On synchronized multi-tape and multi-head automata

Oscar H. Ibarra a,∗, Nicholas Q. Tran b

a Department of Computer Science, University of California, Santa Barbara, CA 93106, USA
b Department of Mathematics & Computer Science, Santa Clara University, Santa Clara, CA 95053, USA

Abstract

Motivated by applications to verification problems in string manipulating programs, we look at the problem of whether the heads in a multi-tape automaton are synchronized. Given an n-tape pushdown automaton M with a one-way read-only head per tape and a right end marker $ on each tape, and an integer $k ≥ 0$, we say that $M$ is $k$-synchronized if at any time during any computation of $M$ on any input n-tuple $(x_1, . . . , x_n)$ (whether or not it is accepted), no pair of input heads that are not on $ are more than $k$ cells apart. This requirement is automatically satisfied if one of the heads has reached $. Notethatan an n-tuple $(x_1, . . . , x_n)$ is accepted if $M$ reaches the configuration where all n heads are on $ and $M$ is in an accepting state. The automaton can be deterministic (DPDA) or nondeterministic (NPDA) and, in the special case, may not have a pushdown stack (DFA, NFA). We obtain decidability and undecidability results for these devices for both one-way and two-way versions. We also consider the notion of $k$-synchronized one-way and two-way multi-head automata and investigate similar problems.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

A serious Web security vulnerability can occur when a user input string is embedded into an interpreted script which is then executed with system privileges. Using special symbols in the scripting language (e.g., the comment symbol), it is possible to craft an embedded input string to alter the intended meaning of the script and bypass security checks, such as in the case of SQL injection attacks. A current research effort to combat this problem calls for reachability analysis of values stored in string variables during the execution of a script; a script is considered secure if the analysis shows that it satisfies certain properties, e.g., a particular string variable will never contain a specific set of characters. Earlier reachability analyses model each string variable using a separate finite-state automaton [1,7]. More recently, multi-track and then multi-tape finite-state automata have been used to model sets of string variables, allowing assertions about relationships between the string members [8–10].

Although multi-tape automata support a richer class of assertions and are easier to program than multi-track automata, decision problems involving them are often undecidable. (Here we are assuming the usual convention that strings in an input tuple to multi-track automata are left-justified, and the shorter strings are padded to the right with the symbol λ to have the same length as the longest string; on the other hand, if λ’s are allowed to appear anywhere in the strings, then multi-track and multi-tape automata are equivalent.) Naturally we propose to study those multi-tape automata that have equivalent multi-track counterparts, since they are versatile and yet have decidable properties. One such subclass of multi-tape automata can be identified: multi-tape automata can be easily converted to equivalent multi-track automata if the distance between any two tape heads can be bounded by a constant k during any computation. We say such machines are $k$-synchronized or simply synchronized if the constant is not important or not known.

∗ Corresponding author. Tel.: +1 805 893 4171; fax: +1 805 893 8553.
E-mail addresses: ibarra@cs.ucsb.edu (O.H. Ibarra), ntran@math.scu.edu (N.Q. Tran).

0304-3975/$ − see front matter © 2012 Elsevier B.V. All rights reserved.
Two decision problems arise immediately: (i) given a multi-tape automaton, is it a k-synchronized multi-tape automaton for a given k or some unknown k? and (ii) given a multi-tape automaton, is there an equivalent k-synchronized multi-tape automaton for a given k or some unknown k? We call the former decision problem synchronization and latter synchronization.

In this paper we investigate the boundaries between decidability and undecidability for the above decision problems for variants of multi-tape pushdown automata that may include additional reversal-bounded counters. The main results of this paper are as follows.

1. It is decidable to determine, given an n-tape NPDA M, whether it is k-synchronized for some k, and if this is the case, the smallest such k can be found.
2. Any synchronized n-tape NPDA M can be converted to a 0-synchronized NPDA. In the case of NFA, we show that any synchronized n-tape NFA can be converted to an equivalent 0-synchronized n-tape DFA. Moreover, the sets of tuples accepted by synchronized n-tape NFAs are closed under union, intersection, and complementation.
3. It is undecidable to determine, given a 2-ambiguous 2-tape NFA M, whether there is a 0-synchronized 2-tape NFA (resp., DFA) M′ with 5 states such that L(M′) = L(M). (A machine is k-ambiguous if there are at most k accepting computations for any input. Note that being unambiguous is the same as being 1-ambiguous, and being deterministic is a special case of being unambiguous.)
4. It is decidable to determine, given a 1-ambiguous, n-tape NFA M and s ≥ 1, whether there is a 0-synchronized n-tape NFA (resp., DFA) M′ with s states such that L(M′) = L(M).

We also obtain decidability and undecidability results for two-way multi-tape NFAs and DFAs. We then consider the notion of k-synchronized one-way and two-way multi-head automata and investigate similar problems.

2. Preliminaries

A (one-way) n-track deterministic finite automaton (DFA) is a finite automaton whose input alphabet is of the form $\Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$ for some finite sets $\Sigma_1, \ldots, \Sigma_n$. The projection of such an input string to $\Sigma_i$ is called its ith track.

A (one-way) n-tape DFA M is a finite automaton with n tapes where each tape contains a string over input alphabet $\Sigma$. Each tape is read-only and has an associated one-way input head. We assume that each tape has a right end marker $\$$ (not in $\Sigma$). Sometimes the end markers are not shown, but they are assumed to be always present. On a given n-tape input $x = (x_1, \ldots, x_n) \in \Sigma^+ \times \ldots \times \Sigma^+ (n$ times), where each $x_i$ is delimited on the right by $\$, M starts in initial state $q_0$ with all the heads on the first symbols of their respective tapes. The transition function of M consists of rules of the form $\delta(q, a_1, \ldots, a_n) = (p, d_1, \ldots, d_n)$ (resp., $\delta(q, a_1, \ldots, a_n) = \epsilon$). This rule means that if M is in state $q$, with head $H_i$ on symbol $a_i$, then the machine moves $H_i$ in direction 1 or 0 (for right move or stationary move), and enters state $p$ (resp., halts). When a head reaches the end marker $\$$, that head has to remain on the end marker. The input $x$ is accepted if M reaches the configuration where all n heads are on $\$$ and M eventually enters an accepting state.

Suppose a set of n-tuples is accepted by a multi-tape DFA M if and only if the tuple components, when appropriately interleaved with $\lambda$ symbols and stacked vertically to form multi-track inputs, are accepted by a 1-tape multi-track DFA M′. When this holds, we say that M and M′ are equivalent.

Let M be an n-tape DFA and $k \geq 0$. M is k-synchronized (or k-aligned) if at any time during the computation on any input n-tuple $(x_1, \ldots, x_n)$ (accepted or not), no two heads that are both on symbols in $\Sigma$ are more than $k$ tape cells apart.” Notice that, since the condition in the definition concerns pairs of heads that are both on symbols in $\Sigma$, if one of these two heads is on $\$$, then we can stipulate that the condition is automatically satisfied, irrespective of the distance between the heads. Note that if $k = 0$, then all heads move to the right synchronously at the same time (except for heads that reach the right end marker early). M is finitely-synchronized if it is k-synchronized for some k.

A 0-synchronized n-tape DFA M can be represented by a 1-tape n-track DFA M′ as follows: an n-tuple input $x = (x_1, \ldots, x_n)$ to M, is represented by an n-track tape $x' = t_1 \ldots t_n$, where $s = \max \{|x_1|, \ldots, |x_n|\}$, and for $1 \leq j \leq s$, $t_j$ is an n-track symbol consisting of symbols in position j of each $x_i$, with the convention that when $j > |x_i|$, the symbol in the i-th track is $\lambda$. We will also use this equivalent representation of 0-synchronized n-tape DFAs in the sequel.

The above definitions generalize to n-tape nondeterministic finite automata (NFAs). Now, being k-synchronized requires that for any computation on any input n-tuple $(x_1, \ldots, x_n)$ (accepted or not), no two heads that are both on symbols in $\Sigma$ are more than k tape cells apart.

The definitions can also be generalized to n-tape deterministic pushdown automata (DPDAs) and n-tape nondeterministic pushdown automata (NPDAs), which may even be augmented with a finite number of reversal-bounded counters. At each step, each counter can be incremented by 1, decremented by 1, or left unchanged and can be tested for zero. The counters are reversal-bounded in the sense that during any computation, no counter can change mode from increasing to decreasing and vice-versa more than a specified fixed number of times.

We will need the following result from [4].

**Theorem 1.** The emptiness (is $L(M) = \emptyset$?) and infiniteness (is $L(M)$ infinite ?) problems for 1-tape NPDA with reversal-bounded counters are decidable.

**Corollary 1.** The emptiness and infiniteness problems for multi-tape NPDA with reversal-bounded counters are decidable.
The construction in the proof of Theorem 2 applies. In fact, since

**Proof.** It is decidable to determine, given an n-tape NPDAs, whether it is finitely-synchronized. 

**Proof.** Let M be an n-tape NPDAs. Clearly, M is not finitely-synchronized if for any given d ≥ 0, there is an input x = (x₁, . . . , xₙ) and some 1 ≤ i < j ≤ n such that on input x, M has a computation in which head Hᵢ (on tape i) has processed some portion yᵢ of xᵢ and head Hⱼ has processed some portion yⱼ of xⱼ (note that the last symbols of yᵢ and yⱼ are not the right end marker) such that ||yᵢ| − |yⱼ|| ≥ d.

We construct a 1-tape NPDAs M′ with two 1-reversal counters C₁ and C₂ that are initially zero. Given a unary input w = 1ᵈ, M′, without moving on the unary input, simulates the computation of M on (x₁, . . . , xₙ), by guessing the symbols that the heads of M read on each tape (a new symbol is guessed when the head has moved right until the right end marker $|$ is guessed when thereafter, no new symbol is guessed for that head). Before the simulation, M′ also guesses some 1 ≤ i < j ≤ n and during the computation, counter C₁ (resp., counter C₂) is incremented every time head Hᵢ (resp., head Hⱼ) moves right, provided the end marker $|$ has not yet been guessed by the heads.

At some point, M′ terminates the simulation of M and checks and accepts 1ᵈ if d ≤ |v₁ − v₂|, where v₁ (resp., v₂) is the value of counter C₁ (resp., counter C₂). Checking this condition is accomplished by decrementing the counters until one of them becomes zero and then verifying that the remaining value of the other counter is at least d. It follows that M is not finitely-synchronized if and only if M′ accepts an infinite language, which is decidable by Theorem 1. □

**Corollary 2.** It is decidable to determine, given an n-tape NPDAs M and an integer k ≥ 0, whether M is k-synchronized.

**Proof.** The construction in the proof of Theorem 2 applies. In fact, since k is fixed, M′ does not need the 1-reversal counters. M′ can simply keep track of the “distance” between heads Hᵢ and Hⱼ in the state. So M is just an ordinary NPDAs.

The following follows from the previous two results.

**Corollary 3.** It is decidable to determine, given an n-tape NPDAs M, whether it is k-synchronized for some k. Moreover, if it is, we can effectively determine the smallest such k.

Clearly, the above results generalize to machines with reversal-bounded counters; in particular to the following.

**Theorem 3.** It is decidable to determine, given an n-tape NPDAs M augmented with reversal-bounded counters, whether it is k-synchronized for some k. Moreover, if it is, we can effectively determine the smallest such k.

The results above are best possible in the sense that they do not hold if the NPDAs is augmented with an unrestricted counter. In fact, we have the following.

**Theorem 4.** It is undecidable to determine, given a 2-tape DFA augmented with two unrestricted counters (i.e., non-reversal bounded), whether it is 0-synchronized.

**Proof.** It is known that the halting problem for 2-counter machines (where both counters are initially zero) is undecidable [6]. Given any such machine M, we construct a 2-tape DFA M′ over a unary alphabet, augmented with two counters. Given input (x₁, x₂), M′ first simulates the computation of M on the two counters. Clearly, if M does not halt, M is 0-synchronized (since the heads on the two tapes of M′ are not moved). If M halts, then M′ moves the head on x₁ to the right end marker and then moves the head on x₂ to the right end marker. Since x₁ and x₂ are arbitrary unary strings, M′ is not 0-synchronized; in fact, it is not k-synchronized for any k. The result follows. □

We conclude this section with a result which shows that if a multi-tape DFA M is k-synchronized, then k is smaller than the number of states of M, i.e., there is a “gap” between (q − 1)-synchronized and q-synchronized for machines with q states.

**Theorem 5.** Let M be an n-tape DFA with q states. If M is not (q − 1)-synchronized, then M is not k-synchronized for any k ≥ 0.

**Proof.** Since M is not (q − 1)-synchronized, there is an input x = (x₁, . . . , xₙ) and 1 ≤ i < j ≤ n such that on input x, M has a computation in which head Hᵢ (on tape i) has processed some portion yᵢ of xᵢ and head Hⱼ has processed some portion of yⱼ of xⱼ, where:

1. the last symbols of yᵢ and yⱼ are not the right end marker;
2. |yᵢ| − |yⱼ| ≥ q.
5. Synchronizability of multi-tape automata

This follows from Theorems 6 and 7 and the fact that closure under union is obvious. □

5. Synchronizability of multi-tape automata

In this section, we look at the problem of deciding, given a multi-tape automaton of a certain type, whether there exists an equivalent synchronized multi-tape automaton of the same type.

5.1. Synchronizability of an n-tape DFA/NFA

Note that by Theorems 6 and 7, in what follows, results that involve 0-synchronized (resp., k-synchronized for a given k, k-synchronized for some k) n-tape NFAs also hold for the corresponding n-tape DFAs, and vice-versa.
Theorem 8. For any \( n \geq 2 \), there exists an \( n \)-tape DFA \( M \) that cannot be converted to an equivalent synchronized \( n \)-tape DFA \( M' \).

Proof. Let \( L = \{(a^i c^j \mid i, j \geq 1\} \). Clearly, \( L \) can be accepted by a 2-tape DFA. Suppose \( L \) can be accepted by a synchronized 2-tape DFA. Then it can also be accepted by a 0-synchronized 2-tape DFA. This DFA in turn can be represented as a 1-tape 2-track DFA \( M' \), and hence the pumping lemma applies to the language that \( M' \) accepts. Let \( M' \) have \( s \) states. Consider the 2-track string

\[
w = \left( a^{i+1} c^{i+1} \right)_{c^{i+1} \lambda^{i+1}}.
\]

Then \( w \) is accepted by \( M' \). Then there exist \( i, k \geq 0 \) and \( j \geq 1 \) such that \( w \) decomposes into

\[
w = \left( a^i a^{i+1-i-j} c^{i+1} \right)_{c^i c^{i+1-i-j \lambda^{j+1}}}
\]

and

\[
\left( a^{m} a^{i+1-i-j} c^{i+1} \right)_{c^{m} c^{i+1-i-j \lambda^{j+1}}}
\]

is accepted by \( M' \) for every \( m \geq 0 \). Let \( m = 2 \). Then

\[
w' = \left( a^{2i} a^{i+1-i-j} c^{i+1} \right)_{c^{2i} c^{i+1-i-j \lambda^{j+1}}}
\]

is accepted by \( M' \). But now, the first track of \( w' \) contains the string \( a^{i+1+j} c^{i+1} \), and the second track contains \( c^{i+1+j} \). Since \( j \geq 1 \), this is a contradiction since the number of \( c \)'s in the first track is less than the number of \( c \)'s in the second track. \( \square \)

On the other hand, there are examples of non-synchronized \( n \)-tape DFAs which can be converted to synchronized \( n \)-tape DFAs. Consider, e.g., the set \( L = \{(a^i, a^j) \mid i, j \geq 1\} \). We can construct a 2-tape DFA \( M \) which reads the first tape until its head reaches the end marker, and then reads the second tape until its head reaches the end marker, and then \( M \) accepts. This machine is not synchronized. But, we can construct a 0-synchronized 2-tape DFA \( M \), which when given, \( (a^i, a^j) \), the two heads move to the right reading \( (a, a) \)'s until one of the head reaches the end marker. Then \( M' \) reads the remaining \( a \)'s on the other tape and accepts. Thus, the following is an interesting problem.

Open: Is it decidable to determine, given an \( n \)-tape DFA \( M \), whether there exists a synchronized \( n \)-tape DFA \( M' \) such that \( L(M') = L(M) \)?

We do not know the answer to the above problem at this time. However, we can show that the corresponding problem for NFA is undecidable. In fact, we prove a stronger result.

Theorem 9. It is undecidable to determine, given a 2-ambiguous 2-tape NFA \( M \), whether there exists a 2-tape NFA \( M' \) such that \( L(M') = L(M) \) and \( M' \) is 0-synchronized (resp., \( k \)-synchronized for a given \( k \), \( k \)-synchronized for some \( k \)).

Proof. We reduce to this problem the halting problem for single-tape Turing machines on blank input. Note that if such a machine \( Z \) has a halting sequence of configurations, the sequence is unique. Without loss of generality, we may assume that the number of steps is odd, and that the Turing machine does not write blank symbols. Hence if \( C \) is a configuration and \( D \) is its valid successor configuration, then the length of \( D \) is at most one more than that of \( C \). Thus, the unique halting sequence of configurations has the form

\[
C_1 # D_1 # C_2 # D_2 # C_3 # D_3 # \cdots # C_k # D_k
\]

where:
- \( k \geq 1 \);
- \( D_i \) is the successor of configuration \( C_i \) for \( i = 1, 2, \ldots, k \);
- \( C_{i+1} \) is the successor of configuration \( D_i \) for \( i = 1, 2, \ldots, k - 1 \);
- \( C_1 \) is the initial configuration and \( D_k \) is a halting configuration.

Let \( d \) be a new symbol. Construct a 2-tape NFA \( M_2 \) which, when given a tuple \( w = (d^i x, y) \), operates by nondeterministically selecting one of processes (1) and (2) below to execute.

1. \( M_2 \) checks and accepts \( w = (d^i x, y) \) if one of the following holds:
   (a) \( (x, y) \) is not of the form
   \[
   (C_1 # C_2 \# \cdots # C_k, D_1 # D_2 \# \cdots # D_k)
   \]
   where the \( C \)'s and \( D \)'s are configurations and the difference in lengths between \( C_i \) and \( D_i \) is at most 1, and \( C_1 \) is the initial configuration, and \( D_k \) is a halting configuration.
   (b) \( D_i \) is not the successor of \( C_i \) for some \( i \).

(2) $M_2$ checks and accepts $w = (d^i x, y)$ if one of the following holds:
   (a) $(x, y)$ is not of the form
      $$(C_1 # C_2 # \cdots # C_k, D_1 # D_2 # \cdots # D_k)$$
   where the $C$'s and $D$'s are configurations and the difference in lengths between $C_i$ and $D_i$ is at most 1, and $C_1$ is the initial configuration, and $D_k$ is a halting configuration.
   (b) $C_{i+1}$ is not the successor of $D_i$ for some $i$. 

Clearly, $M_i$ is 2-ambiguous, since it can execute each of (1) and (2) deterministically.

Let $L$ be the set of tuples accepted by $M_2$. If the Turing machine $Z$ does not halt on blank tape, $L$ consists of all tuples of the form $(d^i x, y)$ (for any $j$, $x$, $y$), and it is straightforward to construct a 0-synchronized 2-tape NFA to accept $L$.

However, if $Z$ halts on blank tape, and
$$C_1 # D_1 # C_2 # D_2 # \cdots # C_k # D_k$$
is the halting sequence of configurations of $Z$, then $M_2$ will not accept
$$(d^i C_1 # C_2 # \cdots # C_k, D_1 # D_2 # \cdots # D_k)$$
for any $j$. We now show that $L$ cannot be accepted by any 0-synchronized 2-tape NFA.

To see this, recall that a 0-synchronized 2-tape NFA $N$ is equivalent to a 1-tape 2-track NFA $N'$ in the following sense: $N$ accepts a pair $(x, y)$ if and only if $N'$ accepts the 2-track input $(x')$ obtained by stacking vertically left-justified $x$ and $y$ and padding the shorter string with λ's so that they have the same length. Define $L' = \{(x') \mid (x, y) \in L\}$. Our claim is equivalent to showing that $L'$ is not regular.

Suppose $L'$ is accepted by some DFA $N'$ with $q$ states, so $N'$ rejects the input
$$\left(\begin{array}{c}
  d^{q+1} C_1 # C_2 # \cdots # C_k \\
  D_1 # D_2 # \cdots # D_k
\end{array}\right).$$
(To simplify notation, we omit above the padded λ's needed for the two tracks to have the same length.) Clearly, there must be some $i < j \leq q + 1$ such that $N'$ is in the same state when it reads the $i$th and $j$th symbols of this input. Let $s = \left(\begin{array}{c}
  d^{j-i+1} \\
  \alpha
\end{array}\right)$ be the substring starting at position $i$ and ending at position $j$ inclusive. It is clear that $N'$ must also reject all inputs obtained by repeating $s k$ times for any $k$.

If the lower-track component $\alpha$ of $s$ contains a #, then repeating $s$ causes the lower track of such a “pumped” input to have more #s than the upper track. Since such input does not have the correct format, it should be in $L'$. On the other hand, if $\alpha$ does not contain a #, then it must be a substring of some $D_l$ for some $l \leq q$. Again, repeating $s$ twice will cause $D_l$ to have length more than the length of $C_l$ plus 1, so the result is again a string that does not have the correct format and hence should be in $L'$. In both cases, we obtain a contradiction, since $N'$ rejects a string that is in $L'$. □

**Open:** Can the above theorem be strengthened to hold for 1-ambiguous 2-tape NFAs?

**Corollary 5.** It is undecidable to determine, given a 2-ambiguous 2-tape NFA $M$, whether there is a 0-synchronized 2-tape DFA $M'$ with 5 states such that $L(M') = L(M)$. 

**Proof.** From the proof of Theorem 9, if the Turing machine (TM) $Z$ does not halt on blank tape, then $L(M) = \{(d^j x, y) \mid$ for any $j, x, y\}$ can clearly be accepted by a 0-synchronized 2-tape DFA $M'$ with 5 states. If the TM $Z$ halts, then there is no 0-synchronized 2-tape NFA $M'$ (with any number of states) such that $L(M') = L(M)$. □

**Corollary 6.** It is undecidable to determine, given a 2-ambiguous 2-tape NFA $M$ and a 0-synchronized 2-tape DFA $M'$ with 5 states, whether $L(M) = L(M')$. 

**Proof.** If this problem is decidable, we would contradict Corollary 5, since given $M$, we can systematically enumerate all 0-synchronized 2-tape DFAs $M'$ with 5 states and check if one of these is equivalent to $M$. □

In contrast to the above corollaries, we have the following.

**Proposition 1.** It is decidable to determine, given a 1-ambiguous $n$-tape NFA $M$ and $s \geq 1$, whether there is a 0-synchronized 1-ambiguous $n$-tape NFA (resp., 0-synchronized $n$-tape DFA) $M'$ with $s$ states such that $L(M') = L(M)$. 

**Proof.** This follows from the fact that equivalence of $n$-tape DFAs (resp. 1-ambiguous $n$-tape NFAs) is decidable [2], and the observation that we can systematically enumerate all 0-synchronized $n$-tape DFAs (resp., 0-synchronized 1-ambiguous $n$-tape NFAs) with $s$ states and check if one of them is equivalent to $M$. □
5.2. Synchronizability of an n-tape DPDA/NPDA

Again we note that from Theorem 6, any k-synchronized n-tape NPDA (DPDA) can be converted to an equivalent 0-synchronized n-tape NPDA (DPDA).

**Theorem 10.** For any \( n \geq 2 \), there exists an n-tape DPDA (in fact, an n-tape deterministic counter machine whose counter makes only one reversal, i.e., n-tape 1-reversal DCM) that cannot be converted to an equivalent synchronized n-tape NPDA.

**Proof.** Let \( L = \{(a^i b^j c^k) \mid i, k \geq 1 \} \). Clearly, \( L \) can be accepted by a 2-tape DPDA (in fact, by a 1-turn DCM). Suppose \( L \) can be accepted by a synchronized 2-tape NPDA. Hence, \( L \) can be accepted by a 0-synchronized 2-tape NPDA. This NPDA can be represented as a 1-tape 2-track NPDA. By an argument similar to the proof of Theorem 8, using the pumping technique to show that \( \{c^i a^j b^k \mid k \geq 1 \} \) is not a context-free language (i.e., not accepted by a 1-tape NPDA), we will arrive at a contradiction. □

We note that \( L = \{(a^i c^k) \mid i, k \geq 1 \} \), which cannot be accepted by a synchronized 2-tape NFA can be accepted by a 0-synchronized 2-track 1-turn DCM.

**Theorem 11.** It is undecidable to determine, given a 2-ambiguous 2-tape NPDA \( M \), whether there exists a 2-tape DPDA \( M' \) such that \( L(M') = L(M) \) and \( M' \) is 0-synchronized (resp., \( k \)-synchronized for a given \( k \), \( k \)-synchronized for some \( k \)).

**Proof.** As in the proof of Theorem 9, we reduce to this problem the halting problem for single-tape Turing machines on blank input. Given a single-tape Turing machine \( Z \), let

\[
C_1 \# D_1 \# \cdots \# C_k \# D_k
\]

be the unique halting sequence of configurations of \( Z \) if it exists.

Let \( a, b, c \) and \( d \) be new symbols. Construct a 2-tape NPDA \( M_Z \) which, when given a tuple \((xd^i d^j b^2 c^i, yd^i b^2 c^j a^j)\) where \( r, s, i, j, x, y, j \) are nonnegative integers, operates by nondeterministically selecting one of processes (1) and (2) below to execute.

1. \( M_Z \) checks and accepts \( w = (xd^i b^2 c^i, yd^i b^2 c^j a^j) \) if one of the following holds:
   - \( (x, y) \) is not of the form \((C_1 \# C_2 \# \cdots \# C_k, D_1 \# D_2 \# \cdots \# D_k)\)
     where the \( C_i \)'s and \( D_i \)'s are configurations and the difference in lengths between \( C_i \) and \( D_i \) is at most 1, and \( C_1 \) is the initial configuration, and \( D_1 \) is a halting configuration.
   - \( D_1 \) is not the successor of \( C_i \) for some \( i \).
   - \( l_1 \neq l_2 \) or \( l_2 \neq j_2 \) or \( i_3 \neq j_3 \).

2. \( M_Z \) checks and accepts \( w = (xd^i a^j b^2 c^i, yd^i b^2 c^j a^j) \) if one of the following holds:
   - \( (x, y) \) is not of the form \((C_1 \# C_2 \# \cdots \# C_k, D_1 \# D_2 \# \cdots \# D_k)\)
     where the \( C_i \)'s and \( D_i \)'s are configurations and the difference in lengths between \( C_i \) and \( D_i \) is at most 1, and \( C_1 \) is the initial configuration, and \( D_1 \) is a halting configuration.
   - \( C_{i+1} \) is not the successor of \( D_i \) for some \( i \).
   - \( l_1 \neq l_1 \) or \( l_2 \neq j_2 \) or \( i_3 \neq j_3 \).

Clearly, \( M_Z \) is 2-ambiguous, since it can execute each of (1) and (2) deterministically.

Let \( L \) be the set of tuples accepted by \( M_Z \). If the Turing machine \( Z \) does not halt on blank tape, \( L \) consists of all tuples of the form \((xd^i d^j b^2 c^i, yd^i b^2 c^j a^j)\) (for any \( r, s, i, j, x, y, j \), and it is straightforward to construct a 0-synchronized 2-tape DPDA (in fact, DFA) to accept \( L \).

However, if \( Z \) halts on blank tape, and

\[
C_1 \# D_1 \# C_2 \# D_2 \# C_3 \# D_3 \# \cdots \# C_k \# D_k
\]

is the halting sequence of configurations of \( Z \), then \( M_Z \) will not accept

\[
(C_1 \# C_2 \# C_3 \# \cdots \# C_n \# d^i b^j c^i, D_1 \# D_2 \# D_3 \# \cdots \# D_n \# d^i b^j c^i)
\]

for any \( r, s, n \). We now show that \( L \) cannot be accepted by any 0-synchronized 2-tape DPDA.

Again, we note that a 0-synchronized 2-tape DPDA \( N \) is equivalent to a 1-tape 2-track DPDA \( N' \) in the following sense: \( N \) accepts a pair \((x, y)\) if and only if \( N' \) accepts the 2-track input \((\lambda x y)\) obtained by stacking vertically left-justified \( x \) and \( y \) and padding the shorter string with \( \lambda \)'s so that they have the same length.

Define \( L' = \{(x, y) \mid (x, y) \in L \} \). Our claim is equivalent to showing that \( L' \) is not accepted by any DPDA; further, since the class of languages accepted by DPDAs is closed under complementation, our claim is equivalent to showing that \( L' \), the complement of \( L' \), is not accepted by any DPDA, i.e., it is not a context-free language.
Suppose $\bar{L}$ is a context-free language. Then it contains

$$w = \left( C_1 \# C_2 \# C_3 \ldots \# C_d \# \bar{a}^n \bar{b}^n \bar{c}^n \\
D_1 \# D_2 \# D_3 \ldots \# D_d \# \bar{b}^n \bar{c}^n \bar{a}^n \right),$$

where $n$ is the constant in Ogden’s lemma for context-free languages, and $r$ and $s$ are chosen so that the strings on both tracks have the same length, and hence

$$\left( \bar{a}^n \bar{b}^n \bar{c}^n \right)$$

is a suffix of $w$. We mark all positions of this suffix. Ogden’s lemma states that $w$ can be written as $uxyzv$ such that

1. $xz$ has at least one marked position;
2. $xyz$ has at most $n$ marked positions, and
3. $ux^iy^jz^kvy^l \in \bar{L}$ for every $i \geq 0$.

It is clear that $xyz$ must be a substring of $(C_1 \# \ldots \# C_d \# \bar{a}^n D_1 \# \ldots \# D_d \# \bar{b}^n) \cdot (\bar{a}^n \bar{b}^n) \cdot (\bar{b}^n \bar{c}^n) \cdot (\bar{c}^n \bar{a}^n)$.

If $xyz$ is a substring of $(\bar{a}^n \bar{b}^n \bar{c}^n)$, pumping in $w$ results in too many $a$’s or too many $b$’s on the upper track, contradicting the definition of $\bar{L}$; analogous arguments apply to the remaining cases. \hfill \Box

Open: Can the above theorem be shown to hold for $M'$ being a 2-tape NPDA?

6. Two-way multi-tape NFAs

In this section, we consider the synchronization problem for two-way multi-tape NFAs. A two-way $n$-tape NFA $M$ is a generalization of an $n$-tape NFA in that the input heads can now move two-way ($-1$, $0$, $+1$) on their respective input tapes which are provided with left end marker # and right end marker $. Initially, all heads are on $. $M$ accepts a $n$-tuple $(\#x_1, \ldots, \#x_n, \ldots, \#x_k)$ if all heads reach $\$, and the machine eventually enters an accepting state. $M$ is $k$-synchronized if at any time during any computation on any input $n$-tuple (accepted or not), no two heads that are not on $\$ are more than $k$ cells apart (as measured from their left end marker #). $M$ is finitely-synchronized if it is $k$-synchronized for some $k$.

Theorem 12. It is undecidable to determine, given a two-way 2-tape DFA (hence, also for two-way 2-tape NFA) $M$, whether $M$ is finitely-synchronized.

Proof. It is well-known that it is undecidable to determine, given a deterministic 2-counter machine $Z$, where both counters are initially zero (there is no input tape), whether $Z$ will halt. Furthermore, it is well-known that it is undecidable to determine, given a deterministic 2-counter machine $Z$, whether $Z$ will halt in [6]. (There is no input tape.) A close look at the proof of the undecidability of the halting problem, where initially one counter has value $d_1$ and the other counter is zero in [6] reveals that the counters behave in a regular pattern. $Z$ operates in phases in the following way. Let $C_1$ and $C_2$ be its counters. Then the machine’s operation can be divided into phases, where each phase starts with one of the counters equal to zero and the other counter equal to some positive integer $d_i$. During the phase, the first counter is increasing, while the second counter is decreasing. The phase ends with the first counter having value $d_{i+1}$ and the second counter having value 0. Then in the next phase the modes of the counters are interchanged. We can also assume that if $Z$ does not halt, the values of counters during the computation are increasing, i.e., the $d_i$’s cannot be bounded by a constant. It follows that if $Z$ goes into an infinite loop, the difference between the values of the counters will grow unboundedly.

We construct a two-way 2-tape DFA $M$ which, when given input $(\#a^j \$, $\#a^j)$ where $i, j \geq 0$ with both heads on $\$, simulates $Z$ faithfully using the two tapes to simulate the counters (thus, the heads on the tapes move right for increment and move left for decrement). When $Z$ halts, $M$ then moves both heads simultaneously to the right until one head reaches $\$ and then moves the other head to $\$ and then accepts. If one of the heads of $M$ reaches $\$ before $Z$ halts, $M$ moves the other head to $\$ and accepts.

Clearly, if $Z$ halts after at most $k$ steps (for some $k$), then $M$ accepts all tuples in $\#a^k \times \#a^k$ and the heads are at most $k$ cells apart during any computation.

If $Z$ does not halt, then $M$ also accepts all tuples in $\#a^k \times \#a^k$, but the distance between the two heads cannot be bounded by any constant during all computations.

It follows that $M$ is finitely-synchronized if and only if $Z$ halts, which is undecidable. \hfill \Box

On the other hand, for given $k$, the $k$-synchronizability problem is decidable.

Theorem 13. It is decidable to determine, given a two-way $n$-tape NFA $M$ and a nonnegative integer $k$, whether $M$ is $k$-synchronized.

Proof. Given $M$, we construct a two-way 1-tape NFA $M'$ operating on an input string that has $n$ tracks. If the input to $M$ is $(\#x_1, \ldots, \#x_n)$, then the input string $y$ to $M'$ consists of $n$ tracks, where track $i$ is the string $\#x_i \lambda^j \$ for some $d_i \geq 0$ (where $\lambda$ is a new symbol representing a blank), such that the $x_i$’s are left-justified and blank-filled so that the lengths of the tracks are all the same.
Clearly, $M'$ with only one head can simulate the computation and keep track of the movements of the $n$ heads of $M$, using a finite buffer in its states. If, during the simulation, a pair of heads attempts to “separate” more than $k$ cells apart, $M'$ accepts the input. It follows that $M$ is not $k$-synchronized if and only if $M'$ accepts a nonempty language, which is decidable since $M'$ accepts a regular set. \(\square\)

7. Multi-head automata

In this section, we look at finitely-synchronized multi-head automata. Like an $n$-tape automaton, an $n$-head automaton has $n$ independent heads that operate either one-way or two-way on one input tape with end markers. The automaton is $k$-synchronized if at any time during the computation on any input, no two heads are more than $k$ cells apart. It is finitely-synchronized if it is $k$-synchronized for some $k$.

7.1. The one-way model

**Theorem 14.** It is undecidable to determine, given a one-way 2-head DFA $M$, whether $M$ is finitely-synchronized.

**Proof.** We will use the fact that it is undecidable to determine, given a single-tape Turing machine $Z$, whether it will halt on an initially blank tape.

We may assume that if $Z$ loops does not halt), it does not loop on a finite amount of tape, since we can easily construct an equivalent TM $Z'$ as follows: $Z'$ simulates $Z$ one step and a time, and between steps of $Z$, $Z'$ searches for the closest blank symbol to the right and replaces it with a pseudo-blank symbol that will be treated as a blank in the simulation. Clearly, if $Z$ goes into an infinite loop, $Z'$ will go into an infinite loop on infinitely many cells.

Thus, if $Z$ loops, and $C_1 \# C_2 \# C_3 \# \cdots$ (where $C_i$ is the initial configuration of $Z$ on blank tape, and $C_{i+1}$ is a direct successor of $C_i$ for $i \geq 1$) is the non-halting computation of $Z$, then for every $r$, there is a $C_i$ such that the length of $C_i > r$.

We construct a one-way 2-head DFA $M$ which, on input $w$, uses the two heads to check if $w$ is a halting computation of $Z$ on blank tape, i.e., $w = C_1 \# \cdots \# C_n$, where $C_1$ is the initial configuration, $C_i$ is a halting configuration, and $C_{i+1}$ is a direct successor of $C_i$ for $1 \leq i \leq n - 1$. Note that in the computation of $M$, one head is lagging behind the other head by one configuration. We omit the details.

Clearly, if $Z$ halts on blank tape, then there exists a $k$ such that it is $k$-synchronized. If $Z$ does not halt on blank tape, $M$ will not be $k$-synchronized for any $k$. \(\square\)

However, for restricted multi-head automata, we can prove a positive result.

**Theorem 15.** It is decidable to determine, given a one-way $n$-head NPDA $M$ whose inputs come from $w_1^n \cdots w_t^n$ for some (not necessarily distinct) non-null strings $w_1, \ldots, w_t$, whether $M$ is finitely-synchronized.

**Proof.** First consider the case when $w_1, \ldots, w_t$ are distinct symbols $a_1, \ldots, a_r$. Given $M$ with heads $H_1, \ldots, H_n$, we construct an $n$-head NPDA $M'$ with the same heads as $M$. Let $b$ be a new symbol different from $a_1, \ldots, a_r$. The input to $M'$ is a string of the form $y' = a_1^r \cdots a_r^r$. Now $M'$ simulates $M$ on $y$ with heads $H_1, \ldots, H_n$. At some point (nondeterministically chosen), $M'$ terminates the simulation. $M'$ guesses $1 \leq i, j \leq n$ with $i \neq j$ and checks that $H_i$ and $H_j$ have not reached symbol $b$. It then moves both heads simultaneously to the right and accepts if one of these heads reaches the right end marker $\$ at the same time that the other head reaches the first $b$ of the input segment $b'$ (indicating that distance between heads $H_i$ and $H_j$ when the simulation of $M$ was terminated was $r$).

Now $M'$ accepts a bounded language $L(M) \subseteq a_1^r \cdots a_r^r b^\ast$. It is known that the Parikh map of the bounded language accepted by a multi-head NPDA is an effectively computable semilinear set $Q \subseteq N^{r+1}$. It follows that the projection of this semilinear set on the last coordinate (corresponding to the multiplicity $r$ of symbol $b$) is also an effectively computable semilinear set $Q'$. Clearly, $M$ is finitely-synchronized if and only if $Q'$ is finite, which is decidable since finiteness of semilinear sets is decidable.

The result for the general case follows from the observation that given $M$ whose inputs come from $w_1^n \cdots w_t^n$, we can construct an $n$-head NPDA $M'$ whose inputs come from $a_1^r \cdots a_r^r$ such that $L(M') = \{a_1^{n_1} \cdots a_r^{n_r} | w_1^{n_1} \cdots w_t^{n_t} \in L(M)\}$, and $M'$ is finitely-synchronized if and only if $M$ is finitely-synchronized. Note that nondeterminism is essential for the above construction. \(\square\)

We showed in Theorem 14 that it is undecidable to determine, given a one-way 2-head DFA $M$, whether it is finitely-synchronized and, obviously, this is true when $M$ is a one-way 2-head DPDA. In contrast when $k$ is given, we have the following.

**Theorem 16.** It is decidable to determine, given a one-way $n$-head NPDA $M$ augmented with reversal-bounded counters and an integer $k$, whether $M$ is $k$-synchronized.

**Proof.** Construct a 1-head NPDA $M'$ augmented with reversal-bounded counters that tries to simulate the $n$ heads of $M$ in its finite control. $M'$ accepts an input $x$ if at least two heads of $M$ on $x$ become separated by more than $k$ cells during the computation. Clearly, $M'$ is $k$-synchronized if and only if $L(M')$ is empty, which is decidable by Theorem 1. \(\square\)
7.2. The two-way model

(a) Finite-turn case

If in the two-way model, the \( n \) heads of the NPDA \( M \) are finite-turn in the sense that the heads can make turns on the input tape (which is provided with left and right end markers) at most a fixed number of times, then the proof of Theorem 15 generalizes since the Parikh map of the bounded language (i.e., inputs coming from \( a_1^* \cdots a_t^*b^* \)) accepted by a multi-head NPDA whose input heads are two-way finite turn is also an effectively computable semilinear set [3]. Then we have the following.

**Theorem 17.** It is decidable to determine, given a two-way finite-turn \( n \)-head NPDA \( M \) whose inputs come from \( w_1^* \cdots w_t^* \) for some strings \( w_1, \ldots, w_t \), whether \( M \) is finitely-synchronized.

(b) Unrestricted-turn case

**Theorem 18.** It is decidable to determine, given a two-way multi-head NFA \( M \) and an integer \( k \), whether \( M \) is \( k \)-synchronized.

**Proof.** Similar to the proof of Theorem 16, but now \( M' \) is a two-way 1-head NFA. The result follows since the emptiness problem of two-way 1-head NFAs is decidable. \( \square \)

The above theorem does not hold for two-way multi-head NPDA. In fact, we can prove that it does not hold even for two-way 2-head DCMs (deterministic 1-counter machines).

**Theorem 19.** It is undecidable to determine, given a two-way 2-head DCM \( M \) and an integer \( k \), whether \( M \) is \( k \)-synchronized.

**Proof.** Let \( M \) be a two-way 1-head DCM. It is known that the emptiness problem for these machines is undecidable (this follows from the undecidability of halting for 2-counter machines [6]). Construct a two-way 2-head DCM \( M' \) which operates as follows: if \( x \) is an input to \( M \), the input to \( M' \) is \( x#^d \), where \# is a new symbol and \( d \geq 0 \). So for a given \( x \), there are infinitely many inputs to \( M' \). The two heads of \( M' \) move simultaneously (with no separation) as a pair simulating the single head of \( M \). If \( M \) accepts \( x \), then the first head of \( M \) remains on the last symbol of \( x \) and the other head moves to the end of \( #^d \) and accepts. If \( M \) does not halt or \( M \) rejects, the two heads are not separated. It follows that \( M' \) is \( k \)-synchronized (in fact 0-synchronized) if and only if \( L(M) \) is empty, hence, the result. \( \square \)

**Corollary 7.** It is undecidable to determine, given a two-way 2-head DPDA \( M \), and an integer \( k \), whether \( M \) is \( k \)-synchronized.

Finally, in contrast to Theorem 19, since the emptiness problem for two-way 1-head DCMs whose counter is reversal-bounded is decidable [5], we have the following result using an argument similar to the proof of Theorem 18.

**Theorem 20.** It is decidable to determine, given a two-way 2-head DCM \( M \) whose counter is reversal-bounded and an integer \( k \), whether \( M \) is \( k \)-synchronized.

8. Conclusion

We investigated the boundaries between decidability and undecidability of the synchronization and synchronizability decision problems for variants of one-way multi-tape pushdown automata, two-way multi-tape NFAs, and multi-head automata.

For one-way multi-tape automata, we showed that the synchronization decision problem (for both known and unknown \( k \)) is decidable for pushdown automata even when augmented with reversal-bounded counters. This result is tight in the sense that synchronization when \( k = 0 \) is undecidable for 2-tape DFAs with two unrestricted counters.

For two-way multi-tape automata, we proved that synchronization for NFAs is decidable when \( k \) is given and yet undecidable when \( k \) is not known.

For one-way multi-head automata, we showed that the synchronization for 2-head DFAs is undecidable when \( k \) is not known. The problem becomes decidable when \( k \) is given or when the input language is bounded, i.e., of the form \( w_1^* \cdots w_t^* \) for non-null strings \( w_1, \ldots, w_t \), even for multi-head NPDA.

For two-way multi-head automata, we showed that synchronization is decidable even when \( k \) is not known for finite-turn NPDA with bounded input but undecidable for general NPDA even when \( k \) is given.

Finally, we proved that the synchronizability problem is undecidable for 2-ambiguous 2-tape NFAs/NPDA even when \( k = 0 \).

Acknowledgments

The authors thank Zhe Dang, Bala Ravikumar, and Fang Yu for helpful comments on this work. They also thank the referees for their detailed comments which improve the presentation of this paper. The first author was supported in part by NSF Grants CCF-1143892 and CCF-1117708.
References


