AN AUTOMATA-THEORETIC CHARACTERIZATION OF THE OI-HIERARCHY

Werner Damm Andreas Goerdt

Lehrstuhl für Informatik II, RWTH Aachen

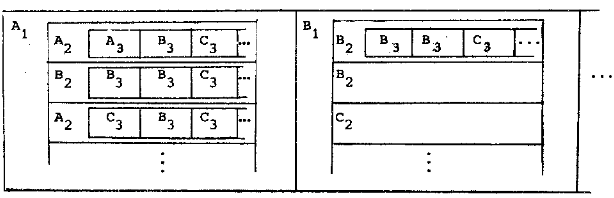
1 INTRODUCTION

One of the main objects of formal language theory has been to provide tools and models for analyzing syntactical or semantical concepts of programming languages in an abstract setting. This paper provides an operational model for the run-time behaviour of programs involving recursively defined procedures on higher types much in the same way as the classical pushdown-automaton corresponds to parameterless recursive procedures.

The storage structure for the *level-n pushdown automata* providing this characterization - originally defined in [Mas] - can be described by

- a level-1 pushdown-store consists of a (classical) pushdown-list
- a level-(n+1) pushdown-store consists of a pushdown list of pairs (pushdown symbol, level-n pushdown store).

Figure 1 below gives a typical level-3 store.

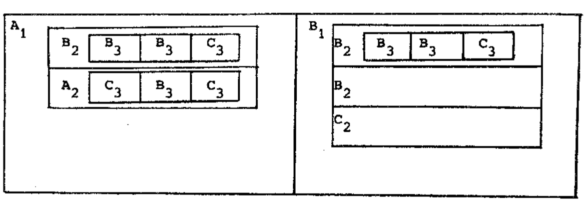


- figure 1 -

We denote by ex-pd the "undotted" version of the above store.

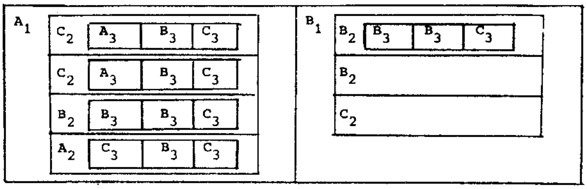
Reading on the storage structure is determined by its inductive definition: only the top pd-symbols of the pd-lists "on the top" are accessible, hence A_1, A_2 and A_3 for ex-pd. We choose to make the move of a level-n pda dependent on all of the n top symbols.

Popping at level j will delete - together with the top level-j pd-symbol - the top level-(n-j) - pd-store, in the example for $j = 2$ leading to



- figure 2 -

For the *push*-operation *at level* j a choice had to be made with respect to its implementation, since the top level-j pd-symbol in general (for $j < n$) will be "flagged" by a $(n-j)$ pd-store. The crucial observation is, that the information stored in the "flag" has to be passed to all pushed symbols (and not just to the leftmost, say), hence pushing involves *copying*. The reader familiar with indexed grammars [Aho] will have noticed the similarity to the "flag-passing-mechanism" in non-index derivation steps. Figure 3 shows the store resulting from pushing C_2C_2 at level-2 on ex-pd.



- figure 3 -

Clearly 2-pda's are equivalent (using any standard notion of acceptance) to the *indexed pushdown automata* of Duske et al., thus by theorem 2.2 in [PDS] they are equivalent to the *nested stack automata* [Aho] , i.e. they accept exactly the class of indexed-languages. This paper extends these automata-theoretic characterizations to the language families in the *OI-hierarchy* [Da 1 , ES, Wa] using n-pda's, thus lending more support to the claim, that the OI-hierarchy forms *the* natural extension of the Chomsky-hierarchy [Wa] . In particular it shows how to implement the combination of copying and parallel processing inherent in *level-n grammars* (which are generalized from *macro-grammars* [Fi] by allowing nonterminals to carry up to n levels of parameters) by superimposing pushdown lists.

The abstraction process leading from higher-type procedures to level-n grammars has been discussed in detail elsewhere ([Da 1, DF]) using a subset of finitely typed ALGOL 68 programs (see also [Kot] for the case $n = 1$). We note that a direct attempt to simulate the run-time behaviour of such programs on level-n pda's turned out to be extremly difficult and could only be proved for $n \geqslant 3$ by imposing additional restrictions on the programming language [Kle] .

To our knowledge, the concept underlying n-pda's was first mentioned in [Gre] and then defined in [Mas] to provide an automata-model for *generalized indexed languages* (obtained by allowing flags to be flagged and iterating this process). Because of the similarity of structure - we will in fact use a notational variant of generalized indexed expressions as denotations for level-n pd-stores - accepting such languages on an n-pda is straightforward. The opposite inclusion - though only sketched in [Mas] - demands a series of normalizations and will be proved in detail elsewhere [Goe 2] . Finally, [DGu] characterizes level-n languages by *level-n stack-automata* which are obtained from stack-automata by increasing the complexity of the operations on the stack (rather than the storage-structure itself).

2 LEVEL-N PDA'S

We start the formal treatment of n-pda's by translating the intuitive pictures of the introduction into a mathematically handier notation. As a motivation, let us describe ex-pd in an "index-oriented" fashion: then A_1 can be viewed as a (base-level) nonterminal flagged by three flags f_1, f_2, f_3 corresponding to the three pd-lists in the second component of the top of ex-pd. We attach the flags as a list to the nonterminal; hence the top of ex-pd is represented by $A_1[f_1f_2f_3]$. Following this pattern, we "unfold" the structure of f_1, f_2, f_3 yielding the expression

$$\underbrace{A_1[\,A_2}_{topsymbols}\!\!\underbrace{[A_3B_3C_3]}_{}\;\underbrace{B_2[B_3B_3C_3]}_{f_2}\;\underbrace{A_2[C_3B_3C_3]}_{f_3}]$$

with f_1 = $A_2[A_3B_3C_3]$.

The reader can easily construct the full representation of ex-pd by concatenating the representation of its top and its bottom. Note, that the symbols accessible to the automaton appear as left "slope" of the expression. The following definition formalizes this notation.

2.1 Definition
Let Γ be a set of *pushdown-symbols* and $n \in \omega$. The $[n+1]$-set[1] $n\text{-}pds(\Gamma)$ of *level-n pushdown stores* is defined inductively by

$$n\text{-}pds(\Gamma)^{n+1} = \{e\}^{2}, \quad n\text{-}pds(\Gamma)^{j} = (\Gamma[\, n\text{-}pds(\Gamma)^{j+1}])^{*} \qquad \square$$

Intuitively, $pds \in n\text{-}pds(\Gamma)^{j}$ describes a level-n pd-store, where only the levels $\geq j$ are specified. Note that $pds \neq e$ has a unique decomposition $A[\,j{+}1\text{-}flag]\,j\text{-}rest$ with $A \in \Gamma$, $j{+}1\text{-}flag \in n\text{-}pds(\Gamma)^{j+1}$), $j\text{-}rest \in n\text{-}pds(\Gamma)^{j}$. We will identify $A[e]$ with A.

Let us now formalize the operations on the store.

2.2 Definition

(1) For $j \leq n+1$, $topsyms: n\text{-}pds(\Gamma)^{j} \to \Gamma^{n-j+1}$ is defined inductively by $topsyms(e)=e$, $topsyms(A[m{+}1\text{-}flag]\,m\text{-}rest) = A \cdot topsyms(m{+}1\text{-}flag)$ for $m \leq n$.

(2) For $j \leq n$, $pop_j : n\text{-}pds(\Gamma) \dashrightarrow n\text{-}pds(\Gamma)$ is defined inductively by
- $pop_j(e)$ is undefined
- $pop^{j}(A[\,m{+}1\text{-}flag]\,m\text{-}rest) = m\text{-}rest$
  $pop^{4}_{j+1}(A[\,m{+}1\text{-}flag]\,m\text{-}rest) = A[\,pop_j(m{+}1\text{-}flag)]\,m\text{-}rest$ for $m \leq n$.

(3) For $j \leq n$, $\alpha = \alpha(1)\cdot\ldots\cdot\alpha(r) \in \Gamma^{+}$, $push_j(\alpha) : n\text{-}pds(\Gamma) \dashrightarrow n\text{-}pds(\Gamma)$ is defined by
- $push_1(\alpha)(e) = \alpha$, $push_{j+1}(\alpha)(e)$ is undefined [3]
- $push^{1}_{1}(\alpha)(A[\,m{+}1\text{-}flag]\,m\text{-}rest) = \alpha(1)[\,m{+}1\text{-}flag]\cdot\ldots\cdot\alpha(r)[\,m{+}1\text{-}flag]\,m\text{-}rest$
  $push^{1}_{j+1}(\alpha)(A[\,m{+}1\text{-}flag]\,m\text{-}rest) = A[\,push_j(\alpha)(m{+}1\text{-}flag)]\,m\text{-}rest$ for $m \leq n$.

Since 2.2 captures exactly the possible operations of an n-pda, the formal definition of its syntax and semantics is now routine and can safely be skipped on first reading.

2.3 Definition (*syntax of n-pda's*)

(1) Let $POP := \{j\text{-}pop \mid j \in [n]\}$, $PUSH(\Gamma) = \{j\text{-}push(\alpha) \mid \alpha \in \Gamma^{+} \wedge j \in [n]\}$, and $TOPSYMS(\Gamma) = \bigcup_{l \in [n]} \Gamma^{l}$ (the list of accessible pd-symbols).

(2) A *level-n pushdown automaton* over a terminal alphabet Σ is a structure $A = (Q,\Sigma,\Gamma,\delta,q_o,Z)$ with
- Q is a finite set of *states*, $q_o \in Q$ denoting the *initial state*
- Γ is a finite set of *pushdown-symbols* with $Z \in \Gamma$ as start symbol.
- the *transition function* δ maps $Q \times (\Sigma \cup \{e\}) \times TOPSYMS(\Gamma)$ into the finite subsets of $Q \times (PUSH(\Gamma) \cup POP)$ subject to $(q, j\text{-}push(\alpha)) \in \delta(p,a_e,topsyms) \rightsquigarrow j \leq l(topsyms)+1$ and $(q, j\text{-}pop) \in \delta(p,a_e,topsyms) \rightsquigarrow j \leq l(topsyms)$

(3) The class of n-pda's over Σ will be denoted $n\text{-}PDA(\Sigma)$ $\square$

2.4 Definition (*semantic of n-pda's*)

Let $A \in n\text{-}PDA(\Sigma)$ as above.

(1) The set of *configurations* of A is $Con_A = Q \times \Sigma^{*} \times n\text{-}pds(\Gamma)^{1}$.

---

1 $[n]$ denotes $\{1,\ldots,n\}$; for any set I (of *sorts*), an *I-set* S is a family of sets $(S^{i} \mid i \in I)$.

2 e denotes the empty string

3 this case will not arise in 2.4

(2) A determines its *single step relation* $\vdash_A \subseteq Con_A \times Con_A$ by
$(p,w,pds) \vdash_A (q,v,pds')$ iff
- $\delta(p,a_e,topsyms(pds)) \ni (q,k\text{-}push(\alpha))$ and $a_e v = w$ and $pds' = push_k(\alpha)(pds)$

or
- $\delta(p,a_e,topsyms(pds)) \ni (q,j\text{-}pop)$ and $a_e v = w$ and $pds' = pop_j(pds)$

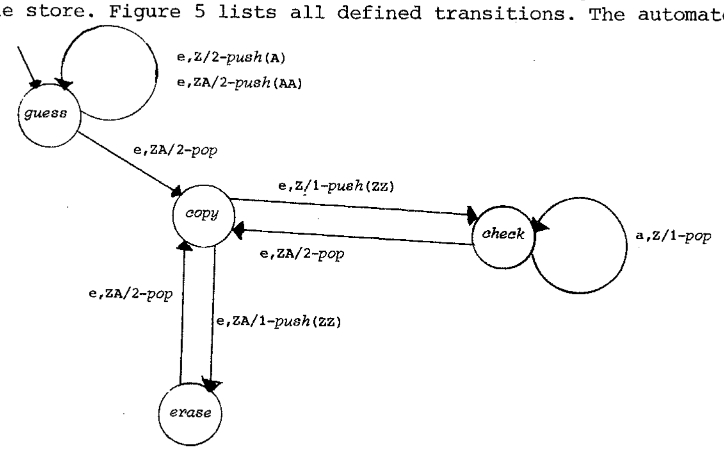
(3) The *language accepted by* A (with empty store) is defined by
$L(A) = \{w \in \Sigma^* \mid (q_o,w,Z) \vdash_A^* (q,e,e)\}$.

(4) The class of languages accepted by n-pda's over Σ is denoted n-$PDA(\Sigma)$. □

The usual techniques for pda's can be used to prove that acceptance by empty store, with final states, with final states and by empty store define the same class of languages. Rather than stating this exercise in automata-theory we give an example (which is simple enough to be used when trying out the constructions of section 3).

2.5 Example

The following 2-pda accepts $\{a^{2^m} \mid m \in \omega\}$ by generating $Z\underbrace{A\ldots A}_{m\text{-times}}$ and then iteratively erasing A, 1-*pushing* Z Z (which causes duplication of the flags!), and if possible checking the input against the store. If m was guessed correctly, this will empty the store. Figure 5 lists all defined transitions. The automaton starts by *guessing*.



- figure 5 - □

While the above example is simple, it does illustrate the control of copying on lower levels by higher levels. Superimposing this idea to up to n-levels leads to the characteristic n-exponential growth of level-n languages.

We note that restricting the push-operation by just passing one copy of the associated flag still takes us outside the context-free languages (see [Goe 1] for an example of a restricted 2-pda recognizing $\{a^n b^n c^n \mid n \in \omega\}$).

We close this section by normalizing the length of the strings to be pushed. This will simplify the construction in the following section.

2.6 Lemma

Any n-pda A is equivalent to an n-pda A' which pushes only strings of length two.

proof:

apply the following transformations:

case 1: $\delta(q,a_e,A_1\cdot\ldots\cdot A_l) \ni (p,j\text{-}push(X))$
is replaced by
$\delta'(q,a_e,A_1\cdot\ldots\cdot A_l) \ni ([p,j\text{-}pop],j\text{-}push(XX))$
and $\delta'([p,j\text{-}pop],e,A_1\ldots X\ldots A_l) = \{(p,j\text{-}pop)\}$

case 2: $\delta(q,a_e,topsyms(pds)) \ni (p,j\text{-}push(\alpha))$ with $\alpha = A_1\ldots A_k$ and $k \geq 3$ is simulated using states $[p,\alpha,r]$ where r is counted downward from $k-1$ to 1 while outputting the corresponding length 2 substring of α. □

3 SIMULATING N-PDA'S BY LEVEL-N GRAMMARS

Let us start this section by showing how a level-2 grammar would generate the example language of 2.5.

3.1 Example

The grammar has three nonterminals *start*, *copy*, and *guess*. It starts by calling *guess* with the terminal symbol 'a' at first level and the symbol for the empty string at level 0 (hence *guess* expects two one-element lists as parameters). *guess* nondeterministically generates m calls of the *copy* functional, which applies its first argumentlist twice to its zero-level argument-list. The formal parameters at level 1 and level 0 are y_1 and y_o, respectively.

$$start \rightarrow guess(a)(e) \qquad copy(y_1)(y_o) \rightarrow y_1(y_1(y_o))$$
$$guess(y_1)(y_o) \rightarrow guess(copy(y_1))(y_o) \mid y_1(y_o)$$ □

Again the example is typical in exhibiting the inherent copying power of higher level grammars: by successively applying j-level copy-functions to j-1-level-copy-functions (with j decreasing from n to 1) it is easy to generate functions with n-exponential growth. Note that copying has to be explicitly specified in higher-level grammars by double occurences of the same formal parameter.

The example is special in that it contains no *parallel* processing: both parameterlists have length 1. It is essentially the power of parallelism which will be exploited when simulating n-pda's.

We now briefly review the formal definition of level-n grammars. The concept of *level* of parameterlists is formalized by associating to each nonterminal a functional type over the base type l (denoting formal languages). The right-hand-sides of productions in a level-n grammar consist of finitely typed applicative terms over nonterminals, terminals, and formal parameters.

3.2 Definition

(1) The set of *derived types* over l is defined inductively by
$D^o := \{l\}$, $D^{n+1} := D^{n*} \times D^n$, $n\text{-}D := \bigcup_{m\leq n} D^m$.
Note that each $\tau \in D^n$ has a unique decomposition
$(\alpha_n,\ldots,(\alpha_o,l)\ldots)$ with $\alpha_j \in D^{j*}$.

(2) For $\alpha = \alpha(1)\ldots\alpha(k) \in D^{n*}$ we let $y_\alpha = (y_{1,\alpha(1)},\ldots,y_{k,\alpha(k)})$ and $Y_\alpha = \{y_{j,\alpha(j)} \mid j \in [k]\}$. If τ is as above, the set of *formal parameters* of *type* τ is $Y^\tau := \bigcup_{j=o}^{n} Y_{\alpha_j}$.

(3) Let N,Y denote n-D-sets, and Σ an alphabet. The n-D-set $T_{\Sigma,N,Y}$ of applicative terms over Σ,N, and Y is the smallest n-D-set satisfying
- $e \in T^l$, $\Sigma \subset T^{(l,l)}$, $N^\tau \cup Y^\tau \subset T^\tau$ [5]
- $t_o \in T^{(\alpha,\tau)}$, $t \in T^\alpha \rightsquigarrow t_o t \in T^\tau$

(4) For $t \in T_{\Sigma,N,Y}$ and τ as above, we define $t\!\downarrow := t y_{\alpha_n}\ldots y_{\alpha_o}$.

[5] For any I-set S and $\alpha \in I^*$, $S^\alpha := S^{\alpha(1)} \times \ldots \times S^{\alpha(k)}$.

(5) A *level-n grammar* over a terminal alphabet Σ is a structure $G = (N,\Sigma,P,S)$ where

- N is a finite n-D-set of nonterminals and $S \in N^{l}$ is the *startsymbol*
- P is a finite set of productions of the form $A\downarrow \rightarrow t$ with $A \in N^{\tau}$ for some τ and $t \in T^{l}_{\Sigma,N,Y^{\tau}}$.

(6) The class of level-n grammars over Σ is denoted $n\text{-}N\lambda(\Sigma)$. □

Since terms can be uniquely decomposed according to their types, brackets will be omitted (except for examples to increase readability). Note that T_{Σ} is isomorphic to the *left-concatenation-algebra over* Σ^* , in particular, concatenation itself is not allowed to construct terms (c.f. [BD]) .

It is easy to check, that the example grammar complies to the above definition using the types *start* : l , *guess* : $((l,l),(l,l))$, *copy* : $((l,l),(l,l))$ and identifying $y_1 \equiv y_{1,(l,l)}, y_o \equiv y_{1,l}$.

We refer the reader to [Da 1] for a detailed discusion and motivation of this concept. Justified by the Chomsky-normalform result proved in [Da 1] (see section 4), we specialized the above definition by allowing only applicative rather than arbitrary typed λ-terms as rigth-hand-side of a production. To generate strings using such a grammar, simply apply the ALGOL 60 copy-rule to calls of nonterminals, where all actual parameters (down to level o) are supplied. The *level-n language generated by* G,L(G) , is the set of terminal strings derivable in this way from the start symbol. We will use the fact, that outside-in - derivations - which in this monadic case coincide with leftmost derivations - are sufficient to generate all trees in L(G) [Da 1] .

It has been shown in [Da 1 , Da 2] that the classes $n\text{-}\mathcal{L}_{OI}$ of level-n languages form an infinite hierarchy of substitution closed AFL's , which starts with the regular, context-free, and macro-languages.

We now give a precise definition of leftmost derivations in level-n grammars.

3.3 Definition

Let $G \in n\text{-}N\lambda(\Sigma)$ as above.

(1) The set of *sentential forms* of G is $T^{l}_{\Sigma,N}$. Note that each sentential form can be uniquely written as $w\gamma$ where $w \in \Sigma^*$, $\gamma \equiv A\gamma_m \ldots \gamma_o$, $A \in N^{\tau}$ for some $\tau = (\alpha_m, \ldots, (\alpha_o, l)\ldots) \in n\text{-}D$, $\gamma_j \in T^{\alpha_j}_{\Sigma,N}$ for $j \in \{o,\ldots,m\}$. If γ is as above, we denote by *head*(γ) its top nonterminal A and k-*list*(γ) its k-th parameterlist γ_k .

(2) The derivation relation $\Rightarrow_G \subseteq Sen_G \times Sen_G$ is defined by $w\gamma \Rightarrow_G sen$ iff there is a productuin $A\downarrow \rightarrow t$ in P s.t. $A = head(\gamma)$ and $sen = wt[y_{\alpha_m}/m\text{-}list(\gamma)] \ldots [y_{\alpha_m}/o\text{-}list(\gamma)]$ [6] where A has type $(\alpha_m, \ldots, (\alpha_o, l)\ldots)$. It is easy to see that $sen \in Sen_G$.

(3) The OI-*language* generated by G is defined by $L_{OI}(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow}_G we\}$. □

We now turn to encoding the n-pds structure as an applicative term. Clearly, since a single move of an n-pda depends on all topsymbols of the current pds and the current state, and leftmost-derivations in level-n grammars depend on the *head* nonterminal, this will have to encode the current state and topsymbols, hence we take N to be Q·TOPSYMS .

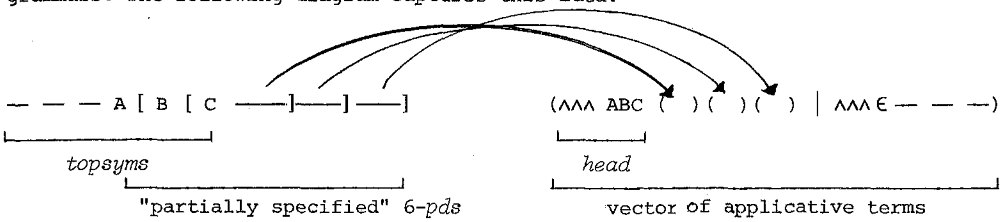
The principle of encoding a pds can then roughly be described by

---

6 For $s = (s_1, \ldots, s_q) \in T^{\alpha}_{\Sigma,N,Y}$, $t[y_{\alpha}/s]$ denotes the term obtained from t by simultaneously substituting s_j for $y_{j,\alpha(j)}$ for all $j \in [q]$

- the "slope" of topsymbols is memorized in the *head* nonterminal
- the *rests* are memorized in the parameter*lists* in decreasing order.

Now clearly - because of the inductive definition of n-*pds*(Γ) - the encoding has to cope with only "partially specified" n-pds' , i.e. elements of n-*pds*$(\Gamma)^j$ for some $j > 1$. The problem of encoding such partially specified structures has been solved by encoding *all possible extensions* to an n-pds "in parallel" - i.e. in different parameter positions - in such a way, that a *particular* extension can be recovered by means of a *projection* , i.e. a production of the form $A\, y_{\alpha_m} \dots y_{\alpha_o} \to y_{r,\alpha_r(k)}{}^{\downarrow}$, hence by an extensive use of the parallelism inherent in level-n grammars. The following diagram captures this idea.



- figure 6 : the encoding of pds' -

In the above figure, ABC denotes the fully specified part of *topsyms* , with one possible extension being ΛΛΛ ABC . Note that the type of the nonterminals has to be defined in a way providing for the storage of all possible extensions in its argumentlists.

3.4 Construction

Let $\mathcal{A} \in$ n-PDA(Σ) with states Q and pushdown-symbols Γ be in normalform according to 2.6.

(1) Let $k_j := |Q\cdot\Gamma^j|$ [7] and $nr_j : Q\cdot\Gamma^j \to [k_j]$ denote a bijective numbering with inverse w_j .

(2) The types needed can be defined inductively by $\tau_o := l$, $\tau_{j+1} := (\tau_j^{k_j}, \tau_j) \in D^{j+1}$ for $o \leqslant j < n$.

(3) The n-D-set of nonterminals is defined by $N^{\tau_o} := \{S\} \cup Q$, $N^{\tau_{j+1}} := Q\Gamma^{j+1}$ for $o \leqslant j < n$.

(4) For $1 \leqslant j \leqslant n$ we define the encoding $f_j : n\text{-}pds(\Gamma)^j \to T_N^{\tau_{j-1}^{k_{j-1}}}$ inductively by
- $f_j(e) = (w_{j-1}(1), \dots, w_{j-1}(k_{j-1}))$ (= "all possibilities")
- for $1 \leqslant n$ $f_j(A_j[\dots A_1\ 1\text{-}rest \dots]\, j\text{-}rest)$
$= (w_{j-1}(1) A_j \dots A_1 f_1(1\text{-}rest) \dots f_j(j\text{-}rest), \dots$
$\dots, w_{j-1}(k_{j-1}) A_j \dots A_1 f_1(1\text{-}rest) \dots f_j(j\text{-}rest))$

(5) The coding of configurations into sentential forms is given by
$cd : Q \times n\text{-}pds(\Gamma) \to T_N^l \quad (q, pds) \mapsto pr^{nr_o(q)}\ (f_1(pds))$ [8] □

We pause in the formal construction to illustrate the encoding by an example.

3.5 Example

Consider the 3-*pds* pds = A[B[AB]B] . For simplicity assume $Q = \{q\}$, $\Gamma = \{A, B\}$. We take the numbering induced from the lexicographical ordering with $A < B$.
$f_1(\text{pds}) = q\ ABA\ f_3(B) f_2(B) f_1(e)$

---

7 $|S|$ denotes the *cardinality* of S

8 pr^j denotes the projection on the j-th component

$$= \underbrace{qABA(qAAB(all_3),qABB(all_3),qBAB(all_3),qBB(all_3))}_{f_3(B)}\underbrace{(qAB(all_2),qBB(all_2))}_{f_2(B)}(q)$$

where $all_3(e) = (qAA,qAB,qBA,qBB)$ and $all_2 = f_2(e) = (qA,qB)$ □

3.6 Construction (cont.)

(6) In the following definition of the set of productions P we abbreviate
$y_j \equiv y_{\tau_j k_j}$

(6.1) $\delta(p,a_e,A_1\dots A_l) \ni (q, j\text{-}push(BC))$ iff [9]

$$pA_1\dots A_l{\downarrow} \rightarrow a_e qA_1\dots A_{j+1}BA_{j+1}\dots A_l y_{l-1}\cdots y_j (w_{j-1}(1)\ CA_{j+1}\dots A_l y_{l-1}\cdots y_{j-1},\dots ,w_{j-1}(k_{j-1})CA_{j+1}\dots A_l y_{l-1}\cdots y_{j-1})y_{j-2}\cdots y_o \in P$$

(6.2) $\delta(p,a_e,A_1\dots A_l) \ni (q, l+1\text{-}push(BC))$ iff [9]

$$pA_1\dots A_l{\downarrow} \rightarrow a_e qA_1\dots A_l B (w_1(1)\ C(w_1(1),\dots,w_1(k_1)),\dots ,w_1(k_1)C(w_1(1),\dots,w_1(k_1)))y_{l-1}\cdots y_o \in P$$

(6.3) $\delta(p,a_e,A_1\dots A_l) \ni (q, j\text{-}pop)$ iff [9]

$$pA_1\dots A_l{\downarrow} \rightarrow a_e y_{nr_{j-1}(qA_1\dots A_{j-1}),\tau_{j-1}} y_{j-2}\cdots y_o \in P$$

(6.4) $S \rightarrow q_o Z(w_o(1),\dots,w_o(k_o)) \in P$ (initial production)
$q \rightarrow e \in P$ (terminal productions)

Clearly the grammar G_A associated by the above construction to a level-n pda A is a level-n grammar. Note that G_A has the special property that $Sen_{G_A} \subseteq T_N^l$.

The equivalence of A and G_A rests on the following key lemma. Let, for a production $\pi \in P$, $\underset{\pi,G_A}{\Rightarrow}$ denote a derivationstep involving π.

3.7 Lemma

Let $pds \in n\text{-}pds(\Gamma)\setminus\{e\}, t \in Sen_{G_A}$, $a_e \in \Sigma \cup \{e\}$.

Then

$\exists\, pds' \in n\text{-}pds(\Gamma), q \in Q \quad (p,a_e w,pds) \vdash_A (q,w,pds') \wedge cd(p,pds') = t$

iff

$\exists \pi \in P \setminus \{\text{startproduction , terminal productions}\}\ cd(p,pds) \underset{\pi,G_A}{\Rightarrow} a_e t$

proof:

The assertion is proved by considering the cases (6.1) to (6.3). □

By a simple induction on the length of the derivation- and/or computation sequence, we obtain as a corollary to the above lemma the correctness of construction 3.6.

3.7 Corollary

$L(A) = L(G_A)$ □

4 IMPLEMENTING LEVEL-N GRAMMARS BY N-PDA'S

Consider first the problem of encoding the set mT_N of those terms which only involve *monadic* application. Since the type of such terms is uniquely determined by their (functional) level, mT_N can be viewed as an $[n]$-set (where n is the maximal level of a nonterminal in N). The coding *mcd* of such a term into a generalized indexed expression can be explained by:

- The *head* nonterminal should be the leftmost symbol having no flag.
- By viewing application of t of type (l,l) to a string as concatenation, split $sen \in Sen_G$ into its factors $sen_1 \cdot\ldots\cdot sun_k \cdot e$ and concatenate the coding of the factors.

---

9 in case $a_e \equiv e$ we identify $a_e w$ and w

- Each factor can be pictured as

$$A(\text{———}_m\ (mp_m))\ (\text{———}_{m-1}\ (mp_{m-1}))\ldots(\text{———}_1\ (mp_1))\quad,$$

level m m-1 ... 1

where the arrows point to the subexpressions of *minimal level*. Such a factor is - roughly - coded to $mp_1[\ldots mp_{m-1}[mp_m[A\ med(\text{——}_m)]\,med(\text{——}_{m-1})]\ldots med(\text{——}_1)]$.

4.1 Example

Let $sen = \underbrace{A_3(B_4(A_3)(C_2))(C_2(A_1))}_{sen\ 1}\ \underbrace{(A_3(A_2)(A_1)(e))}_{sen\ 2}$

then $med(\text{sen } 2) = A_1[A_2[A_3]]$ and $med(\text{sen } 1) = A_1[C_2[A_3\ med(B_4(A_3))]\,med(C_2)]$

hence $mcs(\text{sen}) = A_1[C_2[A_3A_3[B_4]]C_2]A_1[A_2[A_3]]$. □

It was crucial for the extension of this encoding to observe, that sentential forms generated by level-n grammars in Chomsky-Normalform (CNF) have a characteristic feature, which makes a "linearization" into monadic terms possible. To explain this property, let us recall the following result.

4.2 Theorem

Let $m \geq 1$.

Any $G' \in n\text{-}N\lambda(\Sigma)$ is equivalent to a level-n grammar $G = (N,\Sigma,S,P)$ satisfying

- S is the only nonterminal of type l
- all $A \in N \setminus \{S\}$ have *exactly one* parameter at the lowest level
- all $A \in N \setminus \{S\}$ use only *nonempty* parameterlists
- all productions in P are of one of the following forms

(1a) $Ay_{1,l} \to ay_{1,l}$ (1b) $Ay_{1,l} \to y_{1,l}$

(2a) $Ay_{1,l} \to BCy_{1,l}$

(2b) $Ay_{\alpha_m}\ldots y_{1,l} \to B(B_1y_{\alpha_m},\ldots,B_ky_{\alpha_m})y_{\alpha_{m-1}}\ldots y_l$

(3) $A{\downarrow} \to B{\downarrow}$ with $type\ A = type\ B$

(4a) $A(y_{1,(l,l)},y_{2,(l,l)})(y_{1,l}) \to y_{1,(l,l)}(y_{2,(l,l)}(y_{1,l}))$

(4b) $Ay_{\alpha_m}\ldots y_{1,l} \to y_{1,\alpha_m(1)}(y_{2,\alpha_m(2)}y_{\alpha_{m-1}},\ldots,y_{k,\alpha_m(k)}y_{\alpha_{m-1}})y_{\alpha_{m-2}}\ldots y_{1,l}$ and $m \geq 2$

(5) $Ay_{\alpha_m}\ldots y_{1,l} \to y_{j,\alpha_m(j)}y_{\alpha_{m-1}}\ldots y_{1,l}$

(6) $Ay_{\alpha_m}\ldots y_{1,l} \to B(C_1,\ldots,C_k)y_{\alpha_m}\ldots y_{1,l}$

(7) $S \to Ae$

proof:

specialize the Chomsky normalform theorem 7.3 in [Da 1] to the monadic case as in the proof of 7.17. □

The following two features of such a grammar G will be exploited:

- all sentential forms can be uniquely decomposed into factors as above (i.e. function application at the base level can be viewed as concatenation)
- G passes all actual parameterlists only as a whole (cases (1),(2),(6)) except for possibly decomposing its highest level parameterlist.

The second property induces a characteristic of G's sentential forms which allows for a coding into monadic applicative terms: parameters belonging to the *same nonterminal* and occuring at the *same fuctional level* differ at most in their head-nonterminal.

4.3 Example

The following expression satisfies the above "symmetric-list property":

$sym \equiv A_3(B_3(C_3(E_2),D_3(E_2)),F_3(C_3(E_2),\ D_3(E_2)))(G_2(H_1),K_2(H_1))(e)$

The expression can be restructured without loosing information by combining the differing topsymbols into one nonterminal:

$A_3(\underline{B_3F_3}(\underline{C_3D_3}(\underset{\uparrow}{E_2})))(\underline{G_2K_2}(\underset{\uparrow}{H_1}))(e)$. Coding this monadic term yields

$H_1[E_2[A_3\ \underline{B_3\ F_3}\ \underline{C_3\ D_3}]\ \underline{G_2\ K_2}]$. □

We now formalize the above concept of *symmetric terms* (which generalizes the notion developped by Fisher [Fi]) and prove that (the nonterminal part of) a sentential form of G is a symmetric term.

4.4 Definition

Let N an n-D-set. Denote the maximal arity occuring in a type of a nonterminal in N by M . Let $I = \{\alpha \in \bigcup_{m \in [n]} D^{m*} \mid l(\alpha) \leq M\}$.

(1) The set SL_N of *symmetric lists over* N is the smallest set $SL \subseteq \bigcup_{\alpha \in I} T_N^{\alpha}$
- $N^{\alpha} \subseteq SL$ for $\alpha \in I$
- $A_1,\ldots,A_r \in N$ and $\overline{t}_m,\ldots,\overline{t}_k \in SL$ $\leadsto$ $(A_1\overline{t}_m\ldots\overline{t}_k,\ldots,A_r\overline{t}_m\ldots\overline{t}_k) \in SL$.

(2) The set ST_N of *symmetric terms over* N is the smallest set $ST \subseteq T^{l}_{\Sigma,N}$ satisfying
- $e \in ST$
- $(slist) \in SL^{(l,l)}$, $t \in ST$ $\leadsto$ $slist\ t \in ST$ □

4.5 Lemma

Let $G = (N,\Sigma,P,S) \in n\text{-}N\lambda(\Sigma)$ be in Chomsky-Normalform.
Then
$$S \overset{+}{\Rightarrow}_G w\,A\,\overline{t}_m\ldots\overline{t}_o \leadsto A\,\overline{t}_m\ldots\overline{t}_o \in ST_N$$

proof:
The proof proceeds by induction on the length of the derivation. The base step is trivial. The induction step is proved by considering cases (1) to (6) in 4.2 for the last derivation step. □

Now that we have established the symmetric-list-property for a sufficiently rich class of terms.,let us combine the two conceptual transformation described above into a formal definition of the encoding of symmetric terms into pushdown-expressions.

4.6 Definition

Let N denote a finite n-D-set.

(1) Denote the maximal arity occuring in a type of a nonterminal in N by M . We define the set Γ_N by
$\Gamma_N = \{\underline{A_1\ldots A_l} \mid 1 \leq l \leq M\ ,\ level\ A_1 = \ldots = level\ A_l\} \cup \{Z\}$

(2) The *minimal parameter* of a symmetric list, $mp : SL_N \to \Gamma_N$ is defined inductively by
- $mp(A_1,\ldots,A_l) = \underline{A_1\ldots A_l}$
- $mp(A_1\overline{t}_m\ldots\overline{t}_k,\ldots,A_l\overline{t}_m\ldots\overline{t}_k) = mp(\overline{t}_k)$

(3) We need an auxiliary function $'$: $n\text{-}pds(\Gamma_N) \to n\text{-}pds(\Gamma_N)$ defined by

$$A_1[A_2[\ldots[A_m\,m\text{-}rest]\ldots]\,2\text{-}rest]\,1\text{-}rest \mapsto \begin{cases} A_2[\ldots[A_m\,m\text{-}rest]\ldots]\,2\text{-}rest & \text{iff } 1\text{-}rest \equiv e \\ \text{undefined otherwise.} \end{cases}$$

(4) The coding of symmetric lists $slcd : SL_N \to n\text{-}pds(\Gamma_N)$ is defined inductively by
- $slcd(A_1,\ldots,A_l) = \underline{A_1\ldots A_l}$
- $slcd(A_1\overline{t}_m\ldots\overline{t}_k,\ldots,A_l\overline{t}_m\ldots\overline{t}_k) =$
$mp\ \overline{t}_k[mp\ \overline{t}_{k-1}\ldots mp\ \overline{t}_m[\underline{A_1\ldots A_l}\ slcd\ \overline{t}_m{}']\ldots slcd\ \overline{t}_k{}']$.

(5) The coding $stcd : ST_N \to n\text{-}pds(\Gamma_N)^1$ of symmetric terms is defined by

- $std(e) = e$
- $std(\text{slist } t) = sld(\text{slist}) \cdot std(t)$ □

We now describe the simulation of G's productions following the numbering in 4.2. Note that the simulation of rules like (4 b) demands the decomposition of complex symbols and in general copying of some part of the store. A simulation of a production will be completed if the automaton reaches again its "normal" state q .

4.7 Construction

Let $G = (N,\Sigma,P,S) \in n\text{-}N\lambda(\Sigma)$ in Chomsky normalform. We define $A_G = (Q,\Sigma,\Gamma_N,\delta,q_0,Z) \in n\text{-}PDA(\Sigma)$ by

- $Q = \{q_0,q\}$

  $\cup \{[q,\pi],[q,\pi,\gamma],[p,\pi,\gamma] \mid \pi \in P$ is a type-(4 a) production, $\gamma = \underline{A_1A_2} \in \Gamma_N\}$

  $\cup \{[q,\pi],[q,\pi,\gamma],[p,\pi,\gamma] \mid \pi \in P$ is a type-(4 b) production, $\gamma = \underline{A_1\ldots A_l} \in \Gamma_N$, level $A_1 > 1$ and l is the length of the top-level parameterlist in $\pi\}$

  $\cup \{[q,\pi] \mid \pi \in P$ is a type-(5) production$\}$

- δ is defined by

(1a) - production in P iff $\delta(q,a,A) \ni (q,1\text{-}pop)$

(1b) - production in P iff $\delta(q,e,A) \ni (q,1\text{-}pop)$

(2a) - production in P iff $\delta(q,e,A) \ni (q,1\text{-}push(BC))$

(2b) - production in P iff $\delta(q,e,A_1\ldots A_mA) \ni (q,m+1\text{-}push(B\ \underline{B_1\ldots B_k}))$
for all $A_j \in \Gamma_N$.

(3) - production in P iff $\delta(q,e,A_1\ldots A_mA) \ni (q,m+1\text{-}push(B))$
for all $A_j \in \Gamma_N$.

(4a) - production in P iff $\delta(q,e,A_1A) \ni ([q,\pi],2\text{-}pop)$
for all $A_1 \in \Gamma_N$ ("erase A")

and

(i) $\delta([q,\pi],e,\underline{BC}) \ni (q,\underline{BC})$ for all $\underline{BC} \in \Gamma_N$

(ii) $\delta([q,\pi],e,A_1\ldots A_l\underline{DE}) \ni ([q,\pi,\underline{DE}],\ l+1\text{-}push(E))$
("decompose and store $\underline{DE}$ in finite control to recall D")
$\delta([q,\pi,\underline{DE}],e,A_1\ldots A_lE) \ni ([p,\pi,\underline{DE}],1\text{-}push(A\ A\ ))$
("copy;memorize copying by changing state")
$\delta([p,\pi,\underline{DE}],e,A_1\ldots A_lE) \ni (q,l+1\text{-}push(D))$
("replace the 'incorrect' E by the 'correct' D"
for all $n > l \geq 1, A_j \in \Gamma_N, \underline{DE} \in \Gamma_N$.

(4b) - production in P iff $\delta(q,e,A_1\ldots A_mA) \ni ([q,\pi],\ m+1\text{-}pop)$
for all $A_j \in \Gamma_N$ ("erase A")

and

(i) $\delta([q,\pi],e,A_1\ldots A_{m-1}\underline{A_1\ldots A_k}) \ni (q,m\text{-}push(A_1\underline{A_2\ldots A_k}))$
for all $A_j \in \Gamma_N, \underline{A_1\ldots A_k} \in \Gamma_N$ ("decompose")

(ii) $\delta([q,\pi],e,A_1\ldots A_l\underline{B_1\ldots B_k}) \ni ([q,\pi,\underline{B_1\ldots B_k}],l+1\text{-}push(\underline{B_2\ldots B_k}))$
("decompose and store in finite control to recall B_1")
$\delta([q,\pi,\underline{B_1\ldots B_k}],e,A_1\ldots A_l\underline{B_2\ldots B_k}) \ni ([p,\pi,\underline{B_1\ldots B_k}],m\text{-}push(A_mA_m))$
("copy at level m; memorize copying by changing state")
$\delta([p,\pi,\underline{B_1\ldots B_k}],e,A_1\ldots A_l\underline{B_2\ldots B_k}) \ni (q,l+1\text{-}push(B_1))$
("replace 'incorrect' $\underline{B_2\ldots B_k}$ by the 'correct' B_1")
for all $n > l \geq m$, $A_j \in \Gamma_N, \underline{B_1\ldots B_k} \in \Gamma$

(5) - production in P iff $\delta(q,e,A_1\ldots A_mA) \ni ([q,\pi],m+1\text{-}pop)$
and $\delta([q,\pi],e,A_1\ldots A_l\underline{B_1\ldots B_k}) \ni (q,l+1\text{-}push(B_j))$
for all $n > l \geq m-1, \underline{B_1\ldots B_k} \in \Gamma_N$

(6) - production in P iff $\delta(q,e,A_1\ldots A_mA) \ni ([q,\pi],m+1\text{-}push(\underline{C_1\ldots C_k}))$
and $\delta([q,\pi],e,A_1\ldots A_m\underline{C_1\ldots C_k}) \ni (q,m+2\text{-}push(B))$
for all $A_j \in \Gamma_N$

(7) - production in P iff $\delta(q_0,e,Z) \ni (q,1\text{-}push(A))$ □

The correctness of 4.7 is due to the following key Lemma:

4.8 Lemma

Let $G = (N,\Sigma,P,S) \in n\text{-}N\lambda(\Sigma)$ be in Chomsky-Normalform, and let $t \in ST_N$, $pds \in n\text{-}pds(\Gamma_N)^1, v, w \in \Sigma^*$. Then

$\exists s \in ST_N \quad t \underset{G}{\Rightarrow} vs$ and $pds = \mathit{stcd}(s)$

iff

$(q, vw, \mathit{stcd}(t)) \overset{+}{\vdash}_{A_G} (q, w, pds)$

without entering q in intermediate computation steps

proof:

by considering the cases (1a) to (7) □

4.9 Corollary

$L_{OI}(G) = L(A_G)$ □

5 CONCLUSION

Though is was "obvious" to "insiders", that the level-n pds - which circulated in unformalized versions prior to the knowledge of Maslov's papers - just had to be *the* automata model fitting to level-n languages, the complexity of the encodings in both directions shows, how far apart both concepts are. We hope that the technics devellop-ed in establishing

5.1 Theorem

$\forall n \geq 1 \qquad n\text{-}\mathcal{L}_{OI}(\Sigma) = n\text{-}\mathcal{PDA}(\Sigma)$ □

will turn out to be useful in further applications, e.g. reducing the equivalence problem of level-n schemes [Da 1] to that of deterministic n-pda's (c.f. [Cou], [Gal] for the case n = 1).

ACKNOWLEDGEMENTS

We would like to thank Joost Engelfriet for many helpfull comments on a first draft of this paper.

REFERENCES

[Aho] AHO, A.V. *Nested stack automata*, JACM 16, 3 (1969), 383-406

[BD] BILSTEIN, J. / DAMM, W. *Top-down tree-transducers for infinite trees* I, Proc. 6th CAAP, LNCS 112 (1981), 117-134

[Cou] COURCELLE, B. *A representation of trees by languages*, TCS 6, (1978), 255-279 and 7, (1978), 25-55

[Da 1] DAMM, W. *The IO- and OI-hierarchies*, TCS 20, (1982), to appear

[Da 2] DAMM, W. *An algebraic extension of the Chomsky-hierarchy*, Proc. MFCS'79, LNCS 74 (1979), 266-276

[DF] DAMM, W. / FEHR, E. *A schematalogical approach to the analysis of the procedure concept in ALGOL-languages*, Proc. 5th CAAP, Lille, (1980), 130-134

[DGu] DAMM, W. / GUESSARIAN, I. *Combining T and level-N*, Proc. MFCS'81, LNCS 118 (1981), 262-270

[ES] ENGELFRIET, J. / SCHMIDT, E.M. *IO and OI*, JCSS 15, 3 (1977), 328-353 and JCSS 16, 1 (1978), 67-99

[Fi] FISCHER, M.F. *Grammars with macro-like productions*, Proc. 9th SWAT, (1968), 131-142

[Gal] GALLIER, J.H. *Deterministic finite automata with recursive calls and DPDA's* technical report, University of Pennsylvania, (1981)

[Goe 1] GOERDT, A. *Eine automatentheoretische Charakterisierung der OI-Hierarchie*, to appear

[Goe 2] GOERDT, A. *Characterizing generalized indexed languages by n-pda's* Schriften zur Informatik und Angewandten Mathematik, RWTH Aachen, to appear

[Gre] GREIBACH, S.A. *Full AFL's and nested iterated substitution*, Information and Control 16, 1 (1970), 7-35

[Kle] KLEIN, H.-J. personal communication

[Kot] KOTT, L. *Sémantique algébrique d'un langage de programmation type ALGOL* RAIRO 11, 3 (1977), 237-263

[Mas] MASLOV, A.N. *Multilevel stack automata*, Problemy Peredachi Informatsii 12, 1 (1976), 55-62

[PDS] PARCHMANN, R. / DUSKE, J. / SPECHT, J. *On deterministic indexed languages*, Information and Control 45, 1 (1980), 48-67

[Wa] WAND, M. *An algebraic formulation of the Chomsky-hierarchy*, Category Theory Applied to Computation and Control, LNCS 25 (1975), 209-213