CS–2007–03

Predictable Semiautomata

Janusz Brzozowski and Nicolae Santean

Technical Report 03

David R. Cheriton School of Computer Science
University of Waterloo

2007

Predictable semiautomata[⋆]

Janusz Brzozowski and Nicolae Santean

David R. Cheriton School of Computer Science
University of Waterloo,
Waterloo, ON, Canada N2L 3G1
{brzozo, nsantean}@uwaterloo.ca

Abstract

We introduce a new class of nondeterministic semiautomata: A nondeterministic semiautomaton $\mathscr{S}$ is *predictable* if there exists an integer $k \geq 0$ such that, if $\mathscr{S}$ knows the present input a and the next k inputs, then the transition under a is deterministic. Nondeterminism may occur only when the length of the unread input is less than $k+1$. We develop a comprehensive theory of predictable semiautomata. Using a novel semiautomaton, called the core, we present a test for predictability. We then introduce the predictor semiautomaton, based on a look-ahead semiautomaton, that is essentially deterministic. We describe two ways of using the predictor to simulate a nondeterministic semiautomaton. The first simulation predicts the set of states reachable by every prefix of the input word as long as the prefix is in the language of the semiautomaton. The second simulation is similar, but it stops as soon as it infers that the input word is not in the language of the semiautomaton. Moreover, the membership of a word in the language of a semiautomaton can be decided completely deterministically. Finally, we show that, if a semiautomaton with n states over a one-letter alphabet is k-predictable, k being the smallest such integer, then $k \leq n-1$, and this bound can be reached. For semiautomata over arbitrary alphabets, $k \leq (n^2-n)/2$, and this bound can be reached for a suitable input alphabet.

Key words: Automaton, delegator, look-ahead, nondeterminism, predictor, selector, semiautomaton, simulation

⋆ This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871 and fellowship No. PDF-32888-2006.

8 May 2007

1 Introduction

Nondeterministic automata are ubiquitous in theoretical computer science. They serve as models for various nondeterministic processes, constitute valuable design tools (often more convenient than their deterministic counterparts), and are inevitable in many applications. On the other hand, they also have some drawbacks, such as increased simulation time and space, and inefficient minimization algorithms.

Several attempts have been made recently to overcome the disadvantages of nondeterminism. Nondeterministic finite automata (NFA) have been used as formal models for service-oriented computing [1], and as tools for automated web service composition [2]. In both of these applications, it became imperative to overcome the problems introduced by nondeterminism. For this purpose, the concept of a "delegator" of an NFA was informally introduced in [2]. A delegator is an equivalent deterministic finite automaton (DFA) based on the transition graph of an NFA. It has a look-ahead buffer of a fixed length, and the look-ahead word permits it to determine which of several possible nondeterministic steps should be taken. This concept, also known as "look-ahead delegation", was studied systematically and in a more abstract framework in [5].

We address a problem similar to delegation, but we formulate it in the more general model of semiautomata. We introduce semiautomata, called "predictable", in which it is possible to replace a nondeterministic step by a deterministic one, with the aid of a bounded number of input letters from a look-ahead buffer. Our goal is to compute the set of states reached from the initial set of states of a semiautomaton by a given input word. Although our development is in terms of semiautomata, our results extend to automata as well, without resorting to the theory of delegators.

Our theory is substantially different from the work in [2,5]. Since our model addresses nondeterministic semiautomata, rather than automata, it takes advantage of their special properties, notably of the prefix-closure of their languages. Consequently, problems left open in [5] for NFAs are resolved in our framework. For example, the decidability of the NFA delegation is still open, whereas predictability of semiautomata is decidable, and we provide an algorithm for it. This algorithm uses a novel semiautomaton called "core".

As observed in [5], delegation appears to be a global automaton property, whereas our concept of predictability is a local property of nondeterministic branches that we call "forks". Consequently, our method can be applied at the fork level even to semiautomata which are not globally predictable, whereas an NFA which has no delegator cannot be partially determinized.

We modify a given semiautomaton by adding to it some look-ahead information; the resulting semiautomaton is called a "predictor". In contrast to [5], we do not always use the entire buffer content, but only as much information as is needed; hence we reduce the predictor's complexity. Moreover, a predictor computing the set of states reachable by a word does not completely determinize a semiautomaton, but may leave some nondeterminism at the end of its computations, when the remaining input is shorter than the buffer length. However, the decision concerning the membership of a word in the language of a semiautomaton can be made completely deterministically. Unlike delegators, for which simulation is determined by their definition, predictors can be simulated in two ways. The first simulation predicts the set of next states, as long as the input word has a prefix that is in the language of the semiautomaton. The second simulation stops as soon as it infers that the input word is not in the language of the semiautomaton.

Another difference between predictors and delegators is the uniqueness of the predictor: there is a bijection between semiautomata and predictors. In contract to this, an NFA can have many delegators that may be homomorphically unrelated.

We give a precise upper bound for the size of the predictor's look-ahead buffer; the bound is linear in the number of states of the semiautomaton for unary alphabets and quadratic for larger alphabets; nothing similar is known for delegators.

In view of these and other differences, our predictor has little in common with the delegation model, beside the motivation and the look-ahead paradigm.

The remainder of the paper is structured as follows. In Section 2, we introduce the terminology for semiautomata. Predictable semiautomata are defined in Section 3. The properties of certain types of words, called "minimal selectors" and "maximal nonselectors", and their relation to predictability are studied in Section 4. In Section 5, we define a deterministic semiautomaton, called "product", which provides a test for predictability. A simpler version of the product semiautomaton, called a "core", is described in Section 6; the core is used for finding minimal selectors and maximal nonselectors. The process of predicting reachable states is developed in Section 7, where a "predictor" of a semiautomaton is defined and two methods of simulating nondeterministic semiautomata are characterized. In Section 8 we derive bounds on the size of the look-ahead buffer, and Section 9 concludes the paper.

2 Semiautomata

We base our notation loosely on that of Eilenberg [4]. If $f : X \to Y$ (also denoted $X \xrightarrow{f} Y$) is a function, we write xf for the value of f at x. If $g : Y \to Z$ is another function, then xfg is unambiguous without parentheses. Also, an element $x \in X$ can be interpreted as a function $x : S \to X$, where S is some singleton, and the value of this function is x. Then xfg is the composition of functions $S \xrightarrow{x} X \xrightarrow{f} Y \xrightarrow{g} Z$. For a set X, we denote its cardinality by $X\#$.

If Σ is an alphabet, then Σ^+ and Σ^* denote the free semigroup and the free monoid, respectively, generated by Σ. The empty word is 1. For $k \geq 1$, let $\Sigma^{\leq k} = 1 \cup \Sigma \cup \ldots \cup \Sigma^k$. 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 and v is a *suffix* of w. A language L is *prefix-free* if no word of L is a prefix of another word of L. It is *prefix-closed* if $uv \in L$ implies $u \in L$. If $u \in \Sigma^*$, $v \in \Sigma^+$, then uv is an *extension* of u.

A *semiautomaton* [3] $\mathscr{S} = (\Sigma, Q, P, E)$ consists of an *alphabet* Σ, a set Q of *states*, a set $P \subseteq Q$ of *initial states*, and a set E of *edges* of the form (q, a, r), where $q, r \in Q$ and $a \in \Sigma$. An edge (q, a, r) *begins* at q, *ends* at r, and has *label* a. It is also denoted as $q \xrightarrow{a} r$. A *path* π is a finite sequence $\pi = (q_0, a_1, q_1)(q_1, a_2, q_2) \ldots (q_{k-1}, a_k, q_k)$ of consecutive edges, $k > 0$ being its *length*, q_0, its *beginning*, q_k, its *end*, and $w = a_1 \ldots a_k$, its *label*. We also write $q_0 \xrightarrow{w} q_k$ for π. Each state q has a *null path* 1_q from q to q with label 1.

If $T \subseteq Q$ and $w \in \Sigma^*$, then $Tw = \{q \in Q \mid t \xrightarrow{w} q, \text{ for some } t \in T\}$. If $T = \{t\}$, we write tw for Tw; if $Tw = \{q\}$, we write $Tw = q$. A state q of a semiautomaton $\mathscr{S}$ is *accessible* if there exists $p \in P, w \in \Sigma^*$ such that there is a path $p \xrightarrow{w} q$, that is, if $q \in pw$. A semiautomaton is *accessible* if all of its states are accessible.

The *language* $|\mathscr{S}|$ *of a semiautomaton* $\mathscr{S} = (\Sigma, Q, P, E)$ is the set of all labels of paths starting in initial states of $\mathscr{S}$, that is, $|\mathscr{S}| = \{w \in \Sigma^* \mid Pw \neq \emptyset\}$. Note that $|\mathscr{S}|$ is prefix-closed; in particular, if $|\mathscr{S}| \neq \emptyset$, then $1 \in |\mathscr{S}|$. If q is a state of $\mathscr{S} = (\Sigma, Q, P, E)$, the *language of* q is $R_q = \{w \in \Sigma^* \mid qw \neq \emptyset\}$. The *language of a set* $T \subseteq Q$ is $R_T = \bigcup_{t \in T} R_t$. In particular, $R_P = |\mathscr{S}|$.

A semiautomaton is *complete* if $P \neq \emptyset$ and, for every $q \in Q$ and $a \in \Sigma$, there is an edge $(q, a, r) \in E$, for some $r \in Q$. In a complete semiautomaton, $qw \neq \emptyset$, for all $q \in Q, w \in \Sigma^*$. The language of a complete semiautomaton is Σ^*. If $\mathscr{S}$ is complete, each $w \in \Sigma^*$ belongs to every language R_q, $q \in Q$.

A semiautomaton $\mathscr{S}$ is *deterministic* if it has at most one initial state, and for every $q \in Q$, $a \in \Sigma$, there is at most one edge (q, a, r). If $\mathscr{S}$ is deterministic and has initial state p, we write $\mathscr{S} = (\Sigma, Q, p, E)$.

3 Predictable semiautomata

We introduce nondeterministic semiautomata, called "predictable", in which the knowledge of a limited number of symbols read ahead from the input tape removes nondeterminism. We restrict our attention to finite semiautomata.

Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton. If $q \in Q$, $a \in \Sigma$, then a *fork* (with origin q and input a) is the set $\langle q, a \rangle = \{(q, a, r_1), \ldots, (q, a, r_h)\}$ consisting of all the edges from q labeled a. The set $\langle\langle q,a\rangle\rangle = \{r_1,\dots,r_h\}$ is called the *fork set* of $\langle q,a\rangle$. We assume that $h > 0$, since empty forks are of no interest. Note, however, that forks with single edges are permitted; they are called *deterministic transitions.* Allowing such forks has the advantage that a semiautomaton can be viewed as a set of initial states and a set of forks.

A set $T \subseteq Q$ is *critical* if either $T = P$ or $T = \langle\langle q,a\rangle\rangle$, for a fork $\langle q,a\rangle$ in $\mathscr{S}$. A critical set is the set of all possible next states in a step of a (deterministic or nondeterministic) computation.

The following definition states the conditions under which it is possible to decide which state of a critical set, if any, should be chosen, if we know the next k symbols on the input tape:

Definition 1 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton, and let $k \geq 0$ be an integer. A set $T \subseteq Q$ is k-*predictable *if any two distinct states s,t of T satisfy*

$$R_s \cap R_t \cap \Sigma^k = \emptyset.$$

*A semiautomaton $\mathscr{S}$ is k-*predictable *if every critical set of $\mathscr{S}$ is k-predictable, and $\mathscr{S}$ is* predictable *if it is k-predictable for some k.*

The condition of k-predictability can be satisfied in two ways. First, if R_T has no words of length k, then the condition cannot be violated. This means that no path of length k spelling w can originate in any state of T. Second, if $w \in R_s \cap \Sigma^k$ for some $s \in T$, and w does not belong to any R_t, with $s \neq t$, then the condition holds again. Now state s is the only state in T from which there is a path spelling w.

A set is 0-predictable if and only if it consists of a single state. Consequently, a semiautomaton is 0-predictable if and only if it is deterministic. A predictable semiautomaton is either deterministic or incomplete, because in a complete semiautomaton, $R_s = R_t = \Sigma^*$ and hence $R_s \cap R_t \cap \Sigma^k = \Sigma^k \neq \emptyset$, for all states s and t.

Example 1 *The fork $\{(p,a,q)\}$ in Fig. 1 (a) is a deterministic transition, and the fork $\langle q,a\rangle = \{(q,a,q),(q,a,r)\}$ has fork set $\langle\langle q,a\rangle\rangle = \{q,r\}$. This set is 1-predictable, since a word of length 1 (here, only a) belongs only to R_q, and not to R_r. The fork set $\{q,r\}$ in Fig. 1 (b) is 1-predictable, because there are no words of length 1 in R_q or R_r. Thus the semiautomata of Fig. 1 (a) and (b) are 1-predictable. The fork set $\{p,q\}$ in Fig. 1 (c) is not k-predictable for any $k \geq 0$, because $a^k \in R_p \cap R_q \cap \Sigma^k$ for all k.*

Remark 1 *If a set is k-predictable, then it is k'-predictable for all $k' > k$.*

By the definition of k-predictable sets, testing for k-predictability is reduced to testing whether a finite language is empty, and this problem is decidable.

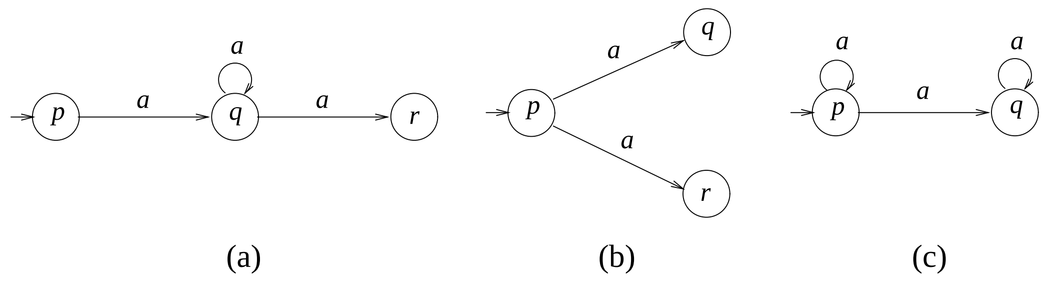

Fig. 1. Illustrating predictability.

4 Selectors and nonselectors

We now define two types of words that play an important role in predictability: “selectors” and “nonselectors”. Selectors are look-ahead words that permit us to choose only one state from a set T, whereas nonselectors limit the choice to a subset of T that has at least two states.

Definition 2 *If $\mathscr{S} = (\Sigma, Q, P, E)$ is semiautomaton, and $T \subseteq Q$, then a word $w \in \Sigma^*$ is a t-*selector in T *if $w \in \sigma(t,T)$, where*

$$\sigma(t,T) = \Big(R_t \setminus \bigcup_{s\in T, s\neq t} R_s\Big).$$

A word w is a selector in T *if it is a t-selector in T for some t. The set of all selectors in T is*

$$\sigma(T) = \bigcup_{t\in T} \sigma(t,T).$$

A selector w in T is minimal *if no prefix of w is a selector in T.*

*We also define the complementary set $\overline{\sigma}(t,T)$ of t-*nonselectors *in T:*

$$\overline{\sigma}(t,T) = R_t \setminus \sigma(t,T).$$

The set of all nonselectors *in T is*

$$\overline{\sigma}(T) = \bigcup_{t\in T} \overline{\sigma}(t,T) = R_T \setminus \sigma(T) = \bigcup_{s,t\in T, s\neq t} (R_s \cap R_t).$$

A t-nonselector u is maximal *if no extension of u is in R_t.*

Example 2 *In Fig. 1 (a), the set of q-selectors in the fork set $\langle\langle p,a\rangle\rangle = \{q\}$ is a^*, and 1 is the only minimal q-selector in $\{q\}$. There are no q-nonselectors in $\{q\}$. Fork $\langle q,a\rangle$ has critical set $T = \{q,r\}$. The set of q-selectors in T is a^+, and a is a minimal q-selector in T. The empty word 1 is the only q-nonselector in T, and it is not maximal because $a = 1a \in R_q$. There are no r-selectors in T, and 1 is the only r-nonselector in T; it is maximal because no extension of 1 is in R_r.*

In Fig. 1 (b), the fork set of fork $\langle p,a\rangle$ is $T = \{q,r\}$. Here $R_T = \{1\}$, there are no selectors, and 1 is a maximal q-nonselector in T and a maximal r-nonselector in T. Thus, there exist sets that are predictable and yet have no selectors. Also, a set that is not predictable may have selectors, as we shall see later.

In Fig. 1 (c), there is a fork $\langle p,a\rangle$ with fork set $T = \{p,q\}$. There are no selectors, since $R_p \cap R_q = a^*$. Every word in a^* is a p-nonselector and a q-nonselector, and there are no maximal nonselectors.

Example 3 *The semiautomaton $\mathscr{S}$ of Fig. 2 illustrates the usefulness of minimal selectors and maximal nonselectors. The only critical set with more that one element is $P = \{p_1, p_2, p_3\}$. One verifies that $\mathscr{S}$ is 2-predictable. There is a minimal p_1-selector aa, and maximal nonselectors a for p_2, and 1 for p_3. Minimal selectors and maximal nonselectors are indicated by square brackets and "floor" brackets, respectively; thus $[aa]$ is a minimal selector and $\lfloor a\rfloor$ is a maximal nonselector.*

If the input word to $\mathscr{S}$ is 1, then any state in P can be the initial state, and there is no further computation. If the input word is a, then the initial state could not be p_3, but is limited to $\{p_1, p_2\}$. Finally, if the input word begins with aa, then the initial state is necessarily p_1.

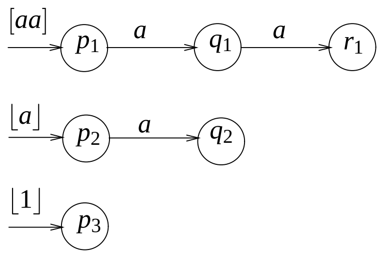

Fig. 2. Selectors and nonselectors.

Selectors and nonselectors have the following prefix properties:

Proposition 1 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton and $T \subseteq Q$.*

(1) *The set of all nonselectors in T is prefix-closed.*
(2) *If an s-selector u is a prefix of a t-selector w, then $s = t$.*
(3) *The set of all minimal selectors in T is prefix-free.*
(4) *No selector is a prefix of a nonselector.*
(5) *For any $t \in T$, no maximal t-nonselector is a prefix of a t-selector.*
(6) *For any $t \in T$, the set of all maximal t-nonselectors is prefix-free.*

Proof:

(1) If w is a nonselector, there exist $s,t \in T$, such that $w \in R_s \cap R_t$. Since R_s and R_t are prefix-closed, we have $u \in R_s \cap R_t$, for every prefix u of w.
(2) This follows because $w \in R_t$ implies $u \in R_t$, since R_t is prefix-closed.
(3) This follows from the definition of minimal selector.

(4) This follows from (1).

(5) If u is a maximal t-nonselector, then $ua \notin R_t$, for all $a \in \Sigma$. Hence no extension of a maximal t-nonselector is in R_t.

(6) This follows by the same reasoning as (5). □

The next result provides three characterizations of k-predictability.

Theorem 1 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton and $T = \{t_1, \ldots, t_h\} \subseteq Q$. The following are equivalent:*

(1) T is k-predictable.

(2) Every word of length k in R_T is a selector in T.

(3) Every word of length $\geq k$ in R_T is a selector in T, and hence has a minimal selector in T as a prefix.

(4) Every nonselector in T is of length $< k$.

Proof:

(1) ⇒ (2) Suppose $w \in \Sigma^k$. If w is nonselector in T, then $w \in R_s \cap R_t \cap \Sigma^k$, for some $s, t \in T$, contradicting (1). Hence w must be a selector in T.

(2) ⇒ (3) Every word w of length $\geq k$ in R_T has a prefix u of length k, and u is a selector in T by (2). By Proposition 1 (4), w must be a selector. Then u and w have a minimal selector in T as prefix.

(3) ⇒ (4) If w is a nonselector in T, then $w \in R_T$. Thus $|w| < k$; otherwise, w would be a selector by (3).

(4) ⇒ (1) If a longest nonselector in T is of length $< k$, then $R_s \cap R_t \cap \Sigma^k = \emptyset$, for all $s, t \in T, s \neq t$, and T is k-predictable. □

It follows that testing whether a set is predictable is equivalent to testing whether the regular language $\overline{\sigma}(T)$ is finite, and the latter property is decidable.

Corollary 1 *If T is a k-predictable set of a semiautomaton $\mathscr{S}$, then every minimal selector in T is of length $\leq k$.*

Proposition 2 *Let $\mathscr{S} = (\Sigma, Q, P, E)$, $T \subseteq Q$, and $t \in T$. If T is k-predictable, t has either a minimal t-selector in T or a maximal t-nonselector in T.*

Proof: If t has a selector in T, then it has a minimal selector in T. Assume now that t has no selectors in T. If R_t is finite, let w be a longest word in R_t, necessarily a nonselector. Then $wa \notin R_t$ for all $a \in \Sigma$, and w is a maximal t-nonselector in T. By Theorem 1 (4), the case where R_t is infinite is impossible. □

5 Product semiautomata

We now describe a semiautomaton construction which leads to a test for predictability. To determine the predictability of a set $T = \{t_1, \ldots, t_h\} \subseteq Q$ in a semiautomaton $\mathscr{S} = (\Sigma, Q, P, E)$, we need to find intersections of the languages R_{t_i}, where R_{t_i} is the language of the semiautomaton $\mathscr{S}_i = (\Sigma, Q, t_i, E)$, $t_i \in T$. For this, we could determinize $\mathscr{S}_i$, and construct their direct product $\widehat{\mathscr{D}}(T)$. However, it is also possible to obtain a deterministic direct product by using the subset construction in each step of the direct product construction.

When T is fixed, and there is no danger of ambiguity, we use the term "selector" and "nonselector" instead of "selector in T" and "nonselector in T".

For a set Q, let 2^Q be the set of all subsets of Q. The direct product of h copies of 2^Q is denoted $(2^Q)^h$.

Definition 3 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton and let $T = \{t_1, \ldots, t_h\} \subseteq Q$. Define the deterministic semiautomaton*

$$\widehat{\mathscr{D}}(T) = (\Sigma, (2^Q)^h, \gamma_0, \widehat{E}_{\mathscr{D}}),$$

where $\gamma_0 = (\{t_1\}, \ldots, \{t_h\})$, and, for every h-tuple $(S_1, \ldots, S_h)$ of sets of states of $\mathscr{S}$ and every $a \in \Sigma$, there is an edge $((S_1, \ldots, S_h), a, (S_1 a, \ldots, S_h a)) \in \widehat{E}_{\mathscr{D}}$, where $S_i a$ is the set of successor states of the set S_i under input a in the semiautomaton $\mathscr{S}$.

The product semiautomaton *for T is the accessible subsemiautomaton of $\widehat{\mathscr{D}}(T)$, and it is denoted by*

$$\mathscr{D}(T) = (\Sigma, \Gamma, \gamma_0, E_{\mathscr{D}}).$$

Note that $\widehat{\mathscr{D}}(T)$ and $\mathscr{D}(T)$ are complete.

We distinguish several types of states in Γ:

- The state $\gamma_\emptyset = (\emptyset, \ldots, \emptyset) \in (2^Q)^h$ is called *null.*
- A state in which only the ith component is nonempty is called t_i*-singular.* A state is *singular* if it is t_i-singular for some i.
- Any state in which at least two components are nonempty is called *plural.* A plural state in which the ith component is nonempty is called t_i*-plural.*
- A state γ is t_i*-ultimate* if it is t_i-plural and, for all $a \in \Sigma$, the ith component of γa is empty. A state is *ultimate* if it is t_i-ultimate for some i.
- A state is *cyclic* if it appears in a cycle; otherwise, it is *noncyclic.*

Since $\mathscr{D}(T)$ is deterministic, each word defines a unique path. We define several types of words:

- A word w defining a path $(\gamma_0, a_1, \gamma_1) \ldots (\gamma_{m-1}, a_m, \gamma_m)$, where $\gamma_0, \ldots, \gamma_{m-1}$ are plural and $\gamma_m = \gamma_\emptyset$, is called *nullary*.
- A word w defining a path $(\gamma_0, a_1, \gamma_1) \ldots (\gamma_{m-1}, a_m, \gamma_m)$, where $\gamma_0, \ldots, \gamma_{m-1}$ are plural and γ_m is t_i-singular, is called *t_i-primary*. If such a word exists, state γ_m is also called *t_i-primary*. A word or state is *primary* if it is t_i-primary for some i.
- A word w is *t_i-plural* if $\gamma_0 w$ is t_i-plural; it is *plural* if $\gamma_0 w$ is plural.

The types of states in a product semiautomaton are illustrated in Fig. 3. The "core" part is discussed in the next section.

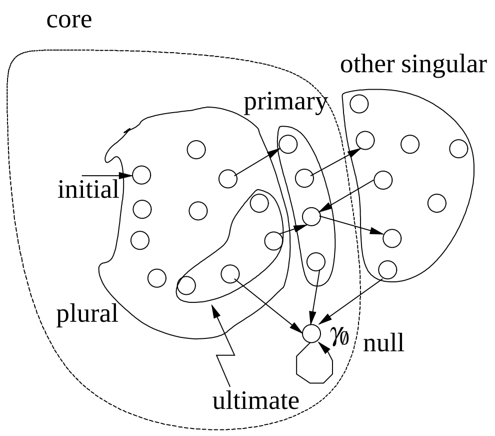

Fig. 3. States in a product semiautomaton.

The next result states some basic properties of product semiautomata and their relations to selectors and nonselectors.

Proposition 3 *Let $\mathscr{D}(T)$ be the product semiautomaton of a set T in a semiautomaton $\mathscr{S}$. Then the following hold:*

(1) Let $\gamma = (S_1, \ldots, S_h)$ and $\gamma' = (S'_1, \ldots, S'_h)$ be two states in Γ such that $\gamma' = \gamma w$, for some $w \in \Sigma^*$. If $S_i = \emptyset$, for some $i \in \{1, \ldots, h\}$, then also $S'_i = \emptyset$.
(2) A word w is in the language R_T if and only if $\gamma_0 w \neq \gamma_\emptyset$.
(3) A word w is a t_i-selector if and only if $\gamma_0 w$ is t_i-singular.
(4) A word is a minimal t_i-selector if and only if it is t_i-primary.
(5) A word w is a t_i-nonselector if and only if $\gamma_0 w$ is t_i-plural.
(6) A word w is a maximal t_i-nonselector if and only if $\gamma_0 w$ is t_i-ultimate.

Proof: Properties (1)–(3) and (5) follow from the definition of $\mathscr{D}(T)$. For (4), if w is a minimal t_i-selector, then $\gamma_0 w$ is t_i-singular by (3). If w has a proper prefix u, then u it must be a nonselector. Thus every state of the form $\gamma_0 u$ is plural, and hence w is primary. Conversely, if w is primary, then it defines a path $(\gamma_0, a_1, \gamma_1) \ldots (\gamma_{m-1}, a_m, \gamma_m)$, where $\gamma_0, \ldots, \gamma_{m-1}$ are plural and γ_m is singular. Therefore no proper prefix of w is a selector, and w is minimal.

For (6), if $\gamma = \gamma_0 w$ is t_i-ultimate, then γ is t_i-plural. Since w is not a t_i-selector and $w \in R_{t_i}$, w is a t_i-nonselector. Because every extension wa leads to a state with an empty ith component, $wa \notin R_{t_i}$, and no extension wau is in R_{t_i}, since R_{t_i} is prefix-closed. Hence w is maximal. Conversely, if w is a maximal t_i-nonselector, then $w \in R_{t_i}$ and $w \in R_{t_j}$ for some $j \neq i$. Hence $\gamma_0 w$ is a state with nonempty ith and jth components. If $\gamma_0 w$ is not t_i-ultimate, then there exists $a \in \Sigma$ such that $\gamma_0 wa$ has a nonempty ith component. But then $wa \in R_{t_i}$ and w is not maximal. Therefore $\gamma_0 w$ is t_i-ultimate. $\square$

Example 4 *Figure 4 (a) shows a semiautomaton with one initial state and one fork $\langle p, b \rangle$, with fork set $T = \{p, q\}$. The product semiautomaton $\mathscr{D}(T)$ is given in Fig. 4 (b), where, for simplicity, we represent sets of states as words; for example, $\{p, q\}$ is written pq. There is an infinite number of primary words (and hence of minimal selectors); the set of all such words is denoted by the regular expression: $(aab)^*(b + ab + aaa)$. However, there are only three primary states $(pq, \emptyset)$, $(\emptyset, q)$, and $(\emptyset, t)$. Note that a primary state may be also reached by words that are not primary. For example, $(\emptyset, t)$ can be reached by aba. There are no nullary words.*

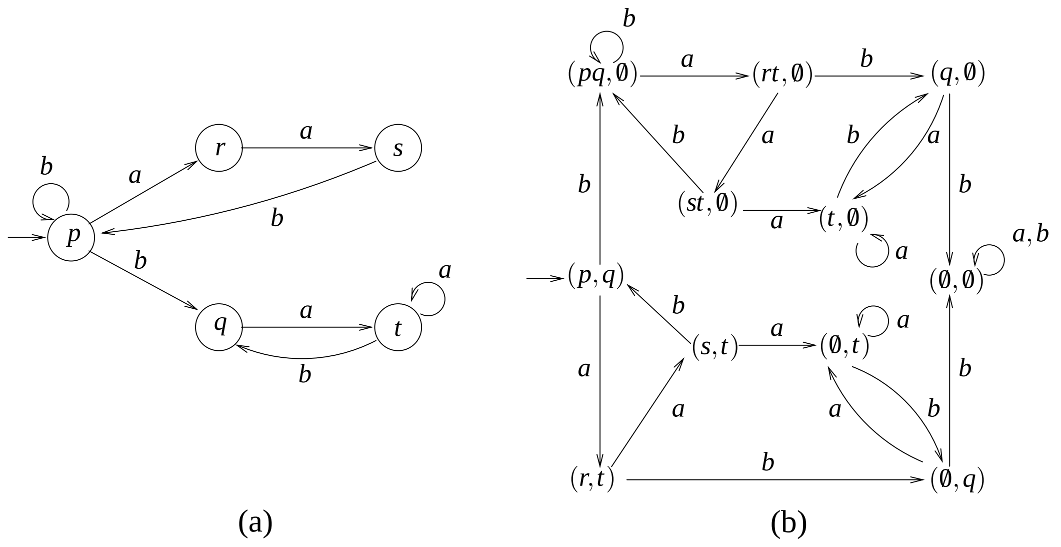

Fig. 4. An unpredictable semiautomaton and its product semiautomaton.

Product semiautomata that do not have any cyclic plural states are of particular interest, since they lead to a test for predictability.

Theorem 2 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ and $T \subseteq Q$. The following are equivalent:*

(1) The length of a longest plural word in $\mathscr{D}(T)$ is $k - 1$.
(2) T is k-predictable, but not $(k-1)$-predictable.

Moreover, T is predictable if and only if $\mathscr{D}(T)$ does not have cyclic plural states.

Proof: Let $k - 1$ be the length of a longest plural word w in $\mathscr{D}(T)$. Then w is a nonselector in T, by Proposition 3 (5). By Theorem 1, T is not $(k-1)$-predictable. Since there are no plural words of length k in $\mathscr{D}(T)$, every word u of length k is either nullary or singular. In the first case, $u \notin R_T$ by Proposition 3 (2). In the second case, u is a selector by Proposition 3 (3). Thus every word of length k in R_T is a selector, and T is k-predictable by Theorem 1. Hence (1) implies (2).

If T is k-predictable, then all nonselectors in T and (by Proposition 3 (5)) all plural words in $\mathscr{D}(T)$ are of length $< k$, by Theorem 1. If T is not $(k-1)$-predictable, then R_T must have a nonselector of length $k-1$, and hence there is a plural word of that length in $\mathscr{D}(T)$. Hence (2) implies (1).

If $\mathscr{D}(T)$ has a cyclic plural state γ, then γ has two nonempty components, say i and j. Since $\mathscr{S}$ is accessible, γ is reachable by some word $u \in \Sigma^*$ from the initial state γ_0. Since γ is cyclic, there is a word $v \in \Sigma^*$ such that $\gamma v = \gamma$. This implies that $uv^n \in R_{t_i} \cap R_{t_j}$ for all n. Since n can be arbitrarily large, the set $\overline{\sigma}(T)$ of all nonselectors in T is not finite and T is not predictable, by Theorem 1. Thus, if T is predictable, then $\mathscr{D}(T)$ has no cyclic plural states.

If $\mathscr{D}(T)$ has no cyclic plural states, then the length of a longest plural word (and hence of the longest nonselector) is $k-1$, for some k. By Theorem 1, T is k-predictable and hence predictable. This proves the second claim. $\square$

6 Core semiautomata

We now show that, for predictable semiautomata, a part of the product semiautomaton $\mathscr{D}(T) = (\Sigma, \Gamma, \gamma_0, E_{\mathscr{D}})$ suffices to give us all the information we need. Let Γ_{pl} (respectively, Γ_{pr}) be the set of all plural (respectively, primary) states of Γ.

Definition 4 *The* core semiautomaton *of a product semiautomaton $\mathscr{D}(T)$ is an incomplete deterministic semiautomaton $\mathscr{C}(T) = (\Sigma, \Omega, \gamma_0, E_{\mathscr{C}})$, where*

$$\Omega = \begin{cases} \Gamma_{pl} \cup \Gamma_{pr} \cup \{\gamma_\emptyset\} & \text{if there is an edge from a plural state to } \gamma_\emptyset, \\ \Gamma_{pl} \cup \Gamma_{pr} & \text{otherwise,} \end{cases}$$

and $E_{\mathscr{C}}$ consists of edges of $\mathscr{D}(T)$ that join a plural state to a plural state, a primary state, or $\gamma_\emptyset$.

Example 5 *Consider the semiautomaton of Fig. 4 (a). State (p,q) is cyclic in the product semiautomaton $\mathscr{D}(T)$ of Fig. 4 (b). The construction of $\mathscr{D}(T)$ could stop as soon as this cycle is detected. By Theorem 2, the semiautomaton of Fig. 4 (a) is not predictable.*

Example 6 *The semiautomaton $\mathscr{S}$ of Fig. 5 (a) has one critical set that is not a singleton, namely, $T = \{p,q\}$, corresponding to the fork $\langle p,a\rangle$. The product semiautomaton is shown in Fig. 5 (b). Since no plural state is cyclic, $\mathscr{S}$ is predictable. The core semiautomaton $\mathscr{C}(T)$ is shown in Fig. 6. Since the length of a longest plu-*

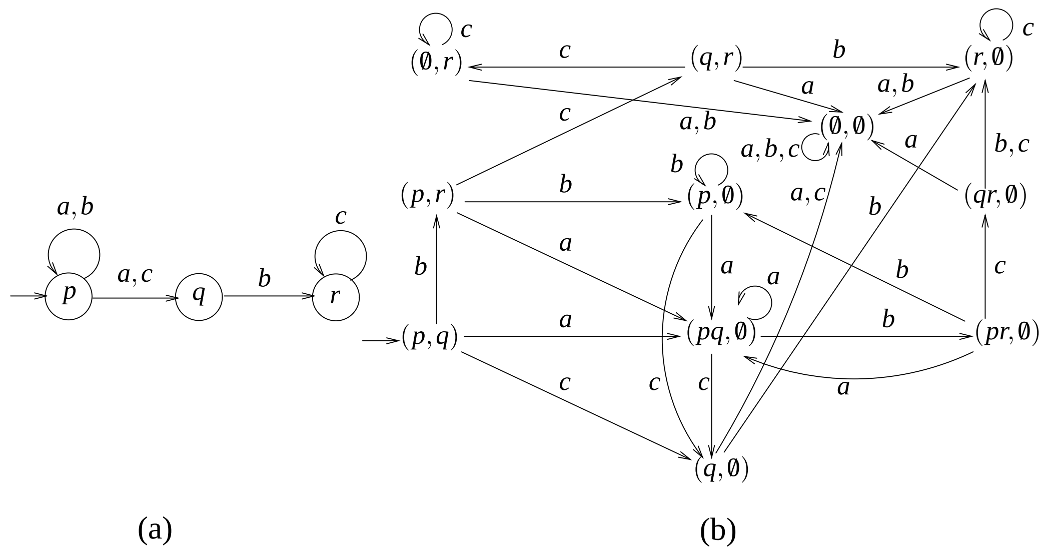

Fig. 5. A predictable semiautomaton and its product semiautomaton.

ral word is 2, the set $\{p,q\}$ *and* $\mathscr{S}$ *are 3-predictable by Theorem 2. The primary words are: a, c, ba, bb, bcb, and bcc. The minimal p-selectors in* T *are a, c, ba, bb and bcb, and the only minimal q-selector in* T *is bcc. The nonselectors in* T *are* 1*, b, and bc, and none is maximal. There is one nullary word bca. In each deterministic transition in Fig. 5 (a),* 1 *is a minimal selector.*

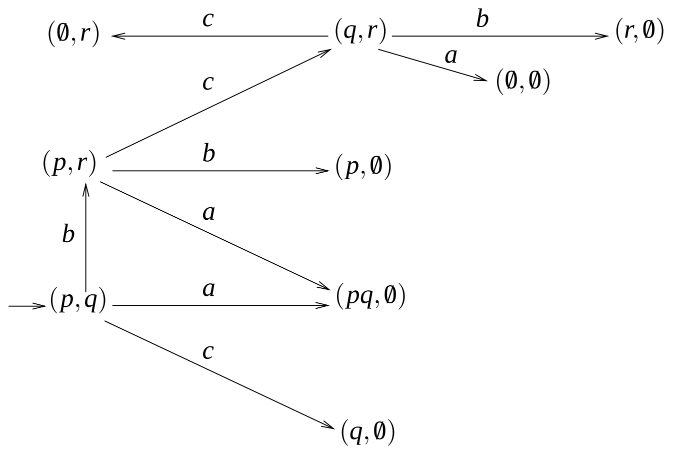

Fig. 6. The core semiautomaton of the product semiautomaton in Fig. 5 (b).

Example 7 *In the semiautomaton of Fig. 7 there are two initial states* q_1 *and* q_6 *and two forks. The core semiautomata corresponding to the critical sets are shown in Fig. 8. The critical set* $\{q_1,q_6\}$ *has minimal* q_1*-selectors a, ba, and bb, and maximal* q_6*-nonselector b. In Fig. 7, minimal selectors and maximal nonselectors of a state are shown on the arrows leading to the state. The critical set* $\{q_2,q_3\}$ *has minimal* q_2*-selectors a and bb, and minimal* q_3*-selector ba. The critical set* $\{q_4,q_5,q_6\}$ *has minimal* q_4*-selector a, minimal* q_6*-selector b, and maximal* q_5*-nonselector* 1*. The empty word* 1 *is a minimal selector in each deterministic transition. The semiautomaton is 2-predictable.*

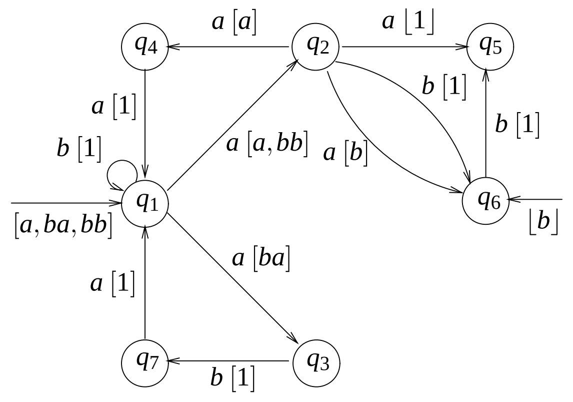

Fig. 7. Illustrating selectors and nonselectors.

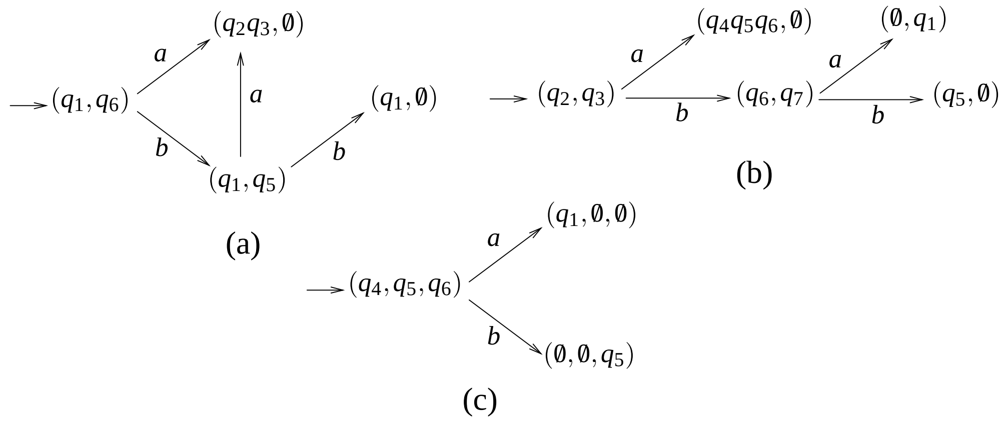

Fig. 8. Core semiautomata for Example 7.

7 Predictors

The concepts of the previous sections are now used to simulate a predictable semiautomaton almost deterministically. Starting with a semiautomaton $\mathscr{S}$, we define a semiautomaton $\mathscr{P}$ that has $\Sigma \times \Sigma^{\leq k}$ as input alphabet; the new input consists of the current input letter a and up to k letters of look-ahead information.

Definition 5 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a k-predictable semiautomaton, $k \geq 0$. The* predictor *of $\mathscr{S}$ is a semiautomaton $\mathscr{P}(\mathscr{S}) = \mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$, where*

(1) The set of initial states is P. The sets of minimal p-selectors and maximal p-nonselectors in P are associated with each state $p \in P$.
(2) If $\langle q, a \rangle$ is a fork, and (q, a, r), an edge in $\mathscr{S}$, then $(q, (a, [u]), r) \in E_{\mathscr{P}}$, if u is a minimal r-selector, and $(q, (a, \lfloor u \rfloor), r) \in E_{\mathscr{P}}$, if u is a maximal r-nonselector.

By Proposition 2, each state t in any set $T \subseteq Q$ has either a minimal selector or a maximal nonselector u. In particular, each state in P and in $\langle\langle q, a \rangle\rangle$, for each $q \in Q$, $a \in \Sigma$, has a minimal selector or a maximal nonselector.

Remark 2 *There is a bijective correspondence between predictable semiautomata and predictors. Each predictable semiautomaton uniquely defines a predictor. To reconstruct the semiautomaton from the predictor replace all edges of the form $(q,(a,\lceil u \rceil),r)$ and $(q,(a,\lfloor u \rfloor),r)$ by a single edge (q,a,r).*

7.1 Keys

One objective of a predictor is to find the set of all states reachable from the set of initial states, and to do this with as little nondeterminism as possible. For this purpose, we first study prefixes of the input word that provide useful look-ahead information.

Definition 6 *In a predictor $\mathscr{P}$, for a word $w \in \Sigma^*$ and $T \subseteq Q$, the longest prefix x of w which is also a prefix of a minimal selector or a maximal nonselector of a state in T is the* key *of w in T. The key* applies *to a state $t \in T$ if it is a prefix of a minimal t-selector or a maximal t-nonselector.*

The key always exists, since 1 is a prefix of every word. For every $T \subseteq Q$ and $w \in \Sigma^*$, there is always at least one state $t \in T$ to which the key applies.

Remark 3 *If T is a k-predictable set and w is an arbitrary word, then the key of w in T must belong to R_T. Consequently, if w' is the longest prefix of w that is in R_T, then the keys of w and w' in T coincide.*

The next result characterizes words in R_t, where $t \in T$, and T is k-predictable.

Lemma 1 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton, and let $T \subseteq Q$ be k-predictable. If $w \in R_t$, for some state $t \in T$, then one of the following conditions holds:*

(1) A prefix u of w is a minimal t-selector.
(2) $|w| < k$, and w is a prefix of a minimal t-selector.
(3) $|w| < k$, and w is a prefix of a maximal t-nonselector.

Proof: If $|w| \geq k$, then w has a prefix which is a minimal selector, by Theorem 1 (3). If $|w| < k$, and w is a selector, then it has a prefix which is a minimal selector, by the definition of the latter. Assume now that w is a t-nonselector and $|w| < k$. Consider any extension wx of w. This extension can be a t-selector, a t-nonselector, or not in R_t. If wx is a t-selector, then it has a prefix u which is a minimal t-selector. Now u cannot be a prefix of w, since no selector is a prefix of a nonselector, by Proposition 1 (4). Hence u is an extension of w, and (2) holds. If neither (1) nor (2) holds, then all extensions of w are either t-nonselectors or are not in R_t. If, for all $a \in \Sigma$, the extension wa is not in R_t, then w is a maximal t-nonselector. Otherwise, there is an a such that wa is a t-nonselector. Continuing with this argument we obtain longer and longer t-nonselectors. By Theorem 1 (4), every nonselector is of length less than k. Therefore we must eventually reach a maximal t-nonselector, and (3) holds. $\square$

The next lemma provides a characterization of keys.

Lemma 2 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton, let $T \subseteq Q$ be k-predictable and let $w \in \Sigma^*$ be an input word. Then the following holds:*

(1) If w' is the longest prefix of w that is in R_T, then the key of w in T is either a minimal selector or it is w' itself.
(2) If w' is an arbitrary prefix of w that is in R_T, and $t \in T$, then $w' \in R_t$ if and only if the key of w' in T applies to t.

Proof: Let w' be the longest prefix of w that is in R_T. Then there exists $t \in T$ such that $w' \in R_t$, and Lemma 1 applies; thus one of the three cases occurs. If a prefix u of w' is a minimal selector, then u is the key of w', and also of w, in T. This follows from Remark 3 and the fact that no minimal selector or maximal nonselector can be an extension of a minimal selector, by Proposition 1. If one of the other two conditions of Lemma 1 holds, w' is a prefix of a minimal t-selector or of a maximal t-nonselector. Then clearly w' is the key of w in T, since no prefix of w longer than w' is in R_T, again by Remark 3. This proves (1).

For the second claim, suppose w' is an arbitrary prefix of w that is in R_T. We consider two main cases:

- If w' has a prefix u which is a minimal selector in T, by the reasoning used in the proof of (1) above, u is the key of w' in T. Now, if $w' \in R_t$, for some $t \in T$, then u is a minimal t-selector and u applies to t, since u is its own prefix. Conversely, if the key u of w' in T applies to t, then w' can only be in R_t, since u is then a minimal t-selector, being a minimal selector in T.
- Now suppose that w' does not have a prefix which is a minimal selector. By Theorem 1 (3), we must have $|w'| < k$.

  If $w' \in R_t$ for some $t \in T$, either condition (2) or condition (3) of Lemma 1 holds. Hence w' is a prefix of a minimal t-selector or a maximal t-nonselector u in T. Since w' is its own longest prefix, the key in this case is w' itself, and w' applies to t.

  Conversely, assume that the key x of w' in T applies to $t \in T$; then x is a prefix of a minimal t-selector u or of a maximal t-nonselector u'. In either case, u and u' are in R_t, and so is the prefix x. We claim that $x = w'$; from this it follows that $w' \in R_t$.

  To prove the claim, note that, if $w' \notin R_t$, then $w' \in R_s$ for some $s \in T$, since $w' \in R_T$. Since $w' \in R_s$, either (2) or (3) of Lemma 1 holds, and w' is a prefix of a minimal s-selector or a maximal s-nonselector. It is its own key in T, since it is its own longest prefix. Thus our claim that $x = w'$ holds. $\square$

The purpose of the first simulation is to compute the set of states that can be reached by any prefix $w' \in |\mathscr{S}|$ of the input word w; if a prefix w' is not in $|\mathscr{S}|$, then the set of states reached is empty. The predictor continues looking for the next state, until it reaches the longest prefix of w that is in $|\mathscr{S}|$. This is done even though in some cases the predictor may know that the remaining input word is not in the language of the semiautomaton; we call this *maximal* simulation.

Definition 7 *Given a predictor $\mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$ of a k-predictable semiautomaton $\mathscr{S} = (\Sigma, Q, P, E)$ and an input word w, a prefix y of w* derives *a state $s \in Q$, written $y \Rightarrow s$, as follows:*

1. *(1) Basis Step (Step 0):*
   $1 \Rightarrow s$ if $s \in P$ and the key of w in P applies to s.
2. *(2) Induction Step (Step $m+1$, $m \geq 0$):*
   The induction is on the number $m \geq 0$ of derivation steps. Assume now that $w = yaz$, for some $a \in \Sigma$, $y, z \in \Sigma^*$; then $ya \Rightarrow s$ if $y \Rightarrow r$, for some $r \in Q$, $s \in \langle\langle r, a \rangle\rangle$, and the key of z in $\langle\langle r, a \rangle\rangle$ applies to s.

There may be more that one state that can be derived in each step. We pick an arbitrary state, and continue the derivation. If we wish to find all the states derivable by a given word, then we must backtrack and eventually consider all the choices. However, that most of the derivation turns out to be deterministic, and nondeterminism may occur only when the input word is of length $\leq k$.

Next, we show that all the words that derive a state are in the language of $|\mathscr{S}|$.

Proposition 4 *If $w \Rightarrow s$, for some $s \in Q$, then $w \in |\mathscr{S}|$.*

Proof: If $w = 1$, then $1 \Rightarrow s$ implies $s \in P$, showing that P is not empty. But then $1 \in R_s \subseteq |\mathscr{S}|$. Now assume that $y \Rightarrow r$ implies $y \in |\mathscr{S}|$, and consider ya, for some $a \in \Sigma$. If ya derives s, then $s \in \langle\langle r, a \rangle\rangle$, and there is an edge (r, a, s) in $\mathscr{S}$. Hence $ya \in |\mathscr{S}|$, and our claim follows. □

The next result deals with correctness and termination of maximal simulation.

Theorem 3 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a k-predictable semiautomaton, and $\mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$, its predictor. Given an input $w \in \Sigma^*$, let w' be the longest prefix of w that is in $|\mathscr{S}| = R_P$. Also, let $w' = yv$, where y is an arbitrary prefix of w'. Then*

1. *(1) The predictor operation is correct in the sense that $y \Rightarrow q$ in predictor $\mathscr{P}$ if and only if $q \in Py$ and $v \in R_q$.*
2. *(2) The simulation stops with the remaining input v if and only if $y \Rightarrow q$, for some $q \in Q$, and one of the following holds:*

(a) $v = 1$*; this implies that* $w \in |\mathscr{S}|$.
(b) $v = az$*, for some* $a \in \Sigma$, $z \in \Sigma^*$ *and there is no fork* $\langle q,a\rangle$ *in* $\mathscr{S}$*; this implies that* $w \notin |\mathscr{S}|$.

Proof: First, we show that, if $y \Rightarrow q$, then $q \in Py$ and $v \in R_q$. We proceed by induction on the length of the prefix y. If $1 \Rightarrow s$, then $s \in P$ by Definition 7 (1). Thus $s \in P1 = P$. Since w' is in R_P by assumption, and the key of w' in P applies to s, we have $w' \in R_s$, by Lemma 2 (2). Therefore the claim holds for the basis. Now assume that, for an arbitrary prefix y of $w' = yaz$, if $y \Rightarrow r$, then $r \in Py$ and $v \in R_r$. Suppose that $ya \Rightarrow s$. Then $y \Rightarrow r$, for some $r \in Q$, $s \in \langle\langle r,a\rangle\rangle$, and the key x of z in $\langle\langle r,a\rangle\rangle$ applies to s. By the induction hypothesis, $r \in Py$ and $az \in R_r$. Since $s \in \langle\langle r,a\rangle\rangle$, there is an edge $(r,a,s) \in E$; hence $s \in Pya$. Since $z \in R_{\langle\langle r,a\rangle\rangle}$ because $az \in R_r$, and the key of z in $\langle\langle r,a\rangle\rangle$ applies to s, by Lemma 2 (2) we have $z \in R_s$. Thus the induction goes through, and the claim holds.

Second, assume that $q \in Py$ and $v \in R_q$; we show that $y \Rightarrow q$. Again, we proceed by induction on the length of the prefix. Consider first the factorization $w' = yv = 1w'$. By Lemma 2 (2), if $p \in P$ and $w' \in R_p$, then the key of w' in P applies to p. By Definition 7 (1), the empty prefix 1 of w' derives p. Now assume that, for an arbitrary prefix y of $w' = yv$, if $r \in Py$ and $v \in R_r$, then $y \Rightarrow r$. Suppose now that $w' = yaz$. Since $w' \in R_P$, there exists $r \in Py$ such that $az \in R_r$, and hence a fork $\langle r,a\rangle$. Let s be any state in $T = \langle\langle r,a\rangle\rangle$ such that $z \in R_s$. By the induction hypothesis, $y \Rightarrow r$. Since $s \in T$ and $z \in R_s$, the key of z in T applies to s by Lemma 2 (2). Therefore $ya \Rightarrow s$, and the claim holds.

Now consider termination. Since 1 is a prefix of all words, the input word always has a key in P, and Step 0 of Definition 7 is always executed. Consequently, the derivation can stop only if $w = yv$ and some state q has been derived by y.

If $v = 1$, then v does not begin with a letter, the induction step cannot be carried out, and the derivation stops. Here $w = y$ is clearly in $|\mathscr{S}|$, since $q \in Py$ by Theorem 3 (1), and that implies $y \in R_P = |\mathscr{S}|$.

If $v = az$, for some $a \in \Sigma$, $z \in \Sigma^*$, and there is a fork $\langle q,a\rangle$ in $\mathscr{S}$, the derivation continues, since z always has a key in $\langle\langle q,a\rangle\rangle$. Thus the derivation can stop only if $v = az$, but there is no fork $\langle q,a\rangle$ in $\mathscr{S}$. Clearly, $az \notin R_q$. Suppose now that $w = yaz \in |\mathscr{S}|$. Since $y \Rightarrow q$, we have $q \in Py$ and $az \in R_q$, by Theorem 3 (1). This is a contradiction, and $w \notin |\mathscr{S}|$. □

The predictor is optimal in the sense that there is no unnecessary nondeterminism, that is, no prefix y of $w' = yaz$ derives a state from which it is impossible to continue the derivation. Moreover, if $w' = yaz$, and $|z| \geq k$ in a k-predictable semiautomaton $\mathscr{S}$, the induction step of the predictor is deterministic, because z is guaranteed to have a prefix u which is a minimal selector, by Theorem 1. Thus nondeterminism occurs only for words of length less than or equal to k. As the next result shows, even that nondeterminism can be avoided, if one is interested only in determining whether $w \in |\mathscr{S}|$, rather than in finding all the states reached by w. Thus, for the membership problem, one can arbitrarily select any possible next state in any nondeterministic step, and always reach the same conclusion.

Corollary 2 *In a predictor $\mathscr{P}$, $w \in |\mathscr{S}|$ if and only if $w \Rightarrow q$, for some $q \in Q$.*

Proof: By Proposition 4, if $w \Rightarrow q$, then $w \in |\mathscr{S}|$. Conversely, if $w \in |\mathscr{S}|$, then $w' = w$; also there exists $q \in Pw$, by definition of $|\mathscr{S}|$. Since $1 \in R_q$, by Theorem 3, applied with $y = w'$ and $v = 1$, we have $w = w' \Rightarrow q$. □

Example 8 *Figure 7 without the minimal selectors and maximal nonselectors represents a semiautomaton $\mathscr{S}$. With the minimal selectors and maximal nonselectors added, it can be interpreted as the predictor as follows: The incoming edge of q_1 is labeled with minimal selectors $[a]$, $[ba]$, and $[bb]$, and that of q_6, with maximal nonselector $\lfloor b \rfloor$. The edge (q_1, a, q_2) is replaced by edges $(q_1, (a, [a]), q_2)$ and $(q_1, (a, [bb]), q_2)$, while the edge (q_1, a, q_3) is replaced by $(q_1, (a, [ba]), q_3)$. In the fork $\langle q_2, a \rangle$, edge (q_2, a, q_4) is replaced by $(q_2, (a, [a]), q_4)$, edge (q_2, a, q_5), by $(q_2, (a, \lfloor 1 \rfloor), q_5)$, and edge (q_2, a, q_6), by $(q_2, (a, [b]), q_6)$. All other edges are deterministic transitions and have minimal selectors* 1.

Suppose the input word is $w = aaababaab$. We use the notation $a(z)$ instead of az for a word in $a\Sigma^*$ to make it easier to identify the key of z. The following computation takes place, where q indicates the current state, and v, the remaining input: In Step 0, $v = w = aaababaab$, and the key of w in $\{q_1, q_6\}$ is a. We have $1 \Rightarrow q_1$, since a applies only to q_1. The next seven steps are deterministic:

(1) $q = q_1$, $v = a(aababaab)$, *the key of* $aababaab$ *in* $\{q_2, q_3\}$ *is* a.
$a \Rightarrow q_2$, *since* a *applies only to* q_2; a *is consumed from the input.*

(2) $q = q_2$, $v = a(ababaab)$, *the key of* $ababaab$ *in* $\{q_4, q_5, q_6\}$ *is* a.
$aa \Rightarrow q_4$, *since* a *applies only to* q_4; a *is consumed.*

(3) $q = q_4$, $v = a(babaab)$, *the key of* $babaab$ *in* $\{q_1\}$ *is* 1.
$aaa \Rightarrow q_1$, *since* 1 *applies to* q_1; a *is consumed.*

(4) $q = q_1$, $v = b(abaab)$, *the key of* $abaab$ *in* $\{q_1\}$ *is* 1.
$aaab \Rightarrow q_1$, *since* 1 *applies to* q_1; b *is consumed.*

(5) $q = q_1$, $v = a(baab)$, *the key of* $baab$ *in* $\{q_2, q_3\}$ *is* ba.
$aaaba \Rightarrow q_3$, *since* ba *applies only to* q_3; a *is consumed.*

(6) $q = q_3$, $v = b(aab)$, *the key of* aab *in* $\{q_7\}$ *is* 1.
$aaabab \Rightarrow q_7$, *since* 1 *applies to* q_7; b *is consumed.*

(7) $q = q_7$, $v = a(ab)$, *the key of* ab *in* $\{q_1\}$ *is* 1.
$aaababa \Rightarrow q_1$, *since* 1 *applies to* q_1; a *is consumed.*

In the next step, there are two possibilities:

(8a) $q = q_1$, $v = a(b)$, *the key of* b *in* $\{q_2, q_3\}$ *is* b.
$aaababaa \Rightarrow q_2$, *since* b *applies to* q_2; a *is consumed. Go to 9a.*

(8b) $q = q_1$, $v = a(b)$, the key of b in $\{q_2, q_3\}$ is b.
aaababaa $\Rightarrow q_3$, since b applies to q_3; a is consumed. Go to 9b.

(9a) $q = q_2$, $v = b(1)$, the key of 1 in $\{q_6\}$ is 1.
aaababaab $\Rightarrow q_6$, since 1 applies to q_6; b is consumed. Go to 10.

(9b) $q = q_3$, $v = b(1)$, the key of 1 in $\{q_7\}$ is 1.
aaababaab $\Rightarrow q_6$, since 1 applies to q_7; b is consumed. Go to 10.

(10) $q \in \{q_6, q_7\}$, $v = 1$. The input word $v = 1$ no longer satisfies the condition that $w = yaz$ in the induction step, and the computation stops. The set of states derived by $w =$ aaababaab is $\{q_6, q_7\}$.

In view of Corollary 2, if we were interested only in deciding the membership of w, we could pick either Step 8a followed by 9a, or Step 8b followed by 9b. In either case, the derivation terminates when the remaining input is the empty word, showing acceptance of w by $\mathscr{S}$.

In the following discussion, let c represent all the letters that might appear on the input tape, but for which our semiautomaton has no edges. In general, the predictor $\mathscr{P}$ of Fig. 7 has the following properties;

- *If $w \in \{1, b\}$ or w begins with c or bc, then $1 \Rightarrow q_1$ and $1 \Rightarrow q_6$, that is, the initial state set is $\{q_1, q_6\}$.*
- *If w begins with a, ba or bb, then $1 \Rightarrow q_1$, and the initial state is q_1.*

In the fork $\langle q_1, a\rangle$, we have the set $T = \langle\langle q_1, a\rangle\rangle = \{q_2, q_2\}$. If we treat T as the initial state set of $\mathscr{S}$, then

- *If $w \in \{1, b\}$ or w begins with c or bc, then $1 \Rightarrow q_2$, and $1 \Rightarrow q_3$; hence the set of states derived by 1 is $\{q_2, q_3\}$. Consequently, we have to consider both q_2 and q_3 as possible successors of q_1 under input a.*
- *If w begins with a or bb, then we only have $1 \Rightarrow q_2$. Thus q_2 is the only successor of q_1 under a.*
- *If w begins with ba, then we only have $1 \Rightarrow q_3$. Thus q_3 is the only successor of q_1 under a.*

In the fork $\langle q_2, a\rangle$, we have the set $T = \langle\langle q_2, a\rangle\rangle = \{q_4, q_5, q_6\}$. If we treat T as the initial state set of $\mathscr{S}$, then

- *If $w = 1$ or w begins with c, then $1 \Rightarrow q_4$, $1 \Rightarrow q_5$, and $1 \Rightarrow q_6$. Thus all three states are possible successors of q_2 under a.*
- *If w begins with a, then $1 \Rightarrow q_4$. Thus q_4 is the only successor of q_2 under a.*
- *If w begins with b, then $1 \Rightarrow q_6$. Thus q_6 is the only successor of q_2 under a.*

7.3 Minimal Simulation

The mandate of maximal simulation was to exhibit the longest computation of the semiautomaton, regardless of the ultimate acceptance or rejection of the input word. In contrast to this, the second simulation decides as soon as possible whether the input is accepted or rejected; for this and other reasons it is more efficient than maximal simulation.

Definition 8 *In a predictor $\mathscr{P}$, for a word $w \in \Sigma^*$ and $T \subseteq Q$, we define the* handle *of w in T as follows. If a minimal selector x in T is a prefix of w, then x is the handle. If w is a prefix of a minimal selector or a maximal nonselector in T, then w itself is the handle. Otherwise, w does not have a handle. If w has a handle in T, the handle* applies *to a state $t \in T$ if either the handle is a minimal t-selector, or it is a prefix of a minimal t-selector or of a maximal t-nonselector.*

Remark 4 *In contrast to the computation of keys in maximal simulation, finding a handle does not involve looking for common prefixes, since a handle is either the input word or a minimal selector. Note also that a word can have at most one handle in a set T.*

We now define minimal simulation by a predictor.

Definition 9 *Given a predictor $\mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$ of a k-predictable semiautomaton $\mathscr{S} = (\Sigma, Q, P, E)$ and an input word w, a prefix y of w* yields *a state $s \in Q$, written $y \rightarrow s$ as follows:*

(1) Basis Step (Step 0):
$1 \rightarrow s$ if $s \in P$ and the handle of w in P applies to s.
(2) Induction Step (Step $m+1$, $m \geq 0$):
Assume now that $w = yaz$, for some $a \in \Sigma$, $y, z \in \Sigma^*$. Then $ya \rightarrow s$ if $y \rightarrow r$, for some $r \in Q$, $s \in \langle\langle r, a \rangle\rangle$ and the handle of z in $\langle\langle r, a \rangle\rangle$ applies to s.

Proposition 5 *If $w \rightarrow s$ for some $s \in Q$, then $w \in |\mathscr{S}|$.*

Proof: The proof is parallel to that of Proposition 4. □

Remark 5 *If w has a handle in T, then that handle is also a key of w in T. Thus, $w \rightarrow q$ implies $w \Rightarrow q$, but the converse is false in general. Using handles may shorten the time for the membership decision, since the absence of a handle stops a minimal derivation before maximal derivation reaches the longest prefix in $|\mathscr{S}|$.*

Lemma 3 *Let $\mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$ be the predictor of a k-predictable semiautomaton $\mathscr{S}$, and $w = yv \in |\mathscr{S}|$. Then $y \Rightarrow q$ if and only if $y \rightarrow q$.*

Proof: If $y \to q$, then $y \Rightarrow q$ by Remark 5. For the converse, we proceed by induction on the length of a prefix of w.

If $y = 1$ and $y \Rightarrow q$, then, by definition, $q \in P$ and the key of w in P applies to q. By Lemma 2 (1), the key of w in P is either a minimal selector or w itself, since w is the longest prefix of w which is in $R_P = |\mathscr{S}|$. If the key is a minimal selector, then it is also a handle of w in P, by definition. If the key is w itself, then, by the definition of a key, w is a prefix of a minimal selector or of a maximal nonselector in P. In either case, w is a handle as well, by definition. Thus, $y \to q$, since the handle of w in P applies to q.

Assume now that $w = yaz$ and the implication holds for y. We prove that, if $ya \Rightarrow q$, then $ya \to q$. Let r be a state such that $q \in \langle\langle r,a \rangle\rangle = T$ and $y \Rightarrow r$. Then the key of z in T applies to q. Since $ya \Rightarrow q$, we have $z \in R_T$ by Theorem 3. By Lemma 2 (1), this key is either z or a minimal selector; in either case, the key is also the handle of z in T, which applies to q. By definition, $ya \to q$, and the induction is complete. $\square$

Theorem 4 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a k-predictable semiautomaton, $\mathscr{P} = (\Sigma \times \Sigma^{\leq k}, Q, P, E_{\mathscr{P}})$, its predictor, and $w = yv \in \Sigma^*$, an input word of $\mathscr{S}$. Then the predictor operation is correct in the following sense:*

(1) *If $w \in |\mathscr{S}|$, then $y \to q$ in $\mathscr{P}$ if and only if $q \in Py$ and $v \in R_q$ in $\mathscr{S}$.*
(2) *The simulation stops with the remaining input v if and only if one of the following holds:*
  (a) *It is Step 0 and w has no handle in P; this implies $w \notin |\mathscr{S}|$.*
  (b) *It is a Step > 0 and $v = 1$; this implies that $w \in |\mathscr{S}|$.*
  (c) *It is a Step > 0 and $v = az$, for some $a \in \Sigma$, $z \in \Sigma^*$ and there is no fork $\langle q,a \rangle$ in $\mathscr{S}$; this implies that $w \notin |\mathscr{S}|$.*
  (d) *It is a Step > 0 and $v = az$, for some $a \in \Sigma$, $z \in \Sigma^*$ there is a fork $\langle q,a \rangle$ in $\mathscr{S}$, but z has no handle in $\langle\langle q,a \rangle\rangle$; this implies $w \notin |\mathscr{S}|$.*

Proof: If $w \in |\mathscr{S}|$ then $y \to q$ if and only if $y \Rightarrow q$, by Lemma 3. By Theorem 3, $y \Rightarrow q$ if and only if $q \in Py$ and $v \in R_q$. Hence (1) holds.

For (a), if w has no handle in P, then $1 \to p$ is false for all $p \in P$, by Definition 9, and the simulation stops. Now if $w \in |\mathscr{S}|$, then $w \in R_P$; hence $w \in R_p$, for some $p \in P$. Since also $p \in P1$, Part (1) of the theorem applies, and $1 \to p$, which is a contradiction, and $w \notin |\mathscr{S}|$.

For (b), if the minimal simulation has consumed y, $y \to q$, and $v = 1$, then $w = y$, and the entire input has been processed. Since v does not have the form az, the simulation stops. Since $w = y \to q$, we have $w \in |\mathscr{S}|$, by Proposition 5.

For (c), assume that $w = yv = yaz$, the minimal simulation has consumed y, $y \to q$, but there is no fork $\langle q,a \rangle$ in $\mathscr{S}$. Clearly, the induction step cannot be carried out, and $az \notin R_q$. Suppose now that $w \in |\mathscr{S}|$. Since $y \to q$, we have $q \in Py$ and $az \in R_q$,

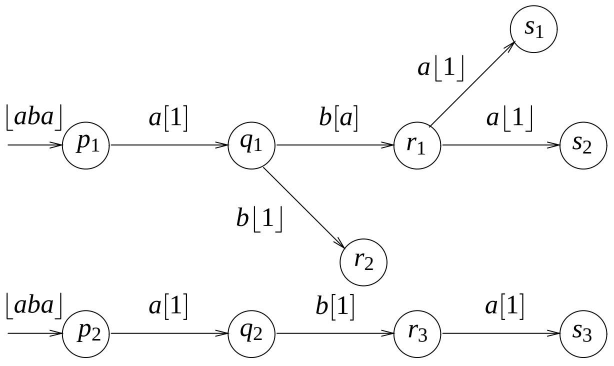

Fig. 9. Illustrating maximal and minimal simulations.

by Theorem 4 (1). This is a contradiction, and $w \notin |\mathscr{S}|$.

For (d), assume that $w = yv = yaz$, the minimal simulation has consumed y, $y \to q$, and z has no handle in $\langle\langle q,a \rangle\rangle$. Then the simulation stops because the induction step cannot be carried out. If $w \in |\mathscr{S}|$, since $y \to q$, we know by Part (1) of the theorem that $q \in Py$ and $v \in R_q$. Also, there exists $r \in \langle\langle q,a \rangle\rangle$ such that (q,a,r) is an edge in $\mathscr{S}$ and $z \in R_r$. But now $r \in Pya$ and $z \in R_r$ implies that $ya \to r$, again by Part (1). Thus z must have a handle in $\langle\langle q,a \rangle\rangle$, which is a contradiction, and so $w \notin |\mathscr{S}|$.

One verifies that, if none of the conditions (a)–(d) holds, then the derivation continues. This concludes the proof of the second claim. $\square$

Corollary 3 *In a predictor $\mathscr{P}$, $w \in |\mathscr{S}|$ if and only if $w \to q$, for some $q \in Q$.*

Proof: By Proposition 5, $w \to q$ implies $w \in |\mathscr{S}|$. Conversely, if $w \in |\mathscr{S}|$, then there exists $q \in Pw$, by definition of $|\mathscr{S}|$. Since also $1 \in R_q$, by Theorem 4 (1) applied with $y = w$ and $v = 1$, we have $w \to q$. $\square$

Example 9 *Consider the semiautomaton of Fig. 9, which is* 4*-predictable. The input word $w = abab$ has no prefix which is a minimal selector in P, and w is not a prefix of a minimal selector or of a maximal nonselector in P. Hence minimal simulation immediately yields the empty set of states, which is equivalent to rejecting w. This is correct, since $w \notin R_P$. In contrast to this, maximal simulation has the following paths corresponding to derivations:*

(1) $p_1 \xrightarrow{a} q_1 \xrightarrow{b} r_1 \xrightarrow{a} s_1$
(2) $p_1 \xrightarrow{a} q_1 \xrightarrow{b} r_1 \xrightarrow{a} s_2$
(3) $p_2 \xrightarrow{a} q_2 \xrightarrow{b} r_3 \xrightarrow{a} s_3$

It stops after consuming aba, because there is no fork of the form $\langle s_i, b \rangle$, for any $i \in \{1,2,3\}$. Thus, for $w = abab$, maximal simulation derives the empty set of states, rejecting w as well.

On the other hand, for $w = aba$, both simulations have the same three deriva-

tions/yields corresponding to the paths above.

Example 10 *In the semiautomaton of Fig. 7, for $w = abba$, both simulations have only one derivation/yield corresponding to the path:*

$$q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_6 \xrightarrow{b} q_5.$$

Then both stop. Look-ahead does not provide enough information to stop the minimal simulation earlier. The handle of abba in $\{q_1, q_2\}$ is a, yielding q_1 as the initial state. The handle of bba in $\{q_2, q_3\}$ is bb, yielding q_2. The handle of ba in $\{q_6\}$ is 1, which yields q_6. The handle of a in $\{q_5\}$ is 1, which yields q_5. Since there is no fork $\langle q_6, a\rangle$, and the next input letter is a, minimal simulation stops.

We have seen in Lemma 3 that minimal simulation of a semiautomaton $\mathscr{S}$ has the same behavior as the maximal one on words in $|\mathscr{S}|$. The observation can be extended, as follows:

Remark 6 *If minimal and maximal simulations run in parallel on the same input word, then the keys and handles coincide on the run of the minimal simulation.*

This observation leads to the idea of an *optimal simulation* of $\mathscr{S}$, which combines minimal and maximal simulations. Start a minimal simulation and let it run as long as it finds handles. If the entire input is consumed during this simulation, then the input has been accepted, and a successful computations of $\mathscr{S}$ has been identified. Otherwise, when minimal simulation stops because the input word has no handle, maximal simulation takes over. In this case, we know that the input is not accepted; however, minimal simulation keeps running as far as the longest prefix of the input word that is in $\mathscr{S}$. Thus, optimal simulation takes advantage of both the efficiency of minimal simulation and the information provided by maximal simulation.

Both maximal and minimal simulations operate "almost deterministically" in finding next states, that is, determinism is guaranteed as long as the look-ahead buffer is full (the remaining input has length at least k). However, by Corollary 3, we can achieve total determinism concerning acceptance. Run minimal simulation until more than one choice appears, and then choose an arbitrary branch in every step. Then minimal simulation is completely deterministic.

8 Predictability Bounds

We now derive bounds on the size of the look-ahead buffer in terms of the number of states in the semiautomaton. We first consider the case of a one-letter alphabet.

Proposition 6 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be a semiautomaton over a one-letter alphabet. If $\mathscr{S}$ has n states, and $k \geq 0$ is the smallest integer for which $\mathscr{S}$ is k-predictable,* *then $k \leq n-1$.*

Proof: If $k = 0$, the bound is trivially satisfied. Hence assume that there is at least one critical set $T = \{t_1, \dots t_h\}$, $h \geq 2$, which is k-predictable.

We claim that T is predictable if and only if at most one of the languages $\{R_{t_i}\}_{1 \leq i \leq h}$ is infinite. Note first that, if a language L over one letter a is prefix-closed, then L is infinite if and only if $L = a^*$. For $1 \leq i \neq j \leq h$, if R_{t_i} and R_{t_j} are infinite then $R_{t_i} \cap R_{t_j} = a^*$, since R_{t_i} and R_{t_j} are prefix-closed. Hence T is not predictable by Theorem 1, and R_{t_i} and R_{t_j} cannot both be infinite.

Without loss of generality, assume now that $R_{t_1}, \dots, R_{t_{h-1}}$ are finite. We distinguish two cases:

(1) R_{t_h} is infinite. Let w be a longest word in $\bigcup_{1 \leq i < h} R_{t_i}$, and assume that $w \in R_{t_j}$, $j \neq h$. Since R_{t_j} is finite, no path originating in t_j and spelling w can have a state repeated. For suppose that $w = uxv$, for some $x \in \Sigma^+, u, v \in \Sigma^*$, and $t_j u = t_j ux$. Then also $ux^2v \in R_{t_j}$, contradicting that w is a longest word of R_{t_j}. We also observe that a path π from t_j spelling w cannot visit t_h, otherwise R_{t_j} would be infinite, since R_{t_h} is infinite. Thus π has at most $n-1$ states, and $|w| \leq n-2$. Now T cannot be $|w|$-predictable, because $w \in R_{t_j} \cap R_{t_h}$, but it is $(|w|+1)$-predictable. Thus we must have $k = |w| + 1 \leq n-1$.

(2) R_{t_h} is finite. Let w be a longest word in $\bigcup_{1 \leq i \leq h} R_{t_i}$, and assume $w \in R_{t_j}$. As above, a path originating in t_j and spelling w can involve at most n states; thus $|w| \leq n-1$. If $|w| < n-1$ then clearly $k \leq |w| + 1 \leq n-1$. When $|w| = n-1$, a path π originating in t_j and spelling w uses all the states of $\mathscr{S}$. There cannot be another path originating in t_i, $i \neq j$, spelling w; for then there would be a loop, contradicting the finiteness of R_{t_j}. Thus, $k = |w| = n-1$ in this case.

The semiautomaton in Fig. 10 has n states and is $(n-1)$-predictable; thus the bound can be reached when $|\mathscr{S}|$ is infinite. If we remove the loop in Fig. 10 and make states 1 and 2 initial, we reach the bound when $|\mathscr{S}|$ is finite. □

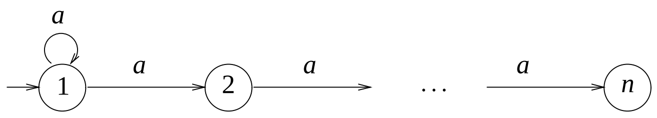

Fig. 10. A n-state unary semiautomaton which is $(n-1)$-predictable.

Before addressing the case of a general alphabet, we develop some necessary conditions on paths originating from critical sets in predictable semiautomata.

Lemma 4 *Let $\mathscr{S} = (\Sigma, Q, P, E)$ be predictable, let $Q = \{1, \dots, n\}$, and let r_1, s_1 be distinct states of a critical set in $\mathscr{S}$. If $w = a_1 \dots a_{m-1}$ is a longest word in $R_{r_1} \cap R_{s_1}$, let $\pi_1 = (r_1, a_1, r_2) \dots (r_{m-1}, a_{m-1}, r_m)$ and $\pi_2 = (s_1, a_1, s_2) \dots (s_{m-1}, a_{m-1}, s_m)$ be two paths spelling w, originating from r_1 and s_1, respectively. Then the sequence*

$\mathscr{L} = (r_1, s_1), \ldots, (r_m, s_m)$ of ordered pairs of states encountered by π_1 and π_2 must satisfy the following conditions: First, $r_1 \neq s_1$, and, for all $i, j \in \{1, \ldots, m\}, i \neq j$,

(1) either $r_i \neq r_j$ or $s_i \neq s_j$,
(2) either $r_i \neq s_j$ or $r_j \neq s_i$,
(3) if $r_i = s_i$, then $r_j \neq r_i$ and $s_j \neq r_i$.

Proof: By our hypothesis, $r_1 \neq s_1$. For the remaining conditions we have:

(1) If there exist i, j, $1 \leq i < j \leq t$, with $r_i = r_j$ and $s_i = s_j$, let x be the label of the path $r_i, \ldots, r_j$ in π_1. Then $w = uxv$ for some $u, v \in \Sigma^*$, and $ux^2v \in R_{r_1} \cap R_{s_1}$, contradicting the maximality of $|w|$.

(2) If there exist i, j, $1 \leq i < j \leq t$, such that $r_i = s_j$ and $r_j = s_i$, let x be the label of the path $r_i, \ldots, r_j$ in π_1, and hence also the label of the path $s_i, \ldots, s_j$ in π_2. Then one verifies that $ux^3v \in R_{r_1} \cap R_{s_1}$, contradicting the maximality of $|w|$.

(3) If there exist i, j, $1 \leq i < j \leq t$, such that $r_i = s_i$ and $r_j = r_i$, let x be the label of the path $r_i, \ldots, r_j = r_i$ in π_1, and let $w = uxv$. Then $ux^2v \in R_{r_1} \cap R_{s_1}$, contradicting the maximality of $|w|$. Similarly, if $s_j = r_i$, let x be the label of the path $s_i, \ldots, s_j$ in π_2, and let $w = uxv$. Since $s_i = s_j = r_i$, there is a loop labeled x on r_i and again $ux^2v \in R_{r_1} \cap R_{s_1}$. □

Next we prove a combinatorial result about sequences $\mathscr{L} = (r_1, s_1), \ldots, (r_m, s_m)$ satisfying the conditions of Lemma 4.

Lemma 5 *Let $n > 0$ be an integer and let $\mathscr{L} = (r_1, s_1), \ldots, (r_m, s_m)$ be a sequence of ordered pairs of elements from $\{1, \ldots, n\}$. If $\mathscr{L}$ satisfies the conditions of Lemma 4, then $m \leq (n^2 - n)/2$ and the bound is sharp.*

Proof: We first show that the bound can be achieved. Consider the sequence

$$\mathscr{L} = (1,1), \ldots, (1,n), (2,1), \ldots, (2,n), \ldots, (n,1), \ldots, (n,n),$$

which has n^2 elements and satisfies Condition (1). If we remove the pairs (i,i), for all $1 \leq i \leq n$, we have a sequence of $n^2 - n$ pairs, in which $r_1 \neq s_1$ and which satisfies Condition (3) as well, since r_i is never equal to s_i. Finally, for all $i \neq j$, remove either (i, j) or (j, i). Now the sequence also satisfies Condition (2). Since there are $(n^2 - n)/2$ pairs removed in the last step, the final sequence has $(n^2 - n)/2$ elements. Thus the bound can be reached.

Next, we prove that $(n^2 - n)/2$ is an upper bound by induction on n. If $n = 1$, then only the empty sequence satisfies all the conditions. Hence $m = 0 = (n^2 - n)/2$. If $n = 2$, then the empty sequence, $(1,2)$ and $(2,1)$ are the only sequences satisfying the conditions. Here $m \leq 1 = (n^2 - n)/2$.

For any $n > 0$, let $M(n)$ be the length of a longest sequence of pairs of elements from $\{1,\ldots,n\}$ satisfying all the conditions. Assume that $M(n-1) \leq [(n-1)^2-(n-1)]/2$, for some $(n-1) \geq 2$. Let $\mathscr{L}$ be a sequence with $M(n)$ pairs of elements from $\{1,\ldots,n\}$ satisfying all the conditions, and assume for the sake of contradiction that $M(n) > (n^2-n)/2$.

If $M(n) > (n^2-n)/2$ and $n \geq 3$, then $M(n) > n$. Thus $\mathscr{L}$ contains at least $n+1$ pairs. There are at most $2n-1$ pairs involving the element n, namely the pairs of the form (n,i) and (i,n). However, if both (n,i) and (i,n) appear in $\mathscr{L}$, then Condition (2) is violated. Hence there are at most n pairs involving n, and at least one pair (i,j) not involving n. Without loss of generality we may assume that the first pair of $\mathscr{L}$ does not contain n, for if it did, we could interchange it with (i,j).

Let $\mathscr{L}'$ be the sequence with m' elements obtained from $\mathscr{L}$ by removing all the pairs containing n. Then $\mathscr{L}'$ satisfies all the conditions as well, and its elements are from the set $\{1,\ldots,n-1\}$. By the induction hypothesis, $m' \leq M(n-1) \leq [(n-1)^2-(n-1)]/2 = (n^2-n)/2-(n-1)$.

In addition to the elements of $\mathscr{L}'$, $\mathscr{L}$ contains elements from the set

$$\{(1,n),(2,n),\ldots,(n,n),(n,1),(n,2),\ldots,(n,n-1)\}.$$

If $\mathscr{L}$ contains (n,n), then it cannot contain any other pair involving n, for this would violate Condition (3). Hence, in this case, we have $M(n) = m'+1 \leq (n^2-n)/2-(n-2) < (n^2-n)/2$, which contradicts our assumption.

If $\mathscr{L}$ does not contain (n,n), it contains at most $(n-1)$ pairs involving n. In this case $M(n) \leq m'+(n-1) \leq (n^2-n)/2$, which is again a contradiction.

Consequently, $M(n) \leq (n^2-n)/2$ and the induction step goes through. $\square$

Theorem 5 *If a semiautomaton $\mathscr{S} = (\Sigma,Q,P,E)$ has n states and k is the smallest integer for which $\mathscr{S}$ is k-predictable then $k \leq (n^2-n)/2$. Moreover, this bound can always be reached for a suitable Σ.*

Proof: For $k = 0$, $\mathscr{S}$ is deterministic and the inequality is trivially satisfied. If $k \geq 1$ then $\mathscr{S}$ has at least one k-predictable critical set T. Let r_1, s_1 be two distinct states of T. By Lemma 4, every sequence $\mathscr{L} = (r_1,s_1),\ldots,(r_m,s_m)$ must satisfy all the conditions of the lemma, and by Lemma 5, the length of a longest word $w \in R_{r_1} \cap R_{s_1}$ is $|w| = m-1 \leq (n^2-n)/2-1$. This implies that, if T is k-predictable, then necessarily $k \leq |w|+1 \leq (n^2-n)/2$. Since this holds for all critical sets, it holds for $\mathscr{S}$.

Next we prove that this bound is achievable for a suitable alphabet. Let $n \geq 1$, and let $\mathscr{L} = (r_1,s_1),\ldots,(r_k,s_k)$ be a sequence satisfying the conditions of Lemma 5, with $k = (n^2-n)/2$. Let $\mathscr{S} = (\Sigma,Q,P,E)$, where $\Sigma = \{a_1,\ldots,a_k\}$, $Q = \{1,\ldots,n\}$,

$P = \{r_1\}$, $E = E_1 \cup E_r \cup E_s$, $E_1 = \{(r_1, a_1, r_1), (r_1, a_1, s_1)\}$, $E_r = \{(r_i, a_{i+1}, r_{i+1}) \mid 1 \leq i < k\}$, and $E_s = \{(s_i, a_{i+1}, s_{i+1}) \mid 1 \leq i < k\}$. We show that $\mathscr{S}$ is k-predictable, but not $(k-1)$-predictable.

Observe that $E_1 = \langle r_1, a_1 \rangle$ is a fork in $\mathscr{S}$ and its fork set $\langle\langle r_1, a_1 \rangle\rangle = \{r_1, s_1\}$ is a critical set in $\mathscr{S}$. Consider the word $w = a_2 \ldots a_k$; clearly, $w \in R_{r_1} \cap R_{s_1}$, implying that the automaton is not $(k-1)$-predictable, since $|w| = k-1$. Observe now that $R_{r_k} \cap R_{s_k} = \emptyset$; for if both (r_k, a_j, r) and (s_k, a_j, s) were in E for some $r, s \in Q$ and $j \in \{1, \ldots, m\}$, then either $(r_k, s_k) = (r_{j-1}, s_{j-1})$ for $j > 1$, or $(r_k, s_k) = (r_1, s_1)$ for $j = 1$. In both cases, this would violate Condition (1) of Lemma 4. Thus w is a longest word in $R_{r_1} \cap R_{s_1}$, implying that $\langle\langle r_1, a_1 \rangle\rangle$ is k-predictable.

Since $P\# = 1$, P is 0-predictable. If there exists a fork other than $\langle r_1, a_1 \rangle$ in $\mathscr{S}$, then there must be a pair (r_i, s_i) with $r_i = s_i$, since only such states have outgoing edges with a same label, by the construction of $\mathscr{S}$. But then, by an argument used in the proof of Lemma 5, the sequence $\mathscr{L}$ cannot be of maximal length. Hence there are no other forks in $\mathscr{S}$, and $\mathscr{S}$ is k-predictable, where $k = (n^2 - n)/2$ is the smallest such integer.

In summary, the minimal predictability bound of $(n^2 - n)/2$ can be reached at the cost of a large alphabet. □

The theorem implies that, to test whether $\mathscr{S}$ is predictable, it suffices to test whether it is $(n^2 - n)/2$-predictable.

Example 11 *For $n = 4$, the semiautomata in Fig. 11(a) and 11(b) correspond to the sequences*

$$\mathscr{L}_1 = \{(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)\}$$

and

$$\mathscr{L}_2 = \{(2,3), (3,4), (1,4), (2,4), (1,3), (2,1)\},$$

respectively. Both sequences obey the conditions of Lemma 5 and are of maximal length. Therefore, both semiautomata are 6-predictable, and reach the upper-bound for 4 states.

Remark 7 *For fixed alphabets, the bounds may turn out to be much smaller. The semiautomaton in Fig. 12 over a two-letter alphabet has n states and is k-predictable, where*

$$k = \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n+1}{2} \right\rfloor.$$

This bound may be tight for binary alphabets. In this particular case, predictability is maximized precisely for $i = \lfloor n/2 \rfloor$ and $j = \lfloor (n+1)/2 \rfloor$ or vice versa, since the product ij, with $i + j = n$, is maximized for these values only.

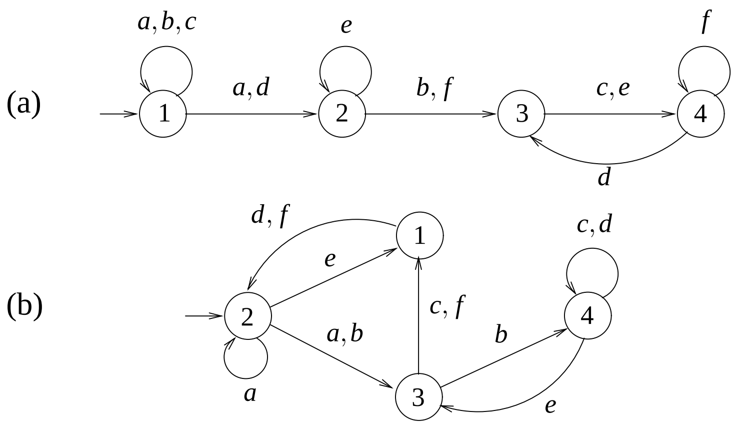

Fig. 11. Semiautomata reaching the predictability bound.

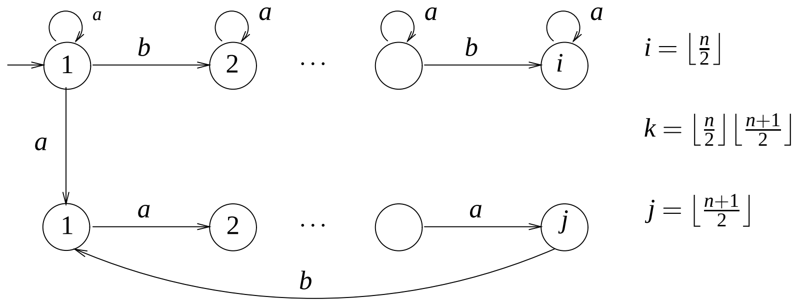
a
b
b
$i = \lfloor \frac{n}{2} \rfloor$
$k = \lfloor \frac{n}{2} \rfloor \lfloor \frac{n+1}{2} \rfloor$
$j = \lfloor \frac{n+1}{2} \rfloor$
i
j

Fig. 12. A *k*-predictable semiautomaton over a two-letter alphabet.

9 Conclusions

We have introduced the class of predictable semiautomata, which have a rich mathematical structure. In such semiautomata, it is possible to remove most of the nondeterminism by using a finite amount of look-ahead information from the input tape.

We have presented an algorithm for testing for predictability using a new construct: the core semiautomaton. To reduce nondeterminism, we have used predictor semiautomata with look-ahead buffers. In the worst case, the minimal length k of a predictor's look-ahead buffer is quadratic in the number n of states of the semiautomaton.

We have considered two objectives of simulation of a nondeterministic semiautomaton: finding the set of states reachable by an input word in a semiautomaton, and testing whether the input word belongs to the language of the semiautomaton. We have also introduced two types of simulation, maximal and minimal. In both simulations, as long as the input word has length greater than k, the computation is deterministic. For testing for membership, both simulations can be completely deterministic. In finding the set of states reachable by a word, some nondeterminism is unavoidable, but it is limited to the last k letters of the input word. Moreover, if the remaining input word is not in the language, minimal simulation stops the computation before reaching this residual nondeterminism.

The application of this theory to nondeterministic automata (semiautomata with accepting states) is straightforward. To determine whether a word *w* is accepted by an automaton, find the set of states reached by *w* and check whether any of these states are accepting. The look-ahead information ensures that these computations are fewer than in the standard NFA simulation, since many computation paths are pruned, having been detected as unsuccessful by the look-ahead information.

References

[1] D. Berardi, D. Calvanese, G. D. Giacomo, M. Lenzerini, M., Mecella, Automatic composition of e-services that export their behavior. In E. Orłowska, M. Papazoglou, S. Weerawarana, J. Yang, eds., ICSOC 2003, *LNCS* **2910** (2003) 43–58.

[2] Z. Dang, O. H. Ibarra, J. Su, Composability of infinite-state activity automata, ISAAC 2004, R. Fleischer, G. Trippen, eds., *LNCS* **3341** (2004) 377–388.

[3] A. Ginzburg, *Algebraic Theory of Automata,* Academic Press, New York (1968).

[4] S. Eilenberg, *Automata, Languages, and Machines* **A,** Academic Press, New York (1974).

[5] B. Ravikumar, N. Santean, *Deterministic simulation of NFA with k-symbol lookahead, Int. J. of Foundations of Computer Science*, to appear.