$
- for all $p \in Pos_t$,

$$t(p) = \begin{cases} \langle f, g\rangle & \text{if } t \in Pos_{t_1} \cap Pos_{t_2}, t_1(p) = f, t_2(p) = g \\ \langle f, -\rangle & \text{if } t \in Pos_{t_1} \setminus Pos_{t_2}, t_1(p) = f \\ \langle -, g\rangle & \text{if } t \in Pos_{t_2} \setminus Pos_{t_1}, t_2(p) = g \end{cases}$$

Example:



Tree relations

We consider (binary) relations $R \subseteq T(\mathcal{F})^2$.

- Let $\mathfrak{R}_2$ be the class of recognizable relations (= recognizable languages over $\mathcal{F}'$).
- Let $\mathfrak{X}_2$ be the class of *finite unions of cross products* $R \in \mathfrak{X}_2$ iff $R = \bigcup_{i=1}^{n} \left(L_1^{(i)} \times L_2^{(i)} \right)$, for some $n \geq 0$ and $L_1^{(i)}, L_2^{(i)}$ recognizable for all i
- Let $\mathfrak{T}_2$ be the class of relations recognizable by GTT.

Definition: Ground Tree Transducer

A *ground tree transducer* (GTT) is pair $\mathcal{G} = \langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ of bottom-up NFTA over $\mathcal{F}$. (The states of $\mathcal{A}_1$ and $\mathcal{A}_2$ may overlap.)
The relation accepted by $\mathcal{G}$ is

$$\begin{aligned}\{ \langle t, u \rangle \quad | \quad & \exists n \geq 0,\ C \in \mathcal{C}^n(\mathcal{F}), \\ & \qquad t_1, \ldots, t_n \in T(\mathcal{F}),\ u_1, \ldots, u_n \in T(\mathcal{F}),\ q_1, \ldots, q_n : \\ & t = C[t_1, \ldots, t_n] \wedge u = C[u_1, \ldots, u_n] \\ & \qquad \wedge \forall i : t_i \rightarrow^*_{\mathcal{A}_1} q_i \ {}_{\mathcal{A}_2}^{*}\!\!\leftarrow u_i \}\end{aligned}$$

Relations between $\mathfrak{R}_2, \mathfrak{X}_2, \mathfrak{T}_2$

Propositions

1. $\mathfrak{R}_2 \not\subseteq \mathfrak{X}_2$ and $\mathfrak{T}_2 \not\subseteq \mathfrak{X}_2$
2. $\mathfrak{R}_2 \not\subseteq \mathfrak{T}_2$ and $\mathfrak{X}_2 \not\subseteq \mathfrak{T}_2$
3. $\mathfrak{X}_2 \subseteq \mathfrak{R}_2$
4. $\mathfrak{T}_2 \subseteq \mathfrak{R}_2$
5. $\mathfrak{X}_2 \cup \mathfrak{T}_2 \subsetneq \mathfrak{R}_2$

Proofs:

1. $\{\, \langle t, t \rangle \mid t \in T(\mathcal{F}) \,\}$ is in $\mathfrak{T}_2 \cap \mathfrak{R}_2$ but not $\mathfrak{X}_2$
2. $\emptyset$ is in $\mathfrak{X}_2 \cap \mathfrak{R}_2$ but not $\mathfrak{T}_2$
3. see next slides
4. see next slides
5. see next slides

Proof of $\mathfrak{X}_2 \subseteq \mathfrak{R}_2$

3. Let $A_i = \langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ (for $i = 1, 2$) be NFTA and let $R = \mathcal{L}(\mathcal{A}_1) \times \mathcal{L}(\mathcal{A}_2) \in \mathfrak{X}_2$.

   Construct NFTA $\mathcal{A} = \langle Q, \mathcal{F}', G_1 \times G_2, \Delta \rangle$ with $\mathcal{L}(\mathcal{A}) = R$:

   - $Q = (Q_1 \cup \{-\}) \times (Q_2 \cup \{-\})$
   - for every $f \in \mathcal{F}_m$, $g \in \mathcal{F}_n$, $m \geq n$, $\neg(f = g = -)$ Δ contains
     - $\langle f, g \rangle(\langle q_1, q_1' \rangle, \ldots, \langle q_n, q_n' \rangle, \langle q_{n+1}, - \rangle, \ldots, \langle q_m, - \rangle) \rightarrow \langle q, q' \rangle$ if $f(q_1, \ldots, q_m) \rightarrow q \in \Delta_1$ and $g(q_1', \ldots, q_n') \rightarrow q' \in \Delta_2$
     - $\langle g, f \rangle(\langle q_1, q_1' \rangle, \ldots, \langle q_n, q_n' \rangle, \langle -, q_{n+1}' \rangle, \ldots, \langle -, q_m \rangle) \rightarrow \langle q, q' \rangle$ if $f(q_1', \ldots, q_m') \rightarrow q \in \Delta_2$ and $g(q_1, \ldots, q_n) \rightarrow q' \in \Delta_1$

   (reminder: we assume that $-$ is a fresh symbol in $\mathcal{F}_0$)

Intuition: Modified cross-product construction.

Proof of $\mathfrak{T}_2 \subseteq \mathfrak{R}_2$

4. Let $\mathcal{G} = \langle \mathcal{A}_1, \mathcal{A}_2 \rangle$, $A_i = \langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ (for $i = 1, 2$).
   We construct NFTA $\mathcal{A}' = \langle Q', \mathcal{F}', \{q_f\}, \Delta' \rangle$ with $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{G})$.

   Construct NFTA $\mathcal{A} = \langle Q, \mathcal{F}', G, \Delta \rangle$ from $\mathcal{A}_1, \mathcal{A}_2$ as in previous proof.
   Then:

   - $Q' = Q \uplus \{q_f\}$
   - $\Delta' = \Delta \cup \Delta_1 \cup \Delta_2$
     $\Delta_1 = \{ \langle q, q \rangle \to q_f \mid q \in Q_1 \cap Q_2 \}$
     $\Delta_2 = \{ \langle f, f \rangle (q_f, \ldots, q_f) \to q_f \mid f \in \mathcal{F}_n, \; f \neq - \}$

   Intuition:
   Δ reads pairs of trees from $\mathcal{A}_1, \mathcal{A}_2$;
   Δ_1 allows to plug pairs of subtrees into some context C;
   Δ_2 reads the remaining context C.

Proof of $\mathfrak{X}_2 \cup \mathfrak{T}_2 \subsetneq \mathfrak{R}_2$

5. Let $\mathcal{F} = \{f(1), g(1), a\}$.
   Let $R = \{\, \langle t_1, t_2 \rangle \mid \exists C \in \mathcal{C}(\mathcal{F}), t \in T(\mathcal{F}) : t_1 = C[t] \wedge t_2 = C[f(t)] \,\}$.
   - $R \notin \mathfrak{X}_2$:
     By pigeonhole principle using $\langle f^i(a), f^{i+1}(a) \rangle$, $i \geq 0$.
   - $R \notin \mathfrak{T}_2$:
     Suppose that R is accepted by GTT $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ with n states in common.
     For all $i \geq 0$, let q_i such that $g^i(a) \rightarrow^*_{\mathcal{A}_1} q_i$ and $f(g^i(a)) \rightarrow^*_{\mathcal{A}_2} q_i$.
     Contradiction follows from pigeon-hole principle.
   - $R \in \mathfrak{R}_2$:
     Let $\mathcal{A} = \langle \{q_a, q_f, q_g, q\}, \mathcal{F}', \{q\}, \Delta \rangle$ with:

     $$\langle -, a \rangle \to q_a \quad \langle x, y \rangle(q_x) \to q_y \quad q_f \to q \quad \langle x, x \rangle(q) \to q$$

     for $x, y \in \{f, g, a\}$

Closure properties

Boolean closure

$\mathfrak{X}_2$ and $\mathfrak{R}_2$ are closed under boolean operations.

Transitive closure

If $R \in \mathfrak{T}_2$, then $R^* \in \mathfrak{T}_2$.

Proof: Let $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$ with states Q_1, Q_2 a GTT accepting R.
We construct $\langle \mathcal{B}_1, \mathcal{B}_2 \rangle$ accepting R^* by adding transitions to $\mathcal{A}_1$ and $\mathcal{A}_2$ using the following saturation rule:

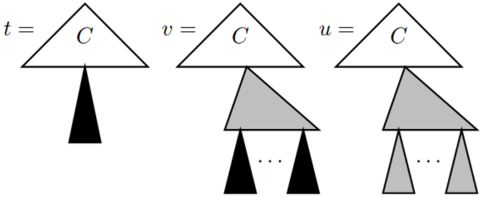
- For $i \neq j$ and all $q \in Q_1 \cap Q_2$, $q' \in Q_j$, if there exists a tree t s.t.

$$t \rightarrow^*_{\mathcal{B}_i} q \quad \text{and} \quad t \rightarrow^*_{\mathcal{B}_j} q'$$

  then add $q \rightarrow q'$ to $\mathcal{B}_j$.

Transitive closure: Intuition

Suppose that $\langle t, v\rangle, \langle v, u\rangle \in R$. The interesting case is illustrated below:

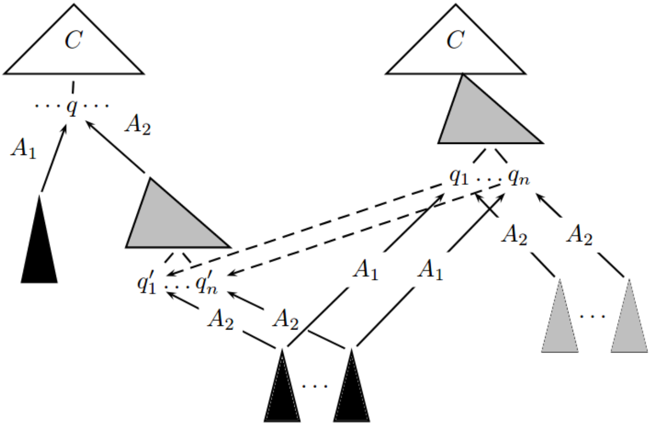


Suppose that $\langle t, v\rangle$ differ in a position p
and $\langle v, u\rangle$ in positions $pp_1, \ldots, pp_n$.

Then in $\mathcal{A}_2$ we want the subtrees of u at $pp_1, \ldots, pp_n$ to be substitutable for the corresponding subtrees in v.

Transitive closure: Intuition

Consider the runs of t, v, u in $\langle \mathcal{A}_1, \mathcal{A}_2 \rangle$:



Adding $q_i \to q'_i$ to the right-hand side automaton achieves the objective.

Transitive closure: $R^* \subseteq \mathcal{L}(\langle \mathcal{B}_1, \mathcal{B}_2 \rangle)$

Proof by induction: Let $\langle t, u \rangle \in R^i$, for $i \geq 0$.

- $i = 0$: trivial
- $i \to i + 1$: Let v s.t. $\langle t, v \rangle \in R^i$ and $\langle v, u \rangle \in R$.
  Then (by induction) $\langle t, v \rangle$ is accepted by $\langle \mathcal{B}_1, \mathcal{B}_2 \rangle$.
  Let P be the positions in which $\langle t, v \rangle$ differ
  and P' be the positions in which $\langle v, u \rangle$ differ.
  All incomparable pairs in $P \times P'$ are handled by the definition of GTT.
  For $p \in P$ and $pp1, \ldots, pp_n \in P'$ consider the previous drawings.
  The case $pp1, \ldots, pp_n \in P$ and $p \in P'$ is symmetric.

Transitive closure: $R^* \supseteq \mathcal{L}(\langle \mathcal{B}_1, \mathcal{B}_2 \rangle)$

Let $\langle \mathcal{B}_1^i, \mathcal{B}_2^i \rangle$ denote the GTT after adding i transitions and show that its language is included in R^*.

- $i = 0$: trivial
- $i \to i+1$: Let $q \to q'$ be the transition added in the $(i+1)$-th step (to $\mathcal{B}_1$, say) and let $q \to q'$ be used j times in accepting some $\langle t, u \rangle$.

  If $j = 0$, then $\langle t, u \rangle \in R^*$ by induction hypothesis. Otherwise:

  1. there exist $n \geq 0$, $C \in \mathcal{C}^n(\mathcal{F})$ etc such that $t = C[t_1, \ldots, t_n]$, $u = C[u_1, \ldots, u_n]$ and $\forall k : t_k \to^*_{\mathcal{B}_1^{i+1}} q_k \;{}_{\mathcal{B}_2^{i+1}}^{\;*}\!\leftarrow u_k$.
  2. Suppose $t_k = C'[t'] \to^*_{\mathcal{B}_1^{i+1}} C'[q] \to C'[q'] \to^*_{\mathcal{B}_1^{i+1}} q_k$ for some k, C', t'.
  3. There must be some $v \in T(\mathcal{F})$ with $v \to^*_{\mathcal{B}_2^i} q$ and $v \to^*_{\mathcal{B}_1^i} q'$.
  4. From (2) et (3) we have $C'[v] \to^*_{\mathcal{B}_1^{i+1}} q_k$.
  5. Replacing t_k by $C'[v]$ in (1) we get $\langle t[t'/v], u \rangle \in \mathcal{L}(\langle \mathcal{B}_1^{i+1}, \mathcal{B}_2^{i+1} \rangle)$ with fewer than j times $q \to q'$, thus by ind.hyp. $\langle t[t'/v], u \rangle \in R^*$.
  6. From (2) and (3), $t' \to^*_{\mathcal{B}_1^{i+1}} q \;{}_{\mathcal{B}_2^{i}}^{\;*}\!\leftarrow v$, with fewer than j times $q \to q'$.
  7. From (6) by ind.hyp. $\langle t, t[t'/v] \rangle \in R^*$.

Application: XML

XML = Extensible Markup Language

- Conceived for platform-independent exchange of *structured data*
- An XML document consists of *tags* with *attributes* and text (parsed character data, *pcdata*)

Example:

```
<html><head><meta charset="UTF-8"/>
<title>My web page</title></head>
<body><p>Bonne ann&eacute;e !</p></body></html>
```

- A *well-formed* XML document forms a tree (balanced tags, one single root tag)
- Testing for validity / generating tree from document: visibly pushdown automaton, LL/LR parser

Valid XML documents

- Languages of XML documents defined by *schemas* (DTD, XML Schema, Relax NG)
- Schemas define permissible tag (+attributes) and their nesting
- Examples of XML languages: HTML, SVG, KML, ...

- *Valid* XML document: well-formed document satisfying a schema
- Example: XML-Schema for KML

DTD for XML

DTD = Document Type Definition

DTD define a (restricted) subclass of XML languages.
Essentially, defines a regular language of child tags for each tag type.

Example (from Wikipedia):

```
<!ELEMENT html (head,body)>
<!ELEMENT hr EMPTY>
<!ELEMENT div (#PCDATA | p | ul | | table | pre | hr |
          h1|h2|h3|h4|h5|h6 | blockquote | ...)*>
<!ELEMENT dl (dt|dd)+>
```

Validity checking of DTD

The language of XML documents defined by DTD is accepted by NHA.

Restrictions on DTD

Expressivity of DTD

There are hedge-recognizable languages that cannot be defined by DTD.

Example: $\{f(g(a)), f'(g(b))\}$

DTD contain another restriction:

> *It is an error if the content model allows an element to match more than one occurrence of an element type in the content model.*

E.g., $(ab|ac)$ is not allowed (but $a(b|c)$ is).

Deterministic regular expressions

Definition: Marked RE

Let e be a RE over Σ. The *marked RE* $\bar{e}$ is a RE over $\Sigma \times \mathbb{N}$ obtained by adding a unique subscript to each letter in e.

Example: $e = (ab|ac)$, then $\bar{e} = (a_1 b_2 | a_3 c_4)$

Definition: Deterministic RE

Let e a RE over Σ. We call e *deterministic* if $\bar{e}$ satisfies the following: for all $u, v, w \in (\Sigma \times \mathbb{N})^*$ and $a \in \Sigma$, if $ua_i v, ua_j w \in L(\bar{e})$ then $i = j$.

Example: $e = (ab|ac)$, $\bar{e} = (a_1 b_2 | a_3 c_4)$, not deterministic because $a_1 b_2, a_3 c_4 \in L(\bar{e})$

Parsing deterministic RE

Parsing det. RE

Let e be a deterministic RE. A DFA for e can be constructed in polynomial (linear) time. [Brüggemann-Klein 1993, Groz et al 2012]

Proof (sketch): Construction of Glushkov automaton from e.

Expressivity of det. RE

Not every regular language can be defined by a deterministic RE.

XML Schema

XML Schema can define more expressive XML languages.
Example:

```
<xsd:complexType name="track">
 <xsd:sequence minOccurs="1" maxOccurs="unbounded">
  <xsd:choice>
   <xsd:element name="invSession" type="invSession"
    minOccurs="1" maxOccurs="1"/>
   <xsd:element name="conSession" type="conSession"
    minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
  <xsd:element name="break" type="xsd:string"
    minOccurs="0" maxOccurs="1"/>
 </xsd:sequence>
</xsd:complexType>
```

XML Schema and Hedge Automata

XML Schema = NHA

XML Schema (restricted to occurrence and nesting conditions) correspond to the class of hedge-recognizable languages.

Moreover, XML Schema also permit non-hedge-recognizable features:

- constraints on data types in attributes and pcdata
- consistency constraints (e.g., unique keys)

XSL Transformation

- XSLT allows to transform XML documents into other documents (incl. non XML)
- XQuery used to specify nodes on which to apply a transformation

Example (from Wikipedia):

```
<xsl:template match="//title">
  <em>
    <xsl:apply-templates/>
  </em>
</xsl:template>
<xsl:for-each select="book">
  <xsl:sort select="price" order="ascending" />
</xsl:for-each>
```