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 $N^*$:

![Diagram of a tree]

**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 $w_i \in Pos$ 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 $t : Pos \rightarrow (\mathcal{F} \cup \mathcal{X})$ 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 $Pos \cap pN = \emptyset$. 

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.
Definition: Subtree

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

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

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

\( t \triangleright t' \) (resp. \( t \triangleright\triangleright t' \)) 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 $\sigma(t)$

**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:

\[ abc \overset{\triangleleft}{=} a(b(c($)))) \]

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

- $a \rightarrow q_0$
- $g(q_0) \rightarrow q_1$
- $g(q_1) \rightarrow q_1$
- $f(q_1, q_1) \rightarrow q_f$
Move relation

Let \( t, t' \in T(\mathcal{F}, Q) \). We write \( t \xrightarrow{A} t' \) 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 \( \xrightarrow{A}^* \) for the transitive and reflexive closure of \( \xrightarrow{A} \).

Computation

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

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

A tree $t$ is accepted by $A$ iff $t \rightarrow^* A q$ 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 \( \epsilon \)-moves**

**Definition:**
An \( \epsilon \)-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 \( \epsilon \)-NFTA**
For every \( \epsilon \)-NFTA \( \mathcal{A} \) there exists an equivalent NFTA \( \mathcal{A}' \).

**Proof (sketch):** Construct the rules of \( \mathcal{A}' \) by a saturation procedure.
An NFTA is \emph{deterministic} if no two rules have the same left-hand side. An NFTA is \emph{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 \emph{at most} one run per tree. A DCFTA as \emph{exactly} one run per tree.

A state $q$ of $A$ is \emph{accessible} if there exists a tree $t$ s.t. $t \rightarrow^* A q$. $A$ is said to be \emph{reduced} if all its states are accessible.
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 \( q, q_1, \ldots, q_n \in Q \).

Move relation: \( t \rightarrow_\mathcal{A} t' \) 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 \( q(f) \rightarrow (q_1, \ldots, q_n) \).

\( t \) is accepted by \( \mathcal{A} \) if \( q(t) \rightarrow^*_\mathcal{A} t \) 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 $A = \langle Q, \mathcal{F}, G, \Delta \rangle$ iff it is accepted by T-NFTA $A' = \langle Q, \mathcal{F}, G, \Delta' \rangle$, with

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

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

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

  $$t = a \rightarrow^*_A q \iff a \rightarrow^{\Delta} q \iff q(a) \rightarrow^{\Delta'} \varepsilon \iff q(a) \rightarrow^*_{A'} a$$

- **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.

**Claim (subset construct.):** Let \( A = \langle Q, \mathcal{F}, G, \Delta \rangle \) an NFTA recognizing \( L \). The following DCFTA \( A' = \langle 2^Q, \mathcal{F}, G', \Delta' \rangle \) 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 \( t \rightarrow^*_A \{ q \mid t \rightarrow^*_A q \} \), 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.

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

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

Remarks:

- If \( A_1, A_2 \) are D(C)FTA, then so is \( A \).
- If \( A_1, A_2 \) 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 $fr(t) \in F_0^*$ 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 $L = fr(L(\mathcal{A}))$.

Proof (idea):

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