Church–Rosser Thue Systems and Formal Languages

ROBERT McNAUGHTON

Rensselaer Polytechnic Institute, Troy, New York

PALIATH NARENDRAN

General Electric CR&D, Schenectady, New York

AND

FRIEDRICH OTTO

State University of New York at Albany, Albany, New York

Abstract. Since about 1971, much research has been done on Thue systems that have properties that ensure viable and efficient computation. The strongest of these is the Church–Rosser property, which states that two equivalent strings can each be brought to a unique canonical form by a sequence of length-reducing rules. In this paper three ways in which formal languages can be defined by Thue systems with this property are studied, and some general results about the three families of languages so determined are studied.

Categories and Subject Descriptors: F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*classes defined by grammars or automata; decision problems; operations on languages*

General Terms: Algorithms, Languages, Theory, Verification

Additional Key Words and Phrases: Church–Rosser Thue systems

1. *Definitions, Notation, and Discussion*

A *Thue system* (after the Norwegian mathematician and logician Axel Thue who introduced it in 1914) is a set of unordered pairs of strings. We write $u_1 \leftrightarrow_T u_2$ (sometimes omitting the subscript when T is understood) when, for some $(x, y) \in T$ and strings w and z, $u_1 = wxz$ and $u_2 = wyz$. We write $\leftrightarrow_T^*$ for the reflexive and transitive closure of $\leftrightarrow_T$ (which is already symmetrical). Thus $u \leftrightarrow_T^* v$ if and only if there exist $u_0, u_1, \ldots, u_n$, $n \geq 0$, such that $u = u_0$, $v = u_n$, and, for $0 \leq i \leq n - 1$, $u_{i+1} \leftrightarrow_T u_i$. The *alphabet* of a Thue system is the set of all characters appearing in the paired strings.

The work of R. McNaughton and P. Narendran was supported in large part by National Science Foundation grant MCS 83-02123. The work of F. Otto was supported in part by the Deutsche Forschungsgemeinschaft and by the National Science Foundation under grants MCS 80-11979 and MCS 83-14977 while he was visiting at the Department of Mathematics of the University of California at Santa Barbara.

Authors' addresses: R. McNaughton, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, N.Y. 12180; P. Narendran, KW-C266, General Electric CR&D, Schenectady, NY 12345; F. Otto, Department of Computer Science, State University of New York at Albany, Albany, NY 12222.

We write $[x]_T$ to mean $\{x' \mid x' \leftrightarrow^*_T x\}$; in words it is *the congruence class* of x *modulo* T. It is well known that $\{x'y' \mid x' \in [x]_T, y' \in [y]_T\} \subseteq [xy]_T$, and that by defining $[x]_T \cdot [y]_T$ to be $[xy]_T$ we get a monoid whose elements are the congruence classes and whose identity is $[\lambda]$, where λ is the null string. Rather little of monoid (or semigroup) theory is used in this paper. However, one principle thereof that we shall encounter is that, if z is a string such that (za, z), $(az, z) \in T$, for all a in the alphabet of T, then $[z]_T$ acts as a monoid *zero* on the monoid. Thus for all x and y, $[x]_T[z]_T[y]_T = [z]_T$, corresponding to the fact that $xzy \leftrightarrow^* z$.

We write $u \rightarrow_T v$ if $u \leftrightarrow_T v$ and $|u| > |v|$. We call $\rightarrow^*_T$ reduction, which is the reflexive and transitive closure of $\rightarrow_T$. A Thue system T has the *Church–Rosser property* (after Alonzo Church and J. Barkley Rosser and their work in 1941 and 1942 on the lambda calculus), or T is a *CR system*, if for all u and v, $u \leftrightarrow^*_T v$ implies there is a t such that $u \rightarrow^*_T t$ and $v \rightarrow^*_T t$. Clearly, if $u \rightarrow_T v$, then, for some w, z and $(x, y) \in T$, where $|x| > |y|$, $u = wxz$ and $v = wyz$. We include the possibility that $|y| = 0$; that is, $y = \lambda$.

If T is a CR system, $(x, y) \in T$, $|x| = |y|$ and $T' = T - \{(x, y)\}$, then it is a consequence of the definition that T' is a CR system and T' is equivalent to T (i.e., for all w_1 and w_2, $w_1 \leftrightarrow^*_{T'} w_2$ if and only if $w_1 \leftrightarrow^*_T w_2$). We henceforth assume that no CR system has length-preserving rules.

Generally, when we work with CR systems we are interested in reductions. When we present a CR system in practice we list the pairs not in the form (x, y) but in the form $x \rightarrow y$, where $|x| > |y|$; $x \rightarrow y$ is called a length-reducing rule or, simply, *rule*. Thus $\rightarrow$ has two meanings, which does not cause confusion.

THEOREM 1.1 (Cochet and Nivat [5]). *A Thue system consisting of length-reducing rules has the Church–Rosser property if and only if*:

(1) (*Overlap condition*) *For all* $u_1, u_2, u_3 \neq \lambda$ *if* $u_1u_2 \rightarrow y_1$ *and* $u_2u_3 \rightarrow y_3$ *are rules* (*not necessarily distinct*), *then there is a* t *such that* $y_1u_3 \rightarrow^* t$ *and* $u_1y_3 \rightarrow^* t$ (*so that* $u_1u_2u_3 \rightarrow y_1u_3 \rightarrow^* t$ *and* $u_1u_2u_3 \rightarrow u_1y_3 \rightarrow^* t$).

(2) (*Substring condition*) *For all* u_1, u_2, u_3 *where* $u_1u_3 \neq \lambda$, *if* $u_1u_2u_3 \rightarrow y_{13}$ *and* $u_2 \rightarrow y_2$ *are rules, then there is a* t *such that* $y_{13} \rightarrow^* t$ *and* $u_1y_2u_3 \rightarrow^* t$ (*so that* $u_1u_2u_3 \rightarrow y_{13} \rightarrow^* t$ *and* $u_1u_2u_3 \rightarrow u_1y_2u_3 \rightarrow^* t$).

(3) (*Identity condition*) *If* $u \rightarrow y_1$ *and* $u \rightarrow y_2$ *are rules, then there is a* t *such that* $y_1 \rightarrow^* t$ *and* $y_2 \rightarrow^* t$.

An obvious, but useful, consequence of this principle is the following:

COROLLARY. *If in a Thue system T no left side of a rule properly overlaps with the left side of another rule or itself, no left side is a proper substring of another left side, and no two rules have identical left sides but different right sides, then T has the Church–Rosser property.*

Both the theorem and the corollary have been formulated in a way that seems to us to be most convenient for visual checking of a list of rules for the Church–Rosser property.

A CR system offers a convenient computation method for dealing with strings: Given a string, we simply reduce it as far as possible until we obtain an irreducible equivalent, whereupon the original string is identified. As pointed out in Book's 1982 paper [4], the time is linear in the length of the string. The irreducible equivalent of the string is truly a canonical form for its congruence class. The Church–Rosser property ensures that there cannot be two congruent irreducible strings. For example, the word problem for a CR system T can be solved in this way: Given two strings, we reduce each to its canonical form. If the two canonical forms are identical, then the two given strings are congruent modulo T; otherwise they are not.

In proving that the canonical form of a given string can be obtained in linear time, Book introduces an interesting automaton to do the computation, which is a kind of generalization of the pushdown automaton. The Book automaton has a scanning head large enough to be capable of scanning a substring as long as the longest left side of a rule in the CR system. The scanning head begins at the left end of the string. It advances rightward on the string until it sees the left side of a rule in its scan, whereupon it replaces that left side by the right side of the same rule, thus shortening the entire string. At that point it must go back on the modified string so that there is at most one new character at the right end of the scan, since conceivably that one new character might be enough to make possible a reduction that was not possible before. The scanning head then advances to the right until it again finds the left side of a rule in its scan, and so forth. When it gets to the right end of the string with no left side of a rule in its scan, the string in its present form is the canonical form of the original string. The Church–Rosser property guarantees that backtracking is never necessary, since a string can have only one irreducible equivalent. Moreover, the left-to-right order of processing the string is arbitrary. Right-to-left would have worked just as well, as would any other order that ensures that no possible reduction has been overlooked.

The constant term in the linear time function depends on the size of the CR system, especially the length of the longest left side of a rule. In general, the computation seems somewhat less efficient than a deterministic pushdown automaton. But then the languages that CR systems are capable of processing are a significant generalization of the deterministic context-free languages, as we demonstrate in Sections 2 and 3. We think that CR systems will provide an important systematic approach to string computation, although we do not think they will improve on the current techniques for writing compilers.

The linear time bound is not the only significant feature of the computation that we have been describing. To determine whether a string in a random-access memory is a palindrome, we can simply compare the first character with the last, then compare the second with the next to last, and so on. This computation has a linear time bound, but it has the feature that it simultaneously processes characters arbitrarily far apart in the given string. In contrast, the Book automaton (and a Thue-system computation in general) processes at any one time only a substring whose length is bounded by a constant. If we wanted to test for a palindrome by local processing, it seems we would have to begin by locating the middle of the given string. In Section 4 we present our reasons for conjecturing that a language of palindromes is not a language that can be represented by a CR system.

We investigate three ways in which CR systems can be used to define (and hence to test for membership in) a formal language. First, L is a *Church–Rosser congruential language* if L is the union of finitely many congruence classes of some CR system.

Second, $L \subseteq A_0^*$ is a *Church–Rosser language* (CR language) if there is a CR system T whose alphabet A_1 has A_0 as a proper subset, strings $t_1, t_2 \in (A_1 - A_0)^*$, and $Y \in A_1 - A_0$ such that, for all $w \in A_0^*$, $t_1 w t_2 \rightarrow_T^* Y$ if and only if $w \in L$. T is a *defining CR system* for the CR language L.

Third, $L \subseteq A_0^*$ is a *Church–Rosser-decidable* (*or CR-decidable*) *language* if L is a CR language, and, where T, t_1 t_2, and Y are as mentioned in that definition, there is an $N \in A_1 - A_0$ such that, for all $w \in A_0^*$, $t_1wt_2 \rightarrow_T^* N$ if and only if $w \notin L$. T is then a *defining CR system* for the CR-decidable language L.

The characters of $A_1 - A_0$ of a defining CR system of a CR or CR-decidable language L can be thought of as control characters.

Thus, in the defining CR system T of a CR-decidable language, for any $w \in A_0^*$, we can reduce t_1wt_2 to either Y (meaning yes) or N (no). It should be noted that if L is merely a CR language, we have a decision procedure that is almost as good. We reduce t_1wt_2 as far as we can. If we arrive at a string other than Y to which no rule can be applied, then we know that $w \notin L$.

We emphasize that languages in all three classes have linear-time membership algorithms. The family of CR languages is the largest family of languages we know that enjoys the computation method of CR systems.

We do not know whether there exist any CR languages that are not CR decidable. The proposition that all CR languages are CR decidable is clearly equivalent to the proposition that every CR language L has a CR system and strings t_1 and t_2, as in the definition, such that $t_1A_0^*t_2$ is contained in the union of finitely many congruence classes, where A_0 is the alphabet of L.

We shall work with the CR-decidability concept as much as possible, because we feel the reduction to Y or to N in every case is useful. However, there are certain results about CR languages that we have not been able to extend to CR-decidable languages. The following will be useful in the cases in which we are able to prove that a language is CR decidable:

THEOREM 1.2. *Let T be a CR system, $L \subseteq A_0^*$, where $A_0 \subseteq A_1$ (the alphabet of T), N, $Y \in A_1 - A_0$, t_1, $t_2 \in (A_1 - A_0)^*$. If* (1) *for all $w \in L$, $t_1wt_2 \rightarrow^* Y$,* (2) *for all $w \in A_0^* - L$, $t_1wt_2 \rightarrow^* N$, and* (3) *N and Y are irreducible in T, then $L \in CRDL$ with T its defining system.*

PROOF. It suffices to demonstrate that, for $w \in L$, $t_1wt_2 \not\rightarrow^* N$, and, for $w \in A_0^* - L$, $t_1wt_2 \not\rightarrow^* Y$. Or, equivalently, that, for all $w \in A_0^*$, not both $t_1wt_2 \rightarrow^* Y$ and $t_1wt_2 \rightarrow^* N$. But, since Y and N are not reducible, they are not equivalent, T being Church–Rosser. Hence no string can reduce to both Y and N. □

Congruential languages are quite limited, although they have been the object of some serious study; for example, there are several papers on NTS grammars (see [3] for an exposition and references to other papers). CR-congruential languages seem to be even more limited. We propose the study of CR languages and CR-decidable languages as a fruitful way of overcoming the limitations. We feel that our results in Sections 2 and 3 represent a partial success toward this objective.

In this paper we initiate an investigation into how extensive CR-language, CR-decidable-language, and CR-congruential-language classes are. We have chosen to begin by comparing our three families of languages with the well-established families of formal languages. Along these lines we have several results and several unanswered questions, which will be offered in the remaining sections.

Probably many readers will be less interested in our theorems than in our constructions, used in the proofs of the theorems, which reveal general techniques for constructing Thue systems that process well-known formal languages. We believe that these constructions offer rather strong evidence of the utility of Thue systems in processing strings.

We use the following abbreviations for certain families of languages:

CRCL	(the family of) CR congruential languages,
CRL	CR languages,
CRDL	CR-decidable languages,
CSL	context-sensitive languages,
CFL	context-free languages,
UCFL	unambiguous context-free languages,
DCFL	deterministic context-free languages,
REG	regular languages.

It is easy to see that REG $\subseteq$ CRL, CRCL $\subseteq$ CRL, CRDL $\subseteq$ CRL, and CRL $\subseteq$ CSL. Some of these are proved explicitly below.

In Sections 2 and 3 we prove that DCFL $\subseteq$ CRDL, (CFL $-$ DCFL) $\cap$ CRDL $\neq \emptyset$, (CFL $-$ UCFL) $\cap$ CRDL $\neq \emptyset$, and (CSL $-$ CFL) $\cap$ CRDL $= \emptyset$. In Section 4 we argue for the plausibility of our conjecture that UCFL $-$ CRL $\neq \emptyset$ (which implies that CFL $-$ CRL $\neq \emptyset$).

In Section 5 we compare CRCL and CRDL, proving that CRDL $-$ CRCL $\neq \emptyset$. We shall also prove there that CRCL $\cap$ (UCFL $-$ DCFL) $\neq \emptyset$, CRCL $\cap$ (CFL $-$ UCFL) $\neq \emptyset$, and CRCL $\cap$ (CSL $-$ CFL) $\neq \emptyset$. In Section 6, we prove three undecidability results about CRL.

For a string w, w^R is the string w reversed (e.g., $(abbab)^R = babba$) and $L^R = \{w \mid w^R \in L\}$, where L is a language. L_1/L_2 and $L_2\backslash L_1$ are the right and left quotients, respectively, of L_1 by L_2; that is, $\{w \mid x \in L_2,\ wx \in L_1\}$ and $\{w \mid x \in L_2,\ xw \in L_1\}$, respectively.

There are many unanswered questions. For example, although we know that CRL and CRDL are both closed under reversal (Theorem 2.3) and quotient by a single string (Theorem 6.4), we do not know whether either of them is closed under union or intersection or whether CRL is closed under complementation. (Clearly, if CRL = CRDL, then CRL is closed under complementation. It is conceivable that CRL is closed under complementation, but CRL $\neq$ CRDL.) We do not know whether CRL or CRDL is closed under concatenation, star closure, homomorphism, inverse homomorphism, or quotient. We conjecture that they are closed under none of these and under neither union nor intersection. We entertain some hope, however, that CRL may equal CRDL and thus be closed under complementation.

2. *Some Church–Rosser-Decidable Languages*

In this section we present some techniques for constructing the defining CR systems for some CR-decidable languages. We begin with an easy construction, which we promptly generalize.

THEOREM 2.1. *REG $\subseteq$ CRDL.*

PROOF. Let $(K, A_0, \mathrm{Tr}, q_0, F)$ be the deterministic finite-state automaton accepting R. (The elements of the quintuple are the set of states, the input alphabet, the transition function, the initial state, and the set of accepting states, respectively.) Let $A_1 = A_0 \cup \{Q_i \mid q_i \in K\} \cup \{Y, N, \$\}$ be the alphabet for T whose rules are as follows:

$Q_i a \rightarrow Q_j$, for every $q_i, q_j \in K$, $a \in A_0$ such that $\mathrm{Tr}(q_i, a) = q_j$.

$$Q_f\$ \rightarrow Y \quad \text{for each} \quad q_f \in F.$$
$$Q\$ \rightarrow N \quad \text{for each} \quad q \in K - F.$$

(Note: When we write "Q_i" we ask the reader to think of a single character of A_1; the subscript is not regarded as a character.) It is easy to see that, for any $w \in A_0^*$, $Q_0w\$ \to^* Y$ if and only if $w \in R$, and $Q_0w\$ \to^* N$ if and only if $w \notin R$. T is Church–Rosser by the Corollary to Theorem 1.1. □

THEOREM 2.2. *DCFL ⊆ CRDL.*

We offer some informal discussion before the proof. It would seem that we should try to construct a CR system to simulate a given deterministic pushdown automaton (pda). The system should work on a string that represents the concatenation of the pushdown and unread portion of the input string, and should be applicable only to that part of the string where the two come together. However, we find we cannot carry through the construction directly from the definition of pda because epsilon moves would seem to force us to include transformations that are not length decreasing. There are probably many ways to overcome this obstacle, but we have chosen the shift-reduce deterministic parsing of an LR(1) grammar as our point of departure.

PROOF. Let $L \subseteq A_0^*$ be a given deterministic CFL and let d_1, d_2, d_3, d_4 be new characters. That is, $d_1, d_2, d_3, d_4 \notin A_0$. Put

$$\begin{aligned} L_1 = {} & \{wd_1 \mid w \in L \text{ and } |w| \equiv 3 \pmod 4\} \\ & \cup \{wd_1d_2 \mid w \in L \text{ and } |w| \equiv 2 \pmod 4\} \\ & \cup \{wd_1d_2d_3 \mid w \in L \text{ and } |w| \equiv 1 \pmod 4\} \\ & \cup \{w \mid w \in L \text{ and } |w| \equiv 0 \pmod 4\}, \end{aligned}$$

that is, the result of padding strings in L in such a way that the resulting strings have lengths that are multiples of 4 (d_4 is used later). Clearly, $L_1 \in$ DCFL, since $L \in$ DCFL (an easy exercise using well-known results).

Let $S = (A_0 \cup \{d_1, d_2, d_3, d_4\})^4$ and let $N = |S|$. Let $A_{(4)} = \{a_1, a_2, \ldots, a_N\}$, where these are new characters. Define the one–one homomorphism h_1: $A_{(4)}^* \to S^*$, where for each i, $h_1(a_i) =$ some element of S. Let $L_2 = h_1^{-1}(L_1)$, which is a DCFL.

Construct the LR(1) grammar for L_2 in Greibach normal form. (This is possible; see [7].) This ensures that there are no productions of the form $A \to B$, where A and B are nonterminals. Also construct the LR-style parser as in [8, pp. 525ff], letting $T_0, T_1, \ldots, T_k$ be the tables from this construction. We treat each of these T's as a new character.

Form the new alphabets $C_1 = \{c_1 \mid c \in A_{(4)}\}$ and $C_2 = \{c_2 \mid c \in A_{(4)}\}$ and the homomorphism h_2: $A_{(4)}^* \to (C_1 \cup C_2)^*$ determined by $h_2(c) = c_1c_2$.

We are now ready to construct the defining CR system for L. The alphabet $A_1 = A_0 \cup \{d_1, d_2, d_3, d_4\} \cup A_{(4)} \cup C_1 \cup C_2 \cup \{T_0, \ldots, T_k\} \cup K \cup \{¢, \$, Y, N\}$, where K is the set of nonterminals for the grammar for L_2. (The characters ¢, \$, Y, and N are assumed not to exist already in the other parts of A_1.)

(i) If the action taken when the top of the stack is T_i and the input is $b \in A_{(4)}$ is SHIFT j, then we have the rule

$$T_ih_1(b) \to T_ih_2(b)T_j.$$

(ii) If the action taken is REDUCE p, where p is of the form $A \to cw$, $c \in A_{(4)}$, $w = B_1 \cdots B_k$, each B_i a nonterminal, then the stack must contain, for some $j_0, j_1, \ldots, j_k$, $T_{j_0}c_1c_2T_{j_1}B_1T_{j_2}B_2 \cdots T_{j_k}B_kT_i$ at the top. Therefore, we take the rule

$$T_{j_0}c_1c_2T_{j_1}B_1 \cdots B_kT_ih_1(b) \to T_{j_0}AT_jh_1(b),$$

where $T_j = \text{goto}(T_{j_0}, A)$. However, if $\text{goto}(T_{j_0}, A)$ indicates an error, we take instead

$$T_{j_0}c_1c_2T_{j_1}B_1 \cdots B_kT_ih_1(b) \to N.$$

Here all possibilities of T_{j_0}, etc., must be exhausted. If the reduction is done on null input, then we have to add

$$T_{j_0}c_1c_2 \cdots B_iT_ic \to T_{j_0}AT_jc$$

for all c in $A_0 \cup \{d_1, d_2, d_3, d_4, \$\}$.

(iii) If an error is indicated when the top of the stack is T_i and the input is b, then we include the rule

$$T_ih_1(b) \to N.$$

(We note that the presence of \$ at the right end of w allows us, in effect, to treat L_2 as a *strict* deterministic language in the sense of Harrison [8, p. 347]).

Before we complete the construction of our system, we note that all these rules are length reducing. We can show that there are no overlaps, substrings, or identities among the left sides (which implies that, so far, the system is Church–Rosser) by the following argument: In both kinds of rules, the symbols appearing to the right of the rightmost T_i are entirely different from those appearing to its left. Therefore, no overlap is possible, unless the left side of a rule as in case (i) appears as a suffix of a rule in case (ii). But this is ruled out since there are no shift-reduce conflicts.

Let $T_F = \text{goto}(T_0, S)$. Now, it is clear that for all u in L, $T_0ud_1d_2d_3d_4$ reduces to

(i) $T_0ST_Fd_4$ if $|u| \equiv 1 \pmod 4$ (i.e., $ud_1d_2d_3 \in L_1$),
(ii) $T_0ST_Fd_3d_4$ if $|u| \equiv 2 \pmod 4$,
(iii) $T_0ST_Fd_2d_3d_4$ if $|u| \equiv 3 \pmod 4$,
(iv) $T_0ST_Fd_1d_2d_3d_4$ if $|u| \equiv 0 \pmod 4$.

Hence, we add the rules

$$\begin{aligned}&¢T_0ST_Fd_4\$ \to Y,\\ &¢T_0ST_Fd_3d_4\$ \to Y,\\ &¢T_0ST_Fd_2d_3d_4\$ \to Y,\\ &¢T_0ST_Fd_1d_2d_3d_4\$ \to Y.\end{aligned}$$

Finally, we add $xN \to N$ and $Nx \to N$ for all $x \in A_1 - \{Y\}$. It should be clear that $w \in L$ if and only if $¢T_0wd_1d_2d_3d_4\$ \to^* Y$. For $w \notin L$, this string reduces to N, which establishes that L is a CR-decidable language. □

THEOREM 2.3. *CRL, CRCL, and CRDL are each closed under reversal.*

PROOF. Let $L \in$ CRL and T be its defining CR system. The Thue system $T^R = \{(u^R, v^R) \mid (u, v) \text{ in } T\}$ is clearly Church–Rosser and $t_1wt_2 \to^* Y \pmod T$ if and only if $t_2^Rw^Rt_1^R \to^* Y \pmod{T^R}$. Thus, $L^R \in$ CRL with T^R its defining CR system. Similarly, the result follows for CRCL and CRDL. □

THEOREM 2.4. *CRDL − DCFL ≠ ∅.*

PROOF. Let $L = \{a^nb^nc \mid n > 0\} \cup \{a^nb^{2n}d \mid n > 0\}$. It is known that L is not a DCFL (an easy exercise following Harrison [8, pp. 390–392]). Yet L^R is a DCFL and hence a CRDL. By Theorem 2.3, L is a CRDL. □

One is inclined to think, from what has been proved so far, that for every $L \in$ CFL $\cap$ CRDL either $L \in$ DCFL or $L^R \in$ DCFL. This is not so:

THEOREM 2.5. *CRDL* $-$ (*DCFL* $\cup$ *DCFL*^{R}) $\neq \varnothing$.

PROOF. Let $L = \{a^{2m}b^{2m}ce^{2n}f^{2n} \mid m, n \geq 0\} \cup \{a^{2m}b^{4m}de^{4n}f^{2n} \mid m, n \geq 0\}$. We prove Theorem 2.5 by showing that $L \in$ CRDL. That neither L nor L^R is deterministic is an easy exercise following Harrison [8, pp. 390–392].

Taking $A_1 = A_0 \cup \{a', b', b'', e', e'', f', ¢, \$, N, Y\}$, where $A_0 = \{a, b, c, d, e, f\}$, the rules of a defining CR system for L are as follows:

Group I
(1) $bbc \to b'c$
(2) $bbb' \to b'b'$
(3) $aab' \to \lambda$

Group II
(1) $bbd \to b''d$
(2) $bbb'' \to b''b''$
(3) $aab''b'' \to \lambda$

Group III
(1) $cee \to ce'$
(2) $e'ee \to e'e'$
(3) $e'ff \to \lambda$

Group IV
(1) $dee \to de''$
(2) $e''ee \to e''e''$
(3) $e''e''ff \to \lambda$

Group V
(1) $¢c\$ \to Y$
(2) $¢d\$ \to Y$

Group I*N*
(1) $abc \to N$
(2) $abb' \to N$
(3) $¢ab' \to N$
(4) $cb' \to N$
(5) $ac \to N$ (repeated)

Group II*N*
(1) $abd \to N$
(2) $abb'' \to N$
(3) $¢ab'' \to N$
(4) $ab''d \to N$
(5) $cb'' \to N$
(6) $ad \to N$ (repeated)

Group III*N*
(1) $cef \to N$
(2) $e'ef \to N$
(3) $e'f\$ \to N$
(4) $e'\$ \to N$
(5) $cf \to N$ (repeated)

Group IV*N*
(1) $def \to N$
(2) $e''ef \to N$
(3) $e''f\$ \to N$
(4) $de''f \to N$
(5) $e''\$ \to N$
(6) $df \to N$ (repeated)

Group V*N*
(1) $xN \to N$, for all $x \in A_1$
(2) $Nx \to N$, for all $x \in A_1$
(3) $x \to N$, $x \in A_0^2$, $x \neq aa, ab, bb, bc, bd, ce, de, ee, ef, ff$
(4) $¢x \to N$, $x \in A_0 - \{a, c, d\}$
(5) $x\$ \to N$, $x \in A_0 - \{c, d, f\}$
(6) $¢\$ \to N$

Note. Rules (5) in groups I*N* and III*N* and rules (6) in II*N* and IV*N* are all repeated in V*N*(3) for the sake of exposition. Note the role of V*N*(1) and (2): Once an *N* appears in a string, the whole string can be reduced to *N*.

In verifying that this Thue system has the CR property, we are aided by certain properties of the rules. We first restrict our attention to groups I, II, III, IV, and V.

The only possibilities, among these rules, where a left side properly overlaps or properly contains a left side are the overlap of I(1) and III(1) and the overlap of II(1) and IV(1). However, although $bbcee \to b'cee$ and $bbcee \to bbce'$, we have both $b'cee \to b'ce'$ and $bbce' \to b'ce'$. The overlap between II(1) and IV(1) is taken care of similarly.

We next focus on overlaps and containments involving a left side from groups I–V and a left side of groups IN–VN. The only such cases are again the c and d overlaps. For example, from III(1) and IN(1) we find that $abcee \to abce'$ and $abcee \to Nee$. But $abce'$ and Nee both go to N, using IN(1) and VN(1). We get similar results from the other overlaps.

Finally, note that all right sides of rules in groups IN–VN are N. Since N acts like a semigroup zero in the Thue system (i.e., all strings with an N anywhere reduce to N, by virtue of VN(1) and (2)), all cases in which a left side of IN–VN overlaps with or is a substring of another left side of IN–VN can be taken care of.

Thus the Thue system has the CR property. By Theorem 1.2, it remains only to prove that, for $w \in A_0^*$, $¢w\$ \to^* Y$ for $w \in L$ and $¢w\$ \to^* N$ for $w \in A_0^* - L$.

LEMMA 1. *$¢a^{2i}b^{2i}c \to^* ¢c$, $¢a^{2i}b^{4i}d \to^* ¢d$, $ce^{2i}f^{2i}\$ \to^* c\$$ and $de^{4i}f^{2i}\$ \to^* d\$$.*

PROOF. Each of these derivations uses only one group of rules: Group I, II, III, or IV, respectively.

LEMMA 2. *If j is odd, each $¢a^ib^jc$, $¢a^ib^jd$, $ce^jf^i\$$, and $de^jf^i\$$ yields N.*

PROOF. We demonstrate this fact for $¢a^ib^{2j+1}d$, $i, j \geq 1$, leaving the other cases for the reader: $¢a^ib^{2j+1}d \to^* ¢a^ib(b'')^jd \to ¢a^{i-1}N(b'')^{j-1}d \to^* N$, the N being introduced by IIN(2), group II being used up to that point.

LEMMA 3. *Each of $¢a^ib^{2j}c$ and $ce^{2j}f^i\$$ yields N, for $i \neq 2j$.*

We demonstrate the former for $i > j$: $¢a^ib^{2j}c \to^* ¢a^i(b')^jc \to^* ¢a^{i-j}c \to ¢a^{i-j-1}N \to^* N$, using I$N$(5).

LEMMA 4. *Each of $¢a^ib^{4j+2}d$ and $de^{4j+2}f^id$ yields N.*

PROOF. Each of these has three cases: $i > 2j + 1$; $i \leq 2j + 1$ and i odd; $i \leq 2j$ and i even. Each of the six cases is left to the reader.

LEMMA 5. *Each of $¢a^ib^{4j}d$ and $de^{4j}f^i\$$ yields N for $i \neq 2j$.*

PROOF. For each of the three cases are $i < 2j$; $i > 2j$ and i odd; $i > 2j$ and i even. Each case is left to the reader.

From Lemma 1 (and the two rules in group V) we conclude that, for $w \in L$, $¢w\$ \to^* Y$.

Assume now that $w \in A_0^* - L$. If $w \notin a^*b^*ce^*f^* \cup a^*b^*de^*f^*$, then $¢w\$$ must (1) be equal to $¢\$$, (2) have a substring $x \in A_0^2$ such that $x \to N$ is an instance of rule VN(3), (3) have the prefix $¢e$ or $¢f$, or (4) have the suffix $a\$$ or $b\$$. In any case N can be derived by Group VN.

If $w = a^hb^kce^mf^n$ or $w = a^hb^kde^mf^n$, then all cases of $w \notin L$ are covered by Lemmas 2–5, completing the proof (with the aid of VN(1) and (2)) that $¢w\$ \to^* N$. □

The Theorem that follows will give us a large class of strictly context-sensitive languages in CRDL. A $0L$ *system* (pronounced zero-el system; see [16, p. 235]) is a triple (A_0, P, u) where A_0 is the alphabet; P is the set of productions, which are of the form $a \to w$, where $a \in A_0$ and $w \in A_0^*$; and u is the initial word. A word x_2

is *generated* from a word x_1 in one step if $x_1 = a_1 \cdots a_n$ and $x_2 = w_1 \cdots w_n$, where, for each i, $a_i \rightarrow w_i$ is in P. The *language* of this 0L system is the set of all words generated from u in a finite number of steps.

THEOREM 2.6. *Let S be a 0L system in which every right side of P has length at least* 2, *and no right side is a prefix of another right side or of u. Then the language of this system is in CRDL.* (*Alternatively, "prefix" can be replaced by "suffix."*)

PROOF. Note that every production of P is strictly expanding; that is, for every production $a \rightarrow w$, $|w| > |a|$. Also, since no right side is a prefix (proper or otherwise) of another in the 0L-growth sequence starting from u, the predecessor of each word in the sequence can be uniquely determined. This is because every word in the sequence save u can be uniquely factorized into distinct occurrences of right sides of productions.

Thus, we construct a CR system T from S. The alphabet of T is $A_1 = A_0 \cup \{¢, \$, F, Y, N\}$, where these five characters are not in A_0. For every pair of productions $a_1 \rightarrow w_1$ and $a_2 \rightarrow w_2$ of P, we include in T the rule

(1) $¢w_1w_2 \rightarrow ¢a_1a_2F$, where $¢$ and F are new characters.

For every rule $a \rightarrow w$ of P we include

(2) $Fw \rightarrow aF$ and
(3) $¢w\$ \rightarrow ¢a\$$

in T. We include

(4) $F\$ \rightarrow \$$,
(5) $¢u\$ \rightarrow Y$,

and the following N rules:

(N1) $¢xa \rightarrow N$,

where xa is not a prefix of any right side of P or of u, but x is a proper prefix of one of these:

(N2) $¢wxa \rightarrow N$ and
(N3) $Fxa \rightarrow N$,

where xa is not a prefix of any right side of P, x is a proper prefix of one of these, and w is a right side of P;

(N4) $¢x\$ \rightarrow N$, where $x \neq u$ and x is a proper prefix of u or of some right side of P.

(This includes $¢\$ \rightarrow N$.)

(N5) $¢wx\$ \rightarrow N$ and
(N6) $Fx\$ \rightarrow N$,

where w is a right side of P and x is a nonnull proper prefix of some right side of P. Finally for every x in A_1,

(N7) $xN \rightarrow N$ and
(N8) $Nx \rightarrow N$.

(An example of this construction is given in proof of the corollary.)

We note that all the rules of T are length decreasing. Because no right side of P is a prefix of another or of u, we can verify that no two left sides of T are overlapping, nor is any left side identical to or a substring of another. Hence T is CR by the Corollary to Theorem 1.1.

Now, let w_{i+1} be in the language generated by u and w_i be its predecessor. Then it is not hard to see that $¢w_{i+1}\$ \rightarrow^* ¢w_i\$$. Hence, for any w in the 0L language, $¢w\$ \rightarrow_T^* ¢u\$ \rightarrow Y$.

Assume now that $w \in A_0^*$ but not in the 0L language. We construct a derivation $¢w\$ \rightarrow ¢\alpha_1\$ \rightarrow ¢\alpha_2\$ \rightarrow \cdots \rightarrow ¢\alpha_n\$$ as long as possible, subject to the following stipulations:

(a) Only rules (1)–(4) are used.
(b) If α_i contains F, then rule (2) or (4) must be used in $¢\alpha_i\$ \rightarrow ¢\alpha_{i+1}\$$.
(c) If α_i does not contain F, then (1) or (3) is used.
(d) Either α_n contains F but neither (2) nor (4) is applicable or else α_n does not contain F and neither (1) nor (3) is applicable.

$\alpha_n \neq u$; otherwise w would be in the 0L language. Thus if α_n contains F, $¢\alpha_n\$ \rightarrow \beta_1 N \beta_2$ by ($N3$) or ($N6$) and $\beta_1 N \beta_2 \rightarrow^* N$ by ($N7$) and ($N8$). And if α_n does not contain F, $¢\alpha_n\$ \rightarrow N\beta_2$ by ($N1$), ($N2$), ($N4$), or ($N5$), and $N\beta_2 \rightarrow^* N$ by ($N8$).

Thus, if $w \in A_0^*$ but w is not in the 0L language, then $¢w\$ \rightarrow N$. By Theorem 1.2, this completes the proof. □

COROLLARY. *CRDL* $\cap$ (*CSL* $-$ *CFL*) $\neq \varnothing$.

PROOF. Consider $L = \{a^{2^n} \mid n \geq 0\}$, a strictly context-sensitive language. L is generated by $S = (\{a\}, \{a \rightarrow aa\}, a)$. □

Let us look at a defining CR system for the L of the proof of Theorem 2.6:

$$\begin{aligned}
¢aaaa &\rightarrow ¢aaF,\\
Faa &\rightarrow aF,\\
¢aa\$ &\rightarrow ¢a\$,\\
F\$ &\rightarrow \$,\\
¢a\$ &\rightarrow Y,\\
¢\$ &\rightarrow N,\\
¢aaa\$ &\rightarrow N,\\
Fa\$ &\rightarrow N,\\
Nx &\rightarrow N \qquad \text{for all characters } x,\\
xN &\rightarrow N \qquad \text{for all characters } x.
\end{aligned}$$

We point out that the restrictions on the strings $w_1, \ldots, w_n$ (the right sides of P) and u in Theorem 2.6 can be relaxed somewhat, at the expense of complicating the construction. We could have merely stipulated that none of the w's be prefix of another, without any restriction on u. In that case we would have had to write the rules in such a way that no left side beginning with ¢ is a prefix of $¢u$.

3. *Inherently Ambiguous Languages*

THEOREM 3.1. (*CFL* $-$ *UCFL*) $\cap$ *CRDL* $\neq \varnothing$.

PROOF. Let $L = \{a^i b^j c^k \mid i, j, k \geq 1 \text{ and } i = j \text{ or } j = k\}$, a well-known inherently ambiguous CFL. L is shown to be in CRDL by a defining CR system T in which A_1 has a, b, c, ¢, and \$ and finitely many variants of a, b, c, and \$, which we indicate by placing subscripts on them. T is somewhat akin to the system constructed in the proof of Theorem 2.5.

T is designed to work on strings that begin with ¢, continuing with a's, then b's, then c's, and terminating with \$. A computation proceeds by a marker moving from left to right in the string; the marker is not a symbol, but its presence is indicated by a subscript on one of the letters. We think of the subscript as moving to the right from one letter to another, having come into existence at the ¢ and vanishing at the \$; the subscript changes as it moves. Where X equals the set of possible such subscripts on a, b, c, each $x \in X$ contains information about how many a's, how many b's, and how many c's it has seen since it "began its existence" at the ¢; it counts only up to 3, so the possibilities are 0, 1, 2, or at least 3. After it has seen its first b (first c), it also records whether it has seen an even or odd number of a's (of b's). As the subscript goes from left to right, the string must shrink, so that the rules are contracting; the information in the subscript is how many a's, b's, and c's it has seen since its origin at ¢, not how many are left by the time it terminates at \$.

Information is also accumulated at the \$. For $z \in Z$, the set of subscripts on \$, z may indicate that the number of a's in the original string (i.e., the string being tested for membership in L) was equal or unequal to the number of b's; and it may indicate that the number of b's in the original string was equal or unequal to the number of c's. This subscript may be modified when a subscript of the X variety comes to (and vanishes at) the \$, in effect, updating the information about the original string.

X is finite although large, and we shall not enumerate all its elements. Z has nine members. The \$ with no subscript at all indicates no information.

In the presentation of the rules of T, q, r, and s, each stands ambiguously for a, b, or c, with the understanding that alphabetic order in left and right sides of rules may not be violated; for example, if $q = b$, then $r = b$ or c but not a; also, if q occurs after b, then $q = b$ or c but not a. Each type of rule listed includes finitely many rules for different values of the subscripts and p, q, and r. The types themselves are arranged in five groups.

Group I

(1) $¢a^ib^jc^k\$_z \to ¢\$_{z'}$, $i, j \le 1$, $k \le 2$, $i + j + k \ge 1$,
(2) $¢a^ibb\$_z \to ¢\$_{z'}$, $i \le 1$,
(3) $¢aa\$_z \to ¢\$_{z'}$.

Group II

(1) $¢aaq \to ¢a_xq$,
(2) $¢abbq \to ¢b_xq$,
(3) $¢abccc \to ¢c_xc$,
(4) $¢accc \to ¢c_xc$,
(5) $¢bbq \to ¢b_xq$,
(6) $¢bccc \to ¢c_xc$,
(7) $¢ccc \to ¢c_xc$.

Group III

(1) $a_xaaq \to aa_{x'}q$,
(2) $a_xabbq \to ab_{x'}q$,
(3) $a_xabccc \to ac_{x'}c$,
(4) $a_xaccc \to ac_{x'}c$,
(5) $a_xbbq \to ab_{x'}q$,
(6) $a_xbccc \to ac_{x'}c$,
(7) $a_xccc \to ac_{x'}c$,

(8) $b_x bbq \rightarrow bb_{x'}q$,
(9) $b_x bccc \rightarrow bc_{x'}c$,
(10) $b_x ccc \rightarrow bc_{x'}c$,
(11) $c_x ccc \rightarrow cc_{x'}c$.

Group IV

(1) $q_x r\$_z \rightarrow q\$_{z'}$,
(2) $q_x rs\$_z \rightarrow q\$_{z'}$, for $r \neq s$,
(3) $q_x rr\$_z \rightarrow qr\$_{z'}$,
(4) $q_x qrr\$_z \rightarrow qr\$_{z'}$, for $q \neq r$,
(5) $a_x abc\$_z \rightarrow a\$_{z'}$,
(6) $a_x abcc\$_z \rightarrow ac\$_{z'}$.

Group V

(1) $¢\$_z \rightarrow Y$, provided z contains the information that, in the original string, either the number of a's equals the number of b's or the latter equals the number of c's.
(2) $¢\$_z \rightarrow N$, where z contains contrary information.
(3) $ba \rightarrow N$, $ca \rightarrow N$ and $cb \rightarrow N$.

We must specify how the values of the subscripts on the right sides are determined. We specify this only in a few sample types, leaving the details of the remaining rules to the reader.

In II(1), x contains the information that the number of a's seen is two; in II(2) that the number of a's seen is one and the number of b's seen is two; in II(7) that the number of a's and b's seen is zero and the number of c's seen is two.

In III(7), x' contains, in addition to the information that x has (which is only about the number of a's), the information that it has seen zero b's and two c's. In III(1), x' will be the information that it has seen at least three a's, regardless of x, which may be the information that it has seen two a's or that it has seen at least three a's.

In IV(1), if $q = r = c$, z represents no information, and x is the information that it has seen, one a and one b, then z' is the information that the number of a's equals the number of b's but not the number of c's in the original string. On the other hand, if z already represents the information that the number of a's does not equal the number of b's in the original string with no information about b's and c's, where $q = r = c$, then z' represents the information that the number of a's does not equal the number of b's, which does not equal the number of c's in the original string.

No rules are listed in Group IV in cases in which the z subscripts on the left side represent information that contradicts information in the remainder of the left side. Such would be the string $a_x abc\$_z$, where z indicated that the number of a's equals the number of b's in the original string; such a string would not have reduced down to one in which there is a single b and several a's. Another such example would be $c_x cc\$_z$, where x indicates that the number of a's and the number of b's in the present scan have been of opposite parity, and z indicates that the number of a's equals the number of b's in the original string.

To prove that T has the CR property, we observe that the left side of every rule is a string that begins with ¢ or a subscripted a, b, or c and contains none of these characters elsewhere. It follows that there can be no proper overlap among these left sides. If one is a substring of another, then it must be a prefix of it. That this never occurs can be verified by inspection. It remains to prove that T is a decision procedure for L.

LEMMA 6. *$¢a^{i_1}b^{i_2}c^{i_3}\$_z \to^* ¢a^{j_1}b^{j_2}c^{j_3}\$_{z'}$, where j_1, j_2, j_3, z' are as follows:*

(1) *For $k = 1$, 2, or 3, $j_k = [i_k/2]$, with the following exceptions: If $i_3 = 2$* and *$i_1, i_2 \leq 1$, then $j_3 = 0$; if $i_3 = 0$, $i_1 \leq 1$, and $i_2 = 2$, then $j_2 = 0$; and if $i_2 = i_3 = 0$ and $i_1 = 2$, then $j_1 = 0$.*

(2) *z' represents no information about the number of a's and b's (the number of b's and c's) if and only if $i_1, i_2 \geq 3$ ($i_2, i_3 \geq 3$), i_1 and i_2 (i_2 and i_3) are of the same parity, and z represents no information about the number of a's and b's (b's and c's).*

The proof, though detailed, is straightforward and reminiscent of certain considerations in the proof of Theorem 2.5. Accordingly, it is left to the reader.

LEMMA 7. *$¢a^{i_1}b^{i_2}c^{i_3}\$ \to^* Y$ if $i_1 = i_2$ or $i_2 = i_3$. It yields N otherwise.*

PROOF. By applying Lemma 6 successively we get $¢a^{i_1}b^{i_2}c^{i_3}\$ \to ¢\$_z$, where z contains the information about whether $i_1 = i_2$ and whether $i_2 = i_3$.

With the aid of Theorem 1.2, we conclude that T is a decision procedure for L. □

By a slight modification in the construction in this proof we come up with an alternative proof of the Corollary to Theorem 2.6, showing that $\{a^nb^nc^n\} \in$ CRDL.

Moreover, the technique used in designing system T is capable of fruitful generalization. Let us describe this technique in fresh terms: We start with $¢w\$$, where $w \in A_0^*$ is a word to be tested. We scan w with a finite automaton from ¢ to \$, ending up with a piece of information that is recorded in a finite-state memory located at the \$. This finite automaton is permitted to scan repeatedly; however, the string scanned is always shortened by a scan to a fraction of its length, the fraction being no more that a constant r, when $0 < r < 1$ ($r = \frac{1}{2}$ in the above construction). An important feature of the construction is that the string after the scan shares certain properties with the string before the scan so that the information gathered during all future scans is relevant information about the original string w.

For example, let $L \subseteq \{a, b, c\}^*$ be the set of strings w such that $|w_a| = |w_b|$ or $|w_b| = |w_c|$ ($|w_a|$ is the number of a's in w). That $L \in$ CRDL follows by a construction somewhat more elaborate because the letters need not be in order in the string. We do not give a detailed construction, but a brief suggestion. Again, we let a subscript on a letter indicate the presence of a marker (or finite automaton) that is scanning the string. One rule might be

$$a_xbcbaa \to abc_{x'}a,$$

where x represents the information that there have been an odd, even, and even number of a's, b's, and c's, respectively, in the string so far, and x' represents the information that there have been an even, even, and odd number of a's, b's, and c's, respectively. Similarly,

$$a_xbcbcb \to abc_xb$$

would be a rule for any x. With rules such as these, the number of a's, b's and c's could be counted in the appropriate way so as to determine whether or not the original string is in L. We omit further details and call upon the imagination of the reader.

It is not difficult to generalize: For $w \in A^*$, where $A = \{a_1, \ldots, a_n\}$, and B is any Boolean function of the $n(n+1)/2$ propositions of the form $|w_{a_i}| = |w_{a_j}|$, for $1 \le i \le j < n$, put $L = \{w \mid w \text{ satisfies } B\}$. Then $L \in$ CRDL. However, we omit the details, and we omit discussion of any further generalizations, which naturally suggest themselves.

4. *The Limits of CRL*

It is widely believed, although (as far as we know) no one has proved that there are CFLs whose membership problem cannot be solved in linear time. From this conjecture it would follow that CFL − CRL $\neq \varnothing$. We have a conjecture that is more specific than this. Let $L_p = \{xx^R \mid x \in \{a, b\}^+\}$, that is, the set of even-length palindromes over $\{a, b\}$. Our conjecture is that $L_p \notin$ CRL, and it is the purpose of this section to present in some detail our reasons for believing that it is true.

We begin our discussion of the conjecture by trying to imagine how the defining CR system for L_p might work. We naturally think of canceling a character in the ith position with the same character in the $(2n - i + 1)$st position of a word w of length $2n$ suspected of being an even-length palindrome. For a Thue system this is possible only if the two characters are near enough to each other to be covered by the left side of a rule. For the purposes of this informal discussion let us assume $i = n$, so that the nth and $(n+1)$st characters are the first to be canceled if identical.

Let us refer to the Thue system that we are seeking to construct as T. Before such cancellation can occur, T must somehow find the middle of the string. The hope is that we can design the system so that it will start at (say) the left end of $t_1 w t_2$ (recall the definition of CR language in Section 1) and, by a sequence of rule applications, get a certain designated character from $A_1 - A_0$ to appear for the first time at the middle of the string. This character would appear in all cancellation rules, guaranteeing that canceling could occur only at the middle.

The system T must not have the capability of canceling characters of A_0, except at the center of w. Moreover, until the center is found, exact information about all parts of the string w must be preserved.

As mentioned in Section 1, in CR Thue systems length-preserving rules are superfluous. Indeed, linear-time computation on strings is possible in Thue systems precisely because at every step the length of the string is being decreased. We can assume in this discussion, therefore, that the Thue system T we are seeking for L_p has no length-preserving rules.

We use the word "descendant" to denote a string that results from a given string by one or more applications of length-reducing rules of T. It is not difficult to imagine how a descendant of $t_1 w t_2$ could have a shortened version of the w part and yet have all the information about w. What we have in mind are rules such as

$$x_1 abc\binom{d}{e} \to a\binom{b}{c}\binom{d}{e}x_2,$$

where x_1, x_2, $\binom{d}{e}$, and $\binom{b}{c} \in A_1 - A_0$ and $a, b, c, d, e \in A_0$; the single characters $\binom{d}{e}$ and $\binom{b}{c}$ represent the substrings de and bc, respectively, in the original string w. The characters x_1 and x_2 would be like a scanning head sweeping over the string, the subscript change indicating an update of information (as in some of the examples in the preceding section).

However, if the technique for finding the middle of w requires visiting parts of w (or, as we should say, the image of w in the descendants of $t_1 w t_2$) arbitrarily many times for w arbitrarily long, then this compression technique would not work. It would require compressing arbitrarily much information into a single character of $A_1 - A_0$, which would require an infinite alphabet and an infinite set of rules in T.

The technique we have in mind would, it seems to us, need to be deterministic; only one rule could be applied to any descendant of t_1wt_2 and in only one place. If this is so, then it is impossible to utilize a method depending on parallel action. For example, finding the middle by a method suggested by the well-known solution to the firing squad problem would not be realizable in a Thue system.

We are led to believe that we can prove the following helpful hypothetical statement: Any technique using length-decreasing rules of a Thue system for finding the middle of w could also be used by a Turing machine (with a single head) to find the middle of a blank portion of tape flanked by two marked squares. The latter would be accomplished in linear time since the former is accomplished in linear time.

The next step of our argument would be the easy exercise in automata theory that the existence of a (single-head) deterministic Turing machine that could find the middle of a tape in linear time implies the existence of such a Turing machine to recognize the language $\{a^nb^n \mid n \geq 1\}$ in linear time. We would then cite the known result (see [9]) that any language recognized in linear time by a (single-head) deterministic Turing machine is regular. Since $\{a^nb^n \mid n \geq 1\}$ is not regular, we would then have a contradiction.

These thoughts are at the basis of our conjecture that L_p is not a CR language. In summary, we feel that a Thue system for L_p acting on t_1wt_2 would have to find the middle of w before losing any information about w. At the same time, we feel it would be impossible for a Thue system to do this deterministically by length-decreasing rules.

5. *The Family CRCL*

In this section we demonstrate, among other things, the utility of the control characters (i.e., the characters of $A_1 - A_0$) in the Thue systems for CR-decidable languages by proving that CRDL $-$ CRCL $\neq \emptyset$. We begin with an almost obvious observation.

THEOREM 5.1. *CRCL* $\subseteq$ *CRL.*

PROOF. For $L \in$ CRCL, $L \subseteq A_0^*$, there exists a Thue system T_0 and irreducible strings $x_1, \ldots, x_k$ such that $L = [x_1] \cup \cdots \cup [x_k]$. The Thue system $T_1 = T_0 \cup \{(¢x_i\$ \to Y) \mid 1 \leq i \leq k\}$ shows that $L \in$ CRL, since T_1 is CR and for all $w \in A_0^*$, $¢w\$ \to_{T_1}^* Y$ if and only if $w \in L$. □

THEOREM 5.2. *CRDL* $-$ *CRCL* $\neq \emptyset$.

PROOF. Let $L = \{a^mb^{2m}ca^nb^n \mid m, n \geq 1\}$. That $L \in$ CRDL follows from Theorem 2.2, since L is deterministic.

Assume now that L is a CRCL. Then there exists a CR Thue system $T = \{l_i \to r_i \mid l \leq i \leq p\}$ such that $L = [x_1]_T \cup [x_2]_T \cup \cdots \cup [x_k]_T$ where $x_1, \ldots, x_k$ are irreducible strings.

Now let w be any string of the form $a^{m_1}b^{2m_1}ca^{n_1}b^{n_1}$, where $m_1, n_1 > \max(|l_1|, \ldots, |l_p|, |x_1|, \ldots, |x_k|)$. Since w is not irreducible, there is at least one reduction rule $l_j \to r_j$ that can be applied to w. Any application of this rule to w, being congruent to $w \bmod T$, must also be in L.

Case 1. l_j contains c. Then $l_j = b^{h_1}ca^{h_2}$; $h_1, h_2 \geq 0$. But the replacement $a^m b^{2m-h_1} r_j a^{n-h_2} b^n$ cannot be in L since $|r_j| < |l_j|$.

Case 2. l_j does not contain c. Then $l_j = a^{h_1}b^{h_2}$; $h_1, h_2 \geq 0$. We have two replacements $a^{m-h_1} r_j b^{2m-h_2} ca^n b^n$ and $a^m b^{2m} ca^{n-h_1} r_j b^{n-h_2}$. The only way for both of these to be in L would be for $r_j = a^{h_1}b^{h_2} = l_j$, which violates $|r_j| < |l_j|$.

Thus $L \notin$ CRCL. □

COROLLARY. *CRL − CRCL* $\neq \emptyset$.

PROOF. Obviously, CRDL $\subseteq$ CRL. □

THEOREM 5.3. *CRCL* $\cap$ *(UCFL − DCFL)* $\neq \emptyset$.

PROOF. Consider the following Thue system T over the alphabet $\{a, b, b_1, b_2, A, B, c, d\}$:

$$\{bbc \to cb_1, \quad aacb_1 \to A, \quad aaAb_1 \to A, \quad bbd \to db_2, \quad aadb_2b_2 \to B, \quad aaBb_2b_2 \to B\}$$

T is CR by the Corollary to Theorem 1.1. Now the following two sets are context free:

$$[A]_T = \{a^{2i}Ab_1^i \mid i \geq 0\} \cup \{a^{2i}b^{2j}cb_1^{i-j} \mid i > j > 0\} \cup \{a^{2i}b^{2i}c \mid i \geq 1\},$$
$$[B]_T = \{a^{2i}Bb_2^{2i} \mid i \geq 0\} \cup \{a^{2i}b^{2j}db_2^{2i-j} \mid 2i > j > 0\} \cup \{a^{2i}b^{4i}d \mid i \geq 1\}.$$

Thus, $[A]_T \cup [B]_T$ is context free and, by an easy exercise, is unambiguous. But $([A]_T \cup [B]_T) \cap \{a, b, c, d\}^* \notin$ DCFL, showing that $[A]_T \cup [B]_T \notin$ DCFL (since DCFL's are closed under intersection with regular sets). □

Theorem 5.3 answers a question raised by Boasson.

THEOREM 5.4. *(CFL − UCFL)* $\cap$ *CRCL* $\neq \emptyset$.

PROOF. The Thue system we use here is obtained by altering the construction in the proof of Theorem 3.1. This time we allow the subscript to go both from left to right and from right to left so as to ensure that the string being worked on has exactly one subscript each time. We begin with $¢_{z_0}w\$$, where $w \in a^*b^*c^*$ is the word being tested.

The subscript z_0 on ¢ means zero information. It is this subscript that "travels" right as before. But when it reaches the \$, it commences in the reverse direction until it returns to the ¢, whereupon it again moves right, and so on. Since no information is left at the \$ as in the Theorem 3.1 construction, all essential information must be retained in the one traveling subscript. After the subscript has traveled from ¢ to \$, or from \$ to ¢, the string is approximately half what it was before, as in the Theorem 3.1 construction.

There are two sets of subscripts: X, the set of right-going subscripts, and Z, the set of left-going subscripts. In the following, $u \in a^*b^*c^*$; p, q, and r each represents a, b, or c, with the understanding that letters in the left side (and hence right side) of each rule are in alphabetic order. There are eight types of rules:

I. $¢_z u\$ \to ¢\$_x$ for all $u \neq \lambda$ with no dls (double-letter substring, i.e., *aa*, *bb*, or *cc*) except possibly as a suffix.
II. $p_x u\$ \to p\$_{x'}$ for all u as in I.
III. $¢_z uqr \to ¢q_x r$ for all u ending in q but without any dls.
IV. $p_x uqr \to pq_{x'} r$ for all u as in III.

(This set of rules is identical to the set of rules in group III in the Theorem 3.1 construction.)

V. $¢u\$_x \to ¢_z\$$ for all $u \neq \lambda$ with no dls except possibly as a prefix.
VI. $¢up_z \to ¢_{z'}p$ for all u as in V.
VII. $pu\$_x \to p_z\$$ for all u beginning with p but without any dls.
VIII. $puq_z \to p_{z'}q$ for all u as in VII.

In all these rules we assume that the information on the left side is consistent. The subscript on the right side is determined from the left side, as in the Theorem 3.1 construction.

A subscript on ¢ or \$ indicates whether or not $|w_a| = |w_b|$ or no information and whether or not $|w_b| = |w_c|$ or no information, where w is the original string. A subscript on a, b, or c indicates this also, but it may also indicate that $|w'_a|$, $|w'_b|$, or $|w'_c|$ is odd or even or no information (27 possibilities), where w' is the string at the beginning of the scan.

Since no left side overlaps with, is identical to, or is a substring of another, this system is CR. It has as some of its irreducible strings, strings of the forms $¢_z\$$ and $¢\$_x$. The congruence classes of these will contain strings of $¢a^*b^*c^*\$$ modified by having exactly one character subscripted.

Take $L = \bigcup_{z \in F} [¢_z\$] \cup \bigcup_{x \in F} [¢\$_x]$, where F is the set of all subscripts indicating that $|w_a| = |w_b|$ or that $|w_b| = |w_c|$. That L is a CFL follows from the fact that L is the union of eight sets, each of which is context free. Three of these sets are given here:

(1) $\{¢_z a^{i_1} b^{i_2} c^{i_3}\$ \mid i_1 = i_2$, and z does not indicate that $|w_a| \neq |w_b|$; or $i_2 = i_3$, and z does not indicate that $|w_b| \neq |w_c|\}$.
(2) $\{¢a^{j_1} a_x a^{k_1} b^{i_2} c^{i_3}\$ \mid 2(j_1 + 1) + k_1 = i_2$, and x does not indicate $|w_a| \neq |w_b|$; or $i_2 = i_3$, and x does not indicate that $|w_b| \neq |w_c|\}$.
(3) $\{¢a^{i_1} b^{j_2} b_x b^{k_2} c^{i_3} \mid 2i_1 = 2(j_2 + 1) + k_2$, x does not indicate $|w_a| \neq |w_b|$, and k_2 has the same parity as $|w'_a|$ as indicated by x; or $2(j_2 + 1) + k_2 = i_3$, and x does not indicate that $|w_b| \neq |w_c|\}$.

Note that where z_0 denotes no information, $L \cap ¢_{z_0} a^*b^*c^*\$ = \{¢_{z_0} a^{i_1} b^{i_2} c^{i_3}\$ \mid i_1 = i_2$ or $i_2 = i_3\}$, which is inherently ambiguous. It follows that L is inherently ambiguous, since the unambiguous CFLs are closed under intersection with a regular set (see [8, p. 243]). □

THEOREM 5.5. *CRCL − CFL* $\neq \varnothing$.

PROOF. Let $A = \{a, ¢, \$, F, Y, N\}$ and T be the system given in the proof of the Corollary to Theorem 2.6. Then $[Y]_T \notin$ CFL since $[Y]_T \cap ¢a^*\$ = \{¢a^{2^n}\$ \mid n \geq 0\} \notin$ CFL (CFL being closed under intersection with regular sets). □

An interesting, although perhaps not vital, question is whether REG $\subseteq$ CRCL. Our attempts to answer this question failed. We could not even settle the question about whether $(\{a, b\}^p)^* \in$ CRCL for all values of p; we know only that there is an affirmative answer for $p = 2$ and $p = 3$.

It seems that regular languages ought to be congruential. The easiest way to make this so is to introduce a generalization of the CR property. Accordingly, a Thue system T is *almost confluent* (in the sense of Book [4]) if $x \leftrightarrow_T^* y$ implies that for some $z_1, \ldots, z_n$, $n \geq 1$, $x \to_T^* z_1$, $y \to_T^* z_n$, and for each i, $1 \leq i \leq n-1$, $|z_i| = |z_{i+1}|$ and $z_i \leftrightarrow_T z_{i+1}$. A congruence class in such a system may have several minimum-length strings connected by length-preserving rules. Almost confluence is called *confluence* in some papers (see [1] and [11]). We do not discuss this notion any further in this paper.

6. *Undecidability Results*

We begin by citing two results from O'Dunlaing's [14] dissertation (see also [13]):

THEOREM 6.1. *It is undecidable whether the intersection of a given congruence class of a given CR Thue system and a given regular set is empty (finite).*

THEOREM 6.2. *It is undecidable whether a given congruence class of a given CR Thue system is a CFL.*

THEOREM 6.3. *The emptiness and finiteness problems for CR languages are undecidable.*

PROOF. We demonstrate that, if either problem of Theorem 6.3 were decidable, there would be an algorithm for the corresponding undecidable problem of Theorem 6.1. Accordingly let $[x]$ be a congruence class in a given CR system T; without loss of generality, assume x is the shortest-length string in $[x]$. Let R be a regular set sharing with T a common alphabet Σ. Where $M = (K, \Sigma, \mathrm{Tr}, q_0, F)$ is a deterministic finite automaton accepting R, define the character sets

$$\begin{aligned} S_1 &= \{Q_i \mid q_i \in K\}, \\ S_2 &= \{¢, \$, Y\}, \\ S_3 &= \{a_1 \mid a \in \Sigma\} \cup \{a_2 \mid a \in \Sigma\}, \\ \Sigma' &= \Sigma \cup S_1 \cup S_2 \cup S_3. \end{aligned}$$

Let T' be the result of augmenting T with the rules (called Δ rules)

$$\begin{array}{ll} Q_i a_1 a_2 \to aQ_j & \text{for all } q_i,\ q_j,\ \ a \text{ such that } \mathrm{Tr}(q_i, a) = q_j, \\ ¢xQ_f\$ \to Y & \text{for all } q_f \text{ in } F. \end{array}$$

Note that T' is a CR system, for T is assumed to be CR and the left side of no Δ rule overlaps with, is a substring of, or is identical to the left side of any rule of T or another Δ rule.

Let $h\colon \Sigma^* \to S_3^*$ be the homomorphism determined by $h(a) = a_1a_2$ for each $a \in \Sigma$.

Let $L' = \{w' \mid ¢Q_0w'\$ \to_{T'}^{*} Y\}$. Thus, $L' \in$ CRL, with $A_0 = S_3$, and we shall prove (1) that, for all $w \in \Sigma^*$, $¢Q_0h(w)\$ \to_{T'}^{*} Y$ if and only if $w \in [x]_T \cap R$ and (2) that, for $w' \in S_3^* - (h(\Sigma))^*$, $¢Q_0w'\$ \not\to_{T'}^{*} Y$. Since h is a one-one homomorphism, (1) and (2) imply that $[x]_T \cap R$ is empty or finite if and only if L' is empty or finite, respectively, and will complete our proof.

LEMMA 8. *$Q_ih(w) \to_{T'}^{*} wQ_j$ if and only if $\mathrm{Tr}(q_i, w) = q_j$.*

PROOF. By induction on $|w|$: For $|w| = 0$, the result is immediate. Assume it holds for all w of length up to k. Consider a word w of length $k + 1$.

Assume $\mathrm{Tr}(q_i, w) = q_j$. Where $w = ay$, $\mathrm{Tr}(q_i, a) = q_{i1}$ and $\mathrm{Tr}(q_{i1}, y) = q_j$, we have $Q_{i1}h(y) \to_{T'}^{*} yQ_j$ by the induction hypothesis. Since $Q_ia_1a_2 \to aQ_{i1}$ is a Δ rule, $Q_ia_1a_2h(y) \to_{T'}^{*} ayQ_j$.

Assume now that $Q_ih(w) \to_{T'}^{*} wQ_j$. If $\mathrm{Tr}(q_i, w) = q_k \neq q_j$, then, by what has just been proved, $Q_ih(w) \to_{T'}^{*} wQ_k$, which implies, by the CR property, that wQ_k and wQ_j have a common descendant. But this is impossible since, by inspection of the rules, both Q_k and Q_j are unchangeable in the reductions of wQ_k and wQ_j, respectively. Hence, $Q_k = Q_j$.

LEMMA 9. *For all $w \in \Sigma^*$, $Q_i h(w) \rightarrow^*_{T'} xQ_j$ if and only if $Tr(q_i, w) = q_j$ and $w \in [x]_T$.*

PROOF. Assume $\mathrm{Tr}(q_i, w) = q_j$ and $w \in [x]_T$. By Lemma 8, $Q_i h(w) \rightarrow^*_{T'} wQ_j$. Since T is CR and x has minimal length in $[x]_T$, $w \rightarrow^*_T x$. Hence, $Q_i h(w) \rightarrow^*_{T'} xQ_j$.

Assume now that $Q_i h(w) \rightarrow^*_{T'} xQ_j$. Let $\mathrm{Tr}(q_i, w) = q_k$. By Lemma 8, $Q_i h(w) \rightarrow^*_{T'} wQ_k$. By the reasoning used in the proof of Lemma 8, wQ_k and xQ_j cannot have a common descendant unless $q_k = q_j$.

Since T' is CR, there exists an x' such that $wQ_j \rightarrow^*_{T'} x'$ and $xQ_j \rightarrow^*_{T'} x'$. But since $w, x \in \Sigma^*$, there is no reduction rule that can change Q_j in either of the above. Hence, $x' = x'_1 Q_j$ for some x'_1 and no Δ rule is involved. Since x is irreducible in T, $x'_1 = x$, and hence $w \in [x]_T$.

LEMMA 10. *For $w \in \Sigma^*$, $¢Q_0 h(w)\$ \rightarrow^*_{T'} Y$ if and only if $w \in R \cap [x]_T$.*

PROOF. The "if" part follows directly from Lemmas 8 and 9. So assume $¢Q_0 h(w)\$ \rightarrow^*_{T'} Y$, which implies $¢Q_0 h(w)\$ \rightarrow^*_{T'} ¢xQ_f\$$ for some $q_f \in F$. Inspection of the rules shows that neither ¢ nor \$ is changed in the latter. Hence, by Lemma 9, $\mathrm{Tr}(q_0, w) = q_f$, and $w \in [x]_T$.

From the construction of T' it is clear that, for $w' \in S_3^* - (h(\Sigma))^*$, $¢Q_0 w'\$$ cannot be reduced to Y in T'. Hence, Lemma 10 concludes the proof of Theorem 6.3. □

We need another closure result:

THEOREM 6.4. *CRL and CRDL are both closed under left quotient and right quotient, with a single string.*

PROOF. Let $L \in \mathrm{CRL}$, $L \subseteq A_0^*$. Thus for some CR system T with alphabet A_1, $A_0 \subseteq A_1$, $t_1, t_2 \in (A_1 - A_0)^*$, and $Y \in A_1 - A_0$, $L = \{w \in A_0^* \mid t_1 w t_2 \rightarrow^* Y\}$. Let $x \in A_0^*$, and let $M = |x|$.

Construct T' with alphabet $A'_1 = A_1 \cup \{V, Z\}$, where V and Z are new. Let $T' = T \cup \{VZ|^m \rightarrow x\}$. T' is CR and, for all $w \in A_0^*$, $t_1 wVZ|^m t_2 \rightarrow^*_{T'} Y$ if and only if $t_1 wxt_2 \rightarrow^*_T Y$, showing that $L/\{x\}$, $\in$ CRL. Similarly, if $L \in$ CRDL, $L/\{x\} \in$ CRDL. We can prove the analogous results about $\{x\}\backslash L$ for both CRL and CRDL. □

THEOREM 6.5. *If CFL − CRL $\neq \varnothing$ (CFL − CRDL $\neq \varnothing$), then it is undecidable for a given context-free language L whether $L \in$ CRL ($L \in$ CRDL).*

PROOF. Theorem 6.5 follows immediately from Theorem 6.4, Theorem 2.1, and a theorem of Greibach (see [10, p. 205]) that implies the following: If P is any family of languages such that (1) CFL $- P \neq \varnothing$, (2) REG $\subseteq P$, and (3) if $L \in P$, then $L/\{a\} \in P$ for any character a, then it is undecidable for a given $L \in$ CFL whether $L \in P$. □

THEOREM 6.6. *It is undecidable, for a given $L \in$ CRL, whether $L \in$ CFL.*

PROOF. An algorithm for this problem would give us an algorithm for the undecidable problem of Theorem 6.2, since every congruence class of a CR system is a CR language. □

At present we are unable to prove analogs of Theorems 6.3 and 6.6 for CRDL.

REFERENCES

Note: References [2], [6], [12], [15], and [17] are not cited in text.

1. BERSTEL, J. Congruences plus que parfaites et langages algébriques. In *Seminair d'Informatique Theorique*, Institut de Programmation, Paris, France, 1976–1977, pp. 123–147.
2. BOASSON, L. Dérivations et réductions dans les grammaires algébriques. In *Automata, Languages, and Programming*. Lecture Notes in Computer Science, vol. 85. Springer-Verlag, New York, 1980, pp. 109–118.
3. BOASSON, L., AND SENIZERGUES, G. NTS languages are deterministic and congruential. *J. Comput. Syst. Sci. 31* (1985), 332–342.
4. BOOK, R. V. Confluent and other types of Thue systems. *J. ACM 29*, 1 (Jan. 1982), 171–182.
5. COCHET, Y., AND NIVAT, M. Une généralisation des ensembles de Dyck. *Isr. J. Math. 9* (1971), 389–395.
6. FROUGNY, C. Une famille de langages algébriques congruentiels: Les langages à nonterminaux separés. Thèse de 3ème cycle, Laboratoire Informatique Théorique et Programmation (1980).
7. GELLER, M. M., HARRISON, M. A., AND HAVEL, I. M. Normal forms of deterministic grammars. *Discrete Math. 16* (1976), 313–321.
8. HARRISON, M. A. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, Mass., 1978.
9. HARTMANIS, J. Computational complexity of one-tape Turing machine computations. *J. ACM 15*, 2 (Apr. 1968), 325–339.
10. HOPCROFT, J. E., AND ULLMAN, J. D. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Mass., 1979.
11. KAPUR, D., AND NARENDRAN, P. Almost-confluence and related properties of Thue systems. Rep. 83CRD258, General Electric CR&D Center, Schenectady, N.Y., 1983.
12. NARENDRAN, P. Church-Rosser and related Thue systems. Doctoral dissertation, Rensselaer Polytechnic Institute, Troy, N.Y., 1984.
13. NARENDRAN, P., O'DUNLAING, C., AND ROLLETSCHECK, H. Complexity of certain decision problems about congruential languages. *J. Comput. Syst. Sci. 30* (1985), 343–358.
14. O'DUNLAING, C. Finite and infinite regular Thue systems. Doctoral dissertation, Univ. of California at Santa Barbara, Santa Barbara, Calif., 1981.
15. O'DUNLAING, C. Infinite regular Thue systems. *Theoret. Comput. Sci. 25* (1983), 171–192.
16. SALOMAA, A. *Formal Languages.* Academic Press, Orlando, Fla., 1973.
17. THUE, A. Probleme uber Veranderungen von Zeichenreichen nach gegeben Regeln. *Skr. Vidensk. Kristiania, I. Math. Naturvidensk. Klasse 10* (1914).

RECEIVED APRIL 1986; REVISED JUNE 1987; ACCEPTED JUNE 1987