TRANSDUCTIONS AND THE PARALLEL GENERATION OF LANGUAGE*

K. Culik II
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada

Tom Head
Department of Mathematical Sciences
University of Alaska
Fairbanks, Alaska, U.S.A.

Research Report CS-81-03

January 1981

* This research was supported by the National Sciences and Engineering Council of Canada under grant No. A-7403 and by the National Science Foundation of the United States of America under grant MCS-8003348.

Parallel Generation of Language

K. Culik II
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Abstract

Each regular expression R over a finite vocabulary P of appropriately chosen ordered pairs provides a type of parallel rewriting scheme here called an RL scheme. The traditional IL schemes (resp., TIL schemes) are identified with certain RL schemes for which R is a strictly locally testable language (resp., a finite union of strictly locally testable languages). Using the virtual identity of RL schemes with a-transducers, a collection of decision procedures are imported into L theory from the theory of a-transducers. A collection of undecidability results for RL theory are proved by simulating Post tag systems with RL systems.

Concepts of determinism are defined for RL schemes and systems. Determinism is shown to be decidable for RL schemes but not for RL systems. Equivalence is shown to be decidable for deterministic RL schemes.

1. Introduction

The rewriting schemes established in various theories of the parallel generation of languages define direct derivation relations which are finite (rational) transductions, i.e. which are binary relations determined by a-transducers. This observation was used in [8] and [9] to apply algorithms developed for a-transducers to the investigation of Lindenmayer systems with interactions (IL systems). These results suggested the establishment of a type of L system that, in its relation generating power, would coincide with the class of a-transducers (making no move on null input). In the present article we introduce a level of L theory, RL theory, with precisely this relation defining power.

An RL scheme is based on a finite alphabet A and a finite set P of context free productions. The productions are elements of $A \times A^*$. We take P as a vocabulary and choose a regular language R over P. In examples we generally present R by means of a regular expression over P. Such a regular expression is in essence a finite transduction. An RL system is then an RL scheme augmented with an axiom string and an RL language is a language that is generated by an RL system. In the succeeding sections of this article we show that the RL formalism has an integrative capacity that allows Lindenmayer theory to be related conveniently to various other subdisciplines of theoretical computer science.

The formal definitions and elementary concepts of RL theory are given in Section 2. In Section 3 we show how the classical L schemes, from OL through TIL, are assimilated into RL theory. We identify these schemes with certain RL schemes having regular languages that are

locally testable. We explain in Section 4 the connection between RL schemes and the regular global context L schemes introduced and studied in [4]. The main results of this article are contained in Sections 5 and 6 : In Section 5 we introduce into L theory a collection of decision procedures from the theory of a-transducers. In Section 6 we simulate Post tag systems with RL systems and produce a list of undecidability results for RL systems. We conclude in Section 7 with remarks on further topics for study.

For basic references on L systems, see [10] and [17]. As a background reference on a-transducers we suggest [6], but all necessary definitions are included in Section 5. For an explanation of the biological motivation of L theory see in particular Chapter 0 and 1 of [10].

2. The RL Formalism

Definition 1. An RL scheme $S = (A, P, R)$ consists of a finite non-empty set called the alphabet of the scheme, a finite subset P of $A \times A^*$ called the set of productions, and a regular subset R over the vocabulary P of ordered pairs.

An RL scheme $S = (A, P, R)$ is a finitary device for defining relations between strings and relations between the occurrences of symbols in strings: Suppose that $u = u_1 \ldots u_n$ and $v = v_1 \ldots v_n$ are strings with the u_i in A and the v_i in A^*. Suppose further that $(u_1, v_1) \ldots (u_n, v_n)$ is a string in R. Under these circumstances we say that u directly derives v via S and write $u \Rightarrow_S v$, or sometimes merely $u \Rightarrow v$. We also say that the string $(u_1, v_1) \ldots (u_n, v_n)$ provides mother-daughter relations between the occurrences of symbols in u and symbols in v: The i'th symbol occurrence in u is u_i and it is the mother of those symbol occurrences in v provided by v_i. Any symbol occurrence in v occurs within a unique v_j and is the daughter of u_j. In the present study almost as much significance is attached to the mother-daughter relations provided by RL schemes as to the direct derivation relations themselves. The mother-daughter relations provide the theory with the tranditional tree structures which we consider to be fundamental. With respect to the usual biological metaphor, to ignore these mother-daughter relations would be to ignore cell lineages.

An RL scheme (A, P, R) is reduced if each symbol in A occurs in the left or right side of some production in P and each production in P occurs in some string in R. From a regular expression for R those productions that occur in strings in R can be read off.

The symbols of A that occur in productions occurring in R can likewise be read off. Thus it can be decided if a scheme is reduced and if it is not reduced it can be reduced by deleting productions from P and symbols from A without changing R. Apparently R itself determines a unique reduced scheme. In most contexts only reduced schemes are of significance and consequently we often identify the scheme (A, P, R) with the language R .

Defintion 2. An RL system G = (A, P, R, w) consists of an RL scheme (A, P, R) and a string w in A^+ called the axiom of the system. The language generated by G is $L(G) = \{v \text{ in } A^* \mid w \Rightarrow^* v\}$. An RL language is a language that is generated by an RL system.

Example 1. $R_1 = (a, aa)(a, a)^*(b, bb)(b, b)^*(c, cc)(c, c)^*$. This RL scheme defines the direct derivation relation $\{a^ib^jc^k \Rightarrow a^{i+1}b^{j+1}c^{k+1} \mid i,j,k \geq 1\}$. In $a^ib^jc^k \Rightarrow a^{i+1}b^{j+1}c^{k+1}$ the first occurrence of a (resp. b , resp. c) on the left is the mother of the first two occurrences of a (resp. b, resp. c) on the right. The remaining symbol occurrences on the left have unique daughters. The RL system formed by adjoining the axiom abc to R_1 generates the language $\{a^nb^nc^n \mid n \geq 1\}$.

Example 2. $R_2 = (a, a^2) \cup (a, a^3) \cup [(a, a^2)^2]^+ \cup (a, a^3)[(a, a^3)^2]^+$. This scheme defines the relation $\{a \Rightarrow a^2, a \Rightarrow a^3\} \cup \{a^{2i} \Rightarrow a^{4i} \mid i \geq 1\} \cup \{a^{2i+1} \Rightarrow a^{6i+3} \mid i \geq 1\}$. The RL system formed by adjoining the axiom a to R_2 generates the language $\{a^{2^n} \mid n \geq 0\} \cup \{a^{3^n} \mid n \geq 0\}$.

The domain of an RL scheme (A, P, R) is {u in A* | there is a v in A* such that u => v} . The domain is a regular subset of A* that can be computed from a regular expression for R by projection into the first coordinate. Thus for $R = (a, a^2)[(a, b)^2 \cup (b, a)^*]$ we have domain $a[a^2 \cup b^*]$. A complete RL scheme (A, P, R) is one for which the domain is A* . Note that completeness of RL schemes is decidable. The traditional definitions of L schemes from OL to TIL have required completeness. If an RL scheme (A, P, R) is not complete it is possible to complete it: Compute the domain D . Compute a regular expression E for the complement of D . For each occurrence of each symbol a in E replace a by (a, a) . This yields a regular expression R' over the alphabet P' = {(a, a) | a occurs in E} . The scheme (A, PUP', RUR' is complete. It defines the same relation on D as (A, P, R) and defines the identity relation on the completement of D .

A complete RL system G = (A, P, R, w) is one for which for each u in L(G) there is at least one v in A* (equivalently in L(G)) for which u => v . Completeness of RL systems is undecidable as is shown in Section 6. (See also Lemma 2 of [4].) In traditional L theory the possible lack of completeness of L systems is avoided by using only complete schemes. This could be done for RL systems or the weaker pair of conditions: axiom included in domain and range contained in domain, could be required. Note that this pair of conditions is also algorithmically testable. We prefer to make no completeness requirements at all. Even though completeness of systems is undecidable it is often clear that specific systems are complete. The systems given in Examples 1 and 2 are apparently complete.

The range of an RL scheme (A, P, R) is {v in A* | there is a u in A* such that u => v} . The range is a regular subset of A* that can be computed from a regular expression for R by projection into the second coordinate. In the question period recorded in [12] a question was raised which might be formalized: For an L scheme what are the strings, if any, that do not occur as derived strings? Such strings were called Garden of Eden strings in analogy with the use of this phrase in cellular automata. The set of all Garden of Eden strings for an RL scheme is the complement of the range of the scheme and is therefore a computable regular set. By the results of Section 3 the same is true for the classical L schemes from OL through TIL . Thus the various decision procedures available for regular sets allow most questions concerning Garden of Eden strings to be answered.

Recently Rozenberg and others [16, 18] have been developing generating schemes that include as special cases both parallel and sequential rewriting. One goal of such efforts is to provide a unified theory of generation processes that includes these two extremes. The utility of RL schemes for this purpose has not yet been investigated, but we note here that RL schemes allow simulation of both these extremes:

Let W = (A, P) consist of an alphabet A and a finite subset P of A x A* . If we wish to use W as a parallel rewriting system this can be accomplished by incorporating W into the RL scheme (A, P, P*). If we wish to use W as a sequential rewriting system this can be simulated by the RL scheme (A, PUI, I*PI*) where I is the identity relation {(a, a) | a in A} .

3. Locally Testable RL Schemes

Many finitary formalisms that determine binary relations in free monoids also provide mother-daughter relations between the symbol occurrences in each domain string and the symbol occurrences in each corresponding range string. This is true of the various types of L schemes defined in the literature and it is true also for those a-transducers (defined in Section 5) that have the property that the length of the input string associated with each edge is 1. It is with such examples in mind that we make the following informal definition.

Definition 3. Two finitary formalisms that determine binary relations and accompanying mother-daughter relations in a free monoid are congruent if they define both the same binary relation and the same mother-daughter relations.

In the usual biological metaphor congruent formalisms describe identical behavior for strings of cells.

In this section it is indicated how the various classical L schemes from OL to TIL may be identified via congruence with types of RL schemes. For the definitions of the various schemes and the relations they define see [10] or [17].

An OL scheme (A, P) consists of a finite alphabet A and a finite subset P of $A \times A^*$. Each such OL scheme (A, P) is apparently congruent to the RL scheme (A, P, P^*).

In discussing IL schemes we will follow the definition given in Herman and Rozenberg [10]. Each production of an (m, n)L scheme takes into account m symbols to the left and n symbols to the right of

each symbol being rewritten. A special symbol g not in the alphabet of the scheme is used to express the background of the string. Each string over the alphabet is considered to have m extra g's adjoined at the left end and n at the right end. Our notation for $(m, n)L$ strings will be (A, P, g) where A is the finite alphabet; P is the set of productions, which are now sensitive to context; and g is the special background symbol.

Recall that a language L over an alphabet A is strictly k-testable, for a positive integer k, if there are three subsets L_k, W_k, R_k of A^k such that a string $a_1 \ldots a_n$ of length $n \geq k$ is in L precisely if:

(1) $a_1 \ldots a_k$ is in L_k;

(2) for $1 \leq i \leq n-k-1$, $a_{i+1} \ldots a_{i+k}$ is in W_k; and

(3) $a_{n-k+1} \ldots a_n$ is in R_k.

The intersection of L with the finite set of strings of length less than k may be specified arbitrarily.

A language is strictly locally testable if it is strictly k-testable for some positive integer k. These languages are among the simplest of the regular languages. As a background reference on (strict) local testability see McNaughton and Papert [14]. For examples of more recent work see de Luca and Restivo [5] and the references listed there.

We will say that an RL scheme (A, P, R) is strictly k-testable (resp., strictly locally testable) if R is strictly k-testable (resp., strictly locally testable). The scheme in Example 1 in Section 2 is strictly 2-testable but the scheme of Example 2 in that

section is not (strictly) locally testable. The following notation is convenient in working with an RL scheme (A, P, R) : For each a in A let P_a denote the set of all those pairs in P that have a as first coordinate.

Proposition 1. Each (m, n)L scheme (A, P', g) is congruent to a strictly $m+n+1$-testable RL scheme (A, P, R) .

Proof (construction only). Let $P = \{(a, w) \mid$ there are $c(i)$ $(1 \leq i \leq m+n)$ in $A \cup \{g\}$ such that $(c(1),\ldots,c(m),a,c(m+1),\ldots,c(m+n)) \rightarrow w$ is in $P'\}$.

Begin with the settings $L = L_{m+n+1} = \phi$, $W = W_{m+n+1} = \phi$ and $X = R_{m+n+1} = \phi$. Consider in turn each production $(c(1),\ldots,c(m),a,c(m+1),\ldots,c(m+n)) \rightarrow w$ in P' for which either $c(1) \neq g$ or $c(m+n) \neq g$. For each such production enlarge either L , W, or X as follows:

Case 1. Suppose that for an i satisfying $1 \leq i \leq m$, $c(1) = \ldots = c(i) = g$ and $c(j) \neq g$ for $j > i$. Adjoin to L the set: $P_{c(i+1)}\cdots P_{c(m)}(a, w)P_{c(m+1)}\cdots P_{c(m+n)}P^i$.

Case 2. Suppose $c(i) \neq g$ for $1 \leq i \leq m+n$. Adjoin to W the set: $P_{c(1)}\cdots P_{c(m)}(a, w)P_{c(m+1)}\cdots P_{c(m+n)}$.

Case 3. Suppose that for an i satisfying $1 \leq i \leq n$ $c(m+i) = \ldots = c(m+n) = g$ and $c(j) \neq g$ for $j < m+i$. Adjoin to X the set: $P^{n-i}P_{c(1)}\cdots P_{c(m)}(a, w)P_{c(m+1)}\cdots P_{c(m+i-1)}$.

Let $E = LP^* \cap [\sim P^+(P^{m+n+1} \setminus W)P^+] \cap P^*X$. For each of the finite number of strings $a_1\ldots a_i$ in A^+ having length $i < m+n+1$ list the finite number of strings $(a_1, w_1)\ldots(a_i, w_i)$ for which $a_1\ldots a_i$ directly

derives $w_1 \ldots w_i$ with respect to the $(m, n)L$ scheme (A, P', g) and where, moreover, the mother of each symbol occurrence in w_j is a_j for $1 \leq j \leq i$. Let F be the finite set of all such strings of pairs that are associated with any of the strings in A^+ of length less than $m+n+1$. Let $R = E \cup F$. Then (A, P, R) is an RL scheme congruent to the original $(m, n)L$ scheme. #

The construction given in Prop. 1 is illustrated in the following example.

Example 3. For the $(1, 1)L$ scheme $(\{a\}, \{(g, a, g) \rightarrow a^2, (g, a, a) \rightarrow a^2, (a, a, a) \rightarrow a, (a, a, g) \rightarrow a^2\}, g)$ we have $P = (a, a^2), (a, a)$, $L_3 = (a, a^2)PP$, $W_3 = P(a, a)P$, and $R_3 = PP(a, a^2)$ so that $E = (a, a^2)P^2P^* \cup \{[P^+(P^3 \backslash P(a, a^2)P)P^+]\} \cup P^*P^2(a, a^2)$. The two strings a and a^2 of length less than 3 yield: (a, a^2) and $(a, a^2)^2$, respectively. Thus $F = (a, a^2) \cup (a, a^2)^2$. The congruent RL scheme given by the construction procedure of Prop. 1 is therefore $(\{a\}, \{(a, a^2), (a, a)\}, E \cup F)$. Notice that the strictly 3-testable $E \cup F$ is also strictly 1-testable via: $L_1 = (a, a^2)$, $W_1 = (a, a)$, $R_1 = (a, a^2)$. #

A TOL scheme $(A, T_1, \ldots, T_k)$ consists of a finite alphabet A and k finite sets (tables) of productions $T_i \subseteq A \times A^*$, $1 \leq i \leq k$. Each TOL scheme $(A, T_1, \ldots, T_k)$ is apparently congruent to the RL scheme $(A, T_1 \cup \ldots \cup T_k, T_1^* \cup \ldots \cup T_k^*)$. Each of the languages T_i^*, for $1 \leq i \leq k$, is strictly 1-testable via $L_1 = W_1 = R_1 = T_i$. A similar result holds for TIL schemes:

Proposition 2. Each TIL scheme $(A, T_1, \ldots, T_k, g)$, where $T_1, \ldots, T_k$ are k tables of productions, is congruent to an RL scheme

(A, P, R) where R in the union of k strictly locally testable language.

Proof (construction only). With a TIL scheme $(A, T_1, \ldots, T_k, g)$ there are k associated IL schemes (A, T_i, g), $1 \leq i \leq k$. For each of these IL schemes carry out the construction of Prop. 1 to produce a congruent RL scheme (A, P_i, R_i). Then $(A, P_1 U \ldots U P_k, R_1 U \ldots U R_k)$ is an RL scheme that is congruent to the original TIL scheme. #

A regular language over an alphabet A is locally testable if it is a member of the Boolean closure of the family of strictly locally testable languages over A [14, ex. 7, p. 23]. We say that an RL scheme (A, P, R) is locally testable if R is a locally testable language over the alphabet P. We have shown that each OL, TOL, IL, and TIL scheme is congruent to a locally testable RL scheme. Not all locally testable RL schemes, nor even all strictly locally testable RL schemes are congruent to TIL schemes. The following example gives a strictly 2-testable RL scheme which is not congruent to any TIL scheme.

Example 4. Consider the RL scheme $(\{a, b\}, \{(b, b), (a, a^2), (c, c), (a, a^3)\}, R)$ where R is the strictly 2-testable language defined by setting $L_2 = \{(b, b)(a, a^2)\}$, $W_2 = \{(a, a^2)^2, (a, a^2)(c, c), (c, c)(a, a^3), (a, a^3)^2\}$, $R_2 = \{(a, a^3)^2\}$ and including no string of length one in R. When baca is adjoined as an axiom to this scheme the language generated is $\{ba^{2^n}ca^{3^n} \mid n \geq 0\}$ which is not a TIL language.

4. RL Schemes and Regular Global Context L Schemes

In [4] Culik and Opatrny introduced a class of generating systems that generate precisely the same class of languages as the RL systems. These systems are the regular global context L systems which we will abbreviate RGCL systems:

Definition 4. An RGCL scheme $S = (A, B, P, C)$ consists of

(1) a finite non-empty set A called the alphabet of S,

(2) a finite non-empty set B called the set of labels,

(3) a finite subset P of $p(B) \times A \times A^*$, where $p(B)$ is the set of non-empty subsets of B, and

(4) C is a regular language over B called the control language.

The RGCL scheme $S = (A, B, P, C)$ defines a relation $u \Rightarrow_S v$ in A^* as follows: $u \Rightarrow_S v$ if $u = a_1 \ldots a_n$ with the a_i in A; $v = v_1 \ldots v_n$ with the v_i in A^*; and there are $(T_1, a_1, v_1) \ldots (T_n, a_n, v_n)$ in P with $T_1 \ldots T_n \cap C$ not empty.

An RGCL system $G = (A, B, P, C, w)$ consists of an RGCL scheme $S = (A, B, P, C)$ and a string w in A^+. The language generated by G is $L(G) = \{v \text{ in } A^* \mid w \Rightarrow_S v\}$. A language L over A is an RGCL language if $L = L(G)$ for some RGCL system G.

The RL schemes may be identified with those RGCL schemes $S = (A, B, P, C)$ for which the T in each (T, a, u) in P is a singleton and for which distinct triples in P have distinct first coordinates:

Proposition 3. Each RL scheme (A, P, R) is congruent to an RGCL scheme (A, P, P', R).

Proof (construction only). The set P itself suffices as the set of labels. To each (a, v) in P we attach the singleton set $\{(a, v)\}$

as label to produce $(\{(a, v)\}, a, v)$. This allows R to serve as the control language of the scheme (A, P, P', R). #

Although RL schemes are essentially special cases of RGCL schemes, they define the same class of relations and the same class of languages as the RGCL schemes:

Proposition 4. Each RGCL scheme (A, B, P, C) is congruent to an RL scheme (A, P', R) .

Proof (construction only). Let $P' = \{(a, v)$ in $A \times A^* \mid$ there is a subset T of B for which (T, a, v) is in $P\}$. For each b in B let $P'(b) = \{(a, v)$ in $P' \mid$ there is a subset T of B for which b is in T and (T, a, v) is in $P\}$. Let C be given as a regular expression in the symbols of B . Replace each occurrence of each symbol b in C by the finite set $P'(b)$. This substitution process yields a regular expression over the vocabulary P' . Let R be the language determined by this latter regular expression. #

Several results in the present article are closely related to results in [4]. This is especially true of the results in Section 3. Moreover, RL systems have a close affinity not only to the RGCL systems, but also to the rule context L systems introduced in [4]. A particularly valuable result of [4] that has no relative here is that every RGCL language, and therefore every RL language, is exponentially dense in the following sense: A language is exponentially dense if there exist constants c_1 and c_2 having the following property: For any $n \geq 0$ there exists a string u in L such that $c_1 e^{(n-1)c_2} \leq |u| \leq c_1 e^{nc_2}$. In [4] it was observed that the language $L = \{a^{2^{2^n}} \mid n \geq 0\}$ is context

sensitive but not exponentially dense. This language L is therefore CS but not RL .

By using a standard method of automata theory [14, p. 18 and ex. 4, p. 20] it can be shown that every RGCL scheme is congruent to an RGCL scheme that has a strictly 2-testable control language. Consequently we cannot restrict the relations generated by RGCL schemes by placing local testability condition on the control languages of these schemes. This is one reason we have preferred the RL formalism to the RGCL formalism for this exposition.

5. A-Transducers and Decidability

Definition 5. An a-transducer $T = (K, A, E, I, F)$ consists of

(1) finite non-empty sets K and A called the set of states and the alphabet, respectively,

(2) a finite subset E of $K \times A^* \times A^* \times K$ called the set of edges, and

(3) subsets I and F called the initial and final states, respectively.

Without loss of generality we have collapsed the usually separately specified input and output alphabets into a single alphabet.

Let $T = (K, A, E, I, F)$ be an a-transducer. Two edges given in a specified order $(p, u, v, q)(r, x, y, s)$ are abutted if $q = r$. A sequence $(q_0, u_1, v_1, q_1)(q_1, u_2, v_2, q_2)\ldots(q_{n-1}, u_n, v_n, q_n)$ of abutted edges is a transduction with input $u_1u_2\ldots u_n$ and output $v_1v_2\ldots v_n$ if q_0 is in I and q_n is in F. For x in A^* we define $Tx = \{y \text{ in } A^* \mid \text{there exists a transduction having input } x \text{ and output } y\}$. A-transducers T and T' with alphabet A are equivalent if $Tx = T'x$ for all x in A^*. With $T = (K, A, E, I, F)$ we associate the inverse a-transducer $T^{-1} = (K, A, E', I, F)$ where (q_i, b, a, q_j) is an edge of T^{-1} precisely if (q_i, a, b, q_j) is an edge of T. Note that u is in $T^{-1}v$ precisely if v is in Tu.

Let $S = (A, P, R)$ be an arbitrary RL scheme. Let $M = M(S)$ be the minimal state deterministic automaton that recognizes R. Since the alphabet of M is the subset P of $A \times A^*$, M may alternately be regarded as an a-transducer $T(S)$ with A as alphabet. For this a-transducer we have, for u, v in A^*: v in $T(S)u$ precisely if

$u \Rightarrow_S v$. Moreover, the same mother-daughter relations relating the occurrences of symbols in u and v are determined by T(S) as by S , i.e. S and T(S) are congruent. Their virtual identity provides a means for transferring algorithms known for a-transducers to the theory of parallel generation of languages. The proofs of all the decidability propositions in this section use this congruence of S with T(S) .

An RL scheme (A, P, R) is monotonic if whenever $u \Rightarrow v$ we have $|u| \leq |v|$. The schemes of Examples 1, 2, and 4 in preceding sections are monotonic.

Proposition 5. Monotonicity is decidable for RL schemes.

Proof. An RL scheme S is monotonic precisely if its associated a-transducer T(S) is monotonic. The algorithm given in [9, Thm. 1] (or see [11, Prop. II-1]) may be applied to decide the monotonicity of T(S) . #

An RL system with axiom w is monotonic if whenever $w \Rightarrow^* u \Rightarrow v$ we have $|u| \leq |v|$. An a-transducer with input alphabet A is monotonic on a subset L of A* if whenever u is in L and v is in Tu we have $|u| \leq |v|$.

Proposition 6. Monotonicity is decidable for those RL systems that generate context free languages.

Proof. Let G be an RL system for which L(G) is context free. Let S be the scheme of G . Then G is monotonic precisely if T(S) is monotonic on L(G) . Leguy-Cordellia [11, Prop. II-2] has given an algorithm that may be used to decide whether an a-transducer is monotonic on a context free language. Apply this algorithm to decide whether T(S)

is monotonic on $L(G)$. #

Proposition 7. The relation $\Rightarrow^*$ is decidable for monotonic RL schemes.

Proof. Let S be a monotonic RL scheme having alphabet A and let u and v be strings in A^* . Let n be the number of strings, x , in A^* satisfying $|u| \leq |x| \leq |v|$. Then $u \Rightarrow^* v$ holds precisely if $u \Rightarrow^i v$ holds for some i satisfying $0 \leq i < n$. #

Corollary 1. Membership is decidable for monotonic RL systems.

#

An RL scheme $S = (A, P, R)$ is propagating if, for a regular expression representing R , no production of the form (a, λ) , where λ is the null string and a is in A , appears in the regular expression. If S is reduced this is equivalent to requiring that P contains no production of the form (a, λ) . Thus the following assertion holds:

Propagativity of RL schemes is decidable.

An RL scheme S with alphabet A is functional if for any u in A^* there is at most one v in A^* such that $u \Rightarrow v$. The scheme is cofunctional (or codeterministic as defined in [8]) if for any v in A^* there is at most one u in A^* such that $u \Rightarrow v$. The schemes of Examples 1 and 4 are both functional and cofunctional. The scheme of Example 2 is cofunctional but not functional.

Proposition 8. Both functionality and cofunctionality are decidable for RL schemes.

Proof. Since S and $T(S)$ are congruent, S is functional precisely if $T(S)$ is functional ($\equiv$ single valued). Apply the algorithm

given in [2, Cor. 4] (see also [1, Thm. 1.2]) to decide functionality of $T(S)$. Note that S is cofunctional precisely if $T(S)^{-1}$ is functional. Consequently to decide the cofunctionality of S , apply the same algorithm to decide the functionality of $T(S)^{-1}$. #

An RL scheme $S = (A, P, R)$ is <u>deterministic</u> if for each u in A^* there is at most one string $(a_1, v_1)\ldots(a_n, v_n)$ in R for which $a_1\ldots a_n = u$.

In the usual biological metaphor it is determinism in the sense defined here that corresponds to the hypothesis: A cell in a specific state embedded in a specific way in a configuration of cells in specified states should have only one possible behavior. A. Lindenmayer [13] has emphasized the central significance of deterministic models for development of cell systems. Notice that <u>a deterministic scheme must be functional</u>. However, the scheme $(a, aa)(a, a)^* \cup (a, a)^*(a, aa)$, for example, is functional but not deterministic. The schemes of Examples 1 and 4 are deterministic.

<u>Proposition 9</u>. Determinism is decidable for RL schemes.

<u>Proof</u>. Let $S = (A, P, R)$ be an RL scheme and $T(S)$ the associated a-transducer. Let T_1 be the automaton obtained from $T(S)$ by ignoring the output strings associated with the edges of $T(S)$. The present proof and the proof of the next proposition require a feature of $T(S)$ not required by any of the other proofs in this article: $T(S)$ was obtained by re-interpreting a deterministic automaton M recognizing R to be an a-transducer. Since M is deterministic, S is deterministic precisely if T_1 is unambiguous (i.e. if no input string is accepted via two distinct paths through T_1). It can be decided algorithmically whether

the automaton T_1 is unambiguous ([3, §3] is one source of an algorithm for deciding unambiguity of an automaton). #

An RL scheme $S = (A, P, R)$ has deterministic passing if for each v in A^* there is at most one string $(a_1, v_1)\ldots(a_n, v_n)$ in R for which $v_1\ldots v_n = v$. A scheme that has deterministic parsing must be cofunctional. However, the scheme $(a, aa)(a, aa)^* \cup (a, a)^*(a, aa)$ is cofunctional but does not have deterministic parsing. The schemes of Examples 1, 2, and 4 have deterministic parsing.

Proposition 10. It is decidable whether an RL scheme has deterministic parsing.

Proof. Let $S = (A, P, R)$ be an RL scheme and $T(S)$ the associated a-transducer. Recall that $T(S)$ is the a-transducer interpretation of a deterministic automaton M recognizing R. Observe that since M is deterministic, S has deterministic parsing precisely if $T(S)^{-1}$ is single-accepting (i.e. if no input string is accepted via two distinct paths through $T(S)^{-1}$). Apply [2, Cor. 5] to decide if $T(S)^{-1}$ is single-accepting. #

Observe that RL schemes (A, P, R) and (A', P', R') are congruent precisely if the regular languages R and R' are equal. Since equality of regular expressions is decidable we have:

Congruence is decidable for RL schemes.

Two RL schemes S and S' are equivalent if they determine the same direct derivation relations, i.e. if $\Rightarrow_S = \Rightarrow_{S'}$. For equivalence there is no requirement that S and S' give rise to the same mother-daughter relations between symbol occurrences. Notice that congruent RL schemes must be equivalent. However, the schemes

(a, aa)(a, a)* and (a, a)*(a, aa) are equivalent but not congruent.

Proposition 11. Equivalence of RL schemes is undecidable.

The proof of this Proposition will use the following construction. Let T be an a-transducer with the property that the input string associated with each of its edges is of length one, i.e. is an element of A . Let P = {(a, v) in A x A* | there are states p , q of T such that (p, a, v, q) is an edge of T}. Then T may be viewed as an automaton M with P as alphabet. Let R be the language, with alphabet P , recognized by M . We associate with each such T the RL scheme S(T) = (A, P, R) and we observe that T and S(T) are congruent.

Proof (of Prop. 11). Let T and T' be an arbitrary pair of a-transducers each having the property that the length of the input string associated with each edge is one. Let S(T) and S(T') be the associated RL schemes. Since S(T) and S(T') are congruent to T and T' , respectively, S(T) and S(T') are equivalent precisely if T and T' are equivalent. The undecidability result of Griffiths [7] concerning equivalence of transducers can now be applied to conclude that equivalence of RL schemes is undecidable. #

In some significant special cases equivalence of RL schemes can be decided:

Proposition 12. If either of two RL schemes is either functional or cofunctional it can be decided whether the schemes are equivalent.

Proof. Let S and S' be RL schemes and let T(S) and T(S') be the associated a-transducers. The following assertions are equivalent:
(1) S and S' are equivalent;

(2) $T(S)$ and $T(S')$ are equivalent;

(3) $T(S)^{-1}$ and $T(S')^{-1}$ are equivalent.

If one of the schemes is functional then either $T(S)$ or $T(S')$ is functional and [2, Cor. 3] (see also [1, Cor. 1.3]) can be applied to decide equivalence of $T(S)$ and $T(S')$. If one of the schemes is cofunctional then either $T(S)^{-1}$ or $T(S')^{-1}$ is functional and [2, Cor. 3] can be applied to decide equivalence of $T(S)^{-1}$ and $T(S')^{-1}$. #

Corollary 2. Equivalence is decidable for deterministic schemes. #

The RL schemes that we regard as the most likely candidates for use as formal models of natural systems are those that are deterministic or at least functional. The following result assures that direct derivations can be calculated for these schemes with time and space requirements linear in the length of the input string. A proof can be given by analysing the operation of the a-transducer version $T(S)$ of an RL scheme S.

Proposition 13. For each RL scheme $S = (A, P, R)$ there is a linear-time linear-space algorithm which provides for each u in A^* either a v in A^* for which $u \Rightarrow v$ or the information that no such v exists. #

6. Post Tag Systems and Undecidability

All of the undecidability results in this section will be demonstrated by using Post tag systems and an undecidability result concerning these systems that was proved by Minsky [15].

A Post tag system $T = (B, f, s, p)$ consists of a finite non-empty set B, a function $f : B \to B^*$, a string s in B^+, and a positive integer p. The tag system generates a sequence of strings in B^* according to the following iterative procedure. The initial string is s. For the last string x of the sequence generated thusfar: If the length of x is less than p the sequence terminates at this x; otherwise x has as its successor the string obtained by deleting the first p elements from the string $xf(a)$ where a is the first symbol of x.

M. Minsky proved in [15] that the problem of determining whether an arbitrarily given Post tag system generates a terminating sequence is undecidable. This undecidability result is used in the proof of each of the propositions of this section.

Let $T = (B, f, s, p)$ be an arbitrary Post tag system. With T we associate the RL system $G(T)$ with alphabet $B \cup \{0, 1\}$ (where we assume that 0 and 1 are symbols not in B), axiom $s1$ and scheme:

$$\cup\{(0, 0)^*(x_1, 0)\ldots(x_i, 0)(1, 1) \mid 1 \leq i < p,\ x_1,\ldots,x_i \text{ in } B\} \cup$$
$$\cup\{(0, 0)^*(x_1, 0)\ldots(x_p, 0)[\cup\{(y, y) \mid y \text{ in } B\}]^*(1, f(x_1)1) \mid$$
$$x_1,\ldots,x_p \text{ in } B\}\ .$$

Observe that G(T) generates a string of the form 0*1 precisely if T terminates. The domain of the scheme contains no string of the form 0*1, but every string in the range of the scheme that is not of the form 0*1 does lie in the domain of the scheme. Thus the following assertions are equivalent:

(1) T does not terminate; and

(2) G(T) is a complete RL system.

Since the scheme of G(T) is propagating and deterministic we have:

Proposition 14. Completeness is undecidable for RL systems having propagating deterministic schemes. #

In all further propositions in this section it may be assumed that the schemes are complete. We specify only enough of each scheme to define a complete system. These schemes may then be completed as schemes as described in Section 2. The scheme completion method will not alter the validity of the various hypotheses on schemes appearing in the propositions, such as determinism, cofunctionality, etc.

Let $G_1(T)$ be the RL system that is identical with G(T) except that the term $(0, \lambda)^*(1, \lambda)$ is adjoined to the scheme. Note that the only length deminishing term of the scheme, $(0, \lambda)^*(1, \lambda)$, will be employed precisely if a string of the form 0*1 is generated and that when it is employed the empty string, λ, is produced. Thus the following assertions are equivalent:

(1) T does not terminate;

(2) $G_1(T)$ is monotonic; and

(3) $G_1(T)$ does not generate the null string.

Since the scheme of $G_1(T)$ is deterministic we have:

Proposition 15. For the class of RL systems, G, having complete deterministic schemes the following are undecidable:

(1) monotonicity of G,

(2) whether λ is in $L(G)$, and

(3) the membership relation for $L(G)$. #

Corollary 3. The relation $\Rightarrow^*$ is undecidable for complete deterministic schemes. #

An RL system $G = (A, P, R, w)$ is propagating if whenever $(a_1, v_1)\ldots(a_n, v_n)$ is in R with $a_1\ldots a_n$ in $L(G)$ we have $v_i \neq \lambda$ for $1 \leq i \leq n$.

Let $G_2(T)$ be the RL system that is identical with $G(T)$ except that the term $(0, 0)^*[(1, 11) \cup (1, 111)(1, \lambda) \cup (1, 1)^3]$ is adjoined to the scheme. Note that the production $(1, \lambda)$ will be employed precisely in the case that a string of the form 0^*1 is generated by $G_2(T)$. Thus the following assertions are equvialent:

(1) T does not terminate; and

(2) $G_2(T)$ is propagating.

Since the scheme of $G_2(T)$ is monotonic and deterministic we have:

Proposition 16. Propagativity is undecidable for RL systems having complete monotonic deterministic schemes. #

An RL system G with alphabet A is functional if for each u in $L(G)$ there is at most one v in A^* for which $u \Rightarrow v$.

Let $G_3(T)$ be the RL system that is identical with $G(T)$ except that the term $(0, 0)^*[(1, 1) \cup (1, 11) \cup (1, 1)^2]$ is adjoined to the scheme. Note that it is precisely from strings of the form 0^*1 that more than one string can be directly derived. Thus the following assertions are equivalent:

(1) T does not terminate; and

(2) $G_3(T)$ is functional.

Since the scheme of $G_3(T)$ is propagating we have:

Proposition 17. Functionality is undecidable for RL systems having complete propagating schemes. #

An RL system $G = (A, P, R, w)$ is deterministic if for each u in $L(G)$ there is at most one string $(a_1, v_1)\ldots(a_n, v_n)$ in R for which $a_1\ldots a_n = u$.

Let $G_4(T)$ be identical with $G(T)$ except that the term $(0, 0)^*[(1, 11) \cup (1, 1)(1, 11) \cup (1, 11)(1, 1) \cup (1, 1)^3]$ is adjoined to the scheme. Observe that the resulting scheme is functional. However, if a string of the form 0^*1 is generated by the system then a string of the form 0^*11 will be generated and such a string will give rise to a string of the form 0^*111 in two distinct ways, i.e., with two distinct mother-daughter relations. Thus the following assertions are equivalent:

(1) T does not terminate; and

(2) $G_4(T)$ is deterministic.

Since the scheme of $G_4(T)$ is also propagating we have:

Proposition 18. Determinism is undecidable for the class of RL systems having complete propagating functional schemes. #

An RL system G is cofunctional if for each v in L(G) there is at most one u in L(G) for which u => v .

An RL system G = (A, P, R, w) has deterministic parsing if for each v in L(G) there is at most one string $(a_1, v_1)\ldots(a_n, v_n)$ in R for which $a_1\ldots a_n$ is in L(G) and $v_1\ldots v_n = v$.

Let $G_5(T)$ be identical with G(T) except that the term (0, 0)*(1, 1) is adjoined to the scheme. The following assertions are equivalent:

(1) T does not terminate;

(2) $L(G_5(T))$ is infinite;

(3) $G_5(T)$ is cofunctional; and

(4) $G_5(T)$ has deterministic parsing.

Since the scheme of $G_5(T)$ is propagating and deterministic we have:

Proposition 19. For the class of RL systems having complete propagating deterministic schemes, the following properties are undecidable:

(1) the property of generating a finite language,

(2) cofunctionality, and

(3) the property of having deterministic parsing. #

Two RL systems G = (A, P, R, w) and G' = (A', P', R', w') are congruent if they generate the same language L and for each string $a_1\ldots a_n$ in L , a string $(a_1, v_1)\ldots(a_n, v_n)$ is in R precisely if it is in R' . Congruent systems needn't have the same axiom. The system obtained by adjoining to the scheme (a, b) U (b, a) the axiom b is

congruent to the system obtained by adjoining to the same scheme the axiom a .

Let $G_6(T)$ be identical with $G(T)$ except that the term $(0, 0)^*[(1, 11) \cup (1, 11)(1, 1) \cup (1, 1)^3]$ is adjoined to the scheme. Let $G_7(T)$ be identical with $G(T)$ except that the term $(0, 0)^*[(1, 11) \cup (1, 1)(1, 11) \cup (1, 1)^3]$ is adjoined to the scheme. The following assertions are equivalent:

(1) T does not terminate; and

(2) $G_6(T)$ and $G_7(T)$ are congruent.

Since the schemes of $G_6(T)$ and $G_7(T)$ are propagating, deterministic, and equivalent we have:

Proposition 20. Congruence is undecidable for pairs of RL systems having the same axiom and having equivalent complete propagating deterministic schemes. #

Since the language and sequence equivalence problem for PD1L systems are undecidable [17] these problems are also undecidable for RL systems having complete propagating deterministic schemes.

7. Concluding Remarks

1. As noted at the end of Section 2, language generation schemes that include both parallel and sequential rewriting are under study by Rozenberg and others. Can the RL formalism contribute to this work? To what extent does RL theory overlap these other models? Do the decision procedures and undecidability results of RL theory yield or suggest similar results for these new models?
2. As shown in Section 3, IL schemes are identifiable with RL schemes that are strictly locally testable and yet there are strictly 2-testable RL schemes that are not congruent to any IL scheme. This suggests the class of strictly 2-testable RL schemes (systems and languages) as an object of study. This class could be taken as the first step in a sequence that would progress through a hierarchy envolving RL schemes which as regular languages over alphabets of ordered pairs are non-counting [14]. In the biological metaphor a 2-testable RL scheme may arise from consistency restrictions on processes taking place concurrently in adjacent cells.
3. Are there phenomena of biological development that allow only highly artificial TIL models but allow simpler or more intuitively plausible (deterministic) RL models? (The RL systems are capable of expressing global constraints on development.)
4. All regular languages are RL but not all CS languages are RL [4]. Are all CF languages RL ?

5. We have made no investigation of questions arising from a partition of the alphabet into terminals and non-terminals. Such questions may have significance even for topic 1 listed above. We do note however that every recursively ennumerable language is EIL [17] and therefore also ERL .

References

1. J. Berstel, "Transductions and Context-Free Languages", B.G. Feubner, Stuttgart, 1979.

2. M. Blattner and T. Head, Single-valued a-transducers, J. Computer and System Sciences, 15 (1977), 310-327.

3. M. Blattner and T. Head, Automata that recognize intersections of free monoids, Information and Control, 35 (1977), 173-176.

4. K. Culik, II and J. Opatrny, Context in parallel rewriting, in "L Systems", G. Rozenberg and A. Salomaa, eds., Springer-Verlag, New York, 1974.

5. A. de Luca and A. Restivo, A characterization of strictly locally testable languages and its application to subsemigroups of a free semigroup, Information and Control, 44 (1980), 300-319.

6. S. Ginsburg, "Algebraic and Automata-Theoretic Properties of Formal Language," North-Holland, American Elsevier, New York, 1975.

7. T.V. Griffiths, The unsolvability of the equivalence problem for Λ-free non-deterministic generalized machines, J. Association for Computing Machinery, 15 (1968), 409-413.

8. T. Head, Codeterministic Lindenmayer schemes and systems, J. Computer and System Sciences, 19 (1979), 203-210.

9. T. Head, A-transducers and the monotonicity of IL schemes, J. Computer and System Sciences, in press.

10. G.T. Herman and G. Rozenberg, "Developmental Systems and Languages," North-Holland, Amsterdam, 1975.

11. J. Leguy-Cordellier, "Transductions Rationelles Decroissantes et Susbstitution," Thèse 3ème cycle, Université de Lille (1980).

12. A. Lindenmayer, Developmental algorithms - an application of formal language theory to biology, Materialienhefte IX, "Gestaltbildung - processe 3", Schwerpunkt Mathematisierung der Einzelwissenschaften, Universität Bielefeld, (1978), 27-44.

13. A. Lindenmayer, Personal communication (1980).

14. R. McNaughton and S. Papert, "Counter-Free Automata", M.I.T. Press, Cambridge, Massachusetts, 1971.

15. M.L. Minsky, Recursive unsolvability of Post's problem of "TAG" and other topics in the theory of Turing Machines, Annals of Mathematics, 74 (1961), 437-455.

16. G. Rozenberg, Selective substitution grammars (towards a framework for rewriting systems) Part I: Definitions and examples, EIK, 13 (1977), 455-463.

17. G. Rozenberg and A. Salomaa, "The Mathematical Theory of L Systems," Academic Press, New York, 1980.

18. G. Rozenberg and D. Wood, Context-free grammars with selective rewriting, Acta Informatica, 13 (1980), 257-268.