F - iteration Languages*

(extended abstract)

Jan Van Leeuwen

Berkeley - California. 1973. Utrecht - Netherlands

1. Notation and terminology will be as in Salomaa [5].
Development languages have been under investigation for some time, and a number of standard theorems have emerged, especially concerning the "parallel context-free" or oL - case (see Lindenmayer [3]). An implicit unification for some results was given in Salomaa [6], but in this note we will go much further.

The purpose of this note is to show that various results concerning the complexity of oL-languages and their extensions are in fact instances of much more general, structural theorems for F - iterated substitution languages. As such, this note may also indicate how oL-analysis, originally emerged from Lindenmayer's biological studies, fits in the wider mathematical context of the study of (nested) iterated substitution and super-AFL's (Greibach [1]).

2. Let F be some family of languages.

---

*) This research was supported by the Dutch Organization for the Advancement of Pure Research (ZWO).

Definition. An F - iteration grammar (or F-i grammar) is a 4-tuple $G = \langle V, \Sigma, S, \tau \rangle$, where V and Σ are alphabets, $\Sigma \subseteq V$, $S \in V - \Sigma$, and τ: $V^* \to V^*$ a partial F - substitution. If τ is λ-free, then G is a λ-free F-i grammar.

To allow τ to be partial is for mathematical convenience only; with the next definition of an F-i language it is seen that renaming and $\notin$ - extension would give an equivalent "total" grammar.

Definition. The language generated by an F-i grammar $G = \langle V, \Sigma, S, \tau \rangle$ is the set $L_G = \bigcup_{n \geq 0} \tau^n(S) \cap \Sigma^*$.

A language L is an F-i language iff either L or $L - \{\lambda\}$ (or both) can be generated by an F-i grammar.

The collection of F-i languages is denoted: F-I.

Note that oL-grammars as defined in [8] are precisely the F_o-i grammars, where F_o is the family of finite languages. F_o-i languages are also called canonical restrictions, extensions, or canonical extensions of oL-languages.

For later use, we abstract the following concept:

Definition. A quasoid is a family of languages containing C^*, which is closed under (arbitrary) finite substitution and $\cap R$.

3. A major difficulty in manipulations with F-i grammars is the possible presence of λ-productions. The following lemma is fundamental.

Lemma 3.1. Let F be a non-trivial family closed under (arbitrary) finite substitution and $\cap R$. Let L be generated by a λ-free F-i grammar, and h an arbitrary homomorphism. Then $h(L)$ (or possibly $h(L) - \{\lambda\}$) is again generated by a λ-free F-i grammar.

Proof. Note that F-i languages are closed under $\cap R$, λ-free h, and contain all finite languages.

Follow the lines of proof of Theorem 12 in [8], and note that all necessary modifications of productions can be attained within the assumed closure-properties of F. If the original grammar is λ-free, then so is the resulting grammar.

From this lemma we immediately derive

Theorem 3.2. Let F be a non-trivial family of languages closed under (arbitrary) finite substitution and $\cap R$.

Then each F-i grammar is equivalent (possibly up to λ) to a λ-free F-i grammar.

Proof. Insert $\notin$ for λ.

Observe that 3.1 and 3.2 apply (also) to F_0.
It is known that F_0-i languages do not form a (full) AFL (Herman[2]; for studies related to the least AFL enclosing F_0-I, see e.g. [8]). However:

Theorem 3.3 If F is a quasoid, then F-I is a full AFL

Proof. Apply Salomaa [5], IV, theorem 1.4. Note that F contains all regular sets, and theorem 3.2 permits reduction of most constructions to the λ-free case. Closure under arbitrary h follows from 3.1.

4. The merit of 3.2 is that it reduces various problems to the monotonic case.

The following few results are stated for the case that F is a quasoid, but conclusions go through for any $F' \subseteq F$.

Let F be a quasoid.

Theorem 4.1. When F is recursive, then so are all F-i languages.

Theorem 4.2. When F is included in the family of context-sensitive languages, then so are all F-i languages.

Theorem 4.3. When F is included in the family of λ-free context-free rule-labeled languages, then F-I is strictly included in the family of context-sensitive languages.

Proof. In fact, inclusion in the λ-free cfrl-languages ([7]) follows because λ-free F-i grammars can be step-wise simulated. In [7], λ-free cfrl-languages were shown to coincide with the λ-free context-free programmed languages under the free interpretation.

5. In this section we give an application to automata theory, and solve a problem implicitly left open in [8].

We begin with the following corollary to 4.3.

Let R denote the family of regular languages.

Lemma 5.1. The family of R-i languages is strictly included in CS.

R-i languages are of a particular interest, because of the following result taken from [9].

Theorem 5.2. R-I coincides with the family of languages acceptable by (unrestricted) pre-set push-down automata.

(Pre-set pda were introduced in [8]).

It then follows immediately that (unrestricted) pre-set pda are (still) strictly weaker than linear bounded automata.

Theorem 5.3. The family of (unrestricted) ppda-languages is strictly included in the family of context-sensitive languages.

This result may be strengthened (see 4.3) to inclusion in the λ-free cfrl-languages.

REFERENCES

[1] Greibach, S.A., Full AFL's and nested iterated substitution, Inf. Contr. 16 (1970), 7-35

[2] Herman, G.T., Closure properties of some families of languages associated with biological systems, (to appear).

[3] Lindenmayer, A., Cellular Automata, formal languages and developmental systems, Proc IVth Int. Congress on Logic, Method.and Phil. of Sc., Bucharest 1971.

[4] Rozenberg, G., and P.G. Doucet, On oL-Languages, Inf. Contr. 19 (1971) 302-318.

[5] Salomaa, A., Formal Languages, Acad Press (1973)

[6] Salomaa, A., On Lindenmayer AFL's, memorandum,1973.

[7] Van Leeuwen, J., Rule-labeled programs, Utrecht, 1973 (Ph.D.Thesis)

[8] Van Leeuwen, J., Preset push-down automata and oL-grammars, Technical Report No. 10, U.C. Berkeley, 1973.

[9] Van Leeuwen, J., Preset Push-down automata and their languages, (in preparation).