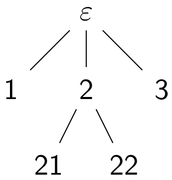
Trees

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

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

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



Definition: Tree

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

Ranked Trees

Ranked symbols

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

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

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

Definition: Ranked tree

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

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

Trees and Terms

Definition: Terms

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

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

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

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

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

We confuse terms and trees in the obvious manner.

Subterms / subtrees

Definition: Subtree

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

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

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

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

Substitutions and Context

Definition: Substitution

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

Definition: Context

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

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

Tree automata

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

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

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

Two variants: bottom-up vs top-down labelling

Basic results (preview)

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

Bottom-up automata

Definition: (Bottom-up tree automata)

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

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

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

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

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

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

Move relation and computation tree

Move relation

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

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

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

Computation

Let $t : Pos \to \mathcal{F}$ a ground tree. A *run* or *computation* 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 $f(q_1, \ldots, q_n) \to q \in \Delta$

Regular tree languages

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

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

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

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

NFTA with ε-moves

Definition:

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

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

Equivalence of ε-NFTA

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

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

Deterministic, complete, and reduced NFTA

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

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

A state q of $\mathcal{A}$ is *accessible* if there exists a tree t s.t. $t \rightarrow^*_{\mathcal{A}} q$.
$\mathcal{A}$ 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

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

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

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

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

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

From top-down to bottom-up

Theorem (T-NFTA = NFTA)

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

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

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

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

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

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

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

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

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

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

From NFTA to DFTA

Theorem (NFTA=DFTA)

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

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

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

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

DFTA with accessible states

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

Deterministic top-down are less powerful

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

Closure properties

Theorem (Boolean closure)

Recognizable tree languages are closed under Boolean operations.

Negation (invert accepting states)

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

Union (juxtapose)

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

Cross-product construction

Direct intersection

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

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

Remarks:

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

Tree languages and context-free languages

Front

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

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

Leaf languages

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

Proof (idea):

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