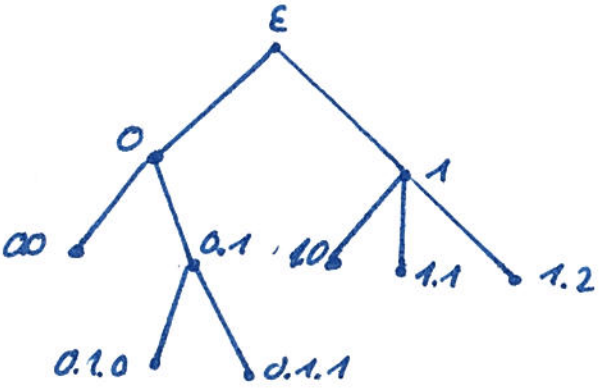
Example:



Definition (Ranked alphabet, Σ-trees)

A ranked alphabet is a

finite set Σ

together with a rank function $rk : \Sigma \to \mathbb{N}$.

Call

$rk(a)$ = rank of letter a

Intuitively:

- a expects $rk(a)$ children
- similar to arity of function/predicate symbols

A Σ-tree is a function

$t : T \to \Sigma$

where T is a finite tree as above.

Moreover, for all $w \in T$ and all $a \in \Sigma$ with $t(w) = a$,

t satisfies $w.i \in T$ iff $i < rk(a)$ f.a. $i \in \mathbb{N}$.

// If w is labelled by a then w has precisely $rk(a)$ children.

Let $\Sigma_n = \{ a \in \Sigma \mid rk(a) = n \}$

Moreover, T_Σ = set of all Σ-trees.

Note:

- Impossible to find two nodes with same label but different number of children
- Alphabet gives upper bound on number of children in a tree.

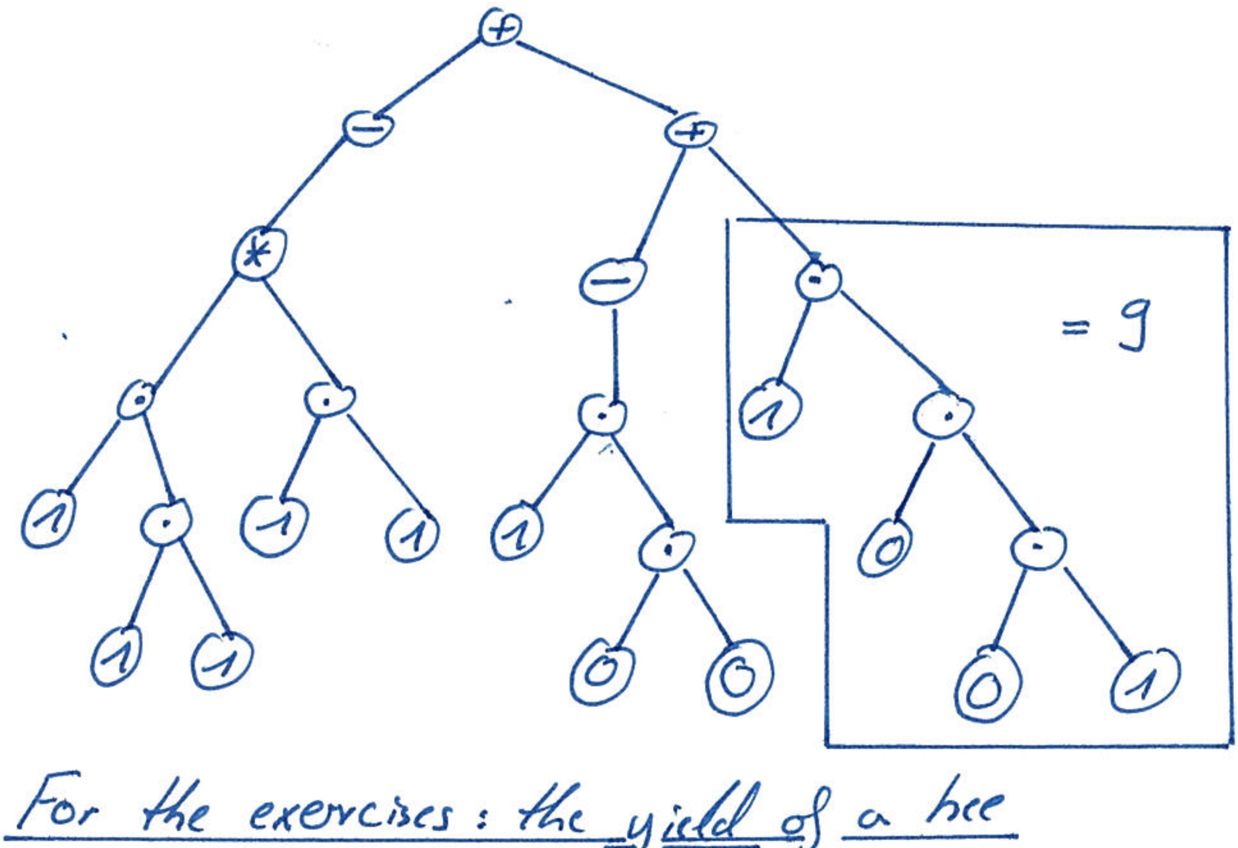
Example:

Let $\Sigma = \{+/2, */2, \cdot/2, -/1, 0/0, 1/0\}$ a ranked alphabet.

Arithmetic expression

$$-(7*3) + (-4+9)$$

in binary encoding can be represented by



For the exercises: the yield of a tree

Read word consisting of letters on the leaves.

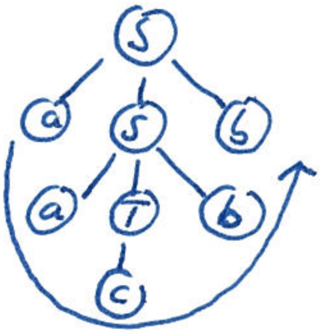
Let $t: T \to \Sigma$ a tree.

Its yield is defined inductively:

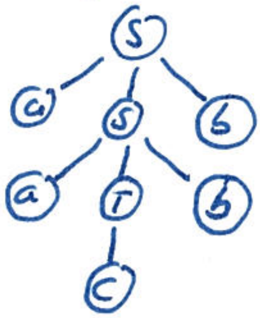
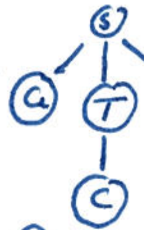
If $T = \{\varepsilon\}$ then $\text{yield}(t) := t(\varepsilon)$.

Otherwise, t has subtrees $t_0, \dots, t_n$.
Let
$$T = \{\varepsilon\} \cup 0.T_0 \cup \dots \cup n.T_n.$$
Define
$$t_i : T_i \to \Sigma \quad \text{by} \quad t_i(w) = t(i.w) \quad \text{f.a. } 0 \le i \le n.$$
With this
$$\mathrm{yield}(t) := \mathrm{yield}(t_0) \dots \mathrm{yield}(t_n).$$

<u>Example:</u>

yield() = aacbb.

Apply the definition:

yield() = yield(a). yield(). yield(b)

= a. yield(a). yield(). yield(b). b

= a. a. yield(c). b. b

= aa. c. b. b.

<u>Two different automaton models for trees:</u>

↳ Finite automata read words from left to right
↳ Theory would not change if they read words from right to left.
↳ Trees look different when read from top to bottom vs. bottom up.

Difference:

- From top to bottom distribute information from one node to many
- Bottom-up aggregates information from children.

=> yields different theories.

Definition (Bottom-up tree automaton)

A bottom-up tree automaton (BUTA) over Σ is a tuple

$$A = (Q, \rightarrow, Q_F)$$

with

- set of states Q (finite)
- transition relation $\rightarrow = (\rightarrow_a)_{a\in\Sigma}$ with

$$\rightarrow_a \subseteq Q^n \times Q \quad \text{where } n = rk(a).$$

- set of final states $Q_F \subseteq Q$.

A run of a BUTA labels nodes of a tree by states

↳ starting at the leaves
↳ stopping at the root
↳ transitions read states of (root of) subtrees

} bottom-up.

No initial state:

↳ Encoded into transition relation for a with $rk(a) = 0$.

↳ Define $\rightarrow_a \subseteq Q^0 \times Q$ as $\rightarrow_a \subseteq Q$.

↳ This means initial state chosen according to leaf letter

↳ Slight difference when compared to finite automata

=> can always extend finite automata by one state to achieve this effect.

Definition ((accepting) run, tree language):

A run of BUTA $A = (Q, \to, Q_F)$ on a Σ-tree $t: T \to \Sigma$ is a function

$$r: T \to Q$$

so that for all $w \in T$ we have

$$(r(w.0), \ldots, r(w.(n-1))) \to_a r(w)$$
$$(\overset{\shortparallel}{q_0}, \ldots, \overset{\shortparallel}{q_{n-1}}) \qquad \overset{\shortparallel}{q}$$

where $a = t(w)$ and $n = rk(a)$.

A run is accepting if $r(\varepsilon) \in Q_F$.

Then $\mathcal{L}(A) \subseteq T_\Sigma$ is the (tree) language of A:

$$\mathcal{L}(A) := \{ t \in T_\Sigma \mid A \text{ has an accepting run on } t \}.$$

Call the class of tree languages that can be accepted by BUTA the regular tree languages.

Example:

Let $\Sigma = \{ \vee/2, \wedge/2, \neg/1, t/0, f/0 \}$

- Allows us to encode all variable-free Boolean expressions as trees.
- Language of all expressions that evaluate to true is accepted by following BUTA:

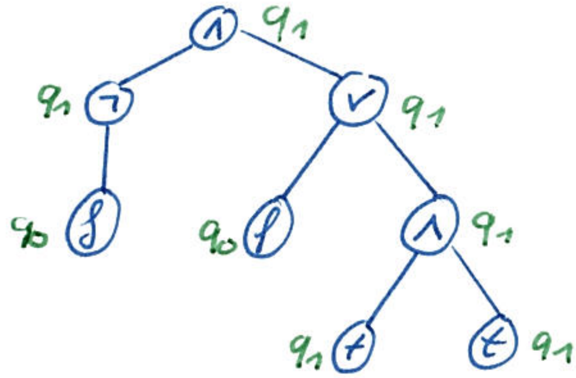
$$A = (\{q_0, q_1\}, \to, \{q_1\})$$

with

$$\to_f q_0 \qquad q_0 \to_\neg q_1 \qquad (q_0, q_0) \to_\vee q_0 \qquad (q_0, q_0) \to_\wedge q_0$$
$$\to_t q_1 \qquad q_1 \to_\neg q_0 \qquad (q_0, q_1) \to_\vee q_1 \qquad \ldots$$
$$\ldots \qquad (q_1, q_1) \to_\wedge q_1.$$

Note that this BUTA is deterministic.

Consider



Run is accepting since

$r(\varepsilon) = q_1 \in Q_F = \{q_1\}$.

__Definition (Deterministic BUTA):__

A BUTA $A = (Q, \to, Q_F)$ is called __deterministic (DBUTA)__ if for all $a \in \Sigma$ and all $(q_0, \dots, q_{n-1}) \in Q^n$ with $n = rk(a)$ we have precisely one $q \in Q$ so that

$$(q_0, \dots, q_{n-1}) \to_a q.$$

Are deterministic BUTA as powerful as non-deterministic BUTA?

↳ Yes, apply powerset construction.

__Theorem:__

A tree language is accepted by a BUTA iff it is accepted by a DBUTA.

__Proof:__

$\Leftarrow$ By definition

$\Rightarrow$ Let $L = L(A)$ with $A = (Q^A, \to^A, Q_F^A)$.

Construct

$$A' := (\mathcal{P}(Q^A), \to, Q_F)$$

with

$$Q_F := \{ Q \subseteq Q^A \mid Q \cap Q_F^A \neq \emptyset \}$$

and

$$(Q_0, \dots, Q_{n-1}) \to_a Q \text{ where}$$

$$Q = \{ q \in Q^A \mid \text{there are } q_0 \in Q_0, \dots, q_{n-1} \in Q_{n-1} \text{ so that } (q_0, \dots, q_{n-1}) \to^A_a q \}$$

with $n = rk(a)$. $\square$

As a consequence, regular tree languages closed under complementation.

<u>Lemma</u>:

Let A a DBUTA accepting L.

Then there is a DBUTA $\overline{A}$ accepting $\overline{L}$.

<u>Proof</u>:

Swap final and non-final states.

If $A = (Q, \rightarrow, Q_F)$, set $\overline{A} = (Q, \rightarrow, Q \setminus Q_F)$.

□

Regular tree languages also closed under union.

Second possibility of reading trees: top-down.

<u>Definition (Top-down tree automata)</u>:

A <u>top-down tree automaton (TDTA) over Σ</u> is a tuple $A = (Q, q_0, \rightarrow)$ with

- set of <u>states</u> Q (finite)
- <u>initial state</u> $q_0 \in Q$
- <u>transition relation</u> $\rightarrow = (\rightarrow_a)_{a \in \Sigma}$ with

  $\rightarrow_a \subseteq Q \times Q^n$ with $n = rk(a)$.

A TDTA A is called <u>deterministic (DTDTA)</u> if for all $a \in \Sigma$ and all $q \in Q$ there is precisely one vector $(q_0, \ldots, q_{n-1}) \in Q^n$ with $n = rk(a)$ so that

$$q \rightarrow_a (q_0, \ldots, q_{n-1}).$$

<u>Definition (Run of TDTA, language)</u>:

A <u>run of a TDTA $A = (Q, q_0, \rightarrow)$ on a Σ-tree</u> $t : T \rightarrow \Sigma$ is a function

$$r : T \rightarrow Q$$

with

$r(\varepsilon) = q_0$ and $r(w) \rightarrow_a (r(w.0), \ldots, r(w.(n-1)))$ f.a. $w \in T$ with $a = t(w)$ and $n = rk(a)$.

Then

$$\mathcal{L}(A) := \{ t \in T_\Sigma \mid \text{there is a run } r \text{ of } A \text{ on } t \}.$$

There are no final states:

↳ Modelled by the fact that

$r(w) \to_a ()$ defined by $r(w) \in Q$ $(\times Q^0)$.

(Vs. non existence of such a transition).

<u>Example:</u>

Let $\Sigma = \{ a/2, b/2, c/0 \}$

- Consider language of all trees that contain at least one b.
- Is TDTA acceptable by

$$A = (\{q_-, q_+\}, q_-, \to)$$

with

$q_+ \to_a (q_+, q_+)$

$q_+ \to_b (q_+, q_+)$

$q_+ \to_c$

$q_- \to_a (q_+, q_-)$

$q_- \to_a (q_-, q_+)$

$q_- \to_b (q_+, q_+)$

Intuition:

q_- = still need to find a "b"

q_+ = have already seen a "b" somewhere.