Theoretical Computer Science 20 (1982) 95–207
North-Holland Publishing Company

THE IO- AND OI-HIERARCHIES

Werner DAMM
Lehrstuhl für Informatik II, RWTH Aachen, D-5100 Aachen, Fed. Rep. Germany

Communicated by M. Nivat
Received December 1979
Revised September 1981

Abstract. An analysis of recursive procedures in *ALGOL 68 with finite modes* shows, that a denotational semantics of this language can be described on the level of program schemes using a *typed λ-calculus with fixed-point operators.* In the first part of this paper, we derive classical schematological theorems for the resulting class of *level-n schemes.* In part two, we investigate the language families obtained by *call-by-value* and *call-by-name* interpretation of level-*n* schemes over the algebra of formal languages. It is proved, that differentiating according to the functional level of recursion leads to two infinite hierarchies of recursive languages, the *IO-* and *OI-hierarchies,* which can be characterized as canonical extensions of the regular, context-free, and IO- and OI-macro languages, respectively. Sufficient conditions are derived to establish strictness of *IO-like hierarchies.* Finally we derive, that *recursion on higher types* induces an infinite hierarchy of control structures by proving that level-*n* schemes are strictly less powerful than level-*n* + 1 schemes.

Introduction

During the last ten years, tree language theory has established its relevance as an important tool for the analysis of structured objects and the structure of computations and as such has had profound influences both within the area of formal semantics and within string language theory. As some examples of the first connection let us mention different methods for specifying programming language semantics on the derivation trees of programs (such as denotational semantics, initial algebra semantics, attribute grammars, syntax-directed translation), representing the meaning of programs by infinite trees through Mezei–Wright-like result (a denotation introduced in [28] for theorems, which allow to lift solutions of equations to a syntactic level, see [45, 32, 53]), and modelling operational semantics as derivations in tree grammars (as in [52, 53, 8, 67]). Note that all applications allow to lift properties of programs to the level of tree languages.

The close relation between string- and tree language theory can best be described using Fig. 1.

To begin with, let us note that string language theory can be viewed as a subtheory of tree language theory by identifying strings with *monadic trees* (where each node has at most one son). Generalizing from the monadic to

0304-3975/82/0000–0000/$02.75

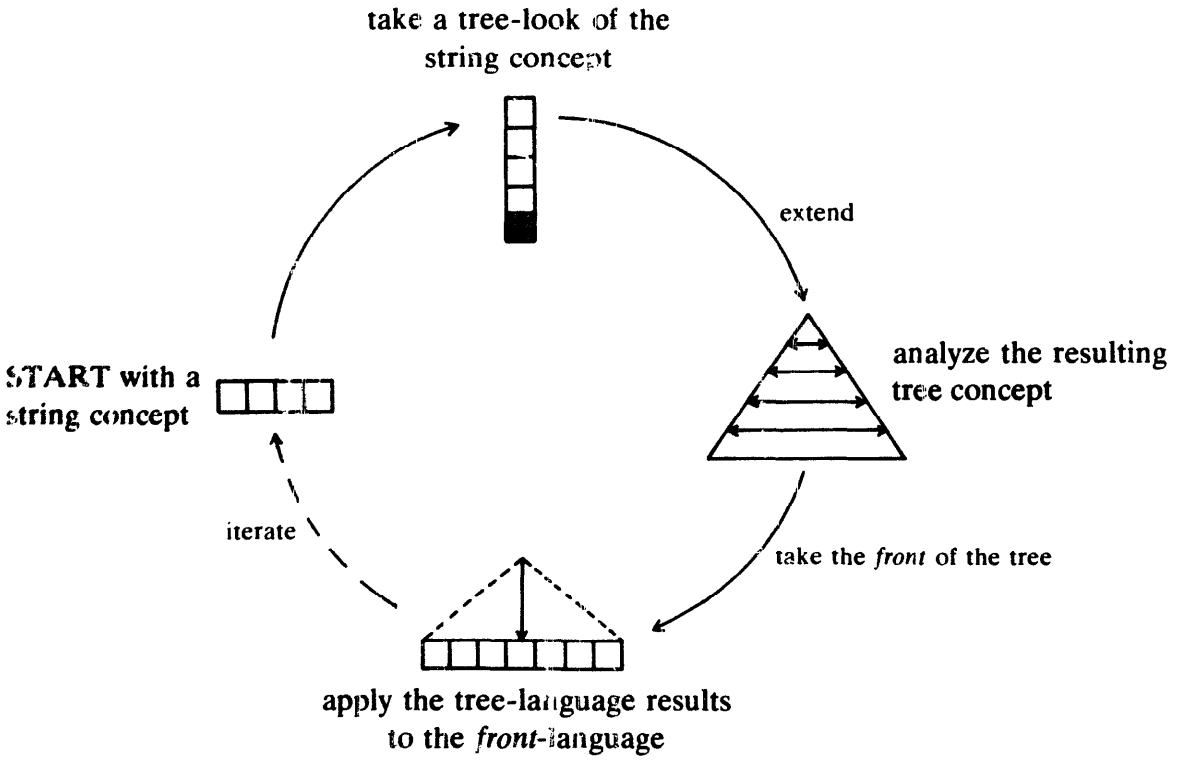


Fig. 1.

the *ranked* case (where symbols of arbitrary finite arity are allowed to construct trees) gives a canonical way of defining tree concepts out of string concepts. As two examples observe, that regular-tree grammars have the characteristic feature, that nonterminals may only appear as leaves (which is clearly the canonical generalization of right-linear grammars), whereas context-free tree grammars allow nonterminals of arbitrary ranks in the right-hand sides of the productions. In many cases, this process 'preserves proofs', i.e. substituting the tree concept for the 'generating' string concept establishes a generalization of the corresponding result to trees (perhaps the most prominent counterexample is the tree-transducer hierarchy resulting from nonclosure of top-down tree automata – the generalization of gsm's – under composition, see [27]). This again may be used to obtain simplified proofs for string-theoretic results by lifting string-theory via the *front*-operation to theorems on the corresponding class of derivation trees. In general, both the *front*- (or *yield*-) and the *branch* (or *path*-) operation allow to go in a uniform way from trees to strings.

The striking point of the above diagram, however, is the suggestion to *iterate* this process, producing in a canonical way increasingly complex classes of languages. Here we investigate the *IO-* and *OI-hierarchies* obtained by starting the iteration with right-linear grammars. It is well known that one iteration turns regular into context-free languages, while the next iteration yields the IO- or OI-macro languages [30], depending on whether we generalize rightmost or leftmost deriva-

tions in context-free grammars (the classes of tree languages generated by context-free tree grammars in *innermost–outermost* and *outermost–innermost* mode are incomparable [30]). In this paper we prove both hierarchies to be infinite and derive conditions on the input language-family to establish strictness of the resulting *IO-like hierarchy*. This gives a positive answer to conjectures in [35, 28, 70, 42] and a correct proof of the strictness-result stated in [66]. In preparing these results we will derive a number of 'classical' properties such as fixed-point characterizations, normalform-theorems, structural theorems, decidability of emptiness and membership and closure under intersection with regular languages. Together with the facts, that the language-families in the OI-hierarchy form substitution closed AFL's and can be characterized as languages accepted by *level-n pushdown automata* (introduced in [44]; this characterization is proved in [21]), these results seem to substantiate the claim raised by Wand [70], that the OI-hierarchy forms *the* natural extension of the Chomsky-hierarchy.

We will now indicate, how these results relate to semantics of programming languages (and thus work out the abovementioned connection for this particular example). In fact, this paper was motivated from the problem within formal semantics to analyse the impact of *higher type procedures* (discussed in more detail in the following paragraph) on the definability of new objects of base type.

One characteristic of the type discipline in ALGOL 68 is the fact that the type of a procedure (i.e. the type of its parameters and the type of its value, if any) has to be completely specified. Types are defined in *mode declarations*, which in particular allow the definition of recursive (more precisely: regular) types, using *base types* such as **bool**, **int**, **char**, ... and *constructors* as **proc**, ... Thus the type of a procedure can have an arbitrary finite or even infinite functional *level*: the definable modes include *zero* = **int**, *one* = **proc**(*zero*)*zero*, *two* = **proc**(*one*)*one*, ... and *untyped* = **proc**(*untyped*)*untyped*. The following two examples show how such procedures can be used in the definition of base objects.

Example 1. The following procedure EXP, embedded in a main program P, computes for an integer function f and positive integers n, k the expression

$$2^{\cdot^{\cdot^{\cdot^{2^{f(k)}}}}} \Big\} n$$

```
begin
 int input 1, input 2, output;
 proc EXP = (proc(int)int f, int n, int k) int:
  begin int result;
   if n = 0 then result := f(k)
            else result := 2↑ EXP(f, n − 1, k) fi;
   result
  end of EXP;
```

output := EXP(square, input 1, input 2)
end of P

Example 2. The following program P' (essentially borrowed from [40]), computes the factorial function using a recursively typed procedure SELF:

begin
mode M = **proc**(M, **int**) **int**;
int input, output;
proc SELF = (Mf, **int** n) **int**:
begin
if $n = 0$ **then** 1 **else** $n \cdot f(f, n-1)$ **fi**
end of SELF;
output := SELF(SELF, input)
end of P'

Clearly the first example shows, that it is *convenient* to have higher type procedures available. The obvious question then is: in what sense are such procedures *necessary*? It is well known that a simple language such as WHILE (with the usual arithmetical basic) is universal, hence the above question has to be formulated more carefully in order to be reasonable. A look at denotational semantics will help to find the right level of abstraction.

In a *denotational semantics* of an imperative programming language, the meaning of a program is defined abstractedly as a function which associates with the initial store the store obtained after executing the program, by reducing it inductively to the meaning of its constituents using some meta-linguistic concepts and a (essentially invariant from the discussed programming language) set of *auxiliary functions*. What metalanguage is used, depends essentially on how close the semantics should be to an implementation; to serve as a *standard* it is clearly desirable to retain a level of abstraction, which allows a *transparent* and *direct* simulation of the typical features of the given language, which is the reason why the *λ-calculus* (involving *abstraction*, *application*, and *recursion*) is chosen in the direct semantics of ALGOL 60 and PASCAL (see [51, 64]). By stressing the role of the meta-language we can split such a direct semantics into a translation of the programs into a λ-expression over the auxiliary functions, followed by the standard semantics of the λ-calculus ([60, 68, 22]). It should, however be clear, that we still have to eliminate the influence of the base operations (contained in the auxiliary functions) to give a concise formulation to the problem. To this end, we abstract from the λ-calculus to *λ-schemes* by denoting the auxiliary functions occuring in the translation by *operation symbols*, whose meaning has to be explicitly specified in some *semantic algebra*. These two levels of abstraction are depicted in Fig. 2.

To illustrate these ideas, let us give the λ-schemes P_λ and P'_λ corresponding to the sample programs (the algorithm for the translation is given in [19]).

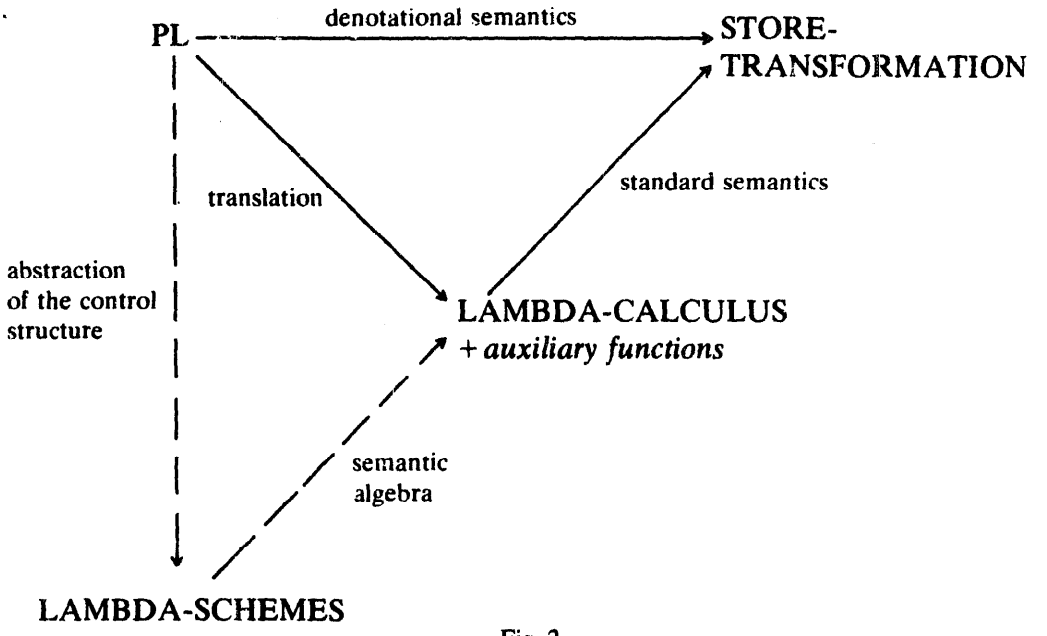
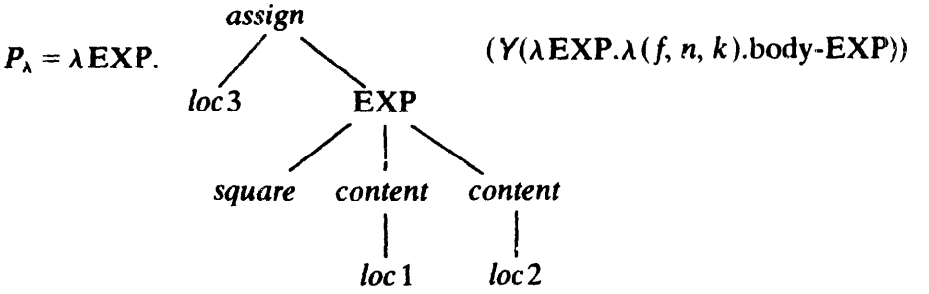
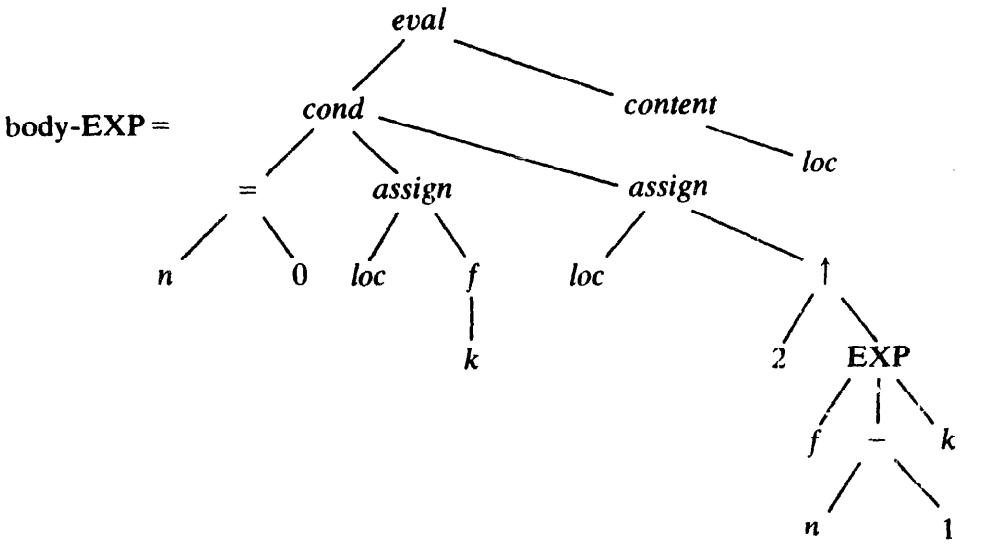


Fig. 2.

Example 1 (continued). Since EXP is defined recursively, its declaration semantics uses the *fixed-point-operator Y.*

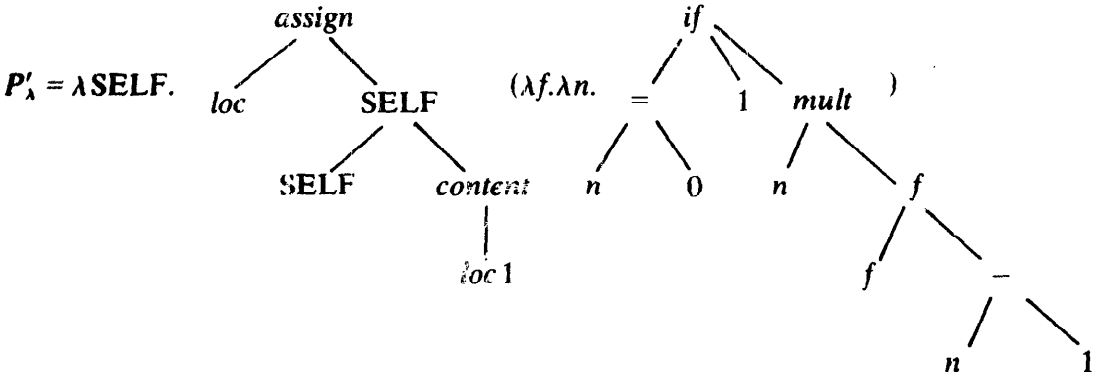


where



In the examples, we use $\Omega = \{0, 1, 2, -, \uparrow, square, mult, if, =, loc, \ldots, assign, content, eval, cond\}$ as denotations for the (more or less obvious) auxiliary functions.

Example 2 (continued).



A comparison of the λ-schemes resulting from the translation of ALGOL 68 procedures shows

– *non-recursive* procedures with *finite mode* induce λ-schemes, which contain only *typed* application and *typed* abstraction.

– *recursive* procedures with at most n nested occurrences of the **proc** constructor in their mode can be modelled by *n-λ-schemes* which in addition contain an atom Y_n for the *fixed-point-operation on the nth function space.*

– *recursively typed* procedures lead to λ-schemes which allow *self-application* and thus in general cannot be finitely typed.

The link between level-n schemes and the OI-hierarchy is provided by the fixed-point characterization: interpreting level-n schemes over the *algebra of formal (tree-) languages* gives exactly the nth language family in the OI-hierarchy, hence n-loops in Fig. 1 correspond to taking fixed points on the nth function space. By the OI-hierarchy result, this proves that recursion on higher types induces an infinite hierarchy of control structures, since more and more languages can be defined by increasing the level of recursion. In particular, when abstracting from the meaning of the base operations, higher type procedures cannot be simulated by procedures which make only use of base type parameters.

The IO-hierarchy is the one intended by Maibaum [42] and treated further by Turner [66]. (Unfortunately in both papers IO and OI seem to be confused.) Wand [70] defines the OI-hierarchy in a category-theoretic framework as morphisms of base type in certain derived theories and indicates a proof, that this hierarchy starts with the regular, context-free and indexed language of [2]. Maslov [43] uses a canonical generalization of indexed grammars to define a hierarchy of formal languages, which seems to be related to the OI-hierarchy, and provides an automata-theoretic characterization using multilevel stack-automata [44]. In [28], Engelfriet and Schmidt define the IO- and OI-hierarchies using a generalization of the fixed-point characterization of the IO- and OI-context-free tree languages. The

authors prove that the IO-hierarchy starts with the regular, context-free, and IO-macro languages and sketch a proof of the corresponding result for OI.

The study of recursion on higher types as control structure for programming languages was started by Milner [48] and Plotkin [54]. Program schemes modelling higher type recursion were first introduced by Indermark [35]. Parts of the results of this paper were presented in [12–14]. Models for a typed λ-calculus with restricted recursive types are discussed in [3]. A comprehensive treatment of untyped procedure calls on the level of schemes is given by Fehr [29].

This paper is split into two parts according to the semantical and syntactical aspects of higher type recursion.

In part one we generalize the theory of recursive program schemes as developed in [52, 53, 10] to the class of $n-\lambda$-schemes. Following the mathematical background, Section 2 gives an introduction into the results and proof-techniques of the following sections, using *regular systems of equations* [32, 52, 69] to demonstrate the main ideas. $n-\lambda$-schemes are defined in Section 3 as systems of equations, whose right-hand sides consist of (finitely) typed λ-terms over base operation symbols, procedure identifiers and formal parameters the main procedure being of a base type (hence nested declarations of recursive procedures are simulated by simultaneous recursion, see [19]). In connection with the *fixed-point semantics* we introduce a semantic algebra *Algol*, which induces a denotational semantics for a subset of ALGOL 68 via Fig. 2. A *combinatoric model* of higher type recursion is derived in Section 4 by viewing the functions over an algebra as carrier of the *derived algebra* with projection and functional substitution as new operations (cf. [42, 28]). Here, recursion on level n is modelled by regular systems of equations, which in addition to the base operation symbols contain projection- and substitution-symbols up to level n. This model allows for easy algebraic proofs of a *Mezei–Wright-like theorem* and a *closure-result* for the class of infinite trees generated by level-n schemes. By the equivalence of the combinatoric and applicative approach proved in the appendix we inherit a *Chomsky-normalform* for $n-\lambda$-schemes, which implies the equivalence of $1-\lambda$-schemes and recursive program schemes. The *Kleene-characterization* proved in Section 5 characterizes the infinite tree of a scheme as the join of the chain of approximations obtained by iterated parallel substitution of the procedure bodies followed by a $\perp$-replacement of the remaining procedure identifiers. This result is used in Section 6 to obtain the *completeness of a call-by-name operational semantics* over discrete interpretations: by viewing an $n-\lambda$ scheme as a *level-n grammar* (using the copy-rule to apply 'productions') with a $\perp$-replacement rule as additional choice, we can generate sufficiently many approximations using only OI-derivations to represent the infinite tree of the scheme. For the programming language considered, this implies the equivalence of denotational and copy-rule semantics.

Part two investigates the language families generated by level-n grammars in OI- and IO-mode. We start by presenting sample languages, which exhibit the typical *n-exponential growth* at the nth level. Two fixed-point characterizations

provide the discussed link between level-n schemes and level-n languages. The resulting characterization of OI-languages as *homomorphic images of regular infinite trees* and IO-languages as homomorphic images of *regular languages* are used in the following section to prove *decidability of the emptiness problem.* In the next section we provide a *branch-language* and *front-language-characterization*, which in particular show, that the OI-hierarchy satisfies the diagram in Fig. 1. Finally we prove, that both language families are *closed under intersection with regular languages.* In the OI-case, substitution closure implies, that level-n OI string languages form a substitution closed AFL, while IO languages are only closed under homomorphisms.

Since the proofs of these results require different techniques depending on the mode of derivation, they are presented separately in Sections 7 and 8.

In the last section we use the *rational index*, a complexity measure for languages introduced in [6], to prove both hierarchies infinite. An analysis of the proof in the IO-case leads to sufficient conditions on the input-language-family to prove strictness of the resulting IO-like hierarchy. Examples for families satisfying these conditions are (deterministic) top–down translations of regular languages and the families in the OI-hierarchy. Finally we apply these results to prove strictness of the scheme-hierarchy.

PART I. LEVEL-N SCHEMES

1. Terminology, definitions, and basic facts

The reader is assumed to be familiar with the basic concepts of tree language theory (as presented e.g. in [23, 65]) and initial algebra semantics [32]. For the sake of completeness and to fix notation, we will review some definitions and state a few basic properties needed in the sequel. On first reading, this section may be skipped except for the definition of *set* in Subsection 1.3.

1.1. Continuous algebras

Let $A = (\boldsymbol{A}, \leqslant)$ be a *partial order* (*po*). If $T \subseteq \boldsymbol{A}$ has a *least upper bound* in A, we denote it by $\bigsqcup T$. $T \neq \emptyset$ is *directed* in A iff for each $t_1, t_2 \in T$ there exists $t_3 \in T$ with $t_1, t_2 \leqslant t_3$.

A is called *Δ-complete* (*cpo*) iff there is $\bot \in \boldsymbol{A}$ which is *minimal in A* and each directed T in A has $\bigsqcup T \in \boldsymbol{A}$.

A is called $\bigsqcup$-*complete* (*complete lattice*) iff each subset T of A has $\bigsqcup T$ in A.

For $\bigsqcup$-complete (Δ-complete) po's A and B we denote by $f: A \to B$ a $\bigsqcup$-continuous (Δ-continuous) mapping from A to B, i.e. f is monotonic and $f(\bigsqcup T) = \bigsqcup f(T)$ for each (directed) subset of A. If $A = B$, then, by Tarski's fixed-point theorem [63], there exists $\bigsqcup\{f^\nu(\bot) \mid \nu \in \mathbb{N}\}$, the *minimal fixpoint of* f, which we denote by $\mu(f)$.

Let I be a set of *base types*. A family of sets $A = (A^i \mid i \in I)$ is called an *I-set*. We extend this notation to $I^* := I^+ \cup \{e\}$ by A^e is the set consisting of the empty tuple, and $A^{wi} := A^w \times A^i$. The cartesian product is taken associative.

For I-sets A and B we write $A \subseteq B$ if $A^i \subseteq B^i$ for each $i \in I$, and define $A \cup B := (A^i \cup B^i \mid i \in I)$. An *I-mapping* $f: A \to B$ is a family $(f^i: A^i \to B^i \mid i \in I)$. We extend this notation to I^* by $f^e := () \mapsto ()$, $f^{wi} := (a, a) \mapsto (f^w(a), f^i(a)): A^{wi} \to B^{wi}$. We will omit the superscript w in f^w if no ambiguity arises.

The elements of $D(I) := I^* \times I$ are called *derived types* of I. An I-set A determines the $D(I)$-set $D(A) := (A^w \to A^i \mid (w, i) \in D(I))$ where $A^w \to A^i := \{f \mid f: A^w \to A^i\}$.

A family of partial orders $A = (A^i \mid i \in I)$ is called $\bigsqcup$-*complete* (Δ-*complete*) iff each A^i is $\bigsqcup$-complete (Δ-complete). In this case, $D(A)$ is $\bigsqcup$-complete (Δ-complete), where the partial order relations are defined componentwise on products and pointwise on function spaces.

Let Ω be a $D(I)$-set. $f \in \Omega^{(w,i)}$ is called an *operation symbol of type* (w, i). An Ω-*algebra* $\mathcal{A} = (A, \varphi_A)$ consists of an I-set A as *carrier* and a $D(I)$-mapping $\varphi_A: \Omega \to D(A)$. φ_A assigns to each $f \in \Omega^{(w,i)}$ a *base operation* $\varphi_A(f): A^w \to A^i$. For $z \in \{\bigsqcup, \Delta\}$ we say that $\mathcal{A}$ is *z-continuous* ($\mathcal{A} \in z\text{-}alg\,\Omega$) iff A is z-complete and all $\varphi_A(f)$ are z-continuous.

Any z-continuous Ω-algebra $\mathcal{A}$ can be viewed as an $\Omega_\bot := \Omega \cup (\{\bot_i\} \mid (e, i) \in D(I))$-algebra $\mathcal{A}_\bot$ by letting $\bot_i$ denote the minimal element in A^i. We omit the index $\bot$ if no ambiguity arises.

An Ω-algebra A canonically induces the $\bigsqcup$-continuous *powerset-algebra* $\mathcal{P}\mathcal{A}$ with carrier $(\mathrm{P}A^i \mid i \in I)$ and assignment-function given by $\varphi_{\mathrm{P}A}(f)(T_1, \ldots, T_n) = \{\varphi_A(f)(a_1, \ldots, a_n) \mid a_j \in T_j\}$.

Any $\bigsqcup$-continuous Ω-algebra $\mathcal{B}$ can be viewed as an Δ-continuous $\Omega_+ := \Omega \cup (\{+_i\} \mid (ii, i) \in D(I))$-algebra $\mathcal{B}_+$ with $+_i$ denoting the binary join operation $\bigsqcup$ in A^i. Again we omit the index $+$ if no ambiguity arises.

1.2. Initial algebras

A structure-preserving I-mapping $h: A \to B$ between carriers of Ω-algebras $\mathcal{A}$ and $\mathcal{B}$ is called an Ω-*homomorphism* (notation $h: \mathcal{A} \to \mathcal{B}$). The following result, guaranteeing the existence of unique homomorphisms $\mathcal{T} \to \mathcal{A}$ for suitable tree-algebras $\mathcal{T}$ is basic to our work.

Theorem 1.1. (1) [32]. *The Δ-continuous algebra $\mathcal{CT}_\Omega$ of infinite-trees over Ω is initial in Δ-alg Ω, i.e. for all $\mathcal{A} \in \Delta\text{-}alg\,\Omega$ there exists a unique Δ-continuous $\Omega_\bot$-homomorphism $h_\mathcal{A}: \mathcal{CT}_\Omega \to \mathcal{A}$.*

(2) [28]. *The $\bigsqcup$-continuous algebra $\mathcal{PT}_\Omega$ of tree-languages over Ω is initial in $\bigsqcup$-alg Ω, i.e. for all $\mathcal{B} \in \bigsqcup$-alg Ω there exists a unique $\bigsqcup$-continuous Ω-homomorphism $H_\mathcal{B} : \mathcal{PT}_\Omega \to \mathcal{B}$.*

We will not make use of any particular representation of the elements of CT_Ω (e.g. as partial mappings on tree domains) but will rely on initiality arguments. Note, that the following basic properties can be derived from Theorem 1.1:

Corollary 1.2. (1) *Each infinite tree $t \in \mathrm{CT}_\Omega$ is either atomic (i.e. $t = a := \varphi_{\mathrm{CT}_\Omega}(a)(\ )$ for some $a \in \Omega_\perp^{(e,i)}$) or can be uniquely decomposed as $t = ft_1 \cdots t_n := \varphi_{\mathrm{CT}_\Omega}(f)(t_1, \ldots, t_n)$ for an operation symbol f and infinite trees t_j.*

(2) *The set of finite trees FT_Ω in CT_Ω can be identified with the carrier of the usual $\Omega_\perp$-word algebra $\mathcal{T}_{\Omega_\perp}$. Clearly all trees in FT_Ω can be obtained using the rules of (1).*

FT_Ω is partially ordered by $t \leqslant t'$ iff $t = \perp$ or $\exists f \in \Omega^{(w,i)}$, $t_j, t_j' \in T_{\Omega_\perp}^{w(j)}$ s.t. $t = ft_1 \cdots t_n \wedge t' = ft_1' \cdots t_n' \wedge (\forall j \in [n]\ t_j \leqslant t_j')$ where $n = l(w)$.

(3) *For each infinite tree $t \in \mathrm{CT}_\Omega$ there exists a directed subset $\Delta_t \subseteq \mathrm{FT}_\Omega$ s.t. $t = \bigsqcup \Delta_t$. E.g. Δ_t may be chosen to be $\{t' \in \mathrm{FT}_\Omega \mid t' \leqslant t\}$, the ideal generated by t.*

As auxiliary functions on (finite) trees we use *depth*(t), *front*(t) (the concatenation of t's leaves from left to right), *breadth*(t) (counting the number of leaves of t), and *br*(t) (the set of *branches* of t).

1.3. Derived operations, set, and tree-homomorphisms

In this section, we will consider some examples of mappings defined using initiality arguments. We start by recalling the definition of derived operations.

For $w \in I^*$, define the set of (formal) parameters Y_w by $Y_e := \emptyset$, $Y_{wi} := Y_w \cup \{y_{l(w)+1,i}\}$. Let $Y = \bigcup_{w \in I^*} Y_w$, then Y can be viewed as an I-set by defining $Y^i := \{y_{j,w(j)} \mid w(j) = i \text{ for some } w \in I^*\}$. Note, that $y_w := (y_{1,w(1)}, \ldots, y_{n,w(n)}) \in Y^w$. Finally, let $\Omega(Y_w)$ denote the $D(I)$-set $\Omega \cup (Y_w^i \mid (e, i) \in D(I))$.

To define derived operations (in the Δ-continuous case; the $\bigsqcup$-continuous case is completely analogous and will not be restated), consider some assignment $a : Y_w \to A$ of the parameters into the carrier of some Δ-continuous Ω-algebra $\mathcal{A}$, then a induces an $\Omega(Y_w)$-algebra $\mathcal{A}_a$ with $y_{j,w(j)}$ denoting $a(y_{j,w(j)})$. Hence any infinite tree $t \in \mathrm{CT}_\Omega(Y_w) := \mathrm{CT}_{\Omega(Y_w)}$ defines a *derived operation* $derop_\mathcal{A}(t) : A^w \to A^i$ given by $a \mapsto h_{\mathcal{A}_a}(t)$, where $h_{\mathcal{A}_a}$ is the unique Δ-continuous $\Omega(Y_w)_\perp$-homomorphism guaranteed by initiality of $\mathcal{CT}_{\Omega(Y_w)}$ (note, that we identified $a = (a_1, \ldots, a_m) \in A^w$ with the assignment $y_{j,w(j)} \mapsto a_j$).

The derived operation of an infinite tree (a tree language) over a Δ-($\bigsqcup$)-continuous algebra is Δ-continuous (for a proof, see [32, 28]).

In case $\mathcal{A} = \mathcal{CT}_{\Omega(Y)}$, $t \leftarrow (t_1, \ldots, t_n) := derop_\mathcal{A}(t)(t_1, \ldots, t_n)$ denotes the tree obtained by substituting $(t_1, \ldots, t_n)$ for the parameters y_w in t. We also use $\leftarrow$ in

the $\bigsqcup$-continuous case; here $\leftarrow$ coincides with OI-*substitution* of tree languages [28] defined by

$$a \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k) = \{a\} \quad \text{for } a \in \Omega^{(e,i)},$$

$$y_{j,v(j)} \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k) = L_j,$$

$$ft_1 \cdots t_n \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k) = \{fs_1 \cdots s_n \mid s_j \in t_j \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k)\},$$

$$L \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k) = \bigcup_{t \in L} t \underset{\text{OI}}{\longleftarrow} (L_1, \ldots, L_k).$$

It is straightforward to show, that substitution is Δ-continuous in all its arguments [32, 28].

The next lemma, taken from [32], shows that substitution of terms corresponds to functional substitution.

Lemma 1.3. *Let* $t \in \mathrm{CT}_\Omega(Y_w)$, $s \in \mathrm{CT}_\Omega(Y)^w$, *and* $\mathscr{A} \in \Delta\text{-}alg\,\Omega$.
Then

$$derop_{\mathscr{A}}(t \leftarrow s) = derop_{\mathscr{A}}(t) \circ derop_{\mathscr{A}}(s)$$

We will also refer to Lemma 1.3 in proofs involving the $\bigsqcup$-continuous case. An immediate corollary of Lemma 1.3 is associativity of substitution.

Lemma 1.4. *Let* $u \in \mathrm{CT}_\Omega(Y_v)$, $t = (t_1, \ldots, t_m) \in \mathrm{CT}_\Omega(Y_w)^v$, $s \in \mathrm{CT}_\Omega(Y)^w$.
Extend substitution to vector arguments t *by* $t \leftarrow s := (t_1 \leftarrow s, \ldots, t_m \leftarrow s)$. *Then*

$$u \leftarrow (t \leftarrow s) = (u \leftarrow t) \leftarrow s.$$

We now turn to the interpretation of infinite trees over Ω involving $+$. Intuitively, by viewing $+$ as union of tree languages, such a tree t defines a tree language over Ω, which will be denoted $set(t)$.

Definition 1.5. *set* denotes the unique Δ-continuous $(\Omega_+)_\perp$-homomorphism $\mathscr{CT}_{\Omega_+} \to (\mathscr{P}\mathscr{T}_\Omega)_+$.

Note, that $set(\perp) = \emptyset$ and hence, by $\bigsqcup$-continuity of operations in $\mathscr{P}\mathscr{T}_\Omega$, $set(t) = \emptyset$ for any $t \in \mathrm{FT}_\Omega$ containing $\perp$. By Corollary 1.2 and Δ-continuity of *set* this implies $set(t) = \emptyset$ for any nonfinite $t \in \mathrm{CT}_\Omega$.

The next lemma shows, that *set* commutes with substitution.

Lemma 1.6.

$$\forall t \in \mathrm{CT}_\Omega(Y_w), s \in \mathrm{CT}_\Omega(Y)^w \quad set(t \leftarrow s) = set(t) \leftarrow set(s).$$

Proof. Since all involved mappings are Δ-continuous, it suffices by Corollary 1.2 to prove the assertion for finite trees. The straightforward proof is omitted. □

We close this section by extending the concept of homomorphisms (in the theory of formal languages) to infinite trees.

Definition 1.7. (1) Consider $D(I)$-sets Ω, Σ. Note, that $CT_\Sigma(Y)$ can be viewed as a $D(I)$-set $(CT_\Sigma(Y_w)^i \mid (w, i) \in D(I))$. By initiality of $\mathscr{CT}_\Omega$, any $D(I)$-mapping $\sigma : \Omega \to CT_\Sigma(Y)$ extends uniquely to a Δ-continuous $\Omega_\perp$-homomorphism $\hat{\sigma} : \mathscr{CT}_\Omega \to (\mathscr{CT}_\Sigma)_\sigma$ the *tree-homomorphism induced by* σ, where $(\mathscr{CT}_\Sigma)_\sigma$ denotes the Δ-continuous Ω-algebra with carrier CT_Σ and assignment-function $\leftarrow \circ \sigma$ given by $\leftarrow \circ \sigma(f)$ $(t_1, \ldots, t_n) := \sigma(f) \leftarrow (t_1, \ldots, t_n)$.

(2) A family $\mathscr{T}$ of infinite trees is *closed under tree-homomorphism* iff for all $D(I)$-sets Ω, Σ and $D(I)$-mappings $\sigma : \Omega \to CT_\Sigma(Y)$ *range*$(\sigma) \subseteq \mathscr{T}(\Sigma)$ implies $\hat{\sigma}(\mathscr{T}(\Omega)) \subseteq \mathscr{T}(\Sigma)$.

To view string-theory as a subtheory of tree-theory, we identify strings with monadic trees as explained below.

Definition 1.8. Let Ω be some $D(\{i\})$-set.

Ω is called *monadic* iff $|\Omega^{(e,i)}| = 1 \wedge \forall k > 1\ \Omega^{(ik,i)} = \emptyset$.

Clearly the left-concatenation algebra V^* of strings over V and $\mathscr{T}_{\Omega_V}$ with $\Omega_V^{(e,i)} = \{e\}$, $\Omega_V^{(i,i)} = V$ are isomorphic.

2. Regular equations

This section is tutorial in nature. We will review some standard results in schematology for one of the simplest class of schemes: regular systems of equations. In doing so, we hope to familiarize the reader with some proof techniques used in the sequel, and, at the same time, prepare results in later paragraphs.

Syntactically, a regular system of equations will be defined as a function mapping *procedurenames* to the *body* of the procedure, which in this simple case merely consists of a tree over operation symbols and procedurenames.

Definition 2.1. Let I be a set of base types, and Ω a $D(I)$-set of operation symbols.

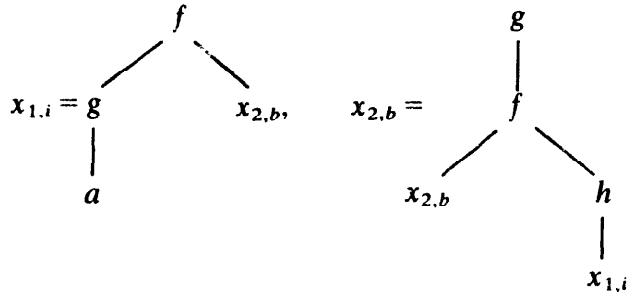
A *regular system of equations over* Ω of type $i \in I$ is an I-mapping $S : X_v \to T_\Omega(X_v)$ s.t. $v \in I^+$ and $v(1) = i$.

We denote by *var*(S) the I-set X_v of procedurenames of S, and by $R(\Omega)$ the I-set of all regular systems of equations over Ω.

Note that no formal parameters are allowed, hence procedurenames occur only as leaves in the body of a procedure. In order to deal with higher type recursion,

the main procedure $x_{1,v(1)}$ will be allowed to call procedures of an arbitrarily high type.

Example. Let $I=\{i, b\}$, $\Omega=\{f, g, h, a\}$ with types (bb, i), (i, b), (i, b), (e, i). The following system of equations defines a scheme S in $R(\Omega)^i$:



The notation as a system of equations suggests a *fixed-point semantics* for schemes in $R(\Omega)$: over an *interpretation* $\mathscr{A} \in \Delta\text{-alg}\,\Omega$, S induces a function

$$S_{\mathscr{A}} := (derop_{\mathscr{A}}(S(x_{1,v(1)})), \ldots, derop_{\mathscr{A}}(S(x_{m,v(m)})))$$

which maps A^v into itself. The semantics of S over $\mathscr{A}$ is the first component of the least fixed-point of $S_{\mathscr{A}}$.

Definition 2.2. Let $S \in R(\Omega)^i$, $\mathscr{A} \in \Delta\text{-alg}\,\Omega$.

The *semantics of S over the interpretation* $\mathscr{A}$ is

$$[\![S, \mathscr{A}]\!] := pr_1(\mu S_{\mathscr{A}}) \in A^i.$$

Example (continued). According to Tarski's fixed-point theorem, the semantics of S over $\mathscr{CT}_\Omega$ is the least upper bound of the chain

$n \mapsto pr_1 S^n_{\mathscr{CT}_\Omega}(\perp_i, \perp_b)$

n	pr_1	pr_2
0	$\perp_i$	$\perp_b$
1	$fga\ \perp_b$	$gf\ \perp_b h\ \perp_i$
2	$fga\,gf\ \perp_b h\ \perp_i$	$gf\,gf\ \perp_b h\ \perp_i hfga\ \perp_b$
3	$fga\,gf\,gf\ \perp_b h\ \perp_i hfga\ \perp_b$	$gf\,gf\,gf\ \perp_b h\ \perp_i hfga\ \perp_b hfga\,gf\ \perp_b h\ \perp_i$
	$\vdots$	

Having defined semantics, we immediately have a notion of equivalence: two schemes S_1, S_2 over Ω are equivalent (notation: $S_1 \sim S$) iff they compute the same value under all interpretations.

Can we decide equivalence? In order to answer this and similar questions (such as divergence, reachability, macro-property [39], ...), it is useful to lift these problems to some syntactic domain and then apply language-theoretic tools. For regular systems of equations it is well known that the equivalence class of a scheme S can be characterized by its *infinite tree* $T(S)$, which is simply the interpretation of S over the initial continuous algebra. This stems from the following *MW-like theorem.*

Theorem 2.3 ([32, 53]).

$$\forall S \in R(\Omega)\, \forall \mathscr{A} \in \Delta\text{-}alg\, \Omega \;\; [\![S, \mathscr{A}]\!] = h_{\mathscr{A}}(T(S)).$$

The proof of this result follows immediately from the observation that the nth approximation of the semantics of S over $\mathscr{A}$ is the homomorphic image of the nth approximation over $\mathscr{CT}_\Omega$.

The syntactic objects we gain through the MW-like result are still quite unmanageable. We will now work towards a more operational description of the infinite tree of a scheme which relies on regular tree languages.

Note that in the computation of the infinite tree of S according to the Tarski fixed-point theorem one substitutes in the $(n+1)$st step the nth approximate of $T(S)$ into the right-hand sides of the scheme. The following *Kleene-characterization* gives a top–down algorithm to compute $T(S)$.

Definition 2.4. Let $S \in R(\Omega)$ with $var(S) = X_v$.

By initiality S determines uniquely a Δ-continuous $\Omega_\perp$-homomorphism $\hat{S}: \mathscr{CT}_\Omega(X_v) \to \mathscr{CT}_\Omega(X_v)$.

The *Kleene-sequence of* S, $K(S)$, is defined by

$$K(S)(n) := \hat{\perp} \circ \hat{S}^n(x_{1,v(1)}),$$

where $\hat{\perp}$ is the unique homomorphism generated by

$$x_{j,v(j)} \mapsto \perp_{v(j)}.$$

A simple induction argument shows that the chains

$$n \mapsto \hat{\perp} \circ \hat{S}(x_{j,v(j)}) \quad \text{and} \quad n \mapsto pr_j S^n_{\mathscr{CT}_\Omega}(\perp_v)$$

coincide. Thus we have

Theorem 2.5 ([52]). $\forall S \in R(\Omega)\; T(S) = \bigsqcup K(S)$.

Let us now discuss the relation to regular tree languages.

A *regular tree grammar* $G \in \mathrm{NR}(\Omega)$ with *terminals* Ω, *nonterminals* X_v, and *axiom* $x_{1,v(1)}$ is simply a nondeterministic regular system of equations, where each procedurename may have a finite number of procedure bodies (the *productions*). Since

trees (written in Polish notation) are just strings, regular tree grammars are just a certain kind of context-free (string-) grammars, hence the notion of derivation ($\Rightarrow^*$) is well defined. The *tree language generated by G* is the set of all terminal trees derivable from the axiom:

$$L(G) = \{t \in T_\Omega \mid x_{1,v(1)} \overset{*}{\underset{G}{\Rightarrow}} t\}.$$

Example (continued). Define a regular tree grammar over $\Omega_\perp$ by adding to S the productions $x_{1,i} = \perp_i$, $x_{2,b} = \perp_b$. We show how to derive $K(S)(3) \in \mathrm{FT}_\Omega = T_{\Omega_\perp}$:

$$\begin{aligned}
x_{1,i} &\Rightarrow f\,ga\,x_{2,b} \Rightarrow f\,ga\,gf\,x_{2,b}\,hx_{1,i} \\
&\Rightarrow f\,ga\,gf\,gf\,x_{2,b}\,hx_{1,i}\,hx_{1,i} \\
&\Rightarrow f\,ga\,gf\,gf\ \perp_b hx_{1,i}\,hx_{1,i} \\
&\Rightarrow f\,ga\,gf\,gf\ \perp_b h \perp_i hx_{1,i} \\
&\Rightarrow f\,ga\,gf\,gf\ \perp_b h \perp_i hf\,ga\,x_{2,b} \\
&\Rightarrow f\,ga\,gf\,gf\ \perp_b h \perp_i hf\,ga\ \perp_b.
\end{aligned}$$

With each (deterministic) system of equations S over Ω we associate a regular tree grammar $S_\perp$ over $\Omega_\perp$ by adding the productions $x_{j,v(j)} = \perp_{v(j)}$. The tree language $L(S_\perp) \subseteq \mathrm{FT}_\Omega$ is called the *schematic tree language generated by S*. It is easy to see that each tree in the Kleene-sequence can be generated by $S_\perp$. On the other hand, a suitable induction on the length of a derivation shows that a tree derived in n steps is always majorized by $K(S)(n)$. Hence $L(S_\perp)$ is directed and its join is equal to the join of the Kleene-sequence.

Theorem 2.6 ([52]). $\forall S \in R(S) \quad T(S) = \bigsqcup L(S_\perp)$.

It follows from the normalform theorem 2.7 that S can be chosen in such a way that its schematic language is an ideal and thus characterizes the equivalence class of S. Hence, by the corresponding result for regular tree grammars, equivalence (and divergence) for regular systems of equations is decidable.

We conclude this section by establishing a closure property for the class

$$\mathcal{T} := \{T(S) \mid S \in R(\Omega) \text{ for some } \Omega\}$$

of *regular infinite trees*.

We would like to show that $\mathcal{T}$ is closed under tree-homomorphisms, i.e. replacing all operation symbols in a regular infinite tree by regular infinite trees of the same arity yields again a regular infinite tree. To this end we need two normalform results which are of interest in their own right. They show that the syntactic definition of $R(\Omega)$ can be altered in two extremes without changing the class of computable elements.

Let us first consider the restriction. Obviously, we can reduce the depth of the right-hand sides of a scheme by successively substituting $x = ft_1 \cdots t_k$ by the equations $x = fx_1 \cdots x_k$, $x_j = t_j$.

Theorem 2.7 ([45], normalform).

$$\forall S \in R(\Omega)\ \exists S' \in R(\Omega)\ S \sim S'$$
$$\wedge \forall x \in var(S')\ S'(x) \in \{x', fx_1 \cdots x_k \mid x', x_j \in var(S')\}.$$

Note that the construction of S' according to this idea will only give a polynomial increase in the size of the scheme.

On the other hand, one can allow regular infinite trees as right-hand sides. Note that the definition of semantics extends straightforwardly to this case.

Theorem 2.8 ([69]). *Let $S: X_v \to \mathcal{T}(\Omega(X_v))$ be an I-mapping. Then there exists $S' \in R(\Omega)$ with $S \sim S'$.*

The obvious idea behind the construction of S' is to replace each right-hand side of S by a call to the main procedure of the scheme defining the right-hand side.

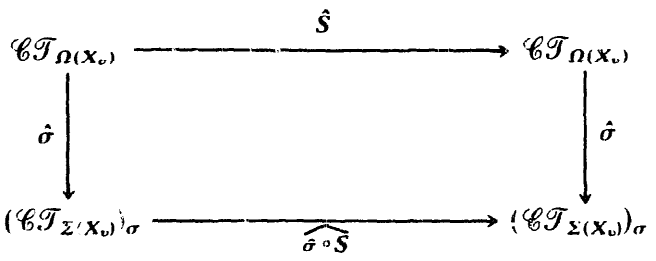
Both of these results are corollaries of Wand's normalform theorems [69]. A rigorous proof can be found in [36].

We can now derive closure of regular infinite trees under tree-homomorphisms.

Theorem 2.9. *$\mathcal{T}$ is closed under tree-homomorphisms.*

Proof. Let $S \in R(\Omega)$ be in normalform, and let $\hat{\sigma}$ be a tree-homomorphism (see Definition 1.7) s.t. for all $f \in \Omega^{(w,i)}$ $\sigma(f) \in \mathcal{T}(\Sigma(Y_w))^i$. By Theorem 2.8 there exists $S' \in R(\Sigma)$ such that $S' \sim \hat{\sigma} \circ S$, hence it is sufficient to prove $\hat{\sigma}(T(S)) = T(\hat{\sigma} \circ S)$.

Consider the following diagram:



Since $\hat{\sigma} \circ \hat{S}$ and $\widehat{\hat{\sigma} \circ S} \circ \acute{\sigma}$ coincide (by definition) on the generators X_v and $\widehat{\hat{\sigma} \circ S}$ is a strict Δ-continuous Ω-homomorphism, the above diagram commutes. By an easy induction argument this implies

$$\widehat{\hat{\sigma} \circ S^n}(x_{1,v(1)}) = \hat{\sigma}(\hat{S}^n(x_{1,v(1)})). \qquad (*)$$

But then

$$T(\hat{\sigma} \circ S) =$$

$$= \bigsqcup_n \hat{\perp} \circ \widehat{\hat{\sigma} \circ S^n}(x_{1,v(1)}) \quad \text{by Theorem 2.5}$$

$$= \bigsqcup_n \hat{\perp} \circ \hat{\sigma} \circ \hat{S}^n(x_{1,v(1)}) \quad \text{by } (*)$$

$$= \bigsqcup_n \hat{\sigma} \circ \hat{\perp} \circ \hat{S}^n(x_{1,v(1)})$$

$$= \hat{\sigma}(\bigsqcup_n \hat{\perp} \circ \hat{S}^n(x_{1,v(1)}))$$

$$= \hat{\sigma}(T(S)) \quad \text{by Theorem 2.5.} \ \square$$

3. *N*-lambda schemes

In this section we introduce a λ-calculus oriented class of schemes which models higher type recursion.

For the purpose of this paper it is convenient to use a representation slightly different from the 'typed λ-calculus + Y' framework suggested in the introduction. Rather than allowing nested occurrences of the fixed-point operator, we generalize regular systems of equations and allow typed λ-terms as right-hand sides. In the terminology of programming languages, this requires elimination of nested procedure declarations. Moreover, our definition of typed λ-terms differs slightly from the standard definition in the restriction to derived types. We note that these variations in the syntax do not change the class of computable objects [18].

To aid intuition, we start with an example.

Example. Consider the sample ALGOL 68 program of the introduction. The types involved are functional types over the base types

$$\begin{aligned} I = \{ & i && \text{integers} \\ , & b && \text{booleans} \\ , & l && \text{locations} \\ , & c\} && \text{continuations.} \end{aligned}$$

The types of the operation symbols are given by

$$\begin{aligned} \Omega = \{ & 0, 1, 2 && : (e, i) \\ , & -, \uparrow && : (ii, i) \\ , & \text{square} && : (i, i) \\ , & = && : (ii, b) \end{aligned}$$

$$
\begin{aligned}
&, \text{loc1}, \text{loc2}, \text{loc3}, \text{loc}: && (e, l)\\
&, \text{content} && : (l, i)\\
&, \text{assign} && : (li, c)\\
&, \text{eval} && : (ci, i)\\
&, \text{cond}\} && : (bcc, c).
\end{aligned}
$$

We use the procedure identifiers

$$
\begin{aligned}
X = \{&x_0 && : c\\
&, x_1\} && : ((i, i), (ii, i)).
\end{aligned}
$$

The formal parameters have their type as subscript

$$Y = \{y_{1,i}, y_{2,i}, y_{1,(i,i)}\}.$$

We represent the program by the equations (omitting '( )')

$$x_0 = \text{assign}(\text{loc3}, x_1(\text{square})(\text{content})(\text{loc1}), \text{content}(\text{loc2}))$$

$$
\begin{aligned}
&x_1(y_{1,(i,i)})(y_{1,i}, y_{2,i})\\
&\quad = \text{eval}(\text{cond}(=(y_{1,i}, 0),\\
&\qquad \text{assign}(\text{loc}, y_{1,(i,i)}(y_{2,i})),\\
&\qquad \text{assign}(\text{loc}, \uparrow(2, x_1(y_{1,(i,i)})(-(y_{1,i}, 1), y_{2,i})))),\\
&\qquad \text{content}(\text{loc}))
\end{aligned}
$$

In general, the right-hand sides of such equations consist of typed λ-terms which are inductively built up using operation symbols, procedure identifiers and formal parameters.

Definition 3.1. (1) Let I be a set of basetypes. The set $D^*(I)$ of *derived types over* I is defined by

$$D^0(I) := I, \quad D^{n+1}(I) := D^n(I)^* \times D^n(I), \quad D^*(I) := \bigcup_n D^n(I).$$

(2) Let Ω be a $D(I)$-set of operation symbols, and X, Y be a $D^*(I)$-set of *variables* and *parameters*, respectively. The $D^*(I)$-set $T_{\Omega,X,Y}$ of *typed λ-terms over Ω, X and Y* is the smallest $D^*(I)$-set T with

$$
\begin{aligned}
&\Omega \subseteq T, \quad X \cup Y \subseteq T,\\
&t \in T^{(\alpha,\nu)} \wedge s \in T^{\alpha} \rightsquigarrow t(s) \in T^{\nu},\\
&t \in T^{\nu} \wedge (\alpha, \nu) \in D^*(I) \rightsquigarrow \lambda y_{\alpha} \,.\, t \in T^{(\alpha,\nu)}.
\end{aligned}
$$

Due to the restriction to derived types, each term can be uniquely decomposed into its subterms.

We interrupt the formal development to introduce some notation.

Notation. (1) The *level* of a type $\tau \in D^n(I)$ is its functional depth n.

(2) The type of a term t is denoted by $type(t)$. The *level* of a term is the level of its type.

(3) $var(t)$ and $par(t)$ denotes the set of variables and parameters of t, respectively.

(4) The set of *free parameters* of t, $free(t)$, is defined inductively by

$$free(f) = \emptyset = free(x) \quad \text{for } f \in \Omega,\ x \in X,$$

$$free(y_{i,\nu}) := \{y_{i,\nu}\},$$

$$free(t_0(t_1, \ldots, t_r)) := \bigcup_{0 \leq j \leq r} free(t_j),$$

$$free(\lambda y_\alpha . t) := free(t) \setminus Y_\alpha.$$

(5) t is *closed* iff $free(t) = \emptyset$.

(6) We denote by $t\downarrow$ the level-0 term obtained from t by applying t to all its formal parameters: if $type(t) = (\alpha_{n-1}, \ldots, (\alpha_0, i) \cdots)$, then

$$t\downarrow := t(y_{\alpha_{n-1}}) \cdots (y_{\alpha_0}) \in T_{\Omega,X,Y}.$$

Formally, we will define such systems of equations as type-preserving mappings from procedure identifiers to procedure bodies. As in programming languages, the main program has to be of a ground type, and all parameters have to be declared, i.e. we consider only closed procedure bodies.

Definition 3.2. A *typed λ-scheme over Ω* is a $D^*(I)$-mapping

$$S : X \to T_{\Omega,X,Y}$$

s.t.

(i) $X = \{x_0, \ldots, x_N\}$ is a finite $D^*(I)$-set,

(ii) $type(x_0) = (\underbrace{e, \ldots, (e}_{k}, i) \cdots)$ for some $k \in \omega$,

(iii) $\forall x \in X \quad S(x)$ is closed.

Notation. (7) $var(S)$ denotes the set of variables of S.

(8) The *level* of a scheme S is the highest level occurring in its definition:

$$level(S) := max\{level(x) \mid x \in var(S)\}.$$

(9) The class of all typed λ-schemes over Ω with level less than or equal n will be denoted $n - \lambda(\Omega)$.

In examples, we shall write a typed λ-scheme as a system of equations

$$\begin{cases} x_j\downarrow = t_j\downarrow \\ j \in \{0, \ldots, N\}, \end{cases}$$

where t_j is obtained from $S(x_j)$ by omitting all outer abstractions.

Hence the *right-hand side of* x_j refers to $t_j\downarrow$ and $rhs(S) = \{t_i\downarrow \mid x_j \in var(S)\}$.

As is usual in denotational semantics, the meaning of the procedure bodies is first computed relative to fictitious values for procedure- and parameter-identifiers, which are given by an environment.

Definition 3.3. Let $\mathscr{A} \in \Delta\text{-}alg\ \Omega$.

(1) The $D^*(I)$-cpo A^* of functions over A is given by

$$A^* := (A^\tau \mid \tau \in D^*(I))$$

with

$$A^{(\alpha,\nu)} := A^\alpha \to A^\nu.$$

(2) The set of *environments over* $\mathscr{A}$ is given by

$$U_{\mathscr{A}} := \{\rho : X \cup Y \to A^* \mid \rho \text{ is a } D^*(I)\text{-map}\}.$$

(3) The *semantics of terms over the interpretation* $\mathscr{A}$,

$$[\![\ , \mathscr{A}]\!] : T_{\Omega,X,Y} \to U_{\mathscr{A}} \to A^*$$

is defined inductively by

$$[\![f, \mathscr{A}]\!] := \varphi_A(f) \quad \text{for } f \in \Omega,$$

$$[\![x, \mathscr{A}]\!] := \rho[\![x]\!], \qquad [\![y_{j,\nu}, \mathscr{A}]\!] := \rho[\![y_{j,\nu}]\!],$$

$$[\![t(t_1, \ldots, t_r), \mathscr{A}]\!]\rho := [\![t, \mathscr{A}]\!]\rho([\![t_1, \mathscr{A}]\!]\rho, \ldots, [\![t_r, \mathscr{A}]\!]\rho),$$

$$[\![\lambda y_\alpha.t, \mathscr{A}]\!]\rho := a \mapsto [\![t, \mathscr{A}]\!]\rho[y_\alpha/a].$$

The definition of semantics of typed λ-schemes is a straightforward generalization of the regular case. As the semantics of a procedure body only depends on the value of the procedure identifiers, $[\![S(x), \mathscr{A}]\!]$ denotes a 'higher type derived operation': any assignment $\rho_a := (x_j \to a_j)$ extends uniquely to $[\![S(x), \mathscr{A}]\!]\rho_a \in A^{type(x)}$. Hence S induces a continuous functional

$$S_{\mathscr{A}} : A^{type(x_0)} \times \cdots \times A^{type(x_N)} \to A^{type(x_0)} \times \cdots \times A^{type(x_N)},$$

$$a = (a_0, \ldots, a_N) \mapsto ([\![S(x_0), \mathscr{A}]\!]\rho_a, \ldots, [\![S(x_N), \mathscr{A}]\!]\rho_a).$$

The semantics of S over $\mathscr{A}$ will be defined essentially as the main program component of the least-fixed point of $S_{\mathscr{A}}$.

Note that by Definition 3.2 the type of the main program may be any ground type of the form $(e, \ldots, (e, i), \ldots)$. In order to make the semantics comparable, we apply to the appropriate number of empty parameter lists.

Notation. (10) For $a \in A^\tau$ with $\tau = (\alpha_{n-1}, \ldots, (\alpha_0, i) \cdots)$,

$$a\downarrow_A := a(\perp_{A^{\alpha_{n-1}}}) \cdots (\perp_{A^{\alpha_0}}).$$

Definition 3.4. Let $S \in n-\lambda(\Omega)$, $\mathscr{A} \in \Delta\text{-alg}\,\Omega$.
The *semantics of S over the interpretation* $\mathscr{A}$ is defined by

$$[\![S, \mathscr{A}]\!] := pr_1(\mu S_{\mathscr{A}})\downarrow_A.$$

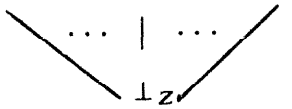
In the following example we shall define an interpretation *Algol* which, together with the standard semantics of typed λ-schemes and the translation indicated in the introduction, defines a denotational semantics for a subset of ALGOL 68 with finite modes.

Example. Let I, Ω be given as in the previous example.
(1) The carrier of *Algol* of type

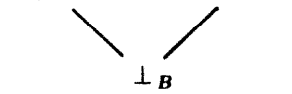
i $\quad$ $\boldsymbol{S} \to \mathbb{Z}$
b is $\boldsymbol{S} \to \mathbb{B}$
l $\quad$ $\boldsymbol{L}$
c $\quad$ $\boldsymbol{C}$

where

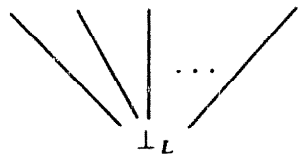
$\mathbb{Z} =$ $\cdots$ -1 0 1 $\cdots$ *integers*



$\mathbb{B} =$ **true** **false** *booleans*



$\alpha \in \boldsymbol{L} =$ loc loc1 loc2 $\cdots$ *locations*



$\sigma \in \boldsymbol{S} = \boldsymbol{L} \to \mathbb{Z}$ *store*

$\theta \in \boldsymbol{C} = \boldsymbol{S} \to \boldsymbol{S}$ *continuations*

(2) The assignment function of *Algol* is given by

$n \mapsto ((\,) \mapsto (\sigma \mapsto n))$ $(n \in \omega)$

$- \mapsto ((\kappa_1, \kappa_2) \mapsto (\sigma \mapsto \kappa_1(\sigma) - \kappa_2(\sigma)))$

(*strict* extension; square, ↑ accordingly)

$= \mapsto ((\kappa_1, \kappa_2) \mapsto (\sigma \mapsto$ strict equality of $\kappa_1(\sigma), \kappa_2(\sigma))$

$\mathrm{loc}j \mapsto ((\,) \mapsto (\sigma \mapsto \mathrm{loc}j))$ $(j \in \omega)$

$\mathrm{content} \mapsto (\alpha \mapsto (\sigma \mapsto \sigma(\alpha)))$

$\mathrm{assign} \mapsto ((\alpha, \kappa) \mapsto (\sigma \mapsto \sigma[\alpha/\kappa(\sigma)]))$

$\mathrm{eval} \mapsto ((\theta, \kappa) \mapsto (\sigma \mapsto \kappa(\theta(\sigma))))$

$\mathrm{cond} \mapsto ((\beta, \theta_1, \theta_2) \mapsto (\sigma \mapsto (\beta(\sigma) \to \theta_1(\sigma), \theta_2(\sigma)))$ ('strict' conditional)

$; \mapsto ((\theta_1, \theta_2)) \mapsto (\sigma \mapsto \theta_2(\theta_1(\sigma)))$.

It is but an exercise in denotational semantics to check that the semantics of the sample scheme over this interpretation satisfies

$[\![S, Algol]\!](\sigma)(\mathrm{loc}3) = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ (with $\sigma(\mathrm{loc}1)$ and $\sigma(\mathrm{loc}2)^2$ in the exponent)

Note that this connection allows us to lift problems from programming languages to schemes. In [18] we give a detailed treatment of this relation and apply this method to decidability of the macro- and formal-termination property [38].

4. *N*-rational schemes

Before we proceed to generalize the results of Section 2 to typed λ-schemes, we shall take a look at a combinatoric formulation of higher type recursion. This model is essentially algebraic in nature and thus allows easy proofs of algebraic results as the MW-theorem and closure under tree homorphisms. The equivalence of the two approaches proved in the appendix allows us to carry over these results to typed λ-schemes.

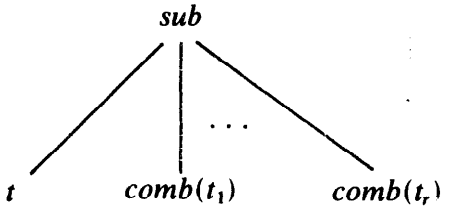
The key to the algebraic treatment of higher types is to view the function of an algebra as an algebra itself with projections, constant abstraction and substitution of functions added to the operations of the underlying algebra. The corresponding operation symbols *pr, abs, sub* are called *derived operation symbols*, cf. [42, 28]. In

this framework higher type procedures are described by regular systems of equations over (possibly n-times) derived operation symbols.

To illustrate these notions we shall describe by means of an example the translation of typed λ-schemes into the combinatoric model. The translation-mapping *comb* is based on the simple idea of replacing an application

$$t(t_1, \ldots, t_r)$$

by the substitution of functions



This makes it necessary to lift the type of the arguments $t_1, \ldots, t_r$ to the level of the function t. The necessary type-information is given by abstractions and memorized as a subscript to the translation function.

Notation. It is convenient to abbreviate the *integer types* in $D^*(\{i\})$ by $\mathbf{0} := i$, $\boldsymbol{n+1} := (\boldsymbol{n}, \boldsymbol{n})$.

Example. Let $I = \{i\}$, $\Omega = \{e: (e, i), a: \mathbf{1}, +: (ii, i)\}$, $X = \{x_0: (e, (e, i)), x_1: \mathbf{2}, x_2: \mathbf{2}\}$ and $S \in 2 - \lambda(\Omega)$ be defined by

$$S(x_0) := \lambda.\lambda.x_2(a)(e(\,)),$$

$$S(x_1) := \lambda y_1.\lambda y_0.y_1(y_1(y_0)),$$

$$S(x_2) := \lambda y_1.\lambda y_0.+(x_2(x_1(y_1))(y_0), y_1(y_0)).$$

The translation is done in two steps, each of which eliminates one level of abstractions. We abbreviate *comb* to c.

$$\begin{aligned} c(S(x_0)) &= \lambda.c(\lambda.x_2(a)(e(\,))) \\ &= \lambda.c_e(x_2(a)(e(\,))) \\ &= \lambda.sub(x_2(a'(\,)), c_e(e(\,))) \\ &= \lambda.sub(x_2(a'(\,)), abs_{(e,i)}(e'(\,))) \\ &\sim \lambda.sub(x_2(a'(\,)), e'(\,)), \end{aligned}$$

$$
\begin{aligned}
c(S(x_1)) &= \lambda y_1.c(\lambda y_0.y_1(y_1(y_0))) \\
&= \lambda y_1.c_0(y_1(y_1(y_0))) \\
&= \lambda y_1.sub(y_1, c_0(y_1(y_0))) \\
&= \lambda y_1.sub(y_1, sub(y_1, c_0(y_0))) \\
&= \lambda y_1.sub(y_1, sub(y_1, pr_1^0)) \\
&\sim \lambda y_1.sub(y_1, y_1), \\
c(S(x_2)) &= \lambda y_1.c(\lambda y_0.+(x_2(x_1(y_1))(y_0), y_1(y_0))) \\
&= \lambda y_1.c_0(+(x_2(x_1(y_1))(y_0), y_1(y_0))) \\
&= \lambda y_1.sub(+'(\ ), c_0(x_2(x_1(y_1))(y_0)), c_0(y_1(y_0))) \\
&= \lambda y_1.sub(+'(\ ), sub(x_2(x_1(y_1)), c_0(y_0)), sub(y_1, c_0(y_0))) \\
&= \lambda y_1.sub(+'(\ ), sub(x_2(x_1(y_1)), pr_1^0), sub(y_1, pr_1^0)) \\
&\sim \lambda y_1.sub(+'(\ ), x_2(x_1(y_1)), y_1).
\end{aligned}
$$

The scheme $c \circ S$ is an example of a *recursive program scheme* in the sense of [53]. It should be clear that the next step yields a regular system of equations $c \circ c \circ S$ (see also the next example).

We shall now define the derived alphabets and the corresponding algebras by induction on the functional level.

Definition 4.1. (1) Let Ω be a $D(I)$-set.

The *derived alphabet of* Ω, $D(\Omega)$, is the smallest $D^2(I)$-set with

$$
\begin{aligned}
&f \in \Omega^{(w,i)} \sim f' \in D(\Omega)^{(e,(w,i))}, \\
&w \in I^+, j \in [l(w)] \sim pr_j^w \in D(\Omega)^{(e,(w,w(j)))}, \\
&(w,i) \in D(I) \sim abs_{(w,i)} \in D(\Omega)^{((e,i),(w,i))}, \\
&v \in I^+, (w,i) \in D(I), l(v) = r \\
&\sim sub_{(w,i)}^v \in D(\Omega)^{((v,i)(w,v(1))\cdots(w,v(r)),(w,i))}.
\end{aligned}
$$

(2) Let $\mathcal{A} \in \Delta\text{-}alg\ \Omega$.

The *derived algebra of* $\mathcal{A}$, $D(\mathcal{A}) = (D(A), \varphi_{D(A)})$ is the Δ-continuous $D(\Omega)$-algebra with assignment function

$$
\begin{aligned}
&f' \mapsto ((\ ) \mapsto \varphi_A(f)), \\
&pr_j^w \mapsto ((\ ) \mapsto ((a_1, \ldots, a_n) \mapsto a_j)), \\
&abs_{(w,i)} \mapsto (a \mapsto ((a_1, \ldots, a_n) \mapsto a(\ ))), \\
&sub_{(w,i)}^v \mapsto ((g, g_1, \ldots, g_r) \mapsto (a \mapsto g(g_1(a), \ldots, g_r(a)))).
\end{aligned}
$$

To treat higher level function spaces, we iterate these constructions:

$$D^{n+1}(\Omega) := D(D^n(\Omega)), \qquad D^{n+1}(\mathscr{A}) := D(D^n(\mathscr{A})).$$

We can now formally define the combinatoric model of higher type recursion. The class of n-rational schemes consists of those regular equations over $D^n(\Omega)$ whose defining equation is of the ground type

$$b_n(i) := \underbrace{(e, \ldots, (e, i) \ldots)}_{n}.$$

The semantics of such a scheme is simply the ( )-application of the semantics of the underlying regular system of equations over the nth derived algebra.

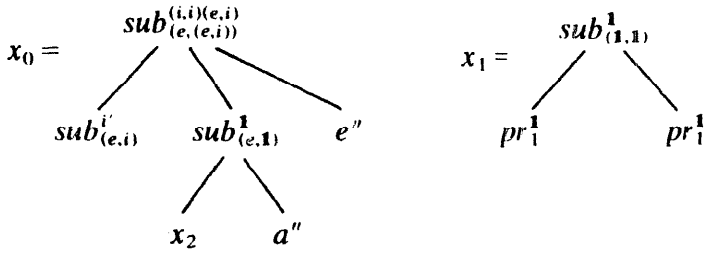
Definition 4.2. (1) The *class of n-rational schemes over* Ω of type $i \in I$ is defined by

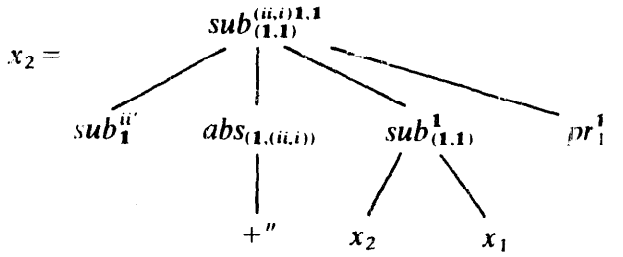
$$n - R(\Omega)^i := R(D^n(\Omega))^{b_n(i)}.$$

(2) Let $\mathscr{A} \in \Delta\text{-}alg\,\Omega$, $S \in n - R(\Omega)$.
The *semantics of S over the interpretation* $\mathscr{A}$ is defined by

$$[\![S, \mathscr{A}]\!] := [\![S, D^n(\mathscr{A})]\!]\downarrow_A.$$

Example. Let Ω and $S \in 2 - \lambda(\Omega)$ be as in the previous example. The scheme $c \circ c \circ S \in 2 - R(\Omega)$ is equivalent to the scheme S' with



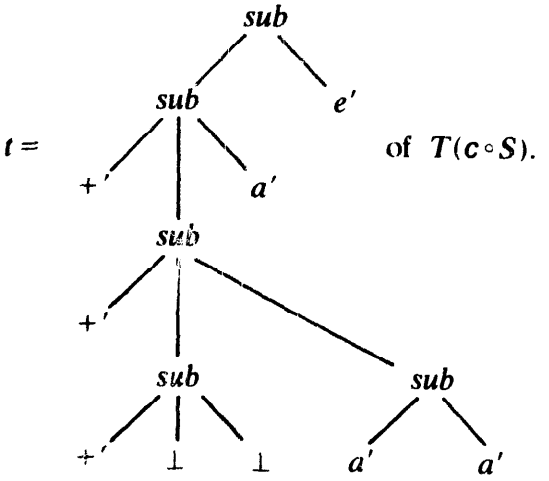


It is easy to see that the semantics of S' over Pa^* (with + denoting union and e denoting $\{e\}$) is equal to the macro language $\{a^{2^m} \mid m \in \omega\}$.

By the MW-theorem for $R(D^n(\Omega))$, the semantics of an n-rational scheme is up to ( )-application equal to the homomorphic image of $T(S) \in \mathrm{CT}_{D^n(\Omega)}^{b_n(i)}$. However, it should be obvious that $T(S)$ no longer characterizes the equivalence class of S, since we are dealing with partially interpreted schemes. Instead, we expect the semantics of S to be uniquely determined by the semantics over the initial interpretation $\mathscr{CT}_\Omega$. We now show that this tree can be obtained purely syntactically from $T(S)$ by interpreting *sub* as substitution of infinite trees. This translation will be described by a mapping *yield*.

Example. Let Ω and S be as above.

Consider the approximation



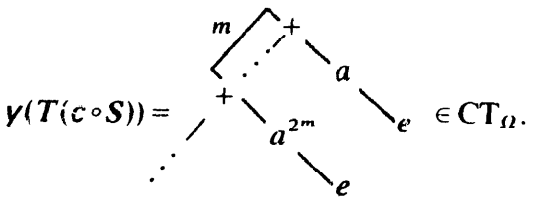
Then, abbreviating *yield* to γ, we have

$$\gamma(t) = \gamma(sub+'\ sub+'\ sub+'\perp\perp\ suba'a'\ a') \leftarrow e$$

$$= (+y_{1,i}\ y_{2,i} \leftarrow (\gamma(sub+'\ sub+'\perp\perp\ suba'a'), \gamma(a'))) \leftarrow e$$
$$= (+\ +y_{1,i}\ y_{2,i} \leftarrow (\gamma(sub+'\perp\perp), \gamma(suba'a'))\ ay_{1,i}) \leftarrow e$$

$$= +\ +\ +\perp\perp\ aay_{1,i}\ ay_{1,i} \leftarrow e = +\ +\ +\perp\perp\ aae\ ae.$$

The reader can check



Definition 4.3 ([42, 28]). Let *der*-$\mathscr{CT}_\Omega$ be the Ω-continuous $D(\Omega)$-algebra with carrier $(\mathrm{CT}_\Omega(Y_w)^i \mid (w, i) \in D(I))$ and assignment function φ_{der} given by

$f' \mapsto ((\ ) \mapsto f y_w)$ for $f \in \Omega^{(w,i)}$,

$pr_j^w \mapsto ((\ ) \mapsto y_{j,w(j)})$,

$abs_{(w,i)} \mapsto (t \mapsto t)$,

$sub^v_{(w,i)} \mapsto ((t, t_1, \ldots, t_r) \mapsto t \leftarrow (t_1, \ldots, t_r))$.

The unique Δ-continuous $D(\Omega)_\perp$-homomorphism $\mathscr{CT}_{D(\Omega)} \to der\text{-}\mathscr{CT}_\Omega$ will be denoted *yield*.

For the proof of the 'sharper' MW-result we need the following lemma.

Lemma 4.4.

$$derop_{\mathscr{A}} : der\text{-}\mathscr{CT}_\Omega \to D(\mathscr{A})$$

is a Δ-continuous $D(\Omega)_\perp$-homomorphism.

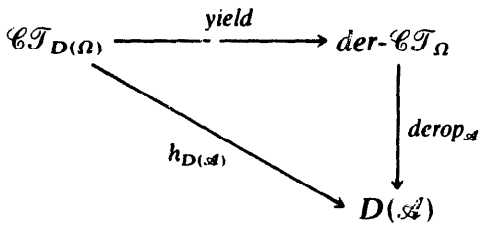
Proof. Strictness and continuity carry over from $h_{\mathscr{A}_a}$ (see Section 1.3).

The only interesting case is to check the homomorphism property for *sub*, which is immediate by the commutativity of $\leftarrow$ and $derop_{\mathscr{A}}$ proved in Lemma 1.3. □

Since both $h_{D(\mathscr{A})}$ and $derop_{\mathscr{A}} \circ yield$ are strict Δ-continuous $D(\Omega)$-homomorphism, we have by initiality of $\mathscr{CT}_{D(\Omega)}$

Corollary 4.5. *Let $\mathscr{A} \in \Delta\text{-}alg\ \Omega$.*

Then



commutes.

As $der\text{-}\mathrm{CT}^{b_{m+1}(i)}_{D^{m+1}(\Omega)} = \mathrm{CT}^{b_m(i)}_{D^m(\Omega)}$, we can compose the translations γ to a map

$$yield^{(n)} : \mathrm{CT}^{b_n(i)}_{D^n(\Omega)} \to \mathrm{CT}^i_\Omega.$$

Theorem 4.6. *Let $\mathscr{A} \in \Delta\text{-}alg\ \Omega$, $S \in n - R(\Omega)$.*

Then

$$[\![S, \mathscr{A}]\!] = h_{\mathscr{A}} \circ yield^{(n)}(T(S)).$$

Proof. Let $n > 0$. By Corollary 4.5 we have for any $t \in \mathrm{CT}^{b_{m+1}(i)}_{D^{m+1}(\Omega)}$ with $m < n$

$$h_{D^m(\mathscr{A})}(\gamma(t)) = h_{D^{m+1}(\mathscr{A})}(t)(\ ). \qquad (*)$$

But then

$$\begin{aligned}[\![S, \mathscr{A}]\!] &= [\![S, D^n(\mathscr{A})]\!]\downarrow_A \\ &= h_{D^n(\mathscr{A})}(T(S))\downarrow_A \quad \text{by Theorem 2.3} \\ &= h_{\mathscr{A}}(\gamma^{(n)}(T(S))) \quad \text{by } (*). \qquad \square\end{aligned}$$

For the case $\mathscr{A} = \mathscr{CT}_\Omega$, the above theorem shows $\gamma^{(n)}(T(S)) = [\![S, \mathscr{CT}_\Omega]\!]$, hence the semantics of an n-rational scheme is just the homomorphic image of the semantics of the scheme over the initial interpretation.

As a typical application of the MW-theorem, we now prove that the class

$$n\text{-}\mathscr{T} := \{\gamma^{(n)}(T(S)) \mid S \in n - R(\Omega) \text{ for some } \Omega\}$$

of *n-rational trees* is closed under tree homomorphisms. Since $\mathscr{T}$ satisfies this property, it is sufficient to show that tree homomorphisms can be lifted via *yield*. The next two lemmata prove that this can be done.

Lemma 4.7. *Let Ω, Σ be $D(I)$-sets, $\tilde{D} := D(\Omega)\backslash\Omega'$, and let $\hat{\sigma}$ be a tree homomorphism canonically extended to parameters. Then*

$$\hat{\sigma} : der\text{-}\mathscr{CT}_\Omega \to der\text{-}\mathscr{CT}_\Sigma$$

is a Δ-continuous $\tilde{D}_\perp$-homomorphism.

Proof. Strictness and continuity follow by definition. The only interesting case is again the homomorphism property for *sub*:

$$\hat{\sigma}(t \leftarrow s) = \hat{\sigma}(t) \leftarrow \hat{\sigma}(s) \quad \text{for } t \in \mathrm{CT}_\Omega(Y_v)^i,\ s \in \mathrm{CT}_\Omega(Y_w)^v.$$

Since $\hat{\sigma}$ is continuous, this can be checked by induction on t, using associativity of substitution (Lemma 1.4). $\square$

The next lemma is taken from [28].

Lemma 4.8. *Let $\sigma : D(\Omega) \to \mathrm{CT}_{D(\Sigma)}(Y)$ be a $D^2(I)$-mapping satisfying*

$$\sigma(pr_j^w) = pr_j^w, \qquad \sigma(abs_{(w,i)}) = abs_{(w,i)}\downarrow$$

$$\sigma(sub_{(w,i)}^v) = sub_{(w,i)}^v\downarrow,$$

and let

$$\sigma_\gamma := \gamma \circ \sigma \restriction \Omega'.$$

Then

$$\gamma \circ \hat{\sigma} = \hat{\sigma}_\gamma \circ \gamma.$$

Proof. Immediate by Lemma 4.7 and initiality of $\mathscr{CT}_{\tilde{D}(\Omega')}$. $\square$

The proof of the closure property is now straightforward.

Theorem 4.9. *$n-\mathcal{T}$ is closed under tree-homomorphisms.*

Proof. Let $n>0$ and $S\in n-R(\Omega)$, $\sigma:\Omega\to \mathrm{CT}_\Sigma(Y)$ be a $D(I)$-mapping with

$$\Omega^{(w,i)}\ni f\mapsto y^{(n)}(T(S_f))\in \mathrm{CT}_\Sigma(Y_w)^i$$

for some $S_f\in n-R(\Sigma)^{(w,i)}$ $(:= R(D^n(\Sigma))^{(e,\ldots,(w,i)\cdots)})$.

For $k\in\{0,\ldots,n\}$, define $D^{k+1}(I)$-mappings

$$\sigma_k: D^k(\Omega)\mapsto \mathrm{CT}_{D^k(\Sigma)}(Y)$$

by

$$\overbrace{f'\ldots'}^{k}\mapsto \overbrace{y\circ\cdots\circ y}^{n-k}(T(S_f)),$$

$$D^k(\Omega)\backslash\overbrace{\Omega'\ldots'}^{k}\ni s\mapsto s(y_\alpha)\quad \text{where } type(s)=(\alpha,\nu).$$

Clearly $\sigma_0=\sigma$. Moreover, the σ_k satisfy the assumptions of Lemma 4.8 since $\sigma_{k-1}=(\sigma_k)_\gamma$. Thus we have

$$\begin{aligned}&\hat{\sigma}(y^{(n)}(T(S)))\\ &=y^{(n)}(\hat{\sigma}_n(T(S)))\quad \text{by Lemma 4.8}\\ &=y^{(n)}(T(S'))\quad \text{for some } S'\in n-R(\Sigma) \text{ by Theorem 2.9.}\ \square\end{aligned}$$

We hope that the simplicity of the algebraic proofs justifies the introduction of the combinatoric model.

Before we turn to an operational characterization of n-rational trees, we state the equivalence of the two models and list as consequences the MW- and normal-form theorems for typed λ-schemes.

From the introductory example of this section it should be clear that level-n λ-schemes can be translated (by n applications of *comb*) into n-rational schemes. To prove the converse, simply replace the combinators by the corresponding λ-expressions—e.g. $pr_j^w\mapsto \lambda.\lambda y_w.y_{j,w(j)}$. It should be obvious that this translation, called γ_λ, preserves equivalence. Moreover, both translations are *size-preserving*: the size of the translation is polynomially bounded by the size of the scheme.

Theorem 4.10. $n-R(\Omega)\sim n-\lambda(\Omega)$.

The proof is given in the appendix.

As an immediate consequence we obtain the MW-result for typed λ-schemes.

We note that an appropriate extension of the unique homomorphism to higher types allows a direct proof along the lines indicated in Section 2.

Corollary 4.11. *For* $S \in n-\lambda(\Omega)$, *let* $T(S) := [\![S, \mathscr{CT}_\Omega]\!]$.
Then

$$\forall \mathscr{A} \in \Delta\text{-}alg\,\Omega \quad [\![S, \mathscr{A}]\!] = h_{\mathscr{A}}(T(S)).$$

In contrast, a direct proof of the following *Chomsky-Normalform-Theorem* seems to be more complicated.

By the normalform result for regular equations, the *comb*-translation of a typed λ-scheme is equivalent to a system of regular equations where the right-hand sides consist of either a combinator or a base operation symbol or a procedure identifier or a combinator applied to procedure identifiers. Translating the combinators into the corresponding λ-expressions gives the following normalform for typed λ-schemes.

Corollary 4.12. *Let* $S \in n-\lambda(\Omega)$. *Then there exists an equivalent* $S' \in n-\lambda(\Omega)$ *such that*

(1) $size(S') \leq p(size(S))$ *for some polynomial* p.

(2) $\forall x \in var(S')$ *with* $type(x) = (\alpha_m, \ldots, (\alpha_0, i)\ldots)$

$$x\downarrow = f(y_{\alpha_0}) \quad \text{with } f \in \Omega, \quad \alpha_j = e \text{ for } j \geq 1$$

or $$= x'(y_{\alpha_m})\cdots(y_{\alpha_0}) \text{ or } x'(\,)(y_{\alpha_{m-1}})\cdots(y_{\alpha_0}) \quad \text{with } x' \in var(S')$$

or $$= y_{i,\alpha_{p(j)}}(y_{\alpha_{p-1}})\cdots(y_{\alpha_0}) \quad \text{with } \alpha_{p(j)} = (\alpha_{p-1}, \ldots, (\alpha_0, i)\ldots)$$

or $$= x_1(x_2(y_{\alpha_p}), \ldots, x_r(y_{\alpha_p}))(y_{\alpha_{p-1}})\cdots(y_{\alpha_0}) \quad \text{with } x_j \in var(S')$$

or $$= y_{1,\alpha_p(1)}(y_{2,\alpha_p(2)}(y_{\alpha_{p-1}}), \ldots, y_{k,\alpha_p(k)}(y_{\alpha_{p-1}}))(y_{\alpha_{p-2}})\cdots(y_{\alpha_0}) \quad \text{with } p \leq m \wedge l(\alpha_p) = k.$$

In particular, it is sufficient to consider applicative terms as right-hand sides, hence $1-\lambda(\Omega)$ is equivalent to the class RRS(Ω) of recursive program schemes over Ω.

5. The Kleene characterization

As an intermediate step towards an operational semantics, we shall generalize the Kleene characterization theorem to typed λ-schemes.

Recall that the Kleene sequence of a regular system of equations was generated by iterated substitution of the procedure bodies for the corresponding procedure identifiers. Finally, the procedure identifiers were eliminated by substituting the nowhere defined procedure. The generalization of this idea to typed λ-schemes is

straightforward, except that we have to take care of parameter passing: the canonical extension $\hat{S}$ of a scheme as a mapping on typed λ-terms defines only a textual substitution of procedure bodies without evaluating actual parameters.

Example. Consider the scheme $S \in 2-\lambda(\Omega)$ of the previous example.
Then, e.g.

$$\hat{S}^2(x_0) = \lambda.\lambda.\lambda y_1.\lambda y_0.+(x_2(x_1(y_1))(y_0), y_1(y_0))(a)(e(\ ))$$

rather than

$$\lambda.\lambda.+(x_2(x_1(a))(e(\ )), a(e(\ )))$$

and

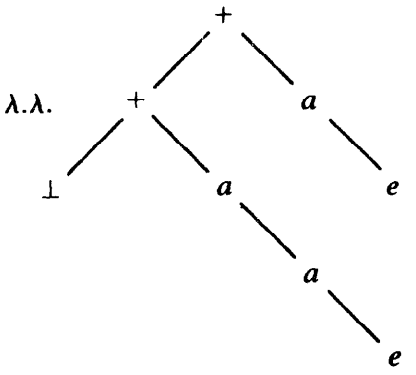
$$\hat{\perp}(\hat{S}^3(x_0)) = \lambda.\lambda.\lambda y_1.\lambda y_0.+(s_2(s_1(y_1))(y_0), y_1(y_0))(a)(e(\ ))$$

where

$$s_2 = \hat{\perp}(\hat{S}(x_2)) = \lambda y_1.\lambda y_0.+(\perp_2(\perp_1(y_1))(y_0), y_1(y_0)),$$

$$s_1 = \hat{\perp}(\hat{S}(x_1)) = \lambda y_1.\lambda y_0.y_1(y_1(y_0))$$

rather than



which we would expect since

$$\hat{\perp}(\hat{S}^3(x_0)) \xrightarrow[\beta]{*} \lambda.\lambda.+(s_2(s_1(a))(e(\ )), a(e(\ )))$$

$$\xrightarrow[\beta]{*} \lambda.\lambda.+(+(\perp_2(\perp_1(s_1(a)))(e(\ )), s_1(a)(e(\ ))), a(e(\ )))$$

$$\xrightarrow[\perp]{*} \lambda.\lambda.+(+(\perp, s_1(a)(e(\ ))), a(e(\ )))$$

$$\xrightarrow[\beta]{*} \lambda.\lambda.+(+(\perp, a(a(e(\ )))), a(e(\ ))).$$

The reason for choosing the λ-calculus as a formal model was of course that it has the appropriate operational semantics to deal with parameter passing. As indicated in the example, the evaluation of actual parameters is split into elementary steps called *β-reductions*. Since the nowhere defined procedure $\perp_\tau$ of type $\tau = (\alpha_m, \ldots, (\alpha_0, i) \ldots)$ is represented by $\lambda y_{\alpha_m} \cdots \lambda y_{\alpha_0} \perp_i$, $\perp$-reductions are a special case of β-reductions.

We now list some basic operational notions and properties of the λ-calculus as needed in the sequel.

Definition 5.1. Let $t, t', s_j \in T_{\Omega,X,Y}$.

(1) For $a = (a_1, \ldots, a_k) \in (X \cup Y)^k$ with $type(a_j) = type(s_j)$, we denote by $t[a/(s_1, \ldots, s_k)]$ the term obtained from t by substituting s_j for all free occurrences of a_j in t.

(2) A *β-redex* is any term $\lambda y_\alpha . t(s_1, \ldots, s_k)$ with $level(\alpha) = level(t)$.

(3) t is *in normalform* iff t contains no β-redex.

(4) The set of *bound parameters of t* is defined by $bound(a) = \emptyset$ for $a \in \Omega \cup X \cup Y$, $bound(s_0(s_1, \ldots, s_v)) = \bigcup_{0 \le j \le v} bound(s_j)$, $bound(\lambda y_\alpha . t') = bound(t') \cup Y_\alpha$.

(5) Let $\Lambda := \Omega \cup X \cup Y \cup \{\lambda, ., ,, (,)\}$. A *context* of type τ, τ' is a pair $C = (\gamma, \gamma') \in \Lambda^* \times \Lambda^*$ such that for all t of type τ

$$C[t] := \gamma t \gamma' \in T^{\tau'}_{\Omega,X,Y}.$$

(6) t is *β-reducible* to t' $(t \to t')$ iff there exists a context C and a redex $R = \lambda y_\alpha . s(s_1, \ldots, s_k)$ such that

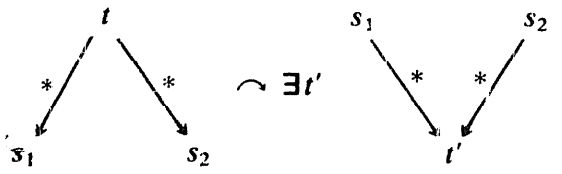
$$\forall j \in [n] \quad free(s_j) \cap bound(s) = \emptyset \wedge t = C[R] \wedge t' = C[s[y_\alpha/(s_1, \ldots, s_k)]].$$

One can always ensure that a β-redex can be reduced by renaming of bound variables, called *α-conversion*. As usual we shall identify two terms if they are identical up to α-conversion.

Fact 5.2. β-reduction preserves semantics. In particular

$$[\![t, \mathscr{A}]\!]\rho[a/([\![s_1, \mathscr{A}]\!]\rho, \ldots, [\![s_k, \mathscr{A}]\!]\rho)] = [\![t[a/(s_1, \ldots, s_k)], \mathscr{A}]\!]\rho.$$

Fact 5.3 (Church–Rosser property).



Fact 5.4 ([50]). In the typed λ-calculus, any term t has a unique normalform $nf(t)$ which can be obtained from t by an arbitrary reduction strategy.

The above definitions cover the case of λ-terms in $FT_{\Omega,X,Y} := T_{\Omega_\perp,X,Y}$ arising as approximations. We sometimes refer to the reduction of $t = C[\perp_{(\alpha,\nu)}(s)]$ to $t' = C[\perp_\nu]$ as a *$\perp$-reduction* ($t \to_\perp t'$). Terms in $FT_{\Omega,X,Y}$ are ordered canonically: $t \leqslant t'$ iff $t = \perp_i$ for some $i \in I$ or $t = t_0(t_1, \ldots, t_r)$, $t' = t'_0(t'_1, \ldots, t'_r) \wedge \forall j \in \{0, \ldots, r\} t_j \leqslant t'_j$ or $t = \lambda y_\alpha.s$, $t' = \lambda y_\alpha.s' \wedge s \leqslant s'$. It is easy to see that the normalform operator is monotone with respect to this ordering.

Lemma 5.5. $t \leqslant t' \curvearrowright nf(t) \leqslant nf(t')$.

Proof. Clearly $\leqslant$ can be extended canonically to contexts. We prove first

$$t \leqslant t' \wedge t \to s \curvearrowright \exists s' \geqslant s \; t' \to^{=} s'. \tag{*}$$

Case 1: $\to = \to_\perp$. Take $s' := t$.

Case 2: no $\perp$-reduction

$$\curvearrowright t = C[\lambda y_\alpha.t_0(t_1, \ldots, t_r)] \wedge s = C[t_0[y_\alpha/(t_1, \ldots, t_r)]].$$

By assumption $\lambda y_\alpha.t_0 \neq \perp_{(\alpha,\nu)}$, hence since $t \leqslant t'$ there exists a context C' and terms $t'_0, \ldots, t'_r$ s.t.

$$t' = C'[\lambda y_\alpha.t'_0(t'_1, \ldots, t'_r)], \qquad C' \geqslant C, \qquad t'_j \geqslant t_j.$$

Assume (without loss of generality) that the redex in T' can be reduced to give

$$s' := C'[t'_0[y_\alpha/(t'_1, \ldots, t'_r)]].$$

A trivial induction on t_0 proves $s \leqslant s'$ and thus (*).

By a simple induction on the length of the reduction sequence using (*) we obtain

$$t \leqslant t' \curvearrowright \exists s' \geqslant nf(t)\; t' \xrightarrow{*} s'.$$

By definition of $\leqslant$ all β-redexes s' are situated at $\perp$-positions of $nf(t)$, hence $nf(t) \leqslant nf(t')$. □

By now it should be clear that the evaluation of actual parameters can be described by taking the normalform of a term. Hence the following definition is the proper generalization of the Kleene-sequence to typed λ-schemes.

Definition 5.6. (1) Let $S \in n - \lambda(\Omega)$ with $var(S) = \{X_0, \ldots, X_N\}$.

S uniquely determines a $D^*(I)$-mapping

$$\hat{S} : T_{\Omega,X,Y} \to T_{\Omega,X,Y}$$

by

$$\hat{S}(f) := f, \qquad \hat{S}(y_{j,\nu}) := y_{j,\nu}, \qquad \hat{S}(x_j) := S(x_j),$$

$$\hat{S}(t(t_1, \ldots, t_r)) := \hat{S}(t)(\hat{S}(t_1), \ldots, \hat{S}(t_r)),$$

$$\hat{S}(\lambda y_\alpha.t) := \lambda y_\alpha.\hat{S}(t).$$

(2) Let $\hat{\perp}$ be generated by $x_j \mapsto \perp_{type(x)}$.

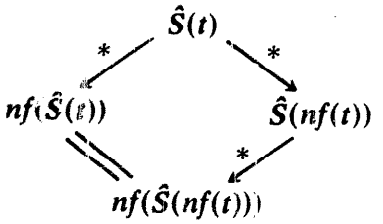
The *Kleene-sequence generated by S*, $K(S)$, is defined by

$$K(S)(m) := nf \circ \hat{\perp} \circ \hat{S}^m(x_0\downarrow).$$

The above definition allows a number of syntactic variations. In particular by the Church–Rosser property, we can freely mix taking normalforms and expanding according to the definitions of a scheme.

Lemma 5.7. $nf \circ \hat{S} \circ nf = nf \circ \hat{S}$.

Proof. By induction on t we have $t \to t' \curvearrowright \hat{S}(t) \to^* \hat{S}(t')$, hence by the Church–Rosser property



□

The following lemma gives the connection between syntactic and semantic approximations of the infinite tree.

Let $\rho_\perp$ denote the bottom environment, and $t\downarrow$ be the term obtained from t by applying t to all its $\perp$-arguments:

$$type(t) = (\alpha_m, \ldots, (\alpha_0, i)\ldots) \curvearrowright t\downarrow = t(\perp_{\alpha_m}) \cdots (\perp_{\alpha_0}).$$

Lemma 5.8. *For all closed* $t \in T_{\Omega,X,Y}$ *in normalform*

$$[\![t, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT} = nf \circ \hat{\perp}(t\downarrow).$$

Proof. By induction on the structure of closed terms in normalform.

– $t = \lambda y_{\alpha_m} \cdots \lambda y_{\alpha_1}.f$ with $f \in \Omega^{(w,i)}$, $\alpha_j \in D^j(I)^*$:

$$[\![t, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT} = f(\perp_w)$$
$$= nf(\lambda y_{\alpha_m} \cdots \lambda y_{\alpha_1}.f(\perp_{\alpha_m}) \cdots (\perp_{\alpha_1})(\perp_w)).$$

– $t = \lambda y_{\alpha_m} \cdots \lambda y_{\alpha_k}.x$:

$$[\![t, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT} = \perp_{CT^{type(t)}}\downarrow_{CT}$$
$$= \perp_{CT} \quad \text{of level 0}$$
$$= nf(\lambda y_{\alpha_m} \cdots \lambda y_{\alpha_k}.\perp_{type(x)}\downarrow).$$

– $t = \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_k}. y_{j,\nu}$ with $\alpha_k(j) = \nu$:

$$[\![t, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT} = \perp_{CT^{type(t)}}\downarrow_{CT}$$
$$= \perp_{CT} \quad \text{of level 0}$$
$$= nf(\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_k}. y_{j,\nu}\downarrow) \quad \text{since } \alpha_k(j) = \nu.$$

– $t = \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. a(\ )$ with $a \in \Omega^{(e,i)}$: follows as in the first case.
– $t = \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. f(t_1, \ldots, t_r)$ with $f \in \Omega^{(w,i)}$, $\alpha_j \in D^j(I)^*$ s.t. $\forall j \in [r]\ \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_j$ is closed and in normalform:

$$[\![t, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT}$$
$$= [\![f(t_1, \ldots, t_r), \mathscr{CT}_\Omega]\!]\rho_\perp[y_{\alpha_m}/\perp_{CT}\alpha_m] \cdots [y_{\alpha_0}/\perp_{CT}\alpha_0]$$
$$= f([\![t_1, \mathscr{CT}_\Omega]\!]\rho_\perp[y_{\alpha_m}/\perp_{CT}\alpha_m] \cdots [y_{\alpha_0}/\perp_{CT}\alpha_0], \ldots,$$
$$[\![t_r, \mathscr{CT}_\Omega]\!]\rho_\perp[y_{\alpha_m}/\perp_{CT}\alpha_m] \cdots [y_{\alpha_0}/\perp_{CT}\alpha_0])$$
$$= f([\![\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_1, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT}, \ldots,$$
$$[\![\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_r, \mathscr{CT}_\Omega]\!]\rho_\perp\downarrow_{CT})$$
$$= f(nf \circ \hat{\perp}(\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_1\downarrow), \ldots,$$
$$nf \circ \hat{\perp}(\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_r\downarrow))$$
$$= nf \circ \hat{\perp}(f(\lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_1\downarrow, \ldots, \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_0}. t_r\downarrow))$$
$$= nf \circ \hat{\perp}(t\downarrow).$$

– $t = \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_k}. x(s_p) \cdots (s_k)$ with $level(x) = p + 1$, $level(s_j) = j$, $level(\alpha_r) = r$ and

$t = \lambda y_{\alpha_m}. \cdots \lambda y_{\alpha_k}. y_{j,\nu}(s_r) \cdots (s_k)$ with $\alpha_{r+1}(j) = \nu$, $k \leq r + 1 \leq m$, $level(s_j) = j = level(\alpha_j)$:

both sides reduce to $\perp_{CT}$ of level 0. □

We can now prove the Kleene characterization theorem for typed λ-schemes: the infinite tree can be obtained by iterated expansion of all procedure calls, starting with the main program.

Theorem 5.9. *For all $S \in n\text{-}\lambda(\Omega)$, $K(S)$ is a chain and $T(S) = \bigsqcup K(S)$.*

Proof. A straightforward induction on m using Fact 5.2 shows that the semantics of the mth expansion in the bottom environment yields the mth approximation of the least fixed point of $S_{\mathscr{CT}_\Omega}$:

$$pr_j S^m_{\mathscr{CT}_\Omega}(\perp, \ldots, \perp) = [\![\hat{S}^m(x_j), \mathscr{CT}_\Omega]\!]\rho_\perp. \qquad (*)$$

But then

$$
\begin{aligned}
T(S) &= (\mathrm{pr}_1 \bigsqcup_{m\in\omega} S^m_{\mathscr{CT}_\Omega}(\bot,\ldots,\bot))\downarrow_{CT} \\
&= \bigsqcup_{m\in\omega} (\mathrm{pr}_1 S^m_{\mathscr{CT}_\Omega}(\bot,\ldots,\bot))\downarrow_{CT} \\
&= \bigsqcup_{m\in\omega} [\![\hat{S}^m(x_0), \mathscr{CT}_\Omega]\!]\rho_\bot\downarrow_{CT} && \text{by (*)} \\
&= \bigsqcup_{m\in\omega} [\![nf(\hat{S}^m(x_0)), \mathscr{CT}_\Omega]\!]\rho_\bot\downarrow_{CT} && \text{by Fact 5.2} \\
&= \bigsqcup_{m\in\omega} nf\circ\hat{1}(nf(\hat{S}^m(x_0))\downarrow) && \text{by Lemma 5.8} \\
&= \bigsqcup_{m\in\omega} nf\circ\hat{1}\circ\hat{S}^m(x_0\downarrow) && \text{by Lemma 5.7} \\
&= \bigsqcup K(S) && \text{since } x_0\downarrow\downarrow = x_0\downarrow. \quad \square
\end{aligned}
$$

6. A call-by-name operational semantics

In this chapter we prove the completeness of a leftmost-outermost reduction strategy to obtain the infinite tree of a typed λ-scheme. In terms of the programming language considered, this implies the equivalence of denotational and copy-rule semantics.

The section is organized as follows. We start with the definition of the schematic language generated by a typed λ-scheme: besides expanding procedure calls according to the (rewriting-) rules of the scheme and evaluation of actual parameters, we allow the substitution of the nowhere defined procedure of proper type. It is then easy to see that each tree in the Kleene sequence of a scheme can be generated by the associated schematic grammar. We proceed by stating the analogon of the standardization theorem which implies that any tree in the schematic language can be derived by leftmost-outermost reductions only. In such a derivation a textual substitution of the procedure body is always followed by a complete evaluation of all actual parameters. Such a sequence of reduction forms one OI-derivation step (as defined e.g. for the special case of context-free tree grammars). It is then straightforward to show that a tree generated by m OI-derivation steps is majorized by the mth approximation in the Kleene-sequence. By summarizing these results we obtain a characterization of the infinite tree of a scheme as the join of its schematic (OI-) language.

As in the regular case, we start by considering *nondeterministic* schemes and then specialize to schematic grammars.

A *level-n tree grammar* G with *terminals* Ω, *nonterminals* $X = \{x_0, \ldots, x_N\}$, *parameters* Y, and *axiom* $x_0\downarrow$ is simply a nondeterministic level-n scheme where each

procedure identifier in X may have a finite number of procedure bodies (the *productions*). We denote by $n-N\lambda(\Omega)$ the class of level-n tree grammars over Ω.

Definition 6.1. Let $G \in n-N\lambda(\Omega)$ and $t, t' \in T_{\Omega,X,Y}$.

Then t is *G-reducible to* t' ($t \to_G t'$) iff $t \to t'$ or there exists a context C and a nonterminal x such that $t = C[x]$ and $t' = C[s]$ for some $s \in G[x]$.

The *level-n tree language generated by G* is defined by

$$L(G) := \{t \in T_\Omega \mid x_0\downarrow \xrightarrow[G]{*} t\}.$$

For an example we refer to Section 8.1.

With each (deterministic) level-n scheme S over Ω we associate a level-n tree grammar $S_\perp$ over $\Omega_\perp$ by adding the productions $S_\perp(x_i) \ni \perp_{type(x_i)}$. The tree language $L(S_\perp) \subseteq FT_\Omega$ is called the *schematic tree language generated by S*. It is easy to see that each tree in the Kleene sequence can be generated by $S_\perp$:

Lemma 6.2. $\forall S \in n-\lambda(\Omega)\ K(S) \subseteq L(S_\perp)$.

Proof. By induction on $t \in T_{\Omega,X,Y}$ we have $t \to_{S_\perp}^* \hat{S}(t)$. From this we obtain immediately by induction on m

$$x_0\downarrow \xrightarrow[S_\perp]{*} \hat{S}^m(x_0\downarrow)$$

hence

$$x_0\downarrow \xrightarrow[S_\perp]{*} \hat{S}^m(x_0\downarrow) \xrightarrow[S_\perp]{*} \hat{\perp}(\hat{S}^m(x_0\downarrow)) \xrightarrow{*} K(S)(m). \quad \square$$

We now want to show that all trees in a level-n language are already derivable by either substituting the leftmost procedure identifier or reducing the leftmost β-redex. In the literature two such results are known: for the untyped λ-calculus this is a consequence of the standardization theorem [11] while for context-free tree languages, this theorem is formulated as equivalence of OI-(=outermost-innermost) and unrestricted derivations [30]. In fact, as indicated in the sequel, the second result is just a special case of the first.

In a certain sense (by solving the equations backwards by allowing explicitly a symbol for the fixed-point operator, see [18]), our language is just a sublanguage of the untyped λ-calculus, hence we shall expect the standardization theorem to hold for derivations in level-n grammars, even though it does not apply directly. Since the proof of this result follows the classical proof exactly, we shall only indicate how the proof as given in [49] can be adapted to the λ-calculus discussed in this paper.

The proof of completeness of leftmost reductions is a corollary to the standardization theorem. This states that any reduction can be simulated by a so-called standard reduction in which the redex of the $n+1$st derivation step is always

situated to the right of the nth redex. In contrast with leftmost reductions, not all redexes have to be reduced.

Definition 6.3. Let $D \equiv t_1 \to_G \cdots \to_G t_{n+1}$ be a derivation in a level-n grammar G and assume that the jth reduction-step is of the form

$$t_j = C_j[R_j] \xrightarrow[G]{} C_j[\bar{R}_j] = t_{j+1} \quad \text{for some context } C_j = (\gamma_j, \gamma_j').$$

D is a *standard derivation* iff $\forall j \in [n]\ \gamma_j \leqslant \gamma_{j+1}$.

Theorem 6.4 ([11]). *Let $G \in n - N\lambda(\Omega)$ and $t \to_G^* t'$. Then there exists a standard derivation from t to t'.*

Proof. The proof given in [49] carries over, if one specifies together with the set *redex*(t) of redexes occurring in some term t for each occurrence of a nonterminal symbol in t the right-hand side to be substituted for this occurrence. Moreover, in proofs one has to treat the additional case that the reduced redex consists of a nonterminal. Since the procedurebodies consist of closed terms, this case is always trivial. □

In particular, for any $t \in L(G)$, there exists a standard derivation $x_0\downarrow \to_G^* t$. Since t is in normalform and contains only terminal symbols, the standard derivation is in fact a leftmost derivation. As in a leftmost derivation of level-0 terms a procedure call is never expanded before all actual parameters are given, a substitution of a procedure body will always be followed by a sequence of β-reductions modelling the evaluation of the parameters. We shall view such a sequence of reductions which describes in fact an application of the copy-rule as one OI-derivation step.

In order to define OI-derivations we need some auxiliary functions.

Notation. (1) A tree $t \in T_\Omega(Y)$ is *linear in* $w \in I^*$ iff $t \in T_\Omega(Y_w)$ and each parameter occurs exactly once in t.

(2) If t is linear in w and $(s_1, \ldots, s_k) \in T^w_{\Omega,X,Y}$, we abbreviate $t[y_w/(s_1, \ldots, s_k)]$ to $t[s_1, \ldots, s_k]$.

(3) The head-atom of a *level*-0 term in normalform will be denoted *head*(t). Let *tail*(t) be the unique string such that $t = head(t) \cdot tail(t)$.

(4) Let $t \in T_{\Omega,X,Y}$ be a closed *level*-0 term in normalform. Then $(s, s_1, \ldots, s_k)$ is called a *linearization of* t iff

$$\exists w \in I^*\ s \text{ is linear in } w$$
$$\wedge\, \forall j \in [k]\ s_j \in T^{w(j)}_{\Omega,X,Y} \text{ is in normalform}$$
$$\wedge\, t = s[s_1, \ldots, s_k] \wedge head(s_j) \in X.$$

Note that for such t a linearization always exists which is unique up to renaming of parameters.

Definition 6.5. Let $G \in n-N\lambda(\Omega)$ and t be a closed level-0 term in normalform. *t is OI-derivable to t' in G* ($t \Rightarrow_{OI,G} t'$) iff there exists a linearization $(s_0, s_1, \ldots, s_k)$ of t with $k \geq 1$ s.t.

$$t' = s_0[nf(s\ tail(s_1)), s_2, \ldots, s_k] \quad \text{for some } s \in G(head(s_1)).$$

The *OI-language generated by G* is defined by

$$L_{OI}(G) := \{t \in T_\Omega \mid x_0\downarrow \overset{*}{\underset{OI,G}{\Longrightarrow}} t\}.$$

Example. Consider the sample scheme $S \in 2-\lambda(\Omega)$ of the previous example. Then

$$x_0(\ )(\ ) \underset{OI}{\Longrightarrow} y_{1,i}[nf(S(x_0)(\ )(\ ))] = x_2(a)(e(\ ))$$

$$\underset{OI}{\Longrightarrow} y_{1,i}[nf(S(x_2)(a)(e(\ )))]$$

$$= +(x_2(x_1(a))(e(\ )), a(e(\ )))$$

$$\underset{OI}{\Longrightarrow} +(y_{1,i}, a(e(\ )))[nf(S(x_2)(x_1(a))(e(\ )))]$$

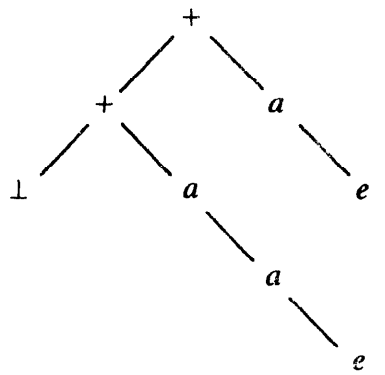
$$= +(+(x_2(x_1^2(a))(e(\ )), x_1(a)(e(\ ))), a(e(\ )))$$

$$\underset{OI}{\Longrightarrow} +(+(y_{1,i}, y_{2,i}), a(e(\ )))[nf(\perp_2(x_1^2(a))(e(\ ))), x_1(a)(e(\ ))]$$

$$= +(+(\perp_i, x_1(a)(e(\ ))), a(e(\ )))$$

$$\underset{OI}{\Longrightarrow} +(+(\perp_i, y_{1,i}), a(e(\ )))[nf(S(x_1)(a)(e(\ )))]$$

$=$



Since OI-derivations preserve the type and the properties of being closed and in normalform, any term t OI-derivable from the axiom can be decomposed according to some linearization.

As an immediate consequence of the standardization theorem we obtain the completeness of OI-derivations.

Corollary 6.6. $\forall G \in n-N\lambda(\Omega)\ L_{OI}(G) = L(G)$.

We can now show that the infinite tree of a typed λ-scheme can be characterized as the join of all approximations generated by its schematic grammar. In fact, we prove that a tree derived in m OI-derivation steps is majorized by the mth member of the Kleene sequence. Since the Kleene sequence is contained in the schematic language of S, it follows that $L(S_\perp)$ is directed. Together with the previous corollary and the Kleene characterization this yields:

Theorem 6.7. $\forall S \in n\text{-}\lambda(\Omega)\ L(S_\perp)$ *is directed and*

$$T(S) = \bigsqcup L(S_\perp).$$

Proof. By Theorem 5.9, Lemma 6.2 and Corollary 6.6 it is sufficient to prove

Assertion $*$. Let $\sigma_0, \ldots, \sigma_m$ be such that $x(\sigma_m)\cdots(\sigma_0)$ is a closed level-0 term in normalform and assume

$$x(\sigma_m)\cdots(\sigma_0) \overset{r}{\underset{\mathrm{OI}}{\Longrightarrow}} t \in \mathrm{FT}_{\Omega,X,Y}.$$

Then

$$nf \circ \hat{\mathbb{I}}(t) \leqslant nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(x(\sigma_m)\cdots(\sigma_0)).$$

We show $*$ by induction on r.

For $r = 0$ there is nothing to prove.

Consider a derivation

$$x(\sigma_m)\cdots(\sigma_0) \underset{\mathrm{OI}}{\Longrightarrow} t' \overset{r}{\underset{\mathrm{OI}}{\Longrightarrow}} t,$$

and let $(s, s_1, \ldots, s_k)$ be a linearization of t'. Then there exist $t_1, \ldots, t_k \in \mathrm{FT}_{\Omega,X,Y}$ s.t.

$$\forall j \in [k]\ \ s_j \overset{\leqslant r}{\underset{\mathrm{OI}}{\Longrightarrow}} t_j \wedge t = s[t_1, \ldots, t_k].$$

By induction hypothesis and monotonicity of nf we have

$$\forall j \in [k]\ \ nf \circ \hat{\mathbb{I}}(t_j) \leqslant nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(s_j).$$

But then

$$\begin{aligned}
&nf \circ \hat{\mathbb{I}}(t) \\
&\quad = s[nf(\hat{\mathbb{I}}(t_1)), \ldots, nf(\hat{\mathbb{I}}(t_k))] \\
&\quad \leqslant s[nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(s_1), \ldots, nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(s_k)] \\
&\quad = nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(s[s_1, \ldots, s_k]) \\
&\quad = nf \circ \hat{\mathbb{I}} \circ \hat{S}^r(nf(S(x)(\sigma_m)\cdots(\sigma_0))) \qquad \text{by definition of } t'
\end{aligned}$$

$$= nf \circ \hat{1} \circ \hat{S}^r(S(x)(\sigma_m) \cdots (\sigma_0)) \qquad \text{by iterated application of Lemma 5.7}$$

$$\leqslant nf \circ \hat{1} \circ \hat{S}^r(S(x)(\hat{S}(\sigma_m)) \cdots (\hat{S}(\sigma_0)))$$

$$= nf \circ \hat{1} \circ \hat{S}^{r+1}(x(\sigma_m) \cdots (\sigma_0)). \qquad \square$$

Remark. It is easy to see, that schematic grammars in Chomsky-normalform generate all approximations of the infinite tree of the corresponding scheme (Lemma 2 in [5]), hence, as in the regular case, the equivalence class of a scheme can be characterized purely operational by a tree language. For the special case $n = 1$ of recursive program schemes it is known, that the equivalence problem for schemes is reducible to the equivalence problem for deterministic pushdown automata [8]. A generalization of this result to higher levels using deterministic level-n pushdown automata [44] seems to be possible using the results in [21] and Subsection 7.4.

We conclude this section with two applications of this result for particular interpretations.

In the event that the semantic of a scheme is taken over a discrete interpretation (where the carrier is a flat cpo and **if** $\cdots$ **then** $\cdots$ **else** $\cdots$ is the only non-strict operation) we can compute the semantics by repeating OI derivation steps until the image of the approximation under the unique homomorphism becomes defined. If (and only if) this happens, the semantic of the scheme is equal to this value, otherwise it is undefined.

For the programming language considered, this result states the equivalence of denotational and copy-rule semantics: since β-reductions preserve semantics, we already know that the copy-rule semantics is correct (with respect to the denotational semantics). Now assume that the denotational semantics of a program P yields a defined integer value for a given initial store and a specified output location. By the Mezei–Wright theorem and the same argument as above, this value is already the image under h_{Algol} (applied to the initial store and the specified output location) of a finite approximation of the *program-tree* $T(P_\lambda)$ which can be derived by a finite number of applications of the copy-rule.

Appendix: Equivalence of $n-R$ and $n-\lambda$

In this appendix we prove the equivalence of both definitions of recursion on higher types. The translation from applicative to combinatoric representation by the mapping c has already been illustrated in Section 4. In order to simulate n-rational schemes by $n-\lambda$ schemes we replace the combinators *sub* and *pr* by the corresponding λ-terms. λ-terms over derived alphabets then occur in the intermediate stages.

We first deal with the simulation of n-rational schemes by level-n λ-schemes. By analogy to the mapping *yield* on trees, we call the translation mapping γ_λ.

Since λ-terms over $D(\Omega)$ have the set $D(I)$ as base types they only contain parameters in $D(Y) := Y \backslash \bigcup_{i \in I} Y_i$.

Definition A.1.

$$\gamma_\lambda : T_{D(\Omega),X,D(Y)} \to T_{\Omega,X,Y}$$

is the following $D^*(D(I))$-mapping:

$$\gamma_\lambda(x) := x, \qquad \gamma_\lambda(y_{j,\nu}) := y_{j,\nu},$$

$$\gamma_\lambda(f') := \lambda.f \quad \text{for } f \in \Omega,$$

$$\gamma_\lambda(pr_j^w) := \lambda.\lambda y_w.y_{j,w(j)},$$

$$\gamma_\lambda(abs_{(w,i)}) := \lambda y_{1,(e,i)}.\lambda y_w.y_{1(e,i)}(\ ),$$

$$\gamma_\lambda(sub_{(w,i)}^v) := \lambda y_\alpha.\lambda y_w.y_{1,\alpha(1)}(y_{2,\alpha(2)}(y_w), \ldots, y_{m,\alpha(m)}(y_w))$$

$$\text{with } \alpha = (v,i)(w,v(1)) \cdots (w,v(m-1)),\ m-1 = l(v) \geq 1,$$

$$\gamma_\lambda(t(t_1, \ldots, t_r)) := \gamma_\lambda(t)(\gamma_\lambda(t_1), \ldots, \gamma_\lambda(t_r)),$$

$$\gamma_\lambda(\lambda y_\alpha.t) := \lambda y_\alpha.\gamma_\lambda(t).$$

The following lemma shows that γ_λ preserves semantics.

Lemma A.2.

$$\forall \mathscr{A} \in \Delta\text{-}alg\ \Omega\ \forall \rho \in \mathscr{U}_{\mathscr{A}}\ [\![t, D(\mathscr{A})]\!]\rho = [\![\gamma_\lambda(t), \mathscr{A}]\!]\rho.$$

Proof. Induction on the structure of t.

- $t \in X \cup Y$: trivial.
- $t = f'$:

$$[\![f', D(\mathscr{A})]\!]\rho = \varphi_{D(A)}(f') = (\ ) \mapsto \varphi_A(f)$$
$$= (\ ) \mapsto [\![f, \mathscr{A}]\!]\rho = [\![\lambda.f, \mathscr{A}]\!]\rho.$$

- $t = pr_j^w$:

$$[\![pr_j^w, D(\mathscr{A})]\!]\rho = \varphi_{D(A)}(pr_j^w) = (\ ) \mapsto ((a_1, \ldots, a_n) \mapsto a_j)$$
$$= (\ ) \mapsto ((a_1, \ldots, a_n) \mapsto [\![y_{j,w(j)}, \mathscr{A}]\!]\rho[y_w/(a_1, \ldots, a_n)])$$
$$= (\ ) \mapsto [\![\lambda y_w.y_{j,w(j)}, \mathscr{A}]\!]\rho$$
$$= [\![\lambda.\lambda y_w.y_{j,w(j)}, \mathscr{A}]\!]\rho.$$

$-t = sub^{v}_{(w,i)}$:

$$[\![sub^{v}_{(w,i)}, D(\mathcal{A})]\!]\rho = \varphi_{D(A)}(sub^{v}_{(w,i)})$$
$$= (g_0, g_1, \ldots, g_m) \mapsto (a \mapsto g_0(g_1(a), \ldots, g_m(a)))$$
$$= (g_0, g_1, \ldots, g_m)$$
$$\mapsto (a \mapsto [\![y_{1,\alpha(1)}(y_{2,\alpha(2)}(y_w), \ldots, y_{m+1,\alpha(m+1)}(y_w)), \mathcal{A}]\!]\rho'')$$
$$\text{with } \rho'' = \rho[y_\alpha/(g_0, \ldots, g_m)][y_w/a]$$
$$= (g_0, \ldots, g_m)$$
$$\mapsto [\![\lambda y_w.y_{1,\alpha(1)}(y_{2,\alpha(2)}(y_w), \ldots, y_{m+1,\alpha(m+1)}(y_w)), \mathcal{A}]\!]\rho'$$
$$\text{with } \rho' = \rho[y_\alpha/(g_0, \ldots, g_m)]$$
$$= [\![\gamma_\lambda(sub^{v}_{(w,i)}), \mathcal{A}]\!]\rho.$$

$-t = abs_{(w,i)}$:

$$[\![abs_{(w,i)}, D(\mathcal{A})]\!]\rho = \varphi_{D(A)}(abs_{(w,i)})$$
$$= f \mapsto (a \mapsto f(\ ))$$
$$= f \mapsto (a \mapsto [\![y_{1,(e,i)}(\ ), \mathcal{A}]\!]\rho[y_{1,(e,i)}/f][y_w/a]$$
$$= f \mapsto [\![\lambda y_w.y_{1,(e,i)}(\ ), \mathcal{A}]\!]\rho[y_{1,(e,i)}/f]$$
$$= [\![\gamma_\lambda(abs_{(w,i)}), \mathcal{A}]\!]\rho.$$

$-t = t_0(t_1, \ldots, t_r)$: immediate by induction.
$-t = \lambda y_\alpha.t'$: immediate by induction. □

Now let S be a level-n λ-scheme over $D(\Omega)$. Since the level is calculated relative to the base type, $\gamma_\lambda \circ S$ is a level-$n+1$ λ-scheme over Ω which in accordance with the above lemma is equivalent to S up to ( )-application.

Corollary A.3.

$$\forall S \in n\text{-}\lambda(D(\Omega))^{(e,i)}\ \forall \mathcal{A} \in \Delta\text{-}alg\,\Omega\ \ [\![S, D(\mathcal{A})]\!](\ ) = [\![\gamma_\lambda \circ S, \mathcal{A}]\!].$$

Proof.

Lemma A.2 $\rightsquigarrow$ $S_{D(\mathcal{A})} = (\gamma_\lambda \circ S)_{\mathcal{A}}$ $\rightsquigarrow$ assertion. □

We now want to examine the conditions under which an inverse mapping to γ_λ exists.

In attempting to define such a function c inductively, one is immediately confronted with the problem of how to deal with abstraction. While dealing with an abstraction $\lambda y_\alpha.t$, c should intuitively memorize the type α so that constants of the

type ν are 'lifted' to constant functions of the type (α, ν) and/or parameters $y_{j,\alpha(j)}$ are replaced by the projection pr_j^α. Since, however, new parameters of the same *level* can be declared in t, such as $t = \cdots \lambda y_\beta . s \cdots$, occurrences of $y_{j,\alpha(j)}$ in s would be dealt with by c_β. If $\alpha(j) \neq \beta(r)$ for all $r \in [l(\beta)]$, then a translation by c would lead to a contradiction.

Example. Let $type(x_2) = ((i, s), (i, i))$, $e \in \Omega^{(e,i)}$:

$$t := \lambda y_{1,s} . x_2(\lambda y_{1,i} . y_{1,s})(e(\ )).$$

This situation results whenever 'global' parameters of the same *level* occur within an abstraction. The following definition characterizes λ-terms which make possible a translation by c.

Definition A.4. $t \in T_{\Omega,X,Y}$ is called *locally closed* iff for each subterm $\lambda y_\alpha . s$ of t it holds

$$y \in free(s) \wedge level(y) = level(\alpha) \curvearrowright y \in Y_\alpha.$$

It is possible to show that each $n-\lambda$ scheme is equivalent to a scheme whose right-hand sides are locally closed, c.f. [18]. Since the construction leads out of the class of homogeneously typed λ-terms, we merely illustrate the proof by using the following example.

Example. Let $type(x_1) = (s, i)$ and

$$x_1 = \lambda y_{1,s} . x_2(\lambda y_{1,i} . y_{1,s})(e(\ ))$$

be an equation of a *level*-2 scheme. In the first stage we replace the critical subterm $\lambda y_{1,i} . y_{1,s}$ by a procedure call whereby the 'global' parameter is passed as actual parameter.

Stage 1: Elimination of 'global' parameters.

$$x_1 = \lambda y_{1,s} . x_2(x_3(y_{1,s}))(e(\ )),$$

$$x_3 = \lambda y_{1,s} . \lambda y_{1,i} . y_{1,s}.$$

In general, the resulting scheme is no longer homogeneously typed. In particular

$$type(x_3) = (s, (i, s)).$$

However, by applications to ( ) and abstraction with respect to the empty parameter list it is always possible to construct an equivalent homogeneously typed λ-scheme [18].

Stage 2: Type homogenization.
Define

$$type(x_3') = ((e, s), (i, s)).$$

The equations for x_1 and x_3 are replaced by

$$x_1 = \lambda y_{1,s}.x_2(x_3'(\lambda.y_{1,s}))(e(\ )),$$

$$x_3' = \lambda y_{1,(e,s)}.\lambda y_{1,i}.y_{1,(e,s)}(\ ).$$

The correctness of this construction follows immediately from textual substitution and β-reduction.

We assume in the following without loss of generality that the right-hand sides of $n-\lambda$ schemes are locally closed.

The inverse mapping c is now defined by induction on the structure of terms. Therefore, if $w(j) \neq i$ (i.e. $y_{j,i}$ is a global parameter), we define $c_w(y_{j,i}) = \bot_{(w,i)}$. We then show that the mapping works correctly on locally closed terms.

Since λ-terms over $D(\Omega)$ have the set $D(I)$ as base types, we have to lift the type of *level*-0 variables.

Let $D(X) := \{x' \mid x \in X\}$, with

$$level(x) = 0, \rightsquigarrow type(x') := (e, type(x)$$

and

$$type(x') := type(x) \text{ otherwise.}$$

Definition A.5. Let $w \in I^*$.

$$comb_w : \mathrm{FT}_{\Omega,X,Y} \to \mathrm{FT}_{D(\Omega),D(X),D(Y)},$$

$$c_w(\bot_i) = \bot_{(w,i)},$$

$$i \in I \rightsquigarrow c_w(y_{j,i}) := \begin{cases} pr_j^w(\ ) & \text{if } w(j) = i, \\ \bot_{(w,i)} & \text{otherwise,} \end{cases}$$

$$\nu \in D^+(I) \rightsquigarrow c_w(y_{j,\nu}) := y_{j,\nu},$$

$$c_w(x) := \begin{cases} abs_{(w,i)}(x') & \text{if } w \neq e \wedge type(x) = i \in I, \\ x' & \text{otherwise,} \end{cases}$$

$$f \in \Omega \rightsquigarrow c_w(f) := f'(\ ),$$

$$t = t_0(t_1, \ldots, t_r) \wedge type(t_0) = (e, i) \in D(I)$$

$$\rightsquigarrow c_w(t_0(\ )) := abs_{(w,i)}(c_e(t_0)),$$

$$type(t_0) = (v, i) \in I^+ \times I$$

$$\rightsquigarrow c_w(t_0(t_1, \ldots, t_r)) := sub^v_{(w,i)}(c_v(t_0), c_w(t_1), \ldots, c_w(t_r)),$$

$$type(t_0) \notin D(I) \rightsquigarrow c_w(t_0(t_1, \ldots, t_r)) := c_w(t_0)(c_w(t_1), \ldots, c_w(t_r)),$$

$$v \in I^* \rightsquigarrow c_w(\lambda y_v.t) := c_v(t),$$

$$level(t) > 0 \rightsquigarrow c_w(\lambda y_\alpha.t) := \lambda y_\alpha.c_w(t).$$

An example of the behaviour of *comb* is given in Section 4.

Now let $\mathscr{A} \in \Delta\text{-}alg\ \Omega$ be an interpretation of Ω. We want to show that t and $c_e(t)$ define the same value over $\mathscr{A}$ up to abstraction to the empty parameter list. Since it is possible that the type of variables occurring in $c_e(t)$ has been lifted, we must evaluate $c_e(t)$ relative to a suitably modified environment:

for

$$\rho \in U_{\mathscr{A}} = X \cup Y \to A^*$$

let

$$\rho' \in U_{D(\mathscr{A})} = D(X) \cup D(Y) \to D(A)^*$$

be defined by

$$\rho'[\![x']\!] = \begin{cases} \rho[\![x]\!] & \text{if } level(x) > 0, \\ (\,) \to \rho[\![x]\!] & \text{otherwise.} \end{cases}$$

Lemma A.6. *Let $t \in T_{\Omega,X,Y}$ be locally closed and*

$$0\text{-}free(t) := free(t) \cap \{y_{j,i} \mid i \in I, j \in \omega\} \subseteq Y_w.$$

Then it holds

(1) $level(t) = 0 \curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho' = [\![\lambda y_w.t, \mathscr{A}]\!]\rho$,

(2) $level(t) > 0 \curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho' = [\![t, \mathscr{A}]\!]\rho$.

Proof.

$$\begin{aligned} t = y_{j,i} \wedge i \in I &\curvearrowright w(j) = i \curvearrowright c_w(t) = pr_j^w(\,) \\ &\curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho' \\ &= \varphi_{D(\mathscr{A})}(pr_j^w)(\,) \\ &= a \mapsto a_j \\ &= [\![\lambda y_w.y_{j,w(j)}, \mathscr{A}]\!]\rho; \end{aligned}$$

$$t = y_{j,\nu} \wedge \nu \notin I \curvearrowright c_w(t) = y_{j,\nu} \quad \text{trivial;}$$

$$\begin{aligned} t = x \wedge type(x) = i \in I \wedge w \neq e &\curvearrowright c_w(t) = abs_{(w,i)}(x') \\ &\curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho' \\ &= \varphi_{D(\mathscr{A})}(abs_{(w,i)})(\rho'[\![x']\!]) \\ &= a \mapsto \rho'[\![x']\!](\,) \\ &= a \mapsto \rho[\![x]\!] \\ &= [\![\lambda y_w.t, \mathscr{A}]\!]\rho; \end{aligned}$$

$t = x \wedge type(x) = i \in I \wedge w = e$

$\curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho'$

$= \rho'[\![x']\!]$

$= (\ ) \mapsto \rho[\![x]\!]$

$= [\![\lambda.t, \mathscr{A}]\!]\rho;$

$t = x \wedge level(x) > 0$ trivial;

$t = f \in \Omega \curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho'$

$= [\![f'(\ ), D(\mathscr{A})]\!]\rho'$

$= \varphi_{D(\mathscr{A})}(f')(\ )$

$= \varphi_{\mathscr{A}}(f)$

$= [\![f, \mathscr{A}]\!]\rho;$

$t = t_0(t_1, \ldots, t_r)$

$\wedge type(t_0) = (e, i) \in D(I)$

$\curvearrowright [\![c_w(t_0(\ )), D(\mathscr{A})]\!]\rho'$

$= [\![abs_{(w,i)}(c_e(t_0)), D(\mathscr{A})]\!]\rho'$

$= \varphi_{D(\mathscr{A})}(abs_{(w,i)})([\![c_e(t_0), D(\mathscr{A})]\!]\rho')$

$= a \mapsto [\![c_e(t_0), D(\mathscr{A})]\!]\rho'(\ )$

$= a \mapsto [\![t_0, \mathscr{A}]\!]\rho(\ )$

by induction hypothesis since t_0 locally closed $\wedge$

$level(t_0) > 0 \curvearrowright 0\text{-}free(t_0) = \emptyset$

$= a \mapsto [\![t_0(\ ), \mathscr{A}]\!]\rho$

$= [\![\lambda y_w.t, \mathscr{A}]\!]\rho$

$\wedge type(t_0) = (v, i) \in I^+ \times I$

$\curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho'$

$= [\![sub^v_{(w,i)}(c_v(t_0), c_w(t_1), \ldots, c_w(t_r)), D(\mathscr{A})]\!]\rho'$

$= \varphi_{D(\mathscr{A})}(sub^v_{(w,i)})([\![c_v(t_0), D(\mathscr{A})]\!]\rho', \ldots, [\![c_w(t_j), D(\mathscr{A})]\!]\rho', \ldots)$

$= a \mapsto [\![c_v(t_0), D(\mathscr{A})]\!]\rho'([\![c_w(t_1), D(\mathscr{A})]\!]\rho'(a), \ldots, [\![c_w(t_r), D(\mathscr{A})]\!]\rho'(a))$

$= a \mapsto [\![t_0, \mathscr{A}]\!]\rho([\![\lambda y_w.t_1, \mathscr{A}]\!]\rho(a), \ldots, [\![\lambda y_w.t_r, \mathscr{A}]\!]\rho(a))$

by induction hypothesis since t locally closed implies

$0\text{-}free(t_0) = \emptyset \wedge \forall j \in [r]\ 0\text{-}free(t_j) \subseteq Y_w$

$= a \mapsto [\![t_0, \mathscr{A}]\!]\rho([\![t_1, \mathscr{A}]\!]\rho[y_w/a], \ldots, [\![t_r, \mathscr{A}]\!]\rho[y_w/a])$

$= a \mapsto [\![t_0(t_1, \ldots, t_r), \mathscr{A}]\!]\rho[y_w/a]$

since $0\text{-}free(t_0) = \emptyset$

$= [\![\lambda y_w.t, \mathscr{A}]\!]\rho$

$\wedge\ level(t_0) > 1$ immediate by induction since t locally closed implies $\forall j \in \{0, \ldots, r\}\ 0\text{-}free(t_j) = \emptyset$;

$t = \lambda y_v.t' \wedge v \in I^* \curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho'$

$= [\![c_v(t'), D(\mathscr{A})]\!]\rho'$

$= [\![\lambda y_v.t', \mathscr{A}]\!]\rho$

by induction hypothesis since t locally closed implies $0\text{-}free(t') \subseteq Y_v$

$= [\![t, \mathscr{A}]\!]\rho$;

$t = \lambda y_\alpha.t' \wedge level(t') > 0$

$\curvearrowright [\![c_w(t), D(\mathscr{A})]\!]\rho'$

$= [\![\lambda y_\alpha.c_w(t'), D(\mathscr{A})]\!]\rho'$

$= f \mapsto [\![c_w(t'), D(\mathscr{A})]\!]\rho'[y_\alpha/f]$

$= f \mapsto [\![t', \mathscr{A}]\!]\rho[y_\alpha/f]$

by induction hypothesis since t locally closed $\curvearrowright$ $0\text{-}free(t') = \emptyset$

$= [\![t, \mathscr{A}]\!]\rho.$ □

Now let $S \in n+1-\lambda(\Omega)$, $var(S) = X$. Corresponding to S we define a level-n λ-scheme over $D(\Omega)$ by

$$c(S) : D(X) \to T_{D(\Omega), D(X), D(Y)},$$

$$c(S)(x') := c_e(S(x)).$$

Corollary A.7.

$\forall S \in n+1-\lambda(\Omega)\ \forall \mathscr{A} \in \Delta\text{-}alg\ \Omega\ \ [\![S, \mathscr{A}]\!] = [\![c(S), D(\mathscr{A})]\!](\ ).$

Proof. Let $S \in n+1-\lambda(\Omega)$, $var(S)=\{x_0,\ldots,x_N\}$, and $\alpha := type(x_0)\cdot\cdots\cdot type(x_N)$, $\alpha' := type(x'_0)\cdot\cdots\cdot type(x'_N)$.

We first show by induction on $k \in \omega$:

$$\mathrm{pr}_j\, c(S)^k_{D(\mathscr{A})}(\bot_{A^{\alpha'}}) = \begin{cases} \mathrm{pr}_j\, S^k_{\mathscr{A}}(\bot_{A^\alpha}) & \text{if } level(x_j)>0, \\ (\,) \mapsto \mathrm{pr}_j\, S^k_{\mathscr{A}}(\bot_{A^\alpha}) & \text{otherwise.} \end{cases} \qquad (*)$$

- $k=0$: trivial since $\bot_{A^{(\sigma,i)}} = (\,) \mapsto \bot_{A^i}$.
- $k \to k+1$:

$$\begin{aligned}
&\mathrm{pr}_j\, c(S)^{k+1}_{D(\mathscr{A})}(\bot_{A^{\alpha'}}) \\
&= [\![c(S)(x'_j), D(\mathscr{A})]\!]\tilde{\rho} \\
&\quad \text{with } \tilde{\rho}[\![x'_r]\!] = \mathrm{pr}_r\, c(S)^k_{D(\mathscr{A})}(\bot_{A^{\alpha'}}) \\
&= [\![c_e(S(x_j)), D(\mathscr{A})]\!]\rho' \\
&\quad \text{with } \rho[\![x_r]\!] = \mathrm{pr}_r\, S^k_{\mathscr{A}}(\bot_{A^\alpha}) \\
&\quad \text{since } level(x_r)=0 \rightsquigarrow \\
&\qquad \rho[\![x'_r]\!] \overset{\text{I.H.}}{=} (\,) \mapsto \mathrm{pr}_r\, S^k_{\mathscr{A}}(\bot_{A^\alpha}) = \rho'[\![x'_r]\!] \\
&\quad (level(x_r)>0 \text{ analogous by induction hypothesis}) \\
&= \begin{cases} [\![S(x_j), \mathscr{A}]\!]\rho & \text{if } level(x_j)>0, \\ [\![\lambda\,.\, S(x_j), \mathscr{A}]\!]\rho & \text{otherwise} \end{cases} \\
&\qquad\qquad \text{by Lemma A.6} \\
&= \begin{cases} \mathrm{pr}_j\, S^{k+1}_{\mathscr{A}}(\bot_{A^\alpha}) & \text{if } level(x_j)>0, \\ (\,) \mapsto \mathrm{pr}_j\, S^{k+1}_{\mathscr{A}}(\bot_{A^\alpha}) & \text{otherwise.} \end{cases}
\end{aligned}$$

We now show the assertion of the corollary.

Case 1:

$$\begin{aligned}
&level(x_0)=0 \\
&\rightsquigarrow [\![c(S), D(\mathscr{A})]\!](\,) \\
&= \mathrm{pr}_1 \bigsqcup_k c(S)^k_{D(\mathscr{A})}(\bot_{A^{\alpha'}})\downarrow_{D(\mathscr{A})}(\,) \\
&= \mathrm{pr}_1 \bigsqcup_k c(S)^k_{D(\mathscr{A})}(\bot_{A^{\alpha'}})(\,) \qquad \text{since } level(x'_0)=0 \\
&\overset{(*)}{=} \mathrm{pr}_1 \bigsqcup_k S^k(\bot_{A^\alpha}) \\
&= [\![S, \mathscr{A}]\!] \qquad \text{since } level(x_0)=0.
\end{aligned}$$

Case 2:

$$level(x_0) > 0$$

$$\rightsquigarrow [\![c(S), D(\mathscr{A})]\!]( )$$

$$= \mathrm{pr}_1 \bigsqcup_k c(S)^k_{D(\mathscr{A})}(\perp_{A^\alpha})\downarrow_{D(\mathscr{A})}( )$$

$$= \mathrm{pr}_1 \bigsqcup_k S^k_{\mathscr{A}}(\perp_{A^\alpha})\downarrow_{D(\mathscr{A})}( )$$

$$= \mathrm{pr}_1 \bigsqcup_k S^k_{\mathscr{A}}(\perp_{A^\alpha})\downarrow_A = [\![S, \mathscr{A}]\!]. \quad \square$$

The corollaries in this chapter show that we can translate λ-schemes in a stepwise fashion into an n-rational scheme and vice versa. This proves the equivalence of the two definitions of recursion on higher types.

Theorem A.8. $n - \lambda(\Omega) \sim n - R(\Omega)$.

Proof. '$\leqslant$' Let $S \in n - \lambda(\Omega)$, $\mathscr{A} \in \Delta\text{-}alg\ \Omega$.

$$[\![S, \mathscr{A}]\!] = [\![c^n(S), D^n(\mathscr{A})]\!]\underbrace{( )\cdots( )}_{n} \quad \text{Corollary A.7}$$

$$= [\![c^n(S), D^n(\mathscr{A})]\!]\downarrow_A$$

$$= [\![c^n(S), \mathscr{A}]\!].$$

'$\geqslant$' similarly using Corollary A.3. $\square$

Remark. Since the size of a translated term is bounded linearly by the size of the original term we have

$$size(c^n(S)) \leqslant p(size(S)),$$

$$size(y^n(S)) \leqslant p'(size(S)) \quad \text{for polynomials } p, p'.$$

PART II. LEVEL-N LANGUAGES

In part one of this paper, we developed a theory of level-n schemes. In particular level-n grammars were introduced to generate (in the schematic case) approximations of the infinite tree of a scheme. It turned out, that OI-derivations are sufficient to generate all trees in a level-n language.

Contrary to the context-free string case, the classes of languages generated by level-n grammars using rightmost (or innermost–outermost) and leftmost (or outermost–innermost) derivations are incomparable [30]. In this part we apply the results of part one to establish closure properties, decidability results, and various characterizations for level-n IO- and OI-languages and prove both hierarchies to be infinite. Sufficient conditions are derived to prove strictness of IO-like hierarchies. It is shown that level-n schemes form a strict hierarchy.

7. OI

In this section we study the families $n-\mathscr{L}_{\mathrm{OI}}$ of languages generated by level-n grammars (in OI- or, equivalently, unrestricted mode of derivation). It will turn out that level-n languages share almost all essential properties with the first three members in the OI-hierarchy and thus indeed form a natural, perhaps *the* natural extension of the (revised) Chomsky-hierarchy (cf. [70]) of regular, context-free and macro languages.

After presenting example languages at each level, we prove a fixed-point characterization, which gives the link between level-n languages and level-n schemes. We show that the family $n-\mathscr{L}_{\mathrm{OI}}(\Omega)$ coincides with the class of languages obtained by interpreting level-n schemes over Ω_+ in the power-set algebra of (tree-) languages over Ω. As an immediate consequence we inherit a Chomsky-normalform from level-n schemes. To demonstrate the usefulness of the fixed-point characterization, we apply the MW-theorem to obtain an easy algebraic proof of the decidability of the emptiness problem of level-n languages.

To substantiate the claim of naturalness of the OI-hierarchy, we show that it 'solves' the language-family producing diagram of the introduction. This involves two more characterization theorems: level-n OI string languages coincide with the path languages of level-n OI tree languages and the frontier languages of level-$(n-1)$ OI tree languages.

We close the section by investigating closure properties. An extension of the classical state-product construction for context-free (tree-) grammars (cf. [61]) shows that level-n OI languages are size-closed under intersection with regular sets. This answers a question raised by Schmidt [58]. Together with the decidability of emptiness this demonstrates recursiveness of level-n languages. Finally, closure of level-n trees under tree-homomorphisms implies substitution-closure of level-n string languages, hence for monadic Ω $n-\mathscr{L}_{\mathrm{OI}}(\Omega)$ forms a substitution closed AFL.

7.1. *Example languages*

It will be shown that level-n languages are in essence exponentially thinner than level-$(n-1)$ languages. In this subsection we define sample languages $L_n \in n-\mathscr{L}_{\mathrm{OI}}$ which are typical in this sense:

Let $\Omega = \{e, a, b\}$ with types (e, i), **1, 1.**
Define the language L_n (for $n > 2$) by

$$\{a^{2^{n-1}} b^{k+1} a^{2^{n-1}} b^{k} \cdots a^{2^{n-1}} b(e) \mid k \in \omega\}.$$

We first show by means of an example how L_3 can be generated by a level-3 grammar. In this section we identify $e(\,)$ and e and define $y_m := y_{1,m}$ for $m \in \omega$.

Example. Let $X = \{x_0, x_1, x_2, x_3, A, B\}$ with types **0, 2, 3, 3, 3, 2.** Let $G_3 \in 3\text{-}N\lambda(\Omega)$ be defined by

$$x_0 = x_3(x_1)(b)(e),$$
$$x_1\downarrow = y_1(y_1(y_0)), \qquad x_2\downarrow = y_2(y_2(y_1))\downarrow,$$
$$A\downarrow = y_2(a)(y_1(y_0)), \qquad B\downarrow = b(y_1(y_0)),$$
$$x_3\downarrow = x_3(x_2(y_2))(B(y_1))(A\downarrow), \qquad x_3\downarrow = A\downarrow.$$

In the following derivation of $a^{16}b^3a^4b^2a^2b(e)$ we view a textual substitution followed by a complete evaluation of all actual parameters as one derivation step.

$$\begin{aligned}
x_0 &\Rightarrow \overset{\downarrow}{x_3}(x_1)(b)(e)\\
&\Rightarrow x_3(x_2(x_1))(B(b))(\overset{\downarrow}{A}(x_1)(b)(e))\\
&\Rightarrow x_3(x_2(x_1))(B(b))(\overset{\downarrow}{x_1}(a)(b(e)))\\
&\Rightarrow \overset{\downarrow}{x_3}(x_2(x_1))(B(b))(\underbrace{a^2b(e)}) && w_1 = a^2b(e)\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(\overset{\downarrow}{A}(x_2(x_1))(B(b))(a^2b(e)))\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(x_2(x_1)(a)(\overset{\downarrow}{B}(b)(a^2b(e))))\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(\overset{\downarrow}{x_2}(x_1)(a)(b^2a^2b(e)))\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(\overset{\downarrow}{x_1}(x_1(a))(b^2a^2b(e)))\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(x_1(a)(\overset{\downarrow}{x_1}(a)(b^2a^2b(e))))\\
&\Rightarrow x_3(x_2^2(x_1))(B^2(b))(\overset{\downarrow}{x_1}(a)(a^2b^2a^2b(e)))\\
&\Rightarrow \overset{\downarrow}{x_3}(x_2^2(x_1))(B^2(b))(\underbrace{a^4b^2a^2b(e)}) && w_2 = a^4b^2a^2b(e)
\end{aligned}$$

$$\Rightarrow \overset{\downarrow}{A}(x_2^2(x_1))(B^2(b))(a^4b^2a^2b(e))$$

$$\Rightarrow x_2^2(x_1)(a)(\overset{\downarrow}{B}(B(b))(a^4b^2a^2b(e)))$$

$$\Rightarrow x_2^2(x_1)(a)(b(\overset{\downarrow}{B}(b)(a^4b^2a^2b(e))))$$

$$\Rightarrow \overset{\downarrow}{x_2}(x_2(x_1))(a)(b^3a^4b^2a^2b(e))$$

$$\Rightarrow \overset{\downarrow}{x_2}(x_1)(x_2(x_1)(a))(b^3a^4b^2a^2b(e))$$

$$\Rightarrow \overset{\downarrow}{x_1}(x_1(x_2(x_1)(a)))(b^3a^4b^2a^2b(e))$$

$$\Rightarrow x_1(x_2(x_1)(a))(\overset{\downarrow}{x_1}(x_2(x_1)(a))(b^3a^4b^2a^2b(e)))$$

$$\Rightarrow x_1(x_2(x_1)(a))(x_2(x_1)(a)(\overset{\downarrow}{x_2}(x_1)(a)(b^3a^4b^2a^2b(e))))$$

$$\Rightarrow x_1(x_2(x_1)(a))(x_2(x_1)(a)(\overset{\downarrow}{x_1}(x_1(a))(b^3a^4b^2a^2b(e))))$$

$$\Rightarrow x_1(x_2(x_1)(a))(x_2(x_1)(a)(x_1(a)(\overset{\downarrow}{x_1}(a)(b^3a^4b^2a^2b(e)))))$$

$$\Rightarrow x_1(x_2(x_1)(a))(x_2(x_1)(a)(\overset{\downarrow}{x_1}(a)(a^2b^3a^4b^2a^2b(e))))$$

$$\Rightarrow x_1(x_2(x_1)(a))(\overset{\downarrow}{x_2}(x_1)(a)(a^4b^3a^4b^2a^2b(e)))$$

$$\Rightarrow x_1(x_2(x_1)(a))(\overset{\downarrow}{x_1}(x_1(a))(a^4b^3a^4b^2a^2b(e)))$$

$$\Rightarrow x_1(x_2(x_1)(a))(x_1(a)(\overset{\downarrow}{x_1}(a)(a^4b^3a^4b^2a^2b(e))))$$

$$\Rightarrow x_1(x_2(x_1)(a))(\overset{\downarrow}{x_1}(a)(a^6b^3a^4b^2a^2b(e)))$$

$$\Rightarrow \overset{\downarrow}{x_1}(x_2(x_1)(a))(a^8b^3a^4b^2a^2b(e))$$

$$\Rightarrow \overset{\downarrow}{x_2}(x_1)(a)(\overset{\downarrow}{x_2}(x_1)(a)(a^8b^3a^4b^2a^2b(e))))$$

$$\overset{2}{\Rightarrow} x_1(a)(\overset{\downarrow}{x_1}(a))(x_1(a)(\overset{\downarrow}{x_1}(a)(a^8b^3a^4b^2a^2b(e))))$$

$$\overset{2}{\Rightarrow} \overset{\downarrow}{x_1}(a)(a^2)(\overset{\downarrow}{x_1}(a)(a^{10}b^3a^4b^2a^2b(e))))$$

$$\overset{2}{\Rightarrow} a^{16}b^3a^4b^2a^2b(e) = w_3.$$

Obviously, x_1 and x_2 define copy-functions of type **2** and **3** respectively. The characteristic exponential growth is due to the possibility of defining an arbitrary

number of iterations of copy functions of the highest type in each case and successively applying them to copy-functions of a lower type.

We now show that the obvious generalizations of G_3 generate the languages L_n.

Lemma 7.1. $\forall n \geq 3 \quad L_n \in n - \mathcal{L}_{OI}(\Omega)$.

Proof. Let $n \geq 3$. Define $G_n \in n - N\lambda(\Omega)$ by

(1) $var(G_n) = \{x_0, \ldots, x_n, A, B\}$

with

$$type(x_0) = \mathbf{0}, \qquad \forall j \in [n-1] \; type(x_j) = j+\mathbf{1},$$

$$type(A) = type(x_n) = \mathbf{n}, \qquad type(B) = \mathbf{2}.$$

(2) $x_0 = x_n(x_{n-2}) \cdots (x_1)(b)(e)$,

$x_j\downarrow = y_j(y_j(y_{j-1}))\downarrow$ for $j \in [n-1]$,

$A\downarrow = y_{n-1}(y_{n-2}) \cdots (y_2)(a)(y_1(y_0))$, $\quad B\downarrow = b(y_1(y_0))$,

$x_n\downarrow = x_n(x_{n-1}(y_{n-1}))(y_{n-2}) \cdots (y_2)(B(y_1))(A\downarrow)$,

$x_n\downarrow = A\downarrow$.

We now show that the copy-functions have the desired characteristics.

Let $m \in [n-1]$, $k \in \omega$. By induction on k it follows immediately

$$(x_m(\sigma_m))^k(\sigma_{m-1}) \cdots (\sigma_0) \overset{*}{\Rightarrow} \sigma_m^{2k}(\sigma_{m-1}) \cdots (\sigma_0)$$

Assertion 1.

$$x_m^k(\sigma_m)(\sigma_{m-1}) \cdots (\sigma_0) \overset{*}{\Rightarrow} \sigma_m^{2^k}(\sigma_{m-1}) \cdots (\sigma_0).$$

– $k = 0$: trivial.

– $k \to k+1$:

$$\begin{aligned} & x_m^{k+1}(\sigma_m)(\sigma_{m-1}) \cdots (\sigma_0) \\ & \quad = x_m^k(x_m(\sigma_m))(\sigma_{m-1}) \cdots (\sigma_0) \\ & \quad \overset{*}{\Rightarrow} (x_m(\sigma_m))^{2^k}(\sigma_{m-1}) \cdots (\sigma_0) \quad \text{induction hypothesis} \\ & \quad \overset{*}{\Rightarrow} \sigma_m^{2^{k+1}}(\sigma_{m-1}) \cdots (\sigma_0). \end{aligned}$$

Thus it follows:

Assertion 2.

$$x_m^k(x_{m-1}) \cdots (x_1)(a)(w) \overset{*}{\Rightarrow} a^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}(w).$$

(m)

Proof.

$-m=1$: $x_1^k(a)(w)=a^{2^k}(w)$ by Assertion 1.

$-m\to m+1$: $x_{m+1}^k(x_m)\cdots(x_1)(a)(w)\overset{*}{\Rightarrow}$

$\overset{*}{\Rightarrow} x_m^{2^k}(x_{m-1})\cdots(x_1)(a)(w)$ by Assertion 1

$\overset{*}{\Rightarrow} a^{\overset{m}{\overset{\diagup}{\underset{\diagdown}{2^{2^k}}}}}(w)$ by induction hypothesis

$=a^{\overset{m+1}{\overset{\diagup}{\underset{\diagdown}{2^k}}}}(w).$

Let $w_0 := e$, $w_{k+1} := a^{\overset{n-1}{\overset{\diagup}{\underset{\diagdown}{2^k}}}} b^{k+1}(w_k)$.

Assertion 3.

$$x_0\overset{*}{\Rightarrow} x_n(x_{n-1}^k(x_{n-2}))(x_{n-3})\cdots(x_1)(B^k(b))(w_k).$$

$-k=0$: trivial.

$-k\to k+1$:

$x_0\overset{*}{\Rightarrow} x_n(x_{n-1}^k(x_{n-2}))(x_{n-3})\cdots(x_1)(B^k(b))(w_k)$ I.H.

$\Rightarrow x_n(x_{n-1}^{k+1}(x_{n-2}))(x_{n-3})\cdots(x_1)(B^{k+1}(b))$

$\underbrace{(A(x_{n-1}^k(x_{n-2}))(x_{n-3})\cdots(x_1)(B^k(b))(w_k))}_{=:t_k}$

It therefore suffices to show $t_k\overset{*}{\Rightarrow} w_{k+1}$.

This follows immediately from Assertion 2:

$t_k\Rightarrow x_{n-1}^k(x_{n-2})(x_{n-3})\cdots(x_1)(a)(B^k(b)(w_k))$

$\overset{k}{\Rightarrow} x_{n-1}^k(x_{n-2})(x_{n-3})\cdots(x_1)(a)(b^{k+1}(w_k))$

$\overset{*}{\Rightarrow} a^{\overset{n-1}{\overset{\diagup}{\underset{\diagdown}{2^k}}}} b^{k+1}(w_k)=w_{k+1}.$

From Assertion 3 it follows immediately $L_n\subseteq L(G_n)$.

It should be obvious that these are also the only derivable words. □

7.2. Fixed-point characterization

Let G be a level-n grammar with terminals Ω. Clearly there is only one canonical way of associating a functional G_{PT_Ω} with G, such that (the first component of) its least fixed-point yields the language generated by G: define $G_+ \in n-\lambda(\Omega_+)$ by viewing the nondeterministic choice in the productions of a nonterminal x as an operation symbol $+$ (cf. Subsection 1.1). Then, by the Mezei–Wright theorem and the operational characterization of the infinite tree of G_+,

$$[\![G_+, (\mathscr{P}\mathscr{T}_\Omega)_+]\!] = \bigcup set(L(G_{+\perp}));$$

hence it suffices to prove (by a straightforward induction on the length of a derivation) that the $\bigcup \circ set$-image of the schematic language generated by G_+ coincides with the language generated by G. Thus the fixed-point characterization is essentially a corollary to the completeness of the operational semantics.

On the other hand, each level-n scheme over Ω_+ can be viewed as a level-n grammar by associating with a nonterminal x the *set*-image of the right-hand side of x in G_+. This proves

Theorem 7.2. $n-\mathscr{L}_{\mathrm{OI}}(\Omega) = \{[\![S, \mathscr{P}\mathscr{T}_\Omega]\!] \mid S \in n-\lambda(\Omega_+)\}$.

Proof. '$\supseteq$' Let $S \in n-\lambda(\Omega_+)$. Let $+$ occur without loss of generality (see Corollary 4.12) only in the rule $P\downarrow = +\downarrow$.

Then $set \circ S \in n-N\lambda(\Omega)$ is identical to S excluding the rules $P\downarrow = y_{1,i}|y_{2,i}$.

Moreover we have

$$\begin{aligned} [\![S, \mathscr{P}\mathscr{T}_\Omega]\!] &= set(T(S)) && \text{Corollary 4.11} \\ &= set(\bigsqcup L(S_\perp)) && \text{Theorem 6.7} \\ &= \bigcup set(L(S_\perp)). \end{aligned}$$

Because of Corollary 6.6 it therefore suffices to show

$$\bigcup set(L_{\mathrm{OI}}(S_\perp)) = L_{\mathrm{OI}}(set \circ S). \qquad (*)$$

Assertion 1.

$$x_0\downarrow \underset{\mathrm{OI},S_\perp}{\overset{k}{\Longrightarrow}} t \rightsquigarrow \forall s \in set(t) \qquad x_0\downarrow \underset{\mathrm{OI},set\circ S}{\overset{*}{\Longrightarrow}} s.$$

– $k = 0$: trivial.

– $k \to k+1$: Consider $x_0\downarrow \underset{\mathrm{OI}}{\overset{k}{\Longrightarrow}} t \underset{\mathrm{OI}}{\Longrightarrow} t'$ in $S_\perp$.

Let $(t_0, t_1, \ldots, t_k)$ be a linearization of t so that $t_1 = x(\sigma_m)\cdots(\sigma_0)$ is replaced.

Case 1: $t' = t_0[\perp, t_2, \ldots, t_k]$.

$s' \in set(t') = set(t_0) \leftarrow (\emptyset, \{t_2\}, \ldots, \{t_k\})$

$\rightsquigarrow \exists s_0 \in set(t_0)$ { without occurrences of $y_{1,i}$ such that $s' = s_0[t_1, t_2, \ldots, t_k]$

$\rightsquigarrow s' \in set(t) \rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} s'$ in $set \circ S$ by induction hypothesis.

Case 2: $t' = t_0[+(\sigma, \sigma'), t_2, \ldots, t_k]$ with $(\sigma, \sigma') = \sigma_0$, $m = 0$.

Linearity of $t_0 \wedge s' \in set(t') = set(t_0) \leftarrow (\{\sigma, \sigma'\}, \{t_2\}, \ldots, \{t_k\})$

$\rightsquigarrow \exists s_0 \in set(t_0)\ s' = s_0[\sigma, t_2, \ldots, t_k] \vee s' = s_0[\sigma', t_2, \ldots, t_k]$

$\rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} t_0[P(\sigma, \sigma'), t_2, \ldots, t_k]$ in $set \circ S$ by I.H.

$\underset{\mathrm{OI}}{\Longrightarrow} s'$ since $P\downarrow = y_{1,i} | y_{2,i}$ in $set \circ S$.

Case 3: $x \neq P$, no $\perp$-rule.

Immediate by induction hypothesis since $set \circ S(x) = \{S(x)\}$ is the only x-rule.

Assertion 2.

$$x_0\downarrow \overset{k}{\underset{\mathrm{OI}, set \circ S}{\Longrightarrow}} s \rightsquigarrow \exists t\, x_0\downarrow \overset{*}{\underset{\mathrm{OI}, S_\perp}{\Longrightarrow}} t \wedge s \in set(t).$$

- $k = 0$: trivial.

- $k \to k+1$: Consider $x_0\downarrow \overset{k}{\underset{\mathrm{OI}}{\Longrightarrow}} s \underset{\mathrm{OI}}{\Longrightarrow} s'$ in $set \circ S$.

Let $(s_0, s_1, \ldots, s_k)$ be a linearization of s, so that $s_1 = x(\sigma_m) \ldots (\sigma_0)$ is replaced.

Case 1: $x = P$, $m = 0$, $\sigma_0 = (\sigma, \sigma')$, $s' = s_0[\sigma, s_2, \ldots, s_k]$ (without loss of generality).

$\rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} t$ by I.H. for a $t = t_0[s_1, \ldots, s_k]$ with $s_0 \in set(t_0)$

$\rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI}, S_\perp}{\Longrightarrow}} t' := t_0[+(\sigma, \sigma'), s_2, \ldots, s_k] \wedge s' \in set(t')$.

Case 2: $x \neq P$: Immediate by induction hypothesis.

This allows to proof (*):

'$\subseteq$'

$$s \in \bigcup set(L_{\mathrm{OI}}(S_\perp)) \rightsquigarrow \exists t \in \mathrm{FT}_{\Omega,\perp}\, x_0\downarrow \overset{*}{\underset{\mathrm{OI}, S_\perp}{\Longrightarrow}} t \wedge s \in set(t)$$

$$\rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI}, set \circ S}{\Longrightarrow}} s \quad \text{by Assertion 1.}$$

'$\supseteq$'

$$s \in L_{\mathrm{OI}}(set \circ S) \rightsquigarrow x_0\downarrow \overset{*}{\underset{\mathrm{OI},set\circ S}{\Longrightarrow}} s \in T_\Omega$$

$$\rightsquigarrow \exists t \in \mathrm{FT}_{\Omega_+,X}\, x_0\downarrow \overset{*}{\underset{\mathrm{OI},S}{\Longrightarrow}} t \wedge s \in set(t) \quad \text{by Assertion 2.}$$

Let $(t_0, t_1, \ldots, t_k)$ be a linearization of t.

Case 1: $k = 0 \rightsquigarrow t \in \mathrm{FT}_{\Omega_+} \rightsquigarrow t \in L_{\mathrm{OI}}(S_\perp)$.

Case 2: $k > 0$.

Since

$$s \in T_\Omega \rightsquigarrow s \in set(t_0) \rightsquigarrow s \in set(t_0[\perp, \ldots, \perp])$$

$$\wedge\, t \overset{*}{\underset{\mathrm{OI},S_\perp}{\Longrightarrow}} t_0[\perp, \ldots, \perp].$$

end of proof of (*).

Thus '$\supseteq$' is proven.

'$\subseteq$' Let $G \in n\text{-}N\lambda(\Omega)$. Associate with G an equivalent grammar G' with $var(G') := var(G) \dot\cup \{P\}$ by

$$x\downarrow = t_1, \ldots, x\downarrow = t_n \text{ all } x\text{-productions in } G$$

$$\rightsquigarrow m = 1 \quad x\downarrow = t_1 \text{ in } G'$$

$$m > 1 \quad x\downarrow = P(t_1, \ldots, P(t_{m-1}, t_m)\ldots) \text{ in } G'$$

and

$$P\downarrow = y_{1,i},\ P\downarrow = y_{2,i} \text{ in } G'$$

$\rightsquigarrow$ G' is deterministic up to the P-productions.

Now let $G'_+ \in n\text{-}\lambda(\Omega_+)$ denote the deterministic variant of G' with $P\downarrow = +(y_{1,i}, y_{2,i})$. Then since $set \circ G'_+ = G'$ it holds

$$\begin{aligned} L(G) &= L(G') = L(set \circ G'_+) \\ &= \bigcup set(L(G'_{+\perp})) && (*) \\ &= set(\bigsqcup L(G'_{+\perp})) \\ &= set(T(G'_+)) && \text{Theorem 6.7} \\ &= [\![G'_+, \mathcal{PT}_\Omega]\!] && \text{Corollary 4.11.} \quad \square \end{aligned}$$

As a first application of the fixed-point characterization we state the nondeterministic variant of the normalform theorem.

Corollary 7.3 (Chomsky–Normalform). *For each grammar $G \in n\text{-}N\lambda(\Omega)$ one can effectively construct an equivalent grammar $G' \in n\text{-}N\lambda(\Omega)$ such that*

(1) $size(G) \leq p(size(G'))$ *for a polynomial* p,

(2) G *is deterministic up to the productions*

$$P(y_{1,i}, y_{2,i}) = y_{1,i} | y_{2,i} \quad \text{for } P \in var(G)$$

(3) *If* $x \in var(G) \backslash \{P\}$ *and* $type(x) = (\alpha_m, \ldots, (\alpha_0, i) \ldots)$, *then*

$$x\downarrow = f y_{\alpha_0} \quad \text{for } f \in \Omega, \quad \alpha_j = e \text{ for } j \geq 1$$

or

$$= x'(y_{\alpha_m}) \cdots (y_{\alpha_0}) \quad \text{for } x' \in var(G)$$

or

$$= y_{i, \pi_p(j)}(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0}) \quad \text{with } \alpha_p(j) = (\alpha_{p-1}, \ldots, (\alpha_0, i) \ldots)$$

or

$$= x_1(x_2(y_{\alpha_p}), \ldots, x_r(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0}) \quad \text{with } x_y \in var(G)$$

or

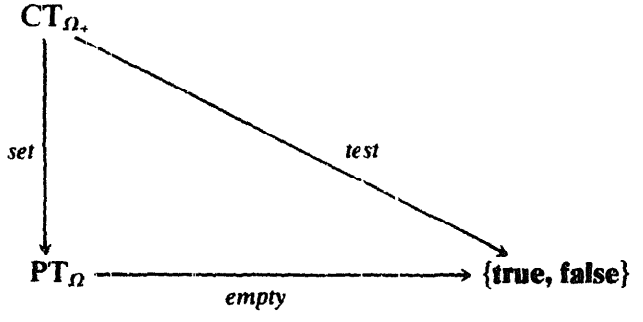
$$= y_{1,\alpha_p(1)}(y_{2,\alpha_p(2)}(y_{\alpha_{p-1}}), \ldots,$$
$$y_{k,\alpha_p(k)}(y_{\alpha_{p-1}}))(y_{\alpha_{p-2}}) \cdots (y_{\alpha_0}) \quad \text{with } p \leq m \wedge k = l(\alpha_p).$$

In particular, $1\text{-}\mathcal{L}_{OI}(\Omega)$ coincides with the class $\mathcal{CF}(\Omega)$ of context-free tree languages over Ω.

7.3. Decidability of emptiness

In this subsection, we shall first give a simple algebraic proof of decidability of the emptiness problem for level-n OI-tree languages and then analyse the algorithm to derive upper bounds for the depth of a tree in a nonempty level-n language.

Consider some OI-language $L \in n\text{-}\mathcal{L}_{OI}(\Omega)$. By Theorem 7.2 we can represent L as the semantics of a level-n scheme $S \in n\text{-}\lambda(\Omega_+)$ over the domain of tree languages over Ω. In order to decide whether $[\![S, (\mathcal{PT}_\Omega)_+]\!]$ is empty, it suffices by the MW-theorem to analyse the infinite tree $T(S)$. Thus we want to define a predicate *test*, such that



commutes. Here, *empty* denotes the predicate which sends L to **true** iff $L = \emptyset$.

To define *test*, we shall give {**true**, **false**} an Ω_+ structure.

Definition 7.4. Let $\mathcal{B} = (B, \alpha)$ denote the Ω_+-algebra with carrier $\{true, false\}$ and assignment function given by:

$$a \mapsto (( ) \mapsto \mathbf{false}) \quad \text{for } a \in \Omega^{(e,i)},$$

$$f \mapsto ((b_1, \ldots, b_n) \mapsto b_1 \vee b_2 \vee \cdots \vee b_n) \quad \text{for } f \in \Omega^{(w,i)},$$

$$+ \mapsto ((b_1, b_2) \mapsto b_1 \wedge b_2).$$

Lemma 7.5. *Setting true $\leqslant$ false makes $\mathcal{B}$ into a Δ-continuous Ω_+-algebra with $\bigsqcup$-complete carrier.*

Proof. For flat cpo's it suffices to prove the operations monotone, and this follows directly by checking the truth tables. Note that the join operation is given by

$$\bigsqcup M = \mathbf{false} \Leftrightarrow \mathbf{false} \in M \quad \text{for } M \subseteq B. \quad \square$$

Lemma 7.6. *empty: $(\mathcal{P}\mathcal{T}_\Omega)_+ \to \mathcal{B}$ is a $\bigsqcup$-continuous Ω_+-homomorphism.*

Proof. (i) *empty* is an Ω_+-homomorphism:

$$empty(\{a\}) = \mathbf{false} = \alpha(a)( ) \quad \text{for } a \in \Omega^{(e,i)},$$

$$empty(f(L_1, \ldots, L_n)) = empty(\{f(t_1, \ldots, t_n) \mid t_j \in L_j\})$$

$$= \begin{cases} \mathbf{true} & \text{iff } \exists j\, L_j = \emptyset \\ \mathbf{false} & \text{otherwise} \end{cases}$$

$$= empty(L_1) \vee \cdots \vee empty(L_n)$$
$$= \alpha(f)(empty(L_1), \ldots, empty(L_n)),$$

$$empty(L_1 \cup L_2) = \begin{cases} \mathbf{true} & \text{iff } L_1 = L_2 = \emptyset \\ \mathbf{false} & \text{otherwise} \end{cases}$$

$$= empty(L_1) \wedge empty(L_2)$$

$$= \alpha(+)(empty(L_1), empty(L_2)).$$

(ii) *empty* is strict by definition.

Now let $\mathcal{L} \subseteq \mathrm{PT}_\Omega$, $\mathcal{L} \neq \emptyset$

$$\Leftrightarrow (empty(\bigcup \mathcal{L}) = \mathbf{true} \Leftrightarrow \forall L \in \mathcal{L}\, empty(L) = \mathbf{true})$$

$$\Leftrightarrow \bigsqcup \{empty(L) \mid L \in \mathcal{L}\}$$

$$= \begin{cases} \mathbf{false} & \text{iff } \exists L \in \mathcal{L}\, L \neq \emptyset \\ \mathbf{true} & \text{otherwise} \end{cases}$$

$$= empty(\bigcup \mathcal{L}). \qquad \square$$

Now let *test* by initiality of $\mathscr{CT}_{\Omega_+}$ denote the unique Δ-continuous Ω_+-homomorphism, then we have

Corollary 7.7. *test* $=$ *empty* $\circ$ *set.*

Thus an infinite tree $t \in \mathrm{CT}_{\Omega_+}$ generates the empty language iff *test*(t) is true. The decidability of the emptiness problem thus follows from the Mezei–Wright-like theorem and the finiteness of B.

Theorem 7.8. *The emptiness problem for level-n OI-tree languages is decidable.*

Proof. For the following application it is convenient to represent L as the solution $L(S) := [\![S, \mathscr{PT}_\Omega]\!]$ of an n-rational scheme $S \in n - R(\Omega_+)$.

$$L = \emptyset$$

$$\curvearrowleft \; \mathit{empty}(L(S)) = \mathbf{true}$$

$$\curvearrowleft \; \mathit{empty}(\mathit{set}(\mathit{yield}^{(n)}(T(S)))) = \mathbf{true} \qquad \text{Theorem 4.6}$$

$$\curvearrowleft \; \mathit{test}(\mathit{yield}^{(n)}(T(S))) = \mathbf{true} \qquad \text{Corollary 7.7}$$

$$\curvearrowleft \; [\![S, \mathscr{B}]\!] = \mathbf{true} \qquad \text{Theorem 4.6}$$

$$\curvearrowleft \; [\![S, D^n(\mathscr{B})]\!]\underbrace{(\,)\cdots(\,)}_{n} = \mathbf{true} \qquad \text{Definition 4.2}$$

$$\curvearrowleft \; \mathrm{pr}_1\Big(\bigsqcup_{k\in\omega} S^k_{D^n(\mathscr{B})}(\perp_{\alpha(1)}, \ldots, \perp_{\sigma(N+1)})\Big)\underbrace{(\,)\cdots(\,)}_{n} = \mathbf{true} \qquad \begin{matrix}\text{Definition 2.2}\\ \text{Theorem 1.1}\end{matrix}$$

where α and $S_{D^n(\mathscr{B})}: D^n(B)^\alpha \to D^n(B)^\alpha$ are determined by S and $N+1 = l(\alpha) = |\mathit{var}(S)|$

$$\curvearrowleft \; \mathrm{pr}_1(S^{\bar{k}}_{D^n(\mathscr{B})}(\perp_{\alpha(1)}, \ldots, \perp_{\alpha(N-1)}))\underbrace{(\,)\cdots(\,)}_{n} = \mathrm{true}$$

with $\bar{k} := D^n(B)^\alpha$ since for iterations $k \mapsto f^k(\perp)$ we have $f^k(\perp) = f^{k+1}(\perp) \curvearrowright \forall m \geq k\, f^k(\perp) = f^m(\perp)$. □

By analysing the above proof we now derive an upper bound for the depth of a tree in an nonempty level-n language. To this end, we first estimate the number of iterations needed to compute the test semantics of a level-n scheme.

Lemma 7.9. *For* $\tau \in D^n(I)$, *define inductively*

$$\mathit{rank}(i) := 0,$$

$$\mathit{rank}(\alpha, \nu) = \max\{l(\alpha), \mathit{rank}(\nu), \mathit{rank}(\alpha(j)) \mid j \in [l(\alpha)]\}.$$

Then it holds

$$|B^\tau| \leq \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_n \cdot rank(\tau)^n}}}}}}_{n}$$

for some $c_n \in \mathbb{R}_+$ not depending on τ.

Proof. We first show

$$\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n}\right)^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n}} \leq \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1 c_2}}}}}}_{n} \quad \text{for } c_1, c_2 \in \mathbb{R}_+ \text{ with } c_1 > 4. \qquad (*)$$

- $n = 0$: trivial.
- $n \to n+1$:

$$\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n+1}\right)^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n+1}} = \left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n} \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n+1}\right)^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n+1}}$$

$$= 2^{ld\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n} \cdot \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n+1}\right) \cdot \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n+1}} \quad \text{since } b^x = 2^{ld(b) \cdot x}$$

$$= 2^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n} \cdot \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n}} \quad \text{since } ld(2^x) = x$$

$$\leq 2^{\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n}\right)^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n}}}$$

since for $1 \leq a \leq b$ $a \cdot 2^b \leq a^b \rightsquigarrow 2^b \leq a^{b-1} \rightsquigarrow b \leq ld(a^{b-1})$

and for $c_1 > 4$

$$\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n} \leq ld\left(\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n}\right)^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n}} - 1\right)$$

$$= ld\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n}\right) \cdot \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_2}}}}}}_{n} - ld\left(\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1}}}}}}_{n}\right)$$

$$\leq 2^{\underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1 c_2}}}}}}_{n}} = \underbrace{2^{\cdot^{\cdot^{\cdot^{2^{c_1 c_2}}}}}}_{n+1} \quad \text{by I.H.}$$

Thus (*) is proven.

The assertion now follows by induction on τ:

– $\tau = i$: define $c_0 = 2$.

– $\tau = (\alpha, \nu) \in D^{n+1}(I)$:

$$|B^{(\alpha,\nu)}| = |B^{\nu}|^{|B^{\alpha(1)}| \cdot \ldots \cdot |B^{\alpha(k)}|}$$

$$\leqslant \left(\underset{2}{\overset{n}{2^{\cdot^{\cdot^{\cdot}}}}} c_n \cdot rank(\nu)^n \right)^{|B^{\alpha(1)}| \cdot \ldots \cdot |B^{\alpha(k)}|} \quad \text{by I.H.}$$

$$\leqslant \left(\underset{2}{\overset{n}{2^{\cdot^{\cdot^{\cdot}}}}} c_n \cdot rank(\nu)^n \right)^{\underset{2}{\overset{n}{2^{\cdot^{\cdot^{\cdot}}}}} c_n \cdot rank(\alpha,\nu)^n + k}$$

by I.H. and trivial arithmetic for $n > 0$ (for $n = 0$ the expression reduces to $c_0^{k \cdot c_0} = 2^{ld(c_0) \cdot k \cdot c_0}$)

$$\leqslant \underset{2}{\overset{n}{2^{\cdot^{\cdot^{\cdot}}}}} (c_n \cdot rank(\nu)^n) c_n \cdot rank(\alpha,\nu)^n + k \quad \text{by (*)}$$

$$\leqslant \underset{2}{\overset{n+1}{2^{\cdot^{\cdot^{\cdot}}}}} ld(c_n \cdot rank(\nu)^n) \cdot (c_n \cdot rank(\alpha,\nu)^n + k)$$

$$\leqslant \underset{2}{\overset{n+1}{2^{\cdot^{\cdot^{\cdot}}}}} (ld(c_n) + n \cdot ld(rank(\nu))) \cdot (c_n \cdot rank(\alpha,\nu)^n + k)$$

$$\leqslant \underset{2}{\overset{n+1}{2^{\cdot^{\cdot^{\cdot}}}}} c_{n+1} \cdot rank(\alpha,\nu)^{n+1} \quad \text{for a suitable } c_{n+1} \in \mathbb{R}. \quad \square$$

Together with the proof of the Kleene characterization, this implies that at most $EXP^n(POL)$ steps of the Kleene sequence of an n-rational scheme are needed to guarantee that the image under $set \circ yield^{(n)}$ is not empty.

We now prove that in translating by *yield* the depth of a tree can increase at most exponentially.

Lemma 7.10. *Let Σ be a many-sorted alphabet, $t \in FT_{D(\Sigma)}$.*
Then it holds

$$depth(yield(t)) \leqslant 2^{depth(t)}.$$

Proof. Induction on t.

– $t = f'$:

$$depth(\gamma(f')) = depth(fy_\alpha) = 2 = 2^{depth(f')}.$$

– $t = \perp_{(\alpha,\nu)}$:

$$depth(\gamma(\perp_{(\alpha,\nu)})) = depth(\perp_\nu) = 1 = depth(\perp_{(\alpha,\nu)}).$$

– $t = pr_j^\alpha$:

$$depth(\gamma(pr_j^\alpha)) = depth(y_{i,a(i)}) = 1 = depth(pr_j^\alpha).$$

– $t = abs_{(\alpha,\nu)}t'$: direct by I.H. since $\gamma(abs_{(\alpha,\nu)}t') = \gamma(t')$.

– $t = sub^\gamma_{(\alpha,\nu)}t_0t_1 \cdots t_r$:

$$\begin{aligned}
&depth(\gamma(sub^\gamma_{(\alpha,\nu)}t_0t_1 \cdots t_r)\\
&\quad = depth(\gamma(t_0) \leftarrow (\gamma(t_1), \ldots, \gamma(t_r)))\\
&\quad \leq depth(\gamma(t_0)) + max\{depth(\gamma(t_j)) \mid j \in [r]\}\\
&\quad \leq 2^{depth(t_0)} + 2^{max\{depth(\gamma(t_j)) \mid j \in [r]\}} \quad \text{by I.H.}\\
&\quad \leq 2 \cdot 2^{max\{depth(\gamma(t_j)) \mid j \in \{0, \ldots, r\}\}}\\
&\quad \leq 2^{depth(t)}. \qquad \square
\end{aligned}$$

It is easy to see that this bound is optimal (consider $t_0 := f'$, $t_{n+1} := sub^{ii}_{(ii,i)}t_nt_n$).

Remark. For the typed λ-calculus, this lemma implies that the size of the normalform of a term t of type 0, whose highest subtype is n, can be restricted by

$$size(nf(t)) \leq \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{n} 2^{size(t)}$$

since $size(t) \cong size(comb^{(n)}(t))$ and the normalform of t coincides with $yield^{(n)}(comb^{(n)}(t))$.

By summarizing, we obtain a $2n$-exponential bound on the depth of a tree in a nonempty level-n language.

Theorem 7.11. *Let $L = [\![S, \mathcal{PT}_\Omega]\!]$ for a level-n scheme S over Ω_+.*

$$L \neq \emptyset \leftrightarrow \exists t \in L \qquad depth(t) \leq \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{2n} 2^{p(size(S))}$$

for a polynomial p.

Proof. Let (without loss of generality) $S \in n - R(\Omega_+)$, with $var(S) = X_\alpha$, $r = l(\alpha)$, $\bar{k} = |B^\alpha|$. From the proof of Theorem 7.8 it follows

$$\begin{aligned}
&L \neq \emptyset \rightsquigarrow pr_1(S^{\bar{k}}_{D^n(\mathfrak{B})}(\bot_\alpha))\downarrow = \textbf{false}\\
&\rightsquigarrow test(yield^{(n)}(pr_1(S^{\bar{k}}_{\mathfrak{B}_{D^n(\Omega_+)}}(\bot_\alpha))\downarrow)) = \textbf{false} && \text{Theorems 2.3, 4.6}\\
&\rightsquigarrow test(yield^{(n)}(K(S)(\bar{k}))) = \textbf{false} && \text{Theorem 2.5 or 5.9}\\
&\rightsquigarrow set(yield^{(n)}(K(S)(\bar{k}))) \neq \emptyset && \text{Corollary 7.7.}
\end{aligned}$$

$L \neq \emptyset$ therefore implies $\exists t \in set(yield^{(n)}(K(S)(\bar{k}))) \subseteq L$.

Let $depth(S) := max\{depth(S(x)) \mid x \in var(S)\}$, then it holds

$$depth(K(S)(\bar{k})) \leqslant \bar{k} \cdot depth(S),$$

and thus

$$depth(t) \leqslant depth(yield^{(n)}(K(S)(\bar{k})))$$

$$\leqslant 2^{\cdot^{\cdot^{\cdot^{2^{depth(K(S)(\bar{k}))}}}}} \quad (n \text{ twos})$$

Lemma 7.10

$$\leqslant 2^{\cdot^{\cdot^{\cdot^{2^{k \cdot depth(S)}}}}} \quad (n \text{ twos})$$

$$\leqslant 2^{\cdot^{\cdot^{\cdot^{2^{2^{c \cdot rank(S) + r \cdot depth(S)}}}}}} \quad (n \text{ twos})$$

Lemma 7.9

with $rank(S) = max\{rank(\alpha(j)) \mid j \in [r]\}$

$$\leqslant 2^{\cdot^{\cdot^{\cdot^{2^{p(size(S))}}}}} \quad (2n \text{ twos})$$

for a suitable polynomial p

since

$$size(S) \geqslant rank(S) \cdot |var(S)| \cdot depth(S). \quad \square$$

7.4. Paths and leaves

In the present subsection we shall prove that the OI-hierarchy solves the language-producing diagram of the introduction. Thus we have to characterize level-$n+1$ string languages as frontier-languages of level-n tree languages in a uniform way.

For one direction of the proof it is convenient to assume that a level-n string grammar only uses one parameter of the base type i. Here we shall prove the more general result that path languages of level-n tree languages can be generated by level-n grammars satisfying this requirement.

In the sequel we shall use the following property of OI-derivations in a grammar in Chomsky normalform.

Lemma 7.12. *Let N denote the det of applicative terms over nonterminals of level 0. If $G \in n - N\lambda(\Omega)$ is in Chomsky normalform, then*

$$x_0\downarrow \overset{*}{\underset{OI,G}{\Longrightarrow}} t \curvearrowright t \in T_\Omega(N).$$

The assertion follows immediately by induction on the length of the derivation and case-splitting according to the form of the right-hand side.

We shall now prove the path language characterization.

As in the context-free case (cf. [8]) we replace a nonterminal of 'arity' k (i.e. with type $(\ldots, (i^k, i)\ldots)$) by k nonterminals x^j, $j \in [k]$, whose right-hand sides will be the set of 'paths leading to $y_{j,i}$' of the right-hand sides of x.

In order to describe this notion formally we shall define a mapping br_j on applicative terms. In the treatment of higher type subterms we first allow all possible paths and in an actual derivation choose the correct one by means of projections. This implies a change of types.

Construction

(1) *Types*

(1.1) The arity of a type $\tau = (\alpha_m, \ldots, (i^k, i)\ldots)$ is $ar(\tau) = k$.

(1.2) Define $br: D^*(I) \to D^*(I)$ by

$$br(i) := i, \qquad br(i^k, i) := (i, i),$$

$$br(\alpha(1) \cdot \ldots \cdot \alpha(r), \nu) := (br(\alpha(1))^{ar(\alpha(1))} \cdot \ldots \cdot br(\alpha(r))^{ar(\alpha(r))}, br(\nu)).$$

(2) *Terms*

(2.1) $ar(t) := ar(type(t))$.

(2.2) For applicative terms $t \in T_{\Omega,X,Y}$ and $0 \leq j \leq ar(t)$ we define $br_j(t)$ by induction on t:

$$br_j(y_{i,i}) := y_{1,i},$$

$$br_j(y_{k,\nu}) := y^j_{k,br(\nu)} \qquad \text{if } level(\nu) > 0,$$

$$br_j(f) := f^j \in \Omega^{(i,i)}_{br} \qquad \text{if } f \in \Omega^{(i^k,i)} \wedge k > 0,$$

$$br_0(a) := e \in \Omega^{(e,i)}_{br} \qquad \text{if } a \in \Omega^{(e,i)},$$

$$br_j(x) := x^j \qquad \text{with } type(x^j) = br(type(x)),$$

$$br_0(t_0(\,)) := br_0(t_0)(\,) \text{ if } level(t_0) = 1,$$

$$br_j(t_0(t_1, \ldots, t_k)) = \bigcup_{r \in [k]} br_r(t_0)(br_j(t_r)) \quad \text{if } level(t_0) = 1,$$

$$br_i(t_0(t_1, \ldots, t_k)) = br_j(t_0)(br_{[ar(t_1)]}(t_1), \ldots, br_{[ar(t_k)]}(t_k)).$$

To simplify notation we identify

t and $\{t\}$,

$br_{[r]}(t)$ and $br_1(t), \ldots, br_r(t)$.

(3) *Grammars.* Let $G \in n - N\lambda(\Omega)$ be a grammar in Chomsky-normalform. We define $br(G) \in n - N\lambda(\Omega_{br})$ by

(3.1) Nonterminal symbols $X_{br} := \{x^j \mid x \in var(G), 0 \leq j \leq ar(x)\}$.

(3.2) Axiom $x_0^0\downarrow$.

(3.3) Productions

$$t \in rhs(x) \text{ in } G \rightsquigarrow \forall 0 \leq j \leq ar(x)\ \ br_j(t) \in rhs(x^j) \text{ in } br(G).$$

Before we prove the correctness of the construction, we first give a simple example.

Example. Let $\Omega = \{e: (e, i), f, g: (ii, i)\}$ and $G \in 2 - N\lambda(\Omega)$ be defined by

$$var(G) = \{x_0: i, x_1: (ii, i), x_2: ((ii, i), (ii, i))\},$$

$$x_0 = x_2(x_1)(e, e), \qquad x_1\downarrow = f(y_{ii}) \mid g(y_{ii}),$$

$$x_2\downarrow = y_{1,(ii,i)}(y_{1,(ii,i)}(y_{2,i}, y_{1,i}), y_{2,i})).$$

Clearly

$$L_{OI}(G) = \{f(f(e, e), e), f(g(e, e), e), g(f(e, e), e), g(g(e, e), e)\}.$$

Then

$$var(br(G)) = \{x_0^0: i, x_1^1, x_1^2: (i, i), x_2^1, x_2^2: ((i, i)(i, i), (i, i))\}$$

and

$$x_0^0 = x_2^1(x_1^1, x_1^2)(e) \mid x_2^2(x_1^1, x_1^2)(e),$$

$$x_1^1\downarrow = f^1(y_{1,i}) \mid g^1(y_{1,i}), \qquad x_1^2\downarrow = f^2(y_{1,i}) \mid g^2(y_{1,i}),$$

$$x_2^1\downarrow = y_{1,(i,i)}^1(y_{1,(i,i)}^2(y_{1,i})),$$

$$x_2^2\downarrow = y_{1,(i,i)}^1(y_{1,(i,i)}^1(y_{1,i})) \mid y_{1,(i,i)}^2(y_{1,i}).$$

An example of an OI-derivation in $br(G)$ is

$$x_0^0 \underset{OI}{\Longrightarrow} x_2^1(x_1^1, x_1^2)(e) \underset{C}{=} \ldots \underset{OI}{\Longrightarrow} f^1(x_1^2(e)) \underset{OI}{\Longrightarrow} f^1(g^2(e)).$$

Clearly $L_{OI}(br(G)) = br(L_{OI}(G))$

Theorem 7.13. *The path languages of $n - \mathcal{L}_{OI}$ coincide with the level-n OI string languages.*

Proof. One inclusion is trivial.

Now let G be a level-n grammar in Chomsky-normalform. Without loss of

generality (see [18]) we may assume

$$\forall A \in N \quad \exists t \in T_\Omega \; A \overset{*}{\underset{G}{\Longrightarrow}} t. \tag{*}$$

The proof of $br(L(G)) = L(br(G))$ is based on the following two assertions (here $br := br_0$).

Assertion 1.

$$N \ni A \overset{*}{\underset{\mathrm{OI},G}{\Longrightarrow}} t \;\frown\; \forall w \in br(t)\, \exists \gamma \in br(A)\; \gamma \overset{*}{\underset{\mathrm{OI},br(G)}{\Longrightarrow}} w.$$

Assertion 2.

$$\exists A \in N\, \gamma \in br(A) \wedge \gamma \overset{*}{\underset{\mathrm{OI},br(G)}{\Longrightarrow}} w \;\frown\; \exists t\, w \in br(t) \wedge A \overset{*}{\underset{\mathrm{OI},G}{\Longrightarrow}} t.$$

Both assertions are proved by induction on the length of the derivation and case-splitting by the form of the applied production.

Proof of Assertion 1. The base step is trivial.

Now let $A \overset{*}{\Rightarrow} t \Rightarrow t'$ be an OI-derivation in G and $(t_0, t_1, \ldots, t_r)$ a linearization of t such that $t' = t_0[t'_1, \ldots, t_r]$ with

$$t'_1 = s[y_{\alpha_m}/\sigma^m] \cdots [y_{\alpha_1}/\sigma^1][y_{\alpha_0}/(s_1, \ldots, s_k)]$$

where

$$t_1 = x(\sigma^m) \cdots (\sigma^1)(s_1, \ldots, s_k) \quad \text{and} \quad s \in rhs(x).$$

By induction hypothesis it suffices to consider the case

$$w' \in w_0[br(t'_1)] \quad \text{with } w_0 = br_1(t_0).$$

Case 1:

$$s = f(y_{1,i}, \ldots, y_{k,i}),\ m = 0$$

$$\frown\; \exists j \in [k]\, \exists v \in br(s_j)\; w' = w_0[f^j(v)]$$

and by construction

$$f^j(y_{1,i}) \in rhs(x^j).$$

As $w_0[x^j(v)] \in br(t)$ we have by I.H.

$$\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w_0[x^j(v)] \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w' \quad \text{in } br(G) \text{ for some } \gamma \in br(A).$$

Case 2:

$s = a(\ ),\ \alpha_0 = e$

$\curvearrowright\ w' = w_0[e(\ )]$ and by construction $e(\ ) \in rhs(x^0)$.

Since $w_0[x^0(\ )] \in br(t)$ we have by I.H.

$$\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w_0[x^0(\ )] \underset{\mathrm{OI}}{\Longrightarrow} w' \quad \text{in } br(G).$$

Case 3: $s = x'(y_{\alpha_m}) \cdots (y_{\alpha_0})$.

(3.1) $level(x) = 0$

$\curvearrowright\ w' = w_0[x'^0]$ and by construction $x'^0 \in rhs(x^0)$

$\curvearrowright$ since $w_0[x^0] \in br(t)$ $\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w_0[x^0] \underset{\mathrm{OI}}{\Longrightarrow} w'$.

(3.2) $level(x) > 0$

$\curvearrowright\ \exists j \in [k]\ \exists v \in br(s_j)\ w' = w_0[br_j(x'(\sigma^m) \cdots (\sigma^1))(v)]$.

Since

$$br(t) \ni w_0[br_j(x(\sigma^m) \cdots (\sigma^1))(v)] = w_0[x^i(br_{[ar(\sigma^m)]}(\sigma^m)) \cdots (br_{[ar(\sigma^1)]}(\sigma^1))(v)]$$

and

$$br_j(x'(y_{\alpha_m}) \cdots (y_{\alpha_1})(y_{\alpha_0})) = x'^j(br_{[ar(\alpha_m)]}(y_{\alpha_m})) \cdots (br_{[ar(\alpha_1)]}(y_{\alpha_1}))(y_{1,i}) \in rhs(x^i)$$

it follows that

$$\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w_0[br_j(x(\sigma^m) \cdots (\sigma^1))(v)] \underset{\mathrm{OI}}{\Longrightarrow} w_0[x'^j](br_{[ar(\alpha_m)]}(\sigma^m)) \cdots (br_{[ar(\alpha_1)]}(\sigma^1))(v)] = w'.$$

Case 4: $s = y_{r,\alpha_p(r)}(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0})$.

(4.1) $p = 0$

$\curvearrowright\ \exists v \in br(s_r)\ w' = w_0[v]$ and

$w_0[br_r(x(\sigma^m) \cdots (\sigma^1))(v)] = w_0[x^r(br_{[ar(\alpha_m)]}(\sigma^m)) \cdots (br_{[ar(\alpha_1)]}(\sigma^1))(v)] \in br(t)$

and by construction $y_{1,i} \in rhs(x^r)$

$\curvearrowright\ \gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w_0[br_r(x(\sigma^m) \cdots (\sigma^1))(v)] \underset{\mathrm{OI}}{\Longrightarrow} w_0[v]$ in $br(G)$ by I.H.

(4.2) $p>0, \quad \sigma^p=(\sigma_1,\ldots,\sigma_q), \quad \nu=\alpha_p(r)$

$\rightsquigarrow \exists j\in[k]\,\exists v\in br(s_j)\; w'=w_0[br_j(\sigma_r(\sigma^{p-1})\cdots(\sigma^1))(v)]$ and

$w_0[br_j(x(\sigma^m)\cdots(\sigma^1))(v)]=w_0[x^j(br_{[ar(\alpha_m)]}(\sigma^m))\cdots$

$(br_{[ar(\alpha_1)]}(\sigma^1))(v)]\in br(t)$

and by construction

$br_j(y_{r,\nu}(y_{\alpha_{p-1}})\cdots(y_{\alpha_0}))=y^j_{r,br(\nu)}(br_{[ar(\alpha_{p-1})]}(y_{\alpha_{p-1}}))\cdots$

$(br_{[ar(\alpha_1)]}(y_{\alpha_1}))(y_{1,i})\in rhs(x^j)$

$\rightsquigarrow \gamma \overset{*}{\underset{OI}{\Longrightarrow}} w_0[br_j(x(\sigma^m)\cdots(\sigma^1))(v)]$

$\overset{}{\underset{OI}{\Longrightarrow}} w_0[br_j(\sigma_r)(br_{[ar(\alpha_{p-1})]}(\sigma^{p-1}))\cdots$

$(br_{[ar(\alpha_1)]}(\sigma^1))(v)]=w'$

since

$br_{[ar(\alpha_p)]}(\sigma^p)=(\ldots,br_1(\sigma_r),\ldots,br_{ar(\nu)}(\sigma_r),\ldots)$

and

$\nu=(\alpha_{p-1},\ldots,(\alpha_0,i)\ldots), \quad j\leq ar(\nu).$

Case 5: $s=x_1(x_2(y_{\alpha_p}),\ldots,x_r(y_{\alpha_p}))(y_{\alpha_{p-1}})\cdots(y_{\alpha_0}).$

(5.1) $p=0$: similar to (6.1).

(5.2) $p>0$

$\rightsquigarrow \exists j\in[k]\,\exists v\in br(s_j)\; w'=w_0[v'(v)]$

with

$v'=br_j(x_1(x_2(\sigma^p),\ldots,x_r(\sigma^p))(\sigma^{p-1})\cdots(\sigma^1))$

$=x^j_1(br_{[ar(x_2)]}(x_2)(br_{[ar(\alpha_p)]}(\sigma^p)),\ldots,br_{[ar(x_r)]}(x_r)(br_{[ar(\alpha_p)]}(\sigma^p)))$

$(br_{[ar(\alpha_{p-1})]}(\sigma^{p-1}))\cdots(br_{[ar(\alpha_1)]}(\sigma^1)).$

Since

$w=w_0[x^j(br_{[ar(\alpha_m)]}(\sigma^m))\cdots(br_{[ar(\alpha_1)]}(\sigma^1))(v)]\in br(t)$

and by construction

$x^j_1(br_{[ar(x_2)]}(x_2)(br_{[ar(\alpha_p)]}(y_{\alpha_p})),\ldots,br_{[ar(x_r)]}(x_r)(br_{[ar(\alpha_p)]}(y_{\alpha_p})))$

$(br_{[ar(\alpha_{p-1})]}(y_{\alpha_{p-1}}))\cdots(br_{[ar(\alpha_1)]}(y_{\alpha_1}))(y_{1,i})\in rhs(x^j)$

it follows by I.H.

$$\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w \underset{\mathrm{OI}}{\Longrightarrow} w' \quad \text{in } br(G).$$

Case 6: $s = y_{1,\nu_1}(y_{2,\nu_2}(y_{\alpha_p}), \ldots, y_{r,\nu_r}(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0})$ with $\alpha_{p+1} = \nu_1 \cdots \nu_r$, $\sigma^{p+1} = (\sigma_1, \ldots, \sigma_r)$.

(6.1) $p = 0$

$$\rightsquigarrow \exists j \in [k]\, \exists v \in br(s_i)\, \exists q \in \{2, \ldots, r\}\; w' = w_0[v_1(v_2(v))]$$

with

$$v_1 = br_q(\sigma_1) \wedge v_2 = br_j(\sigma_q).$$

Since

$$w = w_0[x^j(br_{[ar(\alpha_m)]}(\sigma^m)) \cdots (br_{[ar(\alpha_1)]}(\sigma^1))(v)] \in br(t)$$

and by construction

$$rhs(x^j) \ni y^q_{1,br(\nu_1)}(y^j_{q,br(\nu_q)}(y_{1,i}))$$

it follows by I.H.

$$\gamma \overset{*}{\underset{\mathrm{OI}}{\Longrightarrow}} w \underset{\mathrm{OI}}{\Longrightarrow} w' \quad \text{in } br(G).$$

(6.2) $p > 0$; similarly to (5.2).

Proof of Assertion 2. The base step is trivial.

Now let $A \in N$ and $\gamma \Rightarrow_{\mathrm{OI}} w \Rightarrow^*_{\mathrm{OI}} w'$ be a derivation in $br(G)$ with $\gamma \in br(A)$, such as

$$A = x(\sigma^m) \cdots (\sigma^1)(s_1, \ldots, s_k), \quad type(x) = (\alpha_m, \ldots, (\alpha_0, i) \ldots),$$

$$\gamma = x^j(br_{[ar(\alpha_m)]}(\sigma^m)) \cdots (br_{[ar(\alpha_1)]}(\sigma^1))(v)$$

for $j \in [k]$, $v \in br(s_j)$. Then it holds

$$w = v[br_{[ar(\alpha_m)]}(y_{\alpha_m})/br_{[ar(\alpha_m)]}(\sigma^m)]$$
$$\cdots [br_{[ar(\alpha_1)]}(y_{\alpha_1})/br_{[ar(\alpha_1)]}(\sigma^1)][y_{1,i}/v]$$

for $v \in rhs(x^j)$.

By construction there exists $s \in rhs(x)$ with $v \in br_j(s)$.

Let

$$t := s[y_{\alpha_m}/\sigma_m] \cdots [y_{\alpha_1}/\sigma_1][y_{\alpha_0}/(s_1, \ldots, s_k)].$$

By case-splitting according to the form of s we show that

$$w \in br(t).$$

Case 1: $s = f(y_{1,i}, \ldots, y_{k,i})$, $m = 0 \rightsquigarrow w = f^i(v) \in br(t)$.

Case 2: $s = a(\,) \rightsquigarrow w = e(\,) \in br(t)$.

Case 3: $s = x'(y_{\alpha_m}) \cdots (y_{\alpha_0})$.

(3.1) $level(x) = 0 \rightsquigarrow w = x'^0 \in br(x'^0)$.

(3.2) $level(x) > 0 \rightsquigarrow w = w_1(v) \in br(t)$ with

$$w_1 = x'^j(br_{[ar(\alpha_m)]}(\sigma^m)) \cdots (br_{[ar(\alpha_1)]}(\sigma^1)).$$

Case 4: $s = y_{i,\sigma_{p(r)}}(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0})$.

(4.1) $p = 0$

$$br_j(s) \neq \emptyset \rightsquigarrow r = j \rightsquigarrow w = v \in br(s_j) \wedge t = s_j.$$

(4.2) $p > 0$, $\nu = \alpha_p(r)$, $\sigma^p = (\sigma_1, \ldots, \sigma_q)$

$$\rightsquigarrow w = w_1(v) \in br(t)$$

with

$$w_1 = br_j(\sigma_r(\sigma^{p-1}) \cdots (\sigma^1))$$
$$= br_j(\sigma_r)(br_{[ar(\alpha_{p-1})]}(\sigma^{p-1})) \cdots (br_{[ar(\alpha_1)]}(\sigma^1)).$$

Case 5: $s = x_1(x_2(y_{\alpha_p}), \ldots, x_r(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0})$.

(5.1) $level(x) = 0$: similar to (6.1).

(5.2) $level(x) > 0$

$$\rightsquigarrow w = w_1(v) \in br(t)$$

with

$$w_1 = br_j(x_1(x_2(\sigma^p), \ldots, x_r(\sigma^p))(\sigma^{p-1}) \cdots (\sigma^1))$$
$$= x_1^j(br_{[ar(x_2)]}(x_2)(br_{[ar(\alpha_p)]}(\sigma^p)), \ldots, br_{[ar(x_r)]}(x_r)$$
$$(br_{[ar(\alpha_p)]}(\sigma^p)))(br_{[ar(\alpha_{p-1})]}(\sigma^{p-1})) \cdots (br_{[ar(\alpha_1)]}(\sigma^1)).$$

Case 6: $s = y_{1,\nu_1}(y_{2,\nu_2}(y_{\alpha_p}), \ldots, y_{r,\nu_r}(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_0})$ with $\alpha_{p+1} = \nu_1 \cdot \ldots \cdot \nu_r$, $\sigma^{p+1} = (\sigma_1, \ldots, \sigma_r)$.

(6.1) $p = 0$

$$\rightsquigarrow v = y^q_{1,br(\nu_1)}(y^j_{q,br(\nu_q)}(y_{1,i})) \quad \text{for } q \in \{2, \ldots, r\}$$

$$\rightsquigarrow w = \overbrace{br_q(\sigma_1)(br_j(\sigma_q)(v))}^{w_1} \in br(\sigma_1(\sigma_2(s_1, \ldots, s_k), \ldots, \sigma_r(s_1, \ldots, s_k))).$$

(6.2) $p > 0$: similar to (5.2).

In each case there exist w_0, w_1, a linearization $(t_0, t_1, \ldots, t_r)$ of t and $j \in [r]$, such that

$$w = w_0[w_1(v)], \quad w_0 \in br_j(t_0), \quad w_1(v) \in br(t_j)$$

and

$$w' = w_0[v'] \quad \text{with } w_1(v) \overset{*}{\underset{\text{OI}}{\Longrightarrow}} v'$$

therefore t_j' exists by induction hypothesis with

$$v' \in br(t_j) \wedge A \overset{*}{\underset{\text{OI}}{\Longrightarrow}} t_0[t_1, \ldots, t_j, \ldots, t_r]$$

$$\overset{*}{\underset{\text{OI}}{\Longrightarrow}} t_0[t_1, \ldots, t_j', \ldots, t_r] =: t'.$$

Obviously $w \in br(t')$.

This Assertion 2 is proven.

Now let $w \in br(L(G))$, such as $x_0\downarrow \overset{*}{\Rightarrow}_{\text{OI},G} t \in T_\Omega$ and $w \in br(t)$, then by Assertion 1

$$x_0^0\downarrow \overset{*}{\underset{\text{OI},br(G)}{\Longrightarrow}} w \in T_{\Omega_{br}},$$

therefore $w \in L(br(G))$.

If $w \in L(br(G))$, then since $x_0^0 \in br(x_0)$ by Assertion 2 $w \in br(t)$ for $t \in T_\Omega(N)$ with $x_0\downarrow \overset{*}{\Rightarrow}_{\text{OI},G} t$. Let $(t_0, t_1, \ldots, t_r)$ be a linearization of t.

If Ω is monadic, then since $w \in T_{\Omega_{br}}$, $r = 0$ and hence $t' = t_0 \in L(G)$ hold. If Ω is not monadic, then $w \in br(t_0)$. By (*) there exist $t_1', \ldots, t_r' \in T_\Omega$ with $t_j \overset{*}{\Rightarrow}_{\text{OI}} t_j'$, therefore

$$x_0\downarrow \overset{*}{\underset{\text{OI}}{\Longrightarrow}} t_0[t_1', \ldots, t_r'] =: t \in T_\Omega$$

and

$$w \in br(t_0) \subseteq br(t') \subseteq br(L(G)). \quad \square$$

We now discuss the characterization of frontier languages. In order to show that the front language of a level-n tree language can be generated at the $(n-1)$st level, we replace each operation symbol f with arity k by the functional $\lambda y_{1,(i,i)^k} \cdot y_{1,(i,i)} \circ \cdots \circ y_{k,(i,i)}$ which concatenates the fronts of the subtrees $t_1, \ldots, t_k$ of $f(t_1, \ldots, t_k)$.

Construction

(1) *Types*

$$fr : D^*(\{i\}) \to D^*(\{(i, i)\}),$$

$$i \mapsto (i, i),$$

$$(\alpha(1) \cdot \cdots \cdot \alpha(r), \nu) \mapsto (fr(\alpha(1)) \cdot \cdots \cdot fr(\alpha(r)), fr(\nu)).$$

(2) *Terms.* Let Ω be a $D(\{i\})$-set. Define a monadic alphabet Ω_{fr} by $\Omega_{fr}^{(i,i)} := \Omega^{(e,i)}$.
By induction on the construction of $t \in T_{\Omega,X,Y}$ we define the front of t, $fr_\lambda(t)$, by

$$fr_\lambda(a) := a \quad \text{for } a \in \Omega^{(e,i)},$$

$$fr_\lambda(f) := f := \lambda y_{(i,i)^k} . \lambda y_{1,i} . y_{1,(i,i)}(\cdots(y_{k,(i,i)}(y_{1,i}))\cdots)$$

for $f \in \Omega$ with $ar(f) = k$,

$$fr_\lambda(y_{i,\nu}) := y_{i,fr(\nu)}$$

$$fr_\lambda(x) := x,$$

$$fr_\lambda(t_0(t_1, \ldots, t_r)) := fr_\lambda(t_0)(fr_\lambda(t_1), \ldots, fr_\lambda(t_r)),$$

$$fr_\lambda(\lambda y_\alpha . t') := \lambda y_{fr(\alpha)} . fr_\lambda(t').$$

(3) *Grammar.* Let $G \in n - N\lambda(\Omega)$ with $var(G) = X$ and axiom $S = x_0\downarrow$.
We define $fr(G) \in n+1 - N\lambda(\Omega_{fr})$ by $var(fr(G)) := X \cup \{x_0'\}$ with $type(x_0') := i$ and productions $fr(G)(x_0') := S(e)$, $\forall x \in X\, fr(G)(x) = fr_\lambda(G(x))$.

Since fr_λ is a homomorphism with respect to abstraction, application and substitution, a derivation in G canonically induces a derivation in $fr(G)$ by replacing f by f and applying all terms to z. The following lemma shows that the normalform of the resulting term coincides with the front of the derived tree.

Lemma 7.14. $\forall t \in T_\Omega\ front(t) = nf(fr_\lambda(t)(e))$.

Proof. By induction on t we show that

$$front_0(t) \leftarrow w = nf(fr_\lambda(t)(w)).$$

Since $front(t) = front_0(t) \leftarrow e$ the assertion thus follows.

$-t = a$:

$$nf(fr_\lambda(a)(w)) = a(w)$$
$$= a(y_{1,i}) \leftarrow w = front_0(a) \leftarrow w;$$

$-t = ft_1 \cdots t_r$:

$$nf(fr_\lambda(f(t_1, \ldots, t_r))(w)) =$$
$$= nf(f(fr_\lambda(t_1), \ldots, fr_\lambda(t_r))(w))$$
$$= nf(\lambda y_{1,i} . fr_\lambda(t_1)(\cdots(fr_\lambda(t_r)(y_{1,i}))\cdots)(w))$$
$$= nf(fr_\lambda(t_1)(\cdots(fr_\lambda(t_r)(w))\cdots))$$
$$= nf(fr_\lambda(t_1)(\cdots(nf(fr_\lambda(t_r)(w))\cdots)) \qquad \text{Facts 5.3, 5.4}$$

$$= nf(fr_\lambda(t_1)(\cdots(front_0(t_r) \leftarrow w)\cdots)) \quad \text{I.H.}$$

$$\vdots$$

$$= nf(fr_\lambda(t_1)(front_0(t_2) \leftarrow \cdots \leftarrow front_0(t_r) \leftarrow w)) \quad \text{I.H.}$$

$$= front_0(t_1) \leftarrow front_0(t_2) \leftarrow \cdots \leftarrow front_0(t_r) \leftarrow w \quad \text{I.H.}$$

$$= front_0(t) \leftarrow w. \quad \square$$

The correctness of the construction is based on the following lemma.

Lemma 7.15. $fr_\lambda(t[y_\alpha/s]) = fr_\lambda(t)[y_{fr(\alpha)}/fr_\lambda(s)]$.

Proof. Immediate by induction on the construction of t. $\square$

We now show that $fr(G)$ generates the frontier language of $L(G)$.

Lemma 7.16. $front(L(G)) = L(fr(G))$.

Proof. '$\subseteq$' We prove first by induction on t

$$t \xrightarrow[G]{} s \rightsquigarrow fr_\lambda(t) \xrightarrow[fr(G)]{} fr_\lambda(s). \qquad (*)$$

$-t = x \rightsquigarrow s \in G(x) \rightsquigarrow fr_\lambda(x) = x \xrightarrow[fr(G)]{} fr_\lambda(s)$.

$-t \in \Omega \cup Y$: trivial.

$-t = \lambda y_\alpha.t'$: immediate by I.H.

$-t = t_0(t_1, \ldots, t_r)$:

Case 1: $\exists j\ t_j \xrightarrow[G]{} s_j$: immediate by I.H.

Case 2:

$$t_0 = \lambda y_\alpha.t' \wedge s = t'[y_\alpha/(t_1, \ldots, t_r)]$$

$$\rightsquigarrow fr_\lambda(t) = \lambda y_{fr(\alpha)}.fr_\lambda(t')(fr_\lambda(t_1), \ldots fr_\lambda(t_r))$$

$$\rightarrow fr_\lambda(t')[y_{fr(\alpha)}/(fr_\lambda(t_1), \ldots, fr_\lambda(t_r))]$$

$$= fr_\lambda(s) \quad \text{Lemma 7.15.}$$

Thus (*) is proven.

Now let $S = x_0\downarrow$ be the axiom of G, then it follows immediately from (*)

$$S \xrightarrow[G]{*} t \in T_\Omega \rightsquigarrow X \xrightarrow[fr(G)]{*} fr_\lambda(t)$$

and thus

$$x_0' \xrightarrow[fr(G)]{} S(e) \xrightarrow[fr(G)]{*} fr_\lambda(t)(e) \xrightarrow[\beta]{*} nf(fr_\lambda(t)(e)) = front(t) \quad \text{by Lemma 7.14.}$$

'$\supseteq$' Call an f-redex any redex of the form $f(t_1, \ldots, t_r)$.
We first show

$$\text{if } fr_\lambda(t) \xrightarrow[fr(G)]{} w \text{ is not an } f\text{-reduction, then there exists } s \text{ with } t \xrightarrow[G]{} s \wedge w = fr_\lambda(s) \tag{*}$$

by induction on t.

- $t = x \curvearrowright w = fr_\lambda(s)$ with $s \in G(x) \curvearrowright t \xrightarrow[G]{} s$.

- $t \in \Omega \cup Y$: trivial.

- $t = \lambda y_\alpha . t'$: immediate by I.H.

- $t = t_0(t_1, \ldots, t_r)$:

Case 1: $\exists j\, fr(t_j) \to w_j$: immediate by induction.

Case 2: $fr(t_0) = \lambda y_\alpha . w' \wedge w = w'[y_\alpha/(fr_\lambda(t_1), \ldots, fr_\lambda(t_r))]$.

By the assumption that t is not an f-redex we have $t_0 \neq f$. Since new abstractions are only introduced in the translation of terminal symbols we must have

$$\exists \beta \exists t'\ w' = fr_\lambda(t') \wedge \alpha = fr(\beta)$$

$$\curvearrowright w = fr_\lambda(t')[y_{fr(\beta)}/(fr_\lambda(t_1), \ldots, fr_\lambda(t_r))]$$

$$= fr_\lambda(s) \quad \text{with } s = fr_\lambda(t'[y_\beta/(t_1, \ldots, t_r)]) \quad \text{by Lemma 7.15.}$$

Obviously it holds $t \to s$.
This proves (*).
Now let $w \in L(fr(G))$. By Theorem 6.4 there exists a left reduction leading to w. Without loss of generality we may assume that the evaluation of the actual parameter e is delayed and carried out as the last reduction step. Then the above derivation D has to be of the form

$$x_0' \to S(e) \to w_1(e) \xrightarrow[\beta]{*} nf(w_1)(e) \to \cdots$$

$$\to w_n(e) \xrightarrow[\beta]{*} nf(w_n)(e) \xrightarrow[\beta]{} w.$$

Note that only the last β-reduction is a non-standard derivation step. Since in an f-reduction no actual parameters are copied, we can arrange that all f-reductions

are carried out at the end of derivation by interchanging f-reductions with non-f-reductions, starting from the right. This is possible since D is (essentially) a standard derivation and thus the redexes interchanged are either contained or situated to the right of the corresponding f-redex. Hence D is equivalent to a derivation D'

$$x_0' \to S(e) \to w_1(e) = v_1(e) \xrightarrow[\beta]{*} v_1'(e) \to \cdots$$

$$\to v_n(e) \xrightarrow[\beta]{*} v_n'(e) \xrightarrow[f]{*} nf(w_n)(e) \xrightarrow[\beta]{} w$$

where no f-reductions are carried out from v_j to v_j'. But then $S \to v_1 \to_\beta^* v_1' \to \cdots \to v_n \to_\beta^* v_n'$ is a derivation in $fr(G)$, hence by (*) and the fact that $fr_\lambda(S) = S$ there must exist terms t_j, t_j' with $fr_\lambda(t_j) = v_j$, $fr_\lambda(t_j') = v_j'$ such that

$$S \to t_1 \xrightarrow[\beta]{*} t_1' \to \cdots \to t_n \xrightarrow[\beta]{*} t_n' \quad \text{in } G.$$

Since f is linear, t_n' inherits from v_n' the property of being a term over parameters and operation symbols only. Moreover, v_n' contains only f-redexes, hence t_n' is in normalform and thus by the above $t_n' \in L(G)$. This proves w to be the front of a tree in the language generated by G:

$$\begin{aligned} front(t_n') &= nf(fr_\lambda(t_n')(e)) && \text{Lemma 7.14} \\ &= nf(v_n'(e)) \\ &= nf(nf(w_n)(e)) = w && \text{Facts 5.3, 5.4} \quad \square \end{aligned}$$

We shall now characterize level-$(n+1)$ string languages as frontier languages of level-n tree languages.

By the construction used in the path language characterization it suffices to consider string grammars in Chomsky-normalform, such that all types of nonterminals and/or parameters are either a base-type or of the form $(\alpha_m, \ldots, ((i,i)^n, (i,i)) \ldots)$. It is easy to see that such a grammar is equivalent to a grammar which uses e only in a new initial production $x_0' = x_0(e(\,))$ and where x_0' is the only nonterminal of a ground type (otherwise replace each expression $x\downarrow$ with

$$type(x) = (\underbrace{e, \ldots, (e,i) \ldots}_{k}) \quad \text{by} \quad x\underbrace{(\,)\cdots(\,)}_{k-1}(y_{1,i})$$

and $e(\,)$ by $y_{1,i}$, and add the production

$$x_0' = x_0\underbrace{(\,)\cdots(\,)}_{level(x_0)-1}(e(\,)).$$

We now associate with such a level-$(n+1)$ string grammar G a level-n grammar $c(G)$ (essentially the image under *comb*) by writing an application $t_1(t_2(y_{1,i}))$ as composition $t_1 \circ t_2$ and omitting the parameters $y_{1,i}$ in all higher level productions.

Taking the image of $c(G)$ under *front* yields again the grammar G, hence by the above lemma the frontier language of $c(G)$ coincides with the language generated by G.

Construction

(1) *Types*:

$$c: D^*(\{(i,i)\}) \mapsto D^*(\{i\}),$$

$$(i,i) \mapsto i, \qquad (\alpha, \nu) \mapsto (c(\alpha(1)) \cdot \cdots \cdot c(\alpha(\cdot)), c(\nu)).$$

(2) Let Ω be a monadic $\{i\}$-sorted alphabet. Define an $\{i\}$-sorted alphabet Ω_c by

$$\Omega_c^{(e,i)} := \Omega^{(i,i)}, \qquad \Omega_c^{(i^2,i)} := \{\circ\}.$$

We use the symbol for composition in infix-notation.

(3) Assume, that all productions of $G \in n+1-N\lambda(\Omega)$ are of one of the following forms:

(3.1) axiom $x_0' = x_0(\ ) \cdots (\ )(e(\ ))$
(3.2) $x\downarrow = a(y_{1,i})$
(3.3) $x\downarrow = x_1(x_2(y_{1,i}))$
(3.4) $x\downarrow = x_1(x_2(y_{\alpha_p}), \ldots, x_r(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_1})(y_{1,i})$
(3.5) $x\downarrow = x'(y_{\alpha_p}) \cdots (y_{\alpha_1})(y_{1,i})$
(3.6) $x\downarrow = y_{1,(i,i)}(y_{2,(i,i)}(y_{1,i}))$
(3.7) $x\downarrow = y_{1,\nu_1}(y_{2,\nu_2}(y_{\alpha_p}), \ldots, y_{r,\nu_r}(y_{\alpha_p}))(y_{\alpha_{p-1}}) \cdots (y_{\alpha_1})(y_{1,i})$
(3.8) $x\downarrow = y_{1,\nu}(y_{\alpha_p}) \cdots (y_{\alpha_1})(y_{1,i})$.

$c(G) \in n - N\lambda(\Omega_c)$ has as nonterminals $var(G)\backslash\{x_0'\}$ with re-defined types. The productions of $c(G)$ are obtained by substituting each production of the form

(3.2) by $x\downarrow = a$
(3.3) by $x\downarrow = x_1 \circ x_2$
(3.4) by $x\downarrow = x_1(x_2(y_{c(\alpha_p)}), \ldots, x_r(y_{c(\alpha_p)}))(y_{c(\alpha_{p-1})}) \cdots (y_{c(\alpha_1)})$
(3.5) by $x\downarrow = x'(y_{c(\alpha_p)}) \cdots (y_{c(\alpha_1)})$
(3.6) by $x\downarrow = y_{1,i} \circ y_{2,i}$
(3.7) by $x\downarrow = y_{1,c(\nu_1)}(y_{2,c(\nu_2)}(y_{c(\alpha_p)}), \ldots,$
$y_{r,c(\nu_r)}(y_{c(\alpha_p)}))(y_{c(\alpha_{p-1})}) \cdots (y_{c(\alpha_1)})$
(3.8) by $x\downarrow = y_{1,c(\nu)}(y_{c(\alpha_p)}) \cdots (y_{c(\alpha_1)})$.

This proves the following characterisation of OI-string languages.

Theorem 7.17. *The frontier languages of $n - \mathcal{L}_{OI}$ coincide with the level-$(n+1)$ OI-string languages.*

Proof. '$\subseteq$' Let $L = front(L(G))$ with $G \in n - N\lambda(\Omega)$, then $L = L(fr(G))$ by Lemma 7.16, hence $L \in (n+1) - \mathcal{L}_{OI}(\Omega_{fr})$.

'$\supseteq$' Let Ω be a monadic alphabet and $L = L(G)$ for some $G \in (n+1) - N\lambda(\Omega)$. Then there exists an equivalent level-$n+1$ grammar G' which contains only productions of the form (3.1)–(3.8) (take the Chomsky-normalform G_1 of G and use the above hint to construct G' out of $br(G_1)$). Since $fr(c(\tau)) = \tau \in D^*(\{i\})$ and hence (up to β-reduction) $fr_\lambda(c(G'(x))) = G'(x)$ we have by Lemma 7.16 $L(G) = L(G') = L(fr(c(G))) = front(L(c(G)))$. □

Since $0 - N\lambda(\Omega)$ and $1 - N\lambda(\Omega)$ coincide with the class of regular and/or context-free tree grammars over Ω and frontier-languages of context-free tree languages define exactly the class of macro-languages [56], the OI-hierarchy starts for a monadic alphabet with the regular, context-free and macro-languages.

Corollary 7.18. *Let Ω be a monadic alphabet. Then*

$$0 - \mathscr{L}_{\mathrm{OI}}(\Omega) = \mathscr{REG}(\Omega), \qquad 1 - \mathscr{L}_{\mathrm{OI}}(\Omega) = \mathscr{CF}(\Omega), \qquad 2 - \mathscr{L}_{\mathrm{OI}}(\Omega) = \mathscr{MAC}(\Omega).$$

A sketch of a proof of this result can also be found in [28, 70]. An alternate proof using top–down tree transducers on infinite trees is given in [5].

7.5. Closure properties

In this section we will demonstrate, that level-n OI string languages form a substitution closed AFL.

We start by investigating closure under intersection with regular languages. The construction employed is an extension of the classical state-product construction for context-free tree-grammars (c.f. [61]): in an OI-derivation, the current state of the finite automata is memorized in the top-nonterminals; as soon as a new terminal symbol is derived, it is fed as input to the automata and the new state is again stored in the top-nonterminal. To avoid the (technical!) elimination of rules which generate a special symbol for nonacceptance (c.f. [61]), we simulate the intersection at the level of trees over Ω_+ and use $\perp$ to 'erase' nonaccepted branches.

Essentially, intersection of languages over Ω corresponds to taking meets of the associated languages over Ω_+. For the monadic case considered in this section, the meet operation can be further specialized. We need the following notation.

Notation. Let Ω be a monadic alphabet, $t \in \mathrm{FT}_{\Omega_+}(N)$, $R \subseteq T_\Omega$, and $L = \bigcup set(L_+)$ with $L_+ \subseteq \mathrm{FT}_{\Omega_+}$.

(1) $R_+ := \{t \in \mathrm{FT}_{\Omega_+} \mid set(t) \subset R\}$.

(2) $(s, t_1, \ldots, t_k)$ is a *full linearization of* t iff

(i) $t_i \in N \cup \{e, \perp\}$,

(ii) $front_0(s) = y_{i,1} \leftarrow \cdot \quad \cdot \leftarrow y_{i,k}$,

(iii) $t = s[t_1, \ldots, t_k]$.

Clearly each t has exactly one full linearization. In the sequel we will restrict ourselves to full linearizations.

(3) If $(s, t_1, \ldots, t_k)$ is as above, we set $leave_j(t) := t_j$, $breadth(t) := k$.

(4) $path_j(t)$ is the string in $\Omega^{(i,i)}$ leading to $leave_j(t)$. The set $path(t) = \{path_j(t) \mid j \in [breadth(t)] \wedge leave_j(t) = e\}$ contains all paths of t leading to an e-leave.

(5) Define a partial order $\sqsubseteq$ on $\mathrm{FT}_{\Omega_+}(N)$ by

$$t \sqsubseteq t' \text{ iff } t = s[t_1, \ldots, t_k] \wedge t' = s'[t'_1, \ldots, t'_k] \rightsquigarrow$$
$$s = s' \wedge k = k' \wedge \forall j \in [k]\, t_j \neq t'_j \rightsquigarrow t_j = \bot,$$

i.e. t can be obtained from t' by replacing some *leaves* by $\bot$.

(6) The *meet* of L_+ and R_+ is given by

$$L_+ \sqcap R_+ := \{s \in R_+ \mid \exists t \in L_+\, s \sqsubseteq t\}.$$

It is now straightforward to prove, that the intersection of L and R corresponds to the meet of L_+ and R_+.

Lemma 7.19. $\forall t \in \mathrm{FT}_{\Omega_+}\ path(t)e = set(t)$.

Proof: Immediate by induction over t. □

Lemma 7.20. $\forall t \in \mathrm{FT}_{\Omega_+}\ S \subseteq set(t) \rightsquigarrow \exists s \sqsubseteq t\ S = set(s)$.

Proof. Let $t = t_0[t_1, \ldots, t_k]$. By Lemma 7.19 we have

$$S = \{path_{i_1}(t)e, \ldots, path_{i_r}(t)e\} \quad \text{for some } r \in \{0, \ldots, k\}.$$

Define $s := t_0[t'_1, \ldots, t'_k]$, where

$$t'_j = \begin{cases} e & \text{if } j \in \{i_1, \ldots, i_r\}, \\ \bot & \text{otherwise.} \end{cases}$$

By Lemma 7.19 we have $S = set(s)$. □

Corollary 7.21. $L \cap R = \bigcup set(L_+ \sqcap R_+)$.

Proof. '$\subseteq$' Let $w \in set(t) \cap R$ with $t \in L_+ \rightsquigarrow$ by Lemma 7.20 $\exists s \sqsubseteq t\ w \in set(s) = set(t) \cap R$.

Then $set(s) \subseteq R \rightsquigarrow s \in R_+$, hence $s \in L_+ \sqcap R_+$.

'$\supseteq$' Let $w \in set(s)$ with $s \in L_+ \sqcap R_+$

$\rightsquigarrow s \in R_+ \wedge \exists t \in L_+\, s \sqsubseteq t$

$\rightsquigarrow w \in set(t) \wedge w \in set(s) \subseteq R$

$\rightsquigarrow w \in \bigcup set(L_+) \cap R.$ □

Now let L be a level-n OI language over Ω, and $R = L(\mathcal{Q})$ for some Rabin–Scott-automata $\mathcal{Q}$ with states $[k]$, input alphabet Ω^1, transition function δ, initial state 1, and final states F. Then R_+ is the set of trees such that a path is either accepted by $\mathcal{Q}$ or leads to $\perp$. By Corollary 4.11 and Theorems 6.7 and 7.2 we have $L = \bigcup set(L(S_\perp))$ for some level-n λ-scheme $S \in n-\lambda(\Omega_+)$. Thus it is sufficient to construct some $\bar{S} \in n-\lambda(\Omega_+)$, such that the schematic language generated by $\bar{S}$ contains exactly those trees of R_+, which differ at most at $\perp$-leaves 'by e' from trees in $L(S_\perp)$. Clearly $\bar{S}_\perp$ has to simulate (OI-)derivations of $S_\perp$ except that an e-leave is only generated in case the path leading to this leave takes $\mathcal{Q}$ into a final state. Recall that the current state will be memorized in the head-nonterminals.

While simulating a production $x\downarrow = x_1(x_2(y_\alpha), \ldots, x_r(y_\alpha))\downarrow$, say, we do not know in advance, which state the automata will have reached while working on strings generated from x_j with $j \geqslant 2$. To cope with this, we allow all possible states and, later on, choose the correct state by means of projections.

Construction

(1) *Types*

$$\bar{\ }: D^*(\{i\}) \to D^*(\{i\}),$$

$$\bar{i} := i, \quad \overline{(\alpha(1) \cdot \ldots \cdot \alpha(r), \nu)} := (\overline{\alpha(1)})^k \cdot \ldots \cdot \overline{\alpha(r)}^k, \bar{\nu}).$$

We identify $[k] \times Y_\alpha$ and $Y_{\bar{\alpha}^k}$ via

$$(q, y_{j,\alpha(j)}) \mapsto y_{l,\bar{\alpha}} k_{(l)} \quad \text{with } l = (j-1) \cdot k + q.$$

(2) *Terms*

(2.1) Let $t \in T^\tau_{X,Y}$ be an applicative term.

We define inductively $\bar{t} \in T^{\bar{\tau}^k}_{[k]\times X, [k]\times Y}$ by

$$\bar{x} := (1x, \ldots, kx) \quad \text{with } type(qx) := \overline{type(x)},\ \bar{y} := (1y, \ldots, ky),$$

$$\overline{t(t_1, \ldots, t_r)} := (pr_1(\bar{t})(\bar{t}_1, \ldots, \bar{t}_r), \ldots, pr_k(\bar{t})(\bar{t}_1, \ldots, \bar{t}_r)).$$

(2.2) Let $\bar{N} := \{pr_j(\bar{t}) \mid t \in N \wedge j \in [k]\}$.

$\_: \mathcal{FT}_{\Omega_+}(\bar{N}) \to \mathcal{FT}_{\Omega_+}(N)$ is the unique Ω_+-homomorphism generated by $pr_j(t) \mapsto t$.

(3) Let $S \in n-\lambda(\Omega_+)$ in Chomsky-normalform, $\mathcal{Q} = ([k], \Omega^1, \delta, 1, F)$ a Rabin–Scott automata.

Define $\bar{S} \in n-\lambda(\Omega_+)$ with $var(\bar{S}) := [k] \times var(S)$ by

(3.1) $x\downarrow = e(\ )$ *in* $S \rightsquigarrow \forall q \in F\ qx\downarrow = e(\ )$ in $\bar{S}$.

(3.2) $x\downarrow = +\downarrow$ in $S \rightsquigarrow \forall q \in [k]\ qx\downarrow = +(qy_{1,i}, qy_{2,i})$ in $\bar{S}$.

(3.3) $x\downarrow = a\downarrow$ in $S \rightsquigarrow \forall q \in [k]\ qx\downarrow = a(\delta(q, a)y_{1,i})$ in $\bar{S}$.

(3.4) $x\downarrow = x'\downarrow$ in $S \rightsquigarrow \forall q \in [k]\ qx\downarrow = qx'\downarrow$ in $\bar{S}$.

(3.5) $x\downarrow = x_1(x_2(y_\alpha), \ldots, x_r(y_\alpha))\downarrow$ in S

$\rightsquigarrow \forall q \in [k]\ qx_1(\overline{x_2(y_\alpha)}, \ldots, \overline{x_r(y_\alpha)})\downarrow$ in $\bar{S}$.

(3.6) $x\downarrow = y_{j,\nu}\downarrow$ in $S \rightsquigarrow \forall q \in [k]\, qx\downarrow = qy_{j,\nu}\downarrow$ in $\bar{S}$.

(3.7) $x\downarrow = y_{1,\nu_1}(y_{2,\nu_2}(y_\alpha), \ldots, y_{r,\nu_r}(y_\alpha))\downarrow$ in S

$\rightsquigarrow \forall q \in [k]\, qy_{1,\nu_1}(y_{2,\nu_2}(y_\alpha), \ldots, y_{r,\nu_r}(y_\alpha))\downarrow$ in $\bar{S}$.

To aid intuition, we will first give an example.

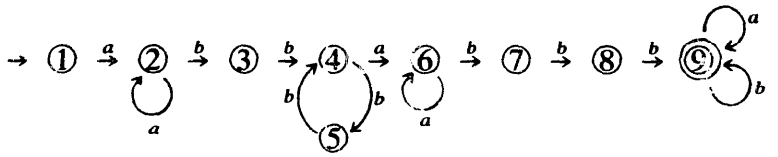
Example. Consider the grammar G_3 of Section 7.1. In order to be able to carry out the construction, it is sufficient to replace each occurrence of a, b, and e in G_{3+} by nonterminals $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{e}$, respectively, and to add the productions

$$\boldsymbol{a}\downarrow = a\downarrow, \quad \boldsymbol{b}\downarrow = b\downarrow, \quad \boldsymbol{e}\downarrow = e.$$

Let $S \in 3\text{-}\lambda(\Omega_+)$ denote the modified version of G_{3+}. Now consider the regular set

$$R_2 := a^+(b^2)^+a^+(b^3)^+(a \vee b)^* = a^+(b^2)^+a^+b^3(a \vee b)^*$$

which will be employed in Section 9 to evaluate lower bounds on the 'complexity' of the sample languages. R_2 is recognized deterministically by



The set of productions of $\bar{S}$ consists of the extended rules of S

$$1x_0 = 1x_3(\overline{x_1})(\bar{b})(\bar{e}), \quad qx_1\downarrow = qy_1(\overline{y_1(y_0)}), \quad qx_2\downarrow = qy_2(\overline{y_2(y_1)})(\overline{y_0}),$$

$$qx_3\downarrow = +(qx_3(\overline{x_2(y_2)})(\overline{B(y_1)})(\overline{A(y_2)(y_1)(y_0)}), qA(\overline{y_2})(\overline{y_1})(\overline{y_0})),$$

$$q\boldsymbol{B}\downarrow = qb(\overline{y_1(y_0)}), \quad q\boldsymbol{A}\downarrow = qy_2(\bar{\boldsymbol{a}})(\overline{y_1(y_0)})$$

and productions modelling the behaviour of the finite automata

$$q\boldsymbol{a}\downarrow = a(\delta(q, a)y_0), \quad q\boldsymbol{b}\downarrow = b(\delta(q, b)y_0), \quad q\boldsymbol{e} = e.$$

An example on an OI-derivation in $\bar{S}_\perp$ is

$$1x_0 \Rightarrow 1x_3(\overline{x_1})(\bar{b})(\bar{e})$$

$$\Rightarrow +(t, 1A(\overline{x_1})(\bar{b})(\bar{e}))$$

with $t := 1x_3(\overline{x_2(x_1)})(\overline{B(b)})(\overline{A(x_1)(b)(e)})$

$$\Rightarrow +(t, 1x_1(\bar{a})(\overline{b(e)}))$$

$$\Rightarrow +(t, 1a(\overline{a(b(e))}))$$

$$\Rightarrow +(t, a(2a(\overline{b(e)})))$$

$$\Rightarrow +(t, a(a(2b(\bar{e}))))$$

$$\Rightarrow +(t, a(a(b(3e))))$$

$$\Rightarrow +(t, aab(\perp))$$

$\Rightarrow +(+(t', 1A(\overline{x_2(x_1)})(\overline{B(b)})(\overline{A(x_1)(b)(e)})), w_1)$

with $w_1 = aab(\perp)$,

$t' = 1x_3(\overline{x_2^2(x_1)})(\overline{B^2(b)})(\overline{A(x_2(x_1)}(\overline{B(b)})(\overline{A(x_1)(b)(e)}))$

$\Rightarrow +(+(t', 1x_2(\overline{x_1})(\bar{a})(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', 1x_1(\overline{x_1(a)})(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', 1x_1(\bar{a})(\overline{x_1(a)}(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', 1a(\bar{a}(\overline{x_1(a)}(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a(\overline{2a}(\overline{x_1(a)}(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a(a(2x_1(\bar{a})(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a(a(2a(\bar{a}(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a(a(a(2a(\overline{B(b)}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a(a(a(a(2B(\bar{b})(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a^4(2b(\bar{b}(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a^4(b(3b(\overline{A(x_1)(b)(e)}))), w_1)$

$\Rightarrow +(+(t', a^4(b(b(4A(\overline{x_1})(\bar{b})(\bar{e})))), w_1)$

$\Rightarrow +(+(t', a^4(b^2(4x_1(\bar{a})(\overline{b)(e)}))), w_1)$

$\Rightarrow +(+(t', a^4(b^2(4a(\overline{a(b(e)}))), w_1)$

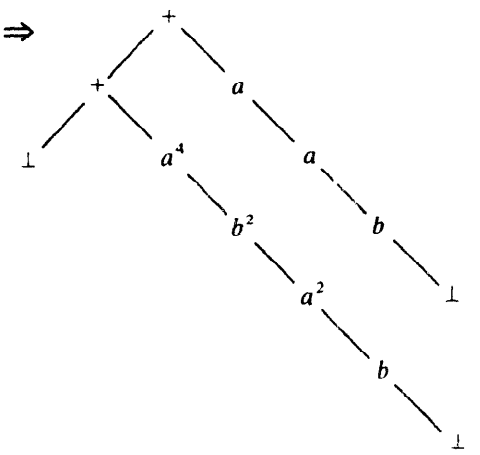
$\Rightarrow +(+(t', a^4(b^2(a(6a(b(e)))), w_1)$

$\Rightarrow +(+(t', a^4(b^2(a^2(6b(e)))), w_1)$

$\Rightarrow +(+(t', a^4b^2a^2b(7e), w_1)$

$\Rightarrow +(+(t', a^4b^2a^2b(\perp)), w_1)$

$\Rightarrow$



It is easy to see that the shortest word in the intersection of $L(G_3)$ and R_2 is

$$a^{256}b^4a^{16}b^3a^4b^2a^2b.$$

The following three lemmata demonstrate the correctness of the construction. Since we only deal with OI-derivations, we will simply write $\Rightarrow$ for $\Rightarrow_{OI}$.

We prove first, that $\bar{S}_\perp$ only generates trees, which are derivable in $S_\perp$.

Lemma 7.22.

$$1x_0\downarrow \overset{*}{\underset{\bar{S}_\perp}{\Longrightarrow}} t \curvearrowright x_0\downarrow \overset{*}{\underset{S_\perp}{\Longrightarrow}} \underline{t}$$

Proof. By induction on the length of a derivation. The base step is trivial. Now consider a derivation

$$1x_0\downarrow \overset{*}{\underset{\bar{S}_\perp}{\Longrightarrow}} t \underset{\bar{S}_\perp}{\Longrightarrow} t'.$$

Let $(s, s_1, \ldots, s_r)$ be the full linearization of t, and assume (without loss of generality) that $s_1 = qx(\bar{\sigma}_m)\cdots(\bar{\sigma}_0)$ is the subterm replaced in the last derivation step. By induction hypothesis we have

$$x_0\downarrow \overset{*}{\underset{S_\perp}{\Longrightarrow}} \underline{t} = s[x(\sigma_m)\cdots(\sigma_0), \underline{s}_2, \ldots, \underline{s}_r].$$

If t' is obtained from t by a $\perp$-production, the assertion is obviously true. We proceed by considering cases (3.1) to (3.7) in the construction of $\bar{S}$.

(3.1) $t' = s[e(\ ), s_2, \ldots, s_r]$ and $\underline{t} \Rightarrow_{S_\perp} \underline{t}'$ since $x\downarrow = e(\ )$ in S.

(3.2) Let $\bar{\sigma}_0 = (\bar{t}_1, \bar{t}_2)$

$$\curvearrowright t' = s[+(pr_q(\bar{t}_1), pr_q(\bar{t}_2)), \underline{s}_2, \ldots, \underline{s}_r]$$

$$\wedge\ \bar{t} \underset{\bar{S}_\perp}{\Longrightarrow} s[+(t_1, t_2), \underline{s}_2, \ldots, \underline{s}_r] \quad \text{since } x\downarrow = +\downarrow \text{ in } S$$

$$= \underline{t}' \quad \text{since } +(t_1, t_2) = \underline{+(pr_q(\bar{t}_1), pr_q(\bar{t}_2))}.$$

(3.3) Similar to (3.2).

(3.4) $t' = s[qx'(\bar{\sigma}_p)\cdots(\bar{\sigma}_0), \underline{s}_2, \ldots, \underline{s}_r]$ with $p \leq m$

$$\curvearrowright \underline{t} \underset{S_\perp}{\Longrightarrow} \underline{t}' = s[\underline{pr_q(\overline{x'(\sigma_p)\cdots(\sigma_0)})}, \underline{s}_2, \ldots, \underline{s}_r]$$

(3.5) $t' = s[qx_1(\overline{x_2(\sigma_p)}, \ldots, \overline{x_r(\sigma_p)})\overline{(\sigma_{p-1})}\cdots\overline{(\sigma_0)}, \underline{s}_2, \ldots, \underline{s}_r]$

$$\curvearrowright \underline{t} \underset{S_\perp}{\Longrightarrow} \underline{t}' = s[\underline{pr_q(\overline{x_1(x_2(\sigma_p), \ldots, x_r(\sigma_p))(\sigma_{p-1})\cdots(\sigma_0)})}, \underline{s}_2, \ldots, \underline{s}_r].$$

(3.6) Similar to (3.4).

(3.7) Similar to (3.5). □

Next we show, that $\bar{S}_\perp$ generates only trees with paths leading to $\perp$ or accepted by $\mathcal{Q}$.

Notation. (7) δ^* denotes the (total) extension of δ to strings.

(8) For $A \in \bar{N}$ we denote by $state(A) := pr_1(head(A))$ the state memorized in the top nonterminal of A.

Lemma 7.23. *Let* $1x_0\downarrow \overset{*}{\Rightarrow}_{\bar{S}_\perp} t$ *and* $l := breadth(t)$. *We have*

$$\forall j \in [l] \quad (leave_j(t) = e \rightsquigarrow \delta^*(path_j(t)) \in F)$$
$$\wedge\, (leave_j(t) \in \bar{N} \rightsquigarrow \delta^*(path_j(t)) = state(leave_j(t))).$$

Proof. By induction on the length of a derivation. Again, the base step is trivial since

$$\delta^*(path_1(1x_0\downarrow)) = \delta^*(e) = 1 = state(1x_0\downarrow).$$

Now consider a derivation $1x_0\downarrow \overset{*}{\Rightarrow}_{\bar{S}_\perp} t \Rightarrow_{\bar{S}_\perp} t'$, and assume, that t satisfies the above assertion. Let $(s, s_1, \ldots, s_l)$ denote the full linearization of t, and assume, $s_1 = qx(\bar{\sigma}_m) \cdots (\bar{\sigma}_0)$ is the subterm replaced, then by induction hypothesis $q = path_1(t)$ $(= path_1(s))$.

In case a $\perp$-production is applied, the assertion is obviously true. We now discuss cases (3.1) to (3.7).

(3.1) $t' = s[e(\ ), s_2, \ldots, s_l] \rightsquigarrow$ by construction

$$q = \delta^*(path_1(t)) = \delta^*(path_1(t')) \in F.$$

(3.2) $\sigma_0 = (t_1, t_2) \wedge t' = s[+(pr_q(\bar{t}_1), pr_q(\bar{t}_2)), s_2, \ldots, s_l]$

$$\rightsquigarrow state(leave_1(t')) = state(pr_q(\bar{t}_1)) = state(pr_q(\bar{t}_2))$$
$$= state(leave_2(t')) = q = \delta^*(path_1(t)) = \delta^*(path_1(t')).$$

(3.3) $t' = s[a(pr_q, (\bar{\sigma}_0)), s_2, \ldots, s_l]$

$$\rightsquigarrow state(leave_1(t')) = state(pr_q, (\bar{\sigma}_0)) = q'$$
$$= \delta(q, a) = \delta(\delta^*(path_1(t)), a) = \delta^*(path_1(t)a)$$
$$= \delta^*(path_1(t'))$$

(3.4)–(3.7) Direct by I.H.,

$$path_1(t') = path_1(t)$$
$$\wedge\, state(leave_1(t')) = state(leave_1(t)). \quad \square$$

The next lemma implies the 'completeness' of $\bar{S}_\perp$: all trees in R_+ which are majorized by a tree in $L(S_\perp)$ can be generated in $\bar{S}_\perp$.

Lemma 7.23. *Let* $x_0\downarrow \overset{*}{\Rightarrow}_{S_\perp} t$, $q_j := \delta^*(path_j(t))$, *and let* $(s, s_1, \ldots, s_l)$ *denote the full linearization of* t.

Then for all $\tilde{s}_j$ *with*

(1) $s_j = e \wedge q_j \notin F \rightsquigarrow \tilde{s}_j = \perp$,
(2) $s_j = e \wedge q_j \in F \rightsquigarrow \tilde{s}_j = \perp \vee \tilde{s}_j = e$,
(3) $s_j = \perp \rightsquigarrow \tilde{s}_j = \perp$,
(4) $s_j \in N \rightsquigarrow \tilde{s}_j = pr_{q_j}(\tilde{s}_j)$

we have

$$1x_0\downarrow \overset{*}{\underset{\bar{S}_\perp}{\Longrightarrow}} s[\tilde{s}_1, \ldots, \tilde{s}_l]$$

Proof. By induction on the length of a derivation. The base step holds trivially, since

$$1x_0\downarrow = pr_1(\overline{x_0\downarrow}) \quad \text{and} \quad \delta^*(path_1(x_0\downarrow)) = \delta^*(e) = 1.$$

Now let t, $(s, s_1, \ldots, s_l)$, and q_j be as above, and assume

$$x_0\downarrow \overset{*}{\underset{S_\perp}{\Longrightarrow}} t \underset{S_\perp}{\Longrightarrow} t'.$$

Let $s_1 = x(\sigma_m)\cdots(\sigma_0)$ be the subterm replaced in the last step. Then, by induction hypothesis, we have with $q := q_1$

$$1x_0\downarrow \overset{*}{\underset{\bar{S}_\perp}{\Longrightarrow}} s[qx(\bar{\sigma}_m)\cdots(\bar{\sigma}_0), \tilde{s}_2, \ldots, \tilde{s}_l] =: \tilde{t}.$$

Clearly the assertion holds, if a $\perp$-rule was applied in the last step. We now discuss cases (3.1) to (3.7).

(3.1) $t' = s[e(\,), s_2, \ldots, s_l]$. Since $qx\downarrow = \perp$ is a production in $\bar{S}_\perp$ we always have $\tilde{t} \Rightarrow_{\bar{S}_\perp} s[\perp, \tilde{s}_1, \ldots, \tilde{s}_l]$.

Moreover, for $q \in F$, $\bar{S}_\perp$ contains the production $qx\downarrow = e(\,)$ and thus $\tilde{t} \Rightarrow_{\bar{S}_\perp} s[e, \tilde{s}_1, \ldots, \tilde{s}_l]$.

(3.2) $\sigma_0 = (t_1, t_2) \wedge t' = s[+(t_1, t_2), s_2, \ldots, s_l]$.
By construction

$$\tilde{t} \underset{\bar{S}_\perp}{\Longrightarrow} s[+(pr_q(t_1), pr_q(t_2), \tilde{s}_2, \ldots, \tilde{s}_l],$$

hence the assertion follows from $path_1(t') = path_2(t') = path_1(t)$.

(3.3) $t' = s[a(\sigma_0), s_2, \ldots, s_l]$

$\rightsquigarrow path_1(t') = path_1(t)a \rightsquigarrow \delta^*(path_1(t')) = \delta(q, a) =: q'$

$\rightsquigarrow qx\downarrow = a(q'y_{1,i})$ in $S_\perp$

$\rightsquigarrow \tilde{t} \underset{\bar{S}_\perp}{\Longrightarrow} s[a(pr_{q'}(\bar{\sigma}_0)), \tilde{s}_2, \ldots, \tilde{s}_l]$.

(3.4) $t' = s[x'(\sigma_p)\cdots(\sigma_0), s_2, \ldots, s_l]$

$\curvearrowright\ path_1(t) = path_1(t') \curvearrowright\ \delta^*(path(t')) = q$

and by construction

$$\tilde{t} \underset{\tilde{S}_\perp}{\Longrightarrow} s[pr_q(\overline{x'(\sigma_p)\cdots(\sigma_0)}), \tilde{s}_2, \ldots, \tilde{s}_l].$$

(3.5) $t' = s[x_1(x_2(\sigma_p), \ldots, x_r(\sigma_p))(\sigma_{p-1})\cdots(\sigma_0), s_2, \ldots, s_l]$

$\curvearrowright\ path_1(t) = path_1(t') \curvearrowright\ \delta^*(path_1(t')) = q$

and by construction

$$\tilde{t} \underset{\tilde{S}_\perp}{\Longrightarrow} s[q x_1(\overline{x_2(\sigma_p)}, \ldots, \overline{x_r(\sigma_p)})(\overline{\sigma_{p-1}})\cdots(\overline{\sigma_0}), \tilde{s}_2, \ldots, \tilde{s}_l]$$

$$= s[pr_q(\overline{x_1(x_2(\sigma_p), \ldots, x_r(\sigma_p))(\sigma_{p-1})\cdots(\sigma_0)}), \tilde{s}_2, \ldots, \tilde{s}_l].$$

(3.6) $\nu = \alpha_{p+1}(j), pr_j(\sigma_{p+1}) = \sigma, t' = s[\sigma(\sigma_p)\cdots(\sigma_0), s_2, \ldots, s_l]$

$\curvearrowright\ path_1(t) = path_1(t') \curvearrowright\ \delta^*(path_1(t')) = q$

and by construction

$$\tilde{t} \underset{\tilde{S}_\perp}{\Longrightarrow} s[pr_q(\bar{\sigma})(\bar{\sigma}_p)\cdots(\bar{\sigma}_0), \tilde{s}_2, \ldots, \tilde{s}_l]$$

$$= s[pr_q(\overline{\sigma(\sigma_p)\cdots(\sigma_0)}), \tilde{s}_2, \ldots, \tilde{s}_l].$$

(3.7) $\nu_1\cdots\nu_r = \alpha_{p+1}, \sigma_{p+1} = (t_1, \ldots, t_r),$

$t' = s[t_1(t_2(\sigma_p), \ldots, t_r(\sigma_p))(\sigma_{p-1})\cdots(\sigma_0), s_2, \ldots, s_l]$

$\curvearrowright\ path_1(t) = path_1(t') \curvearrowright\ \delta^*(path_1(t')) = q$

and by construction

$$\tilde{t} \underset{\tilde{S}_\perp}{\Longrightarrow} s[pr_q(\overline{t_1})(\overline{t_2(\sigma_p)}, \ldots, \overline{t_r(\sigma_p)})(\overline{\sigma_{p-1}})\cdots(\overline{\sigma_0}), \tilde{s}_2, \ldots, \tilde{s}_l]$$

$$= s[pr_q(\overline{t_1(t_2(\sigma_p), \ldots, t_r(\sigma_p))(\sigma_{p-1})\cdots(\sigma_0)}), \tilde{s}_2, \ldots, \tilde{s}_l].$$

Summarizing these results yields the correctness of the construction: $\bar{S}_\perp$ generates exactly the trees in the meet of R_+ with the schematic language generated by S. Moreover, the size of $\bar{S}_\perp$ is bounded polonomially in terms of the size of S and the number of states of $\mathcal{A}$, this solves an open problem in [58] in fact $size(\bar{S}_\perp) \leq size(\mathcal{A})^2 \cdot size(S_\perp)$.

Theorem 7.25. *Level-n OI-string languages are polynomially closed under intersection with regular languages.*

Proof. Let $G \in n\text{-}N\lambda(\Omega)$, where Ω is some monadic alphabet, and let $\mathcal{A}$ be a Rabin–Scott automata with k states and input-alphabet $\Omega^{(i,i)}$. By Corollary 7.3 there exists a Chomsky-normalform G' of G with $size(G') \leq p(size(G))$.

By Theorems 4.11, 6.7, 7.2 and Corollary 7.21 it suffices to prove

$$L(\bar{S}_\perp) = L(S_\perp) \sqcap L(\mathcal{Q})_+.$$

'$\subseteq$' Let $t \in L(\bar{S}_\perp)$. Then

$$1x_0\downarrow \overset{*}{\underset{\mathrm{OI},\bar{S}_\perp}{\Longrightarrow}} \quad \text{Corollary 6.6}$$

$$\curvearrowright\ x_0\downarrow \overset{*}{\underset{\mathrm{OI},\bar{S}_\perp}{\Longrightarrow}} t \wedge (\forall j \in \mathit{breadth}(t)$$

$$(\mathit{leave}(t) = e \ \curvearrowright\ \delta^*(\mathit{path}_j(t) \in F)))$$

Lemmas 7.22, 7.23

$$\curvearrowright\ t \in L(S_\perp) \wedge \mathit{path}(t) \subseteq L(\mathcal{Q})$$

$$\curvearrowright\ t \in L(S_\perp) \cap L(\mathcal{Q})_+ \subseteq L(S_\perp) \sqcap L(\mathcal{Q})_+ \quad \text{Lemma 7.19.}$$

'$\supseteq$' Let $s \in L(\mathcal{Q})_+$ where $s \sqsubseteq t$ for some $t \in L(S_\perp)$, and let $(t_0, t_1, \ldots, t_l)$ denote the full linearization of t.

Then $s = t_0[\tilde{t}_1, \ldots, \tilde{t}_l]$ where the $\tilde{t}_j$ satisfy

(1) $t_j = e \wedge \delta^*(\mathit{path}_j(t)) \notin F \ \curvearrowright\ t_j = \perp$
by definition of $L(\mathcal{Q})_+$ and Lemma 7.19;

(2) $t_j = e \wedge \delta^*(\mathit{path}_j(t)) \in F \ \curvearrowright\ \tilde{t}_j = \perp \vee \tilde{t}_j = e$
by definition of $L(\mathcal{Q})_+$ and Lemma 7.19;

(3) $t_j = \perp \ \curvearrowright\ \tilde{t} = \perp$ by definition of $\sqsubseteq$.

Hence by Lemma 7.24 $1x_0\downarrow \overset{*}{\Rightarrow}_{\bar{S}_\perp} s$, and thus $s \in L(\bar{S}_\perp)$. □

Remark. The construction can easily be generalized to cover arbitrary ranked alphabets Ω. In this case, R is represented by a (finite) *nondeterministic top–down automata* $\mathcal{Q}$ (with *transitions* $\delta_f(q) \ni (q_1, \ldots, q_k)$ for $f \in \Omega^{(i,k,i)}$ and, for each constant $a \in \Omega^{(e,i)}$, a set of *final states* F_a, cf. e.g. [23]).

The following modifications have to be made:

(1) $\sqsubseteq$ is replaced by the usual order $\leqslant$ on trees, and $\sqcap$ is again the meet (now with respect to $\leqslant$). Lemmas 7.20 and 7.21 hold for $\leqslant$ as well.

(2) In the construction of $\bar{S}$, (3.1) and (3.3) are replaced by

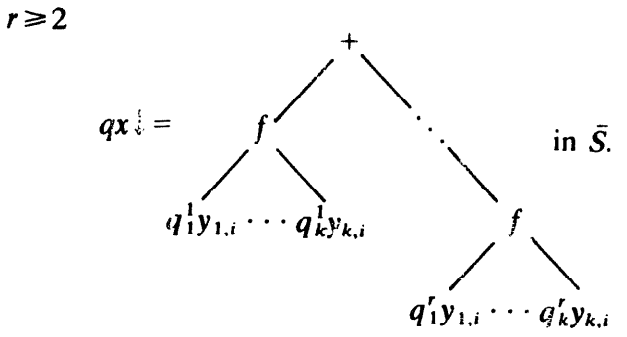
(3.1)' $x\downarrow = a(\ )$ in $S \ \curvearrowright\ \forall q \in F_a \quad qx\downarrow = a(\ )$ in $\bar{S}$

(3.3)' $x\downarrow = f(y_{1,i}, \ldots, y_{k,i})$ in S

$$\curvearrowright\ \forall q \in Q \quad \delta_f(q) = \{(q_1^1, \ldots, q_k^1), \ldots, (q_1^r, \ldots, q_k^r)\}$$

$$\curvearrowright\ r = 0 \quad qx\downarrow = \perp \text{ in } \bar{S}$$

$$r = 1 \quad qx\downarrow = f(q_1^1 y_{1,i}, \ldots, q_k^1 y_{k,i}) \text{ in } \bar{S}$$



Since $w \in L$ iff $L \cap \{w\} \neq \emptyset$, recursiveness of level-n OI-languages is now an immediate consequence of the decidability of the emptiness problem.

Corollary 7.26. $n\text{-}\mathscr{L}_{\mathrm{OI}} \subseteq \mathscr{REC}$.

We finally prove, that level-n OI-languages are closed under substitution and thus form a substitution closed AFL. This result is essentially a corollary to the closure of level-n trees under tree-homomorphism (cf. Theorem 4.9), since any substitution on languages can be lifted to a homomorphism on infinite trees.

The operation corresponding to substitution of languages viewed as monadic tree-languages is given in the following definition.

Definition 7.27. Let Ω, Σ denote monadic alphabets, and let $\sigma : \Omega \to \mathrm{PT}_\Sigma(Y)$ be a $D(\{i\})$-mapping s.t.

$$\sigma(e) = \{e\} \wedge (\forall a \in \Omega^1 \, \forall t \in \sigma(a) \; leave_1(t) = y_{1,i}).$$

The (*monadic*) *substitution induced by* σ,

$$\hat{\sigma} : \mathrm{PT}_\Omega \to \mathrm{PT}_\Sigma$$

is the canonical extension of σ:

$$\hat{\sigma}(e) = \{e\}, \quad \hat{\sigma}(aw) := \sigma(a) \leftarrow \hat{\sigma}(w), \quad \hat{\sigma}(L) := \bigcup_{t \in L} \hat{\sigma}(t)$$

Note that the restrictions on σ are crucial, since we want the substitution to completely 'process' all strings in a language L.

The next lemma shows, that substitution can be lifted to infinite trees.

Lemma 7.28. *Let* $\sigma : \Omega_+ \to \mathrm{CT}_{\Sigma_+(Y)}$ *be a* $D(\{i\})$*-mapping*

s.t. $\sigma(e) = e \wedge \sigma(+) = +\downarrow$

$$\wedge \forall a \in \Omega^1 \, \forall t \in set \circ \sigma(a) \quad leave_1(t) = y_{1,i}.$$

Then

$$\widehat{set \circ \sigma} \circ set = set \circ \hat{\sigma}. \qquad (*)$$

Proof. Since all mappings involved are continuous, the assertion can be proved inductively for finite trees t. The only critical step is $t = a(t')$.

- $t = a(t')$

$$\begin{aligned}
set(\hat{\sigma}(t)) &= set(\sigma(a) \leftarrow \hat{\sigma}(t')) \\
&= set(\sigma(a)) \leftarrow set(\hat{\sigma}(t')) && \text{Lemma 1.6} \\
&= set(\sigma(a)) \leftarrow \widehat{set \circ \sigma}(set(t)) && \text{I.H.} \\
&= \bigcup \{set(\sigma(a)) \leftarrow \widehat{set \circ \sigma}(s') \mid s' \in set(t')\} && (*) \\
&= \bigcup \{\widehat{set \circ \sigma}(a(s')) \mid s' \in set(t')\} \\
&= \widehat{set \circ \sigma}(a(set(t'))) = \widehat{set \circ \sigma}(set(a(t')))
\end{aligned}$$

We now prove (*).

'$\supseteq$' Immediate by monotonicity.

'$\subseteq$' Let $w \in set(\sigma(a)) \leftarrow set \circ \sigma(set(t'))$, then there exits $v \in set(\sigma(a))$ s.t. $w \in v \leftarrow set \circ \sigma(set(t'))$.

Note that $leave_1(v) = y_{1,i}$. A straightforward induction on v proves

$$\exists s' \in set(t') \quad w \in v \leftarrow \widehat{set \circ \sigma}(s'). \quad \square$$

Theorem 7.29. *Level-n OI string languages are closed under substitution.*

Proof. Let $\sigma' : \Omega \to \mathrm{PT}_\Sigma(Y)$ a $D(\{i\})$-mapping with

(1) $\sigma'(e) = \{e\}$,
(2) $\forall a \in \Omega^{(i,i)}\ \forall t \in \sigma'(a)\ leave_1(t) = y_{1,i}$,
(3) $\forall a \in \Omega^{(i,i)}\ \exists G^a \in n - IN\lambda(\Sigma(Y_i))\ \sigma'(a) = L(G^a)$

and $L = L(G) \in n - \mathscr{L}_{\mathrm{OI}}(\Omega)$. We must prove

$$\hat{\sigma}'(L) \in n - \mathscr{L}_{\mathrm{OI}}(\Sigma).$$

Define $\sigma : \Omega_+ \to \mathrm{CT}_{\Sigma_+(Y)}$ by

(1) $\sigma(e) = e$, $\sigma(+) = +(y_{1,i}, y_{2,i})$,
(2) $\forall a \in \Omega^{(i,i)}\ \sigma(a) := T(G^a_+)$.

Clearly $\sigma' = set \circ \sigma$. This implies

$$\begin{aligned}
\hat{\sigma}'(L) &= \hat{\sigma}'(set(T(G_+))) && \text{Corollary 4.11, Theorem 7.2} \\
&= set(\hat{\sigma}(T(G_+))) && \text{Lemma 7.28} \\
&= set(T(S)) && \text{for some } S \in n - \lambda(\Sigma_+) \quad \text{Theorems 4.9, 4.10} \\
&= L(set(S)) && \text{Corollary 4.11, Theorem 7.2.} \quad \square
\end{aligned}$$

Since any family of languages which contains $\mathcal{REG}$ and is closed under intersection with regular languages and substitution forms a full AFL (cf. e.g. [57, p. 129]), we have

Corollary 7.30. *Level-n OI-string languages form a substitution-closed AFL.*

8. IO

In the present chapter, we discuss the language families $n\text{-}\mathcal{L}_{IO}$ in the IO-hierarchy, which are generated from the language-diagram of the introduction by generalizing rightmost derivations in context-free grammars.

Having checked, that the sample languages can also be generated in IO-mode, we proceed to give a fixed-point characterization of level-n IO-languages. Together with the corresponding result for OI-languages, this generalizes the result in [28], that the difference between IO and OI can be explained

- *denotationally* by interpreting with respect to the powerset-algebra of the nth derived algebra and/or the nth derived algebra of the powerset-algebra of the Ω-tree algebra;
- *algebraically* by the order of application of *set* and $yield^{(n)}$.

As in the case of tree-transducers, the order of *choosing* and *copying* is crucial.

The characterization of level-n IO-languages as images under $yield^{(n)} \circ set$ of regular infinite trees and thus as images under $yield^{(n)}$ of regular tree-languages shows, that the IO-hierarchy coincides with the hierarchy defined in [42].

Contrary to the OI-case, the following results are most easily established using the combinatoric representation of IO-languages; the proof always depends on the possibility of 'lifting' the discussed property via *yield* to the level of regular tree-languages. As a simple example, the decidability of the emptiness problem for level-n IO-languages is a trivial consequence of this result for regular tree languages. Due to space restrictions, the section on paths and leaves for IO is omitted. While it can be shown, that front languages of level-n IO tree languages are level-$n+1$ IO string languages, the corresponding result for paths does not seem to hold (in fact, for the grammar $br(G)$ constructed in Section 7.4 we have $br(L_{IO}(G)) \subsetneq L_{IO}(br(G))$, cf. the example in Section 7.4), see [16]. We close the section by establishing closure under intersection with regular languages and homomorphism.

8.1. Example languages

In this section we prove, that, for $n \geq 2$, the language

$$L_{n+1} = \{a^{2^{\cdot^{\cdot^{\cdot^{2^k}}}}}_{n} \; b^{k+1} \; \cdots \; a^{2^{\cdot^{\cdot^{\cdot^{2^0}}}}}_{n} \; b(e) \mid k \in \omega\}$$

can be generated by rightmost derivations from a linear level-$n+1$ grammar G'_{n+1}.

Let us first consider an example.

Example. G_3' has productions

$$x_0 \to x_3(\lambda y_1.\lambda y_0.y_1(y_1(y_0)))(b)(e),$$

$$x_3 \to \lambda y_2.\lambda y_1.\lambda y_0.x_3(\lambda y_1.y_2(y_2(y_1)))$$
$$(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0))),$$

$$x_3 \to \lambda y_2.\lambda y_1.\lambda y_0.y_2(a)(y_1(y_0)).$$

We present a rightmost-derivation for $a^{16}b^3a^4b^2a^2b(e)$. The derivation leads first to three textual substitutions of the body of x_3 for x_3. Then the resulting λ-expression over parameters and terminals is reduced to its normalform.

$$\underset{\downarrow}{x_0} \to x_3(\lambda y_1.\lambda y_0.y_1^2(y_0))(b)(e)$$

$$\to \lambda_2.\lambda y_1.\lambda y_0.$$
$$\underset{\downarrow}{x_3}(\lambda y_1.y_2^2(y_1))(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$
$$(\lambda y_1.\lambda y_0.y_1^2(y_0))(b)(e)$$

$$\to \lambda y_2.\lambda y_1.\lambda y_0.$$
$$\lambda y_2.\lambda y_1.\lambda y_0.$$
$$\underset{\downarrow}{x_3}(\lambda y_1.y_2^2(y_1))(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$
$$(\lambda y_1.y_2^2(y_1))(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$
$$(\lambda y_1.\lambda y_0.y_1^2(y_0))(b)(e)$$

$$\to t \in \hat{G}'^3(x_0)$$

where $t = \lambda y_2.\lambda y_1.\lambda y_0.$
$$\lambda y_2.\lambda y_1.\lambda y_0.$$
$$\underset{\downarrow}{\lambda y_2.\lambda y_1.\lambda y_0.}$$
$$y_2(a)(y_1(y_0))$$
$$(\underline{\lambda y_1.y_2^2(y_1)})(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$
$$(\lambda y_1.y_2^2(y_1))(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$
$$(\lambda y_1.\lambda y_0.y_1^2(y_0))(b)(e)$$

$$\underset{\beta}{\to} \lambda y_2.\lambda y_1.\lambda y_0.$$

$$\lambda y_2.\lambda y_1.\lambda y_0.$$

$$\lambda y_1.\lambda y_0.$$

$$\downarrow$$

$$\lambda v_1.y_2^2(y_1)(a)(y_1(y_0))$$

$$(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$

$$(\lambda y_1.y_2^2(y_1))(\lambda y_0.b(y_1(y_0)))(y_2(a)(y_1(y_0)))$$

$$(\lambda y_1.\lambda y_0.y_1^2(y_0))(b)(e)$$

$$\vdots$$

$$\underset{\beta}{\rightarrow} nf(t) = a^{16}b^2a^4b^2a^2b(e).$$

Lemma 8.1. $\forall n \geq 3\ L_n \in n - \mathrm{CF}_{\mathrm{IO}}(\{a, b, e\})$.

Proof. Denote by G'_n the level-n grammar obtained from G_n by substituting $G_n(x_1), \ldots, G_n(x_{n-1}), G_n(A), G_n(B)$ for $x_1, \ldots, x_{n-1}, A, B$ in the right-hand sides of x_0 and x_4 and taking normalforms. Clearly $L(G'_n) = L(G_n)$, hence by Lemma 7.1 $L(G'_n) = L_n$.

Any rightmost derivation in G'_n can be split into a sequence of textual substitutions of $G'_n(x_n)$ for x_n, followed by a sequence of β-reductions which takes the resulting term over Ω and Y into its normalform; thus any string generated in this way is an element of the kth set in the Kleene sequence of G'_n for some $k \in \omega$. Moreover, by Fact 5.4, any string in $K(G'_n)(k)$ can be obtained in this way, hence $L_{\mathrm{IO}}(G'_n) = \bigcup K(G'_n)$, and thus by Theorems 5.9 and 7.2 $L_{\mathrm{IO}}(G'_n) = L(G'_n) = L_n$. □

8.2. Fixed-point characterization

A fixed-point characterization of IO-languages requires the definition of a higher-type functional $\mathcal{G}$ over the domain of tree languages for each level-n grammar G, such that (the first component of) the least fixed-point of $\mathcal{G}$ coincides with the IO-language generated by G. In the (context-free) case $n = 1$ this was achieved in [28] by considering different notions of substitutions of tree languages (and thus in fact defining different $D(\Omega)_+$-structures over PT_Ω). The IO-semantics of a nondeterministic recursive program scheme can then be defined canonically by using IO-substitution as denotation of functional composition. The generalization of this approach to higher type recursion is not straightforward (see [34]). Instead, we will lift the fixed-point characterization of $L_{\mathrm{IO}}(G)$ via *yield* to a regular grammar $c^{(n)} \circ G$ over $D^n(\Omega)$ in $n - \mathrm{NR}(\Omega)$, whose IO-semantics can easily be defined denotationally as interpretation over $\mathcal{P}D^n(\mathcal{T}_\Omega)$ and $D^n(\mathcal{P}\mathcal{T}_\Omega)$, respectively.

In a first step, we will thus show, that the IO-language generated by a level-n grammar G coincides with the $yield^{(n)}$-image of the regular tree language generated

by $c^{(n)} \circ G$. This can essentially be done by induction on the level of G. We will only indicate a possible proof of this result, since the rest of the paper will only rely on the resulting characterization of the IO-hierarchy as homomorphic images of regular tree-languages.

Theorem 8.2. $n+1-\mathscr{L}_{IO}(\Omega) = yield(n-\mathscr{L}_{IO}(D(\Omega))^{(e,i)})$.

Proof. '$\supseteq$' Let $G \in n-N\lambda(D(\Omega))^{(e,i)}$, then $\gamma_\lambda \circ G \in n+1-N\lambda(\Omega)$.

The proof of $\gamma(L_{IO}(G)) = L_{IO}(\gamma_\lambda \circ G)$ is based on the following assertions:

Assertion 1. $\gamma_\lambda(t)[y_\alpha/\gamma_\lambda(s)] = \gamma_\lambda(t[y_\alpha/s])$.

Assertion 2. $t \to_G t' \curvearrowright \gamma_\lambda(t) \to_{\gamma_\lambda \circ G} \gamma_\lambda(t')$.

Assertion 3. Call a *k-redex* any term of the form $\gamma_\lambda(k)(t_1, \ldots, t_r)$ where $k \in D(\Omega)\backslash\Omega'$.

If $\gamma_\lambda(t) \to_{\gamma_\lambda \circ G} s$ is not a k-reduction, then there exists t' s.t. $\gamma_\lambda(t') = s \wedge t \to_G t'$.

Assertion 4. $\forall t \in T_{D(\Omega)}\, \gamma(t) \leftarrow s = nf(\gamma_\lambda(t)(s))$.

The proof of these assertions is by induction on t.

Now let $t \in L_{IO}(G)$, and assume, that $x_0\downarrow \to t_1 \to \cdots \to t_n = t$ is a rightmost derivation in G, then

$$x_0\downarrow \to \gamma_\lambda(t_1)(\ ) \to \cdots \to \gamma_\lambda(t_n)(\ ) \overset{*}{\underset{\beta}{\to}} nf(\gamma_\lambda(t_n)(\ )) = yield(t)$$

is a derivation in $\gamma_\lambda \circ G$, s.t. $x_0\downarrow \overset{*}{\to} \gamma_\lambda(t_n)(\ )$ is a right-standard derivation, in which only k-reduces are omitted.

The only critical case in the simulation of G's IO-derivations in $\gamma_\lambda \circ G$ arises in the treatment of expressions of the form

$$\gamma_\lambda(sub^{v}_{(w,i)})(F_0, \ldots, F_r)(A_1, \ldots, A_p).$$

Here, it is essential, that all arguments $A_1, \ldots, A_p$ are expanded to terminal terms (and hence all right-hand sides to be substituted are chosen) before the evaluation of $\gamma_\lambda(sub^{v}_{(w,i)})$ leads to copying of arguments. Thus, in order to compute $yield(t)$, any strategy can be chosen which eliminates all nonterminals in argument positions of a k-redex before reducing this redex. In particular, $x_0\downarrow \to^*_{\gamma_\lambda \circ G} yield(t)$ is equivalent to a rightmost derivation.

To prove the opposite inclusion, we observe that any rightmost derivation in $\gamma_\lambda \circ G$ leading to some $t \in L_{IO}(\gamma_\lambda \circ G)$ is equivalent to a reduction sequence

$$x_0\downarrow \to s_1(\ ) \to \cdots \to s_n(\ ) \xrightarrow[k\text{-reductions}]{*} s'_n(\ ) \to t$$

s.t.

(1) $x_0\downarrow \to^* s_n(\ )$ is a right-standard derivation, which omits only k-redexes,
(2) $x_0\downarrow \to^* s_n(\ )$ contains no k-reductions,
(3) $s_n \in T_{\gamma_\lambda(D(\Omega))}$.
By Assertion 3 this implies the existence of a rightmost derivation

$$x_0\downarrow \to t_1 \to \cdots \to t_n$$

in G with $\gamma_\lambda(t_n) = s_n$, hence by Assertion 4 $t = yield(t_n)$.

'$\subseteq$' Let $G \in n+1-N\lambda(\Omega)$, and assume w.l.o.g. $level(x) \geq 1$ for all $x \in var(G)$. Then $c(G)$ (as constructed in Corollary A.7) is in $n - N\lambda(D(\Omega))^{(e,i)}$. Similar to the proof of Lemma A.6 one can prove $\gamma_\lambda(c_e(t)) \to_\beta^* t$, hence by the proof of '$\supseteq$' $L_{IO}(G) = L_{IO}(\gamma_\lambda(c(G))) = yield(L_{IO}(c(G)))$. □

Since $O-N\lambda(D^n(\Omega))$ coincides with the class of regular tree grammars over $D^n(\Omega)$, we have as an immediate corollary a characterization of the language families in the IO-hierarchy as homomorphic images of regular languages.

Corollary 8.3. $n - \mathscr{L}_{IO}(\Omega) = yield^{(n)}(\mathscr{RLG}(D^n(\Omega))^{b_n(i)}$.

To obtain the fixed-point characterization it remains to prove that the translation of a regular language under $yield^{(n)}$ coincides with the denotational semantics of the generating grammar over $\mathscr{P}D^n(\mathscr{T}_\Omega)$. This is a special case of the Mezei–Wright-like result for 'IO-equational sets' as proved in [28].

Since the initiality arguments in the proof of Theorem 4.6 carry over straightforwardly to the $\bigsqcup$-continuous case, we will omit the proof.

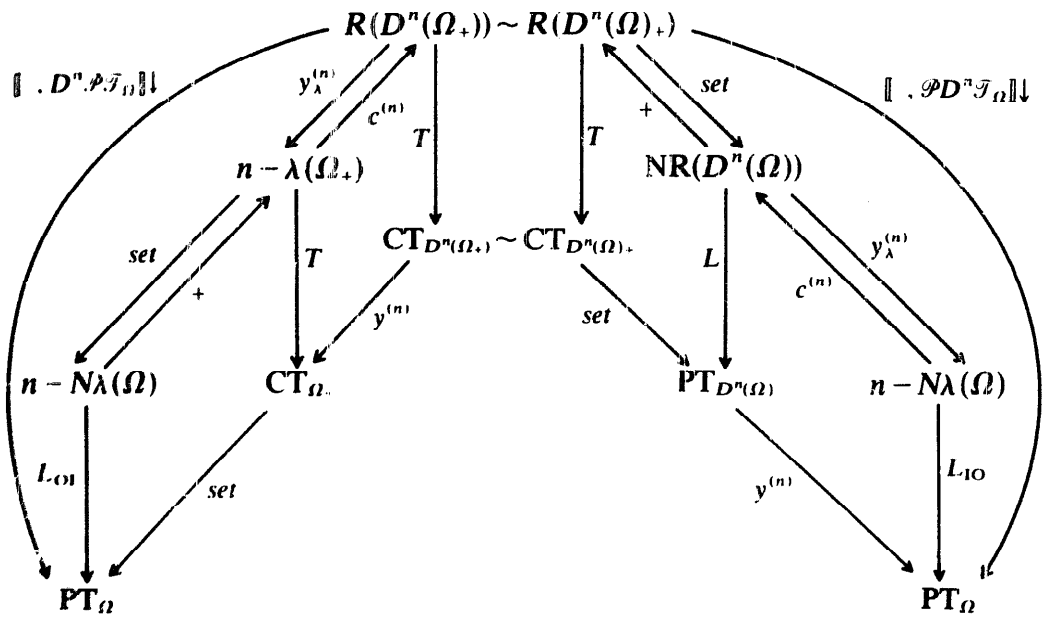
Theorem 8.4 ([28]).

$$\forall G \in n - NR(\Omega)\ [\![G_+, \mathscr{P}D^n(\mathscr{T}_\Omega)]\!]\downarrow_{T_\Omega} = yield^{(n)}(L(G)).$$

As an immediate corollary to Corollary 8.3 and Theorem 8.4 we obtain the fixed-point characterization of level-n IO languages.

Corollary 8.5. $n - \mathscr{L}_{IO}(\Omega) = \{[\![G_+, \mathscr{P}D^n(\mathscr{T}_\Omega)]\!]\downarrow_{T_\Omega} \mid G \in n - \mathrm{NR}(\Omega)\}$.

In the following (commutative) diagram we summarize the operational, denotational, and algebraic characterizations of level-n IO- and OI-languages. All types should be clear from the context and are omitted to increase readability. The correspondence between regular equations over $D^n(\Omega_+)$ and $D^n(\Omega)_+$ is based on the idea, that the nondeterminism can be pushed down to the lowest level, since the join operation in the function space is defined pointwise using the join-operation at the lower level. For a formal proof, see [15].



8.3. Decidability of the emptiness-problem

As mentioned before, the decidability of the emptiness problem is an immediate consequence of the characterization of IO-languages as homomorphic images of regular tree-languages (and the fact, that emptiness is decidable for $L \in \mathcal{REG}$).

Corollary 8.6. *The emptiness-problem for level-n IO languages is decidable.*

Proof.

$$yield^{(n)}(L(G)) = \emptyset \text{ iff } L(G) = \emptyset \text{ for any } G \in n-\mathrm{NR}(\Omega). \quad \square$$

We now use the decision procedure to derive an upper bound for the depth of a tree in a nonempty level-n IO-language.

Theorem 8.7. *Let $G \in n-\mathrm{NR}(\Omega)$ and $L = yield^{(n)}(L(G))$.*
Then

$$L \neq \emptyset \quad \textit{iff} \quad \exists t \in L \quad depth(t) \leq \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{n} 2^{size(G)}$$

Proof. *Let*

$$d := max\{depth(s) \mid s \in rhs(G)\}, \qquad r := |rhs(G)|.$$

By the proof of the pumping-lemma for regular tree languages (see e.g. [23]) we have

$$L(G) \neq \emptyset \rightsquigarrow \exists t \in L(G)\ depth(t) \leq d \cdot r \leq size(G)$$

(otherwise some nonterminal has been called recursively and we could derive a shorter tree using 'cut and paste'). Lemma 7.10 applied to such a tree t yields the assertion of the theorem. □

8.4. Closure properties

In this section we will demonstrate the power of the combinatoric characterization of IO-languages by giving easy algebraic proofs of closure under tree-homomorphisms and intersection with regular languages. The techniques used to lift these properties under *yield* are taken from [28].

Let us first recall the definition of recognizable tree languages in the many-sorted case.

Definition 8.8. Let Ω denote a finite $D(I)$-set.

$L \subseteq T^i_\Omega$ is *recognizable* iff

$$\exists \mathcal{Q} \in alg\,\Omega\ \exists F \subseteq Q^i\ \ size(Q) := \left|\bigcup_{i \in I} Q^i\right| \in \omega \wedge L = L_F(\mathcal{Q}) := h_{\mathcal{Q}}^{-1}(F).$$

For an infinite Ω, $L \subseteq T^i_\Omega$ is *recognizable* iff L is recognizable as a subset of $T^i_{\Omega'}$ for some finite $\Omega' \subseteq \Omega$.

It should be clear, that this definition describes algebraically the recognition of tree languages by a deterministic bottom–up tree automaton with final states F (as defined e.g. in [45, 23]).

Clearly the intersection of the *yield*-translation of a language L with a set R can be simulated at the level of L by taking the intersection with $yield^{-1}(R)$.

Lemma 8.9. *Let $L \subseteq T_{D(\Omega)}$, $R \subseteq T_\Omega$.*

Then $yield(L) \cap R = yield(L \cap yield^{-1}(R))$.

The next lemma shows, that the inverse image under *yield* of a recognizable tree language R is recognized by a finite subalgebra of the derived algebra of the automaton accepting R.

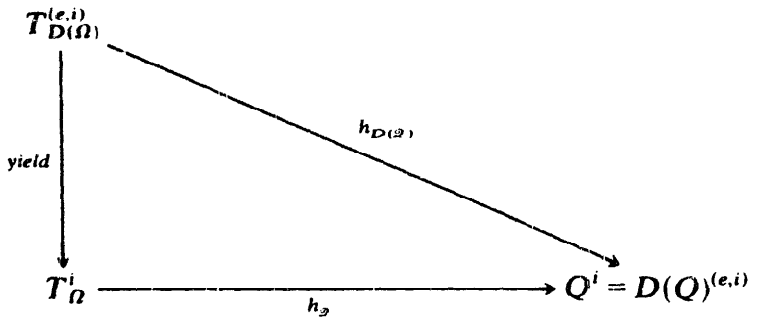
Lemma 8.10. *Let Ω denote a finite $D(I)$-set, and let $R = L_F(\mathcal{Q})$ for some finite automaton $\mathcal{Q} \in alg\,\Omega$ with final states $F \subseteq Q^i$.*

Then for all finite $D' \subseteq D(\Omega)$ there exists a finite subalgebra $\mathcal{Q}' \subseteq D(\mathcal{Q})$ s.t.

(1) *$size(\mathcal{Q}') \leqslant 2^{p(size(\mathcal{Q}))}$ for a polynomial p,*

(2) *$yield^{-1}(R) \cap T^{(e,i)}_{D'} = L_F(\mathcal{Q}')$.*

Proof. (2) By Lemma 4.4



commutes.

Hence we have for $\mathcal{Q}' := h_{D(\mathcal{Q})}(\mathcal{T}_{D'})$,

$$\begin{aligned} L_F(\mathcal{Q}') &= h_{\mathcal{Q}'}^{-1}(F) = h_{D(\mathcal{Q})}^{-1}(F) \cap T_{D'}^{(e,i)} \\ &= yield^{-1}(h_{\mathcal{Q}}^{-1}(F)) \cap T_{D'}^{(e,i)} \\ &= yield^{-1}(R) \cap T_{D'}^{(e,i)}. \end{aligned}$$

(1) Let $D' = D'^{(\alpha_1,\nu_1)} \cup \cdots \cup D'^{(\alpha_m,\nu_m)}$ and $r = max\{1(\alpha_j) \mid j \in [m]\}$. Then

$$\begin{aligned} size(\mathcal{Q}') &\leq |D(Q)^{(\alpha_1,\nu_1)}| + \cdots + |D(Q)^{(\alpha_m,\nu_m)}| \\ &\leq m \cdot size(\mathcal{Q})^{r \cdot size(Q)} \\ &= 2^{ld(m \cdot size(\mathcal{Q})) \cdot r \cdot size(\mathcal{Q})} \\ &\leq 2^{\alpha \cdot size(\mathcal{Q})^2} \quad \text{for some } \alpha \in \mathbb{R}_+. \quad \square \end{aligned}$$

By the above, the intersection of a level-n IO tree language with a regular language can be simulated at the $D^n(\Omega)$-level by intersection of regular tree languages. Since regular tree languages are closed under intersection we have thus proved

Theorem 8.11. *$n - \mathcal{L}_{IO}$ is n-exponentially closed under intersection with regular tree languages.*

Proof. Let $L = yield^{(n)}(L(G))$ for some $G \in n - NR(\Omega)$ and $R = L_F(\mathcal{Q})$ for $\mathcal{Q} \in alg\,\Omega$ with $size(\mathcal{Q}) \in \omega$.

Choose finite subalphabets $D'_m \subseteq D^m(\Omega)$ s.t.

$$\underbrace{yield \circ \cdots \circ yield}_{m}(L(G)) \subseteq T_{D'_m}.$$

Using Lemma 8.10 a straightforward induction on m shows:

The language $R_m \subseteq T_{D'_m}$ (where $R_0 := R$, $R_{k+1} := yield^{-1}(R_k) \cap T_{D'_{k+1}}$) is recognizable by a deterministic bottom-up automaton $\mathcal{Q}_m$ with

$$size(\mathcal{Q}_m) \leq \overbrace{2^{\cdot^{\cdot^{\cdot^{2^{p_m(size(\mathcal{Q}))}}}}}}^{m} \quad \text{for a polynominal } p_m.$$

Since

$$T_{D'_m} \cap \underbrace{yield \circ \cdots \circ yield}_{m}(L(G)) = \underbrace{yield \circ \cdots \circ yield}_{m}(L(G))$$

we have

$$yield^{(n)}(L(G)) \cap R$$

$$= yield^{(n)}(L(G) \cap R_n) \quad \text{by Lemma 8.9}$$

$$= yield^{(n)}(L(G')) \quad \text{by Theorem 7.25}$$

for some $G' \in n - NR(\Omega)$ s.t.

$$size(G') \leq p'(size(G), size(\mathcal{Q}_n))$$

for some polynomial p' and thus

$$size(G') \leq p'(size(G), \overbrace{2^{\cdot^{\cdot^{\cdot^{2^{p_n(size(\mathcal{Q}))}}}}}}^{n}) \leq \overbrace{2^{\cdot^{\cdot^{\cdot^{2^{p(size(G), size(\mathcal{Q}))}}}}}}^{n}$$

for a suitable polynomial p. □

Since $w \in L$ iff $L \cap \{w\} \neq \emptyset$ we obtain as an immediate corollary of Corollary 8.6 and Theorem 8.11 recursiveness of level-n IO languages.

Corollary 8.12. $n - \mathcal{L}_{IO} \subseteq \mathcal{REC}$.

We will now prove closure of IO-languages under tree homomorphisms. Since we already proved in Section 4, that three homomorphisms can be lifted via *yield* to linear 'second-level' tree-homomorphisms, this result is a direct consequence of closure of regular tree languages under linear tree-homomorphisms.

Theorem 8.13. *For $n > 0$, $n - \mathcal{L}_{IO}$ is closed under tree homomorphisms.*

Proof. Consider $L = yield^{(n)}(L(G))$ with $G \in n - \mathrm{NR}(\Omega)$ and a $D(I)$-mapping $\sigma: \Omega \to T_\Sigma(Y)$.

For $0 < m \leq n$, define $D^m(I)$-mappings

$$m - \sigma: D^m(\Omega) \to T_{D^m(\Sigma)}(Y)$$

by

$$f^{\overbrace{\cdots}^{m}} \mapsto \underbrace{comb \circ \cdots \circ comb}_{m}(\lambda y_w.\sigma(f)) \quad \text{for } f \in \Omega^{(w,i)},$$

$$s \mapsto s\downarrow \quad \text{for } s \in D^m(\Omega) \backslash \Omega^{\overbrace{\cdots}^{m}}.$$

By a trivial induction on t $yield \circ comb(t) = t$, thus the $m - \sigma$ meet the assumptions in Lemma 4.8, hence

$$\hat{\sigma}(yield^{(n)}(L(G))) = yield^{(n)}(n - \hat{\sigma}(L(G))) = yield^{(n)}(L(G'))$$

for some $G' \in n - \mathrm{NR}(\Omega)$, since $n - \hat{\sigma}$ is linear and $\mathcal{RCG}$ is closed under linear tree-homomorphisms (take $G' = n - \hat{\sigma} \circ G$). □

We note, that the techniques of this paper are not sufficient to establish closure of level-n IO-tree languages under deterministic bottom–up tree transducers (as conjectured in [28]). First, the construction used in [28] to lift such transducers under *yield* cannot be iterated, essentially, since several initial states are needed at the higher level. Moreover, basically new phenomena occur when passing from IO-context-free to level-2 IO tree languages (in the λ-calculus formulation), since at level 2 nonterminals may have to be expanded before all actual parameters are given. Hence, in a situation where x defines some terminal $f \in \Omega^{((ii,i),(ii,i))}$ and x is called in a right-hand side $x'(x)(a, b)$ (with type $x' = ((ii, i), (ii, i))$) in an IO-derivation we are forced to evaluate x before knowing which translation of f we are to choose, since we do not yet have any information regarding the state of possible arguments for f. In fact, x' might copy x to different positions, where different translations of f may be needed. The 'guessing technique' of Section 7.5 does not work, since we are to simulate bottom–up translations.

9. Hierarchies

In this chapter, we will prove the IO- and OI-hierarchies to be infinite and derive sufficient conditions on a language family $\mathcal{R}$ to establish strictness of hierarchies of the form $yield^{(n)}(\mathcal{R})$. Examples of language families satisfying these conditions include deterministic top–down translations of regular tree languages and the language families in the OI-hierarchy. In particular, this proves strictness of the IO-hierarchy. Moreover, we prove that the concept of recursion on *level-n* leads to a strict hierarchy of program schemes.

As a tool to capture the increase in copying power inherent in both hierarchies we use the *rational index*, a complexity measure for languages introduced by Boasson, Courcelle, and Nivat [6], which, roughly speaking, measures the density of regular m-state components of a given language L as m varies. More precisely, the rational index of L associates with a given number m of states the maximum of the distances between L and all m-state languages, where the distance of two languages is the length of the shortest word in their intersection.

Definition 9.1. The *distance* of two languages L_1 and L_2 with nonempty intersection is given by

$$d(L_1, L_2) = min\{depth(w) \mid w \in L_1 \cap L_2\}.$$

Let $\mathcal{RECG}_m$ denote the class of languages recognized by Rabin/Scott automata with m states. The *rational index of* L, $rat(L)$, is the function

$$rat(L): \omega \to \mathbb{R}_+$$
$$m \mapsto max\{d(R, L) \mid R \cap L \neq \emptyset \wedge R \in \mathcal{RECG}_m\}.$$

We note, that an approach to the hierarchy question along density arguments would require the proof of (some sort of) pumping lemmata as e.g. in [33], which seem to be extremely hard to prove. On the other hand, to establish upper bounds for the rational index of a language in some family $\mathcal{L}$, we have to prove closure of $\mathcal{L}$ under intersection with regular languages (within certain size-restrictions) and an assertion corresponding to the 'downward pumping' part of the pumping lemma, i.e. a bound on the length of the shortest word in a nonempty language in terms of the size of e.g. the grammar generating this language. Hence the results of the previous sections immediately give upper bounds for the rational index of level-n languages.

Notation. Let $\mathcal{F}$ denote the class of functions $\omega \to \mathbb{R}_+$.

For $f, g \in \mathcal{F}$ define $f \leqslant g$ iff $\exists m_0 \in \omega\ \forall m \geqslant m_0\ f(m) \leqslant g(m)$.

We denote by $\mathcal{L}(\mathcal{F}')$ the class of languages whose rational index is majorized by some function in $\mathcal{F}' \subseteq \mathcal{F}$.

Finally,

$$\mathcal{EXP}^n(\mathcal{POL}) := \{m \mapsto \overbrace{2^{\cdot^{\cdot^{\cdot 2}}}}^{n}{}^{\alpha \cdot m^k} \mid \alpha \in \mathbb{R}_+, k \in \omega\}.$$

Theorem 9.2. $n\text{-}\mathcal{L}_{IO} \subseteq \mathcal{L}(\mathcal{EXP}^{2n}(\mathcal{POL}))$.

Proof. Let $L = yield^{(n)}(L(G))$ for some $G \in n\text{-NR}(\Omega)$, and let $R \in \mathcal{RECG}_m(\Omega)$ with $L \cap R \neq \emptyset$. By Theorem 8.11 there exists an n-exponential bounded $G' \in n\text{-NR}(\Omega)$ with $yield^{(n)}(L(G')) = L \cap R$, hence by Theorem 8.7

$$\exists t \in L\ \ depth(t) \leqslant \overbrace{2^{\cdot^{\cdot^{\cdot 2}}}}^{n}{}^{size(G')} \leqslant \overbrace{2^{\cdot^{\cdot^{\cdot 2}}}}^{n}{}^{\overbrace{2^{\cdot^{\cdot^{\cdot 2}}}}^{n}{}^{p(size(G'),m)}}$$

for some $p \in \mathcal{POL}$, and thus

$$rat(L) \leqslant m \mapsto \overbrace{2^{\cdot^{\cdot^{\cdot 2}}}}^{2n}{}^{p'(m)} \in \mathcal{EXP}^{2n}(\mathcal{POL}). \quad \square$$

Theorem 9.3. $n\text{-}\mathcal{L}_{OI} \subseteq \mathcal{L}(\mathcal{EXP}^{2n}(\mathcal{POL}))$.

Proof. Let $L = L(G)$ for some $G \in n-N\lambda(\Omega)$, and let $R \in \mathscr{RES}_m(\Omega)$ s.t. $L \cap R \neq \emptyset$. By Theorem 7.25 there exists a polynomially bounded $G' \in n-N\lambda(\Omega)$ with $L(G') = L(G) \cap R$, hence by Theorem 7.11

$$\exists t \in L \ \ depth(t) \leq \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{2n} 2^{p_1(size(G'))} \leq \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{2n} 2^{p_1(p_2(size(G),m))}$$

for some $p_1, p_2 \in \mathscr{POL}$, and thus

$$ra(L) \leq m \mapsto \overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{2n} 2^{p(m)} \in \mathscr{EXP}^{2n}(\mathscr{POL}). \quad \square$$

In [61], Steyaert gives an example of a $\mathscr{DOL}(\subseteq \mathscr{MAC})$ language, whose rational index grows at least double-exponentially. Using techniques of this paper, we will generalize this result to higher levels by proving that the rational index of the sample language L_n grows at least n-exponentially. To this end, we consider intersections of

$$L_n = \{a^{\overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{n-1} 2^k} b^{k+1} \cdots a^{\overbrace{2^{\cdot^{\cdot^{\cdot}}}}^{n-1} 2^0} b(e) \mid k \in \omega\}$$

with the regular languages

$$R_p = a^+(b^p)^+a^+(b^{p+1})^+ \cdots a^+(b^{2p-1})^+(a \vee b)^*$$

(for $n = 3$, $p = 2$ an OI-grammar generating the intersection was given in the example in Section 7.5). It is easy to see, that R_p can be recognized deterministically with $\frac{3}{2}(p^2+p)$ states.

Consider some string $w \in L_n \cap R_{p+1}$, then the exponent k corresponding to w must satisfy

$k+1$	is a multiple of $p+1$,
k	is a multiple of $p+2$,
	$\vdots$
$k-p+1$	is a multiple of $2p+1$.

To evaluate the exponent k_{p+1} for the shortest word w_{min} in $L_n \cap R_{p+1}$ we need the following lemmata.

Lemma 9.4. *Let $k, p_1, \ldots, p_k \in \omega$, then*

$$min\{m \in \omega \mid m \geq 1 \wedge \forall i \in [k]\ p_i \text{ divides } m - p_i\} = lcm(p_1, \ldots, p_k).$$

Proof. It is obvious, that the least common multiple of $p_1, \ldots, p_k$ satisfies $\varphi(m) \equiv (\forall i \in [k]\ p_i$ divides $m - p_i \wedge m \geq 1)$. To prove minimality, consider some m s.t. $\varphi(m)$ holds and assume $m < lcm(p_1, \ldots, p_k)$, but then

$$\forall i \in [k]\ ((p_i \text{ divides } m - p_i) \wedge (p_i \text{ divides } lcm(p_1, \ldots, p_k) - p_i)).$$

and thus $\forall i \in [k]\, p_i$ divides $lcm(p_1, \ldots, p_k) - m$, contradicting the definition of *lcm*. □

Hence the exponent of $w_{min} \in L_n \cap R_{p+1}$ satisfies

$$k_{p+1} = lcm(p+1, p+2, \ldots, 2p+1) - (p+2).$$

Lemma 9.5.

$$\forall p \geq 1\ \forall m \in \omega\ \frac{1}{m}! \cdot \prod_{0 \leq i \leq m} (p+1) \leq lcm(p, p+1, \ldots, p+m).$$

Proof. Let

$$P := \prod_{0 \leq i \leq m} (p+i) \quad \text{and} \quad g := gcd\left(\frac{P}{p}, \ldots, \frac{P}{p+m}\right).$$

It is easy to see, that

$$P = g \cdot lcm(p, p+1, \ldots, p+m).$$

For the proof of the lemma it is thus sufficient to show $g \leq m!$. To this end, consider for $i \in \{0, \ldots, m-1\}$ and $r \in [m-1]$ the expressions D_r^i defined by

$$D_1^i := \frac{P}{p+1} - \frac{P}{p+i+1}, \qquad D_{r+1}^i := D_r^i - D_r^{i+1}.$$

By definition of g, g divides all D_r^i. Moreover, a straightforward induction on r proves

$$D_r^i = \frac{r! \cdot P}{(p+i) \cdot \cdots \cdot (p+i+r)}.$$

In particular, $D_m^0 = m!$, hence $g \leq m!$. □

Theorem 9.6. $\forall n \geq 3\ L_n \notin \mathcal{L}(\mathcal{EXP}^{n-1}(\mathcal{POL}))$.

Proof. A straightforward induction on $p \geq 3$ shows

$$2^{p+1} < \frac{(2p+1)!}{(p!)^2} - (p+2). \quad (*)$$

Thus, for $p \geq 3$, the exponent of the shortest string w_{min} in $L_n \cap R_{p+1}$ satisfies

$$\begin{aligned} k_{p+1} &= lcm(p+1, p+2, \ldots, 2p+1) - (p+2) && \text{Lemma 9.4} \\ &\geq \frac{(p+1)(p+2) \cdot \cdots \cdot (2p+1)}{p!} - (p+2) && \text{Lemma 9.5} \\ &= \frac{(2p+1)!}{(p!)^2} - (p+2) > 2^{p+1}. && (*) \end{aligned}$$

Since $R_p \in \mathcal{REG}_{3(p^2+p)/2}$ this implies

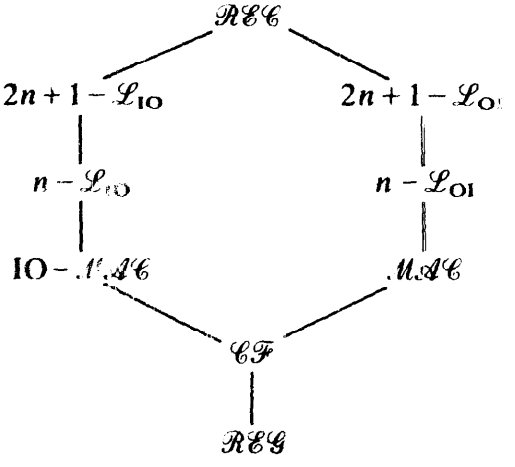
$$rat(L_n)(\tfrac{3}{2}(p^2+p)) \geq d(L_n, R_p) > 2^{\cdot^{\cdot^{\cdot^{2^{2^p}}}}} \quad (n-1 \text{ twos})$$

Since $m \mapsto 2^{\sqrt{2/3} \cdot m^{\sqrt{2}-1/2}}$ cannot be majorized by any polynomial $m \mapsto \alpha \cdot m^k$ this shows

$$rat(L_n) > \mathcal{EXP}^{n-1}(\mathcal{POL}). \quad \square$$

Since the rational index of level-n languages is $2n$-exponentially bounded, L_{2n+1} cannot be generated at the nth level, hence both hierarchies are infinite. Let us recall some characterizations of level-n languages to interpret this result. Mathematically this shows, that taking fixed-points on increasingly higher level function spaces allows to define more and more base objects. Operationally, the possibility of 'storing information' in more and more levels of parameters gives a strict increase in copying power. This is even more explicit in the characterization of OI languages by level-n pushdown automata [44] proved in [21]: the possibility of pushing pushdown-lists, pushdown-lists of pushdown-lists ..., etc. gives you increasingly powerful storage-structures. We summarize the relation of the language classes studied in this paper in the following diagram, where all inclusions are strict for a monadic alphabet with at least two unary operation symbols (and $n \geq 3$).

9.7. Hierarchytheorem.



Proof. Lemma 7.1, Theorem 7.8, Corollary 7.18, Corollary 7.26, Lemma 8.1, Corollary 8.6, Corollary 8.12, Theorem 9.2, Theorem 9.3, Theorem 9.6.

That the IO-hierarchy starts with the regular, context-free, and IO-macro languages was proved in [28, Theorem 7.9]. $\square$

In the OI-case, the fixed-point characterization gives a link between level-n languages and level-n schemes and thus (via the interpretation *Algol*) to semantics of ALGOL 68 programs with finite modes. Hence the hierarchy theorem implies, that ALGOL 68 programs with mode depth $2n+1$ cannot in general be simulated by *formally equivalent* programs (in the sense of [38]) with mode depth n. In fact, due to the uniformity of the definition of level-n schemes (as expressed in the results of the appendix) we can prove the sharper result, that level-$n+1$ schemes cannot be translated to level-n schemes, hence the concept of recursion on higher types induces a strict hierarchy of control-structures. In the theorem below we include the result proved in [17] that the class λ of *λ-schemes* (modelling untyped procedure calls) is more powerful than the class of level-n schemes.

Theorem 9.8. $\forall n \geq 2\ R < \mathrm{RPS} < n-\lambda < n+1-\lambda < \lambda$.

Proof. By Corollary A.3 and Corollary A.7 we have for any $D(I)$-set Ω

(1) $\quad n-\lambda(D(\Omega))^{(e,i)} \sim n+1-\lambda(\Omega)^{i}$.

We now show

(2) $\quad n-\lambda \sim n+1-\lambda \curvearrowright n+1-\lambda \sim n+2-\lambda$.

Let $S \in n+2-\lambda(\Omega)$ for some Ω, then by (1) there exists $S_1 \in n+1-\lambda(D(\Omega))^{(e,i)}$ s.t. $S \sim S_1$. From the assumption in (2) we have in particular $n-\lambda(D(\Omega))^{(e,i)} \sim n+1-\lambda(D(\Omega))^{(e,i)}$, thus there exists $S_2 \in n-\lambda(D(\Omega))^{(e,i)}$ with $S \sim S_1 \sim S_2$. Again applying (1) yields an $S' \in n+1-\lambda(\Omega)$ satisfying $S \sim S'$. To establish

(3) $\quad n-\lambda < n+1-\lambda$

we note that Theorems 7.2 and 9.7 imply $n-\lambda(\Omega) < 2n+1-\lambda(\Omega)$ for any $D(I)$-set Ω satisfying $|\Omega^{(e,i)}| \geq 1$, $|\Omega^{(i,i)}| \geq 2$, $|\Omega^{(ii,i)}| \geq 1$ for some $i \in I$, hence by (2) we have proved (3).

To derive

(4) $\quad n-\lambda \leq \lambda$

we recall $n-\lambda(\Omega) = n-R(\Omega) \leq \lambda(\Omega)$ by Theorem A.8 and the main result in [20]. Strictness follows from the result proved in [29], that $\lambda(\Omega_+)$ interpreted over $(\mathscr{P}\mathscr{T}_\Omega)_+$ defines all recursively enumerable tree languages together with Theorem 7.2 and the decidability of the emptiness-problem for level-n OI languages (Theorem 7.8).

Finally

(5) $\quad 1-\lambda(\Omega) \sim \mathrm{RPS}(\Omega)$

by the Chomsky-normalform theorem 4.12. □

Since the equivalence class of a scheme is characterized by its infinite tree, the above result shows, that the class of level-n trees form a strict hierarchy with increasing n.

Corollary 9.9. $\forall n \in \omega\ n - \mathcal{T} \subsetneq n+1 - \mathcal{T}$.

Proof. Immediate from Theorem 9.8 and Corollary 4.11. □

We will close this section by showing, that IO-*like hierarchies* of the form $yield^{(n)}(\mathcal{R})$ are strict, provided $\mathcal{R}$ satisfies certain weak conditions. The proof is based on an analysis of what properties of $\mathcal{RES}$ are needed to prove, that the IO-hierarchy does not collapse, and the observation, that, due to the uniformity of the definition, this is already sufficient to derive strictness. A natural example of such an IO-like hierarchy is discussed in [24]. Engelfriet defines *macro-tree transducers* (MTT's) by canonically combining top–down tree transducers (T's) and context-free tree grammars and proves, that MTT's (working in IO-mode) can simulate the translation of derivation trees (of a program, say) induced by attribute grammars. Using *comb* it is shown, that a MTT can be simulated at the second level as a composition of top–down tree transducers (in T) and *yield*. (We note, that in the deterministic case these results have been obtained independently by Courcelle and Franco-Zannetacci [9]; there deterministic macro-tree transducers are called *primitive-recursive schemes with parameters*). Moreover, by viewing the semantic functions in denotational semantics $\mathcal{C}, \mathcal{E}, \ldots$ as states and storing the environment in the parameters, DMTT's can realize the translation induced by the denotational semantics of a language (as discussed in the introduction) for sufficiently simple languages as WHILE. To handle semantic equations for more complicated languages, the MTT has to be extended by higher-level parameters. Clearly, such transducers can be simulated at the nth level by considering the $yield^{(n)}$-image of top–down transducers working on level-n trees. The next theorem shows, that this leads to a strict hierarchy of tree-transducers.

Theorem 9.10. *Let $\mathcal{R}$ denote a family of tree-languages satisfying*

$$\exists k \in \omega\ \exists \mathcal{P} \subseteq \mathcal{F}\ \ \mathcal{P} \leq \mathcal{EXP}^k(\mathcal{P}\cup\mathcal{L})$$

s.t.

(1) $\mathcal{RES} \subseteq \mathcal{R}$,

(2) *$\mathcal{R}$ is $\mathcal{P}$-closed under intersection with regular tree languages,*

(3) *For all 'generators' G, if $L(G) \in \mathcal{R}$ is nonempty, then there exists $t \in L(G)$ s.t. $depth(t) \leq p(size(G))$ for some $p \in \mathcal{P}$*

and let $n - \mathcal{R}(\Omega) := yield^{(n)}(\mathcal{R}(D^n(\Omega))^{b_n(i)})$.

Then

$$n - \mathcal{R} \subsetneq n+1 - \mathcal{R}.$$

Proof. We prove first

(*1) $\quad n - \mathcal{R} = n+1 - \mathcal{R} \curvearrowright n+1 - \mathcal{R} = n+2 - \mathcal{R}.$

Assume $n-\mathscr{R}=n+1-\mathscr{R}$. Clearly it suffices to prove $n+2-\mathscr{R}\subseteq n+1-\mathscr{R}$, so let $L\in n+2-\mathscr{R}(\Omega)$, thus $L=yield^{(n+2)}(L_0)$ for some $L_0\in\mathscr{R}(D^{n+2}(\Omega)^{b_{n+2}(i)})$. By the assumption,

$$yield^{(n+1)}(L_0)\in n+1-\mathscr{R}(D(\Omega)^{(e,i)})=n-\mathscr{R}(D(\Omega)^{(e,i)}),$$

and thus $L\in yield(n-\mathscr{R}(D(\Omega)^{(e,i)}))=n+1-\mathscr{R}(\Omega)$.

From (3) and Lemma 7.10 we immediately obtain the following 'downward pumping' result:

(*2) Let $L=yield^{(n)}(L_0)$ where $L_0\in\mathscr{R}$ is generated by G, then

$$L\neq\emptyset \rightsquigarrow \exists t\in L\ depth(t)\leq 2^{\cdot^{\cdot^{\cdot^{2^{p(size(G))}}}}} \quad \text{for some } p\in\mathscr{POL}.$$

(n+k)

Closure of $n-\mathscr{R}$ under intersection with regular languages is proved as in the IO-case using Lemmas 8.9, 8.10.

(*3) $n-\mathscr{R}$ is $n+k$-exponentially closed under intersection with regular languages.

As in the proof of Lemma 9.5, this gives a $2(n+k)$-exponential upper bound on the rational index of a language in $n-\mathscr{R}$:

(*4) $\quad n-\mathscr{R}\subseteq\mathscr{L}(\mathscr{EXP}^{2(n+k)}(\mathscr{POL}))$.

Now by (1) we have $n-\mathscr{L}_{IO}\subseteq n-\mathscr{R}$ and thus $L_{2(n+k)+1}\in 2(n+k)+1-\mathscr{R}$, but by (*4) and Theorem 9.6 $L_{2(n+k)+1}\notin n-\mathscr{R}$ and thus $n-\mathscr{R}\subsetneq 2(n+k)+1-\mathscr{R}$. The assertion now follows from (*1). □

Note, that all language families in the OI-hierarchy satisfy the conditions on $\mathscr{R}$, in particular, the IO-hierarchy is strict.

The term 'generator' in the formulation of Theorem 9.10 is admittedly vague; the precise notion depends on $\mathscr{R}$. E.g. in the case $\mathscr{R}=T(\mathscr{RES})$, the generators are pairs (A, G), where A is a top–down tree transducer and G a regular tree grammar. Let us briefly check, that $T(\mathscr{RES})$ satisfies the assumption of the above theorem. Obviously (1) holds. For (2), we note, that $R_1\cap A(R_2)=A(A^{-1}(R_1)\cap R_2)$. Thus, since $\mathscr{RES}$ is polynomially closed under inverse top–down transductions (as $T\subseteq$ HOM $\circ$ LB and $\mathscr{RES}$ is $\mathscr{POL}$-closed under B^{-1}, see [23]), $T(\mathscr{RES})$ is $\mathscr{POL}$-closed under intersection with regular languages. Assumption (3) follows from the fact, that the domain of a top–down tree-transducer is recognizable [23] (thus downward pumping can be applied to the intersection of the domain with the regular input language) and the observation, that a (finite-state) top–down tree-transducer can only increase the depth of a tree by a constant factor, thus $\mathscr{P}$ can be chosen to be $\mathscr{POL}$. Note, that the same argumentation holds for $\mathscr{R}=\mathrm{DT}(\mathscr{RES})$.

10. Conclusion

We have shown, how a careful analysis of the discussed concepts helped to solve a concrete problem for a concrete programming language within an area of formal language theory, which is of interest in its own. We hope, that the techniques applied in this paper may be useful in treating other generalizations of string concepts induced by the diagram of the introduction, as indicated for MTT's, and establishing more connections between semantical problems for imperative languages and formal language theory.

The question remains open, in what sense the language-producing diagram possesses a limit (when starting, say, with $\mathscr{REG}$), and how this language family could be characterized operationally. We conjecture, that the limit-class can be defined operationally (or denotationally) using the *λ-calculus with reducing reflexive types* introduced in [3], where each nonterminal has a type defined using regular equations over ground types, $\times$, and $\rightarrow$, which reduces to a ground type after a finite number of applications.

Finally, we would like to indicate, that *strictness* of the OI-hierarchy cannot be proved using the approach of this paper. To improve the upper bound for the rational index, it would be necessary to derive a lower complexity for decidability of emptiness. It is easy to see, that an exponential improvement can be obtained by considering only chains in the proof of Theorem 7.11. However, due to the fact that decidability of emptiness for $\mathscr{MAC}$ is complete in EXP-TIME [26] – this follows from Theorem 50 in [23] and [4] – we can at most hope for an exponential upper bound, which would only imply $n-\mathscr{L}_{\mathrm{OI}} \subsetneq n+2-\mathscr{L}_{\mathrm{OI}}$.

Acknowledgment

I want to thank Joost Engelfriet, who suggested the generalization of the IO-hierarchy result as stated in Theorem 9.10, for many valuable discussions, Hans Langmaack for pointing out the relation between n-rational schemes and ALGOL 68 procedures which formed the basis for the analysis discussed in the introduction, and Bruno Courcelle and Jean-Marc Steyaert for many stimulating discussions. The topic of this paper was suggested by Klaus Indermark, whom I want to thank for his constant support in the different stages of this research. Special thanks are due to Elfriede Fehr, for introducing me into the mysteries of λ-calculus and spending many hours in fruitful discussions. Finally I want to thank Sigrid Horenbeek for the typing of the many versions of this manuscript.

References

[1] S.K. Abdali, A lambda-calculus model of programming languages I, II, *J. Comput. Languages* **1** (1976) 287–320.

[2] A.V. Aho, Indexed grammars—an extension of context-free grammars, *J. ACM* **15** (4) (1968) 647–671.

[3] E. Astesiano and G. Costa, Languages with reducing reflexive types, *Proc. 7th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **85** (Springer, Berlin, 1980) 38–50.

[4] C. Beeri, Two-way nested stack automata are equivalent to two-way stack automata, *J. Comput. System Sci.* **10** (1975) 317–339.

[5] J. Bilstein and W. Damm, Top–down tree-transducers for infinite trees I, *Proc. 6'ième Colloque sur les Arbres en Algebre et en Programmation*, Lecture Notes in Computer Science **112** (Springer, Berlin, 1981) 117–134.

[6] B. Boasson, B. Courcelle and M. Nivat, A new complexity measure for languages, *Proc. Conference on Theoretical Computer Science*, University of Waterloo (1977).

[8] B. Courcelle, A representation of trees by languages, *Theoret. Comput. Sci.* **6** (1978) 255–279 and **7** (1978) 25–55.

[9] B. Courcelle and P. Franchi-Zannettacci, Attribute grammars and recursive program schemes, CNRS Report 8008, University Bordeaux 1 (1980).

[10] B. Courcelle and M. Nivat, The algebraic semantics of recursive program schemes, IRIA Laboratory Report 300 (1978).

[11] H.B. Curry and R. Feys, *Combinatory Logic, Vol. 1* (North-Holland, Amsterdam, 1958).

[12] W. Damm, Higher type program schemes and their tree languages, *Proc. 3rd GI Conference on Theoretical Computer Science*, Lecture Notes in Computer Science **48** (Springer, Berlin, 1977) 51–72.

[13] W. Damm, Languages defined by higher type program schemes, *Proc. 4th Internationl Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **52** (Springer, Berlin, 1977) 164–179.

[14] W. Damm, An algebraic extension of the Chomsky-hierarchy, *Proc. Conference on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science **74** (Springer, Berlin, 1979) 266–276.

[15] W. Damm, A note on nondeterministic n-rational schemes, *Schr. Informatik Angew. Math.* (1980).

[16] W. Damm, The IO- and OI-hierarchies, *Schr. Informatik Angew. Math.* **41** (1981) revised.

[17] W. Damm and E. Fehr, On the power of self-application and higher type recursion, *Proc. 5th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science **62** (Springer, Berlin, 1978) 177–191.

[18] W. Damm and E. Fehr, A schematological approach to the analysis of the procedure concept in ALGOL-languages, *Proc. 5 ième Colloque sur les Arbres en Algebre et en Programmation*, Lille (1980) 130–134 (abstract of [19]).

[19] W. Damm and E. Fehr, A schematological approach to the analysis of the procedure concept in ALGOL-languages, *Schr. Informatik Angew. Math.*, to appear.

[20] W. Damm, E. Fehr and K. Indermark, Higher type recursion and self-application as control structures, in: E. Neuhold, Ed., *Formal Description of Programming Concepts* (North-Holland, Amsterdam, 1978) 461–487.

[21] W. Damm and A. Goerdt, An automata-theoretic characterization of the OI-hierarchy, *Schr. Informatik Angew. Math.*, to appear.

[22] H. Egli, Typed meaning in Scott's λ-calculus models, *Proc. Symposium on λ-Calculus*, Lecture Notes in Computer Science **37** (Springer, Berlin, 1975) 220–239.

[23] J. Engelfriet, Tree automata and tree grammars, Datalogisk Afdelning Report, DAIMI FN-10, Aarhus University (1975).

[24] J. Engelfriet, Some open questions and recent results on tree transducers and tree languages, *Proc. Symposium on Formal Language Theory*, Santa Barbara, 1979 (Academic Press, New York) to appear.

[25] J. Engelfriet, Two-way automata and checking automata, in: J.W. de Bakker and J. van Leeuwen, Eds., *Foundations of Computer Science III*, Mathematical Centre Tracts **108**, Vol. 1 (1979) 1–69.

[26] J. Engelfriet, Private communication (1979).

[27] J. Engelfriet, Three hierarchies of transducers, Memorandum 217, Twente University of Technology (1978).

[28] J. Engelfriet and E.M. Schmidt, IO and OI, *J. Comput. System Sci.* **15** (3) (1977) 328–353 and **16** (1) (1978) 67–99.

[29] E Fehr, Lambda-calculus as control structures of programming languages, *Schr. Informatik Angew. Math.* **57** (1980).
[30] M.J. Fischer, Grammars with macro-like productions, *Proc. 9th IEEE Conference on Switching and Automata Theory* (1968) 131–142.
[31] S.J Garland and D.C. Luckham, Program schemes, recursion schemes, and formal languages, *J. Comput. System Sci.* **7** (1979) 113–160.
[32] J.A. Goguen, J.W. Thatcher, E.G. Wagner and J.B. Wright, Initial algebra semantics and continuous algebras, *J. ACM* **24** (1) (1977) 68–95.
[33] T. Hayaschi, On derivations trees of indexed grammars –an extension of the *uvwxy*-theorem, *Publ. Res. Inst. Math. Sci.* **9** (1) (1973).
[34] M.C.B. Hennessy, The semantic of call by value and call by name in a nondeterministic environment, *SIAM J. Comput.* **9** (1) (1980) 67–84.
[35] K. Indermark, Schemes with recursion on higher types, *Proc. 5th Conference on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science **45** (Springer, Berlin, 1976) 352–358.
[36] K. Indermark, On rational definitions in complete algebras without rank, *Schr. Informatik Ange. Math.* **64** (1980).
[37] P.J. Landin, A correspondence between ALGOL 60 and Church's lambda notation, *Comm. ACM* **8** (1965) 83–101.
[38] H Langmaack, On procedures as open subroutines, *Acta Informat.* **2** (1973) 311–333 and **3** (1974) 227–241.
[39] H. Langmaack, On a theory of decision problems in programming languages, *Proc. International Conference on Mathematical Studies of Information Processing*, Lecture Notes in Computer Science **75** (Springer, Berlin, 1979) 538–557.
[40] H. Ledgard, Ten mini languages: A study of topical issues in programming languages, *Comput. Surveys* **3** (3) (1971) 115–146.
[41] D.C. Luckham, D.M.R. Park and M.S. Paterson, On formalised computer programs, *J. Comput. System Sci.* **4** (1970) 220–249.
[42] T.S.E. Maibaum, A generalized approach to formal languages, *J. Comput. System Sci.* **8** (1974) 409–439.
[43] A.N. Maslov, The hierarchy of indexed languages of an arbitrary level, *Soviet. Math. Dokl.* **15** (14) (1974) 1170–1174.
[44] A.N. Maslov, Multilevel stack automata, *Problemy Peredachi Informatsii* **12** (1) (1976) 55–62.
[45] J. Mezei and J.B. Wright, Algebraic automata and context-free sets, *Information and Control* **11** (1967) 3–29.
[46] R. Milne, The mathematical semantics of ALGOL 68, PRG report Oxford (1972).
[47] R. Milne and C. Strachey, *A Theory of Programming Language Semantics, Part a and b* (Chapman and Hall, London, 1976).
[48] R. Milner, Models of LCF, Memo AIM-186, Stanford University (1973).
[49] G. Mitschke, The standardization theorem for λ-calculus, *Z. Math. Logik. Grundlag. Math.* **25** (1) (1979) 29–31.
[50] J.M. Morris, Lambda calculus models of programming languages, Project MAC Report TR-57, Dissertation MIT (1968).
[51] P. Mosses, The mathematical semantics of ALGOL 60, PRG-Report 12, Oxford (1974).
[52] M. Nivat, Langages algébriques sur le magma libre et sémantique des schémas de programme, in: M. Nivat, Ed., *Automata, Languages, and Programming* (North-Holland, Amsterdam, 1972) 293–307.
[53] M. Nivat, On the interpretation of recursive program schemes, Symposia Matematica, *Atti del convegno d'Informatica theorica*, Rome (1972).
[54] G. D. Plotkin, LCF considered as a programming language, *Theoret. Comput. Sci.* **5** (3) (1977) 223–255.
[55] J.C. Reynolds, Towards a theory of type structure, *Colloquium on Programming*, Paris (1974).
[56] W.C. Rounds, Tree-oriented proofs of some theorems on context-free and indexed languages, *2nd Annual ACM Symposium on Theory of Computing* (1970) 109–116.
[57] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).

[58] E.M. Schmidt, Succintness of description of context-free, regular, and finite languages, Datalogisk Afdelning Report, DAIMI PB-84, Aarhus University (1978).
[59] D. Scott, The lattice of flow diagrams, *Symposium on Semantics of Algorithmic Languages*, Lecture Notes in Mathematics **188** (Springer, Berlin, 1971) 311–366.
[60] D. Scott, Continuous lattices, *Proc. Dalhousie Conference*, Lecture Notes in Mathematics **274** (Springer, Berlin, 1972) 97–134.
[61] J.M. Steyaert, Evaluation des index rationnels de quelques familles de langages, IRIA-Report No. 261 (1977).
[62] J. E. Stoy, *Denotational Semantics: The Scott–Strachey Approach to Programming Language Theory* (MIT Press, Cambridge, MA, 1977).
[63] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, *Pacific J. Math.* **5** (1955) 285–309.
[64] R.D. Tennent, The denotational semantics of programming languages, *Comm. ACM* **19** (1976) 437–453.
[65] J.W. Thatcher, Tree automata: an informal survey, in: A.V. Aho, Ed., *Currents in the Theory of Computing* (Prentice-Hall, Englewood Cliffs, NJ, 1973) 143–172.
[66] R. Turner, An algebraic theory of formal languages, *Proc. 4th Conference on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science **32** (Springer, Berlin, 1975) 426–431.
[67] J. Vuillemin, Syntaxe, sémantique et axiomatique d'un language de programmation simple, doctorial thesis, University Paris VI (1974).
[68] C. Wadsworth, The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus, *SIAM J. Comput.* **5** (3) (1976) 488–521.
[69] M. Wand, A concrete approach to abstract recusive definitions, in: M. Nivat, Ed., *Automata, Languages, and Programming* (North-Holland, Amsterdam, 1973) 331–341.
[70] M. Wand, An algebraic formulation of the Chomsky-hierarchy, *Category Theory Applied to Computation and Control*, Lecture Notes in Computer Science **25** (Springer, Berlin, 1975) 209–213.

STANDARD DENOTATIONS

$\exists!$	there exists exactly one	$\rightarrow, \overset{*}{\rightarrow}$	reflexive respectively reflexive and transitive closure of a relation $\rightarrow \subseteq M \times M$
ω	natural numbers		
$[k]$	$\{1, \ldots, k\}$		
$\mathbb{Z}$	integers		
$\mathbb{R}_+, \mathbb{R}$	(positive) reals	$a \mapsto f(a)$	notation for functions f
I^*	strings over I	$b \rightarrow a_1 \mid a_2$	conditional
e	empty word (or: symbol denoting the empty word)	$f[a/b]$	redefinition of f at argument a by b
		$f \upharpoonright N$	restriction of $f: A \rightarrow B$ on $N \subseteq A$
$l(w)$	length of w	ld	Logarithmus dualis
$w(1) \cdot \ldots \cdot w(k)$	notation for $I^* \ni w: [k] \rightarrow I$ of length k	$m \mapsto m!$	factorial
		gcd	greatest common diviser
I^+	nonempty strings over I	*lcm*	least common multiple
$w \leqslant w'$	prefixordering on strings	I.H.	Induction hypothesis
$\emptyset$	empty set		
$\lvert M \rvert$	cardinality of a set M		
$M \setminus N$	$\{m \in M \mid m \notin N\}$		

Index