Algebraic Refinement of the Chomsky Hierarchy

Uwe Mönnich
Universität Tübingen
Wilhelmstr. 113
72074 Tübingen
Germany
um@sfs.nphil.uni-tuebingen.de

Lisbon, 1993

These notes constitute a brief account of some parts of the algebraic language theory. I have kept to the notation and terminology as presented in Engelfriet and Schmidt (1977) and Engelfriet and Schmidt (1978). Where appropriate, I have not refrained from following their exposition word for word.

Abstract Algebra exemplifies the view that mathematics is the study of *structure*.

Algebra is about *calculation*. Reflected in its ancient origins and name.

Calculation can be formulated using a family

$$A = \langle A_s : s \in S \rangle$$

of sets of data, containing a collection F_0 of distinguished elements of the form

$$a \in A_s$$

for some $s \in S$ and a collection F of basic functions of the form

$$f : A_{s_1} \times \ldots A_{s_n} \to A_s$$

for some $s_1, \ldots, s_n \in S$. These ingredients are called **carriers**, **constants** and **operations**, respectively. Together, they constitute a **many–sorted algebra** where S represents the family of **sorts**.

1 Example

$$S = \{1\} \quad F_0 = \{\varepsilon\} \quad F = \{\frown\}$$

$$\frown : V^* \times V^* \to V^*$$

Single–*sorted algebra of* strings *over an alphabet* V. ε *is the empty string and* $\frown$ *stands for concatenation.*

Signatures and algebras

The fundamental notion of a **many–sorted signature** Σ fixes a formal (syntactic) notation for specifying basic components of an algebra $\mathfrak{A}$, i.e., sets, constants and algebraic operations.

2 Definition

A **ranked alphabet** *or single–sorted signature is an indexed family of disjoint sets* $\langle \Sigma_n \rangle_n$.

A symbol f *in* Σ_n *is an* **operator of rank** n. *If* $n = 0$, *then* f *is called a* **constant**.

3 Examples

1. $\Sigma_0 = \{\varepsilon\} \quad \Sigma_2 = \{\frown\}$
   Single–sorted signature of semi–groups.

2. $\Sigma_0 = \{\varepsilon\} \quad \Sigma_1 = \{a \mid a \in V\}$
   Single–sorted signature of a monadic *algebra.*

3. $\Sigma_2 = \{\wedge, \vee\}$
   Single–sorted signature of lattices.

4 Definition

For a ranked alphabet Σ, the set of **trees** *or* **terms** *over Σ is defined to be the smallest set T_Σ such that the following conditions are satisfied:*

1. $\Sigma_0 \subseteq T_\Sigma$

2. *If $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$ then*

$$f(t_1, \ldots, t_n) \in T_\Sigma.$$

We call a subset of T_Σ a **tree language** *over Σ.*

Comment

Terms over the semi–group signature do *not* correspond to sentence strings in the sense of traditional formal language theory.

Take $\Sigma_0 = \{\varepsilon, a, b, c, d\}$ and $\Sigma_2 = \{\frown\}$. The string aab does not form an element of T_Σ. Related members of aab in T_Σ are $(a\sigma)$, $\frown(a, \frown(a, b))$, $\frown(\frown(a, a), b))$, $\frown(\varepsilon, \frown(a, \frown(a, b)))$, $\frown(\frown(\frown(a, a), b), \varepsilon)$, $\ldots$

Context–free languages, e.g., are specified by equations between (variable) terms over Σ, but they are interpreted in Σ_0^*.

5 Definition

For a set of symbols X, disjoint from Σ, $T_\Sigma(X)$ denotes the set of terms $T_{\Sigma(X)}$, where $\Sigma(X)$ is the ranked alphabet with

$$\Sigma(X)_0 = \Sigma_0 \cup X$$

and

$$\Sigma(X)_n = \Sigma_n$$

for $n \geq 1$.

Notation

We will use interchangeably the subscripted T_Σ ($T_{\Sigma(X)}$) and the parenthesized notation $T(\Sigma)$ ($T(\Sigma, X)$) for the set of terms over Σ (and X).

We are now in a position to introduce the important notion of a context–free tree grammar.

6 Definition

A **Context–free tree grammar** *is a triple* $G = \langle \Sigma, \mathcal{F}, P \rangle$*, where* Σ *is a finite ranked alphabet of* terminals, $\mathcal{F}$ *is a finite ranked alphabet of* nonterminals *disjoint from* Σ *and* P *is a finite set of rules of the form*

$$F(x_1, \ldots, x_m) \to t \qquad (n \geq 0)$$

where $F \in \mathcal{F}_m$ *and* $t \in T(\Sigma \cup \mathcal{F}, X_m)$*.**

*X is here assumed to be a denumerable set of variables $X = \{x_1, x_2, \ldots\}$. $X_0 = \emptyset$, $X_m = \{x_1, \ldots, x_m\}$.

7 Definition

Let now $t_1, t_2 \in T(\Sigma \cup \mathcal{F}, X_m)$*.* t_2 *is* **directly derivable** *from* t_1

$$t_1 \Rightarrow t_2$$

if there is a tree $u \in T(\Sigma \cup \mathcal{F}, X_{n+1})$ *containing exactly* one *occurrence of* x_{n+1}*, a rule* $F(x_1, \ldots, x_m) \to t$ *, and trees* $s_1, \ldots, s_m \in T(\Sigma \cup \mathcal{F}, X_n)$ *such that*

$$t_1 = u[x_1, \ldots, x_n, F(s_1, \ldots, s_m)]$$

$$t_2 = u[x_1, \ldots, x_n, t[s_1, \ldots, s_m]]$$

t_2 *is obtained from* t_1 *by replacing an occurrence of a subtree* $F(s_1, \ldots, s_m)$ *by the tree* $t[s_1, \ldots, s_m]$*.*

For $m \geq 0$ *and* $n \geq 0$*,* $t \in T(\Sigma, X_m)$ *and* $t_1, \ldots, t_m \in T(\Sigma, X_n)$

$$t[t_1, \ldots, t_m]$$

denotes the result of **substituting** t_i *for* x_i *in* t*. Observe that* $t[t_1, \ldots, t_m]$ *is in* $T(\Sigma, X_n)$*.*

As is customary $\stackrel{*}{\Rightarrow}$ *denotes the transitive-reflexive closure of* $\Rightarrow$*.*

8 Definition

For $m \geq 0$ *and* $s \in T(\Sigma \cup \mathcal{F}, X_m)$ *we define*

$$\mathcal{L}(G, s) = \{t \in T_\Sigma(X_m) \mid s \overset{*}{\Rightarrow} t\}$$

to be the **context–free tree language** *generated by* G *from* s.*

*Note that what we have introduced is the *unrestricted* mode of derivation. By imposing conditions on the form of the s_i and u in the definition of a direct derivation the resulting classes of languages generated by G vary with these conditions.

9 Example

$$\begin{array}{lcl@{\qquad}lcl} \Sigma_0 & = & \{\varepsilon\} & \Sigma_1 & = & \{a, b, c, d\} \\ \mathcal{F}_0 & = & \{s\} & \mathcal{F}_n & = & \emptyset \textit{ for } n \geq 1 \end{array}$$

$$P = \{s \to \varepsilon \mid a(s) \mid b(s) \mid c(s) \mid d(s)\}$$

$$\mathcal{L}(G, s) = \Sigma_1^*$$

where we have identified $T(\Sigma)$ *with* Σ_1^*.†

As should be obvious from this example, *regular grammars* are context–free tree grammars over a *monadic* signature with $\mathcal{F}_n = \emptyset$ for $n \neq 0$.

†

$$\psi : \Sigma_1^* \to T(\Sigma)$$

$$\begin{array}{lcl} \varepsilon & \longmapsto & \varepsilon \\ w & \longmapsto & w_1(w_2(\ldots w_m(\varepsilon)\ldots)) \end{array}$$

10 Example

$$\begin{array}{rcl} \Sigma_0 & = & \{\varepsilon, a, b\} \qquad \Sigma_2 \;=\; \{\frown\} \\ \mathcal{F}_0 & = & \{s\} \end{array}$$

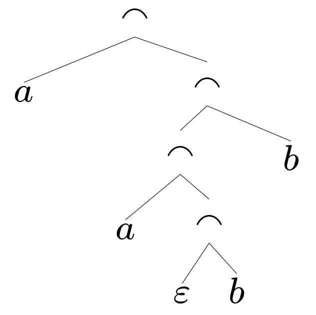
$$P = \{s \to \varepsilon \,|\, \frown(a, \frown(s, b))\}$$

$$\mathcal{L}(G, s) = \{\frown(a, \frown(\ldots, \frown(\varepsilon, b) \ldots b) \ldots)\}$$

where n occurrences of a precede the same number of occurrences of b for $n \geq 0$.

As indicated above this language is not to be confused with $\mathcal{L}' = \{a^n b^n \,|\, n \geq 0\}$.

But look at a typical element of $\mathcal{L}$ in tree notation:



By identifying ε with the empty string and interpreting $\frown$ as concatenation, we can send every term to the string of its leaves. In technical parlance to be introduced below, this is the unique homomorphism from the initial algebra $T(\Sigma)$ to the semi–group with identity over Σ_0^*.

Again it should be clear from this example that the traditional *context–free string* languages are just homomorphic images of *context–free tree* languages generated by context–free tree grammars over the semi–group signature enriched by a finite set of constants in Σ_0. These tree grammars must, of course, satisfy the condition that $\mathcal{F}_n = \emptyset$ for $n \neq 0$.

11 Example

$$\begin{array}{lcl} \Sigma_0 & = & \{\varepsilon, a, b, c, d\} \\ \mathcal{F}_0 & = & \{s\} \end{array} \qquad \begin{array}{lcl} \Sigma_2 & = & \{\frown\} \\ \mathcal{F}_4 & = & \{F\} \end{array}$$

$$P = \left\{ \begin{array}{lll} s & \to & F(a, b, c, d)\,, \\ F(x_1, x_2, x_3, x_4) & \to & \varepsilon \mid \\ & & \frown(\frown(\frown(x_1, x_2), x_3), x_4) \mid \\ & & F(\frown(a, x_1), \frown(b, x_2), \frown(c, x_3), \frown(d, x_4)) \end{array} \right\}$$

$\mathcal{L}(G, s) =$
$\{\frown(\frown(\frown(\frown(a, ...)...), \frown(b, ...)...), \frown(c, ...)...), \frown(d, ...)...))\}$

Again, the number of occurrences of a's, b's, c's and d's, has to be the same. By taking the frontier of the tree terms, we get the language $\mathcal{L}' = \{a^n b^n c^n d^n\}$.

12 Example

$$\begin{array}{lcl} \Sigma_0 & = & \{\varepsilon, a, b, c, d\} \\ \mathcal{F}_0 & = & \{s\} \end{array} \qquad \begin{array}{lcl} \Sigma_2 & = & \{\frown\} \\ \mathcal{F}_4 & = & \{F\} \end{array}$$

$$P = \left\{ \begin{array}{lll} s & \to & F(a, \varepsilon, c, \varepsilon) \mid F(\varepsilon, b, \varepsilon, d) \mid \varepsilon, \\ F(x_1, x_2, x_3, x_4) & \to & \\ & & F(\frown(a, x_1), x_2, \frown(c, x_3), x_4) \mid \\ & & F(x_1, \frown(b, x_2), x_3, \frown(d, x_4)) \mid \\ & & \frown(\frown(\frown(x_1, x_2), x_3), x_4) \end{array} \right\}$$

$\mathcal{L}(G, s) =$
$\{\frown(\frown(\frown(\frown(a, ...)...), \frown(b, ...)...), \frown(c, ...)...), \frown(d, ...)...))\}$

Again, the number of occurrences of a's and c's and of b's and d's, respectively, has to be the same. By taking the frontier of the tree terms, we get the language $\mathcal{L}' = \{a^n b^m c^n d^m\}$.

This last language has played some role in the discussion about the question whether natural languages are context–free (Partee, ter Meulen and Wall 1990, p. 503). Structures of this abstract form seem to be exemplified by *Swiss German*. Swiss German allows cross–serial order in dependent clauses:

13 Example

(a) Jan säit das mer em Hans es huus hälfed aastriiche.

(b) John said that we helped Hans (to) paint the house.

(c) Jan säit das mer d'chind em Hans es huus lönd hälfed aastriiche.

(d) John said that we let the children help Hans paint the house.

The NP's and the V's of which the NP's are objects occur in cross–serial order. *D'chind* is the object of *lönd*, *em Hans* is the object of *hälfe*, and *es huus* is the object of *aastriiche*. Furthermore the verbs mark their objects for case: *hälfe* requires dative case, while *lönd* and *aastriiche* require accusative. It appears that there are no limits on the length of such constructions in grammatical sentences of Swiss German.

Shieber intersects Swiss German with the regular language:

14 Example

(a) Jan säit das mer (d'chind)* (em Hans)* händ wele (laa)* (hälfe)* aastriiche.

(b) John said that we (the children) (Hans)* the house wanted to (let)* (help)* paint.*

The result of the intersection looks as follows:

15 Example

Jan säit das mer (d'chind)^n (em Hans)^m händ wele (laa)^n (hälfe)^m aastriiche.

As is well known, this language is not context–free.

But only a *mild* extension of the *generative power* of context–free grammars is necessary to accommodate such constructions. One way to do this is to employ the concept of **indexed grammars** (Partee et al. 1990, pp. 536). Indexed grammars differ from context–free grammars in that nonterminals may carry a sequence of indices and their productions allow for adding, removing and distributing these indices in the course of a derivation. The following formulation, due to Hopcroft and Ullman (1979), distinguishes three kinds of rules:

16 Rule Types

(a) $A \to \alpha$

Ordinary context–free productions, if A carries any indices, they are copied onto each of the nonterminals in α.

(b) $A[i] \to \alpha$

This production may only be applied if A bears an i on the left of its index sequence. This index sequence with i removed is again copied onto each of the nonterminals in α.

(c) $C \to B[i]$

This rule adds i at the left of the index sequence already present on A and copies the result onto B.

17 Example

$$G = \langle V_T, V_N, I, S, P \rangle$$

$$V_T = \{a, b, c, d\}$$

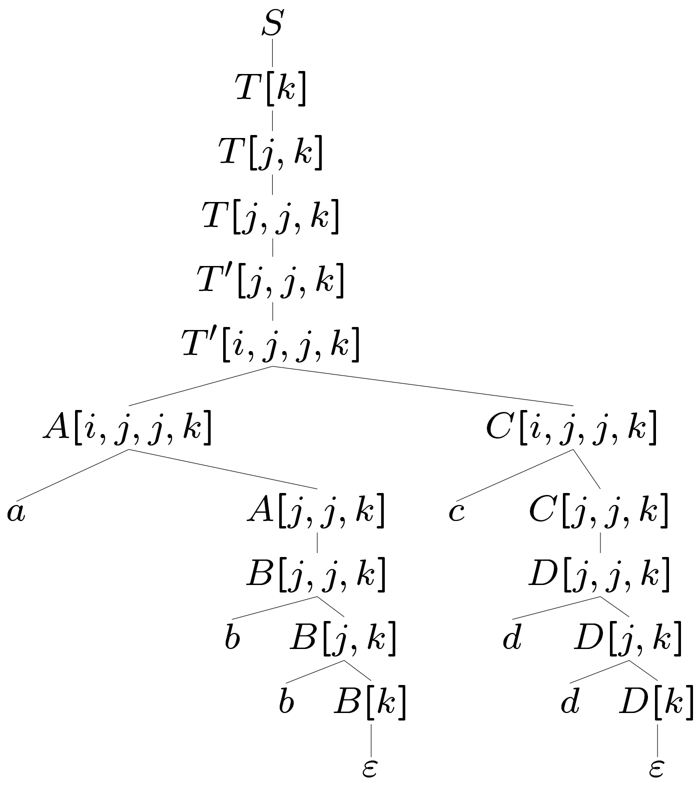
$$V_N = \{S, T, T', A, B, C, D\}$$

$$I = \{i, j, k\}$$

$$P = \left\{ \begin{array}{llllllll} S & \to & T[k] & , & T & \to & T[j] & , \\ T & \to & T' & , & T' & \to & T'[i] & , \\ T' & \to & AC & , & A[i] & \to & aA & , \\ A & \to & B & , & B[j] & \to & bB & , \\ B[k] & \to & \varepsilon & , & C[i] & \to & cC & , \\ C & \to & D & , & D[j] & \to & dD & , \\ D[k] & \to & \varepsilon & & & & & \end{array} \right\}$$

18 Derivation Tree

A sample derivation tree for ab^2cd^2 *looks as follows:*



- There are no known linguistic phenomena which lie beyond the (weak) generative capacity of indexed grammars.

- Indexed grammars have effective parsing algorithms.

- Regular, context–free and indexed languages form the first three steps of an infinite language hierarchy.

In order to be able to formulate this result more precisely, we need the notion of a **many–sorted alphabet** (many–sorted signature, many–sorted operator domain).

19 Definition

Let S be a set of sorts *(categories). A* **many–sorted alphabet** Σ *is an indexed family*

$$\langle \Sigma_{w,s} \mid w \in S^*, s \in S \rangle$$

of disjoint sets. A symbol in $\Sigma_{w,s}$ is called an **operator** *of* **type** $< w, s >$, **arity** w, **sort** s *and* **rank** $l(w)$. *If $w = \varepsilon$ then $f \in \Sigma_{\varepsilon,s}$ is called a* **constant** *of sort s.*

Note that a ranked alphabet can be identified with an S–sorted alphabet where $S = \{s\}$. The set $\Sigma_{s^n,s}$ is then the same as Σ_n.

20 Definition

Let Σ be a ranked alphabet. The **derived** $\mathbb{N}$**–sorted alphabet** *of Σ, designated by $D(\Sigma)$, is defined as follows. For each $n \geq 0$, $\Sigma'_n = \{f' \mid f \in \Sigma_n\}$ is a new set of symbols. For each $n \geq 1$ and each i, $1 \leq i \leq n$, π^n_i is a new symbol, the ith projection symbol of sort n, and for each $n \geq 0$, $k \geq 0$ the new symbol $c_{n,k}$ is called the (n,k)th composition symbol. The $\mathbb{N}$–sorted alphabet $D(\Sigma)$ consists then of the following sets of operators:*

1. $D_{\varepsilon,0} = \Sigma'_0$
2. $D_{\varepsilon,n} = \Sigma'_n \cup \{\pi^n_i \mid 1 \leq i \leq n\}$ *for* $n \geq 1$
3. $D_{nk^n,k} = \{c_{n,k}\}$ *for* $n, k \geq 0$
4. $D_{w,s} = \emptyset$ *otherwise*

In the way of explanation, let us note that the c's will be interpreted as composition of the (interpreted) elements of Σ and the π's as projections.

We have to generalize the notions of trees (terms) and variable trees to the case of a many– sorted operator domain.

21 Definition

For a many–sorted operator domain Σ, we denote by $T(\Sigma)$ the family

$$\langle T(\Sigma, s) \mid s \in S \rangle$$

of **trees of sort** s *over* Σ*.* $T(\Sigma, s)$ *is inductively defined as follows:*

1. *For each sort $s \in S$*

$$\Sigma_{\varepsilon,s} \subseteq T(\Sigma, s)$$

2. *For $n \geq 1$ and $s \in S$, $w \in S^*$, if $f \in \Sigma_{w,s}$ and for $1 \geq i \geq n$, $t_i \in T(\Sigma, w_i)$, $l(w) = n$,*

$$f(t_1, \ldots, t_n) \in T(\Sigma, s)$$

22 Definition

For a many–sorted operator domain Σ and a family of disjoint sets $Y = \langle Y_s \,|\, s \in S \rangle$, the family $T(\Sigma, Y)$ is defined to be $T(\Sigma(Y))$, where $\Sigma(Y)$ is the many–sorted alphabet with

$$\Sigma(Y)_{\varepsilon,s} = \Sigma_{\varepsilon,s} \cup Y_s$$

and for $w \neq \varepsilon$,

$$\Sigma(Y)_{w,s} = \Sigma_{w,s}.$$

23 Example

Let $D(\Sigma)$ be the derived alphabet of the monadic signature Σ that was introduced in example 3.

$$D_{\varepsilon,0} = \{\varepsilon\} \qquad D_{\varepsilon,1} = \{a, b, c, d\}$$

Consider the following context–free tree grammar G *over the original* monadic *alphabet:*

$$\mathcal{F}_0 \;=\; \{s\} \qquad \mathcal{F}_1 \;=\; \{F\}$$

$$P = \left\{ \begin{array}{lcl} s & \to & F(\varepsilon) \,|\, \varepsilon, \\ F(x) & \to & a(F(b(x))) \,|\, a(b(x)) \end{array} \right\}$$

$$\mathcal{L}(G, s) = \{\overbrace{a(a\ldots}^{n}\overbrace{(b(b\ldots}^{n}(\varepsilon)\ldots)\}$$

This is *not* a regular language.

By employing a monadic nonterminal in the rules of our grammar G we have left the realm of finite–state–languages. Observe the similarity of this grammar G to the traditional context–free *string* grammar in (10) where the decisive production had the following form:

$$s \rightarrow a\, s\, b$$

Let us now translate this context–free tree grammar G over Σ into a context–free tree grammar G' over $D(\Sigma)$ in which only nonterminals of arity 0 occur.

24 Example

$$\mathcal{F}_{\varepsilon,0} \;=\; \{s\} \qquad \mathcal{F}_{\varepsilon,1} \;=\; \{F\}$$

$$P' = \left\{ \begin{array}{lcl} s & \rightarrow & c_{1,0}(F,\varepsilon) \,|\, \varepsilon, \\ F(x) & \rightarrow & c_{1,1}(a, c_{1,1}(F, c_{1,1}(b, \pi^1_1))) \,| \\ & & c_{1,1}(a, c_{1,1}(b, \pi^1_1)) \end{array} \right\}$$

$$\mathcal{L}(G', s) = \{\overbrace{c_{1,0}(c_{1,1}(a, c_{1,1}(a...}^{n}\overbrace{c_{1,1}(b, c_{1,1}(b...c_{1,1}(b, \pi^1_1)}^{n}...))), \varepsilon)\}$$

This language looks again suspiciously similar to the 'semi-group' language introduced above:

$$\mathcal{L}(G, s) = \{{}^\frown(a, {}^\frown(\ldots, {}^\frown(\varepsilon, b)\ldots b)\ldots)\}$$

with equal occurrences of a's and b's.

25 Fact

Considered as string languages, the languages generated by context–free tree grammars over the monadic alphabet and the languages generated by regular ($\mathcal{F}_n = \emptyset$, for $n \neq 0$) context–free tree grammars over the derived alphabet are exactly the same.

We can even say what this class of string languages is. As may have been expected from the example grammars, this class is identical with the class of context–free languages:

$$\mathcal{L}_{var}(MON) = \mathcal{L}_{comp}(D(MON)) = CF$$

It should be obvious from looking at the form of grammars that 'variable' context-free tree grammars over a monadic alphabet and their 'regular' or 'constant' images over the derived alphabet generate the same string languages.

Their identity with the family of context–free string languages is also easily shown. We consider only the non–trivial inclusion:

Take an arbitrary context–free string grammar G. Transform it into its Chomsky Normal Form G'. Form the monadic signature Σ corresponding to the terminal vocabulary of G':

$$\Sigma_0 = \{\varepsilon\} \quad \Sigma_1 = \{V_T\}$$

where V_T is the terminal vocabulary of G'. We obtain then the context–free tree grammar $G'' = \langle \Sigma, \mathcal{F}, P \rangle$ which generates the same strings a s G' through the following steps:

1. $\mathcal{F}_0 = \{S\} \qquad \mathcal{F}_1 = \{A \mid A \in V_N\}$

2. P is obtained as follows:
   Replace

$$\begin{array}{llll}
S \to AB & \text{by} & S \to A(B(\varepsilon)) & \\
S \to a & \text{by} & S \to a(\varepsilon) & \\
S \to \varepsilon & \text{by} & S \to \varepsilon & \\
A \to BC & \text{by} & A(x) \to B(C(x)) & \text{for } A \neq S \\
A \to a & \text{by} & A \to a(x) & \text{for } A \neq S
\end{array}$$

$\mathcal{L}(G'', S) = \mathcal{L}(G')$ if we do not distinguish between strings and monadic trees.

That is, by considering only variable–free nonterminals the process of derivation with respect to the monadic alphabet corresponds to the first step in the Chomsky Hierarchy.

Let us record here that the process of translation between the 'variable grammar' over Σ and the 'compositional grammar' over $D(\Sigma)$ was just a special case of a general transformation.

26 Definition

Let Σ be a ranked alphabet. For $k \geq 0$,

$$COMB_k^\Sigma : T(\Sigma, X_k) \to T(D(\Sigma), k)$$

is defined as follows:

1. $COMB_k^\Sigma(x_i) = \pi_i^k$
2. $COMB_k^\Sigma(f) = c_{0,k}(f')$ *for* $f \in \Sigma_0$
3. $COMB_k^\Sigma(f(t_1, \ldots, t_n)) =$
   $c_{n,k}(f', COMB_k^\Sigma(t_1), \ldots, COMB_k^\Sigma(t_n))$
   for $f \in \Sigma_n$ $(n \geq 1)$

On several occasions we have used expressions like 'taking the frontier of a tree', 'concatenating the leaves of a tree', 'considering a tree language as a set of strings', etc. All this can be made precise by taking into account the interpretation of operator domains and the morphisms between them.

27 Definition

*Let Σ be an S–sorted operator domain. A Σ–***algebra** *(or many–sorted algebra) $\mathfrak{A}$ consists of a family*

$$\langle A_s \mid s \in S \rangle$$

of carriers *or* domains *and, for each $< w, s > \in S^* \times S$ and each $f \in \Sigma_{w,s}$, an operation*

$$f_A : A_{s_1} \times \cdots \times A_{s_n} \to A_s$$

where $w = s_1 s_2 \ldots s_n$. For $n = 0$, f_A is simply an element of A_s.

28 Example

Let Σ be a ranked alphabet. Σ_0^*, the set of strings of constants, can be made into a Σ–algebra $\mathfrak{A}$.

If $f \in \Sigma_0$ set $f_A = f$. For $n \geq 1$ define the operation f_A corresponding to $f \in \Sigma_n$ as follows. If $w_1, \ldots, w_n \in \Sigma_0^*$,

$$f_A(w_1, \ldots, w_n) = w_1, \ldots, w_n.$$

Every operator in Σ_n is thus interpreted as n–ary concatenation. In particular, $\frown \in \Sigma_2$ (for Σ the semi–group signature) can be regarded as binary concatenation.

29 Example

The family $\langle T(\Sigma, Y) \,|\, s \in S\rangle$ *is a* Σ*–algebra. (Variable) terms of sort* $s \in S$ $T(\Sigma(Y))_s$ *constitute the domain of the sort* s. *The operators* $f \in \Sigma_{w,s}$ *are interpreted as term building operations.*

$$f_{T(\Sigma(Y))}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$$

These term–algebras play a special role with respect to morphisms between algebras to be introduced in a moment.

30 Example

Let D *be the derived alphabet of a ranked alphabet* Σ. $DT_\Sigma(X)$ *denotes the following* D*–algebra:*

The domain of sort n *is* $T_\Sigma(X_n)$. *For* $f \in \Sigma_n$,

$$f'_{DT_\Sigma(X)} \text{ is the tree } f(t_1, \ldots, t_n).$$

For $n \geq 1$ *and* $1 \leq i \leq n$,

$$\pi^n_{i_{DT_\Sigma(X)}} = x_i.$$

For $n, k \geq 0$, $t \in T_\Sigma(X_n)$ *and* $t_1, \ldots, t_n \in T_\Sigma(X_K)$

$$c_{n,k_{DT_\Sigma(X)}}(t, t_1, \ldots, t_n) = t[t_1, \ldots, t_n].$$

$DT_\Sigma(X)$ *is called the* **tree substitution** *D–algebra.*

Morphisms between Σ–algebras are called Σ–homomorphisms. By sending elements from an algebra $\mathfrak{A}$ to an algebra $\mathfrak{B}$ they make it possible to consider the image of $\mathfrak{A}$ in $\mathfrak{B}$ in a structure preserving manner.

31 Definition

*Let $\mathfrak{A}$ and $\mathfrak{B}$ be Σ–algebras. A Σ–***homomorphism** *$h : \mathfrak{A} \to \mathfrak{B}$ is a family $\langle h_s \,|\, s \in S \rangle$ of mappings $h_s : A_s \to B_s$ such that the following holds:*

1. *for $f \in \Sigma_{\varepsilon,s}$*
$$h_s(f_A) = f_B$$

2. *for $f \in \Sigma_{w,s}$, $a_i \in A_{w_i}$ and $l(w) = n$,*
$$h_s(f_A(a_1, \dots, a_n)) = f_B(h_{w_1}(a_1), \dots, h_{w_n}(a_n))$$

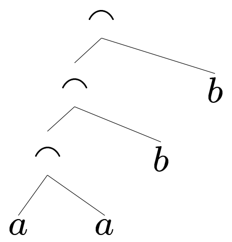
As was mentioned before, the tree algebra $T(\Sigma, X)$ gives rise to uniquely determined homomorphisms. Given a Σ–algebra $\mathfrak{A}$ and a family $h_s : Y_s \to A_s$, there is a unique Σ–homomorphism $\widehat{h} : T(\Sigma, X) \to \mathfrak{A}$ which extends the h_s.

Particular case: $Y = \emptyset$!

32 Example

Let Σ be a ranked alphabet. Consider again the Σ–algebra Σ_0^* of finite strings of constants. The unique Σ–homomorphism from T_Σ to Σ_0^* maps each tree in T_Σ to its frontier in Σ_0^*.

This notion of unique homomorphism makes precise the relation between 'semi–group terms' like $\frown(\frown(\frown(a,a),b),b)$ or



and their string image $aabb$.

In the case of the monadic signature this mapping would transform all trees into the empty word ε.

$$\begin{array}{c} a \\ | \\ a \\ | \\ b \\ | \\ b \\ | \\ \varepsilon \end{array} \longmapsto \varepsilon .$$

Since we do not distinguish between Σ_0^* and the tree terms of a monadic signature, i.e., between $a(a(b(b(\varepsilon))))$ where $a, b \in \Sigma_1$ and $aabb \in \Sigma_0^*$, this situation is without negative impact.

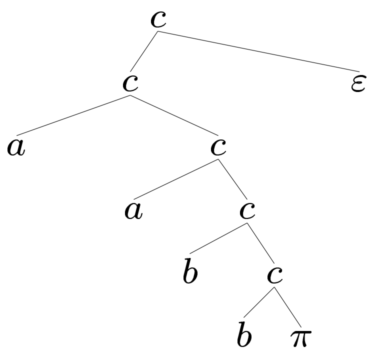
33 Example

We need a more refrained notion of frontier or yield homomorphism to tackle the case of derived alphabets. The unique homomorphism from T_D to $DT_\Sigma(X)$ associates with each tree in $T_{D,n}$ a Σ–tree with variables in $T(\Sigma, X_n)$.

If we return to the case of the regular grammar over the derived signature $D(MON)$, i.e., the signature derived from the monadic alphabet, we can now explain in what sense the monadic trees are to be considered as homomorphic images of 'compositional' trees. Recall that the language generated by the grammar in question contains trees of the following form:

$$\overbrace{c(a, c(a \ldots}^{n} \overbrace{c(b, c(b \ldots c(b, \pi)}^{n} \ldots), \varepsilon)$$

As a concrete example look at the following tree



This is a tree of sort $< \varepsilon, 0 >$ in the set $T_{D(\Sigma)}$. Under the uniquely determined homomorphism from $T_{D(\Sigma)}$ into the tree substitution algebra $DT_{\Sigma}(X)$ every tree of sort $< \varepsilon, 0 >$ in $T_{D(\Sigma)}$ corresponds to a tree in T_{Σ}. This means for our particular example that the tree displayed above is mapped to

$$a(a(b(b(\varepsilon))))$$

We are now in a position to present a more precise formulation of the fact concerning context–free string languages.

34 Fact

Let $\mathcal{L}$ be a context-free string language over the alphabet Σ, i.e., $\mathcal{L} \subseteq \Sigma^*$, and let Σ' be the monadic alphabet corresponding to it. Then there exists a set of trees $\mathcal{L}'$ over the derived alphabet $D(\Sigma')$ generated by a regular grammar such that

$$\psi(L) = Yield(L')$$

where $Yield$ is the unique homomorphism from $T_{D(\Sigma')}$ to $DT_{\Sigma'}(X)$ and ψ is a 1–1 correspondence between Σ^* and T_{Σ}:

$$\begin{array}{rcl} \psi(\varepsilon) & = & \varepsilon \\ \psi(w) & = & w_1(w_2(\ldots w_n(\varepsilon)\ldots)) \end{array}$$

Given that the process of forming a derived alphabet can be iterated (Exercise!), the next question which presents itself concerns the class of languages which are the $Yield^2$ of families of terms generated by regular grammars over the doubly derived monadic alphabet $D(D(\Sigma))$. Since we have seen that context–free tree grammars over Σ and regular grammars over $D(\Sigma)$ are but notational variants and since the same reasoning applies to the relation between context–free tree grammars over $D(\Sigma)$ and regular grammars over $D(D(\Sigma))$ (and, by the way, to the general case as well), the question above boils down to a question about the class of languages characterizable as $Yields$ of context–free tree grammars over $D(\Sigma)$.

35 Example (Maibaum)

$$\begin{array}{lclclcl} D_{\varepsilon,0} & = & \{\varepsilon\} & \qquad & D_{\varepsilon,1} & = & \{a\} \\ \mathcal{F}_{\varepsilon,0} & = & \{s\} & & \mathcal{F}_{\varepsilon,1} & = & \{B\} \end{array}$$

$$P = \left\{ \begin{array}{lcl} s & \to & c_{1,0}(a,\varepsilon) \mid c_{1,0}(B(a),\varepsilon), \\ B(x) & \to & B(c_{1,1}(x,x)) \mid c_{1,1}(x,x) \end{array} \right\}$$

$$\mathcal{L}(G', s) = \{c(a,\varepsilon), c(c(a,a),\varepsilon), c(c(c(a,a),c(a,a)),\varepsilon), \ldots\}$$

$$Yield(\mathcal{L}) = \{a^{2^n}\}$$

This language is not context–free.

36 Example (Formal Swiss German)

$$\begin{array}{lcl@{\qquad}lcl} D_{\varepsilon,0} & = & \{\varepsilon\} & D_{\varepsilon,1} & = & \{a,b,c,d\} \\ \mathcal{F}_{\varepsilon,0} & = & \{s\} & \mathcal{F}_{1111,1} & = & \{F\} \end{array}$$

$$P = \left\{ \begin{array}{lcl} s & \to & c_{1,0}(F(a,\pi,c,\pi),\varepsilon) \mid \\ & & c_{1,0}(F(\pi,b,\pi,d),\varepsilon) \mid \\ & & c_{1,0}(F(x_1,x_2,x_3,x_4),\varepsilon) \mid \varepsilon, \\ F(x_1,x_2,x_3,x_4) & \to & \\ \multicolumn{3}{l}{\quad F(c_{1,1}(a,x_1),x_2,c_{1,1}(c,x_3),x_4) \mid} \\ \multicolumn{3}{l}{\quad F(x_1,c_{1,1}(b,x_2),x_3,c_{1,1}(d,x_4)) \mid} \\ \multicolumn{3}{l}{\quad c_{1,1}(c_{1,1}(c_{1,1}(x_1,x_2),x_3),x_4)} \end{array} \right\}$$

$$Yield(\mathcal{L}) = \{a^n b^m c^n d^m\}$$

37 Fact

The languages characterizable as $Yields$ of sets of terms that are generated by context–free tree grammars over the derived monadic alphabet are exactly the indexed languages.

Summary

The languages generated by regular grammars over the monadic, the derived monadic and the doubly derived monadic alphabet – regarded as string languages – are the finite–state, the context–free and the indexed languages respectively. They form the first three steps in an infinite hierarchy. It is easy to specify a grammar G over the triply derived alphabet $D(D(D(\Sigma)))$ which regularly generates

$$Yield^3(\mathcal{L}(G)) = \{a^{2^{2^n}}\}$$

which is known not to be an indexed language.

Engelfriet, J. and Schmidt, E. M. (1977). IO and OI, I, *Comp. Syst. Sci.* **15**: 328 – 353.

Engelfriet, J. and Schmidt, E. M. (1978). IO and OI, II, *Comp. Syst. Sci.* **16**: 67 – 99.

Hopcroft, J. and Ullman, J. D. (1979). *Introduction to Automata Theory and Computation*, Addison Wesley, Reading, Mass.

Partee, B. H., ter Meulen, A. and Wall, R. E. (1990). *Mathematical Methods in Linguistics*, Vol. 30 of *Studies in Linguistics and Philosophy*, Kluwer Academic Publishers, Dordrecht/Boston/London.