On the Solvability of a Word Problem for Restricted Semigroups[†]

Lawrence Snyder

Research Report #102

Department of Computer Science
Yale University
New Haven, Connecticut 06520

[†] This research was supported by the Office of Naval Research under Grant N00014-75-C-0752.

ON THE SOLVABILITY OF A WORD PROBLEM FOR RESTRICTED SEMIGROUPS

Lawrence Snyder
Department of Computer Science
Yale University
10 Hillhouse Ave, New Haven CT 06520

ABSTRACT: A class of semigroups called 1-restricted semigroups is defined in which there is at most one relation per generator and at most one occurrence of the generator symbol in any word equivalent to it. The word problem for 1-restricted semigroups is shown to be decidable.

Since Post's [1] original work on the subject, semigroup word problems have been a source of interesting computational problems. With the general problem having an undecidable word problem, interest has shifted to the complexity of word problems for restricted semigroups [2],[3]. In this latter paper, Strong, Maggiolo-Schnetti and Rosen [3] abstracted an optimization problem in terms of a word problem for restricted semigroups and conjectured its decidability. In this report a partial answer is given in the affirmative.

Let A be an alphabet. A *restricted semigroup presentation* $S = (A,P)$ where P is a finite set

$$P \subseteq \{\langle a,w\rangle \mid a \in A,\ w \in A^*\}$$

such that for all $\langle a,w\rangle, \langle a',w'\rangle \in P$, $a=a'$ implies $w=w'$. The pairs $\langle a,w\rangle$ are customarily written as $a \equiv w$ and the symbol a is called a *generator*. Thus a restricted semigroup has at most one equivalence per generator and no other relations. For semigroup $S = (A,P)$ and words $\alpha,\beta \in A^*$, α *derives* β in S written

$$\alpha \underset{S}{\Longrightarrow} \beta$$

provided there exists $\langle a,w\rangle \in P$ such that either $\alpha=uav$ and $\beta=uwv$ or $\alpha=uwv$ and $\beta=uav$ for some $u,v \in A^*$. Since $\alpha \underset{S}{\Longrightarrow} \beta$ implies $\beta \underset{S}{\Longrightarrow} \alpha$, derivability induces an equivalence relation on the set of words; accordingly $\alpha \overset{*}{\underset{S}{\Longrightarrow}} \beta$ is customarily written as $\alpha \underset{S}{\equiv} \beta$ by abuse of notation and the semigroup name S is elided where no confusion can result.

The (uniform) *word problem* for restricted semigroups is to decide for any $S = (A,P)$ and words $u,v \in A^*$ whether or not $u \equiv v$. This problem is still open. In [3] this word problem is recast in what is more familiar terminology:

Lemma [3]: The word problem for restricted semigroups is equivalent to the intersection problem for a pair of context-free grammars which differ only in start symbol and are restricted to have one production per non-terminal (except that each non-terminal has an additional rule of the form $B \to b$, where b is a terminal distinct from all terminals corresponding to other non-terminals).

The systems of present interest are a subclass of restricted semigroups. A *1-restricted semigroup* $S = (A,P)$ is a restricted semigroup such that $\langle a,w\rangle \in P$ and $a \equiv \alpha a \beta$ implies $\alpha,\beta \in (A-\{a\})^*$. That is, any word equivalent to a generator contains at most one occurrence of that generator. In terms of the context-free formulation mentioned above, a 1-restricted semigroup corresponds to a context-free grammar in which no word derivable from a non-terminal contains more than one occurrence of that non-terminal. Thus, "pumping" is permitted but in a limited way.

In order to exhibit the objects just defined as well as to motivate parts of the subsequent development, consider the 1-restricted semigroup

$$S = (\{a,b,c,d,e\}, \{\langle a,cddddde\rangle, \langle b,ddddde\rangle, \langle d,cdcc\rangle, \langle e,ce\rangle\})$$

which corresponds to the context-free grammars

$$G_a = (\{a,b,c,d,e\},T,a,P)$$
$$G_b = (\{a,b,c,d,e\},T,b,P)$$

where

$$P = \{a \to cddddde,\ b \to ddddde,\ d \to cdcc,\ e \to ce\}$$

and the terminal productions have been deleted. We ask the word problem: Is $a \equiv b$? The answer is, yes, as can be shown by exhibiting a word in $L(G_a) \cap L(G_b)$. In particular, consider the two derivations in "parallel":

$$a \Rightarrow cddddde \Rightarrow cdcdccde \Rightarrow cdc^2dc^4d^3e \Rightarrow \ldots$$
$$b \Rightarrow ddddde \Rightarrow cdccdddde \Rightarrow cdc^2dcdc^2d^2e \Rightarrow \ldots$$
$$\Rightarrow cdc^2dc^4dc^8dc^{16}dc^{32}e$$
$$\Rightarrow cdc^2dc^4dc^8dc^{16}dc^{32}e \qquad \square$$

The objective of the remainder of the paper is to prove:

Theorem: The word problem for 1-restricted semigroups is decidable.

In the interest of economy, the proof is only sketched. The general logic of the argument is to construct a word in $L(G_A) \cap L(G_B)$ or show that none can exist. The construction involves carrying out a "parallel derivation" so that at each step a word in $L(G_A) \cap L(G_B)$ must include the symbols being generated. If at some point the derivation cannot be extended $L(G_A) \cap L(G_B) = \phi$.

A key tool in the construction is a special derivation dag, which will now be developed.

In the subsequent discussion the grammars G_A and G_B are assumed to be given and is abbreviated G_{AB}. Moreover, without loss of generality it may be assumed that there are no productions of the form $C \to \epsilon$ (where ϵ is the empty word), and no productions of the form $C \to C$ since equivalent problems can be formulated without these productions. Call a sequence of letters $C_1,\ldots,C_n$ a *cycle* if there are productions

$$C_i \to \alpha_i C_{i+1} \beta_i \qquad 1 \le i < n$$

and

$$C_n \to \alpha_n C_1 \beta_n .$$

Cycles have several properties:

(i) The rotation of a cycle is a cycle, i.e., $C_1,\ldots,C_n$ is a cycle if and only if
$$C_{(k \bmod n)+1},\ldots,C_{(k+n \bmod n)+1}$$
is a cycle $0 \le k < n$.

(ii) Two cycles that are not rotations of one another are disjoint, i.e., $C_1,\ldots,C_n,C'_1,\ldots,C'_m$ cycles implies for no i and j does $C_i = C'_j$.

(iii) Cycles *persist*, that is, $A \Rightarrow \ldots \Rightarrow \alpha C_i \beta \Rightarrow \ldots \Rightarrow \tau$ for C_i in cycle $C_1,\ldots,C_n$, then for some j, $\tau = \alpha' C_j \beta'$.

This last fact is extremely crucial in the proof.

A cycle $C_1,\ldots,C_n$ is *trivial* if n=1. We now state a useful simplification:

Lemma: For any 1-restricted G_{AB} there exists a 1-restricted G'_{AB} containing only trivial cycles such that $A \equiv B$ in G_{AB} if and only if $A \equiv B$ in G'_{AB}. The proof relies on the previously ennumerated facts and is constructive. It is complicated only by the fact that cycles can be entered at various points, thus care must be used in "collapsing" cycles. In the sequel G_{AB} is assumed to have only trivial cycles, and C_1 is called the cycle letter.

A *derivation dag* for G_{AB} is an oriented acyclic graph $D = (V,E)$ with vertex set V = the alphabet for G_{AB} and the edge set E defined by

$$E = \{\langle C, D_i \rangle \mid C \to D_1,\ldots,D_n \text{ is a production} \wedge C \ne D_i\}.$$

Evidently D is a dag since a graph cycle in D implies a letter cycle in G_{AB} -- but these are at most trivial and the second condition avoids introducing loops.

Notice that the dag may have multiple sources, but generally only a subset of these will be of interest (e.g., A and B initially). Let $(C_1,\ldots,C_n)$ be used to denote the subdag reachable from vertices $C_1,\ldots,C_n$.

A *reduced derivation dag* is a derivation dag containing only cyclic letters plus the sources and sinks of D formed by adding for every pair of edges <C,D> <D,F> such that D is noncyclic a new edge <C,F> and then deleting the vertex D from V and <C,D>, <D,F> from E. The reduced dag $D(A,B)$ will guide the derivation (if possible) of a word in $L(A) \cap L(B)$. Before arguing that no information has been "lost" in forming the reduced derivation dag, it is necessary to describe its role.

Notation: For a cyclic letter C with production $C \to D_1 \ldots D_k C D_{k+1} \ldots D_n$ and a reduced derivation dag D, $L(C)$ (resp. $R(C)$) is the set of source vertices of the subdag $D(D_1,\ldots,D_k)$ (resp. $D(D_{k+1},\ldots,D_n)$).

Next, the procedure for testing emptiness of $L(A) \cap L(B)$ in G_{AB} is described. The procedure involves a "parallel derivation" as exhibited in the example. At each step the two sentential forms will be

$$\alpha_0 C_1 \alpha_1 C_2 \ldots C_n \alpha_n \qquad (1)$$
$$\beta_0 C_1 \beta_1 C_2 \ldots C_n \beta_n \qquad (2)$$

where (1) is the sentential form in the derivation of A and (2) is the corresponding sentential form in the derivation of B. The C_i will be cycle letters known to match and are called *complimentary* letters. The terms *first* C_i and *second* C_i will refer to occurrences in their respective forms. The argument will proceed by showing how to form n+1 subproblems each involving α_i and β_i which can be solved independently of one another.

A key lemma for limiting the matching problem that will arise shortly is:

Lemma: Given the two forms

$$\alpha_0 C_1 \alpha_1 C_2 \alpha_2 \ldots C_n \alpha_n \in L(G_A) \qquad (3)$$
$$\beta_0 C_1 \beta_1 C_2 \beta_2 \ldots C_n \beta_n \in L(G_B) \qquad (4)$$

if there exists a $\tau \in L(G_A) \cap L(G_B)$ derivative of both (3) and (4), there exists a $\tau' \in L(G_A) \cap L(G_B)$ derivative of (3) and (4) that requires pumping of at most one letter of each complementary letter pair.

Basic step: In forming the initial sentential forms (1) and (2), there are two cases: (a) both A and B are noncyclic letters and (b) one of them is cyclic. By persistence of cyclic letters, both A and B cyclic implies $L(A) \cap L(B) = \phi$. In case (a), (1) (resp. (2)) is simply the sentential form formed from the direct descendants of A (resp. B), in (A,B).

Let

$$\alpha_0 C_1 \alpha_1 C_2 \alpha_2 \cdots C_n \alpha_n \qquad (5)$$
$$\beta_0 D_1 \beta_1 D_2 \beta_2 \cdots D_m \beta_m \qquad (6)$$

be the two words thus constructed where the C_i and D_i are all occurrences of source nodes in $D(A,B)$ when A and B are removed. If $w \in L(A) \cap L(B)$ then the C_i and D_j of (5) and (6) must be in the derivation for w since this is the first step of the derivation. Since there can be no more copies of the source vertices introduced, $w \in L(A) \cap L(B)$ iff m=n and $C_i = D_i$.

For the case (b), assume A cyclic and form descendant word for B:

$$\beta_0 D_1 \beta_1 D_2 \cdots \beta_k A \beta_{k+1} \cdots D_m \beta_m$$

where $D_1 \ldots D_k$ (resp. $D_{k+1} \ldots D_m$) sources in $L(A)$ (resp. $R(A)$).

If A can be pumped to form

$$\alpha_0 C_1 \alpha_1 C_2 \cdots \alpha_k A \alpha_{k+1} \cdots C_n \alpha_n$$

so that m=n and $C_i = D_i$ we continue. If not, source nodes cannot be otherwise introduced and $L(A) \cap L(B) = \phi$.

In either case, the C_i must be in any word derived by persistence of cyclic letters. The α_i, β_i contain cyclic as well as noncyclic words introduced by the transformation from cycles to trivial cycles as well as the operation of reducing the dag. However, a moments reflection indicates that any letters introduced by these two operations cannot be misleading.

Subproblem formation:
The problem is to match X,Y in the context of $D(C_1, \ldots, C_n)$ where

$$X = \alpha_0 C_1 \alpha_1 C_2 \cdots C_n \alpha_n$$
$$Y = \beta_1 C_1 \beta_1 C_2 \cdots C_n \beta_n .$$

The goal is to break this problem into simpler subproblems; however, because of the interdependencies illustrated in the example, this cannot be done directly.

The general procedure is to proceed from left to right through the two sentential forms trying to match corresponding sequences $\alpha_i C_{i+1}$ against $\beta_i C_{i+1}$ $(0 \le i < n)$. Match, here, does not mean $\alpha_i = \beta_i$; but that those letters that can only be introduced by C_{i+1}, namely $L(C_{i+1})$, match as a subsequence. (Denote this match by $\alpha_i = \beta_i \mid L(C_{i+1})$ and read "α_i matches β_i relative to $L(C_{i+1})$".)

There are two steps. Step 1 is used when a certain number of cycles of one of the C_{i+1} complementary pair is dictated by the necessity of matching $\alpha_i = \beta_i \mid L(C_{i+1})$. Under some circumstances (e.g., $L(C_{i+1}) = \phi$) no constraints are immediately imposed. If so, Step 2 postpones resolution and labels C_{i+1} a "filler" -- a form that can be pumped arbitrarily to achieve a match. (The variable d of the example would be a "filler" if all words of the example were rever ed.) When an explicit number of pumps is discovered, the pending fillers are converted to subproblems by a procedure called "cascading." (Both "filler" and "cascade" are explained more fully after step 2.) Finally, it should be emphasized that the matching required in the following steps is a finite process by virtue of the earlier lemma on pumping only one letter of a complementary pair.

Step 1: Given $\alpha_i C_{i+1} \alpha_{i+1} \cdots C_n \alpha_n$
$\beta_i C_{i+1} \beta_{i+1} \cdots C_n \beta_n$.

Case 1: $(L(C_i) \ne \phi, \alpha_i = \beta_i \mid L(C_{i+1}))$. Constrained -- since C_{i+1} can't be cycle without ruining the equality. $X = \alpha_i$, $Y = \beta_i$ in $D(L(C_{i+1}))$ is the new subproblem. Delete $\alpha_i C_{i+1}$ and $\beta_i C_{i+1}$ and return to step 1.

Case 2: $(L(C_i) \ne \phi, \alpha_i \ne \beta_i \mid L(C_{i+1}))$. Constrained -- force $\alpha_i = \beta_i \mid L(C_{i+1})$ if possible. If not possible, then the intersection is empty. If t cycles of (say) the first C_{i+1} force equality relative to $L(C_{i+1})$ and $C_{i+1} \to \gamma C_{i+1} \gamma'$, then $X = \alpha_i \gamma^t, Y = \beta_i$ in $D(C_{i+1})$ is the subproblem. Remove $\alpha_i C_{i+1}$ and $\beta_i C_{i+1}$ and replace α_{i+1} by $\gamma'^t \alpha_{i+1}$; return to step 1.

Case 3: $(L(C_{i+1}) = \phi)$. Constrained -- α_i and β_i must match exactly since cycling C_{i+1} can't help. If $\alpha_i \ne \beta_i$ the intersection is empty, otherwise verify match, delete α_i and β_i and go to step 2, k=i.

Termination: (i=n) Proceed as in case 3 except halt instead of going to step 2.

Step 2: Given $C_k \alpha_k \cdots C_i \alpha_i C_{i+1} \cdots C_n \alpha_n$
$C_k \beta_k \cdots C_i \beta_i C_{i+1} \cdots C_n \beta_n$
where $C_k \ldots C_{i-1}$ are fillers.

Case 1: $(L(C_{i+1}) = \phi)$. Constrained -- cycle C_i to force $\alpha_i = \beta_i \mid R(C_i)$. If not possible -- intersection is empty. If possible with t cycles of (say) the first C_i and $C_i \to \gamma C_i \gamma'$, the subproblem is $X = \gamma'^t \alpha_i, Y = \beta_i$, $D(R(C_i))$. Replace α_{i-1} with $\alpha_{i-1} \gamma^t$ and cascade. Delete everything to the left of C_{i+1} and return to step 2, k=i+1.

Case 2: $(L(C_{i+1}) \neq \phi \wedge L(C_{i+1}) \neq R(C_i)$. Constrained -- cycle C_i and/or C_{i+1} until they match with respect to both $L(C_{i+1})$ and $R(C_i)$, if possible. If not, the intersection is empty. If so, and (say) the first C_i and (say) the second C_{i+1} are cycled t and u times, respectively, and $C_i \rightarrow \gamma C_i \gamma'$ and $C_{i+1} \rightarrow \delta C_{i+1} \delta'$ the new problems are $X = \gamma'^t \alpha_i, Y = \beta_i \delta^u$ in $D(R(C_i) \cup L(C_{i+1}))$. Replace α_{i-1} by $\alpha_{i-1}\gamma^t$ and cascade. Replace β_{i+1} by $\delta'^u \beta_{i+1}$, delete C_{i+1} and everything to the left and go to step 1.

Case 3: $(L(C_{i+1}) \neq \phi \wedge L(C_{i+1}) = R(C_i))$. If C_i is a filler, increment i and return to step 2. If C_i is not a filler, proceed as in case 2.

Termination: (i=n) Treat as case 1.

A filler is a cycle letter C_i in a form

$$C_i \alpha_i C_{i+1}$$
$$C_i \beta_i C_{i+1}$$

that can force a match given that C_{i+1} has cycled $t \in N$ times. To test if C_i is a filler given $C_i \rightarrow \gamma \alpha_i \gamma'$ and $C_{i+1} \rightarrow \delta C_{i+1} \delta'$, cycle (opposite pairs) of C_i and C_{i+1} so that α_i and β_i do not overlap, then test the two resulting words. If $|\gamma'| = |\delta|$ or $u \cdot |\gamma'| = |\delta|$, then to be a filler the two words bounded by C_i and C_{i+1} must match. If $|\gamma| = |\delta| \cdot u$ then the words bounded by C_i and C_{i+1} must match once in every $v \leq u$ cycles of C_{i+1} in order to be a filler. In all other cases, C_i is not a filler. For the case where v>1, dialate C_{i+1} by v cycles -- then any number of C_{i+1} cycles can be matched.

To cascade

$$C_k \ldots C_{i-1} \alpha_{i-1} \gamma^t$$
$$C_k \ldots C_{i-1} \beta_{i-1}$$

where the $C_k \ldots C_{i-1}$ are fillers, force a match by cycling C_{i-1} the appropriate number of times. Thus, if $C_{i-1} \rightarrow \delta C_{i-1} \delta'$, find u such that $\alpha_{i-1}\gamma^t = \delta^u \beta_{i-1} \mid R(C_{i-1})$ and make $X = \alpha_{i-1}\gamma^t$ $Y = \delta^u \beta_{i-1}$ in $D(R(C_{i-1}))$ a subproblem. Then cascade the remainder of the forms.

Finally, note that all of the created subproblems can be solved independently. If they are all solved successfully, a word in $L(A) \cap L(B)$ is found. Otherwise, none can exist.

References:

[1] E. L. Post.
"Recursive unsolvability of a problem of Thue."
JSL, 1947.

[2] E. Cordoza, R. Lipton, and A. Meyer
"Exponential Space Complete Problems for Petri Nets and Communative Semigroups."
8th STOC, 1976.

[3] H. Strong, A. Maggiolo-Schnettini, and B. Rosen.
"Recursion Structure Simplification."
SIAM COMP, 1975.