Vorlesung Formale Sprachen

Baumsprachen und Baumautomaten

Anna Kasprzik

Chomsky Hierarchy for strings

A formal grammar is a quadruplet $G = (N, \Sigma, P, S)$ where N is some nonterminal alphabet, Σ some terminal alphabet, and $S \in N$ the start symbol. P contains grammar rules of the form $w_1 \longrightarrow w_2$ with $w_1, w_2 \in (\Sigma \cup N)^*$.

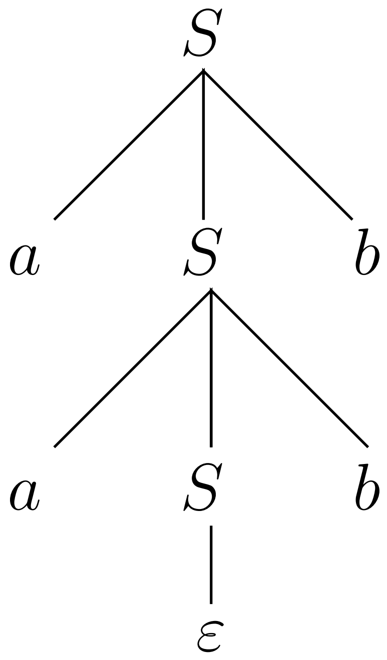
- Regular: $w_1 \in N$, $w_2 \in \Sigma N$
- Context-free: $w_1 \in N$
- Context-sensitive: $|w_1| \leq |w_2|$
- Type 0: No constraints.

Derivation trees

A derivation tree shows which rules have been applied during the derivation of some string w in some grammar G.

Example 1 *Let* $G = (\{S\}, \{a, b\}, \{S \longrightarrow aSb, S \longrightarrow \varepsilon\}, S)$ *be a (context-free) grammar (*ε *is the empty string).*

A derivation for the string $aabb$:
$S \Rightarrow aSb$
$\Rightarrow aaSbb \Rightarrow$
$aa\varepsilon bb = aabb$



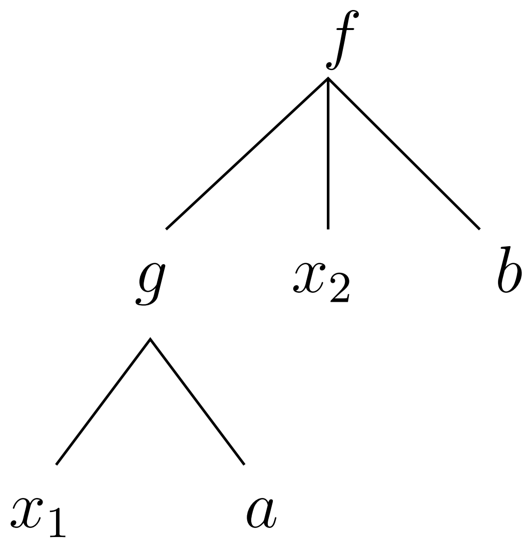
Trees as objects

A *ranked alphabet* is a finite set of symbols, each associated with a rank $n \in \mathbb{N}$. By Σ_n we denote the set of all symbols in Σ with rank n.

The set T_Σ of all trees over Σ is defined inductively as the smallest set of expressions such that $f[t_1, \ldots, t_n] \in T_\Sigma$ for every $f \in \Sigma_n$ and all $t_1, \ldots, t_n \in T_\Sigma$. A subset of T_Σ is called a tree language. $t_1, \ldots, t_n$ are the *direct subtrees* of the tree. The set $subtrees(t)$ consists of t itself and all subtrees of its direct subtrees. The height of a tree t is the length of the longest path from the root to some leaf in t.

Example: Trees described by terms

$f(g(x_1, a), x_2, b)$



Tree grammars

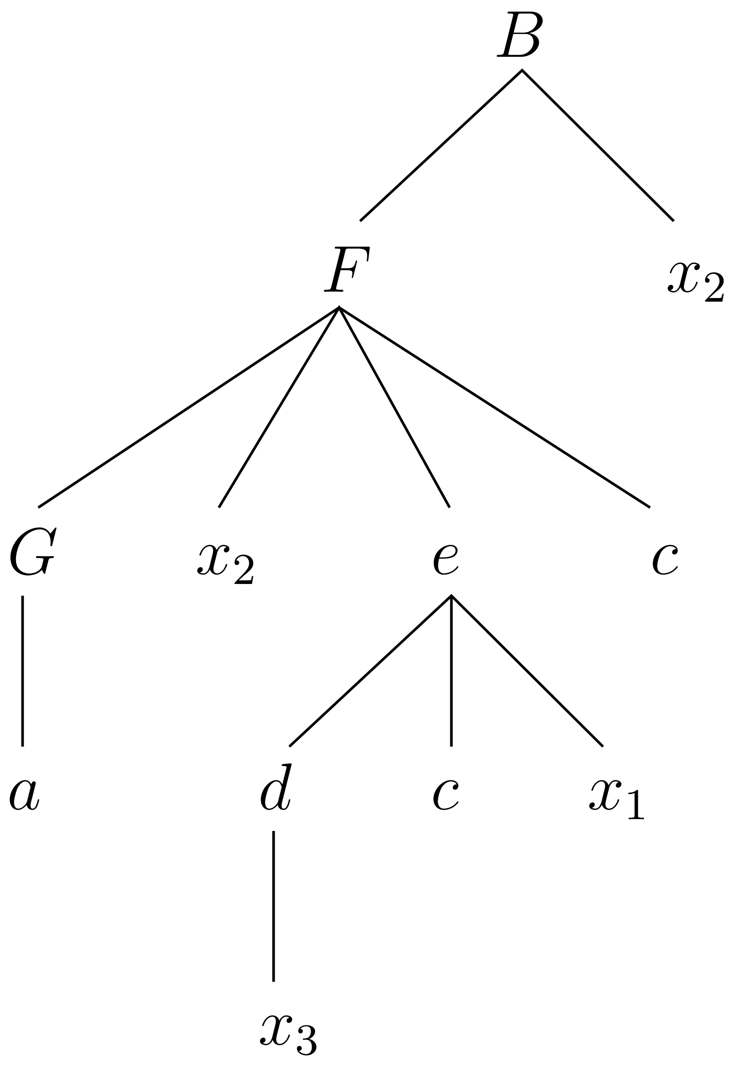
Definition 1 *A* context-free tree grammar (CFTG) *is a quintuple* $G = (\Sigma, \mathsf{F}, S, X, P)$*, where* Σ *and* F *are ranked alphabets of terminals and nonterminals, respectively.* $S \in \mathsf{F}$ *is the start symbol,* X *is a set of variables, and* P *is a finite set of rules of the form* $F(x_1, \ldots, x_n) \longrightarrow t$ *for some* $n \in \mathbb{N}$*, where* $F \in \mathsf{F}_n$*,* $x_1, \ldots, x_n \in X$*, and* $t \in T_{\Sigma \cup \mathsf{F}}(\{x_1, \ldots, x_n\})$*.*

G *is a* regular tree grammar (RTG) *iff* $\mathsf{F}_n = \emptyset$ *for* $n \neq 0$ *(and, consequently,* $X = \emptyset$ *as well). Then* G *is in* normal form *iff all rules in* P *are of the form* $F \longrightarrow f(t_1, \ldots, t_m)$ *where* $f \in \Sigma_m$ *for* $m \geq 0$ *and* $t_1, \ldots, t_m \in \mathsf{F} \cup \Sigma_0$*.*

A set of (nonterminal- and variable-free) trees is a context-free (regular) tree language *if there is a context-free (regular) tree grammar that generates it.*

A context-free tree grammar rule

$$A(x_1, x_2, x_3, x_4) \longrightarrow$$



(Pushdown tree automata for CFTLs)

Definition 2 *A* top-down pushdown tree automaton *is a sextuple* $\mathcal{A} = (Q, \Sigma, \Gamma, P, a_0, z_0)$ *where the set of states* Q *is a ranked alphabet consisting of binary symbols,* Σ *the ranked input alphabet,* Γ *the ranked pushdown alphabet,* $a_0 \in Q$ *the initial state,* $z_0 \in \Gamma_0$ *the start symbol of the pushdown store, and* P *a finite set of productions of the form*

- $a(f(\xi_1, \ldots \xi_m), g(\xi_{m+1} \ldots, \xi_{m+n})) \longrightarrow f(a_1(\xi_1, q_1), \ldots, a_m(\xi_m, q_m))$ for $a, a_1, \ldots, a_m \in Q, f \in \Sigma_m,$ $m \geq 0, g \in \Gamma_n, n \geq 0, q_1, \ldots, q_n \in T_\Gamma(\{\xi_{m+1}, \ldots, \xi_{m+n}\})$, or
- $a(\xi, g(\xi_1, \ldots, \xi_n)) \longrightarrow b(\xi, q)$ for $a, b \in Q, g \in \Gamma_n, n \geq 0, q \in T_\Gamma(\Xi_n)$.

Theorem 3 *A tree language is context-free iff it is recognized by a top-down pushdown tree automaton.*

Automata for regular tree languages

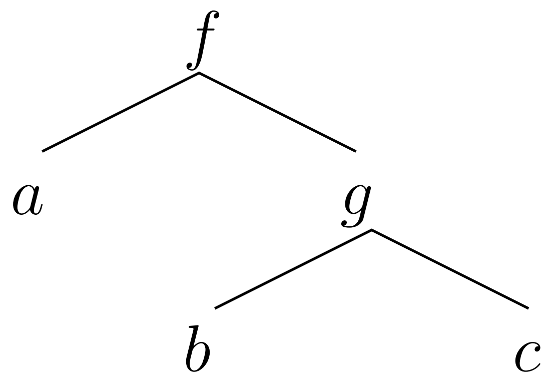
A *(total, deterministic) bottom-up finite-state tree automaton (FTA)* is a tuple $\mathcal{A} = (\Sigma, Q, \delta, F)$ where Σ is the ranked input alphabet, Q is the finite set of states, δ is the transition function assigning to every $f \in \Sigma_n$ and all $q_1, \ldots, q_n \in Q$ a state $\delta(q_1 \cdots q_n, f) \in Q$, and $F \subseteq Q$ is the set of accepting states. The transition function extends to trees: $\delta : T_\Sigma \longrightarrow Q$ is defined such that if $t = f[t_1, \ldots, t_n] \in T_\Sigma$ then $\delta(t) = \delta(\delta(t_1) \cdots \delta(t_n), f)$. The language accepted by $\mathcal{A}$ is $L(\mathcal{A}) = \{t \in T_\Sigma | \delta(t) \in F\}$.

The yield function

Let Σ be a ranked alphabet. The yield $yd(t)$ of a tree $t \in T_\Sigma$ can be defined as follows ($\cdot$ is string concatenation):

$$yd(t) = \begin{cases} t & \text{if } t \in \Sigma_0 \text{, and} \\ yd(t_1) \cdot \ldots \cdot yd(t_n) & \text{if } t = f(t_1, \ldots, t_n) \text{ for some } f \in \Sigma_n \\ & \quad \text{with } n \geq 1 \text{ and } t_1, \ldots, t_n \in T_\Sigma. \end{cases}$$

$t =$



$yd(t) = abc$

Regular tree languages and their yields

Theorem 4 *The following facts hold:*

- *Every context-free string language is the yield of a regular tree language.*
- *The yield of any regular tree language is a context-free string language.*

$\Longrightarrow$ *A string language is context-free iff it is the yield of a regular tree language.*

- *Let G be a context-free string grammar: The set of derivation trees of $L(G)$ is a regular tree language.*
- *There exists a regular tree language which is not the set of derivation trees of a context-free string language.*

Proofs

- – namely the set of its derivation trees.
- T is generated by an RTG $G = (\Sigma, \mathsf{F}, S, P)$ in normal form. If we define a context-free string grammar $G_s = (\mathsf{F}, \Sigma_0, P_s, S)$ with $P_s = \{A \longrightarrow yd(\gamma) | A \longrightarrow \gamma \in P\}$ then it is easy to show that for all $A \in \mathsf{F}$ and $w \in (\Sigma_0 \cup \mathsf{F})^*$, $A \Rightarrow^*_{G_s} w$ iff there exists a $t \in T_\Sigma$ such that $A \Rightarrow^*_G t$ and $yd(t) = w$. This implies that $yd(T) = L(G)$.
- Let $G = (\mathsf{F}, \Sigma, S, P)$ be a cf string grammar. We define the RTG $G' = (\Sigma', \mathsf{F}, P', S)$ with $\Sigma' = \Sigma \cup \{\varepsilon\} \cup \{(A, n) | \ A \in \mathsf{F}, \exists A \longrightarrow \alpha \in P : |\alpha| = n\}$ and $A \longrightarrow (A, 0)(\varepsilon) \in P'$ if $A \longrightarrow \varepsilon \in P$ and $A \longrightarrow (A, p)(a_1 \dots a_p) \in P'$ if $(A \longrightarrow a_1 \dots a_p) \in P$. Then $L(G) = \{yd(s) | s \in L(G')\}$ (per induction on derivation length).
- Consider the RTG $G = (\{a, b, g\}, \{X, Y, Z\}, X, \{X \longrightarrow f(Y, Z), Y \longrightarrow g(a), Z \longrightarrow g(b)\})$ generating the single tree $f(g(a), g(b))$. Assume that $L(G)$ is the set of derivation trees of some cf string grammar. This grammar should contain rules of the form $F \longrightarrow GG$, $G \longrightarrow a$, $G \longrightarrow b$. Then the tree $f(g(a), g(a))$ should also be in $L(G)$, but is not.

RTLs and their yields – fun facts

- If $yd(T)$ for some tree language T is context-free, then T may not be regular (consider $T = \{f(t,t)|t \in T_\Sigma\}$ with $\Sigma = \Sigma_0 \cup \Sigma_2 = \{x\} \cup \{f\}$ and $yd(T) = \{x^{2n}|n \geq 1\}$).
- If L is a context-free string language, then $yd^{-1}(L)$ may not be regular (consider $L = \{x^n y^n|n \geq 1\}$ and $\Sigma = \Sigma_0 \cup \Sigma_2 = \{x, y\} \cup \{f\}$). However, if L is regular then $yd^{-1}(L)$ is regular.
- If yd is not surjective, then $yd^{-1}(L)$ may be regular although L is not context-free (consider $L = \{x^n y^n x^{2n}|n \geq 1\}$ and $\Sigma = \Sigma_0 \cup \Sigma_3 = \{x, y\} \cup \{g\}$ so that $yd^{-1}(L) = \emptyset$). However, if $yd^{-1}(L)$ is regular and $yd(yd^{-1}(L)) = L$, then L is regular.

Closure properties of RTLs

Theorem 5 *The regular tree languages are closed under intersection, union, and complementation.*

Proof (Sketch). Union: Build the union automaton. Complementation: Build the complementary automaton (switch final and non-final states). Intersection: De Morgan.

Definition 6 *The* tree language product $T(x \longleftarrow T_x | x \in \Sigma_0)$ *of an* x*-indexed family* $(T_x | x \in \Sigma_0)$ *of tree languages and a tree language* T *is the set of all trees obtained by replacing in some* $t \in T$ *simultaneously every symbol* $x \in \Sigma_0$ *by a tree from the corresponding set* T_x*; different occurrences of a symbol* x *may be replaced by different trees from* T_x*.*

Theorem 7 *If* T *and* T_x *for all* $x \in \Sigma_0$ *are regular tree languages, then* $T(x \longleftarrow T_x | x \in \Sigma_0)$ *is as well.*

Proof (Sketch). Replace x by the start symbol of a grammar for T_x.

Contexts

Let $\Box$ be a special symbol of rank 0 (i.e., a leaf label). A tree $c \in T_{\Sigma \cup \{\Box\}}$ in which $\Box$ occurs exactly once is called a *context*, and the set of all contexts over Σ is denoted by C_Σ. For $c \in C_\Sigma$ and $s \in T_\Sigma$, $c[[s]]$ denotes the tree obtained by substituting s for $\Box$ in c. The depth of c is the length of the path from the root to the node labeled with $\Box$.

The Myhill-Nerode theorem

The equivalence relation $\equiv_L$ for some string language L over alphabet Σ is defined by: For x, y and all $z \in \Sigma^*$, $x \equiv_L y$ iff $xz \in L \Leftrightarrow yz \in L$.

Theorem 8 (Myhill-Nerode) *A language L is regular iff $\equiv_L$ partitions Σ^* in finitely many equivalence classes.*

For a tree language T over some ranked alphabet Σ the equivalence relation $\equiv_T$ is defined by: For $t_1, t_2 \in T_\Sigma$ and all $c \in C_\Sigma$, $t_1 \equiv_T t_2$ iff $c[[t_1]] \in T \Leftrightarrow c[[t_2]] \in T$.

Minimal automata for RTLs

A corollary of the Myhill-Nerode theorem:

Theorem 9 *For every regular tree language $T \subseteq T_\Sigma$ there is a unique minimal deterministic finite-state tree automaton $\mathcal{A}_{min}$ recognizing it (up to a renaming of the states).*
$\mathcal{A}_{min} = (\Sigma, Q_{min}, \delta_{min}, Q_{min_f})$ where

- *Q_{min} is the set of equivalence classes of $\equiv_T$,*
- *$Q_{min_f} = \{[u] | u \in T\}$, and*
- *δ_{min} is defined by $\delta_{min}(f, [u_1], \ldots, [u_n]) = [f(u_1, \ldots, u_n)]$ for all $f \in \Sigma_n$ and $u_1, \ldots, u_n \in T$.*

The Pumping lemma for RTLs

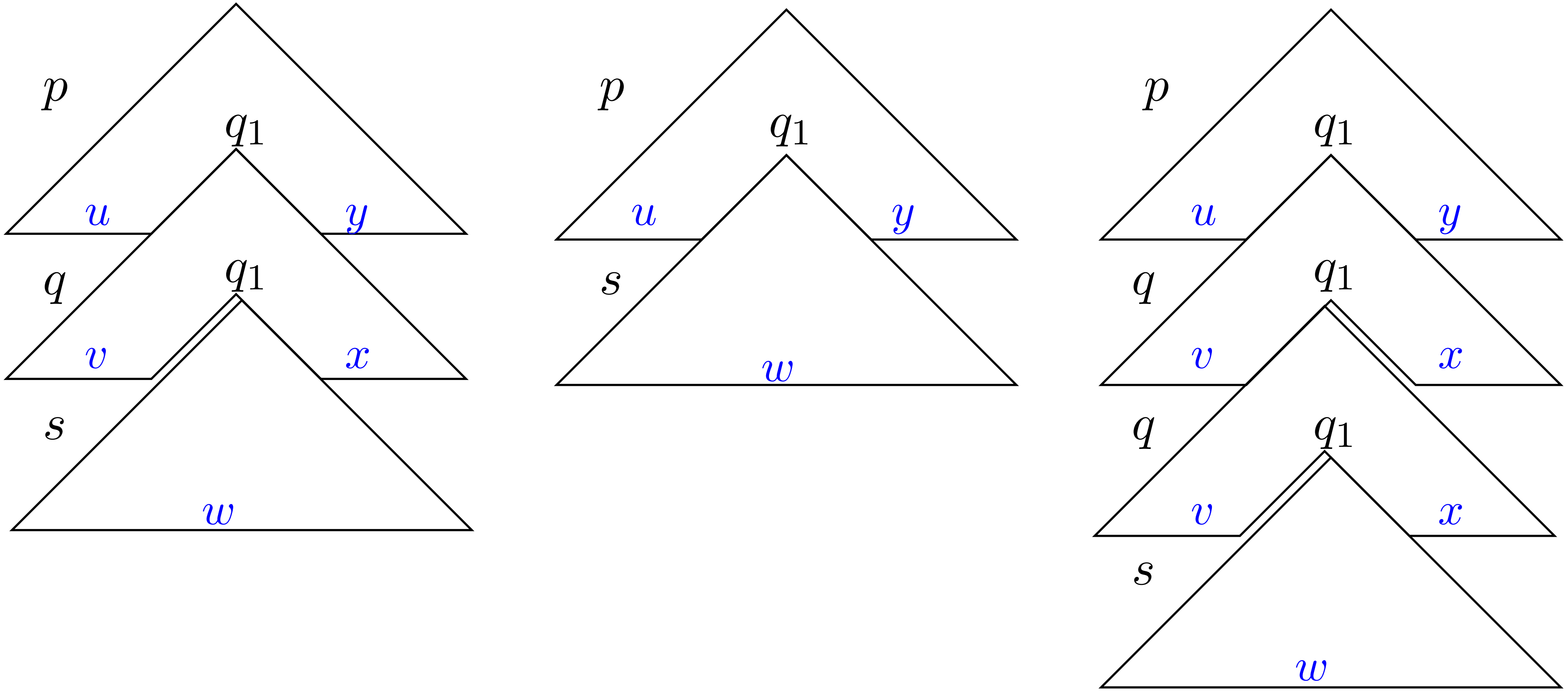
Lemma 1 *For any regular tree language $T \subseteq T_\Sigma$ there is a number $n \geq 1$ such that if $t \in T_\Sigma$ has height $k \geq n$ then for some $s \in T_\Sigma$ and $p, q \in C_\Sigma$*

- $t = q[[p[[s]]]]$ *where p has depth ≥ 1 and*
- $q[[\underbrace{p[[\ldots p[[s]]\ldots]]}_{k \text{ times}}]] \in T_\Sigma$ *for all $k \geq 0$.*

Proof.

Suppose T is recognized by some n-state finite tree automaton $\mathcal{A}$. Any tree $t \in T$ of height $\geq n$ contains a path from some leaf to the root consisting of at least $n+1$ nodes, and $\mathcal{A}$ must arrive in the same state at two of these nodes. A representation $q[[p[[s]]]]$ of the required kind is obtained by 'cutting' the tree at these nodes: $\delta(s) = \delta(p[[s]])$.

The Pumping lemma for RTLs



The PL for cf string languages revisited

(Reminder:)

Lemma 2 (Pumping lemma for CFLs) *Let L be a context-free string language. Then there exists $n \in \mathbb{N}$ such that all strings $z \in L$ with $|z| \geq n$ can be represented as $uvwxy$ such that*

- $|vx| \geq 1$
- $|vwx| \leq n$
- *for all $i \geq 0$: $uv^iwx^iy \in L$.*

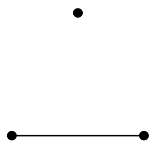
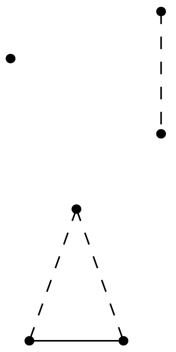
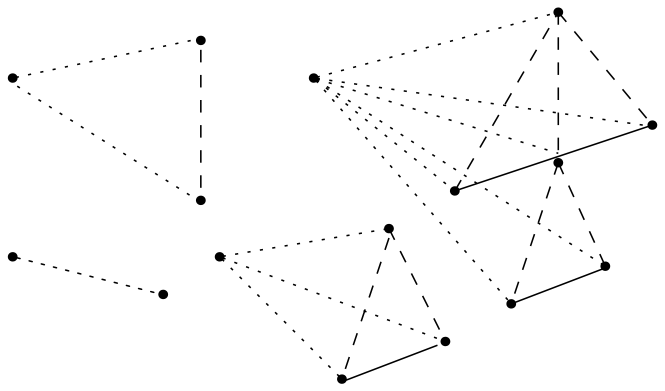
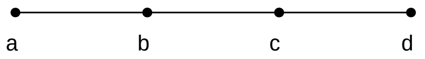
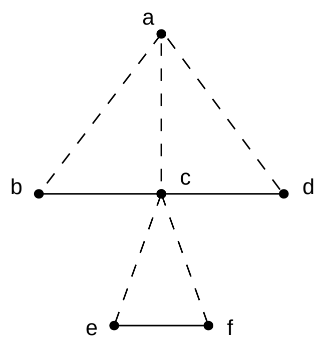
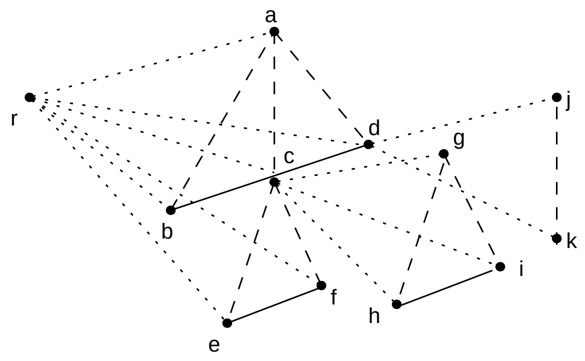
Note the elegance of this lemma when expressed via regular tree languages and their connection to cf string languages!

Trees as a generalization of strings

Strings can be seen as a special case of trees in which each node has (at most) one daughter. For example, the string abc could be represented as the tree $c(b(a(\varepsilon)))$.

Then all known results for (regular, context-free) string languages are a natural consequence of the results for the corresponding tree languages, a part of which has been presented here.

Why not generalize even further?

d	0	1	2	3
local				
composite (labeled)	-			

Multi-dimensional trees (own definition)

We will use finite d-dimensional tree labeling alphabets Σ^d where each symbol $f \in \Sigma^d$ is associated with at least one unlabeled $(d{-}1)$-dimensional tree t specifying the admissible child structure for a root labeled with f. t can be given in any form suitable for trees, as long as it is compatible with the existence of an empty tree. For consistency we use the definition of multi-dimensional trees given below and write t as an expression over a special kind of “alphabet” containing just one symbol ρ for which any child structure is admissible.

Multi-dimensional trees (own definition)

Let Σ_t^d for $d \geq 1$ be the set of all symbols associated with t and Σ^0 a set of constant symbols. The set $\mathbb{T}_{\Sigma^d}$ of all d-dimensional trees can be defined inductively as follows:

Definition 10 *Let ε^d be the empty d-dimensional tree. Then*

- $\mathbb{T}_{\Sigma^0} := \{\varepsilon^0\} \cup \Sigma^0$*, and*
- *for $d \geq 1$: $\mathbb{T}_{\Sigma^d}$ is the smallest set such that $\varepsilon^d \in \mathbb{T}_{\Sigma^d}$ and $f[t_1, \ldots, t_n]_t \in \mathbb{T}_{\Sigma^d}$ for every $f \in \Sigma_t^d$, n the number of nodes in t, $t_1, \ldots, t_n \in \mathbb{T}_{\Sigma^d}$ and $t_1, \ldots, t_n$ are rooted breadth-first in that order at the nodes of t.*

Multi-dimensional trees – terminology

For some tree $t_p = f[t_1, \ldots, t_n]_t$ with $f \in \Sigma^d_t$, $t_1, \ldots, t_n$ are the direct subtrees of the tree, and the rest of the usual tree terminology can be applied in a similar manner. Also, for some fixed d, let $\Box$ be a special symbol associated with ε^{d-1} (leaf label). A tree $c \in \mathbb{T}_{\Sigma^d \cup \{\Box\}}$ in which $\Box$ occurs exactly once is still called a context, and $c[[s]]$ for $c \in C_{\Sigma^d}$ and $s \in \mathbb{T}_{\Sigma^d}$ is defined via substitution as before.

(Multi-dimensional finite-state tree automata)

Definition 11 *A (total, deterministic) finite-state d-dimensional tree automaton is a quadruple $\mathcal{A}^d = (\Sigma^d, Q, \delta, F)$ with input alphabet Σ^d, finite set of states Q, set of accepting states $F \subseteq Q$ and transition function δ with $\delta(t(q_1, \ldots, q_n), f) \in Q$ for every $f \in \Sigma_t^d$ where $t(q_1, \ldots, q_n)$ encodes the assignment of states to the nodes of t (i.e., $t(q_1, \ldots, q_n)$ is isomorphic to t and its nodes are labeled with $q_1, \ldots, q_n$ breadth-first in that order).*
$\delta : \mathbb{T}_{\Sigma^d} \longrightarrow Q$ is defined such that if $t_p = f[t_1, \ldots, t_n]_t \in \mathbb{T}_{\Sigma^d}$ then $\delta(t_p) = \delta(t(\delta(t_1), \ldots, \delta(t_n)), f)$.

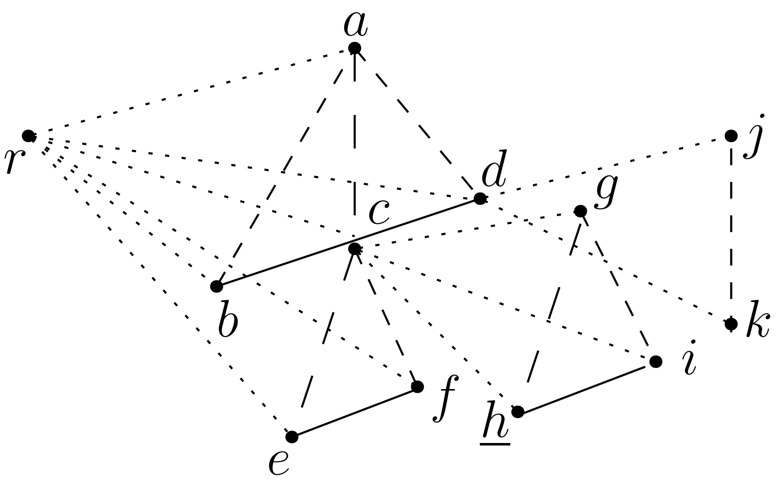
The set of trees accepted by $\mathcal{A}^d$ is $L(\mathcal{A}^d) = \{t_p \in \mathbb{T}_{\Sigma^d} | \delta(t_p) \in F\}$.

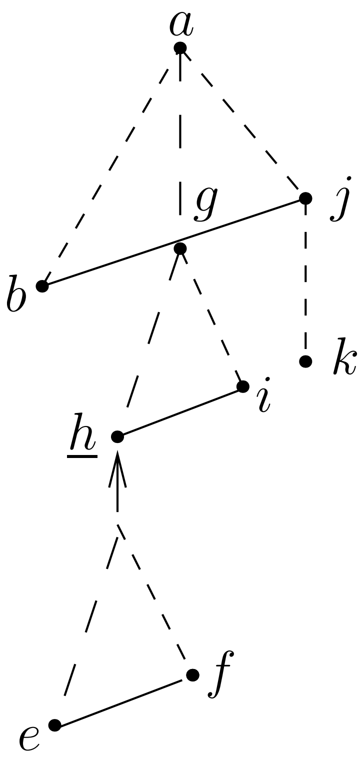
Multi-dimensional trees (own definition)

With the term representation and the adapted definitions of contexts and automata given in this section, results pertaining to the class of regular string or tree languages as for instance the Myhill-Nerode theorem or the Pumping lemma and all their consequences (like the existence of a unique minimal finite-state automaton $\mathcal{A}_L^d$ recognizing L for every recognizable d-dimensional tree language L) carry over directly to multi-dimensional trees.

(Multi-dimensional trees – yield)

As for $d \geq 3$ the yield is not unambiguous, the structures have to be enriched with additional information.





(Multi-dimensional trees – yield)

Assume that, for $d \geq 2$, in every tree $t_p \in \mathbb{T}_{\Sigma^d}$ every labeling symbol $f \in \Sigma^d$ is indexed with a set $S \subseteq \{2, \dots, d\}$. If $x \in S$ then we call a node labeled by f_S a *foot node for the $(x-1)$-dimensional yield of* t_p. For every subtree t_q of t_p:

(1) If t_q has depth 0 the index set in its root label must contain d, otherwise $t_q = f_S[t_1, \dots, t_n]_t$ with $f \in \Sigma^d_t$, $S \subseteq \{2, \dots, d\}$, and $t_1, \dots, t_n \in \mathbb{T}_{\Sigma^d}$ must have exactly one direct subtree $t_i \in \{t_1, \dots, t_n\}$ whose root labeling symbol is indexed with a set containing d and this subtree is attached to a *leaf* in t. In both cases, we will refer to that root as the d-dimensional foot node of t_q.

(2) Foot nodes are distributed such that for every n-dimensional yield of t_p with $n < d$, condition (1) is fulfilled as well.

(Multi-dimensional trees – yield)

For $d \geq 2$, the yield of a tree $t_p \in \mathbb{T}_{\Sigma^d}$ is defined as

$$yd_{d-1}(t_p) = \begin{cases} \varepsilon^{d-1} & \text{for } t_p = \varepsilon^d, \\ a_S & \text{for } t_p = a_S \text{ with } a \in \Sigma^d_{\varepsilon^{d-1}} \text{ and } S \subseteq \{2, \dots, d\}, \\ op_{t_p}(t_1) & \text{for } t_p = f_S[t_1, \dots, t_n]_t \text{ with } t_1, \dots, t_n \in \mathbb{T}_{\Sigma^d},\ f \in \Sigma^d_t, \\ & t \neq \varepsilon^{d-1}, \text{ and } S \subseteq \{2, \dots, d\}, \end{cases}$$

where $op_{t_p}(t_i)$ for $t_i \in \{t_1, \dots, t_n\}$ is defined as the $(d-1)$-dimensional tree obtained by replacing the d-dimensional foot node of t_i in $yd_{d-1}(t_i)$ by $e_R[op_{t_p}(t_j), \dots, op_{t_p}(t_k)]_{t_x}$, where e_R with $e \in \Sigma^d$ and $R \subseteq \{2, \dots, d\}$ is the label of the foot node, t_x is the $(d-2)$-dimensional child structure of the node at which t_i is attached in t and $t_j, \dots, t_k$ are the direct subtrees of t_p that are attached (breadth-first in that order) at the nodes of t_x.

The result $yd_{d-1}(t_p)$ is a $(d-1)$-dimensional tree over an alphabet Σ^{d-1} containing at least all the node labels in $yd_{d-1}(t_p)$, each associated at least with the child structures it occurs with. Obviously, the string yield of a d-dimensional tree for $d \geq 2$ can be obtained by taking the direct yield $d-1$ times.

Bibliography

References

[1] F. Gécseg, M. Steinby: Tree automata. Akademiai Kiado (1984)

[2] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications. Available on: www.grappa.univ-lille3.fr/tata (2005)