Tree Automata and Applications

M1 course, 2021/2022
Organization

Timetable

- Exercises: Thursday 8:30 – 10:30 (Amrita Suresh)
- Course: Thursday 10:45 – 12:45 (Stefan Schwoon)

Exams

- DM or CC (to be specified by Amrita)
- Final Exam: 2h, 14 January
- First session: DM/CC + Exam (50/50)
- Second session: DM/CC + Repeat Exam (50/50)

Course materials

- Website: Wiki MPRI, course 1-18 (exercise sheets, slides, former exams)
Organization

Timetable

- Exercises: Thursday 8:30 – 10:30 (Amrita Suresh)
- Course: Thursday 10:45 – 12:45 (Stefan Schwoon)

Exams

- DM or CC (to be specified by Amrita)
- Final Exam: 2h, 14 January
- First session: DM/CC + Exam (50/50)
- Second session: DM/CC + Repeat Exam (50/50)
Organization

Timetable

- Exercises: Thursday 8:30 – 10:30 (Amrita Suresh)
- Course: Thursday 10:45 – 12:45 (Stefan Schwoon)

Exams

- DM or CC (to be specified by Amrita)
- Final Exam: 2h, 14 January
- First session: DM/CC + Exam (50/50)
- Second session: DM/CC + Repeat Exam (50/50)

Course materials

- Website: 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...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 $N^*$:

![Diagram of tree]

**Definition: Tree**

A (finite, ordered) *tree* is a non-empty, finite, prefix-closed set $\text{Pos} \subseteq N^*$. 
Let $\mathcal{F}_0, \mathcal{F}_1, \ldots$ be disjoint sets of symbols of \textit{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).
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 $t : Pos \rightarrow (\mathcal{F} \cup \mathcal{X})$ satisfying:
- $Pos$ is a tree;
- for all $p \in Pos$, if $t(p) \in \mathcal{F}_n$, $n \geq 1$ then $Pos \cap pN = \{p_1, \ldots, p_n\}$;
- for all $p \in Pos$, if $t(p) \in \mathcal{X} \cup \mathcal{F}_0$ then $Pos \cap pN = \emptyset$. 
**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.
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.
Trees and Terms

Definition: Terms

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

1. $\mathcal{X} \cup \mathcal{F}_0 \subseteq T(\mathcal{F}, \mathcal{X})$;
2. 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: $\mathcal{F} = \{f(2), g(1), a, b\}$, $\mathcal{X} = \{x, y\}$

1. $f(g(a), b) \in T(\mathcal{F})$;
2. $f(x, f(b, y)) \in T(\mathcal{F}, \mathcal{X})$ is linear;
3. $f(x, x) \in T(\mathcal{F}, \mathcal{X})$ 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 $t|_p : Pos_p \rightarrow T(\mathcal{F}, \mathcal{X})$ 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 $t.$

$t \sqsupset t'$ (resp. $t \sqsupsetdot t'$) denotes that $t'$ is a (proper) subtree of $t.$
Substitutions and Context

Definition: Substitution

- (Ground) substitution $\sigma$: 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 $\sigma(t)$


**Substitutions and Context**

**Definition: Substitution**
- (Ground) substitution $\sigma$: 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 $C[t_1, \ldots, t_n] := C\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$.

$C^n(\mathcal{F})$ denotes the contexts with $n$ variables and $C(\mathcal{F}) := C^1(\mathcal{F})$.

Let $C \in C(\mathcal{F})$. 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:

\[ abc \equiv a(b(c($))) \]

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
Tree automata

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

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

- $Q$ is a finite set of states;
- $\mathcal{F}$ a finite ranked alphabet;
- $G \subseteq Q$ are the final states;
- $\Delta$ is a finite set of rules of the form
  \[ f(q_1, \ldots, q_n) \rightarrow q \]
  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
  - $a \rightarrow q_0$
  - $g(q_0) \rightarrow q_1$
  - $g(q_1) \rightarrow q_1$
  - $f(q_1, q_1) \rightarrow q_f$.
**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*;
- $\Delta$ is a finite set of rules of the form $f(q_1, \ldots, q_n) \rightarrow q$ 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

\[
a \rightarrow q_0 \quad g(q_0) \rightarrow q_1 \quad g(q_1) \rightarrow q_1 \quad f(q_1, q_1) \rightarrow q_f
\]
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 $\rightarrow_{\mathcal{A}}^{\star}$ for the transitive and reflexive closure of $\rightarrow_{\mathcal{A}}$. 
Move relation and computation tree

Move relation

Let \( t, t' \in T(\mathcal{F}, Q) \). We write \( t \rightarrow_A t' \) 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 \( A \).

Idea: successively reduce \( t \) to a single state, starting from the leaves. As usual, we write \( \rightarrow_A^* \) for the transitive and reflexive closure of \( \rightarrow_A \).

Computation

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

- for all \( p \in \text{Pos} \), if \( t(p) = f \in \mathcal{F}_n \), \( t'(p) = q \), and \( t'(pj) = q_j \) for all \( pj \in \text{Pos} \cap p\mathcal{N} \), then \( f(q_1, \ldots, q_n) \rightarrow q \in \Delta \).
A tree $t$ is accepted by $A$ iff $t \xrightarrow{\tau}^* q$ for some $q \in G$.

$L(A)$ denotes the set of trees accepted by $A$.

$L$ is regular/recognizable iff $L := L(A)$ for some NFTA $A$.

Two NFTAs $A_1$ and $A_2$ are equivalent iff $L(A_1) = L(A_2)$. 
NFTA with $\varepsilon$-moves

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

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

**Equivalence of $\varepsilon$-NFTA**
For every $\varepsilon$-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 \( f \in F_n \) and \( q_1, \ldots, q_n \in Q \), there exists at least one rule \( f(q_1, \ldots, q_n) \rightarrow q \in \Delta \).

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

A state \( q \) of \( A \) is *accessible* if there exists a tree \( t \) s.t. \( t \rightarrow^*_A q \). \( A \) 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 $C, D \in C(\mathcal{F})$ 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 $A$ recognizing $L$. Then an accepting run for $t$ has positions $p, p'$ (with $p' \neq \varepsilon$) labelled with the same state $q$. Let $C := t[x][p], D := t[x][p'][q], \text{and } u := t[x][pp']$. We have $t = C[D[u]] \in L$, $D[u] \rightarrow^* Aq$, and $u \rightarrow^* Aq$, hence the accepting run of $t$ implies $D[q] \rightarrow^* Aq$ and $C[q] \rightarrow^* Aq_f$, for some final $q_f$. Therefore, $C[u] \rightarrow^* Aq_f$ and for any $n \geq 0$, (by induction) $C[D^n[u]] \in L$. 

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 $A$ 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 $t = C[D[u]] \in L$, $D[u] \xrightarrow{A}^* q$, and $u \xrightarrow{A}^* q$, hence the accepting run of $t$ implies $D[q] \xrightarrow{A}^* q$ and $C[q] \xrightarrow{A}^* q_f$, for some final $q_f$. Therefore, $C[u] \xrightarrow{A}^* q_f$ and for any $n \geq 0$, (by induction)

\[ C[D^{n+1}[u]] \xrightarrow{A}^* C[D^n[D[q]]] \xrightarrow{A}^* C[D^n[q]] \xrightarrow{A}^* C[q] \xrightarrow{A}^* q_f \]
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 \( A \) 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 $\Delta$ contains rules of the form

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

for $f \in \mathcal{F}_n$ and $q, q_1, \ldots, q_n \in Q$. 
Top-down tree automata

Definition

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

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

for $f \in F_n$ and $q, q_1, \ldots, q_n \in Q$.

Move relation: $t \rightarrow_A t'$ iff

1. $t = C[q(f(t_1, \ldots, t_n))]$ for some context $C$, $f \in F_n$, and $t_1, \ldots, t_n \in T(F)$;
2. $t' = C[f(q_1(t_1), \ldots, q_n(t_n))]$ for some rule $q(f) \rightarrow (q_1, \ldots, q_n)$.

$t$ is accepted by $A$ if $q(t) \rightarrow_A^* t$ for some $q \in I$. 
Theorem (T-NFTA = NFTA)

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

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

$$\Delta' := \{ f(q) \to (q_1, \ldots, q_n) \mid f(q_1, \ldots, q_n) \to q \in \Delta \}$$
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 $A = \langle Q, F, G, \Delta \rangle$ iff it is accepted by T-NFTA $A' = \langle Q, F, G, \Delta' \rangle$, with

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

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

- **Base:** $t = a$ (for some $a \in F_0$)

  $$t = a \rightarrow^{*}_A q \iff a \rightarrow^{\Delta} q \iff q(a) \rightarrow^{\Delta'} e \iff q(a) \rightarrow^{*}_{A'} a$$
Theorem (T-NFTA = NFTA)

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

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

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

Proof: Let $t \in T(F)$. We show $t \rightarrow^*_A q$ iff $q(t) \rightarrow^*_A' t$.

- **Base:** $t = a$ (for some $a \in F_0$)
  
  $$t = a \rightarrow^*_A q \iff a \rightarrow^{\Delta} q \iff q(a) \rightarrow^{\Delta'} \varepsilon \iff q(a) \rightarrow^*_A' a$$

- **Induction:** $t = f(t_1, \ldots, t_n)$, hypothesis holds for $t_1, \ldots, t_n$
  
  $$f(t_1, \ldots, t_n) \rightarrow^*_A q \iff \exists q_1, \ldots, q_n : f(q_1, \ldots, q_n) \rightarrow^{\Delta} q \land \forall i : t_i \rightarrow^{*}_A q_i$$
  
  $$\iff \exists q_1, \ldots, q_n : q(f) \rightarrow^{\Delta'} (q_1, \ldots, q_n) \land \forall i : q_i(t_i) \rightarrow^{*}_{A'} t_i$$
  
  $$\iff q(f(t_1, \ldots, t_n)) \rightarrow_{A'} f(q_1(t_1), \ldots, q_n(t_n)) \rightarrow^{*}_{A'} f(t_1, \ldots, t_n)$$
From NFTA to DFTA

Theorem (NFTA=DFTA)
If $L$ is recognizable by an NFTA, then it is recognizable by a DFTA.
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 \( L \):

\[
\begin{align*}
G' &= \{ S \subseteq Q \mid S \cap G \neq \emptyset \} \\
&\text{for every } f \in \mathcal{F}_n \text{ and } S_1, \ldots, S_n \subseteq Q, \text{ let } f(S_1, \ldots, S_n) \rightarrow S \in \Delta', \\
&\text{where } S = \{ q \in Q \mid \exists q_1 \in S_1, \ldots, q_n \in S_n : f(q_1, \ldots, q_n) \rightarrow q \in \Delta \}
\end{align*}
\]

Proof: For \( t \in T(\mathcal{F}) \), show \( t \rightarrow_{A'}^* \{ q \mid t \rightarrow_A^* q \} \), by structural induction.
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 $A = \langle Q, F, G, \Delta \rangle$ an NFTA recognizing $L$. The following DCFTA $A' = \langle 2^Q, F, G', \Delta' \rangle$ also recognizes $L$:

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

Proof: For $t \in T(F)$, show $t \rightarrow^*_A \{ q \mid t \rightarrow^*_A q \}$, by structural induction.

DFTA with accessible states

In practice, the construction of $A'$ can be restricted to accessible states: Start with transitions $a \rightarrow S$, then saturate.
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 \(f(S_1, \ldots, S_n) \rightarrow S \in \Delta'\), where \(S = \{ q \in Q \mid \exists q_1 \in S_1, \ldots, q_n \in S_n : f(q_1, \ldots, q_n) \rightarrow q \in \Delta \}\)

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

DFTA with accessible states

In practice, the construction of \(\mathcal{A}'\) can be restricted to accessible states: Start with transitions \(a \rightarrow S\), 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.
Theorem (Boolean closure)

Recognizable tree languages are closed under Boolean operations.
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 \( T(\mathcal{F}) \setminus L \).
## 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 $A_i = \langle Q_i, F, G_i, \Delta_i \rangle$ be NFTA recognizing $L_i$, for $i = 1, 2$. Then $A = \langle Q_1 \times Q_2, F, G_1 \times G_2, \Delta \rangle$ recognizes $L_1 \cap L_2$, where

$$f(q_1, \ldots, q_n) \rightarrow q \in \Delta_1 \quad f(q'_1, \ldots, q'_n) \rightarrow q' \in \Delta_2$$

$$f(\langle q_1, q'_1 \rangle, \ldots, \langle q_n, q'_n \rangle) \rightarrow \langle q, q' \rangle \in \Delta$$
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

\[
\begin{align*}
f(q_1, \ldots, q_n) &\rightarrow q \in \Delta_1 & f(q'_1, \ldots, q'_n) &\rightarrow q' \in \Delta_2 \\
f(\langle q_1, q'_1 \rangle, \ldots, \langle q_n, q'_n \rangle) &\rightarrow \langle q, q' \rangle \in \Delta
\end{align*}
\]

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 \( L_1 \cup L_2 \).
Definition

Let \( X_n := \{x_1, \ldots, x_n\} \) and \( \mathcal{F}, \mathcal{F}' \) ranked alphabets.

A tree homomorphism is a mapping \( h : \mathcal{F} \rightarrow T(\mathcal{F}', X) \), with \( h(f) \in T(\mathcal{F}, X_n) \) if \( f \in \mathcal{F}_n \).

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

\[
  h(f(t_1, \ldots, t_n)) = h(f)\{x_1 \leftarrow h(t_1), \ldots, x_n \leftarrow h(t_n)\}
\]

Intuition:

\[
  h(f) \text{ “explodes” } f\text{-positions into trees}
\]

\[
  \text{reorders/copies/deletes subtrees.}
\]
Example

\[ \mathcal{F} = \{ f(2), g(1), a \}, \mathcal{F}' = \{ f'(1), g'(2), a, b \} \]

\[ h(f) = f'(g'(x_2, b)), h(g) = g'(x_1, a), h(a) = g'(a, b) \]
Examples

Example

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

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.
Properties of homomorphisms

A homomorphism \( h \) is

- \textit{linear} if \( h(f) \) linear for all \( f \);
- \textit{non-erasing} if \( \mathcal{H}(h(f)) > 0 \) for all \( f \);
- \textit{flat} if \( \mathcal{H}(h(f)) = 1 \) for all \( f \);
- \textit{complete} if \( f \in \mathcal{F}_n \) implies that \( h(f) \) contains all of \( \mathcal{X}_n \);
- \textit{permuting} if \( h \) is complete, linear, and flat;
- \textit{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 $L \subseteq T(F)$ be recognizable and $h : F \rightarrow F'$ a linear tree homomorphism. Then $h(L)$ is 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:

- \( \mathcal{F} = \{ f(2), g(1), a \} \), \( \mathcal{F}' = \{ f'(1), g'(2), a, b \} \)
- \( h(f) = f'(g'(x_2, b)) \), \( h(g) = g'(x_1, a) \), \( h(a) = g'(a, b) \)
- \( L = \{ f(g^i(a), g^k(a)) \mid i, k \geq 0 \} \)
- \( \mathcal{A} = \langle \{ q_0, q_1, q_f \}, \mathcal{F}, \{ q_f \}, \Delta \rangle \) recognizes \( L \) with
  \( \Delta := \{ r_1 : a \to q_0, \quad r_2 : g(q_0) \to q_1, \quad r_3 : g(q_1) \to q_1, \quad r_4 : f(q_1, q_1) \to q_f \} \)
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:

\( \mathcal{F} = \{ f(2), g(1), a \} \), \( \mathcal{F}' = \{ f'(1), g'(2), a, b \} \)

\( h(f) = f'(g'(x_2, b)), h(g) = g'(x_1, a), h(a) = g'(a, b) \)

\( L = \{ f(g^i(a), g^k(a)) \mid i, k \geq 0 \} \)

\( A = \langle \{ q_0, q_1, q_f \}, \mathcal{F}, \{ q_f \}, \Delta \rangle \) recognizes \( L \) with

\( \Delta := \{ r_1 : a \to q_0, \ r_2 : g(q_0) \to q_1, \ r_3 : g(q_1) \to q_1, \ r_4 : f(q_1, q_1) \to q_f \} \)
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:

- \( \mathcal{F} = \{ f(2), g(1), a \} \), \( \mathcal{F}' = \{ f'(1), g'(2), a, b \} \)
- \( h(f) = f'(g'(x_2, b)), \ h(g) = g'(x_1, a), \ h(a) = g'(a, b) \)
- \( L = \{ f(g^i(a), g^k(a)) \mid i, k \geq 0 \} \)
- \( \mathcal{A} = \langle \{ q_0, q_1, q_f \}, \mathcal{F}, \{ q_f \}, \Delta \rangle \) recognizes \( L \) with
  \( \Delta := \{ r_1 : a \to q_0, \ r_2 : g(q_0) \to q_1, \ r_3 : g(q_1) \to q_1, \ r_4 : f(q_1, q_1) \to q_f \} \)
**Theorem:** Linear homomorphisms preserve recognizability

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

**Illustrating example:**

\[ \mathcal{F} = \{ f(2), g(1), a \}, \quad \mathcal{F'} = \{ f'(1), g'(2), a, b \} \]

\[ h(f) = f'(g'(x_2, b)), \quad h(g) = g'(x_1, a), \quad h(a) = g'(a, b) \]

\[ L = \{ f(g^i(a), g^k(a)) \mid i, k \geq 0 \} \]

\[ \mathcal{A} = \langle \{ q_0, q_1, q_f \}, \mathcal{F}, \{ q_f \}, \Delta \rangle \text{ recognizes } L \text{ with} \]

\[ \Delta := \{ r_1 : a \rightarrow q_0, \quad r_2 : g(q_0) \rightarrow q_1, \quad r_3 : g(q_1) \rightarrow q_1, \quad r_4 : f(q_1, q_1) \rightarrow q_f \} \]
Automaton construction for \( h(L) \)

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

- \( Q' := Q \cup \{ \langle r, p \rangle \mid \exists f \in \mathcal{F}, r \in \Delta : p \in Pos_{h(f)}, r = f(\ldots) \rightarrow q \} \);
- \( \Delta' \) contains, for each transition \( r : f(q_1, \ldots, q_n) \rightarrow q \) in \( \Delta \) and \( p \in Pos_{h(f)} \):
  - \( f' (\langle r, p_1 \rangle, \ldots, \langle r, p_k \rangle) \rightarrow \langle r, p \rangle \) if \( h(f)(p) = f' \in \mathcal{F}'_k \)
  - \( q_i \rightarrow \langle r, p \rangle \) if \( h(f)(p) = x_i \)
  - \( \langle r, \varepsilon \rangle \rightarrow q \)
Automaton construction for $h(L)$

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

- $Q' := Q \cup \{ \langle r, p \rangle \mid \exists f \in \mathcal{F}, r \in \Delta : p \in Pos_{h(f)}, r = f(\ldots) \rightarrow q \}$;
- $\Delta'$ contains, for each transition $r : f(q_1, \ldots, q_n) \rightarrow q$ in $\Delta$ and $p \in Pos_{h(f)}$:
  - $f'((\langle r, p_1 \rangle, \ldots, \langle r, p_k \rangle)) \rightarrow \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'$
  - $q_i \rightarrow \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \rightarrow q$
Automaton construction for $h(L)$

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

- $Q' := Q \cup \{ \langle r, p \rangle \mid \exists f \in \mathcal{F}, r \in \Delta : p \in Pos_{h(f)}, r = f(\ldots) \rightarrow q \}$;
- $\Delta'$ contains, for each transition $r : f(q_1, \ldots, q_n) \rightarrow q$ in $\Delta$ and $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \ldots, \langle r, pk \rangle) \rightarrow \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'$
  - $q_i \rightarrow \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \rightarrow q$
Given an NFTA $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$ for $L$, construct NFTA $\mathcal{A}' = \langle Q', \mathcal{F}', G, \Delta' \rangle$ for $h(L)$.

- $Q' := Q \cup \{ \langle r, p \rangle \mid \exists f \in \mathcal{F}, r \in \Delta : p \in \text{Pos}_{h(f)}, r = f(\ldots) \rightarrow q \}$;
- $\Delta'$ contains, for each transition $r : f(q_1, \ldots, q_n) \rightarrow q$ in $\Delta$ and $p \in \text{Pos}_{h(f)}$:
  - $f'(\langle r, p_1 \rangle, \ldots, \langle r, p_k \rangle) \rightarrow \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'$
  - $q_i \rightarrow \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \rightarrow q$
To prove: \( \mathcal{A}' \) accepts \( h(L) \).
Correctness

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

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

To prove: $A'$ accepts $h(L)$.

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

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

Theorem: Inverse homomorphisms preserve recognizability

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

Let \( L \subseteq T(\mathcal{F})' \) be recognizable and \( h : \mathcal{F} \rightarrow \mathcal{F}' \) 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 \( \mathcal{A} = \langle Q \uplus \{ \text{kill} \}, \mathcal{F}, G, \Delta \rangle \) for \( h^{-1}(L) \).

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

- add \( f(\text{kill}, \ldots, \text{kill}) \rightarrow \text{kill} \) to \( \Delta \);
- if \( h(f)\{x_1 \leftarrow p_1, \ldots, x_n \leftarrow p_n\} \rightarrow_{\mathcal{A}'} q \), add \( f(q_1, \ldots, q_n) \rightarrow q \) to \( \Delta \), with:

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

Proof: Show \( t \rightarrow_{\mathcal{A}}^* q \) iff \( h(t) \rightarrow_{\mathcal{A}'}^* q \), for all \( t \in T(\mathcal{F}) \).
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 $pc(L) = \{ t \mid \pi(t) \subseteq \pi(L) \}$.
- $L$ 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 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.
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 $A = \langle Q, F, G, \Delta \rangle$ be a reduced T-NFTA for $L$.

- Construct a finite (word) automaton out of $A$.
- Construct $A' = \langle Q, F, G, \Delta' \rangle$ for $pc(L)$ as follows:

  for all $n \geq 0, f \in F_n$:
  
  $\forall i: f(q_i) \rightarrow \Delta(q_1, \ldots, q_n) \rightarrow f(q_i) \rightarrow \Delta'(q_1, \ldots, q_n)$

- Let $L_q = L(\langle Q, F, \{q\}, \Delta \rangle)$ and $L'_q = L(\langle Q, F, \{q\}, \Delta' \rangle)$.

- Prove $t \in L'_q \iff \pi(t) \subseteq \pi(L_q)$ for all $q \in Q, t \in T(F)$ by induction.
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 $\mathcal{A}$.
Lemma

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

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

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

- Construct a finite (word) automaton out of $A$.
- Construct $A' = \langle Q, F, G, \Delta' \rangle$ for $\text{pc}(L)$ as follows:
  
  for all $n \geq 0$, $f \in F_n$:
  
  $\forall i : f(q) \rightarrow_{\Delta} (q_{i,1}, \ldots, q_{n,1}) \rightarrow f(q) \rightarrow_{\Delta'} (q_{1,1}, \ldots, q_{n,n})$
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 $A = \langle Q, \mathcal{F}, G, \Delta \rangle$ be a reduced T-NFTA for $L$.

- Construct a finite (word) automaton out of $A$.
- Construct $A' = \langle Q, F, G, \Delta' \rangle$ for $pc(L)$ as follows:
  for all $n \geq 0$, $f \in \mathcal{F}_n$:
  $$\forall i : f(q) \rightarrow_\Delta (q_{i,1}, \ldots, q_{n,1}) \rightarrow f(q) \rightarrow_\Delta' (q_{1,1}, \ldots, q_{n,n})$$

Let $L_q = \mathcal{L}(\langle Q, \mathcal{F}, \{q\}, \Delta \rangle)$ and $L'_q = \mathcal{L}(\langle Q, F, \{q\}, \Delta' \rangle)$.

Prove $t \in L'_q \iff \pi(t) \subseteq \pi(L_q)$ for all $q \in Q$, $t \in T(\mathcal{F})$ by induction.
Path closure and T-NFTA

Theorem

Let $L \subseteq T(F)$ be a recognizable tree language. $L$ is path-closed iff it is recognized by a T-DFTA.
Path closure and T-NFTA

Theorem

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

Proof:

- “$\rightarrow$”:
  Let $A = \langle Q, F, G, \Delta \rangle$ be a T-NFTA for $L$. Construct a T-DFTA $A = \langle 2^Q, F, \{G\}, \Delta' \rangle$ with $f(S) \rightarrow'_{\Delta} (S_1, \ldots, S_n)$ where $S_i = \{ q_i | \exists q \in S, f(q) \rightarrow_{\Delta} (q_1, \ldots, q_n) \}$.

- “$\leftarrow$”:
  Let $A$ 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 $q \in Q$, $t \in T(F)$.

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 T-NFTA for $L$.
  Construct a T-DFTA $\mathcal{A} = \langle 2^Q, \mathcal{F}, \{ G \}, \Delta' \rangle$ with
  
  \[ f(S) \xrightarrow{\Delta'} (S_1, \ldots, S_n) \]

  where $S_i = \{ q_i \mid \exists q \in S, f(q) \xrightarrow{\Delta} (q_1, \ldots, q_n) \}$.

- "$\Leftarrow$":
  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 $q \in Q, t \in T(\mathcal{F})$. 

Corollary

It is decidable whether a recognizable tree language is path-closed.
Path closure and T-NFTA

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 T-NFTA for \( L \).
  Construct a T-DFTA \( \mathcal{A} = \langle 2^Q, \mathcal{F}, \{ G \}, \Delta' \rangle \) with
  \[
  f(S) \rightarrow'_{\Delta} (S_1, \ldots, S_n)
  \]
  where \( S_i = \{ q_i \mid \exists q \in S, f(q) \rightarrow \Delta (q_1, \ldots, q_n) \} \).

- **"\( \leftarrow \)"**:
  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 \( q \in Q, t \in T(\mathcal{F}) \).

Corollary
It is decidable whether a recognizable tree language is path-closed.
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 \( u \in L \iff v \in L \).
### Definition: Congruence

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

- $\equiv$ is called a **congruence** if for any $n \geq 0$ and $f \in F_n$, $u_1 \equiv v_1, \ldots, u_n \equiv v_n$ 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 $L \subseteq T(F)$, write $u \equiv_L v$ if 

$$\forall C \in C(F) : C[u] \in L \iff C[v] \in L$$
**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

$$\forall C \in \mathcal{C}(\mathcal{F}) : C[u] \in L \iff C[v] \in L$$

**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 \( L \) is not recognizable.
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 \( L \) is not recognizable.

Proof of the theorem (sketch):

1 → 2: Let \( A \) be DCFTA and let \( u \equiv v \) iff \( u \xrightarrow{\ast} A q \xleftarrow{\ast} A v \).

Then \( \equiv \) is of finite index and saturates \( L \).
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 \( L \) is not recognizable.

Proof of the theorem (sketch):

- **1 \rightarrow 2**: Let \( \mathcal{A} \) be DCFTA and let \( u \equiv v \) iff \( u \xrightarrow{\mathcal{A}}^* q \xleftarrow{\mathcal{A}} v \).
  Then \( \equiv \) is of finite index and saturates \( L \).

- **2 \rightarrow 3**: Let \( \equiv \) be a saturating congruence, \( u \equiv v \) implies \( u \equiv_L v \) (recurrence over height of \( x \) in context \( C \)).
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 \( L \) is not recognizable.

Proof of the theorem (sketch):

1 → 2: Let \( \mathcal{A} \) be DCFTA and let \( u \equiv v \) iff \( u \xrightarrow{\ast}_\mathcal{A} q \xleftarrow{\ast}_\mathcal{A} v \).
Then \( \equiv \) is of finite index and saturates \( L \).

2 → 3: Let \( \equiv \) be a saturating congruence, \( u \equiv v \) implies \( u \equiv_L v \)
(recurrence over height of \( x \) in context \( C \)).

3 → 1: Let \( \mathcal{A} = \langle T(\mathcal{F})/ \equiv_L, \mathcal{F}, L/ \equiv_L, \Delta \rangle, \) with
\[
f([u_1], \ldots, [u_n]) \rightarrow [f(u_1, \ldots, u_n)]
\]
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 \( u \in T(\mathcal{F})\);
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 \( L \) is not recognizable.

Proof of the theorem (sketch):

1. \( \rightarrow \) 2: Let \( \mathcal{A} \) be DCFTA and let \( u \equiv v \) iff \( u \rightarrow_A^* q \leftarrow v \).
   Then \( \equiv \) is of finite index and saturates \( L \).

2. \( \rightarrow \) 3: Let \( \equiv \) be a saturating congruence, \( u \equiv v \) implies \( u \equiv_L v \)
   (recurrence over height of \( x \) in context \( C \)).

3. \( \rightarrow \) 1: Let \( \mathcal{A} = \langle T(F) / \equiv_L, F, L / \equiv_L, \Delta \rangle \), with
   \[ f([u_1], \ldots, [u_n]) \rightarrow [f(u_1, \ldots, u_n)] \]
   for all \( n \geq 0 \), \( f \in F_n \), \( u_1, \ldots, u_n \in T(F) \),
   where \([u]\) is the equivalence class of \( u \in T(F) \);

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

The following problem is EXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $L(A_1) \cap \cdots \cap L(A_n) \neq \emptyset$?

Proof (sketch):
- **Hardness**: Simulate a linear-space ATM $M$ with input of length $n$.
  - If $M$ accepts the input, there is an accepting run.
  - Encode the run of $M$ as a tree.
  - Construct $A_i$, for $i = 1, \ldots, n$, to check:
    1. if $M$ 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 $A_1 \times \cdots \times A_n$.

Detailed proof: Veanes, 1997
The following problem is EXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $L(A_1) \cap \cdots \cap L(A_n) \neq \emptyset$?

Proof (sketch):

- **Hardness**: Simulate an linear-space ATM $M$ with input of length $n$. 

   ▶ Hardness: Simulate an linear-space ATM $M$ with input of length $n$. 

   ▶ Membership: Compute the productive tuples of states in $A_1 \times \cdots \times A_n$. 

   Detailed proof: Veanes, 1997
Theorem

The following problem is EXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $L(A_1) \cap \cdots \cap L(A_n) \neq \emptyset$?

Proof (sketch):

- Hardness: Simulate an linear-space ATM $M$ with input of length $n$. If $M$ accepts the input, there is an accepting run. Encode the run of $M$ as a tree.

Detailed proof: Veanes, 1997
Theorem

The following problem is EXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $L(A_1) \cap \cdots \cap L(A_n) \neq \emptyset$?

Proof (sketch):

- **Hardness:** Simulate an linear-space ATM $M$ with input of length $n$.
  If $M$ accepts the input, there is an accepting run.
  Encode the run of $M$ as a tree.
  Construct $A_i$, for $i = 1, \ldots, n$, to check:
    1. if $M$ 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 $A_1 \times \cdots \times A_n$.
  Detailed proof: Veanes, 1997
Intersection problem

Theorem

The following problem is EXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $L(A_1) \cap \cdots \cap L(A_n) \neq \emptyset$?

Proof (sketch):

- **Hardness:** Simulate an linear-space ATM $M$ with input of length $n$.
  If $M$ accepts the input, there is an accepting run.
  Encode the run of $M$ as a tree.
  Construct $A_i$, for $i = 1, \ldots, n$, to check:
    1. if $M$ 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 $A_1 \times \cdots \times A_n$. 

Detailed proof: Veanes, 1997
Theorem

The following problem isEXPTIME-complete:
Given tree automata $A_1, \ldots, A_n$, is $\mathcal{L}(A_1) \cap \cdots \cap \mathcal{L}(A_n) \neq \emptyset$?

Proof (sketch):

- **Hardness:** Simulate an linear-space ATM $M$ with input of length $n$. If $M$ accepts the input, there is an accepting run. Encode the run of $M$ as a tree. Construct $A_i$, for $i = 1, \ldots, n$, to check:
  1. if $M$ 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 $A_1 \times \cdots \times A_n$.

Detailed proof: Veanes, 1997
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: $t = f(a, g(b, a), c)$, $fr(t) = abac$
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 $L = fr(L(\mathcal{A}))$. 
Visibly pushdown automata

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

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

Visibly pushdown automaton

Let $A = \langle Q, \Sigma, \Gamma, T, q_0 z_0, F \rangle$ be a pushdown automaton. $A$ 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.
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

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

Closure properties

Languages accepted by VPA are closed under boolean operations.

VPA and tree languages

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

Alternative specification for sets of trees
E.g., to describe valid HTML documents:

- A \texttt{p} tag may only appear inside a \texttt{body} tag.
- A \texttt{dl} tag must contain pairs of \texttt{dt} and \texttt{dd} tags.
Logic over trees

Alternative specification for sets of trees
E.g., to describe valid HTML documents:

- A \texttt{p} tag may only appear inside a \texttt{body} tag.
- A \texttt{dl} tag must contain pairs of \texttt{dt} and \texttt{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.
## First-order logic (FO)

Let $\sigma = ((R_i)_{1 \leq i \leq n})$ be a relation signature and $X_1 = \{x_1, x_2, \ldots\}$ a set of variables. The first-order formulas $FO(\sigma)$ are:

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

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

Second-order logic: allow quantifying over relations

Monadic: only quantify over sets

Monadic second-order logic (MSO)

Let $\sigma$ 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:

$$R_i(x_{j_1}, \ldots, x_{j_i}) \mid x = x' \mid x \in X \mid \neg \phi \mid \phi \land \phi' \mid \exists x. \phi \mid \exists X. \phi$$
Recall: First-/second-order logic

First-order logic (FO)

Let $\sigma = ((R_i)_{1 \leq i \leq n})$ be a relation signature and $X_1 = \{x_1, x_2, \ldots\}$ a set of variables. The first-order formulas $FO(\sigma)$ are:

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

Second-order logic: allow quantifying over relations

Monadic: only quantify over sets

Monadic second-order logic (MSO)

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

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

Weak second-order: only quantify over finite sets

WS$k$S (weak MSO over with $k$ successors)

$WSkS = MSO(<_1, \ldots, <_k)$
Semantics of MSO

**Definition**

Let $\mathcal{M}$ a domain, $\sigma$ a signature, $\nu$ a valuation with

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

Definition
Let $\mathcal{M}$ a domain, $\sigma$ a signature, $\nu$ a valuation with

- $\nu(x) \in \mathcal{M}$ for $x \in X_1$
- $\nu(X) \subseteq \mathcal{M}$ for $X \in X_2$

$\mathcal{M}, \sigma, \nu \models R_i(x_{j_1}, \ldots, x_{j_i})$ if $(\nu(x_{j_1}), \ldots, \nu(x_{j_i})) \in R_i$

$\mathcal{M}, \sigma, \nu \models x = x'$ if $\nu(x) = \nu(x')$

$\mathcal{M}, \sigma, \nu \models x \in X$ if $\nu(x) \in \nu(X)$

$\mathcal{M}, \sigma, \nu \models \neg \phi$ if $\mathcal{M}, \sigma, \nu \not\models \phi$

$\mathcal{M}, \sigma, \nu \models \phi \land \phi'$ if $\mathcal{M}, \sigma, \nu \models \phi \land \mathcal{M}, \sigma, \nu \models \phi'$

$\mathcal{M}, \sigma, \nu \models \exists x. \phi$ if $\exists m \in \mathcal{M}$. $\mathcal{M}, \sigma, \nu[x \mapsto m] \models \phi$

$\mathcal{M}, \sigma, \nu \models \exists X. \phi$ if $\exists M \subseteq \mathcal{M}$. $\mathcal{M}, \sigma, \nu[X \mapsto M] \models \phi$

We omit $\mathcal{M}, \sigma$ when clear from context.
**Semantics of MSO**

**Definition**

Let $\mathcal{M}$ a domain, $\sigma$ a signature, $\nu$ a valuation with

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

\[
\begin{align*}
\mathcal{M}, \sigma, \nu \models R_i(x_{j_1}, \ldots, x_{j_i}) & \quad \text{if} \quad (\nu(x_{j_1}), \ldots, \nu(x_{j_i})) \in R_i; \\
\mathcal{M}, \sigma, \nu \models x = x' & \quad \text{if} \quad \nu(x) = \nu(x'); \\
\mathcal{M}, \sigma, \nu \models x \in X & \quad \text{if} \quad \nu(x) \in \nu(X); \\
\mathcal{M}, \sigma, \nu \models \neg \phi & \quad \text{if} \quad \mathcal{M}, \sigma, \nu \not\models \phi; \\
\mathcal{M}, \sigma, \nu \models \phi \land \phi' & \quad \text{if} \quad \mathcal{M}, \sigma, \nu \models \phi \land \mathcal{M}, \sigma, \nu \models \phi'; \\
\mathcal{M}, \sigma, \nu \models \exists x. \phi & \quad \text{if} \quad \exists m \in \mathcal{M}. \mathcal{M}, \sigma, \nu[x \mapsto m] \models \phi; \\
\mathcal{M}, \sigma, \nu \models \exists X. \phi & \quad \text{if} \quad \exists \mathcal{M} \subseteq \mathcal{M}. \mathcal{M}, \sigma, \nu[X \mapsto \mathcal{M}] \models \phi.
\end{align*}
\]

We omit $\mathcal{M}, \sigma$ when clear from context.
Recall: Common abbreviations

- \( \forall x, \forall X, \lor, \) etc can be expressed in the usual way.

- \( X \subseteq Y: \)
  \[
  \forall x.(x \in X \rightarrow x \in Y)
  \]

- \( Z = X \cup Y: \)
  \[
  \forall x.(x \in Z \leftrightarrow x \in X \lor x \in Y)
  \]

- \( \text{Partition}(X, X_1, \ldots, X_m): \)
  \[
  \left( \forall x.(x \in X \leftrightarrow \bigvee_{i=1}^{m} x \in X_i) \right) \land \left( \bigwedge_{i=1}^{m} \bigwedge_{j \neq i} \forall x.(x \notin X_i \lor x \notin X_j) \right)
  \]

- Similarly, \( X = \emptyset, X = \{x\}, X = Y, \ldots \)
WS$kS$ and trees

Let $\mathcal{M} = N^*$, we fix $<;i$ to be the relation $p <;i q$ iff $\exists p'.q = pip'$. We define $< = \bigcup_{i=1}^{k} <;i$ and $\leq$ as usual, and $\varepsilon$ for the minimal element. We write $xi$ to denote the least $q$ s.t. $\nu(x) <;i q$. 
WS$kS$ and trees

Let $\mathcal{M} = \mathbb{N}^*$, we fix $<_i$ to be the relation $p <_i q$ iff $\exists p'.q = pip'$. We define $<_i = \bigcup_{i=1}^{k} <_i$ and $\leq$ as usual, and $\varepsilon$ for the minimal element. We write $x_i$ 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 $f \in \mathcal{F}$, $S_f = \{ p \in \text{Pos}_t \mid t(p) = f \}$.
**WS**<sub>kS</sub> and trees

Let \( \mathcal{M} = \mathbb{N}^* \), we fix \(<_i \) to be the relation \( p <_i q \) iff \( \exists p'. q = pip' \).

We define \(< = \bigcup_{i=1}^{k} <_i \) and \( \leq \) as usual, and \( \varepsilon \) for the minimal element.

We write \( x_i \) 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:

\[
Tree(S, S_{\mathcal{F}}) := S \neq \emptyset \land \text{Partition}(S, S_{\mathcal{F}})
\land \forall x. \forall y. (x \in S \land y < x) \rightarrow y \in S
\land \bigwedge_{n=1}^{k} \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^{n} (x \in S_f \rightarrow x_i \in S)
\land \bigwedge_{n=1}^{k} \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=n+1}^{k} (x \in S_f \rightarrow x_i \notin S)
\]
Semantics of WS\(k\)S on trees

Coded valuation

Let \(F' := F \times 2^{X_1 \cup X_2}\). The arity of \((f, \tau)\) is \(n\) if \(f \in F_n\).

Let \(t \in T(F)\) and \(\nu\) a valuation. The tuple \(\langle t, \nu \rangle\) is coded by a tree \(t' \in T(F')\), as follows, for all \(p \in Pos\) and \(t'(p) = \langle f, \tau \rangle\):

- if \(x \in X_1\) then \(\tau(x) = 1\) iff \(p = \nu(x)\);
- if \(X \in X_2\) then \(\tau(X) = 1\) iff \(p \in \nu(X)\).

A tree \(t' \in T(F')\) is valid (\(t' \in T_v(F')\)) if it codes some \(\langle t, \nu \rangle\).
Semantics of WS\(kS\) on trees

Coded valuation

Let \(\mathcal{F}' := \mathcal{F} \times 2^{\mathcal{X}_1 \cup \mathcal{X}_2}\). The arity of \((f, \tau)\) is \(n\) if \(f \in \mathcal{F}_n\).

Let \(t \in T(\mathcal{F})\) and \(\nu\) a valuation. The tuple \(\langle t, \nu \rangle\) is coded by a tree \(t' \in T(\mathcal{F}')\), as follows, for all \(p \in \text{Pos}\) 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 \((t' \in T_{\nu}(\mathcal{F}'))\) if it codes some \(\langle t, \nu \rangle\).

Semantics of WS\(kS\)

Let \(\phi\) be a formula of WS\(kS\) and \(V \subseteq (\mathcal{X}_1 \cup \mathcal{X}_2) \uplus (\{S\} \cup \mathcal{S}_\mathcal{F})\) its free variables.

\[
\mathcal{L}(\phi) := \{ \langle t, \nu \rangle \in T_{\nu}(\mathcal{F}') \mid \nu[(S, \mathcal{S}_\mathcal{F}) \mapsto C(t)] \models \phi \}
\]
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 $\nu'(x) = 1$
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 \( \nu'(x) = 1 \)

\[
\begin{array}{c}
\langle f, 101 \rangle \\
\downarrow \\
\langle g, 000 \rangle \\
\uparrow \\
\langle a, 011 \rangle \\
\end{array}
\quad
\begin{array}{c}
\langle f, 0 \rangle \\
\downarrow \\
\langle g, 1 \rangle \\
\uparrow \\
\langle a, 0 \rangle \\
\end{array}
\]

We have \( C(t) = (S, S_f, S_g, S_a) \) with \( S = \{ \varepsilon, 1, 11, 2 \} \), \( S_f = \{ \varepsilon \} \), \( S_g = \{ 1 \} \), \( S_a = \{ 11, 2 \} \).

\( \nu'[(S, S_F) \mapsto C(t)] \models x \in S_g \), thus \( \langle t, \nu' \rangle \in L(x \in S_g) \).

\( t \in L(\exists x. x \in S_g) \)
**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 WS\( k \)S.

Proof (sketch)

1. **DCFTA \( A \rightarrow WSkS \)**: Construct formula \( \phi \) that
   - (i) verifies that the structure is a tree;
   - (ii) guesses a computation of \( A \), 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.

2. **WSkS \( \phi \rightarrow NFTA \)**: Proceed by recurrence on \( \phi \), show that all subformulae of \( \phi \) are recognizable.
Theorem

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

Proof: (sketch)

- DCFTA \( \mathcal{A} \rightarrow WSkS \): Construct formula \( \phi \) 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.

WS\( k \)S \( \phi \rightarrow NFTA \mathcal{A} \): Proceed by recurrence on \( \phi \), show that all subformulae of \( \phi \) are recognizable.
Theorem

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

Proof: (sketch)

- **DCFTA $\mathcal{A} \rightarrow WSkS$:** Construct formula $\phi$ 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.

- **WS$kS$ $\phi \rightarrow NFTA \mathcal{A}$:** Proceed by recurrence on $\phi$,
  show that all subformulae of $\phi$ are recognizable.
Example:  DCFTA  →  WS$kS

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

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

Corresponding formula:

$$\phi = \text{Tree}(S, S_{\mathcal{F}})\land \exists Q_0, Q_1, Q_f.\text{Partition}(S, Q_0, Q_1, Q_f)$$

$$\land \forall x.((x \in S_a \rightarrow x \in Q_0)$$

$$\land \forall x.((x \in S_g \land x_1 \in Q_0) \rightarrow x \in Q_1)$$

$$\land \forall x.((x \in S_g \land x_1 \in Q_1) \rightarrow x \in Q_1)$$

$$\land \forall x.((x \in S_f \land x_1 \in Q_1 \land x_2 \in Q_1) \rightarrow x \in Q_f)$$

$$\land \varepsilon \in Q_f$$
Example: WS$kS \rightarrow NFTA$

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

- $\phi = x \in S_g$
  
  $A_\phi = \langle \{q, q'\}, F \times 2^{\{x\}}, \{q'\}, \Delta \rangle$ with transitions
  
  $\langle a, 0 \rangle \rightarrow q$
  
  $\langle g, 1 \rangle(q) \rightarrow q'$  
  $\langle g, 0 \rangle(q) \rightarrow q'$  
  $\langle g, 0 \rangle(q') \rightarrow q'$
  
  $\langle f, 0 \rangle(q, q) \rightarrow q$  
  $\langle f, 0 \rangle(q, q') \rightarrow q'$  
  $\langle f, 0 \rangle(q', q) \rightarrow q'$

accepts $L(x \in S_g)$ (scans for a single $g$-position with $\tau(x) = 1$).
Consider $\mathcal{F} = \{f(2), g(1), a\}$.

- $\phi = x \in S_g$
  \[ A_\phi = \langle\{q, q'\}, \mathcal{F} \times 2^{\{x\}}, \{q'\}, \Delta\rangle \] with transitions
  \[
  \begin{align*}
  \langle a, 0 \rangle & \rightarrow q \\
  \langle g, 1 \rangle(q) & \rightarrow q' \\
  \langle g, 0 \rangle(q) & \rightarrow q \\
  \langle f, 0 \rangle(q, q) & \rightarrow q \\
  \langle f, 0 \rangle(q, q') & \rightarrow q' \\
  \langle f, 0 \rangle(q', q) & \rightarrow q'
  \end{align*}
  \]
accepts $L(x \in S_g)$ (scans for a single $g$-position with $\tau(x) = 1$).

- $\phi' = \exists x. \phi$
  Obtain $A_{\phi'}$ from $A_\phi$ by stripping $\tau(x)$:
  \[ A_{\phi'} = \langle\{q, q'\}, \mathcal{F}, \{q'\}, \Delta\rangle \]
  \[
  \begin{align*}
  a & \rightarrow q \\
  g(q) & \rightarrow q' \\
  g(q) & \rightarrow q \\
  g(q') & \rightarrow q' \\
  f(q, q) & \rightarrow q \\
  f(q, q') & \rightarrow q' \\
  f(q', q) & \rightarrow q'
  \end{align*}
  \]
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."
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*, \ldots tags.”
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*, \ldots tags.”

Definition: Tree (recall)

A (finite, ordered) *tree* is a non-empty, finite, prefix-closed set \( Pos \subseteq N^* \).
A *hedge automaton* (NHA) is a tuple \( \mathcal{A} = \langle Q, \Sigma, G, \Delta \rangle \), where:

- \( Q \) is a finite set of *states*;
- \( \Sigma \) a finite alphabet;
- \( G \subseteq Q \) are the *final states*;
- \( \Delta \) is a finite set of rules of the form
  \[ a(R) \rightarrow q \]
  for \( a \in \Sigma \), \( q \in Q \), and \( R \) a regular (word) language over \( Q \).
Hedge automata

Definition: (Bottom-up) hedge automaton

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

- $Q$ is a finite set of *states*;
- $\Sigma$ a finite alphabet;
- $G \subseteq Q$ are the *final states*;
- $\Delta$ is a finite set of rules of the form $a(R) \rightarrow q$ 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

\[
x(q_h^* q_b) \rightarrow q_x \quad h(\varepsilon) \rightarrow q_h \quad b(q_p^*) \rightarrow q_b \quad p(\varepsilon) \rightarrow q_p
\]

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 $a(R) \rightarrow q$ are called \textit{horizontal languages}.
- They are (finitely) represented by regular expressions or finite automata.
Semantics of hedge automata

Remark:

- The $R$ in $a(R) \rightarrow q$ 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 $A$ 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 $Pos \cap pN = \{p_1, \ldots, p_n\}$, there exists $a(R) \rightarrow q \in \Delta$ such that $t'(p_1) \cdots t'(p_n) \in R$.

Acceptance condition: $t'(\varepsilon) \in G$
Semantics of hedge automata

Remark:
- The $R$ in $a(R) \rightarrow q$ 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 $A$ 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 $Pos \cap pN = \{p_1, \ldots, p_n\}$, there exists $a(R) \rightarrow q \in \Delta$ such that $t'(p_1) \cdots t'(p_n) \in R$.

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

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

An NHA is . . .

- **complete** if for all $t \in T(\Sigma)$, $t \rightarrow^*_A q$ 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
Complete / normalized / deterministic HA

An NHA is . . .

- **complete** if for all $t \in T(\Sigma)$, $t \rightarrow^*_A q$ 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 of NHA

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

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

\[
R_{a,S} := \left( \bigcap_{q \in S} R'_{a,q} \right) \setminus \left( \bigcup_{q \notin S} R'_{a,q} \right)
\]
Bijective encoding of unranked into ranked trees

Let $\Sigma$ an alphabet; $\mathcal{F}_\Sigma := \{\@, 2\} \cup \{a(0) | a \in \Sigma\}$.

Define the coding $C_\@ (t) \in T(\mathcal{F}_\Sigma)$ of $t \in T(\Sigma)$ as

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

Example:

```
    x
   / \  \\
  h   b
 / \  /  \\
 p  p p  p  \\
```

$\Rightarrow$

```
    @
   /  \\
  @   @
 /    /  \\
@ x h @
/    /   \\
@ p @ p  \\
    /  \\
   @ p
  /  \\
 b p
```
Recognizing encoded trees

Theorem
A language \( L \subseteq T(\Sigma) \) is hedge-recognizable iff \( C_\ominus(L) \) 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:

- \( Q' = Q \cup \biguplus_{i=1}^n S_i \)
- \( \Delta' = \bigcup_{i=1}^n (\Delta_1^i \cup \Delta_2^i \cup \Delta_3^i) \)

\[ \begin{align*}
\Delta_1^i &= \{ a_i \rightarrow s_0^{(i)} \} \\
\Delta_2^i &= \{ \ominus(s, q) \rightarrow \delta_i(s, q) \mid s \in S_i, q \in Q \} \\
\Delta_3^i &= \{ s_f \rightarrow q_i \mid s_f \in F_i \}
\end{align*} \]
Example: NHA $\rightarrow$ NFTA

- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules
  
  $x(q_h^?q_b) \rightarrow q_x$  
  $h(\varepsilon) \rightarrow q_h$  
  $b(q_p^*) \rightarrow q_b$  
  $p(\varepsilon) \rightarrow q_p$

- Automaton for first rule:

- Single-state automata with $s_h, s_b, s_p$ for the other rules
Example: NHA → NFTA

- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules
  
  $x(q_h q_b) \rightarrow q_x \quad h(\varepsilon) \rightarrow q_h \quad b(q_p^*) \rightarrow q_b \quad p(\varepsilon) \rightarrow q_p$

- Automaton for first rule:

- Single-state automata with $s_h, s_b, s_p$ for the other rules
Example: NHA $\rightarrow$ NFTA

- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules
  
  $x(q_h q_b) \rightarrow q_x$  
  $h(\varepsilon) \rightarrow q_h$  
  $b(q_p^*) \rightarrow q_b$  
  $p(\varepsilon) \rightarrow q_p$

- Automaton for first rule:
- Single-state automata with $s_h, s_b, s_p$ for the other rules
Example: NHA $\rightarrow$ 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) \rightarrow q_x \quad h(\varepsilon) \rightarrow q_h \quad b(q_p^*) \rightarrow q_b \quad p(\varepsilon) \rightarrow q_p $  

- Automaton for first rule:

- Single-state automata with $s_h, s_b, s_p$ for the other rules
Example: NHA $\rightarrow$ NFTA

- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules
  $x(q_h q_b) \rightarrow q_x$  $h(\varepsilon) \rightarrow q_h$  $b(q_p^*) \rightarrow q_b$  $p(\varepsilon) \rightarrow q_p$

- Automaton for first rule:

- Single-state automata with $s_h, s_b, s_p$ for the other rules
Example: NHA $\rightarrow$ NFTA

- $Q := \{q_x, q_h, q_b, q_p\}$, $\Sigma = \{x, h, b, p\}$, $G := \{q_x\}$, and rules
  
  $x(q_h^?q_b) \rightarrow q_x$  
  $h(\varepsilon) \rightarrow q_h$  
  $b(q_p^*) \rightarrow q_b$  
  $p(\varepsilon) \rightarrow q_p$

- 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 \( C_\oplus(L) \) is recognizable.

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

Define \( \Delta_R := \{ \langle q_0, q_1, q_2 \rangle \mid \ominus(q_0, q_1) \rightarrow \Delta q_2 \} \)
and \( Out := G \cup \{ q \mid \exists q', q'' : \ominus(q', q) \rightarrow \Delta q'' \} \).
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
\[
\Delta' = \{ a(\mathcal{L}(A_{q,q'})) \rightarrow q' \mid a \rightarrow \Delta q, q' \in Out \}.
\]
Recognizing encoded trees

Theorem

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

\begin{itemize}
  \item NFTA \( \rightarrow \) NHA:
    \begin{align*}
      \text{Let } A = \langle Q, \mathcal{F}_\Sigma, G, \Delta \rangle \text{ an NFTA (without } \varepsilon \text{-moves).} \\
      \text{Define } \Delta_R := \{ \langle q_0, q_1, q_2 \rangle \mid \@ (q_0, q_1) \rightarrow \Delta q_2 \} \\
      \text{and } \text{Out} := G \cup \{ q \mid \exists q', q'' : \@ (q', q) \rightarrow \Delta q'' \}. \\
      \text{For } q \in Q, q' \in \text{Out}, \text{ let } A_{q, q'} := \langle Q, Q, q, \{ q' \}, \Delta_R \rangle \text{ a word automaton.}
    \end{align*}
  \end{itemize}

Construct NHA \( A' := \langle Q, \Sigma, G, \Delta' \rangle \), where

\[
\Delta' = \{ a(\mathcal{L}(A_{q, q'})) \rightarrow q' \mid a \rightarrow \Delta q, q' \in \text{Out} \}\]

Corollary

Hedge-recognizable languages are closed under boolean operations.
UTL = weak MSO(\textit{child, next}) interpreted over $\mathcal{M} = N^*$, where

- \textit{child}(x, y) \text{ iff } y = xi \text{ for some } i \in N
- \textit{next}(x, y) \text{ iff } \exists z, i : x = zi \land y = z(i + 1)
Unranked trees and logic

\[ \text{UTL} = \text{weak MSO}(\text{child}, \text{next}) \] interpreted over \( \mathcal{M} = \mathbb{N}^* \), where

- \( \text{child}(x, y) \) iff \( y = x_i \) for some \( i \in \mathbb{N} \)
- \( \text{next}(x, y) \) iff \( \exists z, i : x = z_i \land y = z(i + 1) \)

Further predicates can be defined from this:

- \( \text{right}(x, y) = "y \text{ is a right sibling of } x" \)
- \( \text{desc}(x, y) = "y \text{ is a descendant of } x" = "x \leq y" \)
Unranked trees and logic

UTL = weak MSO(\textit{child}, \textit{next}) interpreted over \( \mathcal{M} = \mathbb{N}^* \), where

- \( \textit{child}(x, y) \) iff \( y = x_i \) for some \( i \in \mathbb{N} \)
- \( \textit{next}(x, y) \) iff \( \exists z, i : x = z_i \land y = z(i + 1) \)

Further predicates can be defined from this:

- \( \textit{right}(x, y) = \text{“} y \text{ is a right sibling of } x \text{”} \)
- \( \textit{desc}(x, y) = \text{“} y \text{ is a descendant of } x \text{”} = \text{“} x \leq y \text{”} \)

Notions like \( \mathcal{L}(\phi) \) are defined in analogy with WS\( k \)S.

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 → NHA:
  Let $\phi$ be an UTL formula. Define $\phi'$ of WS2S s.t. $\mathcal{L}(\phi') = C@\mathcal{L}(\phi)$. 
UTL = NHA: Proof sketch

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

Define \textit{leftmost}(x, y) as

\[
\forall X : (x \in X \land \forall z, z' : (z \in X \land z' = z_1 \rightarrow z' \in X) \land \forall z : (z \in X \rightarrow z = x \lor (\exists z' : z' \in X \land z = z'1))) \rightarrow (y \in X \land \forall z \in X : z \in X \rightarrow z \leq y)
\]

(“y is the maximal position in $x1^*$”)
UTL = NHA: Proof sketch

UTL → NHA:
Let $\phi$ be an UTL formula. Define $\phi'$ of WS2S s.t. $\mathcal{L}(\phi') = \mathcal{C}_@ (\mathcal{L}(\phi))$.

Define $\text{leftmost}(x, y)$ as

$$\forall X : (x \in X \land \forall z, z' : (z \in X \land z' = z_1 \rightarrow z' \in X)$$
$$\land \forall z : (z \in X \rightarrow z = x \lor (\exists z' : z' \in X \land z = z'1)))$$
$$\rightarrow (y \in X \land \forall z \in X : z \in X \rightarrow z \leq y)$$

("$y$ is the maximal position in $x_1^*$")

Then $\text{child}$ and $\text{next}$ can be translated as follows:

$$\text{child}(x, y) \ := \ \exists z : \text{leftmost}(z, x) \land \text{leftmost}(z_2, y)$$
$$\text{next}(x, y) \ := \ \exists z : \text{leftmost}(z_{12}, x) \land \text{leftmost}(z_2, y)$$
NHA $\rightarrow$ 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 $\rightarrow$ WS$^k$ construction is (iii):

(iii): whenever the computation puts $q$ on an $a$-labelled position $p$, guess a run of the automaton for $\mathcal{R}_a$, $q$ over $p$ and its children.
UTL = NHA: Proof sketch

NHA → UTL:

Let $A$ 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 $A$, 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 → WS$k$S 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 \( t_1, t_2 \in T(F) \) ranked trees. Add a fresh symbol \(-\) to \( F_0 \) and let

\[
\mathcal{F}' := \{ \langle f, g \rangle(k) \mid f \in F_m, g \in F_n, k = \max\{m, n\} \}.
\]

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

- \( \text{Pos}_t = \text{Pos}_{t_1} \cup \text{Pos}_{t_2} \)
- for all \( p \in \text{Pos}_t \),
  \[
  t(p) = \begin{cases} 
  \langle f, g \rangle & \text{if } t \in \text{Pos}_{t_1} \cap \text{Pos}_{t_2}, t_1(p) = f, t_2(p) = g \\
  \langle f, - \rangle & \text{if } t \in \text{Pos}_{t_1} \setminus \text{Pos}_{t_2}, t_1(p) = f \\
  \langle -, g \rangle & \text{if } t \in \text{Pos}_{t_2} \setminus \text{Pos}_{t_1}, t_2(p) = g
  \end{cases}
  \]

Example:
\[
\langle f, f \rangle, \langle f, a \rangle, \langle a, - \rangle, \langle a, g \rangle, \langle -, g \rangle, \langle -, a \rangle
\]
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:

- \( \text{Pos}_t = \text{Pos}_{t_1} \cup \text{Pos}_{t_2} \)
- for all \( p \in \text{Pos}_t, \)

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

Example:

\[
\begin{array}{ccc}
\langle f, f \rangle & \Rightarrow & \langle f, a \rangle \quad \langle a, g \rangle \\
| & | & | \\
\langle a, - \rangle \quad \langle a, - \rangle & \Rightarrow & \langle -, g \rangle \\
| & | & | \\
\langle -, a \rangle
\end{array}
\]

\[
\begin{array}{ccc}
f & \quad f & \quad f \\
\quad \mid & \quad \mid & \quad \mid \\
a & \quad a & \quad a \\
\quad \mid & \quad \mid & \quad \mid \\
a & \quad a & \quad a
\end{array}
\]

\[
\begin{array}{ccc}
f & \quad g \\
\quad \mid & \quad \mid \\
a & \quad g \\
\quad \mid & \quad \mid \\
a & \quad a
\end{array}
\]
Tree relations

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

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

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

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

$$\left\{ \langle t, u \rangle \mid \exists n \geq 0, C \in C^n(\mathcal{F}), t_1, \ldots, t_n \in T(\mathcal{F}), u_1, \ldots, u_n \in T(\mathcal{F}), q_1, \ldots, q_n : t = C[t_1, \ldots, t_n] \wedge u = C[u_1, \ldots, u_n] \wedge \forall i : t_i \xrightarrow{A_1} q_i \xleftarrow{A_2} u_i \right\}$$
Relations between $\mathcal{R}_2$, $\mathcal{X}_2$, $\mathcal{T}_2$

### Propositions

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

### Proofs:

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

3. Let $A_i = \langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ (for $i = 1, 2$) be NFTA and let $R = \mathcal{L}(A_1) \times \mathcal{L}(A_2) \in \mathcal{X}_2$.

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

- $Q = (Q_1 \cup \{-\}) \times (Q_2 \cup \{-\})$
- for every $f \in \mathcal{F}_m$, $g \in \mathcal{F}_n$, $m \geq n$, $\neg (f = g = -)$
  $\Delta$ 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 $\mathcal{T}_2 \subseteq \mathcal{R}_2$

4. Let $\mathcal{G} = \langle A_1, A_2 \rangle$, $A_i = \langle Q_i, \mathcal{F}, G_i, \Delta_i \rangle$ (for $i = 1, 2$).

We construct NFTA $A' = \langle Q', \mathcal{F}', \{q_f\}, \Delta' \rangle$ with $L(A') = L(\mathcal{G})$.

Construct NFTA $A = \langle Q, \mathcal{F}', G, \Delta \rangle$ from $A_1, A_2$ as in previous proof. Then:

- $Q' = Q \cup \{q_f\}$
- $\Delta' = \Delta \cup \Delta_1 \cup \Delta_2$

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

Intuition:

$\Delta$ reads pairs of trees from $A_1, A_2$; $\Delta_1$ allows to plug pairs of subtrees into some context $C$; $\Delta_2$ reads the remaining context $C$. 
Proof of $X_2 \cup T_2 \subsetneq R_2$

5. Let $\mathcal{F} = \{ f(1), g(1), a \}$.
   Let $R = \{ \langle t_1, t_2 \rangle \mid \exists C \in C(\mathcal{F}), t \in T(\mathcal{F}) : t_1 = C[t] \land t_2 = C[f(t)] \}.$
   - $R \notin X_2$:
     $\langle a, f(a) \rangle \in R$ and $\langle f(a), f(f(a)) \rangle \in R$, but $\langle a, f(f(a)) \rangle \notin R$
Proof of $X_2 \cup T_2 \subset \mathbb{R}_2$

5. Let $\mathcal{F} = \{ f(1), g(1), a \}$.
Let $R = \{ \langle t_1, t_2 \rangle \mid \exists C \in C(\mathcal{F}), t \in T(\mathcal{F}) : t_1 = C[t] \land t_2 = C[f(t)] \}$.  

- $R \notin X_2$:  
  $\langle a, f(a) \rangle \in R$ and $\langle f(a), f(f(a)) \rangle \in R$, but $\langle a, f(f(a)) \rangle \notin R$

- $R \notin T_2$:  
  Suppose that $R$ is accepted by GTT $\langle A_1, A_2 \rangle$ with $n$ states in common. For all $i \geq 0$, let $q_i$ such that $g^i(a) \rightarrow_{A_1}^* q_i$ and $f(g^i(a)) \rightarrow_{A_2}^* q_i$.  
  Contradiction follows from pigeon-hole principle.
Proof of $X_2 \cup T_2 \subset R_2$

5. Let $F = \{f(1), g(1), a\}$.
   Let $R = \{\langle t_1, t_2 \rangle \mid \exists C \in C(F), t \in T(F) : t_1 = C[t] \land t_2 = C[f(t)]\}$.
   
   ▶ $R \notin X_2$:
   $\langle a, f(a) \rangle \in R$ and $\langle f(a), f(f(a)) \rangle \in R$, but $\langle a, f(f(a)) \rangle \notin R$
   
   ▶ $R \notin T_2$:
   Suppose that $R$ is accepted by GTT $\langle A_1, A_2 \rangle$ with $n$ states in common.
   For all $i \geq 0$, let $q_i$ such that $g^i(a) \rightarrow_{A_1}^* q_i$ and $f(g^i(a)) \rightarrow_{A_2}^* q_i$.
   Contradiction follows from pigeon-hole principle.
   
   ▶ $R \in R_2$:
   Let $A = \langle\{q_a, q_f, q_g, q\}, F, \{q\}, \Delta\rangle$ with:
   
   $\langle-, a\rangle \rightarrow q_a \quad \langle x, y\rangle(q_x) \rightarrow q_y \quad q_f \rightarrow q \quad \langle x, x\rangle(q) \rightarrow q$
   
   for $x, y \in \{f, g, a\}$
Closure properties

**Boolean closure**

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

**Transitive closure**

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

Proof: Let $\langle A_1, A_2 \rangle$ with states $Q_1, Q_2$ a GTT accepting $R$. We construct $\langle B_1, B_2 \rangle$ accepting $R^*$ by adding transitions to $A_1$ and $A_2$ 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. $t \rightarrow^* B_i q$ and $t \rightarrow^* B_j q'$ then add $q \rightarrow q'$ to $B_j$. 
Closure properties

Boolean closure

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

Transitive closure

If $R \in \mathcal{X}_2$, then $R^* \in \mathcal{X}_2$.

Proof: Let $\langle A_1, A_2 \rangle$ with states $Q_1, Q_2$ a GTT accepting $R$.
We construct $\langle B_1, B_2 \rangle$ accepting $R^*$ by adding transitions to $A_1$ and $A_2$
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.
  $t \rightarrow_{B_i}^* q$ and $t \rightarrow_{B_j}^* q'$
  then add $q \rightarrow q'$ to $B_j$. 
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 \( A_2 \) 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 $\langle A_1, A_2 \rangle$:

Adding $q_i \rightarrow q_i'$ to the right-hand side automaton achieves the objective.
Transitive closure:  \( R^* \subseteq L(\langle B_1, B_2 \rangle) \)

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

- \( i = 0 \): trivial
- \( i \rightarrow i + 1 \): 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 \( \langle B_1, B_2 \rangle \).
  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 \( pp_1, \ldots, pp_n \in P' \) consider the previous drawings.
  The case \( pp_1, \ldots, pp_n \in P \) and \( p \in P' \) is symmetric.
Transitive closure: $R^* \supseteq \mathcal{L}(\langle B_1, B_2 \rangle)$

Let $\langle B_1^i, B_2^i \rangle$ 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 $B_1$, 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 $n \geq 0$, $C \in C^n(\mathcal{F})$ etc such that $t = C[t_1, \ldots, t_n]$, $u = C[u_1, \ldots, u_n]$ and $\forall k : t_k \xrightarrow{B_1^{i+1}} q_k B_2^{i+1} u_k$.
2. Suppose $t_k = C'[t'] \xrightarrow{B_1^{i+1}} C'[q] \xrightarrow{B_2^{i+1}} C'[q'] \xrightarrow{B_1^{i+1}} q_k$ for some $k, C', t'$.
3. There must be some $v \in T(\mathcal{F})$ with $v \xrightarrow{B_2^i} q$ and $v \xrightarrow{B_1^i} q'$.
4. From (2) et (3) we have $C'[v] \xrightarrow{B_1^{i+1}} q_k$.
5. Replacing $t_k$ by $C'[v]$ in (1) we get $\langle t[t'/v], u \rangle \in \mathcal{L}(\langle B_1^{i+1}, B_2^{i+1} \rangle)$ 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), $t' \xrightarrow{B_1^{i+1}} q B_2^i v$, 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
<html><head><meta charset="UTF-8"/>
<title>My web page</title></head>
<body><p>Bonne année !</p></body></html>
```
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 année !</p></body></html>
```
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:

```xml
<html><head><meta charset="UTF-8"/>
<title>My web page</title></head>
<body><p>Bonne année !</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 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 defines a (restricted) subclass of XML languages. Essentially, defines a regular language of child tags for each tag type.

Example (from Wikipedia):

```xml
<!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))\}
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.*
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).
Definition: Marked RE

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

Example: $e = (ab|ac)$, then $\bar{e} = (a_1b_2|a_3c_4)$
Deterministic regular expressions

Definition: Marked RE
Let $e$ be a RE over $\Sigma$. 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 $\bar{e} = (a_1b_2|a_3c_4)$

Definition: Deterministic RE
Let $e$ a RE over $\Sigma$. 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: $e = (ab|ac)$, $\bar{e} = (a_1b_2|a_3b_4)$, not deterministic because $a_1b_2, a_3b_4 \in L(\bar{e})$
Parsing deterministic 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 \( e \).
Parsing deterministic 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 can define more expressive XML languages.

Example:

```xml
<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):

```xml
<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>
```