Composability of Infinite-State Activity Automata*

Zhe Dang[1], Oscar H. Ibarra[2,**], and Jianwen Su[2]

[1]School of Electrical Engineering and Computer Science,
Washington State University,
Pullman, WA 99164, USA
[2]Department of Computer Science,
University of California,
Santa Barbara, CA 93106, USA
ibarra@cs.ucsb.edu

Abstract. Let $\mathcal{M}$ be a class of (possibly nondeterministic) language acceptors with a one-way input tape. A system $(A; A_1, ..., A_r)$ of automata in $\mathcal{M}$, is *composable* if for every string $w = a_1...a_n$ of symbols accepted by A, there is an assignment of each symbol in w to one of the A_i's such that if w_i is the subsequence assigned to A_i, then w_i is accepted by A_i. For a nonnegative integer k, a k-lookahead delegator for $(A; A_1, ..., A_r)$ is a *deterministic* machine D in $\mathcal{M}$ which, knowing (a) the current states of $A, A_1, ..., A_r$ and the accessible "local" information of each machine (e.g., the top of the stack if each machine is a pushdown automaton, whether a counter is zero on nonzero if each machine is a multicounter automaton, etc.), and (b) the k lookahead symbols to the right of the current input symbol being processed, can uniquely determine the A_i to assign the current symbol. Moreover, every string w accepted by A is also accepted by D, i.e., the subsequence of string w delegated by D to each A_i is accepted by A_i. Thus, k-lookahead delegation is a stronger requirement than composability, since the delegator D must be deterministic. A system that is composable may not have a k-delegator for any k. We look at the decidability of composability and existence of k-delegators for various classes of machines $\mathcal{M}$. Our results have applications to automated composition of e-services. When e-services are modeled by automata whose alphabet represents a set of activities or tasks to be performed (namely, activity automata), automated design is the problem of "delegating" activities of the composite e-service to existing e-services so that each word accepted by the composite e-service can be accepted by those e-services collectively with each accepting a subsequence of the word, under possibly some Presburger constraints on the numbers and types of activities that can be delegated to the different e-services. Our results generalize earlier ones (and resolve some open questions) concerning composability of deterministic finite automata as e-services to finite automata that are augmented with unbounded storage (e.g., counters and pushdown stacks) and finite automata with discrete clocks (i.e., discrete timed automata).

* The research of Oscar H. Ibarra and Jianwen Su was supported in part by NSF Grants IIS-0101134 and CCR02-08595.

** Corresponding author.

R. Fleischer and G. Trippen (Eds.): ISAAC 2004, LNCS 3341, pp. 377–388, 2004.

1 Introduction

In traditional automata theory, an automaton is a language acceptor that is equipped with finite memory and possibly other unbounded storage devices such as a counter, a stack, a queue, etc. The automaton "scans" a given input word in a one-way/two-way and nondeterministic/deterministic manner while performing state transitions. As one of the most fundamental concept in theoretical computer science, automata are also widely used in many other areas of computer science, in particular, in modeling and analyzing distributed and concurrent systems. For instance, one may view a symbol a in an input word that is read by the automaton as an input/output signal (event). This view naturally leads to automata-based formal models like I/O automata [18]. On the other hand, when one views symbol a as an (observable) activity that a system performs, the automaton can be used to specify the (observable) behavior model of the system; i.e., an activity automaton of the system. For instance, activity automata have been used in defining an event-based formal model of workflow [23]. Recently, activity (finite) automata are used in [4] to model e-services, which are an emerging paradigm for discovery, flexible interoperation, and dynamic composition of distributed and heterogeneous processes on the web or the Internet. An important goal as well as an unsolved challenging problem in service oriented computing [19] such as e-services is *automated composition*: how to construct an "implementation" of a desired e-service in terms of existing e-services.

To approach the automated composition problem, the technique adopted in [4] has two inputs. One input is a finite set of activity finite automata, each of which models an "atomic" e-service. The second is a desired global behavior, also specified as an activity finite automaton, that describes the possible sequences of activities of the e-service to be composed. The output of the technique is a (deterministic) *delegator* that will coordinate the activities of those atomic e-services through a form of delegation. Finding a delegator, if it exists, was shown to be in EXPTIME. The framework was extended in [12] by allowing "lookahead" of the delegator, i.e., to have the knowledge of future incoming activities. A procedure was given which computes a sufficient amount of lookahead needed to perform delegation; however, the procedure is not guaranteed to terminate.

The models studied in [4, 12] have significant limitations:only regular activities are considered since the underlying activity models are finite automata. In reality, more complex and non-regular activity sequences are possible. For instance, activity sequences describing a session of activities `releaseA`s, `allocateA`s, `releaseB`s and `allocateB`s satisfying the condition that the absolute difference between the number of `releaseA`s and the number of `allocateA`s, as well as the absolute difference between the number of `releaseB`s and the number of `allocateB`s, is bounded by 10 (the condition can be understood as some sort of fairness) are obviously non-regular (not even context-free). Therefore, in this paper, we will use the composition model of [12] but focus on, instead of finite automata, infinite-state (activity) automata. Additionally, automata-theoretic techniques are used in our presentation, which are different from the techniques used in [4, 12]. Notice that the problem is not limited only to e-services. In fact, similar automated design problems were also studied in the workflow context [24, 17] and verification communities (e.g., [5, 1, 21, 16]). In the future, we will also look at how our techniques and results can be applied to these latter problems.

In this paper, we use $A_1, ..., A_r$ to denote r activity automata (not necessary finite-state), which specify the activity behaviors of some r existing e-services. We use A to denote an activity automaton (again, not necessary finite-state), which specifies the desired activity behavior of the e-service to be composed from the existing e-services.

The first issue concerns *composability*. The system $(A; A_1, ..., A_r)$ is *composable* if for every string (or sequence) $w = a_1...a_n$ of activities accepted by A, there is an assignment (or delegation) of each symbol in w to one of the A_i's such that if w_i is the subsequence assigned to A_i, then w_i is accepted by A_i. The device that performs the composition is nondeterministic, in general. We start our discussion with $A, A_1, ..., A_r$ being restricted counter-machines (finite automata augmented with counters, each of which can be incremented/decremented by 1 and can be tested against 0). One of the restrictions we consider is when the counters are reversal-bounded [14]; i.e., for each counter, the number of alternations between nondecreasing mode and nonincreasing mode is bounded by a given constant, independent of the computation. As an example, the above mentioned release-allocate sequences can be accepted by a deterministic reversal-bounded counter-machine with 4 reversal-bounded counters. We use notations like DFAs or NFAs (deterministic or nondeterministic finite automata) and DCMs or NCMs (deterministic or nondeterministic reversal-bounded counter-machines). In [12], it was shown that composability is decidable for a system $(A; A_1, .., A_r)$ of DFAs. We generalize this result to the case when A is an NPCM (nondeterministic pushdown automaton with reversal-bounded counters) and the A_i's are DFAs. In contrast, we show that it is undecidable to determine, given DFAs A and A_1 and a DCM A_2 with only one 1-reversal counter (i.e., once the counter decrements it can no longer increment), whether $(A; A_1, A_2)$ is composable. We also look at other situations where composability is decidable. Further, we propose alternative definitions of composition (e.g., T-composability) and investigate decidability with respect to these new definitions.

When a system is composable, a composer exists but, in general, it is nondeterministic. The second issue we study concerns the existence of a deterministic delegator (i.e., a deterministic composer) within some resource bound. We adopt the notion of k-lookahead delegator (or simply k-delegator) from [12] but for infinite-state automata. (We note that [4] only studied 0-lookahead delegators.) This special form of a delegator is assumed to be efficient, since in its implementation, the delegator does not need to look back to its delegation history to decide where the current activity shall be delegated. For a nonnegative integer k, a k-delegator for $(A; A_1, ..., A_r)$ is a DCM D which, knowing (1) the current states of $A, A_1, ..., A_r$ and the signs of their counters (i.e., zero or non-zero), and (2) the k lookahead symbols (i.e., the k "future" activities) to the right of the current input symbol being processed, can deterministically determine the A_i to assign the current symbol. Moreover, every string w accepted by A is also accepted by D, i.e., the subsequence of string w delegated by D to each A_i is accepted by A_i. Clearly, if a system $(A; A_1, ..., A_r)$ has a k-delegator for some k, then it must be composable. However, the converse is not true – a system may be composable but it may not have a k-delegator for any k.

In [4], the decidability of the existence of a 0-lookahead delegator (i.e., no lookahead) when the automata (i.e., $A, A_1, ..., A_r$) are DFAs was shown to be is in EXPTIME. The concept of lookahead was introduced in [12] where the focus was still on DFAs;

algorithms for deciding composability and determining an approximate upper bound on k (if it exists) were obtained. A question left open in [12] is whether there is a decision procedure for determining for a given k, whether a system of DFAs has a k-lookahead delegator. We answer this question positively in this paper, even for the more general case when the automata are not necessarily finite-state (e.g., DCMs). Specifically, we show that it is decidable to determine, given a system $(A; A_1, ..., A_r)$ of DCMs and a nonnegative integer k, whether the system has a k-lookahead delegator.

Our results generalize to composition and lookahead delegation when we impose some linear constraints on the assignments/delegations of symbols. Doing this allows us to further specify some fairness linear constraint on a delegator. For instance, suppose that we impose a linear relationship, specified by a Presburger relation P, on the numbers and types of symbols that can be assigned to $A_1, ..., A_r$. We show that it is decidable to determine for a given k, whether a system $(A; A_1, ..., A_r)$ of DCMs has a k-delegator under constraint P. However, it is undecidable to determine, given a system $(A; A_1, A_2)$, whether it is composable under constraint P, even when A, A_1, A_2 are DFAs and P involves only the symbols assigned to A_2.

Composability and existence of k-lookahead delegators for systems consisting of other types of automata can also be defined and we study them as well. In particular, we show that composability is decidable for discrete timed automata [2].

The paper has four sections, in addition to this section. Section 2 defines (actually generalizes) the notion of composability of activity automata and proves that it is undecidable for systems $(A; A_1, A_2)$, where A, A_1 are DFAs and A_2 is a DCM with one 1-reversal counter. It is also undecidable when A, A_1, A_2 are DFAs when a Presburger constraint is imposed on the numbers and types of symbols that can be delegated to A_1 and A_2. In contrast, composability is decidable for systems $(A; A_1, ..., A_r)$ when $A_1, ..., A_r$ are DFAs (even NFAs) and A is an NPCM. Decidability holds for other restricted classes of automata as well. Section 3 introduces T-composability and shows that T-composability is decidable for various automata. Section 4 looks at the decidability of the existence for a given k of a k lookahead delegator and shows, in particular, that it is decidable to determine, given a system $(A; A_1, ..., A_r)$ of NCMs and a nonnegative integer k, whether the system has a k-delegator (even when A is an NPCM). The decidability holds, even if the delegation is under a Presburger constraint. Section 5 investigates composability of discrete timed automata. Because of space limitation, no proofs are given in this extended abstract. They will be presented in a forthcoming paper.

2 Composability

Recall that, throughout this paper, we will use the following notations: a DFA (NFA) is a deterministic (nondeterministic) finite automaton; DCM (NCM) is a DFA (NFA) augmented with reversal-bounded counters; NPCM (DPCM) is a nondeterministic (deterministic) pushdown automaton augmented with reversal-bounded counters.

Machines with reversal-bounded counters have nice decidable properties (see, e.g., [14, 15, 10]), and the languages they accept have the so-called semilinear property. They

have been useful in showing that various verification problems concerning infinite-state systems are decidable [7, 6, 8, 11, 9, 20].

Assumption: For ease in exposition, we will assume that when we are investigating the composability and k-delegability of a system $(A; A_1, ..., A_r)$ that the machines operate in real-time (i.e., they process a new input symbol at every step). The results can be generalized to machines with a one-way input tape with a right input end marker, where the input head need not move right at every step, and acceptance is when the machine eventually enters an accepting state at the right end marker. This more general model can accept fairly complex languages. For example, the language consisting of all binary strings where the number of 0's is the same as the number of 1's can be accepted by a DCM which, when given a binary input, uses two counters: one to count the 0's and the other to count the 1's. When the input head reaches the right end marker, the counters are simultaneously decremented, and the machine accepts if the two counters reach zero at the same time. Note that the DCM has two 1-reversal counters. In the constructions in proofs of the theorems, we will freely use these non-real-time models with the input end marker. It is known that nondeterministic such machines have decidable emptiness and disjointness problems but undecidable equivalence problem; however, the deterministic varieties have a decidable containment and equivalence problems [14].

Definition 1. *Let $(A; A_1, ..., A_r)$ be a system of activity automata that are DCMs over input (or activity) alphabet Σ. Assume that each DCM starts in its initial state with its counters initially zero. We say that a word (or a sequence of activities) $w = a_1a_2...a_n$ is* composable *if there is an assignment of each symbol a_i to one of the $A_1, ..., A_r$ such that if the subsequence of symbols assigned to A_i is w_i, then w_i is accepted by A_i (for $1 \leq i \leq r$). We say that the system $(A; A_1, ..., A_r)$ is composable if every word w accepted by A is composable.*

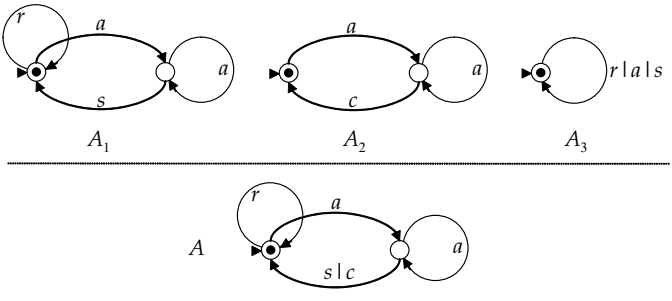

Fig. 1. Four e-Services

Example 1. Consider an online club that offers its customers to first register (represented by *r*), and then pay for their accesses (*a*) with either cash (*s*) or credit cards (*c*). The e-Service is shown as A in Figure 1, which accepts the language $(r|(aa^*(s|c)))^*$. Assume that there are three existing e-Services, A_1, A_2, and A_3, where A_1 handles registration, cash payments for one or more accesses, A_3 is similar to A_1 except that some customers

may use promotion for free accesses, and A_2 can also handle accesses and make credit card transactions. Clearly, the system $(A; A_1, A_2)$ is composable where processing of accesses will be done by whoever collects the payment, cash (A_1) or credit card (A_2).

The system $(A; A_2, A_3)$ is also composable, but in this case, the delegator need only know if the customer will make a credit card payment in the next activity; if so A_2 will perform a, otherwise A_3 does it. Thus this system has a 1-lookahead delegator (to be defined more precisely later). ∎

It is known that it is decidable whether a system $(A; A_1, ..., A_r)$ of DFAs is composable [12]. Somewhat unexpectedly, the following result says that it becomes undecidable when one of the A_i's is augmented with one 1-reversal counter.

Theorem 1. *It is undecidable to determine, given a system $(A; A_1, A_2)$, where A and A_1 are DFAs and A_2 is a DCM with one 1-reversal counter, whether it is composable.*

Remark 1. Obviously, if the machines are NCMs, composability is undecidable. In fact, take A to be the trivial machine that accepts Σ^* (the universe). Take A_1 to be an an arbitrary NCM with one 1-reversal counter. Then the system $(A; A_1)$ is composable iff Σ^* is contained in $L(A_1)$. But the latter problem is known to be undecidable [3]. However, unlike NCMs, equivalence of DCMs is decidable.

Theorem 2. *If A is an NPCM and $A_1, ..., A_r$ are DFAs (or even NFAs), then composability of $(A; A_1, ..., A_r)$ is decidable.*

It is of interest to determine the complexity of the composability problem. For example, a careful analysis of the proof of the above theorem and the use of Savitch's theorem that a nondeterministic $S(n)$ space-bounded TM can be converted to an equivalent deterministic $S^2(n)$ space-bounded TM [22], we can show the following:

Corollary 1. *Composability of a system $(A; A_1, ..., A_r)$ of NFAs can be decided in deterministic exponential space (in the sum of the sizes of the machines).*

There are other cases when composability becomes decidable, if we apply more restrictions to $A, A_1, \ldots, A_r$. A language L is bounded if $L \subseteq w_1^* ... w_k^*$ for some given k and strings $w_1, ..., w_k$ (which may not be distinct).

Theorem 3. *Composability is decidable for a system $(A; A_1, ..., A_r)$ of NCMs when A accepts a bounded language. The result holds even if A and one of the A_i's are NPCMs.*

Another restriction on the A_i's is the following. We assume that Σ_i is the input alphabet of A_i. An input symbol a is *shared* if $a \in \Sigma_i \cap \Sigma_j$ for some $i \neq j$. We say that $(A; A_1, \ldots, A_r)$ is n-composable if every word w accepted by A and containing at most n appearances of shared symbols is composable. Then we have:

Theorem 4. *The n-composability of $(A; A_1, ..., A_r)$ is decidable when A is an NPCM and each A_i is a DCM.*

For our next result, we recall the definitions of semilinear set and Presburger relation [13]. A set $R \subseteq \mathbf{N}^n$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^n$ such that $R = \{v \mid v = v_0 + a_1 v_1 + \ldots + a_t v_t,\ a_i \in \mathbf{N}\}$. The vectors v_0 (referred to as the *constant vector*) and $v_1, v_2, \ldots, v_t$ (referred to as the *periods*) are called the *generators* of the linear set R. A set $R \subseteq \mathbf{N}^n$ is *semilinear* if it is a finite union of linear sets. It is known that R is a semilinear set if and only if it is a Presburger relation (i.e., can be specified by a Presburger formula).

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. For each string w in Σ^*, define the *Parikh map* of w to be $\psi(w) = (num_{a_1}(w), ..., num_{a_n}(w))$, where $num_{a_i}(x)$ is the number of occurrences of a_i in w. For a language $L \subseteq \Sigma^*$, the *Parikh map* of L is $\psi(L) = \{\psi(w) \mid w \in L\}$.

Let $A, A_1, ..., A_r$ is a system of DFAs over input alphabet Σ, and P be a Presburger relation (semilinear set). Suppose that we want to check whether the system is composable under constraint P on the numbers and types of symbols that are assigned/delegated to the A_i's. The constraint is useful in specifying a fairness constraint over the delegations (e.g., it is never true that the absolute value of the difference between the number of activities a assigned to A_1 and the number of activities a assigned to A_2 is larger than 10). Let $\Sigma = \{a_1, ..., a_n\}$ and P be a Presburger relation (formula) over $(r+1)n$ nonnegative integer variables (note that n is the cardinality of Σ and $r+1$ is the number of the DFAs, including A). The P-composability problem might take the the following form:

Presburger-Constrained Composability Problem: Given a system $(A; A_1, ..., A_r)$ of DFAs, is the system composable subject to the constraint that for every string $w \in L(A)$, there is an assignment of the symbols in w such that if $w_1, ..., w_r$ are the subsequences assigned to $A_1, ..., A_r$, respectively,

(1) A_i accepts w_i $(1 \leq i \leq r)$.
(2) $(\psi(w), \psi(w_1), ..., \psi(w_r))$ satisfies the Presburger relation P.

Unfortunately, because of Theorem 1, the above problem is undecidable:

Corollary 2. *The Presburger-constrained composability problem is undecidable for systems $(A;\ A_1, A_2)$ of DFAs and a Presburger formula P (even if the formula only involves symbols assigned to A_2).*

3 *T*-Composability

From the above results, it seems difficult to obtain decidable composability for a system $(A; A_1, ..., A_r)$ when one or more of $A_1, \ldots, A_r$ are beyond DFAs. Below, we will apply more restrictions on how $A_1, \ldots, A_r$ are going to be composed such that a decidable composability can be obtained. We define a mapping $T : \Sigma \rightarrow 2^{\{1,\ldots,r\}}$ such that each symbol $a \in \Sigma$ is associated with a type $T(a) \subseteq \{1, \ldots, r\}$. For $a \in \Sigma$ and $1 \leq i \leq r$, let $(a)_i = a$ if $i \in T(a)$ and $(a)_i = \epsilon$ (the null string) if $i \notin T(a)$. For a string $w = a_1 \ldots a_n$, we use $(w)_i$ to denote the result of $(a_1)_i \ldots (a_n)_i$. For each A_i, its input alphabet Σ_i consists of all a's with $i \in T(a)$. Therefore, $(w)_i$ is the result of projecting

w under the alphabet of A_i. We now modify the definition of composability as follows. $(A; A_1, \ldots, A_r)$ is T-composable if, for every string w accepted by A, each $(w)_i$ is accepted by A_i. Notice that this definition is different from the original one in the sense that every symbol a in w is assigned to *each* A_i with $i \in T(a)$. Therefore, assignments of symbols in w is deterministic in the new definition (there is a unique way to assign every symbol). One can show:

Theorem 5. *The T-composability of $(A; A_1, \ldots, A_r)$ is decidable in the following cases:*

- *A is an NPCM and each A_i is a DCM;*
- *A is an NCM and each A_i is a DPCM.*

Theorem 5 does not generalize to the case when one of the A_i's is an NCM, for the same reason as we stated in Remark 1.

We may take another view of the composition of $A_1, \ldots, A_r$. As we have mentioned earlier, each activity automaton A_i is understood as the behavior specification of an e-service. Each sequence w_i of activities accepted by A_i is an allowable behavior of the service. In the original definition of composability, the activity automata $A_1, \ldots, A_r$ are composed through interleavings between the activities in the sequences $w_1, \ldots, w_r$. Clearly, if activities between two services are disjoint, the original definition of composability becomes T-composability with $T(a)$ being a singleton set for every symbol a (i.e., each activity a belongs to a unique activity automaton). When the activity automata share some common activities (e.g., a belongs to both A_1 and A_2; i.e., $T(a) = \{1, 2\}$), the T-composability definition implies that an a-activity in A_1 must be synchronized with an a-activity in A_2. This is why in T-composability, such a symbol a must be assigned to both A_1 and A_2. Notice that the assignments of each symbol (activity) is deterministic in T-composability. The determinism helps us generalize the above theorem as follows.

A *reset-NCM* M is an NCM that is equipped with a number of *reset states* and is further augmented with a number of *reset counters* (in addition to the reversal-bounded counters). The reset counters are all reset to 0 whenever M enters a reset state. (As usual, we assume that initially the counters start with 0, i.e., with a reset state) We further require that on any execution, the reset counters are reversal-bounded between any two resets. One may similarly define a *reset-NPCM*. Notice that an NCM (resp. NPCM) is a special case of a reset-NCM (resp. reset-NPCM) where there is no reset counter.

Theorem 6. *The emptiness problem for reset-NCMs is decidable.*

We use reset-NPM to denote a reset-NPCM that contains only reset counters and a stack. One can show that the emptiness of reset-NPMs is undecidable.

Theorem 7. *The emptiness problem for reset-NPMs and hence reset-NPCMs is undecidable.*

Now, we generalize Theorem 5 as follows.

Theorem 8. *T-composability of $(A; A_1, \ldots, A_r)$ is decidable when A is an NCM and each A_i is a reset-DCM.*

Let NPDA (DPDA) denote a nondeterministic (deterministic) pushdown automaton. Thus, an NPDA is a special case of a reset-NPM, one that does not have reset counters. Using Theorem 7, one can show,

Theorem 9. *T-composability of $(A; A_1, ..., A_r)$ is undecidable when A is a DPDA and each A_i is a reset-DCM, even for the case when $r = 1$.*

4 Lookahead Delegator

Given k, a k-lookahead delegator (or simply k-delegator) for the system of NCMs $(A; A_1, ..., A_r)$ is a DCM D which, knowing the current states of $A, A_1, ..., A_r$ and the statuses (i.e., signs) of their counters (i.e., zero or non-zero), and the k lookahead symbols to the right of the current input symbol being processed, D can uniquely determine the transition of A, the assignment of the current symbol to one of $A_1, ..., A_r$, and the transition of the assigned machine. Moreover, for every string x accepted by A, D also accepts, i.e., the subsequence of string x delegated by D to each A_i is accepted by A_i. Clearly, if a system has a k-delegator (for some k), then it must be composable. However, the converse is not true, in general. For example, the system in Figure 1(a) is composable, but it does not have a k-delegator for any k.

Example 2. Consider again Example 1 and in particular the system $(A; A_1, A_2)$. It is easy to see that all a activities immediately preceding an s or c has to be delegated to A_1 or A_2, respectively. Without knowing which letter, s or c, will be coming, the delegator cannot correctly determine whether A_1 or A_2 should perform the activities a. Thus, the system has no k-delegator for any k. On the other hand, the system $(A; A_2, A_3)$ has a 1-delegator. It is straightforward to generalize this example (by adding additional states) to show that for every k, there exists a system that has a $(k + 1)$-delegator but not a k-delegator. ∎

So that we can always have k lookahead, let \$ be a new symbol and f be a new state. Extend the transition function of A by defining the transition from any state, including f, on symbol \$ to f. Then make f the only (unique) accepting state. Thus the new NCM accepts the language $L(A)\$^+$ and it has only one accepting state f. We can do the same thing for $A_1, ..., A_r$ with $f_1, ..., f_r$ their unique accepting states. For convenience, call the new machines also $A, A_1, ..., A_r$.

For ease in exposition, in what follows, we assume that $r = 2$, and each of A, A_1, A_2 has only one reversal-bounded counter. Generalizations to any $r \geq 2$ and machines having multiple reversal-bounded counters is straightforward. Note that the transition of A has the form: $\delta_A(q, a, s) = \{..., (p, d), ...\}$, which means that if A in in state q and the input is a and the sign of its counter is s (zero or non-zero), then A can change state to p and increment the counter by d where $d = 0, \ +1, \ -1$, with the constraint that if $s = 0$, then $d = 0, \ +1$. The same holds for transitions δ_1 and δ_2 of A_1 and A_2. We assume that the counters are initially zero.

Let k be a nonnegative integer. We can construct a candidate k-delegator DCM D as follows: each state of D is a tuple (q, p_1, p_2, u), where q is a state of A, p_i is a state of A_i, and u is a string of length k. However, in the case (q^0, p_1^0, p_2^0, u), where q_0 is the initial

state of A and p_i^0 the initial state of A_i, the length of u can be less that k, including zero length, in which case $u = \epsilon$. Then the initial state of D is $(q^0, p_1^0, p_2^0, \epsilon)$. The transition δ of D is defined as follows:

1. $\delta((q^0, p_1^0, p_2^0, \epsilon), 0, 0, 0, a) = ((q^0, p_1^0, p_2^0, a), 0, 0, 0)$ for all symbol a.
2. $\delta((q^0, p_1^0, p_2^0, v), 0, 0, 0, a) = ((q^0, p_1^0, p_2^0, va), 0, 0, 0)$ for all string v such that $|v| < k$ and symbol a.
3. $\delta((q, p_1, p_2, av), s, s_1, s_2, b) = ((q', p_1', p_2', vb), d, d_1, d_2)$ for all q, p_1, p_2, s, s_1, s_2, all string v such that $|v| = k$ and symbols a, b, where:
   (a) $(q', d) \in \delta_A(q, a, s)$;
   (b) either $p_1' = p_1$, $d_1 = 0$, and $(p_2', d_2) \in \delta_2(p_2, a, s_2)$ or $p_2' = p_2$, $d_2 = 0$, and $(p_1', d_1) \in \delta_1(p_1, a, s_1)$.

   Moreover, the choice $((q', p_1', p_2'), d, d_1, d_2)$ once made is unique for the parameters $((q, p_1, p_2, av), s, s_1, s_2)$. (Note that, in general, there are many choices that can be made for the given parameters.)
4. Note that in (q, p_1, p_2, u), any suffix of u may be a string of \$'s.
5. Then $(f, f_1, f_2, \$^k)$ is the accepting state of D, where f, f_1, f_2 are the unique accepting states of A, A_1, A_2.

Now D is a DCM. Since the class of languages accepted by DCMs is effectively closed under complementation, we can construct a DCM E accepting the complement of $L(D)$. Then D is a k-delegator of $(A; A_1, A_2)$ iff $L(A) \cap L(E) = \emptyset$. We can construct from NCM A and DCM E an NCM F accepting $L(A) \cap L(E)$. We can then check the emptiness of $L(F)$ since the emptiness problem for NCMs is decidable. Now D is just one candidate for a k-delegator. There are finitely many such candidates. Every choice that can be made in item 3) above corresponds to one such candidate. By exhaustively checking all candidates, we either find a desired k-delegator or determine that no such k-delegator exists. Thus, we have shown the following:

Theorem 10. *It is decidable to determine, given a system of NCMs $(A; A_1, ..., A_r)$ and a nonnegative integer k, whether the system has a k-delegator.*

Since the emptiness problem for NPCMs is also decidable, we can generalize the above result to:

Corollary 3. *It is decidable to determine, given a system $(A; A_1, ..., A_r)$, where A is an NPCM and $A_1, ..., A_r$ are NCMs, and a nonnegative integer k, whether the system has a k-delegator.*

Corollary 4. *If we impose some Presburger constraint P on the delegation of symbols by the k-delegator (e.g., some linear relationships on the number of symbols delegated to $A_1, ..., A_r$), then the existence of such a P-constrained k-delegator is also decidable.*

Open Question: Is it decidable to determine, given a system of DCMs $(A, A_1, ..., A_r)$, whether it has a k-delegator for some k?

Corollary 5. *It is decidable to determine, given a system $(A; A_1, ..., A_r)$ and a nonnegative integer k, where A is a DPDA (deterministic pushdown automaton), A_1 is a PDA (nondeterministic pushdown automaton) and $A_2, ..., A_r$ are NFAs, whether the system has a DPDA k-delegator. (Here, the delegation depends also on the top of the stack of A_1.)*

For the special case when the machines are NFAs, we can prove the following (from the proof of Theorem 10 and Savitch's theorem):

Corollary 6. *We can decide, given a system $(A; A_1, ..., A_r)$ of NFAs and a nonnegative integer k, whether the system has a k-delegator in nondeterministic exponential time (in k and the sum of the sizes of the machines) and hence, also, in deterministic exponential space.*

5 Composability of Timed Automata

In this section, we study composability of discrete timed automata (DTA) A, which are NFAs augmented with discrete-valued clocks [2]. We say that a word w is *accepted* by A when w is provided on the input tape, if A is able to enter a designated accepting state. We use $L(A)$ to denote the set of words accepted by A. For DTAs, one may develop a similar definition of composability as in Section 2. However, the definition does not justify the intended meaning of composability. For instance, let A_1 and A_2 be two DTAs, and suppose ac (resp. bd) are accepted by A_1 (resp. A_2). Observe that an interleaving like $abcd$ of the two words is not necessarily accepted by the DTA composed from A_1 and A_2. This is because, when composing, A_1 and A_2 share the same global clock. To devise a proper definition of composability for DTAs, we introduce timed words [2]. A timed word is a sequence of pairs

$$(a_1, t_1) \dots (a_n, t_n) \tag{1}$$

such that each $a_i \in \Sigma, t_i \in \mathbf{N}^+$, and $t_1 \leq \dots \leq t_n$. We say that the timed word is *accepted* by A if $w = a_1 \dots a_n$ is accepted by A and this fact is witnessed by some accepting run of A such that each t_i is the timestamp (the value of the global clock) when symbol a_i is read in the run. Thus, the timed word not only records the sequence of symbols $a_1 \dots a_n$ accepted by A but also remembers the timestamp when each symbol is read. Let $A, A_1, \dots, A_r$ be DTAs. A timed word in the form of (1) is *timed composable* if there is an assignment of each pair (a_j, t_j) to one of the $A_1, ..., A_r$ such that, for $1 \leq i \leq r$, the subsequence (also a timed word) of pairs assigned to A_i is accepted by A_i. We say that $(A; A_1, \dots, A_r)$ is timed composable if every timed word accepted by A is timed composable. The main result of this section is the following:

Theorem 11. *The timed composability of discrete timed automata $(A; A_1, \dots, A_r)$ is decidable.*

References

1. M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable specifications of reactive systems. In *Proc. 16th Int. Colloq. on Automata, Languages and Programming*, 1989.

2. R. Alur and D. Dill. Automata for modeling real-time systems. *Theoretical Computer Science*, 126(2):183–236, 1994.
3. B. Baker and R. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8:315–332, 1974.
4. D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *Proc. 1st Int. Conf. on Service Oriented Computing (ICSOC)*, volume 2910 of *LNCS*, pages 43–58, 2003.
5. J. Buchi and L. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
6. Z. Dang. Pushdown time automata: a binary reachability characterization and safety verification. *Theoretical Computer Science*, 302:93–121, 2003.
7. Z. Dang, O. Ibarra, T. Bultan, R. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *Proc. Int. Conf. on Computer-Aided Verification (CAV)*, pages 69–84, 2000.
8. Z. Dang, O. H. Ibarra, and R. A. Kemmerer. Generalized discrete timed automata: decidable approximations for safety verification. *Theoretical Computer Science*, 296:59–74, 2003.
9. Z. Dang, O. H. Ibarra, and P. San Pietro. Liveness Verification of Reversal-bounded Multicounter Machines with a Free Counter. In *FSTTCS'01*, volume 2245 of *Lecture Notes in Computer Science*, pages 132–143. Springer, 2001.
10. Z. Dang, O. H. Ibarra, and Z. Sun. On the emptiness problems for two-way nondeterministic finite automata with one reversal-bounded counter. In *ISAAC'02*, volume 2518 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2002.
11. Z. Dang, P. San Pietro, and R. A. Kemmerer. Presburger liveness verification for discrete timed automata. *Theoretical Computer Science*, 299:413–438, 2003.
12. C. E. Gerede, R. Hull, and J. Su. Automated composition of e-services with lookahead. Technical report, UCSB, 2004.
13. S. Ginsburg and E. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. of Mathematics*, 16:285–296, 1966.
14. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, January 1978.
15. O. H. Ibarra, T. Jiang, N. Tran, and H. Wang. New decidability results concerning two-way counter machines. *SIAM J. Comput.*, 24:123–137, 1995.
16. O. Kupferman and M. Y. Vardi. Synthesizing distributed systems. In *Proc. IEEE Symposium on Logic In Computer Science*, 2001.
17. S. Lu. *Semantic Correctness of Transactions and Workflows*. PhD thesis, SUNY at Stony Brook, 2002.
18. N. Lynch and M. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc. 6th ACM Symp. Principles of Distributed Computing*, pages 137–151, 1987.
19. M. Papazoglou. Agent-oriented technology in support of e-business. *Communications of the ACM*, 44(4):71–77, 2001.
20. P. San Pietro and Z. Dang. Automatic verification of multi-queue discrete timed automata. In *COCOON'03*, volume 2697 of *Lecture Notes in Computer Science*, pages 159–171. Springer, 2003.
21. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1990.
22. W. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.
23. M. Singh. Semantical considerations on workflows: An algebra for intertask dependencies. In *Proc. Workshop on Database Programming Languages (DBPL)*, 1995.
24. W. M. P. van der Aalst. On the automatic generation of workflow processes based on product structures. *Computer in Industry*, 39(2):97–111, 1999.