Representation of Semiautomata by Canonical Words and Equivalences*

Janusz Brzozowski[1] and Helmut Jürgensen[2]

[1] School of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1
`brzozo@uwaterloo.ca` `http://maveric.uwaterloo.ca`

[2] Department of Computer Science, The University of Western Ontario, London, ON, Canada N6A 5B7
and
Institut für Informatik, Universität Potsdam, August-Bebel-Str. 89, 14482 Potsdam, Germany
`helmut@uwo.ca` `http://www.csd.uwo.ca/faculty_helmut.htm`

Abstract. We study a novel representation of semiautomata, which is motivated by the method of trace-assertion specifications of software modules. Each state of the semiautomaton is represented by an arbitrary word, the *canonical word* leading to that state. The transitions of the semiautomaton give rise to a right congruence, the *state-equivalence,* on the set of input words of the semiautomaton: two words are state-equivalent if and only if they lead to the same state. We present a simple algorithm for finding a set of generators for state-equivalence. Directly from this set of generators, we construct a confluent rewriting system which permits us to transform any word to its canonical representative. In general, the rewriting system may allow infinite derivations. To address this issue, we impose the condition of prefix-continuity on the set of canonical words. A set is prefix-continuous if whenever a word w and a prefix u of w are in the set, then all the prefixes of w longer than u are also in the set. Prefix-continuous sets include prefix codes and prefix-closed sets as special cases. We prove that the rewriting system is Noetherian if and only if the set of canonical words is prefix-continuous. Furthermore, if the set of canonical words is prefix-continuous, then the set of rewriting rules is irredundant. We show that each prefix-continuous canonical set corresponds to a spanning forest of the semiautomaton.

1 Introduction

The trace-assertion specification [1, 2, 7] of a software module is based on an automaton defined in a rather indirect way. A set of important traces (sequences of operations), called "canonical," is first identified. Each of the remaining traces is then classified as equivalent to some canonical trace, and a trace rewriting system is used to find the canonical representative of any given trace. The trace-assertion method was introduced in 1977 by Bartussek and Parnas [1] and further developed by several authors. For a short history of the method and for further references see [2].

In this paper we study the most important aspects of the trace assertion method, namely, the choice of canonical traces, the generation of the trace equivalence relation, and the construction of the trace rewriting system. It turns out that these issues

* This paper appeared in *Descriptional Complexity of Formal Systems, 6th Workshop,* L. Ilie and D. Wotschke, eds., Rep. 619, Dept. of Comp. Sc., Univ. of Western Ontario, (2004) 13–27.

can be studied entirely in the realm of semiautomata and without any reference to software modules.

Every state of a semiautomaton is represented by a canonical word, an arbitrarily chosen word leading to that state. A right-congruence relation on the set of words defines the transitions of the semiautomaton, and coincides with the state-equivalence relation, by which words leading to the same state are identified. We describe a simple algorithm for constructing a set of generators for this right congruence. To transform any word to its canonical form algorithmically, we derive a simple confluent rewriting system directly from the generators of the equivalence relation. To avoid infinite derivations, we add the condition that the set of canonical words be prefix-continuous. A set is prefix-continuous if whenever a word w and a prefix u of w are in the set, then all the prefixes of w longer than u are also in the set. Prefix-continuous sets include prefix-closed sets, where every word in the set has all of its prefixes in the set, and prefix codes, where no word in the set is a prefix of any other word in the set, as special cases. We prove that the rewriting system is Noetherian if and only if the set of canonical words is prefix-continuous. Moreover, if the set of canonical words is prefix-continuous, then the rewriting system is irredundant.

The remainder of the paper is structured as follows. Section 2 introduces our terminology and notation. Arbitrary sets of canonical words are studied in Section 3. Prefix-continuous sets of canonical words are discussed in Section 4. Our theory is illustrated in Section 5 with the simple example of a counter. Section 6 concludes the paper.

2 Terminology and Notation

We denote by P the set of nonnegative integers. If Σ is an alphabet (finite or infinite), then Σ^+ and Σ^* denote the free semigroup and the free monoid, respectively, generated by Σ. The empty word is ϵ. For $w \in \Sigma^*$, $|w|$ denotes the length of w. If $w = uv$, for some $u, v \in \Sigma^*$, then u is a *prefix* of w. A set $X \subseteq \Sigma^*$ is a *prefix code* if no word of X is the prefix of any other word of X. Note that, with this definition, the set $\{\epsilon\}$ is a prefix code, in contrast to most of the commonly used definitions. A set X is *prefix-closed* if, for any $w \in X$, every prefix of w is also in X. A set X is *prefix-continuous* if, whenever $x = uav$ is in X, $a \in \Sigma$, then $u \in X$ implies $ua \in X$. Both prefix codes and prefix-closed sets are prefix-continuous.

2.1 Semiautomata and Equivalences

By a *deterministic initialized semiautomaton*, or simply *semiautomaton*, we mean a tuple $S = (\Sigma, Q, \delta, q_\epsilon)$, where Σ is a nonempty input alphabet, Q is a nonempty set of states, $\delta : Q \times \Sigma \to Q$ is the transition function, and $q_\epsilon \in Q$ is the initial state. In general, we do not assume that Σ and Q are finite. As usual, we extend the transition function to words by defining $\delta(q, \epsilon) = q$, for all $q \in Q$, and $\delta(q, wa) = \delta(\delta(q, w), a))$. A semiautomaton is *connected* if every state is reachable from the initial state. We consider only connected semiautomata. Thus, for every $q \in Q$, there exists $w \in \Sigma^*$ such that $\delta(q_\epsilon, w) = q$. For any $w \in \Sigma^*$, we define $q_w = \delta(q_\epsilon, w)$.

For a semiautomaton $S = (\Sigma, Q, \delta, q_\epsilon)$, the *state-equivalence* relation $\equiv_\delta$ on Σ^* is defined by

$$w \equiv_\delta w' \Leftrightarrow q_w = q_{w'}, \tag{1}$$

for $w, w' \in \Sigma^*$. Note that $\equiv_\delta$ is an equivalence relation, and also a right congruence, that is, for all $x \in \Sigma^*$,

$$w \equiv_\delta w' \Rightarrow wx \equiv_\delta w'x. \tag{2}$$

Given any right congruence $\sim$ on Σ^*, we can construct a semiautomaton $S_\sim = (\Sigma, Q_\sim, \delta_\sim, q_\sim)$, as follows. For $w \in \Sigma^*$, let $[w]_\sim$ be the equivalence class of w. Let $Q_\sim$ be the set of equivalence classes of $\sim$, let $q_\sim = [\epsilon]_\sim$, and, for $a \in \Sigma$, let $\delta([w]_\sim, a) = [wa]_\sim$. Note that $S_\sim$ is connected. It is well-known that the semiautomaton $S_\sim$ is isomorphic to S when $\sim = \equiv_\delta$, with the isomorphism mapping $[w]_\sim$ onto q_w; see [6].

2.2 Rewriting Systems

In this paper we are concerned with very special rewriting systems. More information about general rewriting systems can be found in [3].

Let Σ be an alphabet (finite or infinite). A *rewriting system* over Σ consists of a set $\mathbf{T} \subseteq \Sigma^* \times \Sigma^*$ of *transformations* or *rules*. A *transformation* $(u, v) \in \mathbf{T}$ is written as $u \models v$. Then $\models^*$ is the reflexive and transitive closure of $\models$. Thus, $w \models^* w'$ if and only if $w = w_0 \models w_1 \models w_2 \models \cdots \models w_n = w'$ for some n, and n is the length of this derivation of w' from w. In the special cases considered in this paper, the transformations have the pattern $ux \models vx$, where $u, v \in \Sigma^*$ are specific words and x is an arbitrary word in Σ^*. Systems with this type of rules are known as *regular canonical systems* [4, 5], where "canonical" is a term unrelated to our subsequent usage of the term "canonical." Finite regular canonical systems generate precisely the regular languages and have been studied in detail by Büchi [4, 5]. These systems are equivalent to *expansive* systems in which $|u| \leq |v|$, whereas our systems do not have this property. In fact, in the rewriting systems that we propose, only a finite number of words can be derived from any given word. The second important difference between our work and that of [4, 5] is that we have an infinite number of patterns of rules, in general.

A rewriting system is *confluent* if, for any $w, w_1, w_2 \in \Sigma^*$ with $w \models^* w_1$ and $w \models^* w_2$, there is $w' \in \Sigma^*$ such that $w_1 \models^* w'$ and $w_2 \models^* w'$. It is *Noetherian* if there is no word w from which a derivation of infinite length exists. A confluent Noetherian system has the following important property: For every word $w \in \Sigma^*$ there is a unique word $\tau(w)$, such that, for any $u \in \Sigma^*$ with $w \models^* u$, one has $u \models^* \tau(w)$ and there is no word $v \in \Sigma^*$ with $\tau(w) \models v$. For an effectively defined confluent Noetherian system, one can compute $\tau(w)$ for every word w.

3 Arbitrary Sets of Canonical Words

Let $S = (\Sigma, Q, \delta, q_\epsilon)$ be a semiautomaton, and $\chi : Q \to \Sigma^*$, an arbitrary mapping assigning to state q a word $\chi(q)$ such that $\delta(q_\epsilon, \chi(q)) = q$. By definition χ is injective. *Unless stated otherwise, we assume that χ has been selected.* For $w \in \Sigma^*$, we call the word $\chi(q_w)$ the *canonical word* of state q_w, and the *canonical representative* of word w. Let the set of canonical words be $\mathbf{X}$.

Our first objective is to find a suitable generating set for the state-equivalence relation of a given semiautomaton. Our second objective is to transform any word to its canonical representative. In this section we make no assumptions about the nature of the set of canonical words.

Definition 1. *Relation $\equiv$ on Σ^* is the smallest right congruence containing the set $\hat{\mathbf{G}} = \mathbf{G} \cup \{(\epsilon, \chi(q_\epsilon))\}$, where $\mathbf{G}$ is the set of all ordered pairs $(wa, \chi(q_{wa}))$, with $w \in \mathbf{X}$, $a \in \Sigma$, and $wa \notin \mathbf{X}$.*

We refer to the pairs in $\mathbf{G}$ as *basic equivalences*. Note that the pairs are ordered for reasons that will become clear soon. The number of basic equivalences is infinite in general; it is finite when Q and Σ are finite. In the sequel, we write the pairs in $\mathbf{G}$ as equivalences, that is, $wa \equiv \chi(q_{wa})$; moreover, we label the pairs by $\mathbf{E1}, \mathbf{E2}, \ldots$

For finite semiautomata, we calculate the number of equations in $\mathbf{G}$ as follows.

Proposition 1. *Let S be a finite semiautomaton with n states and k input letters, and let $\mathbf{X}$ be a set of canonical words for S. Let n_0 be the number of words $w \in \mathbf{X}$ such that $w = ua$ with $a \in \Sigma$ and $u \in \mathbf{X}$. Then the number of equations in $\mathbf{G}$ is $nk - n_0$.*

Proof. Each equation in $\mathbf{G}$ corresponds to a distinct transition of S. There is a total of nk transitions, since there are k transitions out of each state. If u is a canonical word, transitions of the form $\delta(q_u, a) = q_{ua}$, where ua is canonical do not contribute to $\mathbf{G}$. The number of such transitions is n_0. Every transition in which ua is not canonical contributes one equation to $\mathbf{G}$. □

Note that $0 \leq n_0 \leq n - 1$. If $\mathbf{X}$ is a prefix code, then $n_0 = 0$. The converse does not hold as shown in Example 1 below. At the other extreme, one verifies that $\mathbf{X}$ is prefix-closed if and only if $n_0 = n - 1$.

Lemma 1. $\equiv\, \subseteq\, \equiv_\delta$.

Proof. By the construction of $\hat{\mathbf{G}}$, the words in each pair of $\hat{\mathbf{G}}$ lead to the same state, that is, $\hat{\mathbf{G}} \subseteq\, \equiv_\delta$. By the right-congruence property of $\equiv_\delta$, the claim follows. □

We show later that the converse containment also holds.

We define a set $\mathbf{T}$ of *basic transformations* as follows. If $w \equiv w'$ is a pair $\mathbf{Ei}$ in $\mathbf{G}$, then $wx \models w'x$ is the corresponding basic transformation $\mathbf{Ti}$ in $\mathbf{T}$. In these transformations, w and w' are fixed words and x is any word.

Lemma 2. *For all $w, w' \in \Sigma^*$, $w \models^* w'$ implies $w \equiv w'$ and therefore $w \equiv_\delta w'$.*

Proof. By definition, each transformation preserves $\equiv$, and $\equiv$ is transitive. By Lemma 1, each transformation also preserves the state. □

Lemma 3. *For $w \in \Sigma^*$, the following hold:*

1. *If no prefix of w is canonical, then $w \models^* w'$ implies $w' = w$.*
2. *If w has a canonical prefix and $w \models^* w'$, then w' has a canonical prefix.*
3. *$w \models^* \chi(q_w)$ if and only if w has a canonical prefix.*

Proof. Suppose no prefix of w is canonical. Then no rule applies to w, because all the rules are of the form $ua \equiv \chi(q_{ua})$, where u is canonical. Consequently, w can only derive itself, and it can do so, because $\models^*$ is reflexive.

For the second claim, suppose w has a canonical prefix. If $w = w'$, the claim holds. If $w \models w'$, then w has the form $w = uav$, where $u, v \in \Sigma^*, a \in \Sigma$, u is canonical and ua is not canonical. Then $w' = \chi(q_{ua})v$, where $\chi(q_{ua})$ is canonical. Now the claim follows by transitivity.

For the third claim, suppose that w has a canonical prefix. We show by induction on the length of w that $w \models^* \chi(q_w)$. If $w = \epsilon$, then w can only have one canonical prefix, namely itself. Thus $\epsilon \models^* \epsilon = \chi(q_\epsilon)$, since $\models^*$ is reflexive; hence the claim holds for the basis case. Now suppose that every word of length less than or equal to n that has a canonical prefix satisfies the claim. Consider $w = ua$ with $|u| = n$ and $a \in \Sigma$, where w has a canonical prefix. If w itself is canonical, then $w \models^* w = \chi(q_w)$. Otherwise, we know that u has a canonical prefix. By the induction assumption, $u \models^* \chi(q_u)$, and so $w = ua \models^* \chi(q_u)a$. If $\chi(q_u)a$ is canonical, then $\chi(q_u)a = \chi(q_{ua}) = \chi(q_w)$, and $w \models^* \chi(q_w)$. Otherwise, $\chi(q_u)a \models \chi(q_{ua})$ is a rule in $\mathbf{T}$, and $w = ua \models^* \chi(q_u)a \models \chi(q_{ua})$.

Conversely, if w does not have a canonical prefix, then it can only derive itself. Since w is not canonical, $w \neq \chi(q_w)$. Therefore w cannot derive $\chi(q_w)$. $\square$

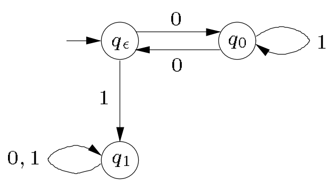

Fig. 1. Semiautomaton S_1

Example 1. Consider the semiautomaton of Fig. 1. The initial state is indicated by an incoming arrow, and each transition between two states is labelled by the input causing the transition.

Suppose $\chi(q_\epsilon) = \epsilon$, $\chi(q_0) = 01$, and $\chi(q_1) = 1$. Then we have the following basic equivalences and corresponding basic transformations for all $x \in \Sigma^*$:

E1 $0 \equiv 01$, **E2** $10 \equiv 1$, **E3** $11 \equiv 1$, **E4** $010 \equiv \epsilon$, **E5** $011 \equiv 01$.
T1 $0x \models 01x$, **T2** $10x \models 1x$, **T3** $11x \models 1x$, **T4** $010x \models x$, **T5** $011x \models 01x$.

On the other hand, let $\chi(q_\epsilon) = 00$, $\chi(q_0) = 0$, and $\chi(q_1) = 1$. Then we have:

E1 $01 \equiv 0$, **E2** $10 \equiv 1$, **E3** $11 \equiv 1$, **E4** $000 \equiv 0$, **E5** $001 \equiv 1$.
T1 $01x \models 0x$, **T2** $10x \models 1x$, **T3** $11x \models 1x$, **T4** $000x \models 0x$, **T5** $001x \models 1x$.

For this choice of χ, ϵ cannot derive $\chi(q_\epsilon) = 00$; this illustrates Lemma 3 (3).

If we use $\chi(q_\epsilon) = \epsilon$, $\chi(q_0) = 01$, and $\chi(q_1) = 10$, then n_0 of Proposition 1 is 0, but $\mathbf{X}$ is not a prefix code. □

Definition 2. *Given a set* $\mathbf{X}$ *of canonical words, we define the following subsets:*

- $\mathbf{W} = \Sigma^* \setminus \mathbf{X}\Sigma^*$ *is the set of* acanonical *words.*
- $\mathbf{X}_0 = \mathbf{X} \setminus \mathbf{X}\Sigma^+$ *is the set of* minimal canonical *words.*
- $\mathbf{Y} = \mathbf{X}_0\Sigma^+$ *is the set of* post-canonical *words.*

Set $\mathbf{W}$ consists of all the words that do not have a canonical prefix; clearly, $\mathbf{W}$ is prefix-closed. Set $\mathbf{X}_0$ is the set of canonical words w such that w has no canonical prefix other than w. This set is a prefix code. Set $\mathbf{Y}$ is the set of all words w such that w has at least one canonical prefix and is not in $\mathbf{X}_0$. Note that both $\mathbf{Y}$ and $\mathbf{X}_0 \cup \mathbf{Y}$ are prefix-continuous. The triple $(\mathbf{W}, \mathbf{X}_0, \mathbf{Y})$ is a partition of Σ^*. In general, all three sets may be infinite.

Theorem 1. $\equiv\, = \,\equiv_\delta$.

Proof. By Lemma 1, $\equiv\, \subseteq \,\equiv_\delta$. To prove the converse, we show that $q_w = q_{w'}$ implies $w \equiv w'$, for all $w, w' \in \Sigma^*$. We do this by showing that each word w is equivalent to its canonical representative. From $q_w = q_{w'}$ it then follows that $w \equiv \chi(q_w) = \chi(q_{w'}) \equiv w'$.

We first claim that each acanonical word is equivalent to its canonical representative. Suppose ϵ is acanonical. Since the pair $(\epsilon, \chi(q_\epsilon))$ is in $\hat{\mathbf{G}}$, $\epsilon \equiv \chi(q_\epsilon)$. So the claim holds for the acanonical word of length 0. Now suppose that the claim holds for all acanonical words of length less than or equal to h, $h \geq 0$. Consider acanonical wa, where $|w| = h$, and $a \in \Sigma$. By the induction hypothesis, $w \equiv \chi(q_w)$. Since $\equiv$ is a right congruence, we have $wa \equiv \chi(q_w)a$. If $\chi(q_w)a$ is canonical, then $\chi(q_w)a = \chi(q_{wa})$, and $wa \equiv \chi(q_{wa})$. Otherwise, by construction of $\mathbf{G}$, the pair $(\chi(q_w)a, \chi(q_{wa}))$ is in $\mathbf{G}$, and our claim follows by transitivity of $\equiv$.

Next, consider a word w in $\mathbf{X}_0 \cup \mathbf{Y}$. By Lemma 3 (3), $w \models^* \chi(q_w)$. By Lemma 2, $w \equiv \chi(q_w)$. This completes the proof. □

It is a disadvantage of the rewriting system $\mathbf{T}$ that an acanonical word cannot derive its canonical representative. To remedy this, we augment $\mathbf{T}$ as follows:

Definition 3. $\hat{\mathbf{T}} = \mathbf{T} \cup \{w \models \chi(q_\epsilon)w \mid w \in \mathbf{W}\}$.

We call the added rules *acanonical*. Note that the acanonical rule $w \models \chi(q_\epsilon)w$ can be applied only to w and to no other word. After this rule is applied, the result is a post-canonical word. By Lemma 3 (2), no acanonical rule is applicable after the first step.

Theorem 2. *The rewriting systems* $\mathbf{T}$ *and* $\hat{\mathbf{T}}$ *are confluent, and every* $w \in \Sigma^*$ *derives its canonical representative* $\chi(q_w)$ *in* $\hat{\mathbf{T}}$.

Proof. First consider derivations in $\mathbf{T}$ only. Suppose $w \in \Sigma^*$. If w has no canonical prefix, then w can only derive itself, by Lemma 3 (1). Hence w cannot possibly contradict the confluence property. On the other hand, if w does possess a canonical prefix, and $w \models^* w_1$ and $w \models^* w_2$, then w_1 and w_2 also have canonical prefixes, by Lemma 3 (2). By Lemma 3 (3), $w_1 \models^* \chi(q_{w_1})$, and $w_2 \models^* \chi(q_{w_2})$. By Lemma 2, $q_w = q_{w_1} = q_{w_2}$. Hence $w_1 \models^* \chi(q_{w_1}) = \chi(q_w)$, $w_2 \models^* \chi(q_{w_2}) = \chi(q_w)$, and $\mathbf{T}$ is confluent.

For $\hat{\mathbf{T}}$, if a derivation starts with an acanonical word w, only the rule $w \models \chi(q_\epsilon)w$ is applicable. The resulting word $\chi(q_\epsilon)w$ is post-canonical, and only the rules of $\mathbf{T}$ apply to it. As $\mathbf{T}$ is confluent, this derivation, like any derivation starting with a post-canonical word, cannot violate confluence.

By Lemma 3 (3), the last claim is true for all post-canonical and canonical words. Now consider an acanonical word w. If $w = \epsilon$, then $\epsilon \models \chi(q_\epsilon)\epsilon = \chi(q_\epsilon)$ in $\hat{\mathbf{T}}$. Suppose that $w \neq \epsilon$. By rule $w \models \chi(q_\epsilon)w$, we convert acanonical word w to post-canonical word $\chi(q_\epsilon)w$, which then derives, in $\mathbf{T}$, the canonical representative $\chi(q_{\chi(q_\epsilon)w})$ of $\chi(q_\epsilon)w$. Thus $w \models^* \chi(q_{\chi(q_\epsilon)w})$ in $\hat{\mathbf{T}}$. Since $\epsilon \equiv \chi(q_\epsilon)$, we have $w \equiv \chi(q_\epsilon)w$ and $q_w = q_{\chi(q_\epsilon)w}$. Hence $\chi(q_w) = \chi(q_{\chi(q_\epsilon)w})$, and so $w \models^* \chi(q_w)$ in $\hat{\mathbf{T}}$. □

Theorem 3. *For any $w, w' \in \Sigma^*$, we have $\chi(q_w) = \chi(q_{w'})$ if and only if $w \equiv w'$.*

Proof. Suppose $\chi(q_w) = \chi(q_{w'})$. Since χ is injective, $q_w = q_{w'}$. By Theorem 1, $w \equiv w'$. Conversely, if $w \equiv w'$, then $q_w = q_{w'}$. Hence $\chi(q_w) = \chi(q_{w'})$. □

One can reconstruct a semiautomaton from its canonical words and equivalences. In fact, let $S = (\Sigma, Q, \delta, q_\epsilon)$ be a semiautomaton, let $\mathbf{X}$ be a set of canonical words, and let $\hat{\mathbf{G}}$ be the set of equivalences derived from S. Let $S_\mathbf{X} = (\Sigma, \mathbf{X}, \delta_\mathbf{X}, \chi(q_\epsilon))$, where, for all $w \in \mathbf{X}, a \in \Sigma$, $\delta_\mathbf{X}(w, a) = wa$ if $wa \in \mathbf{X}$, and $\delta_\mathbf{X}(w, a) = \chi(q_{wa})$, if $(wa, \chi(q_{wa})) \in \mathbf{G}$.

Proposition 2. *The semiautomata $S = (\Sigma, Q, \delta, q_\epsilon)$ and $S_\mathbf{X} = (\Sigma, \mathbf{X}, \delta_\mathbf{X}, \chi(q_\epsilon))$ are isomorphic, with the isomorphism mapping state $q \in Q$ to $\chi(q) \in \mathbf{X}$.*

Proof. By Theorem 1, the right congruence generated by $\hat{\mathbf{G}}$ is precisely $\equiv_\delta$. □

All the results of this section hold for arbitrary sets of canonical words. Equivalence of two words w and w' is provable in the following sense. By Theorem 2, there exist (finite) derivations $w \models^* \chi(q_w)$ and $w' \models^* \chi(q_{w'})$. By Theorem 3, $w \equiv w'$ if and only if $\chi(q_w) = \chi(q_{w'})$. However, we still have the problem that the rewriting system may permit infinite derivations. This problem is addressed in the next section.

4 Prefix-Continuous Sets of Canonical Words

We now show that, if $\mathbf{X}$ is prefix-continuous, the process of reducing a word to its canonical representative by a derivation in $\hat{\mathbf{T}}$ is deterministic. Equivalence of two words is then proved by reducing them to their canonical representatives, and comparing the representatives. Without prefix-continuity, however, $\hat{\mathbf{T}}$ may allow infinite derivations, as in the next example.

Example 2. Return to the semiautomaton of Fig. 1, with $\chi(q_\epsilon) = \epsilon$, $\chi(q_0) = 01$, and $\chi(q_1) = 1$, and the corresponding rules:

$$\mathbf{T1}\ \ 0x \models 01x,\quad \mathbf{T2}\ \ 10x \models 1x,\quad \mathbf{T3}\ \ 11x \models 1x,\quad \mathbf{T4}\ \ 010x \models x,\quad \mathbf{T5}\ \ 011x \models 01x.$$

We consider derivations starting with 0. The first one leads to $\chi(q_0)$:

$$0 \overset{\mathbf{T1}}{\models} 01.$$

Note, however, that rule **T1** can be applied repeatedly, leading to the derivation

$$0 \overset{\mathbf{T1}}{\models} 01 \overset{\mathbf{T1}}{\models} 011 \overset{\mathbf{T1}}{\models} 0111 \overset{\mathbf{T1}}{\models} \ldots,$$

which never terminates. There is yet another derivation

$$0 \overset{\mathbf{T1}}{\models} 01 \overset{\mathbf{T1}}{\models} 011 \overset{\mathbf{T5}}{\models} 01 \overset{\mathbf{T1}}{\models} 011 \overset{\mathbf{T5}}{\models} 01 \ldots$$

which is also infinite. □

We avoid infinite derivations by adding the condition of prefix-continuity.

Lemma 4. *If* $\mathbf{X}$ *is prefix-continuous, the set* $\mathbf{L}$ *of all left-hand sides of the generating equivalences in* $\mathbf{G}$ *is a prefix code. If* $\mathbf{X}$ *(and therefore also the semiautomaton) is finite, the converse also holds.*

Proof. Suppose there exist words $w, w' \in \mathbf{X}$ and letters $a, a' \in \Sigma$, such that wa and $w'a'$ are in $\mathbf{L}$, $wa \neq w'a'$, and wa is a prefix of $w'a'$. Then wa is a prefix of w'. But then wa must be canonical, since w and w' are canonical, w is a prefix of w', and $\mathbf{X}$ is prefix-continuous. This contradicts the fact that wa is the left-hand side of an equivalence. Hence $\mathbf{L}$ is a prefix code.

Conversely, suppose that $\mathbf{X}$ is finite but not prefix-continuous, and $\mathbf{L}$ is a prefix code. Then there exists $w = uav \in \mathbf{X}$ such that $u \in \mathbf{X}$, and $ua \notin \mathbf{X}$. Consider the infinite set of words $w\Sigma^*$. Since $\mathbf{X}$ is finite, all these words cannot be canonical. Hence there exists some extension wxb of w such that wx is canonical and wxb is not. Therefore $\mathbf{G}$ contains the equivalences $ua \equiv \chi(q_{ua})$ and $uaxb \equiv \chi(q_{uaxb})$, showing that $ua, uaxb \in \mathbf{L}$. Therefore $\mathbf{L}$ cannot be a prefix code. □

The next example shows that the converse of Lemma 4 does not hold in general.

Example 3. In the semiautomaton of Fig. 2, the states are labelled with their canonical representatives. From state 00 on (to the right) the semiautomaton consists of an infinite binary tree. The set of canonical words is $\{\epsilon, 1\} \cup 00\Sigma^*$, which is not prefix continuous. The set of basic equivalences is $\{0 \equiv 1, 10 \equiv 00, 11 \equiv 00\}$. The set $\mathbf{L} = \{0, 10, 11\}$ of left-hand sides is a prefix code. □

Lemma 5. *At most one rule of* $\hat{\mathbf{T}}$ *applies to any word if and only if* $\mathbf{L}$ *is a prefix code.*

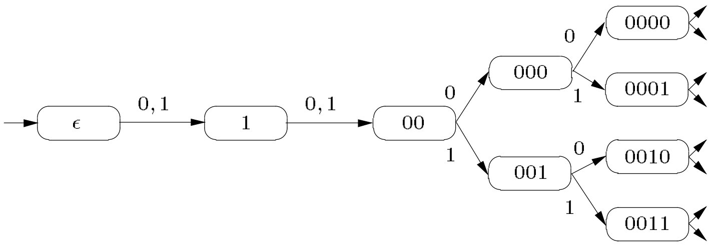

Fig. 2. Semiautomaton illustrating that the converse of Lemma 4 is false

Proof. If w is acanonical, the acanonical rule $w \models \chi(q_\epsilon)w$ is the only rule that applies. If w is minimal canonical, then no rule of $\hat{\mathbf{T}}$ applies. If w is post-canonical, then only the rules of $\mathbf{T}$ can be applicable. If $\mathbf{L}$ is a prefix code, at most one rule applies.

Conversely, if $\mathbf{L}$ is not a prefix code, then there exists a post-canonical word to which at least two rules apply. $\square$

Lemma 6. *If $\mathbf{X}$ is prefix-continuous and $w \in \mathbf{X}$, no rule of $\hat{\mathbf{T}}$ applies to w.*

Proof. As $\mathbf{X}$ is prefix-continuous, w cannot have a canonical prefix u and a non-canonical prefix ua. Hence, by the definition of $\mathbf{T}$, no prefix of w is in $\mathbf{L}$, and no rule applies. Also, no acanonical rule can apply to $w \in \mathbf{X}$. $\square$

Theorem 4. *The rewriting system $\hat{\mathbf{T}}$ is Noetherian if and only if the set $\mathbf{X}$ of canonical words is prefix-continuous.*

Proof. Suppose $\mathbf{X}$ is prefix-continuous. By Lemma 4, $\mathbf{L}$ is a prefix code. By Lemma 5, at most one rule applies to any word. Hence the rewriting process is deterministic. By Theorem 2, each word derives its canonical representative, from which no further derivation is possible, by Lemma 6. Therefore $\hat{\mathbf{T}}$ is Noetherian.

Conversely, suppose that $\mathbf{X}$ is not prefix-continuous. Then there exists $w = uav \in \mathbf{X}$ such that $u \in \mathbf{X}$, but $ua \notin \mathbf{X}$. Therefore $(ua, \chi(q_{ua})) \in \mathbf{G}$, and $w = uav \models \chi(q_{ua})v$. By Lemma 2, w and $\chi(q_{ua})v$ lead to the same state. By Lemma 3 (3), $\chi(q_{ua})v \models^* \chi(q_w) = w$. Thus $w \models \chi(q_{ua})v \models^* w$, and the rewriting system is not Noetherian. $\square$

Theorem 5. *If $\mathbf{X}$ is prefix-continuous, then $\hat{\mathbf{G}}$ is irredundant in the following sense:*

- *If $\epsilon \notin \mathbf{X}$, then $\mathbf{G}$ does not generate $\equiv_\delta$.*
- *For any pair $p = (ua, \chi(q_{ua})) \in \mathbf{G}$, the set $\hat{\mathbf{G}} \setminus p$ does not generate $\equiv_\delta$.*

Proof. Removing a pair from $\hat{\mathbf{G}}$ amounts to removing the corresponding rule from $\hat{\mathbf{T}}$.

If $\epsilon \notin \mathbf{X}$ and $(\epsilon, \chi(q_\epsilon))$ is removed, then the equivalence class of $\equiv$ containing ϵ must be a singleton, since ϵ cannot appear on either side of any rule in $\mathbf{T}$, and the equivalence $\epsilon \equiv \chi(q_\epsilon)$ cannot be derived from any other equivalence by applying the right-congruence property.

Now suppose that $(ua, \chi(q_{ua}))$ is removed from $\mathbf{G}$. By Lemma 6, no rule applies to $\chi(q_{ua})$. On the other hand, ua cannot appear as either side of any other pair in $\mathbf{G}$. By Lemma 5, at most one rule of $\hat{\mathbf{T}}$ applies to any word. Since the only rule applicable to ua has been removed, nothing else is applicable. Hence ua and $\chi(q_{ua})$ must be in different equivalence classes. □

Example 4. This example shows that Theorem 5 does not hold in general. Consider the semiautomaton of Fig. 3, where the canonical traces are shown as state labels. Here, $\mathbf{X} = \{\epsilon, 1, 00, 100\}$ is not prefix-continuous. The set of basic equivalences is

$$\{0 \equiv 1, 10 \equiv 00, 11 \equiv 00, 000 \equiv 100, 001 \equiv 100, 1000 \equiv 100, 1001 \equiv 100\}.$$

The equivalence $0 \equiv 1$ implies $000 \equiv 100$ by right congruence. Hence $000 \equiv 100$ is redundant. □

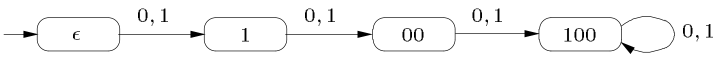

Fig. 3. Semiautomaton with redundant equivalence

Prefix-continuous canonical sets can be found with the aid of certain graph-theoretic constructs. Recall that a directed graph is a pair $G = (V, E)$, where V is the set of vertices of G and $E \subseteq V \times V$ is the set of (directed) edges of G. A *spanning forest* of a directed graph $G = (V, E)$ is a set of pairwise disjoint trees, such that V is the union of all the vertices in the trees. A spanning tree is a spanning forest consisting of a single tree.

To find a prefix-continuous canonical set for a semiautomaton S, we can use a spanning forest. Given such a forest of disjoint trees, for the root r of a tree, choose an arbitrary word w_r leading to state r from the initial state of S. Proceeding by induction, if state q has been assigned word w_q and state q' is a child of q reached from q by applying input a, then state q' is assigned word $w_q a$. In this way we associate a word with each state of S. The set of these words is then the canonical set for S, and it is prefix-continuous.

Example 5. Consider the semiautomaton of Fig. 1, and the forest of three one-vertex trees $\{q_\epsilon\}$, $\{q_0\}$ and $\{q_1\}$. We can choose 00, 01, and 1 for the roots $\{q_\epsilon\}$, $\{q_0\}$ and $\{q_1\}$, respectively, resulting in the set $\{00, 01, 1\}$ of canonical words. This set is a prefix code. The acanonical words are ϵ and 0, and the set of post-canonical words is $\{00, 01, 1\}\Sigma^+$.

On the other hand, we can choose the trees with vertices $\{q_1\}$ and $\{q_\epsilon, q_0\}$. If we pick q_1 and q_0 as roots, and assign 1 to q_1, and 0 to q_0, then q_ϵ is assigned 00, and $\mathbf{X} = \{1, 0, 00\}$.

We can also choose a single tree with vertices $\{q_\epsilon, q_0, q_1\}$ and root q_0. If we assign 0 to the root, then q_ϵ and q_1 are assigned 00 and 001, respectively. □

Conversely, given a prefix-continuous canonical set $\mathbf{X}$, we can construct a spanning forest for S. The states reached from the initial state by the minimal canonical words are the roots of the forest. Continuing by induction, if word $u \in \mathbf{X}$ corresponds to state q, and if $a \in \Sigma$ and $ua \in \mathbf{X}$, then q_{ua} is a child of q under input a.

Thus to each word in $\mathbf{X}$ we associate a vertex in the forest; this is possible because $\mathbf{X}$ is prefix-continuous.

The family of prefix-continuous canonical sets contains two extreme special cases: prefix-closed sets and prefix codes. We now give examples of these types of sets.

To find a prefix-closed set of canonical words we can use a spanning tree of the state graph of the semiautomaton S, with q_ϵ as root, and $\chi(q_\epsilon) = \epsilon$.

Example 6. Consider the semiautomaton S_2 of Fig. 4. We show three spanning trees for S_2. The basic equivalences corresponding to the three spanning trees are, by rows,

E1 $01 \equiv 1$, **E2** $10 \equiv 00$, **E3** $11 \equiv 1$, **E4** $000 \equiv 1$, **E5** $001 \equiv 0$.
E1 $1 \equiv 01$, **E2** $00 \equiv 010$, **E3** $011 \equiv 01$, **E4** $0100 \equiv 01$, **E5** $0101 \equiv 0$.
E1 $00 \equiv 10$, **E2** $01 \equiv 1$, **E3** $11 \equiv 1$, **E4** $100 \equiv 1$, **E5** $101 \equiv 0$.

Note that all three spanning trees define the same number of basic equivalences, as guaranteed by Proposition 1. □

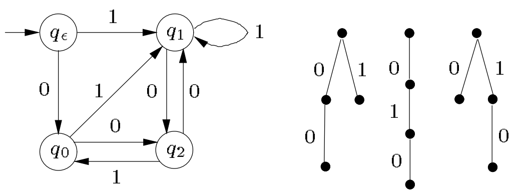

Fig. 4. Semiautomaton S_2 and spanning trees

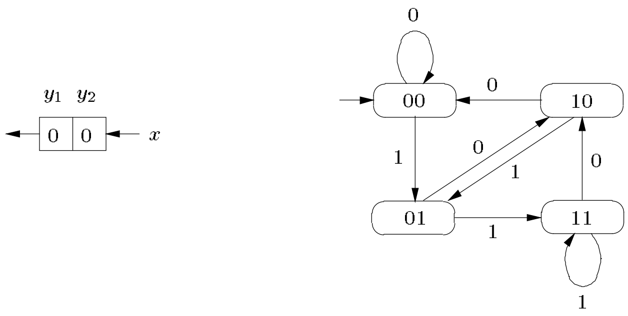

Fig. 5. 2-bit shift register

Example 7. This example illustrates the usefulness of prefix codes as canonical sets. Consider the 2-bit shift register of Fig. 5, started in state $(y_1, y_2) = (0, 0)$, with binary input x. The register contents are shifted to the left, with the value of x shifted to y_2 and the value of y_2 shifted to y_1. Assume that the shifts occur at integral values of time: $1, 2, \ldots t, \ldots$ At time $t+1$, we have $y_2(t+1) = x(t)$ and $y_1(t+1) = y_2(t)$. The semiautomaton of the shift register is shown in the figure, with parentheses and commas omitted from the state tuples for simplicity.

One representation for the states of the register is shown in Fig. 5, where each state corresponds to the register contents. The set of basic equivalences is:

$$\{000 \equiv 00, 001 \equiv 01, 010 \equiv 10, 011 \equiv 11, 110 \equiv 10, 111 \equiv 11, 100 \equiv 00, 101 \equiv 01\}.$$

The set $\{00, 01, 10, 11\}$ of canonical words has the advantage of using the natural state representation, and has much symmetry. For example, all eight equivalences can be summarized in one statement:

$$abc \equiv bc, \quad \forall a, b, c \in \Sigma.$$

This symmetry is lost if a prefix-closed set is used. □

5 Counter

We illustrate our method using the example of a counter. We also consider the effects of changing the initial state, and of choosing different spanning forests.

5.1 Counter with Empty Initial State

The initial count is 0. Only two operations are possible: INCREMENT, denoted by 1, and DECREMENT, denoted by 0. If the count is 0, DECREMENT is illegal and leads to a special illegal state. In any legal state it is possible to INCREMENT the count by 1. If the count is $(n+1)$, where $n \geq 0$, DECREMENT subtracts 1 from the count.

Definition 4. *The* counter semiautomaton *is* $S = (\Sigma, Q, \delta, q_\epsilon)$, *where* $\Sigma = \{0, 1\}$, $Q = P \cup \{\infty\}$, $q_\epsilon = 0$, *and* δ *is defined as*[1]

$$\begin{array}{lll}
\mathbf{C1'} & \delta(n, 1) = n + 1, & \forall n \in P, \\
\mathbf{C2'} & \delta(0, 0) = \infty, & \\
\mathbf{N1'} & \delta(\infty, a) = \infty, & \forall a \in \Sigma, \\
\mathbf{N2'} & \delta(n + 1, 0) = n, & \forall n \in P.
\end{array}$$

The counter semiautomaton is shown in Fig. 6 (a). The first step in constructing a trace specification of a semiautomaton is to select canonical words. We can use the spanning tree with initial state as root; this results in canonical word 1^n for the state with n entries, and in 0 for state ∞. Of course, the set $\{1\}^* \cup \{0\}$ is prefix closed. This step is illustrated in Fig. 6 (b).

The second step consists of finding the set **G** of basic equivalences. These equivalences provide the missing transitions in Fig. 6 (b), resulting in Fig. 6 (c).

The basic equivalences and the corresponding basic transformations are

$$\begin{array}{lllll}
\mathbf{E1'}\ 00 \equiv 0, & \mathbf{E2'}\ 01 \equiv 0, & \mathbf{E3'}\ 10 \equiv \epsilon, & \mathbf{E4'}\ 110 \equiv 1, & \ldots \\
\mathbf{T1'}\ 00x \models 0x, & \mathbf{T2'}\ 01x \models 0x, & \mathbf{T3'}\ 10x \models x, & \mathbf{T4'}\ 110x \models 1x, & \ldots
\end{array}$$

The set of equivalences is, of course, infinite. However, we can represent this infinite set by two patterns:

[1] The reason for the particular labelling of items will become apparent later.

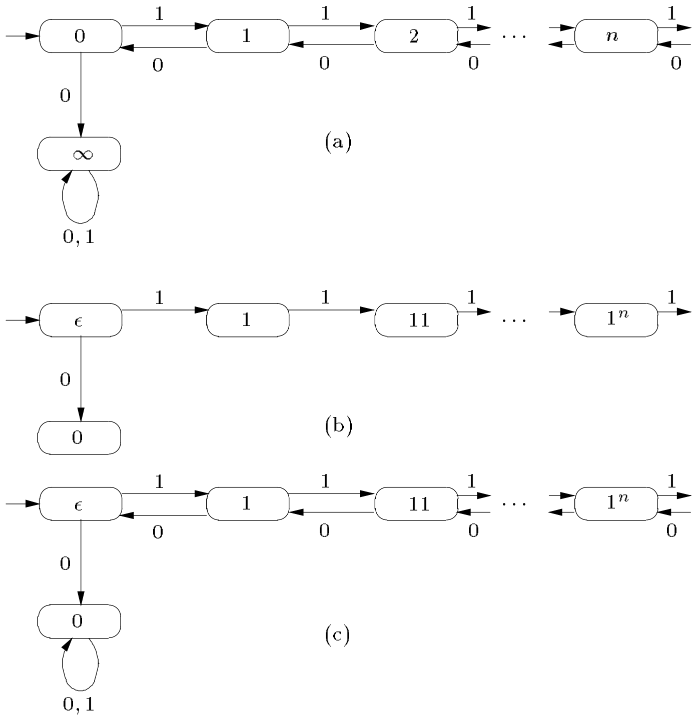

Fig. 6. Counter semiautomaton and canonical words

$$\begin{array}{lll} \textbf{E1} & 0a \equiv 0, & \forall a \in \Sigma, \\ \textbf{E2} & 1^{n+1}0 \equiv 1^n, & \forall n \in P. \end{array}$$

In fact, if we relabel the states with their canonical representatives, the definition of δ becomes

$$\begin{array}{lll} \textbf{C1} & \delta(1^n, 1) = 1^{n+1}, & \forall n \in P, \\ \textbf{C2} & \delta(\epsilon, 0) = 0, & \\ \textbf{N1} & \delta(0, a) = 0, & \forall a \in \Sigma, \\ \textbf{N2} & \delta(1^{n+1}, 0) = 1^n, & \forall n \in P. \end{array}$$

Now there is a 1-1 correspondence between the **Ni** and the **Ei**. Equations **Ni** correspond to noncanonical extensions of canonical words by letters. Equations **Ci** correspond to canonical extensions of canonical words by letters; hence they do not contribute to the equivalences.

In summary, we obtain the following specification for the counter:

Canonical words:

$\{1\}^* \cup \{0\}$

Equivalence:

$$\begin{array}{lll} \textbf{E1} & 0a \equiv 0, & \forall a \in \Sigma, \\ \textbf{E2} & 1^{n+1}0 \equiv 1^n, & \forall n \in P. \end{array}$$

Transformations:

$$\begin{array}{lll} \textbf{T1} & 0ax \models 0x, & \forall a \in \Sigma, x \in \Sigma^*, \\ \textbf{T2} & 1^{n+1}0x \models 1^n x, & \forall n \in P, x \in \Sigma^*. \end{array}$$

Note that the number of rewriting rules is infinite, as it must be by [4], since the semiautomaton is infinite. By Theorem 5, the set of rewriting rules is irredundant. One might be tempted to replace **T2** by $10x \models x$; but then there is no proof of the equivalence $110 \equiv 1$, for example.

5.2 Counter with Nonempty Initial State

Our next example shows that the trivial semiautomaton operation of changing the initial state can lead to some complications in a semiautomaton specified by canonical words and equivalences.

Suppose that we want to change the initial state of the counter from the empty state to the state that contains two 1s. In the specification by automaton, this operation is entirely trivial. For example, in the automaton of Fig. 6 (a), instead of $S = (\Sigma, Q, \delta, 0)$, we now use $S = (\Sigma, Q, \delta, 2)$, and, for Fig. 6 (c), instead of $S = (\Sigma, \mathbf{X}, \delta_{\mathbf{X}}, \epsilon)$, we have $S = (\Sigma, \mathbf{X}, \delta_{\mathbf{X}}, 11)$. In the specification by canonical words, however, we need to find a new spanning forest, and recalculate the equivalences.

In Fig. 7 (a), we show the solution using the spanning tree corresponding to the canonical set $\{\epsilon, 0, 00, 000, 1, 11, 111, \ldots\}$. This solution has the disadvantage that the state label no longer corresponds to the current count. Also, we must calculate a new set of equivalences, in this case:

E1 $01 \equiv \epsilon$,
E2 $001 \equiv 0$,
E3 $000a \equiv 000, \quad \forall a \in \Sigma$,
E4 $1^{n+1}0 \equiv 1^n, \quad \forall n \in P$.

A second solution is shown in Fig. 7 (b), where we use the two trees corresponding to two sets of canonical words: $\{00, 000, 001\}$ and $\{11, 111, \ldots\}$. The advantage of this solution is that, except for three states, the state label denotes the contents of the counter. Now the equivalences are

E1 $110 \equiv 001$,
E2 $0011 \equiv 11$,
E3 $0010 \equiv 00$,
E4 $000a \equiv 000, \quad \forall a \in \Sigma$,
E5 $1^{n+1}0 \equiv 1^n, \quad \forall n \geq 2$.

To complete the specification, we must add the rule $\epsilon \equiv 11$ to take care of the new initial state. The acanonical words are $\epsilon \cup 0 \cup 1 \cup (01 \cup 10)\Sigma^*$. To find the canonical representative of any nonempty acanonical word w, we use the right-congruence property: $\epsilon \equiv 11$ implies $w \equiv 11w$, and then apply the transformation rules from **T** to $11w$, which is post-canonical. This approach would require us to test the given word for membership in the set of acanonical words. Alternatively, one can put $\chi(q_\epsilon)$ in front of *any* word, and then derive the canonical representative as above, thus avoiding the membership test, at the cost of one extra step in the derivation.

6 Conclusions

We have developed a theory of semiautomata which is based on canonical words and equivalences. The motivation for this came from trace-assertion specifications

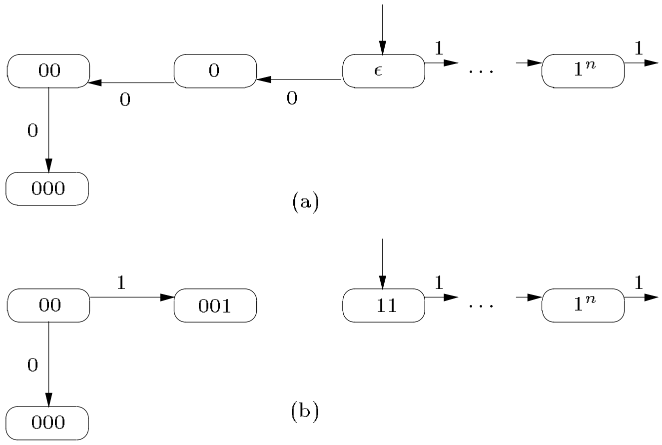

Fig. 7. Counter automaton with changed initial state

of software modules. For that application, our theory provides a simple foundation for the selection of canonical traces, the definition of trace equivalence, and the transformation of traces to canonical form. A key role in the theory is played by prefix-continuous languages, which include both prefix codes and prefix-closed languages as special cases. The use of prefix-continuous canonical languages, along with our constructions, guarantees that the process of rewriting any word to its canonical form is deterministic.

Acknowledgments:

We are very grateful to David Parnas for his insightful critical comments on earlier versions of this paper, and for a crucial example which challenged us to extend our theory beyond prefix-closed canonical sets; this led us to a complete characterization of the canonical sets for which the rewriting system is well behaved.

This research was supported by the Natural Sciences and Engineering Research Council of Canada under grants No. OGP0000871 and OGP0000243.

References

1. Bartussek, W. and Parnas, D.: Using Assertions About Traces to Write Abstract Specifications for Software Modules. Report No. TR77-012, University of North Carolina at Chapel Hill, December (1977) 26 pp. Reprinted in Software Fundamentals (Collected Works by D. L. Parnas), D. M. Hoffman and D. M. Weiss, eds., Addison-Wesley (2001) 9–28
2. Brzozowski, J. A. and Jürgensen, H.: Theory of Deterministic Trace-Assertion Specifications, Technical Report CS-2004-30, School of Computer Science, University of Waterloo, Waterloo, ON, Canada, May 2004: http://www.cs.uwaterloo.ca/cs-archive/CS-2004/CS-2004.shtml
3. Book, R. V. and Otto, F.: *String-Rewriting Systems.* Springer-Verlag, Berlin (1993)
4. Büchi, J. R.: Regular Canonical Systems. Archiv für Math. Logik und Grundlagenforschung, **6** (1964) 91–111
5. Büchi, J. R.: *Finite Automata, Their Algebras and Grammars*, (Siefkes, D., ed.), Springer-Verlag (1988)
6. Gécseg, F. and Peák, I.: Algebraic Theory of Automata. Akadémiai Kiadó, Budapest (1972)
7. Wang, Y. and Parnas, D. L.: Simulating the Behavior of Software Modules by Trace Rewriting. IEEE Trans. on Software Engineering, vol. 20, no. 10, October (1994) 750–759