Tree Automata and Applications

M1 course, 2024/2025

Mostly based on slides by Stefan Schwoon

Organization

Schedule

- Exercises: Wednesday 8:30 – 10:30 (Luc Lapointe)
- Lectures: Wednesday 10:45 – 12:45 (Laurent Doyen)

Organization

Schedule

- Exercises: Wednesday 8:30 – 10:30 (Luc Lapointe)
- Lectures: Wednesday 10:45 – 12:45 (Laurent Doyen)

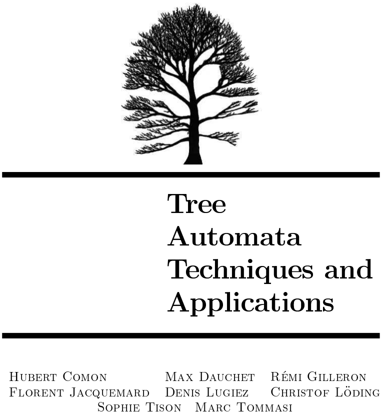
Assessment

- DM or CC (*to be specified by Luc*)
- Final Exam: 2h, 15th January 10am
- First session: DM/CC + Exam (50/50)
- Second session: DM/CC + Repeat Exam (50/50)

Material

Course material

- Website: lecturer's homepage + Wiki MPRI, course 1-18 (exercise sheets, slides, former exams)
- Main reference: H. Comon et al. Tree Automata Techniques and Applications, 2008.



Material

Course material

- Website: lecturer's homepage + Wiki MPRI, course 1-18 (exercise sheets, slides, former exams)
- Main reference: H. Comon et al. Tree Automata Techniques and Applications, 2008.

Other relevant resources

- C. Löding, W. Thomas. Automata on finite trees. Handbook of Automata Theory (I.), pp. 235-264, 2021.
- L. Doyen. Top-Down Complementation of Automata on Finite Trees. IPL 187:106499, 2025.

Motivation

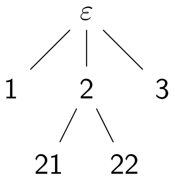
Context

1. Natural extension of formal languages and automata on words
2. Connection with Logic & Games
3. Treatment of tree-like data structures: parse trees, XML documents
4. Applications e.g. in compiler construction, formal verification

Trees

We consider *finite* *ordered* *ranked* trees. Let $\mathbb{N}_0 = \mathbb{N} \setminus \{0\}$

- *finite* set nodes (positions), denoted by $Pos \subseteq \mathbb{N}_0^*$ (with $\varepsilon \in Pos$)
- *ordered* : internal nodes have children $1, \ldots, n$
- *ranked* : number of children fixed by node's label



Definition: Tree

A (finite, ordered) *tree* is a nonempty, finite, prefix-closed set $Pos \subseteq \mathbb{N}_0^*$ such that $w \cdot (i+1) \in Pos$ implies $w \cdot i \in Pos$ for all $w \in \mathbb{N}^*$, $i \in \mathbb{N}_0$.

- In the sequel, we write wi instead of $w \cdot i$
- prefix-closed: $wi \in Pos$ implies $w \in Pos$

Ranked Trees

Ranked symbols

Ranked alphabet $\mathcal{F}$: finite set of symbols, each with an *arity* $0, 1, \ldots$
Denote by $\mathcal{F}_i$ the symbols of arity i (hence $\mathcal{F} := \bigcup_i \mathcal{F}_i$).

- arity 0: constants
- arity ≥ 1: functions (unary, binary, etc.)

Notation (example): $\mathcal{F} = \{f(2), g(1), a, b\}$
Let $\mathcal{X}$ denote a set of variables (of arity 0), disjoint from the other symbols.

Ranked Trees

Ranked symbols

Ranked alphabet $\mathcal{F}$: finite set of symbols, each with an *arity* $0, 1, \ldots$
Denote by $\mathcal{F}_i$ the symbols of arity i (hence $\mathcal{F} := \bigcup_i \mathcal{F}_i$).

- arity 0: constants
- arity ≥ 1: functions (unary, binary, etc.)

Notation (example): $\mathcal{F} = \{f(2), g(1), a, b\}$
Let $\mathcal{X}$ denote a set of variables (of arity 0), disjoint from the other symbols.

Definition: Ranked tree

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

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

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 write $T(\mathcal{F}) := T(\mathcal{F}, \varnothing)$, called the set of *ground terms*.
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 write $T(\mathcal{F}) := T(\mathcal{F}, \varnothing)$, called the set of *ground terms*.
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.

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 write $T(\mathcal{F}) := T(\mathcal{F}, \varnothing)$, called the set of *ground terms*.
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 use 'terms' and 'trees' interchangeably (obvious bijection).

Height and Size

Definition

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

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

Subterms / subtrees

Definition: Subtree

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

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

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

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

Substitutions and Context

Definition: Substitution

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

Substitutions and Context

Definition: Substitution

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

Definition: Context

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

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

Tree automata

Basic idea: Extension of finite automata from words to trees

Direct extension of automata theory when words seen as unary terms:

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

Finite automaton: labels every prefix of a word with a state.

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

Two variants: bottom-up vs top-down labelling

Tree automata

Basic idea: Extension of finite automata from words to trees

Direct extension of automata theory when words seen as unary terms:

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

Finite automaton: labels every prefix of a word with a state.

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

Two variants: bottom-up vs top-down labelling

Basic results (preview)

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

Bottom-up automata

Definition: (Bottom-up tree automata)

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

- Q is a finite set of *states*;
- $\mathcal{F}$ a finite ranked alphabet;
- $G \subseteq Q$ are the *final states*;
- Δ is a finite set of rules of the form

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

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

Bottom-up automata

Definition: (Bottom-up tree automata)

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

- Q is a finite set of *states*;
- $\mathcal{F}$ a finite ranked alphabet;
- $G \subseteq Q$ are the *final states*;
- Δ is a finite set of rules of the form

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

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

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

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

Move relation and Recognized language

$\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$

Move relation

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

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

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

Move relation and Recognized language

$\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$

Move relation

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

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

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

Recognized Language

- A tree t is *accepted* by $\mathcal{A}$ if $t \rightarrow^*_{\mathcal{A}} q$ for some $q \in G$.
- $\mathcal{L}(\mathcal{A})$ denotes the set of trees accepted by $\mathcal{A}$.
- L is *recognizable* if $L = \mathcal{L}(\mathcal{A})$ for some NFTA $\mathcal{A}$.

NFTA with ε-moves

Definition:

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

Semantics: allow to re-label a position from q to q': $C[q] \to_{\mathcal{A}} C[q']$.

Equivalence of ε-NFTA

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

Proof (sketch): construct the rules of $\mathcal{A}'$ by a saturation procedure.
Initialize $\Delta' = \Delta$ and apply:

$$\frac{f(q_1, \ldots, q_n) \to q \in \Delta' \quad q \to q' \in \Delta}{f(q_1, \ldots, q_n) \to q' \in \Delta'}$$

Deterministic, complete, and reduced NFTA

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

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

Top-down tree automata

Definition

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

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

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 $\mathcal{A} = \langle Q, \mathcal{F}, I, \Delta \rangle$, where $Q, \mathcal{F}$ are as in NFTA, $I \subseteq Q$ is a set of *initial states*, and Δ contains rules of the form

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

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

Move relation

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

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

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

From top-down to bottom-up

Theorem (T-NFTA = NFTA)

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

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

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

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

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

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

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

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

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

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

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

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

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

Run (Computation tree)

$\mathcal{A} = \langle Q, \mathcal{F}, I, \Delta \rangle$

Definition: (Run)

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

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

Run (Computation tree)

$\mathcal{A} = \langle Q, \mathcal{F}, I, \Delta \rangle$

Definition: (Run)

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

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

Recognized Language

- A run t' is initialized (or accepting) if $t'(\varepsilon) \in I$.
- A tree t is *accepted* by $\mathcal{A}$ if there exists an initialized run of $\mathcal{A}$ on t.

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

- Notation: $\mathcal{A}_q = \langle Q, \mathcal{F}, \{q\}, \Delta \rangle$ and $L_q(\mathcal{A}) = L(\mathcal{A}_q)$, so $L(\mathcal{A}) = \bigcup_{q \in I} L_q(\mathcal{A})$.

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:

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

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

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:

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

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

DFTA with accessible states

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

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:

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

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

DFTA with accessible states

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

Deterministic top-down are less powerful

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

A pumping lemma for tree languages

Lemma

Let L be recognizable. Then there exists k such that for all $t \in L$ with $\mathcal{H}(t) > k$, there exist contexts $C, D \in T(\mathcal{F}, \{x\})$ 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 $C[D^n[u]] \in L$.

A pumping lemma for tree languages

Lemma

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

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

Proof: Let k be the number of states of an NFTA $\mathcal{A}$ recognizing L.
In an accepting run on a tree $t \in L$, there exist two positions p, pp' ($p' \neq \varepsilon$) labelled by the same state q.
Let $C = t[x]_p$, $D = t_{|p}[x]_{p'}$, and $u = t_{|pp'}$, thus $t = C[D[u]]$.
The accepting run on t entails:

$$u \to^*_{\mathcal{A}} q, D[q] \to^*_{\mathcal{A}} q, \text{ and } C[q] \to^*_{\mathcal{A}} q_f, \text{ for some final state } q_f.$$

Therefore, $D^n[q] \to^*_{\mathcal{A}} q$ for all $n \geq 0$ (by induction, where $D^0[q] := q$) and

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

A pumping lemma for tree languages

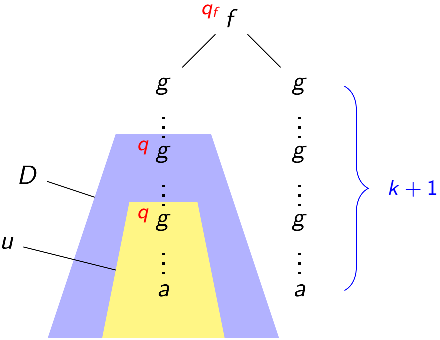
Mostly used in the form:

Lemma

If for all k,
  there exists $t \in L$ with $\mathcal{H}(t) > k$,
    for all contexts $C, D \in T(\mathcal{F}, \{x\})$ and $u \in T(\mathcal{F})$ such that
    $t = C[D[u]]$ and D is non-trivial,
      there exists $n \geq 0 : C[D^n[u]] \notin L$,
then L is not recognizable.

Illustration of pumping lemma

Let $L = \{ f(g^i(a), g^i(a)) \mid i \geq 0 \}$ for $\mathcal{F} = \{f(2), g(1), a\}$.
Given k, let $t = f(g^k(a), g^k(a))$.



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

Closure properties

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 $T(\mathcal{F}) \setminus L$.

Proof hint: uniqueness of the run on the input tree.

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 $T(\mathcal{F}) \setminus L$.

Proof hint: uniqueness of the run on the input tree.

Union (juxtapose)

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

Cross-product construction

Direct intersection

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

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

Cross-product construction

Direct intersection

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

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

Remarks:

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

Single tree

Singleton Language

Given a tree $t : Pos \to \mathcal{F}$, the language $L = \{t\}$ is recognized by $\mathcal{A}_t = \langle Q, \mathcal{F}, I, \Delta \rangle$ where:

- $Q = Pos$
- $I = \{\varepsilon\}$
- $\Delta = \{f(p1, \ldots, pn) \to p \mid f = t(p) \in \mathcal{F}_n\}$

Single tree

Singleton Language

Given a tree $t : Pos \to \mathcal{F}$, the language $L = \{t\}$ is recognized by $\mathcal{A}_t = \langle Q, \mathcal{F}, I, \Delta \rangle$ where:

- $Q = Pos$
- $I = \{\varepsilon\}$
- $\Delta = \{f(p1, \ldots, pn) \to p \mid f = t(p) \in \mathcal{F}_n\}$

Remark: $\mathcal{A}_t$ is deterministic.

Proof: Show $t' \to^*_{\mathcal{A}_t} p$ iff $t' = t_{|p}$

Tree homomorphism

Definition

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

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

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

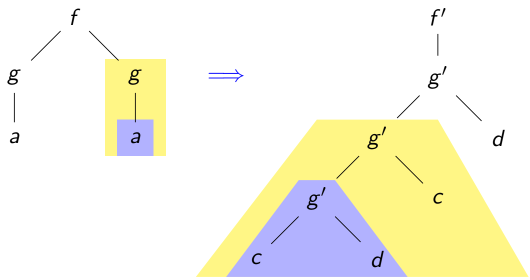
Intuition:

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

Examples

Example

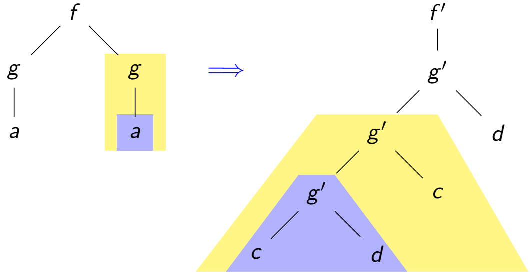
- $\mathcal{F} = \{f(2), g(1), a\}$, $\mathcal{F}' = \{f'(1), g'(2), c, d\}$
- $h(f) = f'(g'(x_2, d))$, $h(g) = g'(x_1, c)$, $h(a) = g'(c, d)$



Examples

Example

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



Example (ternary to binary tree)

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

Properties of homomorphisms

A homomorphism h is

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

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

Properties of homomorphisms

A homomorphism h is

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

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

Non-linear homomorphisms do not preserve recognizability

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

Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

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

Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

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

Illustrating example:

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

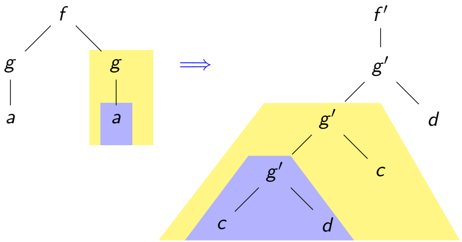
Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

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

Illustrating example:

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



Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

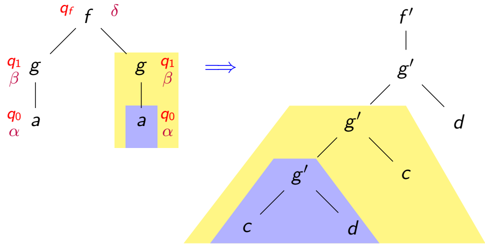
Let $L \subseteq T(\mathcal{F})$ be recognizable and $h : \mathcal{F} \to \mathcal{F}'$ a linear tree homomorphism. Then $h(L)$ is recognizable.

Illustrating example:

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

Run on $\mathcal{A}$

Rules used to produce states



Linear homomorphisms

Theorem: Linear homomorphisms preserve recognizability

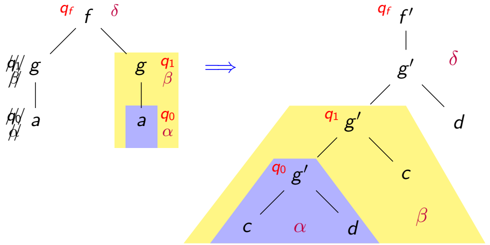
Let $L \subseteq T(\mathcal{F})$ be recognizable and $h : \mathcal{F} \to \mathcal{F}'$ a linear tree homomorphism. Then $h(L)$ is recognizable.

Illustrating example:

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

Run on $\mathcal{A}$

Rules used to produce states

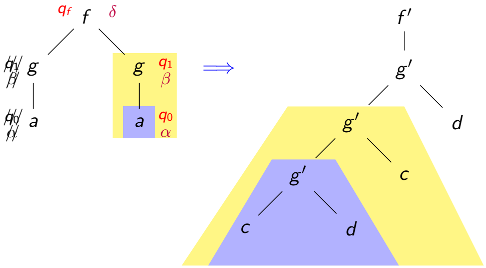


Construct automaton for $h(L)$ preserving state labels from $\mathcal{A}$ + Guess the rules.

Automaton construction for $h(L)$

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

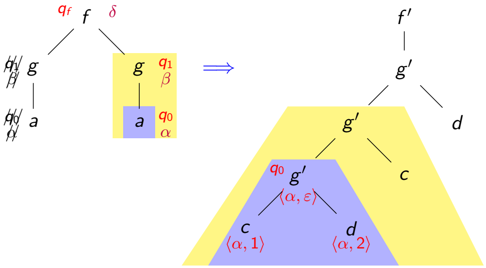
- $Q' := Q \cup \{ \langle r, p \rangle \mid r \in \Delta, \exists f \in \mathcal{F} : r = f(\dots) \to \dots, p \in Pos_{h(f)} \}$;
- $\Delta' = \bigcup_{r \in \Delta} \Delta'_r$ where for each transition $r : f(q_1, \dots, q_n) \to q$ in Δ, the set Δ'_r contains, for all positions $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \dots, \langle r, pk \rangle) \to \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'_k$
  - $q_i \to \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \to q$



Automaton construction for $h(L)$

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

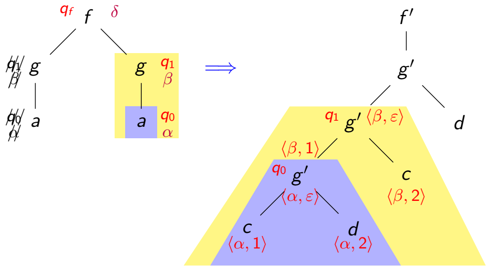
- $Q' := Q \cup \{ \langle r, p \rangle \mid r \in \Delta, \exists f \in \mathcal{F} : r = f(\ldots) \to \ldots, p \in Pos_{h(f)} \}$;
- $\Delta' = \bigcup_{r \in \Delta} \Delta'_r$ where for each transition $r : f(q_1, \ldots, q_n) \to q$ in Δ, the set Δ'_r contains, for all positions $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \ldots, \langle r, pk \rangle) \to \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'_k$
  - $q_i \to \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \to q$



Automaton construction for $h(L)$

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

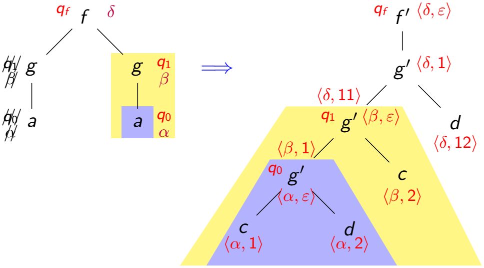
- $Q' := Q \cup \{ \langle r, p \rangle \mid r \in \Delta, \exists f \in \mathcal{F} : r = f(\dots) \to \dots, p \in Pos_{h(f)} \}$;
- $\Delta' = \bigcup_{r \in \Delta} \Delta'_r$ where for each transition $r : f(q_1, \dots, q_n) \to q$ in Δ, the set Δ'_r contains, for all positions $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \dots, \langle r, pk \rangle) \to \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'_k$
  - $q_i \to \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \to q$



Automaton construction for $h(L)$

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

- $Q' := Q \cup \{ \langle r, p \rangle \mid r \in \Delta, \exists f \in \mathcal{F} : r = f(\dots) \to \dots, p \in Pos_{h(f)} \}$;
- $\Delta' = \bigcup_{r \in \Delta} \Delta'_r$ where for each transition $r : f(q_1, \dots, q_n) \to q$ in Δ, the set Δ'_r contains, for all positions $p \in Pos_{h(f)}$:
  - $f'(\langle r, p1 \rangle, \dots, \langle r, pk \rangle) \to \langle r, p \rangle$ if $h(f)(p) = f' \in \mathcal{F}'_k$
  - $q_i \to \langle r, p \rangle$ if $h(f)(p) = x_i$
  - $\langle r, \varepsilon \rangle \to q$



Correctness

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

Correctness

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

- $h(L) \subseteq \mathcal{L}(\mathcal{A}')$:
  For all $t \in T(\mathcal{F})$, prove that $t \rightarrow^*_{\mathcal{A}} q$ implies $h(t) \rightarrow^*_{\mathcal{A}'} q$,

Correctness

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

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

Correctness

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

- $h(L) \subseteq \mathcal{L}(\mathcal{A}')$:
  For all $t \in T(\mathcal{F})$, prove that $t \rightarrow^*_{\mathcal{A}} q$ implies $h(t) \rightarrow^*_{\mathcal{A}'} q$,
  by structural induction over t.
- $h(L) \supseteq \mathcal{L}(\mathcal{A}')$:
  For all $t' \in T(\mathcal{F}')$, prove that if $t' \rightarrow^*_{\mathcal{A}'} q \in Q$,
  then there exists $t \in T(\mathcal{F}) \cap h^{-1}(t')$ with $t \rightarrow^*_{\mathcal{A}} q$,

Correctness

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

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

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

Correctness

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

Correctness

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

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

Correctness

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

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

  - Base case: $t = a$ (leaf) where $a \in \mathcal{F}_0$ (constant)
    $a \rightarrow^*_{\mathcal{A}} q \iff \underbrace{a \rightarrow q}_{r} \in \Delta$
    Then $h(a) \rightarrow^*_{\mathcal{A}'} q$ using rules in Δ'_r (single tree)
    Note: t_a is a ground term, rules $q_i \rightarrow \langle r, p \rangle$ not used

Correctness

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

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

  - Base case: $t = a$ (leaf) where $a \in \mathcal{F}_0$ (constant)
    $a \rightarrow^*_{\mathcal{A}} q \iff \underbrace{a \rightarrow q}_{r} \in \Delta$
    Then $h(a) \rightarrow^*_{\mathcal{A}'} q$ using rules in Δ'_r (single tree)
    Note: t_a is a ground term, rules $q_i \rightarrow \langle r, p \rangle$ not used

  - Inductive case: $t = f(u_1, \ldots, u_n)$
    $t \rightarrow^*_{\mathcal{A}} \underbrace{f(q_1, \ldots, q_n) \rightarrow q}_{r \in \Delta}$ and $u_i \rightarrow^*_{\mathcal{A}} q_i$ $(i = 1, \ldots, n)$
    Then $h(t) = h(f)\{x_1 \leftarrow h(u_1), \ldots, x_n \leftarrow h(u_n)\}$
    and by induction hypothesis $h(u_i) \rightarrow^*_{\mathcal{A}'} q_i$, so $h(t) \rightarrow^*_{\mathcal{A}'} h(f)(q_1, \ldots, q_n)$
    To show: $h(f)(q_1, \ldots, q_n) \rightarrow^*_{\mathcal{A}'} q$ using rules in Δ'_r (single tree)

Correctness

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

- $\mathcal{L}(\mathcal{A}') \subseteq h(L)$:
  For all $t' \in T(\mathcal{F}')$, show that if $t' \rightarrow^*_{\mathcal{A}'} q \in Q$,
  then there exists $t \in T(\mathcal{F})$ such that $t \rightarrow^*_{\mathcal{A}} q$ and $h(t) = t'$,
  by induction on number of states (of Q) in the runs of $\mathcal{A}'$ (with ε-transitions removed) on t' corresponding to $t' \rightarrow^*_{\mathcal{A}'} q$.

Correctness

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

- $\mathcal{L}(\mathcal{A}') \subseteq h(L)$:
  For all $t' \in T(\mathcal{F}')$, show that if $t' \rightarrow^*_{\mathcal{A}'} q \in Q$,
  then there exists $t \in T(\mathcal{F})$ such that $t \rightarrow^*_{\mathcal{A}} q$ and $h(t) = t'$,
  by induction on number of states (of Q) in the runs of $\mathcal{A}'$ (with ε-transitions removed) on t' corresponding to $t' \rightarrow^*_{\mathcal{A}'} q$.

  - Base case: $t' \rightarrow^*_{\mathcal{A}'} q$, with no intermediate state from Q
    Since Δ'_r are disjoint, only rules from Δ'_r for a single r are used.
    (no variable in t' – rules $q_i \rightarrow \langle r, p \rangle$ not used)
    Let $r = f(q_1, \ldots, q_n) \rightarrow q$.
    Then $t' = h(f)$ (single tree) and we construct $t = f(u_1, \ldots, u_n)$ where $u_i \rightarrow^*_{\mathcal{A}} q_i$ (why is it possible ?).

Correctness

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

- $\mathcal{L}(\mathcal{A}') \subseteq h(L)$:
  For all $t' \in T(\mathcal{F}')$, show that if $t' \to^*_{\mathcal{A}'} q \in Q$,
  then there exists $t \in T(\mathcal{F})$ such that $t \to^*_{\mathcal{A}} q$ and $h(t) = t'$,
  by induction on number of states (of Q) in the runs of $\mathcal{A}'$ (with ε-transitions removed) on t' corresponding to $t' \to^*_{\mathcal{A}'} q$.

  - Inductive case: $$t' \to^*_{\mathcal{A}'} v\{x_1 \leftarrow q_1, \ldots, x_n \leftarrow q_n\} \to^*_{\mathcal{A}'} \underbrace{v\{x_1 \leftarrow \langle r_1, p_1\rangle, \ldots, x_n \leftarrow \langle r_n, p_n\rangle\} \to^*_{\mathcal{A}'} q}_{\text{no intermediate state from } Q}$$
    where v is a linear term in $T(\mathcal{F}', \mathcal{X})$
    Hence $t' = v\{x_1 \leftarrow u'_1, \ldots, x_n \leftarrow u'_n\}$ where $u'_i \to^*_{\mathcal{A}'} q_i$ $(i = 1, \ldots, n)$
    (why is it possible ?)
    Also $r_1 = r_2 = \cdots = r_n = r = f(q_1, \ldots, q_n) \to q$ and $v = h(f)$ (single tree)
    By induction hyp., there exist $u_i \to^*_{\mathcal{A}} q_i$ with $h(u_i) = u'_i$ and we construct $t = f(u_1, \ldots, u_n)$ and show that $h(t) = t'$ and $t \to^*_{\mathcal{A}} q$.

Inverse tree homomorphisms

Theorem: Inverse homomorphisms preserve recognizability

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

Inverse tree homomorphisms

Theorem: Inverse homomorphisms preserve recognizability

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

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

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

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

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

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

Tree languages and context-free languages

Frontier

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

Frontier

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

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

Leaf languages

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

Tree languages and context-free languages

Frontier

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

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

Leaf languages

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

Proof (idea):

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

Regular Expressions

Words (alphabet Σ)

- $\emptyset, \varepsilon, a$ ($a \in \Sigma$)
- union, concatenation, iteration (Kleene star)

Trees (ranked alphabet $\mathcal{F}$)

- No empty tree
- n-ary symbols
- concatenation, iteration ?

Regular Expressions

Words (alphabet Σ)

- $\varnothing, \varepsilon, a$ ($a \in \Sigma$)
- union, concatenation, iteration (Kleene star)

Trees (ranked alphabet $\mathcal{F}$)

- No empty tree
- n-ary symbols
- concatenation, iteration ?

  $\rightsquigarrow$ use placeholders

Let $\mathcal{K} = \{\Box_1, \Box_2, \dots\}$ be a set of placeholders (symbols of arity 0).

$T(\mathcal{F}, \mathcal{K})$: set of all terms over ranked alphabet $\mathcal{F} \cup \mathcal{K}$.

Placeholders

Placeholder substitution

- Substitution: $\{\Box \leftarrow L\}$ where L is a tree language
- Can replace different occurrences of $\Box$ by different elements of L

Semantics

Based on semantics of $t\{\Box \leftarrow L\}$, by structural induction on $t \in T(\mathcal{F}, \mathcal{K})$:

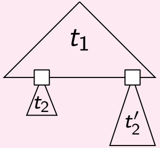
- $t = a$ of arity 0: $a\{\Box \leftarrow L\} = L$ if $a = \Box$ (otherwise $\{a\}$)
- $t = f$ of arity $n > 0$:
  $f(t_1, \ldots, t_n)\{\Box \leftarrow L\} = \{f(s_1, \ldots, s_n) \mid s_i \in t_i\{\Box \leftarrow L\}, 1 \leq i \leq n\}$
- $L_1\{\Box \leftarrow L\} = \{t\{\Box \leftarrow L\} \mid t \in L_1\}$

Abbreviation: $\{\Box_1 \leftarrow L_1, \ldots, \Box_n \leftarrow L_n\} = \{\Box_1 \leftarrow L_1\} \circ \cdots \circ \{\Box_n \leftarrow L_n\}$ $(L_i \subseteq T(\mathcal{F}))$

Concatenation - Iteration

Concatenation

$L_1 \cdot_\square L_2 = \bigcup_{t \in L_1} t\{\square \leftarrow L_2\}$

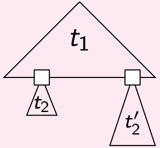


$t_1 \in L_1$
$t_2, t'_2 \in L_2$

Concatenation - Iteration

Concatenation

$L_1 \cdot_\square L_2 = \bigcup_{t \in L_1} t\{\square \leftarrow L_2\}$



$t_1 \in L_1$
$t_2, t'_2 \in L_2$

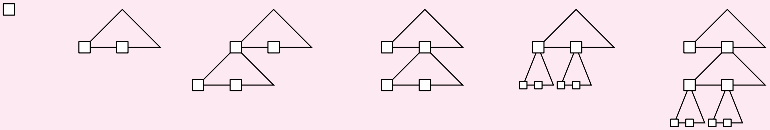
Iteration

$L^{*_\square} = \bigcup_{k\in\mathbb{N}} L^k$ where $L^0 = \{\square\}$ and $L^{k+1} = L^k \cdot_\square (L \cup L^0)$

Note: $\square \in L^{*_\square}$ for all L.

Kleene plus: $L^{+_\square} = \bigcup_{k>0} L^k$.

For $L = \{f(\square, \square, a)\}$:



Regular Expressions

Syntax

A regular expression is obtained by the following grammar:

$$E := \varnothing \mid f \mid E_1 + E_2 \mid E_1 \cdot_{\Box} E_2 \mid E^{*_{\Box}}$$

$f \in \mathcal{F}, \Box \in \mathcal{K}$

Semantics

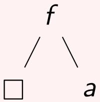
- $[\![\varnothing]\!] = \varnothing$
- $[\![f]\!] = \{f(\Box_1, \ldots, \Box_n)\}$ $(f \in \mathcal{F}_n)$
- $[\![E_1 + E_2]\!] = [\![E_1]\!] \cup [\![E_2]\!]$
- $[\![E_1 \cdot_{\Box} E_2]\!] = [\![E_1]\!] \cdot_{\Box} [\![E_2]\!]$
- $[\![E^{*_{\Box}}]\!] = [\![E]\!]^{*_{\Box}}$

A tree language L is regular if $L = [\![E]\!]$ for some regular expression E.

Shortcut: $f(E_1, \ldots, E_n) = f \cdot_{\Box_1} E_1 \ldots \cdot_{\Box_i} E_i \ldots \cdot_{\Box_n} E_n$

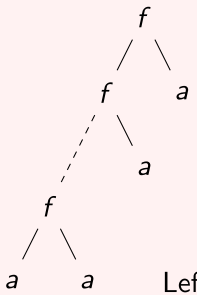
Regular Expressions - Examples

$E = f(\square, a)^{*\square} \cdot_{\square} a$



Regular Expressions - Examples

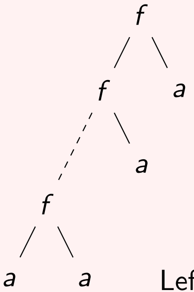
$E = f(\Box, a)^{*\Box} \cdot_{\Box} a$



Left "comb"

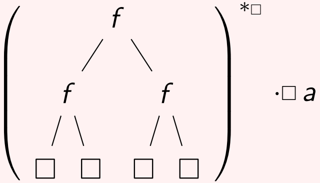
Regular Expressions - Examples

$E = f(\square, a)^{*\square} \cdot_\square a$



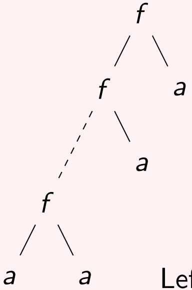
Left "comb"

$E = f(f(\square, \square), f(\square, \square))^{*\square} \cdot_\square a$



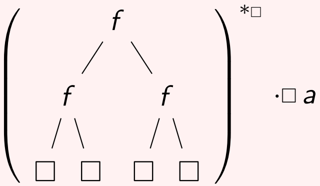
Regular Expressions - Examples

$E = f(\square, a)^{*_\square} \cdot_\square a$



Left "comb"

$E = f(f(\square,\square), f(\square,\square))^{*_\square} \cdot_\square a$



All branches are of even length

Kleene Theorem for Tree Languages

Theorem (RegExp $\equiv$ NFTA)

L is regular iff L is recognizable by an NFTA.

Proof ($\Rightarrow$)

By induction over the structure of regular expressions:

- Base case: $\varnothing$ and $\{f(\square_1, \ldots, \square_n)\}$ are finite languages, hence recognizable.
- Inductive case: given $\mathcal{A}_i = \langle Q_i, \mathcal{F} \cup \mathcal{K}, G_i, \Delta_i \rangle$ NFTAs recognizing $L(\mathcal{A}_i) = [\![E_i]\!]$ $(i = 1, 2)$,
  - $[\![E_1 + E_2]\!]$ is recognized by $\langle Q_1 \uplus Q_2, \mathcal{F} \cup \mathcal{K}, G_1 \cup G_2, \Delta_1 \cup \Delta_2 \rangle$
  - $[\![E_1 \cdot_\square E_2]\!]$ is recognized by $\langle Q_1 \uplus Q_2, \mathcal{F} \cup \mathcal{K}, G_1, \Delta \rangle$ where $\Delta = \Delta_1 \setminus \{\square \to q \mid q \in Q_1\} \cup \Delta_2 \cup \{f(q_1, \ldots, q_n) \to q \mid \exists q' \in G_2 : f(q_1, \ldots, q_n) \to q' \in \Delta_2 \text{ and } \square \to q \in \Delta_1\}$
  - $[\![E_1^{+\square}]\!]$ is recognized by $\langle Q_1, \mathcal{F} \cup \mathcal{K}, G_1, \Delta \rangle$ where $\Delta = \Delta_1 \cup \{f(q_1, \ldots, q_n) \to q \mid \exists q' \in G_1 : f(q_1, \ldots, q_n) \to q' \in \Delta_1 \text{ and } \square \to q \in \Delta_1\}$

Kleene Theorem for Tree Languages

Theorem (RegExp $\equiv$ NFTA)

L is regular iff L is recognizable by an NFTA.

Proof ($\Leftarrow$)

Given NFTA $\mathcal{A} = \langle Q, \mathcal{F}, G, \Delta \rangle$, we construct a regular expression E such that $[\![E]\!] = L(\mathcal{A})$.

- Let $\mathcal{K} = \{\Box_q \mid q \in Q\}$ be a set of placeholders.
- Let $\mathcal{A}' = \langle Q, \mathcal{F} \cup \mathcal{K}, G, \Delta' \rangle$ where $\Delta' = \Delta \cup \{\Box_q \to q \mid q \in Q\}$. Then $L(\mathcal{A}') \cap T(\mathcal{F}) = L(\mathcal{A})$.
- For $q \in Q$ and $N, K \subseteq Q$, let $L(q, N, K)$ be the set of all trees $t \in T(\mathcal{F} \cup \{\Box_q \mid q \in K\})$ having a run r of $\mathcal{A}'$ such that:
  - $r(\varepsilon) = q$
  - $r(p) \in N$ for all positions $p \neq \varepsilon$ such that $t(p) \in \mathcal{F}$
  - (note: the leaves of t are labeled by $\mathcal{F}_0 \cup \{\Box_q \mid q \in K\}$)

Kleene Theorem for Tree Languages

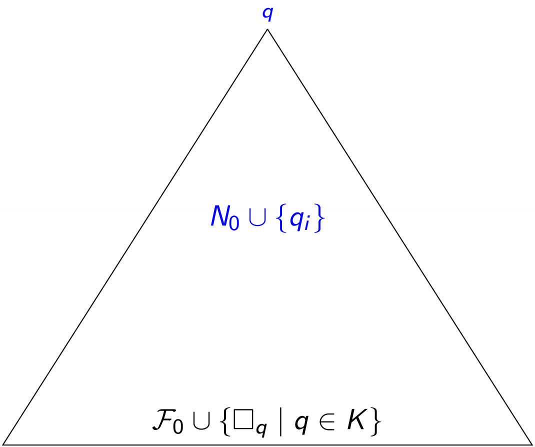
Proof ($\Leftarrow$) (cont'd)

Since $L(\mathcal{A}) = \bigcup_{q \in G} L(q, Q, \varnothing)$, showing that all sets $L(q, N, K)$ are regular is sufficient. By induction on $|N|$:

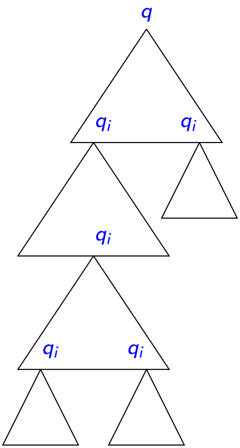
- Base case: $N = \varnothing$, then all languages $L(q, \varnothing, K)$ are finite (trees of height at most 1), hence regular.
- Inductive case: let $N = N_0 \cup \{q_i\}$ $(q_i \notin N_0)$,
  Given a run r on $t \in L(q, N, K)$, decompose t into subtrees with:
  (1) root labeled by q_i in r (by q in topmost subtree),
  (2a) internal nodes are either labeled by states of N_0 in r,
  (2b) or labeled labeled by q_i in r (and their t-symbol is replaced by $\Box_{q_i}$).

  By (2) and induction hyp., we can construct a regular exp. for the subtree-components. We construct a reg. exp. for $L(q, N, K)$ using:

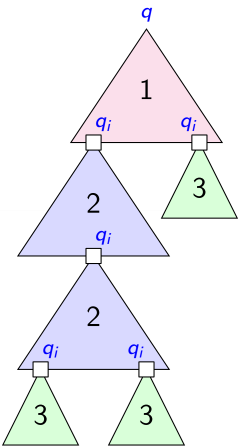
  $$L(q, N_0 \cup \{q_i\}, K) = L(q, N_0, K) +$$
  $$L(q, N_0, K \cup \{\Box_{q_i}\}) \cdot_{\Box_{q_i}} (L(q_i, N_0, K \cup \{\Box_{q_i}\}))^{*_{\Box_{q_i}}} \cdot_{\Box_{q_i}} L(q_i, N_0, K)$$



$$L(q, N_0 \cup \{q_i\}, K) = L(q, N_0, K) +$$

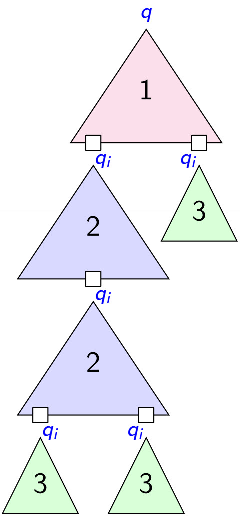


$$L(q, N_0 \cup \{q_i\}, K) = L(q, N_0, K) +$$



$$L(q, N_0 \cup \{q_i\}, K) = L(q, N_0, K) +$$

$$\underbrace{L(q, N_0, K \cup \{\Box_{q_i}\})}_{1} \cdot_{\Box_{q_i}} \big(\underbrace{L(q_i, N_0, K \cup \{\Box_{q_i}\})}_{2}\big)^{*_{\Box_{q_i}}} \cdot_{\Box_{q_i}} \underbrace{L(q_i, N_0, K)}_{3}$$



$$L(q, N_0 \cup \{q_i\}, K) = L(q, N_0, K) +$$

$$\underbrace{L(q, N_0, K \cup \{\Box_{q_i}\})}_{1} \cdot_{\Box_{q_i}} \big(\underbrace{L(q_i, N_0, K \cup \{\Box_{q_i}\})}_{2}\big)^{*_{\Box_{q_i}}} \cdot_{\Box_{q_i}} \underbrace{L(q_i, N_0, K)}_{3}$$

Congruences on trees

Definition: Congruence

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

- $\equiv$ is called a *congruence*
  if for all $n \geq 0$ 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 $u \in L \iff v \in L$.

Congruences on trees

Definition: Congruence

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

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

For $L \subseteq T(\mathcal{F})$, write $u \equiv_L v$ if

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

Congruences on trees

Definition: Congruence

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

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

For $L \subseteq T(\mathcal{F})$, write $u \equiv_L v$ if

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

Myhill-Nerode Theorem for trees

The following are equivalent:

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

Myhill-Nerode Theorem

Application:

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

Myhill-Nerode Theorem

Application:

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

Proof of the theorem (sketch):

- $1 \to 2$: Let $\mathcal{A}$ be DCFTA and let $u \equiv v$ iff $u \to^*_{\mathcal{A}} q \;{}^*_{\mathcal{A}}\!\leftarrow v$.
  Then $\equiv$ is of finite index and saturates L.

Myhill-Nerode Theorem

Application:

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

Proof of the theorem (sketch):

- $1 \to 2$: Let $\mathcal{A}$ be DCFTA and let $u \equiv v$ iff $u \to^*_{\mathcal{A}} q \ {}^*_{\mathcal{A}}\!\leftarrow v$.
  Then $\equiv$ is of finite index and saturates L.
- $2 \to 3$: Let $\equiv$ be a saturating congruence, $u \equiv v$ implies $u \equiv_L v$ (prove $u \equiv v$ implies $C[u] \equiv C[v]$ for all C, by induction on height of position of x in 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 $\equiv_L$ is not of finite index, and L is not recognizable.

Proof of the theorem (sketch):

- $1 \to 2$: Let $\mathcal{A}$ be DCFTA and let $u \equiv v$ iff $u \to^*_{\mathcal{A}} q \; {}^*_{\mathcal{A}}\!\leftarrow v$.
  Then $\equiv$ is of finite index and saturates L.
- $2 \to 3$: Let $\equiv$ be a saturating congruence, $u \equiv v$ implies $u \equiv_L v$ (prove $u \equiv v$ implies $C[u] \equiv C[v]$ for all C, by induction on height of position of x in C).
- $3 \to 1$: Let $\mathcal{A} = \langle T(\mathcal{F})_{/\equiv_L}, \mathcal{F}, L_{/\equiv_L}, \Delta \rangle$, with Δ containing

  $$f([u_1], \ldots, [u_n]) \to [f(u_1, \ldots, u_n)]$$

  for all $n \geq 0$, $f \in \mathcal{F}_n$, $u_1, \ldots, u_n \in T(\mathcal{F})$,
  where $[u]$ is the $\equiv_L$-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 $\equiv_L$ is not of finite index, and L is not recognizable.

Proof of the theorem (sketch):

- $1 \to 2$: Let $\mathcal{A}$ be DCFTA and let $u \equiv v$ iff $u \to^*_{\mathcal{A}} q \ {}^*_{\mathcal{A}}\!\leftarrow v$.
  Then $\equiv$ is of finite index and saturates L.
- $2 \to 3$: Let $\equiv$ be a saturating congruence, $u \equiv v$ implies $u \equiv_L v$ (prove $u \equiv v$ implies $C[u] \equiv C[v]$ for all C, by induction on height of position of x in C).
- $3 \to 1$: Let $\mathcal{A} = \langle T(\mathcal{F})_{/\equiv_L}, \mathcal{F}, L_{/\equiv_L}, \Delta \rangle$, with Δ containing

  $$f([u_1], \ldots, [u_n]) \to [f(u_1, \ldots, u_n)]$$

  for all $n \geq 0$, $f \in \mathcal{F}_n$, $u_1, \ldots, u_n \in T(\mathcal{F})$,
  where $[u]$ is the $\equiv_L$-equivalence class of $u \in T(\mathcal{F})$;

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

Decision Problems (on words)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$?

- **Universality**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = \Sigma^*$?

- **Language Inclusion**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

- **Language Equivalence**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$?

Decision Problems (on words)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$? NL-complete

- **Universality**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = \Sigma^*$? PSPACE-complete

- **Language Inclusion**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$? PSPACE-complete

- **Language Equivalence**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$? PSPACE-complete

Decision Problems (on trees)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$?

- **Universality**

  Given $\mathcal{A}$, is $L(\mathcal{A}) = T(\mathcal{F})$?

- **Language Inclusion**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?

- **Language Equivalence**

  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$?

Decision Problems (on trees)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$? P-complete
- **Universality**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = T(\mathcal{F})$? EXPTIME-complete
- **Language Inclusion**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$? EXPTIME-complete
- **Language Equivalence**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$? EXPTIME-complete

Emptiness of NFTA

Emptiness is P-complete

- in P: reachable states by (bottom-up) saturation algorithm
- P-hard: reduction from AND-OR graph reachability

AND-OR graph: $G = \langle V_A \uplus V_O, E \rangle$

We say that v_t is *reachable* from u in G if

- $u = v_t$, or
- $u \in V_A$, and v_t is reachable from v for all $v \in E(u)$, or
- $u \in V_O$, and v_t is reachable from v for some $v \in E(u)$

AND-OR graph reachability: given $\langle G, v_s, v_t \rangle$, is v_t reachable from v_s ?

Reduction: T-NFTA $\mathcal{A} = \langle V_A \cup V_O, \mathcal{F}, \{v_s\}, \Delta \rangle$ where Δ contains:

- $v_t(a) \rightarrow \varepsilon$,
- $u(f_n) \rightarrow (v_1, \ldots, v_n)$ for all $u \in V_A$ where $E(u) = \{v_1, \ldots, v_n\}$,
- $u(f_1) \rightarrow v$ for all $u \in V_O$, $v \in E(u)$.

Reminder: NFA universality

NFA universality is PSPACE-complete

- in (N)PSPACE: emptiness of subset construction
- PSPACE-hard: reduction from membership problem of PSpace TM

Reminder: NFA universality

NFA universality is PSPACE-complete

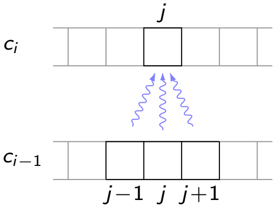
- in (N)PSPACE: emptiness of subset construction
- PSPACE-hard: reduction from membership problem of PSpace TM

Execution of a TM on input word w:

$$c_0 \vdash c_1 \vdash \cdots \vdash c_k$$

where $c_i \in \Sigma^*$ with $\Sigma = \Gamma \cup (Q \times \Gamma)$
$c_0 = (q, w_0)w_1 w_2 \ldots w_n$, accepts if $c_k \in (Q_{acc} \times \Gamma)\Gamma^*$.

The successor relation $\vdash$ is determined by a function $\text{Next} : \Sigma^3 \to \Sigma$:
$c_{i+1,j} = \text{Next}(c_{i,j-1}, c_{i,j}, c_{i,j+1})$



Reminder: NFA universality

NFA universality is PSPACE-complete

- in (N)PSPACE: emptiness of subset construction
- PSPACE-hard: reduction from membership problem of PSpace TM

Given $\mathcal{M}$ with space bounded by $p(\cdot)$ and input word w, construct $\mathcal{A}_\mathcal{M}$ to accept (encoding) of accepting runs of $\mathcal{M}$ on w:

- $\mathcal{A}_\mathcal{M}$ has alphabet Σ
- $\mathcal{A}_\mathcal{M} = \mathcal{A}_{init} \cap \bigcap_{1 \leq i \leq p(|w|)} \mathcal{A}_i \cap \mathcal{A}_{final}$ (interesection of DFAs)
  - $L(\mathcal{A}_{init})$: run starts with c_0
  - $L(\mathcal{A}_i)$: the i-th tape cell is correctly updated along the run
  - $L(\mathcal{A}_{final})$: run contains some $q \in Q_{acc}$

How to proceed deterministically?

How many states in $\mathcal{A}_{init}$? in $\mathcal{A}_i$? in $\mathcal{A}_{final}$?

- $L(\mathcal{A}_\mathcal{M}) \neq \emptyset$ iff $\mathcal{M}$ accepts w.

Let $\mathcal{A} = \overline{\mathcal{A}}_{init} \cup \bigcup_{1 \leq i \leq p(|w|)} \overline{\mathcal{A}}_i \cup \overline{\mathcal{A}}_{final}$

Then $L(\mathcal{A}) \neq \Sigma^*$ iff $\mathcal{M}$ accepts w.

How many states in $\mathcal{A}$?

Decision Problems (on words)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$?
  NL-complete
- **Universality**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = \Sigma^*$?
  PSPACE-complete
- **Language Inclusion**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?
  PSPACE-complete
- **Language Equivalence**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$?
  PSPACE-complete
- **Intersection Emptiness**
  Given DFA $\mathcal{A}_1, \ldots, \mathcal{A}_n$, is $\bigcap_i L(\mathcal{A}_i) = \varnothing$?
  PSPACE-complete

Decision Problems (on trees)

(*) $\mathcal{A}, \mathcal{B}$: nondeterministic automata

- **Emptiness**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = \varnothing$? — P-complete

- **Universality**
  Given $\mathcal{A}$, is $L(\mathcal{A}) = T(\mathcal{F})$? — EXPTIME-complete

- **Language Inclusion**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$? — EXPTIME-complete

- **Language Equivalence**
  Given $\mathcal{A}, \mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$? — EXPTIME-complete

- **Intersection Emptiness**
  Given NFTA $\mathcal{A}_1, \ldots, \mathcal{A}_n$, is $\bigcap_i L(\mathcal{A}_i) = \varnothing$? — EXPTIME-complete

  (even top-down or bottom-up DFTA)

Intersection problem

Theorem

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

Intersection problem

Theorem

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

Proof (sketch):

- in EXPTIME: compute reachable tuples of states in $\mathcal{A}_1 \times \cdots \times \mathcal{A}_n$.
- Hardness: reduction from membership problem of alternating TM with polynomial space.
  Runs of ATM are encoded as trees.
  Construct a product of tree automata to recognize accepting runs of the ATM on input word:
  - the run starts with c_0 ($\mathcal{A}_{init}$)
  - the i-th tape cell is correctly updated along all branches ($\mathcal{A}_i$)
  - all branches contain some $q \in Q_{acc}$ ($\mathcal{A}_{final}$)

Can we proceed (top-down/bottom-up) deterministically?

Path languages

Path languages

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

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

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

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

Path languages

Path languages

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

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

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

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

Path closure

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

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

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

Path closure and T-NFTA

Lemma

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

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

Path closure and T-NFTA

Lemma

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

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

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

- Construct a finite (word) automaton out of $\mathcal{A}$.
  (Easy, but does require $\mathcal{A}$ to be reduced!)

Path closure and T-NFTA

Lemma

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

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

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

- Construct a finite (word) automaton out of $\mathcal{A}$.
  (Easy, but does require $\mathcal{A}$ to be reduced!)
- Construct $\mathcal{A}^{\text{pc}} = \langle Q, \mathcal{F}, G, \Delta' \rangle$ for $pc(L)$ as follows:
  for all $a \in \mathcal{F}_0$:

$$q(a) \rightarrow_{\Delta} \varepsilon \quad \rightarrow \quad q(a) \rightarrow_{\Delta'} \varepsilon$$

  for all $n \geq 1$, $f \in \mathcal{F}_n$:

$$\begin{array}{l} q(f) \rightarrow_{\Delta} (q_{i,1}, \ldots, q_{i,n}) \\ i = 1, \ldots, n \end{array} \quad \rightarrow \quad q(f) \rightarrow_{\Delta'} (q_{1,1}, \ldots, q_{n,n})$$

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

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

Path closure and T-NFTA

Corollary

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

Path closure and T-NFTA

Corollary

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

Theorem

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

Path closure and T-NFTA

Corollary

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

Theorem

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

Proof:

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

  Show that $L_S(\mathcal{A}') = \bigcup_{q \in S} L_q(\mathcal{A})$, for all $S \subseteq Q$.

Path closure and T-NFTA

Corollary

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

Theorem

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

Proof:

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

  Show that $L_S(\mathcal{A}') = \bigcup_{q \in S} L_q(\mathcal{A})$, for all $S \subseteq Q$.
- "$\leftarrow$":
  Let $\mathcal{A}$ be a redcued T-DCFTA for L. Prove that
  if $\pi(t) \subseteq \pi(L_q(\mathcal{A}))$, then $t \in L_q(\mathcal{A})$, for all $q \in Q, t \in T(\mathcal{F})$.