Information and Computation 206 (2008) 1161–1177

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Verification of scope-dependent hierarchical state machines☆

Salvatore La Torre*, Margherita Napoli, Mimmo Parente, Gennaro Parlato

Dipartimento di Informatica e Applicazioni, Università degli Studi di Salerno, 84084 Fisciano, SA, Italy

ARTICLE INFO

Article history:
Received 1 July 2007
Revised 12 February 2008
Available online 5 June 2008

Keywords:
Hierarchical state machines
Model checking
Automata
Temporal logic

ABSTRACT

A hierarchical state machine (HSM) is a finite state machine where a vertex can either expand to another hierarchical state machine (*box*) or be a basic vertex (*node*). Each node is labeled with atomic propositions. We study an extension of such model which allows atomic propositions to label also boxes (SHSM). We show that SHSMs can be exponentially more succinct than SHSMs and verification is in general harder by an exponential factor. We carefully establish the computational complexity of reachability, cycle detection, and model checking against general LTL and CTL specifications. We also discuss some natural and interesting restrictions of the considered problems for which we can prove that SHSMs can be verified as much efficiently as HSMs, still preserving an exponential gap of succinctness.

1. Introduction

Finite state machines (labeled finite transition systems) are widely used for modeling the flow of control of digital systems and are appealing to formal verification such as model checking [1,2]. In model checking, a high-level specification is expressed by a formula of a logic and is checked for fulfillment on an abstract model of the system. Though a typical solution to this problem is linear in the size of the model, it is computationally hard since the model generally grows exponentially with the number of variables which are used to describe the system (*state-space explosion*). As a consequence, an important part of the research on model checking has been concerned with handling this problem.

Complex systems are usually composed of relatively simple modules in a hierarchical manner, and hierarchical structures are also typical of object-oriented paradigms [3,4,5]. A *hierarchical finite state machine* (HSM) [6], is composed of several finite state machines where a vertex can either expand to another hierarchical state machine (*box*) or be a basic vertex (*node*). Each node is labeled with atomic propositions (*AP*) and the outcomes of the model thus generate sequences over 2^{AP}. The complexity of model checking for HSMs is discussed in [6] and the succinctness of such models compared to standard finite state machines is addressed in [7].

In this paper, we consider a variation of the hierarchical state machines where also boxes are labeled with atomic propositions. The intended meaning of such labeling is that when a box b expands to a machine M, all the vertices of M *inherit* the atomic propositions of b (*scope*), such that different vertices expanding to M can place M into different scopes. For this reason, we call such model a *hierarchical state machine with scope-dependent properties* (scope-dependent hierarchical

☆ Work partially supported by Università di Salerno Grant ex-60%. A preliminary version of the results reported in this paper appears in "Hierarchical and Recursive State Machines with Context-Dependent Properties", Proceedings of the ICALP 2003, LNCS 2719, pp. 776–789, and in "Verification of Succinct Hierarchical State Machines", Proceedings of the LATA 2007, Report of University Rovira I Virgili in Terragona, Spain, pp. 485–496.

* Corresponding author. Fax: +39 089 969 600.
E-mail addresses: latorre@dia.unisa.it (S. La Torre), napoli@dia.unisa.it (M. Napoli), parente@dia.unisa.it (M. Parente), parlato@dia.unisa.it (G. Parlato).

0890-5401/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.03.017

state machine, shortly SHSM).[1] We show that, by allowing this more general labeling, it is possible to obtain models of systems that are exponentially more succinct than HSMs. As an example, consider a digital clock with hours, minutes, and seconds. We can construct a hierarchical state machine $\mathcal{M}$ composed of a sequence of three machines M_1, M_2, and M_3 such that the boxes of M_3 expands to M_2 and the boxes of M_2 expands to M_1. In M_3, each box corresponds to a hour and boxes are linked accordingly to increasing time. Analogously, M_2 models minutes and M_1 seconds. The flat model for such digital clock has $24 \cdot 60 \cdot 60 = 86{,}400$ vertices, while $\mathcal{M}$ has only $24 + 60 + 60 + 6 = 150$ vertices (6 are simply entry and exit nodes). If we are interested in checking properties that refer to a precise time (for example, time 10 : 20 : 20), we can show that using HSMs we would need at least 86,400 nodes, that is, no gain with respect to the flat model. In our model instead, we are able to label each box in M_3 with atomic propositions from a set P_h each encoding the corresponding hour. Analogously we can use atomic propositions P_m and P_s to encode minutes and seconds in M_2 and M_1, respectively. This way, each state of $\mathcal{M}$ is labeled with the hour, minute, and second of a precise time in a day and transitions in $\mathcal{M}$ are forced to visit states by increasing times.

We study the complexity of verification on SHSMs. In particular, we consider basic verification questions such as reachability and cycle detection, and the model checking problem against general LTL [9] and CTL [1] specifications. We show that for an SHSM $\mathcal{M}$ and a formula φ: LTL model checking is PSPACE-complete and can be solved in $O(|\mathcal{M}|\, 16^{|\varphi|})$ time; CTL model checking is EXPTIME-complete and can be solved in $O(|\mathcal{M}|\, 2^{|\varphi|\,(d+1)})$ time, where d is the maximum number of exit nodes of $\mathcal{M}$. We also show that reachability and cycle detection are both NP-complete. According to the results shown in [6] for HSMs, we get time complexities increased by an $O(2^{|\varphi|})$ factor which is exactly what we gain in succinctness. Concerning to the reachability and cycle detection problems, we show that if the evaluation of the boolean formula expressing the set of target states is somehow consistent with the hierarchic structure of the given SHSM then we can show a linear-time upper bound for both problems.

A natural restriction on the definition of SHSMs is to require that an atomic proposition which labels a box b cannot label the vertices of any of the machines which directly or indirectly expand from b. This is the case in many concrete models where properties are local to modules, such as in the case of the above clock example. We show that these *restricted* SHSMs are exponentially less succinct than SHSMs and exponentially more succinct than HSMs. We also show that though the complexity of verification does not substantially improve for restricted SHSMs, LTL model checking can be solved in $O(|\mathcal{M}|\, 8^{|\varphi|})$ time, i.e., it has the same upper bound as for HSMs [6]. On the other side reachability and cycle detection problems still remain NP-complete. However, we discuss two conditions each ensuring a linear-time upper bound for such problems. In particular, we show that conjunctions of literals are always consistent with the structure of a restricted SHSM, and therefore, we can prove the claimed upper bound for formulas in disjunctive normal form and restricted SHSMs. It is worth noticing that this result cannot be extended to general SHSMs, since we show that both problems are NP-hard on such models even if we restrict to target sets expressed as conjunctions of literals. Moreover, we get the claimed bound also if the boolean formula expressing the target set is a conjunction of formulas which are "locally valuable" on the vertices of the restricted SHSM.

There are several papers in the literature that have concerned with hierarchical state machines. In [10,11], the verification tool HERMES which is based on hierarchical state machines is discussed. The extension of hierarchical state machines with recursive expansions of nodes (state machines with recursive calls) are studied in [12] and a corresponding temporal logic is introduced in [13]. Recursive state machines turn out to be equivalent to pushdown automata [12]. Recursive calls and scope-dependent properties have been considered in [8]. The impact of concurrency is studied in [7,14] for hierarchical state machines and in [15] for recursive state machines. Finally, modular control synthesis for recursive state machines is studied in [16,17].

The rest of the paper is organized as follows. In Section 2, we give the definitions and introduce our notation. In Section 3, we show the results on the succinctness of (restricted) SHSMs, and compare them to hierarchical and finite state machines. In Section 4, we study the complexity of reachability and cycle detection on the considered models. Model checking against LTL and CTL specifications is addressed in Section 5. We conclude the paper with a discussion on some possible extensions of the model and our final remarks in Section 6.

2. The model

"Model checking" is used to verify properties of an abstract model of a given system. Various kinds of models have been proposed in the literature and the basic one is the so-called *Kripke structure*, a finite *state-transition graph* whose states are labeled with atomic propositions. Formally, given a set AP of atomic propositions, a Kripke structure over AP is a tuple (S,in,R,L), where S is a finite set of *states*, $in \in S$ is the *initial state*, $R \subseteq S \times S$ is the set of *transitions* and $L : S \longrightarrow 2^{AP}$ is the *labeling function* which maps each state s to a set of atomic propositions, with the meaning that $L(s)$ is the set of all the atomic propositions that hold true at s.

In this paper, we model a system by a hierarchically structured graph. In such model, vertices can be either simple nodes or placeholders for other graphs. The use of such placeholders enables us to represent repeated subgraphs only once and thus obtain models that are more succinct than equivalent flat Kripke structures. An example of a hierarchically structured

[1] Note that the term "scope" is used here with the same meaning as "context" was used in [8]. This change is due to avoid conflicts with other known acronyms.

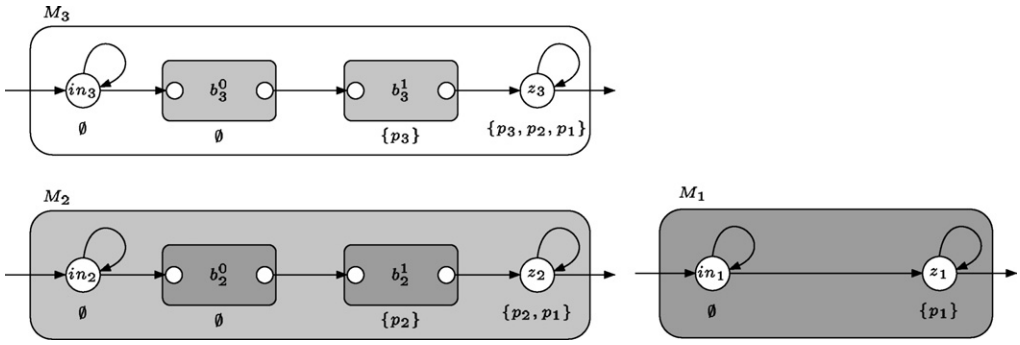

Fig. 1. A simple SHSM $\mathcal{M}$.

graph $\mathcal{M}$ is given in Fig. 1, and the corresponding Kripke structure $\mathcal{M}^F$ is given in Fig. 2 where b_3^0 and b_3^1 are placeholders for M_2 and b_2^0 and b_2^1 for M_1.

In Sections 2.1 and 2.2 we, respectively, give the syntax and the semantics of the model.

2.1. The syntax of the model

Here, we formally define the scope-dependent HSMS.

Definition 1. A scope-dependent hierarchical state machine (SHSM) over AP is a tuple $\mathcal{M} = (M_1, M_2, \ldots, M_k)$, each $M_i = (V_i, in_i, \text{OUT}_i, trueAP_i, expn_i, E_i)$ is called *machine* and consists of:

- a finite set of vertices V_i, an *initial* vertex $in_i \in V_i$ and a set of *output* vertices $\text{OUT}_i \subseteq V_i$;
- a labeling function $trueAP_i : V_i \longrightarrow 2^{AP}$ that maps each vertex with a set of atomic propositions;
- an expansion mapping $expn_i : V_i \longrightarrow \{0,1,\ldots,k\}$ such that $expn_i(u) < i$, for each $u \in V_i$, and $expn_i(u) = 0$, for each $u \in \{in_i\} \cup \text{OUT}_i$;
- a set of edges E_i where each edge is either a couple (u,v), with $u,v \in V_i$ and $expn_i(u) = 0$, or a triple $((u,z),v)$ with $u,v \in V_i$, $expn_i(u) = j, j > 0$, and $z \in \text{OUT}_j$,

In the rest of the paper, we use k as the number of machines of an SHSM $\mathcal{M}$ and M_k is called *top-level* machine.

We assume that the sets of vertices V_i are pairwise disjoint. The set of all vertices of $\mathcal{M}$ is $V = \bigcup_{i=1}^{k} V_i$. The mappings $expn : V \longrightarrow \{0,1,\ldots,k\}$ and $trueAP : V \longrightarrow 2^{AP}$ extend the mappings $expn_i$ and $trueAP_i$, respectively. If $expn(u) = j > 0$, the vertex u expands to the machine M_j and is called *box*. When $expn(u) = 0$, u is called a *node*. Let us define the closure $expn^+ : V \longrightarrow 2^{\{0,1,\ldots,k\}}$, as: $h \in expn^+(u)$ if either $h = expn(u)$ or there exists $u' \in V_{expn(u)}$ such that $h \in expn^+(u')$. We say that a vertex u is an *ancestor* of v and v is a *descendant* from u if $v \in V_h$, for $h \in expn^+(u)$.

As an example of an SHSM $\mathcal{M}$ see Fig. 1, where p_1,p_2,p_3 are atomic propositions labeling nodes and boxes of $\mathcal{M}$, in_i and z_i are, respectively, entry nodes and exit nodes for $i = 1,2,3$, and $expn(b_j^i) = j - 1$ for $i = 0,1$ and $j = 2,3$.

We present now a class of SHSMS on which it is possible to give more efficient algorithms for solving model checking.

Definition 2. A restricted SHSM $\mathcal{M}$ is an SHSM where for all vertices u,v such that u is an ancestor of v in $\mathcal{M}$ it holds that $trueAP(u) \cap trueAP(v) = \emptyset$.

Such a restriction is quite natural and still allows us to succinctly represent interesting systems. Note that the SHSM of Fig. 1 is also restricted.

2.2. The semantics of the model

The semantics of an SHSM $\mathcal{M}$, and thus of a restricted SHSM, is given by defining an equivalent Kripke structure denoted $\mathcal{M}^F$.

A sequence of vertices $\alpha = u_1 \ldots u_m$, $1 \leqslant m$, is called a *well-formed sequence* if $u_{\ell+1} \in V_{expn(u_\ell)}$, for $\ell = 1,\ldots,m-1$. Moreover, α is also *complete* when $u_1 \in V_k$ and u_m is a node.

A state of $\mathcal{M}^F$ is $\langle\alpha\rangle$ where α is a complete well-formed sequence of $\mathcal{M}$. Note that the length of a complete well-formed sequence is at most k, therefore the number of states of $\mathcal{M}^F$ is at most exponential in the number of machines composing $\mathcal{M}$.

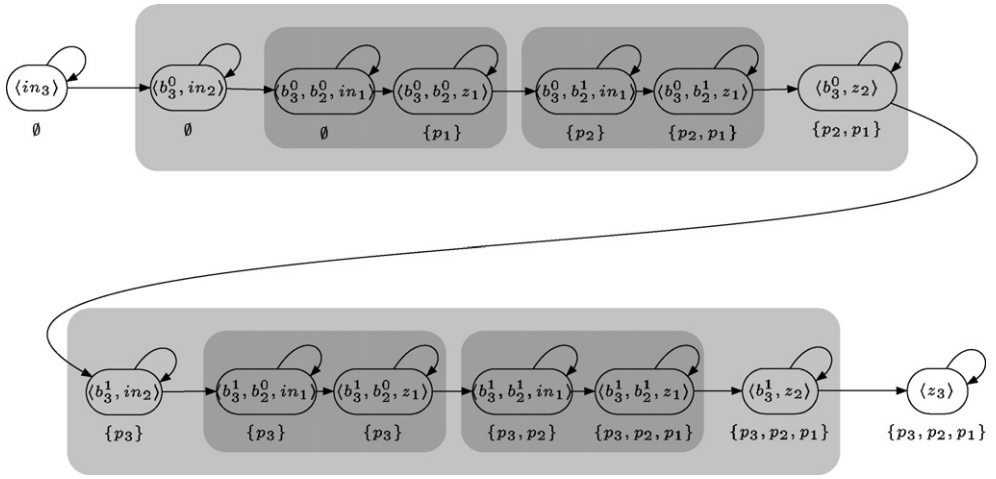

Fig. 2. The Kripke structure obtained by flattening the SHSM $\mathcal{M}$ of Fig. 1.

Transitions of $\mathcal{M}^F$ are obtained by using as templates the edges of $\mathcal{M}$. Fig. 2 shows the Kripke structure which is equivalent to the SHSM of Fig. 1. We formally define $\mathcal{M}^F$ as follows.

Definition 3. Given an SHSM $\mathcal{M} = (M_1,M_2,\ldots,M_k)$, the corresponding flat Kripke structure $\mathcal{M}^F$ is defined as:

- The states of $\mathcal{M}^F$ are $\langle u_1 \ldots u_m\rangle$, for $1 \leqslant m \leqslant k$, where $u_1u_2 \ldots u_m$ is a complete well-formed sequence.
- The initial state of $\mathcal{M}^F$ is $\langle in_k\rangle$, where in_k is the initial vertex of M_k (the top-level machine of $\mathcal{M}$).
- If $X = \langle u_1 \ldots u_m\rangle$ and $Y = \langle v_1 \ldots v_n\rangle$ are states, then (X,Y) is a transition of $\mathcal{M}^F$ if there is an edge $e \in E_i$, $1 \leqslant i \leqslant k$, such that one of the following cases occurs:
  - $n = m$, $v_j = u_j$, for $1 \leqslant j < n$, and $e = (u_m,v_n)$, that is the edge connects two nodes;
  - $m = n-1$, $v_j = u_j$, for $1 \leqslant j < n-1$, $e = (u_m,v_{n-1})$, and $v_n = in_{expn(v_{n-1})}$, that is the edge connects a node u_m to a box v_{n-1} and v_n is the initial vertex of the machine which v_{n-1} expands to;
  - $m-1 = n$, $v_j = u_j$, for $1 \leqslant j < n$, and $e = ((u_{m-1},u_m),v_n)$, that is the edge connects a box u_{m-1} to a node v_n, through the output vertex $u_m \in \text{OUT}_{expn(u_{m-1})}$;
  - $n = m$, $v_j = u_j$, for $1 \leqslant j < n-1$, $e = ((u_{m-1},u_m),v_{n-1})$, $u_m \in \text{OUT}_{expn(u_{m-1})}$, and $v_n = in_{expn(v_{n-1})}$, that is the edge connects two boxes.
- The labeling of $\mathcal{M}^F$ is such that a state $X = \langle u_1 \ldots u_m\rangle$ is labeled by the set of atomic propositions $trueAP(X) = \cup_{j=1}^{m} trueAP(u_j)$.

An alternative recursive definition of $\mathcal{M}^F$. Given an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$, it is immediate to observe that the tuple $\mathcal{M}_h = (M_1,M_2,\ldots,M_h)$, $1 \leqslant h \leqslant k$, is an SHSM as well. Clearly, $\mathcal{M}_k = \mathcal{M}$. In the following, we sketch how to compute recursively the flat Kripke structures $\mathcal{M}_h^F$.

We start with $\mathcal{M}_1^F$ which is obtained from machine M_1 by simply replacing each vertex u with a state $\langle u\rangle$ labeled with $trueAP(\langle u\rangle) = trueAP(u)$ (recall that by definition all vertices of $\mathcal{M}_1$ are nodes). Thus, for each edge $(v,w) \in E_1$ we add a transition $(\langle v\rangle,\langle w\rangle)$ in $\mathcal{M}_1^F$.

For $h > 1$, $\mathcal{M}_h^F$ is obtained from M_h by simply replacing each box u of M_h with a copy of the Kripke structure $\mathcal{M}_{expn(u)}^F$. More precisely, for each node $u \in V_h$, $\langle u\rangle$ is a state of $\mathcal{M}_h^F$ which is labeled with $trueAP(u)$ and for each box $u \in V_h$ and state $\langle\alpha\rangle$ of $\mathcal{M}_{expn(u)}^F$, $\langle u\alpha\rangle$ is a state of $\mathcal{M}_h^F$ and is labeled with $trueAP(u) \cup trueAP(\langle\alpha\rangle)$. The transitions of $\mathcal{M}_{expn(u)}^F$ are all inherited in $\mathcal{M}_h^F$, that is, there is a transition $(\langle u\alpha\rangle,\langle u\beta\rangle)$ in $\mathcal{M}_h^F$ for each transition $(\langle\alpha\rangle,\langle\beta\rangle)$ of $\mathcal{M}_{expn(u)}^F$. The remaining transitions of $\mathcal{M}_h^F$ correspond to the edges of M_h:

- for each node $v \in V_h$ and edge $(u,v) \in E_h$ (respectively, $((u,z),v) \in E_h$) there is a transition from $\langle u\rangle$ (respectively, $\langle uz\rangle$) to $\langle v\rangle$;
- for each box $v \in V_h$ and edge $(u,v) \in E_h$ (respectively, $((u,z),v) \in E_h$) there is a transition from $\langle u\rangle$ (respectively, $\langle uz\rangle$) to $\langle v\, in_{expn(v)}\rangle$.

A box u expanding into M_h is a placeholder for $\mathcal{M}_h^F$ and determines a subgraph in $\mathcal{M}^F$ isomorphic to $\mathcal{M}_h^F$. This is emphasized in Fig. 2, where we have enclosed in shades of the same shape and color the isomorphic subgraphs corresponding to a same graph $\mathcal{M}_h^F$. Therefore, Fig. 2 also illustrates the recursive definition of $\mathcal{M}^F$.

If two distinct boxes u_1 and u_2 both expand into the same machine M_h, that is $expn(u_1) = expn(u_2) = h$, then the states of $\mathcal{M}_h^F$ appear in $\mathcal{M}^F$ in two different scopes, possibly labeled with different sets of atomic propositions: in one scope this set contains $trueAP(u_1)$ and in the other it contains $trueAP(u_2)$. The atomic propositions labeling boxes represent *scope-properties*. In fact, for a given box u, the set $trueAP(u)$ of atomic propositions is meant to hold true at u and at all its possible descendants. Let us note that contrarily to what happens in Kripke structures, in this model the atomic propositions which do not label u are not necessarily to be intended *false* (in Section 4.3.1, we define the function $falseAP(u)$ of the atomic propositions which can be stated *false* at u).

3. Succinctness of the model

The possibility of representing with a single machine M_h more than one subgraph of $\mathcal{M}^F$ makes our model in general more succinct than a traditional Kripke structure. Scope properties make this model possibly even more succinct than the hierarchical state machine introduced by [6]. We recall that a *hierarchical state machine* (Hsm) is an Shsm with $trueAP(u) = \emptyset$, for every box u. In fact, two isomorphic subgraphs of a Kripke structure which differ only on the labeling of the vertices can be represented in an Shsm by the single machine M_h, while it should be represented by two different machines in a Hsm. Before stating this more formally, we need some notation.

Given a Kripke structure K and a state X, a *trace* of K from X is a finite sequence $\sigma_1\sigma_2\dots\sigma_i$ of the labels of the states occurring in a path starting from X. Moreover, for an Shsm $\mathcal{M}$ and a boolean formula φ over the atomic propositions AP, we denote by $\mathcal{L}(\mathcal{M},\varphi)$ the set of traces $\sigma_1\dots\sigma_n$ of $\mathcal{M}^F$ from its initial state such that σ_n fulfills φ. In the rest of this section, we fix a set of atomic propositions $\Sigma = \{p_1,\dots,p_h\}$, for $h \geq 2$. We also use a function $\sharp$ which assigns to each subset $\sigma \subseteq \Sigma$ the natural number whose (h-bit) encoding is $b_h\dots b_1$ where b_i is 1 if and only if $p_i \in \sigma$. Formally, we define $\sharp(\sigma)$ as $\sum_{p_i\in\sigma} 2^{i-1}$ (in particular, $\sharp(\emptyset) = 0$).

Proposition 1. *Restricted* Shsms *can be exponentially more succinct than* Hsms *and finite state machines.*

Proof. Consider the family of languages L_1 of traces $\sigma_1\dots\sigma_n$ over 2^Σ such that: $\sigma_1 = \emptyset$; for $i < n$, $\sigma_i = \sigma_{i+1}$ or $\sharp(\sigma_{i+1}) = \sharp(\sigma_i) + 1$; and $\sharp(\sigma_n) = 2^h - 1$. Let $\varphi = p_1 \wedge \dots \wedge p_h$. For $h = 3$, a restricted Shsm $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M},\varphi) = L_1$ is given in Fig. 1. It is easy to see that $\mathcal{M}$ can be generalized to a restricted Shsm $\mathcal{M}$ such that $\mathcal{L}(\mathcal{M},\varphi) = L_1$, and $|\mathcal{M}| = O(h)$. Since there are 2^h different labels that need to be taken into account, we have that any hierarchical or finite state machine $\mathcal{M}'$ such that $\mathcal{L}(\mathcal{M}',\varphi) = L_1$ requires at least 2^h different nodes. □

There is an exponential gap also between restricted Shsms and Shsms as shown in the following proposition.

Proposition 2. Shsms *can be exponentially more succinct than restricted* Shsms.

Proof. Consider the Shsm $\mathcal{M}$ from Fig. 3, where $p_1,\dots,p_h$ are atomic propositions labeling nodes and boxes, in_i and z_i^j are, respectively, entry and exit nodes, and $expn(b_i^j) = i - 1$ for $i = 1,\dots,h$ and $j = 0,1,2$. Observe that $|\mathcal{M}| = O(h)$.

Fix a formula $\varphi = p_1 \wedge \dots \wedge p_h$, and denote with L_2 the language $\mathcal{L}(\mathcal{M},\varphi)$. Intuitively, L_2 contains sequences over 2^Σ which encode the behavior of a binary counter that besides the usual moves can also jump to a higher value by setting some bits to 1 and all the others to 0. Formally, it is possible to show that any $\sigma_1\dots\sigma_n \in L_2$ is such that: (1) $\sigma_1 = \emptyset$, (2) $\sharp(\sigma_n) = 2^h - 1$, and (3) for $i < n$ either $\sigma_i = \sigma_{i+1}$, or $\sharp(\sigma_{i+1}) = \sharp(\sigma_i) + 1$, or for some $1 \leq j \leq h$, $\{p_1,p_2\dots p_{j-1}\} \subseteq \sigma_i$, $p_j \notin \sigma_i$ and $\sigma_{i+1} = \{p_j,\dots,p_h\}$. Though some sequences satisfying the above three properties do not belong to L_2 (due to stuttering), we can prove the following property which is needed later in the proof:

(*) for each $j = 1,\dots,h$, there is a trace $\sigma_1\dots\sigma_n \in L_2$ such that for some $i < n$: $\{p_1,p_2\dots p_{j-1}\} \subseteq \sigma_i, p_j \notin \sigma_i$ and $\sigma_{i+1} = \{p_j,\dots,p_h\}$.

To complete the proof we need to show that any restricted Shsm $\mathcal{M}'$ such that $\mathcal{L}(\mathcal{M}',\varphi) = L_2$ has size at least exponential in h. Fix $\sigma_1,\sigma_2 \subseteq \Sigma$ such that $p_j \notin \sigma_1 \cup \sigma_2$, $\{p_1,\dots p_{j-1}\} \subseteq \sigma_1 \cap \sigma_2$ and there is an atomic proposition p_ℓ, $\ell > j$, such that $p_\ell \in \sigma_1$ and $p_\ell \notin \sigma_2$ (i.e., p_ℓ distinguishes σ_1 and σ_2). Let $\mathcal{M}'$ be any restricted Shsm such that $\mathcal{L}(\mathcal{M}',\varphi) = L_2$.

For the above property (*) and being $\sigma_1 \neq \sigma_2$, there must be three different states X_1,X_2,X_3 of ${\mathcal{M}'}^F$ such that X_3 is a successor of X_2, $trueAP(X_i) = \sigma_i$ for $i = 1,2$ and $trueAP(X_3) = \{p_j,\dots,p_h\}$. For $i = 1,2$, let $X_i = \langle \alpha_i b_i u_i \rangle$ where u_i is a node, b_i is a box, and α_i is a sequence of boxes. Recall that from the definition of restricted Shsm if p_ℓ labels any of the boxes in α_1 it cannot label any vertex of the machines to which either b_1 or u_1 belongs. Thus, either $b_1 \neq b_2$ or $u_1 \neq u_2$ must hold (otherwise, since X_3 is a successor of X_2 in ${\mathcal{M}'}^F$, we would get the contradiction that also X_3 could not be labeled with p_ℓ). Therefore, for each pair of different sets σ_1 and σ_2 as above, there must be two different vertices of $\mathcal{M}'$. Since we can choose σ_1 and σ_2 among 2^{h-j} many different sets, we can conclude that any restricted Shsm $\mathcal{M}'$ such that $\mathcal{L}(\mathcal{M}',\varphi) = L_2$ must have at least 2^{h-j} different vertices. Therefore, if we pick $j = 1$ we get that such $\mathcal{M}'$ must have at least 2^{h-1} different vertices, and the proposition is proved. □

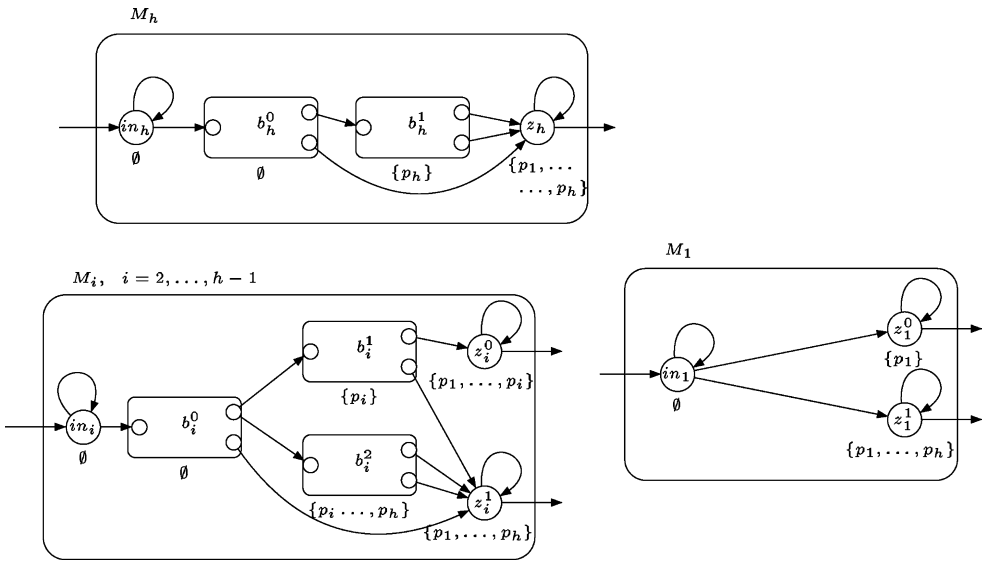

Fig. 3. An SHSM $\mathcal{M}$ for the language L_2.

It is worth noting that the above succinctness results do not add up to each other, in the sense that it is not true that SHSMs can be double exponentially more succinct than HSMs. In fact, HSMs, restricted SHSMs and SHSMs can all be translated to equivalent finite state machines with a single exponential blow-up. From Proposition 1 and the fact that any restricted SHSM is also an SHSM, we have the following.

Corollary 1. SHSM*s can be exponentially more succinct than* HSM*s and finite state machines.*

4. Reachability and cycle detection

In this section, we discuss the computational complexity of the reachability and cycle detection problems for SHSMs. We start defining these problems.

Given a transition system, the *reachability problem* is the problem of determining whether a given state can be reached starting from the initial state of the system. In practice, this problem is relevant in the verification of systems, for example it is related to the verification of *safety requirements*: we want to check whether all the reachable states of the system belong to a given "safe" region (*invariant checking problem*). In the invariant checking, the region of states is usually expressed by a propositional boolean formula φ (*invariant*), and this problem can be solved by solving the reachability problem with respect to any state in the set given by $\neg\varphi$. In this paper, if not otherwise specified, with "reachability" we refer to the problem defined with respect to a set of states represented by a propositional boolean formula. Formally, given an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$ and a propositional boolean formula φ, **Reach** is the problem of deciding if there exist a state X in $\mathcal{M}^F$ at which φ is satisfied and a path in $\mathcal{M}^F$ from $\langle in_k \rangle$ to X.

The *cycle detection problem* is the problem of verifying whether a given state can be reached repeatedly. Cycle detection is the basic problem for the verification of *liveness properties*, such as "something good will repeatedly happen". As for the reachability problem we can ask this question for a single state or for a set of states represented by a propositional boolean formula. Also here with "cycle detection" we refer to the problem defined with respect to a set of states represented by a propositional boolean formula. Formally, given an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$ and a propositional boolean formula φ, the problem **Cycle** is the problem of deciding if there exist a state X in $\mathcal{M}^F$ at which φ is satisfied, a path from $\langle in_k \rangle$ to X and a cycle through X.

In the next two sections, we show that both problems **Cycle** and **Reach** are NP-complete. Then, we discuss efficient solutions for classes of input instances.

4.1. NP*-Hardness*

First let us recall that, as intuition may suggest, **Cycle** is at least as difficult as **Reach**. In fact, given an SHSM $\mathcal{M}$ and a formula φ, let us add self-loops to all the nodes in each of the machines M_i (leaving the boxes unchanged) and call this new SHSM $\mathcal{M}'$. This can be obviously done in time linear in $|\mathcal{M}|$ and has as side effect for all the states of the corresponding $\mathcal{M}'^F$

to have self-loops. Now if a state X in $\mathcal{M}'^F$ exists which is repeatedly (maybe through self-loops) reachable and at which φ is satisfied, then X is reachable in $\mathcal{M}^F$ as well. Clearly if such X does not exist in $\mathcal{M}'^F$, then a reachable state at which φ is satisfied also does not exist in $\mathcal{M}^F$. Thus, we have the following proposition.

Proposition 3. *The problem* **Reach** *can be reduced to* **Cycle** *in polynomial time.*

Let us underline that the above arguments can be repeated (and in fact they are usually applied) to prove the hardness of the cycle detection problem on Kripke structures, once it is known the hardness of the reachability problem.

Lemma 1. *Problems* **Reach** *and* **Cycle** *for restricted* SHSMs *and propositional boolean formulas are* NP*-hard.*

Proof. We give a reduction in linear time with respect to the size of φ from the satisfiability problem SAT. Given a boolean formula φ over the atomic propositions $AP = \{P_1,P_2,\ldots,P_k\}$, we construct a restricted SHSM $\mathcal{M} = (M_1,M_2,\ldots,M_k)$ over AP, as follows. Each machine $M_i, i \geqslant 1$, has four vertices forming a chain: in_i,p_i,$notp_i$ and out_i. For $i > 1$, p_i and $notp_i$ are boxes expanding into M_{i-1}, and there are edges $((p_i,out_{i-1}),notp_i)$ and $((notp_i,out_{i-1}),out_i)$. Vertices p_1,$notp_1$ are nodes. Each vertex p_i is labeled by $\{P_i\}$, whereas the vertices $notp_i$, in_i and out_i are labeled by the empty set.

It is easy to verify that among all the states of $\mathcal{M}^F$ there are 2^k states of the form $\langle u_1 \ldots u_k\rangle$, such that $u_{k-i+1} \in \{p_i,notp_i\}$ for $i = 1,\ldots,k$. Furthermore, since such u_i are all connected, all these states are reachable from the initial state $\langle in_k\rangle$ of $\mathcal{M}^F$. Also, we can define a one-to-one correspondence between the truth assignments over the atomic propositions $P_1,\ldots,P_k$ and such states, such that a truth assignment ν assigns *true* to P_i if and only if $u_{k-i+1} = p_i$ in the corresponding state $X_\nu = \langle u_1 \ldots u_k\rangle$. Thus, from the labeling of $\mathcal{M}$ vertices, a truth assignment ν assigns *true* to P_i if and only if $P_i \in trueAP(X_\nu)$.

Therefore, a reachable state of $\mathcal{M}^F$ whose labeling corresponds to a truth assignment fulfilling φ exists if and only if φ is satisfiable. The overall construction can be done in $O(|\varphi|)$.

By Proposition 3, the NP-hardness for **Cycle** follows as well. □

From the above lemma, NP-hardness for the general problems follows.

Corollary 2. *Problems* **Reach** *and* **Cycle** *are* NP*-hard.*

Let us note now that the above hardness results depend on the sizes of two parameters: the SHSM $\mathcal{M}$ and the formula φ. It is hence natural to ask oneself whether putting some restrictions on the hierarchical machine and/or on the type of formulas, one can get efficient solutions for the problems **Reach** and **Cycle**.

In Section 4.3 we will show that by considering both restricted SHSM and DNF formulas we are guaranteed to get efficient algorithms (actually the conditions in Section 4.3 are even more general than this). In Lemma 1 and in the following lemma it is shown that only one of the two conditions is not enough to get efficient solutions.

When the SHSM is not a restricted SHSM, the hardness follows even when the formula is very simple, as shown in the following lemma. From this result it descends that problems **Reach** and **Cycle** for DNF formulas are NP-hard.

Lemma 2. *Problems* **Reach** *and* **Cycle** *for* SHSMs *and conjunctions of literals are* NP*-hard.*

Proof. We reduce 3-SAT to **Reach**, then by Proposition 3 we get the hardness also for **Cycle**.

Let ψ be a 3-SAT formula $\psi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$, over a set of atomic propositions $\{P_1,P_2,\ldots,P_k\}$, where each C_i is a disjunction of three literals.[2] We construct an SHSM $\mathcal{M} = (M_1,M_2,\ldots,M_k)$, over the set of atomic propositions $AP = \{c_1,c_2,\ldots,c_n\}$, in the following way: each $M_i, i \geqslant 1$ has four vertices forming a chain: in_i, p_i, $notp_i$ and out_i. The vertex p_i is labeled with c_j, if P_i occurs in the clause C_j, while the vertex $notp_i$ is labeled with c_j if $\neg P_i$ is in C_j. For $i > 1$, p_i and $notp_i$ are boxes expanding into M_{i-1}, having edges $((p_i,out_{i-1}),notp_i)$ and $((notp_i,out_{i-1}),out_i)$. Vertices p_1 and $notp_1$ are nodes.

Note that, except for the vertex labeling, $\mathcal{M}$ is as in the proof of Lemma 1, therefore all the observations on the states of $\mathcal{M}^F$ and their correspondence to truth valuations still hold. In addition, let ν be a truth valuation and be X_ν the corresponding state of $\mathcal{M}^F$: from the labeling of $\mathcal{M}$ we have that $c_j \in trueAP(X_\nu)$ if and only if ν fulfills a clause C_j.

Now, define $\varphi = c_1 \wedge c_2 \cdots \wedge c_n$. Clearly, φ evaluates to *true* at a state X_ν if and only if ν satisfies all clauses C_j and thus ψ. Therefore, ψ is satisfiable if and only if there exists a reachable state of $\mathcal{M}^F$ at which φ is satisfied.

The overall construction can be easily done in $O(|\psi|^2)$ time, and thus the lemma is shown. □

Note that the SHSM $\mathcal{M}$ constructed in the above proof is not a restricted SHSM. In general, it is not possible to prove the lemma for restricted SHSMs. In fact, in Section 4.3.3 we show that indeed the restriction of the problems **Reach** and **Cycle** to instances of restricted SHSMs $\mathcal{M}$ and formulas φ in disjunctive normal form can be solved in time linear in the size of $\mathcal{M}$ and φ.

[2] Recall that a literal is either an atomic proposition or the negation of an atomic proposition.

4.2. Membership in NP

The problems **Cycle** and **Reach** are defined in terms of a Kripke structure $\mathcal{M}^F$ and use the notion of path defined for graphs. Since our aim is to solve these problems by analyzing an SHSM, without flattening it, we now consider a notion of path and a notion of cycle on SHSM, that will be used also in the next sections.

Fix an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$. A sequence of vertices $u_0 \ldots u_n$ of M_h is a *local path* in M_h if either $n = 0$ (*zero-edge local path*) or $n > 0$ and for $j = 0, \ldots, n-1$:

- if u_j is a node, then $(u_j, u_{j+1}) \in E_h$;
- else, u_j is a box, then there exists an exit node $z_j \in \text{OUT}_{expn(u_j)}$ such that $((u_j, z_j), u_{j+1}) \in E_h$ and there is a local path from $in_{expn(u_j)}$ to z_j in $M_{expn(u_j)}$.

In the following, when we refer to a local path in M_h, we usually omit the reference to the machine M_h since it is univocally determined by the vertices $u_1, \ldots, u_n$. Moreover, we say that a vertex $u \in V_h$ is *connected* if there is a local path from in_h to u. Observe that if a node $u \in V_h$ is connected then there exists a path from $\langle in_h \rangle$ to $\langle u \rangle$ in the Kripke structure $\mathcal{M}_h^F$, while if a box u is connected, a path exists from $\langle in_h \rangle$ to $\langle u\, in_{expn(u)} \rangle$ (this can be easily proved by induction on h: for the basis note that V_1 contains only nodes and thus local paths in M_1 immediately lead to paths in $\mathcal{M}_1^F$; for the inductive step use the recursive definition of $\mathcal{M}_h^F$). Hence, if a vertex u is connected, this in turn means that there exists a path from $\langle \alpha\, in_h \rangle$ to either $\langle \alpha u \rangle$ or $\langle \alpha u\, in_{expn(u)} \rangle$ in $\mathcal{M}^F$, for any well-formed sequence $\alpha = \epsilon$ or $\alpha = u_1 \ldots u_j$, with $expn(u_j) = h$. Note, on the other side, that if there exists a path in $\mathcal{M}^F$ from $\langle in_k \rangle$ to a state $\langle u_1 \ldots u_m \rangle$, then there are paths from $\langle u_1 \ldots u_{j-1}\, in_{expn(u_{j-1})} \rangle$ to $\langle u_1 \ldots u_{j-1} u_j\, in_{expn(u_j)} \rangle$ for $j = 1, \ldots, m-1$ and from $\langle u_1 \ldots u_{m-1}\, in_{expn(u_{m-1})} \rangle$ to $\langle u_1 \ldots u_{m-1} u_m \rangle$. Therefore, all vertices u_j for $j = 1, \ldots, m$ are connected.

The above observations help to clarify the relation between the notion of local path in an SHSM and paths in the corresponding flat Kripke structure, and prove the following proposition.

Proposition 4. *Given an* SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$, *let* $X = \langle u_1 u_2 \ldots u_m \rangle$ *be a state of* $\mathcal{M}^F$. *There is a path from* $\langle in_k \rangle$ *to* X *in* $\mathcal{M}^F$ *if and only if all the vertices* u_i *are connected*.

In the following, given a box $u \in V_h$, with $[u,z]$ we denote a pair such that either u is a node and $z = \epsilon$, or u is a box and $z \in \text{OUT}_{expn(u)}$ is connected. A *local path through a pair* $[u,z]$ from u_0 to u_n is a non-zero-edge local path $u_0 \ldots u_n$ such that for some $0 \leq i < n$, $u = u_i$ and either $z = \epsilon$ or $((u_i, z), u_{i+1})$ is an edge. A *local cycle through a pair* $[u,z]$ is a local path through $[u,z]$ from u to itself.

We say that *a well-formed sequence* $u_1 \ldots u_m$ *is connected to a vertex* z_1 if there are $z_2, \ldots, z_m$ such that for $j = 2, \ldots, m$, there is a local path through the pair $[u_j, z_j]$ from $in_{expn(u_{j-1})}$ to z_{j-1}. This means that for some well-formed sequence α there is a path of $\mathcal{M}^F$ from $\langle \alpha u_1 in_{expn(u_1)} \rangle$ to $\langle \alpha u_1 z_1 \rangle$ which visits the states $\langle \alpha u_1 \ldots u_j z_j \rangle$ for $j = 2, \ldots, m$.

The following proposition captures the relationship between cycles in $\mathcal{M}^F$ and local cycles in $\mathcal{M}$.

Proposition 5. *Let* $\mathcal{M}$ *be an* SHSM *and* $X = \langle u_1 u_2 \ldots u_m \rangle$ *be a state of* $\mathcal{M}^F$. *Then*, X *belongs to a cycle in* $\mathcal{M}^F$ *if and only if either*:

- *there is a local cycle through* $[u_m, \epsilon]$ *or*
- *there exist* $1 \leqslant i < m$ *and* z_i *such that there is a local cycle through* $[u_i, z_i]$ *and the sequence* $u_i, \ldots, u_m$ *is connected to* z_i.

Proof. Let $X_0 X_1 \ldots X_r$ be a cycle of $\mathcal{M}^F$ through $X_0 = \langle u_1 u_2 \ldots u_m \rangle$. Let $u_1 \ldots u_{i-1}$ be the longest common prefix of all the complete well-formed sequences α_j such that $X_j = \langle \alpha_j \rangle$ for $j = 1, \ldots, r$ (if such prefix is the empty sequence, then $i = 1$). Observe that $i \leq m$ since u_m is a node.

There are two possible cases. If $i = m$, then clearly by definition, there is a local cycle through $[u_m, \epsilon]$. Otherwise, let $\alpha_\ell = u_1 \ldots u_{i-1} u_i z_i$ be such that u_i is a box and $z_i \in \text{OUT}_{expn(u_i)}$ is connected. (Such α_ℓ clearly exists since $u_1 \ldots u_{i-1}$ is the longest common prefix). By the definition of local cycle, it is simple to verify that there must exist a local cycle through $[u_i, z_i]$. Also, since $X_0 X_1 \ldots X_r$ is a cycle for $j = i+1, \ldots, m$ there are states of $\mathcal{M}^F$ of the forms $X_{y_j} = \langle u_1 \ldots u_{j-1} in_{expn(u_{j-1})} \rangle$ and $X_{z_j} = \langle u_1 \ldots u_{j-1} z_j \rangle$ (i.e., corresponding to entering and exiting the machine $M_{expn(u_{j-1})}$ while accessing it from box u_{j-1} on the considered cycle of $\mathcal{M}$) such that: $z_j \in \text{OUT}_{expn(u_{j-1})}$ and the portion of the cycle from X_{y_j} to X_{z_j} goes through a state X_h such that $u_1 \ldots u_{j-1} u_j$ is a prefix of α_h. Thus, by the definition of local path, clearly there are $z_{i+1}, \ldots, z_m$ such that for $j = i+1, \ldots, m$ there is a local path through $[u_j, z_j]$ from $in_{expn(u_{j-1})}$ to z_{j-1}. The converse direction can be shown using analogous arguments. Therefore, the proposition holds. □

Given an SHSM $\mathcal{M}$, using $O(|\mathcal{M}|)$ time we can compute whether its vertices are connected and whether its pairs are part of a cycle. The proof of this statement can be obtained from a rather simple modification of a depth-first search on a graph, see also [6]. Therefore, we just state formally this result in the following proposition.

Proposition 6. *Given an* SHSM $\mathcal{M}$, *the set of connected vertices and the set of pairs* $[u,z]$ *such that there is a local cycle of* $\mathcal{M}$ *through it can be computed in* $O(|\mathcal{M}|)$ *time. Moreover, given a pair* $[u,z]$ *and a vertex v, determining if v is connected through* $[u,z]$ *can be checked in* $O(|\mathcal{M}|)$ *time.*

The following lemma gives the upper bound on the computational complexity of **Reach** and **Cycle**.

Lemma 3. *Problems* **Reach** *and* **Cycle** *are in* NP.

Proof. Fix an SHSM $\mathcal{M} = (M_1,M_2,\ldots,M_k)$, a propositional boolean formula φ and a the state $X = \langle u_1u_2\ldots u_m\rangle$ of $\mathcal{M}^F$. Recall that $m \leq k$.

By Proposition 4, it is possible to verify whether there is a path from $\langle in_k\rangle$ to X in $\mathcal{M}^F$, by just verifying whether each u_i is connected, and from Proposition 6, this can be done in $O(|\mathcal{M}|)$ time. Moreover, $O(|\varphi| + |\mathcal{M}|)$ time is needed to check the truth of φ on X, thus **Reach** belongs to NP.

Now consider the problem **Cycle**. From Proposition 5 deciding **Cycle** reduces to for the existence of a local cycle through a pair and related local paths through pairs. From Proposition 6, this can be done using polynomial time. Therefore, the problem **Cycle** is in NP. □

From Corollary 2 and Lemma 3, we obtain the following theorem.

Theorem 1. *Problems* **Reach** *and* **Cycle** *are* NP*-complete.*

4.3. Efficient solutions

In this section, we present some conditions on the SHSM $\mathcal{M}$ and the formula φ which allow us to give efficient algorithms for both problems **Reach** and **Cycle**. In Section 4.3.1, we define the *partial evaluation* of φ on well-formed sequences. In Section 4.3.2, we give the algorithms for the above two problems.

4.3.1. The partial evaluation on well-formed sequences

Informally speaking, a partial evaluation of φ on a well-formed sequence α is the evaluation of φ on the atomic propositions labeling vertices occurring in α. Given a formula φ and two disjoint sets $T,F \subseteq AP$, let Inst (φ,T,F) denote the formula obtained by instantiating to *true* the atomic propositions of T and to *false* those in F. A propositional formula φ can be evaluated in a state X of $\mathcal{M}^F$, simply by computing Inst $(\varphi,trueAP(X),AP \setminus trueAP(X))$. To solve our problems instead, we would like to find a state in $\mathcal{M}^F$ where φ is satisfied, using the SHSM $\mathcal{M}$, without constructing $\mathcal{M}^F$. To this aim we use a greedy approach to evaluate φ: we visit $\mathcal{M}$ in a top-down way starting from M_k and at each vertex u we instantiate as many atomic propositions as possible. The question is: which atomic propositions of φ can be instantiated? Surely, we can instantiate to *true* all the atomic propositions of $trueAP(u)$, while to determine the atomic propositions which can be instantiated to *false* is more difficult: it depends on the vertices that may follow u in any complete well-formed sequence of $\mathcal{M}$. In other terms, an atomic proposition can be instantiated to *false* if it labels neither u nor vertices having u as an ancestor, that is it does not belong to a set $trueAP^*(u)$ defined as

$$trueAP^*(u) = trueAP(u) \cup \bigcup_{v \in V_{expn^+(u)}} trueAP(v).$$

Thus, we define the set of atomic propositions of φ that can be instantiated to *false* at a vertex $u \in V_h$, as

$$falseAP(u) = AP \setminus trueAP^*(u).$$

We can now inductively define the partial evaluation of φ on a well-formed sequence.

Definition 4. Given an SHSM $\mathcal{M}$, a propositional boolean formula φ over AP and a well-formed sequence α of $\mathcal{M}$, the partial evaluation of φ on α, is defined as

$$\begin{cases} PEval(\varphi,\epsilon) = \varphi \\ PEval(\varphi,\alpha u) = \text{Inst } (PEval(\varphi,\alpha),trueAP(u),falseAP(u)). \end{cases}$$

In the following proposition, we show that our approach for partially evaluating a formula in a top-down way on well-formed sequences leads to the evaluation of the formula on a state of the flat $\mathcal{M}^F$.

Proposition 7. *Given an* SHSM $\mathcal{M}$, *a formula* φ, *and a state* $X = \langle\alpha\rangle$ *of* $\mathcal{M}^F$:

$$PEval(\varphi,\alpha) = \text{Inst}(\varphi,trueAP(X),AP \setminus trueAP(X)).$$

Algorithm **Reachability**$(\mathcal{M}, \varphi)$

initialize the array VISITED to FALSE;
compute the function $falseAP(u)$ for all $u \in V$;
return(Reach(k, φ)).

Function Reach(h, ψ)
1) VISITED$[h] \leftarrow$ TRUE; /* mark as visited the machine M_h */
2) **foreach** $v \in V_h$ **do**
3) $\psi' \leftarrow \text{Inst}(\psi, trueAP(v), falseAP(v))$; $h' \leftarrow expn(v)$;
4) **if** $(\psi' = true)$ **then return** TRUE;
5) **if** $(\psi' \neq false)$ **then** /* v is a box */
6) **if** (VISITED$[h'] =$ FALSE) **then**
/* call Reach recursively */
7) **if** (Reach$(h', \psi') =$ TRUE) **then return** TRUE;
8) **endfor**
9) **return** FALSE;

Fig. 4. An algorithm which solves the problem **Reach** for each SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$ and formula φ such that the partial evaluations of φ in $\mathcal{M}$ are uniquely inherited.

Proof. For every well-formed sequence $\alpha = u_1 u_2 \ldots u_m, m \geqslant 1$, let $T(\alpha) = \cup_{j=1}^{m} trueAP(u_j)$ and $F(\alpha) = AP \setminus (T(\alpha) \cup trueAP^*(u_m))$. Thus, if we are given a state $X = \langle \alpha \rangle$, then $T(\alpha) = trueAP(X)$ and $F(\alpha) = AP \setminus trueAP(X)$. Therefore, to prove the proposition, it is sufficient to show that $PEval(\varphi,\alpha) = \text{Inst}(\varphi, T(\alpha), F(\alpha))$.

The proof is by induction on the length of α. The basis for $|\alpha| = 1$ is trivial. Now, let $\alpha = \beta u$ with $\beta \in V^+$ and $u \in V$. By the definition of *PEval* and the inductive hypothesis, we get that $PEval(\varphi,\alpha)$ is given by $\text{Inst}(\text{Inst}(\varphi, T(\beta), F(\beta)), trueAP(u), falseAP(u))$. Denote such formula as ψ_1 and formula $\text{Inst}(\varphi, T(\beta u), F(\beta u))$ as ψ_2. We now prove that $\psi_1 = \psi_2$ and thus the proposition holds.

Observe that, from the definition of $F(\beta)$, $trueAP(u)$ and $F(\beta)$ are disjoint sets. Thus, since $T(\beta u) = T(\beta) \cup trueAP(u)$, every atomic proposition which is instantiated to true to obtain ψ_1 from φ is also instantiated to true to obtain ψ_2 from φ, and vice-versa.

From the above definition, we get ψ_1 from φ by instantiating to false all the atomic propositions of $F(\beta)$ along with all the atomic propositions of $falseAP(u)$ which are not in $T(\beta)$ (which have been already instantiated to true). Since $falseAP(u) \cap trueAP(u) = \emptyset$, we can rewrite $falseAP(u) \setminus T(\beta)$ as $falseAP(u) \setminus (T(\beta) \cup trueAP(u))$, which is $falseAP(u) \setminus T(\beta u)$. By applying the definition of $falseAP(u)$, we get that such set is $(AP \setminus trueAP^*(u)) \setminus T(\beta u)$ and thus $AP \setminus (trueAP^*(u) \cup T(\beta u))$, which is the definition of $F(\beta u)$.

Therefore, all the atomic propositions which are instantiated to false to obtain ψ_1 from φ are exactly those instantiated to false to obtain ψ_2 from φ. Hence, $PEval(\varphi,\beta u) = \text{Inst}(\varphi, T(\beta u), F(\beta u))$, which concludes the proof. □

In what follows without loss of generality we assume that the formula returned by a partial evaluation is simplified according to the following tautologies.

- $(\psi \wedge true) \equiv \psi$,
- $(\psi \wedge false) \equiv false$,
- $(\psi \vee false) \equiv \psi$,
- $(\psi \vee true) \equiv true$,
- $(\neg true) \equiv false$,
- $(\neg false) \equiv true$.

We say that φ is *constant* if it is either *true* or *false*.

4.3.2. Efficient Algorithms

In this section, we define a condition on SHSMs and formulas to get efficient algorithms for solving **Reach** and **Cycle**.

Let $\mathcal{M}$ be an SHSM and φ be a boolean formula. We say that the partial evaluations of φ in $\mathcal{M}$ are *uniquely inherited* iff: for every two well-formed sequences αu and βv, if $expn(u) = expn(v)$ and both $PEval(\varphi,\alpha u)$ and $PEval(\varphi,\beta v)$ are not constant, then $PEval(\varphi, \alpha u) = PEval(\varphi,\beta v)$.

Consider an SHSM $\mathcal{M}$ and w.l.o.g. assume that all the vertices are connected (if this is not the case all the non-reachable vertices can be canceled in $O(|\mathcal{M}|)$ time according to Proposition 6).

Algorithm **Cycle-Detection**$(\mathcal{M}, \varphi)$
1) initialize the arrays isInCycle, cycleExpPlus, visited, outSat and sat to FALSE;
2) **foreach** machine M_h and $[u, z]$ of M_h **do**
3) **if** $[u, z]$ is in a local cycle in M_h **then** isInCycle$[u, z] \leftarrow$ TRUE;
4) **foreach** $u \in V$ **do**
5) **if** there exists a local cycle in $expn^+(u)$ **then** cycleExpPlus$[u] \leftarrow$ TRUE;
6) **return** Cycle(k, φ).

Fig. 5. Algorithm cycle-detection.

Let us first examine the problem **Reach**. Using mainly Proposition 7, we design an algorithm that looks for a complete well-formed sequence α for which the function $PEval(\varphi,\alpha)$ is TRUE. We give an algorithm that visits the machines of $\mathcal{M}$ in a top-down way and evaluates *PEval* on well-formed sequences by computing iteratively the function Inst.

Theorem 2. *For an* Shsm $\mathcal{M}$ *and a formula* φ *such that the partial valuations of* φ *in* $\mathcal{M}$ *are uniquely inherited, the problem* **Reach** *is decidable in* $O(|\mathcal{M}| \cdot |\varphi|)$ *time.*

Proof. Consider an Shsm $\mathcal{M} = (M_1,M_2,\ldots,M_k)$, in Fig. 4 the algorithm for solving **Reach** is shown. Function Reach(h,ψ) uses a global boolean array visited to record the visited machines, (initialized by the algorithm to FALSE) and computes the function $falseAP(\cdot)$. For each vertex v of M_h (line 3) ψ is evaluated on it according to $trueAP(v)$ and $falseAP(v)$. If ψ' evaluates to the constant *true* on v (line 4), then Reach stops returning TRUE (the algorithm terminates too, returning TRUE). If ψ' still contains atomic propositions, then v is a box expanding into $M_{h'}$. Moreover if $M_{h'}$ has never been visited, then the function is recursively called on h' and ψ'. In case ψ' is *false* or the recursive call returns FALSE, then another vertex in M_h is processed. It is easy to see that function Reach(h,ψ) returns TRUE if and only if ψ is evaluated to *true* on a complete well-formed sequence of $\mathcal{M}$ (see Proposition 7). To conclude the proof, note that if the evaluations of φ are uniquely inherited, multiple visits of a machine M_h are not needed, and thus the overall complexity of the algorithm is linear in $|\mathcal{M}|$ and $|\varphi|$. □

Now we consider the problem **Cycle**. Our algorithm to solve this problem strongly relies on the following lemma.

Lemma 4. *Let* $\mathcal{M}$ *be an* Shsm *with all connected vertices,* φ *be a boolean formula and* $\alpha = u_1 \ldots u_j$ *be a well-formed sequence such that* $PEval(\varphi,\alpha)$ *=true. A well-formed sequence exists* β *such that* $\langle\alpha\beta\rangle$ *belongs to a cycle of* $\mathcal{M}^F$ *and satisfies* φ*, if and only if either*

(a) *there exists a local cycle in* M_h, $h \in expn^+(u_j)$, *or*
(b) *there exists* z_j *such that there is a local cycle through* $[u_j,z_j]$ *or*
(c) *there exist* $i < j$ *and* z_i *such that there is a local cycle through* $[u_i,z_i]$ *and* $u_i,\ldots,u_j$ *is connected to* z_i.

Proof. For the "if" part we show that if one of the three conditions (a), (b), and (c) holds true then a well-formed sequence β exists such that $\langle\alpha\beta\rangle$ belongs to a cycle in $\mathcal{M}^F$. Suppose that condition (a) holds with respect to a local cycle through $[u_i,z_i]$. Since $h \in expn^+(u_j)$ then there is a well-formed sequence $\langle u_j,\ldots,u_i\rangle$. Now, if u_i is a node then choose $\beta = \langle u_{j+1},\ldots,u_i\rangle$, otherwise choose $\beta = \langle u_{j+1},\ldots,u_i,in_{expn(u_i)}\rangle$. Suppose, now, that either condition b) holds, with $z_j \neq \epsilon$, or condition c) holds: in both the cases choose $\beta = \langle in_{expn(u_j)}\rangle$. Finally, if condition b) holds with $z_j = \epsilon$ then choose $\beta = \epsilon$. From Proposition 5, in all the above cases $\langle\alpha\beta\rangle$ belongs to a cycle, and also $PEval(\varphi,\alpha\beta) = PEval(\varphi,\alpha) = true$.

For the "only if" part, let $\langle\alpha\beta\rangle = \langle u_1,\ldots,u_j,\ldots,u_m\rangle$. From Proposition 5, there are $0 \leqslant i \leqslant m$ and z_i and there is a local cycle through $[u_i,z_i]$ such that either $i = m$ or $u_i,\ldots,u_m$ is connected to z_i. If $j < i \leqslant m$ then there exists a local cycle in M_h, $h \in expn^+(u_j)$, and thus condition (a) holds. If $j = i$ then condition (b) holds. Finally, if $j > i$, then the sequence $u_i,\ldots,u_m$ is connected to z_i and thus also $u_i,\ldots,u_j$ is connected to z_i, and thus condition (c) holds. □

Now we give the main theorem.

Theorem 3. *For an* Shsm $\mathcal{M}$ *and a formula* φ *such that the partial valuations of* φ *in* $\mathcal{M}$ *are uniquely inherited, the problem* **Cycle** *is decidable in* $O(|\mathcal{M}| \cdot |\varphi|)$ *time.*

Proof. We propose an efficient algorithm for solving **Cycle** which implements the idea behind the characterization given in Lemma 4 and mainly consists of exploring an Shsm $\mathcal{M}$ by a depth-first visit algorithm (expansions as edges of the graph in this visit). The evaluation of the formula is done using a greedy approach: as soon as an atomic proposition can be instantiated the current partial evaluation of the formula is updated; therefore, the partial evaluation of a formula is done as

```
Function Cycle(h, ψ)
1)  VISITED[h] ← TRUE;    /* mark as visited the machine M_h */
2)  foreach v ∈ V_h do
3)      h' ← expn(v);
4)      ψ' ← Inst(ψ, trueAP(v), falseAP(v));    /* partially evaluate ψ on v */
5)      if (ψ' =true AND h' = 0) then SAT[v, ε] ← TRUE;
6)      if (ψ' =true AND h' > 0) then
7)          foreach z ∈ OUT(v) do SAT[v, z] ← TRUE;
8)          if CYCLEEXPPLUS[v]=TRUE then    /* look for a cycle */
                return TRUE;
9)      if (ψ' ≠ true AND ψ' ≠ false) then
10)         if (VISITED[h'] = FALSE)) then
11)             if ( Cycle(h', ψ') = TRUE) then    /* call Cycle recursively */
                    return(TRUE);
12)     foreach (z ∈ OUT(v) s.t. OUTSAT(z) = TRUE) do SAT[v, z] ← TRUE;
    endfor
13) foreach pair [u, z] of M_h do
14)     if (ISINCYCLE[u, z]=TRUE AND SAT[u, z] = TRUE) then
            return TRUE;
    endfor
15) For z ∈ OUT_h, set OUTSAT(z) = TRUE whenever a local path exists
    from a pair [v', z'] to z s.t. SAT[v', z'] = TRUE;
16) return(FALSE);
```

Fig. 6. Function cycle.

soon as a vertex is discovered. On each vertex, the information about the connectivity in the graph (i.e., paths through pairs, presence of cycles, etc.) is computed on the basis of the information collected during the visit of its neighbors. We make use of some precomputing and global data structures to implement this idea efficiently. We assume that the input SHSM $\mathcal{M}$ to our algorithm is such that all the vertices of $\mathcal{M}$ are connected. Fix an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$. The detailed algorithm is given in Figs. 5 and 6. It consists mainly of a function *Cycle* which is called for the first time on k and φ and visits all the vertices v of M_h. Moreover, it is recursively invoked on the machine $M_{expn(v)}$ and the formula obtained instantiating ψ on v. We need to show that this algorithm returns TRUE if and only if there exists a cycle of $\mathcal{M}^F$ containing a state X at which φ holds true.

The algorithm uses global arrays ISINCYCLE, CYCLEEXPPLUS, VISITED, OUTSAT and SAT with the following meaning:

- ISINCYCLE[u,z] = TRUE if there is a local cycle through [u,z];
- CYCLEEXPPLUS[u] = TRUE if there exists a local cycle in M_h, with $h \in expn^+(u)$;
- VISITED[h] = TRUE if machine M_h has been visited by the algorithm;
- SAT[u,z] = TRUE if formula φ is satisfiable at a state $\langle \alpha\, u\beta \rangle$ such that $u\beta$ is connected to z;
- OUTSAT[z] = TRUE if there exists a local path through [u,z'] from u to an exit z such that SAT[u,z'] = TRUE.

According to their intended meaning, the global arrays are correctly initialized to FALSE in line 1. In lines 2–5, the entries of CYCLEEXPPLUS[u] and ISINCYCLE[u,z] are precomputed. (Recall that from Proposition 6, we can compute ISINCYCLE[u,z] in $O(\mathcal{M})$ time. Once this array is computed the computation of CYCLEEXPPLUS[u] is trivial.) The final step of the algorithm starts recursive invocations of function *Cycle*(k,φ).

The following observation will be useful in what follows: for each sequence *Cycle*(k,φ), ..., *Cycle*(h,ψ), of $r+1$ consecutive recursive calls, there are corresponding well-formed sequences $\gamma_{h,\psi} = u_1 \ldots u_r$ such that $h = expn(u_r)$ and $\psi = PEval(\varphi, \gamma_{h,\psi})$.

Cycle(h,ψ) iteratively visits all the vertices of machine M_h. Given a vertex v, $\psi' = PEval(\psi, v) = PEval(\varphi, \gamma_{h,\psi} v)$ is computed. Now, if ψ' =*true*, the array SAT is updated in lines 5 and 7. Moreover if v is a box then the function looks for a local cycle in $expn^+(v)$ and returns TRUE if and only if condition (a) of Lemma 4 holds.

If ψ' is not a constant (line 9) the function is recursively called on h', if $M_{h'}$ has not been visited yet. Note that since the partial evaluations of φ in $\mathcal{M}$ are uniquely inherited, all the other possible recursive calls involving machine $M_{h'}$ must have the same second parameter ψ', and thus are not needed.

If *Cycle*(h',ψ') returns TRUE, a witness for **Cycle** is found and thus we are done, otherwise array SAT[v,z] is updated for all vertices z with OUTSAT(z) = TRUE (that is, for all vertices z for which a local path through a pair [v',z'] exists from v' to z in $M_{h'}$ such that SAT[v',z'] =TRUE). Now it is easy to see that the following invariant, which explains more precisely the role of SAT, holds: SAT[u,z] =TRUE if and only if either (1) *PEval*(ψ,u) =*true* or (2) there exists a well-formed sequence $u\beta$ such that *PEval*(ψ,$u\beta$) =*true* and $u\beta$ is connected to z.

When all vertices of V_h have been processed, the function looks for a local cycle through a pair $[u,z]$ such that SAT$[u,z]$ =TRUE (lines 13 and 14). Hence the function returns TRUE if and only if there exist a local cycle through $[u,z]$, a well-formed sequence β connected to z and $PEval(\varphi,\gamma_{h,\psi}\beta) = PEval(\psi,\beta) = true$, that is the function returns TRUE if and only if conditions b or c) of Lemma 4 hold. Line 15 of the algorithm consistently updates OUTSAT for the exits of M_h.

Observe that, once the function returns TRUE for the first time, either at line 8 or at line 14, the algorithm returns TRUE, as well. In the other cases it returns FALSE. Thus, from Lemma 4 the given algorithm is correct.

Concerning the time complexity of the algorithm, note that the assumption that all the vertices are connected can be accomplished in linear time, due to Proposition 6. We have already observed that lines 2–5 can be done in linear time. For the function $Cycle(h,\psi)$, it is sufficient to note that lines 2–14 visit at most once all vertices and all edges of V_h, and that each machine is visited just once (line 10). Finally line 15 can be trivially implemented in time $O(|M_h|)$. Thus, the overall complexity is $O(|\mathcal{M}| \cdot |\varphi|)$. □

Note that the above algorithms can be easily modified to work for each φ and $\mathcal{M}$. In fact, in the general case, a machine M_h can inherit different partial evaluations of φ, thus we just need to program a control which ensures that each machine M_h is visited at most once for each partial evaluation of φ. Thus, the following theorem holds.

Theorem 4. *Given an* SHSM $\mathcal{M}$ *and a formula* φ, *the problems* **Reach** *and* **Cycle** *are decidable in* $O(|\mathcal{M}| \cdot 2^{|\varphi|})$ *time.*

Observe that when in $\mathcal{M}$ there are not scope properties ($trueAP(u) = \emptyset$ for every box u), then the partial evaluations of every formula φ in $\mathcal{M}$ are uniquely inherited, and thus Theorems 2 and 3 generalize those given in [6],

4.3.3. Efficient solutions for restricted SHSM

First we show that, in any restricted SHSM, the partial evaluations of a conjunction of literals are uniquely inherited. This will allow us to complete the reasoning started in Section 4.1 about the complexity of the problems **Reach** and **Cycle** (see Lemmas 1 and 2). Then we introduce a further condition which guarantees an efficient solution to our problems.

Lemma 5. *Let* $\mathcal{M}$ *be a restricted* SHSM, *and* φ *be a formula. If* φ *is a conjunction of literals then the partial evaluations of* φ *in* $\mathcal{M}$ *are uniquely inherited.*

Proof. Let $\varphi = l_1 \wedge \cdots \wedge l_r$, where each l_i is a literal, $1 \leqslant i \leqslant r$. Clearly, a partial evaluation of φ deletes some literals and returns either a constant or a conjunction of the remaining literals.

Suppose now that there exist two well-formed sequences of $\mathcal{M}$, say αu and βv, such that (1) $expn(u) = expn(v)$, (2) both $\varphi' = PEval(\varphi,\alpha u)$ and $\varphi'' = PEval(\varphi,\beta v)$ are not constant, and (3) $\varphi' \neq \varphi''$. In such a case, there exists a literal l_i occurring exactly in one between φ' and φ'', say in φ'. If l_i, is the atomic proposition P or the negation of P, then P does not belong to neither $falseAP(u)$ nor $trueAP(u)$ and this implies, from definition of the function $falseAP$, that a vertex w having u as an ancestor is labeled by P. Note that, for every v' occurring in βv, w has v' as ancestors, and thus P does not belong to $falseAP(v')$. On the other hand, P does not belong to $trueAP(v')$ as well, since $\mathcal{M}$ is a restricted SHSM. Therefore, l_i must occur in φ''. □

By combining the above Lemma 5 and Theorems 2 and 3, we can claim that for restricted SHSMs and DNF formulas we obtain more efficient algorithms.

Corollary 3. *Given a restricted* SHSM *and a DNF formula* φ, *the problems* **Reach** *and* **Cycle** *are decidable in* $O(|\mathcal{M}| \cdot |\varphi|)$ *time.*

Now we give a further condition on $\mathcal{M}$ and φ which allows us to get efficient algorithms when $\mathcal{M}$ is a restricted SHSM. Let $AP(\psi)$ be the set of atomic propositions of ψ. We say that ψ is *local* to $\mathcal{M}$ vertices iff for every vertex u of $\mathcal{M}$ either $AP(\psi) \cap (trueAP(u) \cup falseAP(u)) = \emptyset$ or $AP(\psi) \cap (trueAP(u) \cup falseAP(u)) = AP(\psi)$ holds. It is possible to prove that for a restricted SHSM $\mathcal{M}$, if φ is of the form $\varphi_1 \wedge \cdots \wedge \varphi_m$ where each φ_i is local to $\mathcal{M}$ vertices, then the partial evaluations of φ in $\mathcal{M}$ are uniquely inherited. Therefore, we have the following corollary.

Corollary 4. *Given a restricted* SHSM $\mathcal{M}$ *and a formula* φ *which is a conjunction of formulas that are local to* $\mathcal{M}$ *vertices, the problems* **Reach** *and* **Cycle** *are decidable in* $O(|\mathcal{M}| \cdot |\varphi|)$ *time.*

As a final remark, note that from Lemma 2, **Reach** and **Cycle** are NP-hard for SHSMs even if we restrict to conjunctions of formulas which are local to the vertices of the considered SHSM.

5. LTL and CTL model checking

Model checking against temporal logic specifications is defined as: given a system model K and a temporal logic formula φ, does K satisfy φ? The system model usually is a Kripke structure and fulfillment of formulas is defined over the computations

of the model. Temporal logic formulas are built from atomic propositions using the boolean operators, temporal operators (such as *until, next*), and in case of branching-time requirements also path quantifiers. In this section, we consider model checking of (restricted) SHSMs against LTL [9] and CTL [1] requirements. We do not make an explicit use of the syntax and the semantics of these logics, therefore we refer the reader to [18] for a formal definition.

5.1. LTL *model checking*

We follow the automata theoretic approach [19] and solve the model checking problem by a reduction to the emptiness problem for the intersection of an SHSM and a Büchi automaton. A Büchi automaton $A = (Q,q_1,\Delta,L,T)$ is a Kripke structure (Q,q_1,Δ,L) together with a set of accepting states T.

Let $\mathcal{M} = (M_1,M_2,\ldots,M_k)$ be an SHSM and $A = (Q,q_1,\Delta,L,T)$ be a Büchi automaton, where $Q = \{q_1,\ldots,q_m\}$. We will construct an SHSM $\mathcal{M}' = \mathcal{M} \bigotimes A$ which is essentially the Cartesian product of $\mathcal{M}$ and A, informally defined as follows. The vertices of $\mathcal{M}'$ are labeled just with an atomic proposition which we denote *trgt*. Since we allow each machine of an SHSM to have exactly an entry node, we need to make a different copy of each machine M_i of $\mathcal{M}$ for each possible state q_j of A which can be coupled with the entry of M_i. Also, in $\mathcal{M}'$ we keep track of the atomic propositions which are inherited on expanding a box to M_i in $\mathcal{M}$ by making a different copy of M_i for each possible set of atomic propositions P. Thus, along a run[3] of $\mathcal{M}'$, we mimic both $\mathcal{M}$ entering a machine M_i and A moving to a state q_j by entering the copy of M_i which corresponds to q_j and P provided that P is the set of atomic propositions inherited by the nodes of M_i at this point of the computation.

We define the machines $M_{(i,j,P)}$ of $\mathcal{M}' = \mathcal{M} \bigotimes A$, where $1 \leqslant i \leqslant k$, $1 \leqslant j \leqslant m$, and P is a subset of AP such that $P \cup trueAP_{\mathcal{M}}(in_i) = L(q_j)$. The vertices of $M_{(i,j,P)}$ are 4-tuples $[u,q,j,P]$. The third and fourth components of such tuples are the same for all the vertices of the same graph and are used only as an encoding to distinguish between vertices of different graphs. The first and second components are (u,q) belonging to the standard Cartesian product of M_i and A, with the following restriction. If u is a node of M_i, the labeling $L(q)$ of q coincides with the labeling of u augmented with the set P of the atomic propositions that u inherits from its ancestors; for a box u, with $expn(u) = h$, the labeling of q contains also the atomic propositions labeling the initial node in_h of the expansion of u. The edges of $M_{(i,j,P)}$ are obtained in a standard way from those of M_i and A such that moving along an edge of $M_{(i,j,P)}$ corresponds to follow both an edge of M_i and a transition of A.

Formally, we have:

- The set $V_{(i,j,P)}$ of the vertices of $M_{(i,j,P)}$ contains quadruples $[u,q,j,P]$, where $u \in V_i$, $q \in Q$, and
  - either $expn_{\mathcal{M}}(u) = 0$ and $L(q) = P \cup trueAP_{\mathcal{M}}(u)$, or
  - $expn_{\mathcal{M}}(u) = h > 0$ and $L(q) = P \cup trueAP_{\mathcal{M}}(u) \cup trueAP_{\mathcal{M}}(in_h)$.
- The initial node of $M_{(i,j,P)}$ is $[in_i,q_j,j,P]$ and the output nodes are $[u,q,j,P]$ for $u \in \text{OUT}_i$ and $q \in Q$;
- For each $(q',q'') \in \Delta$, $M_{(i,j,P)}$ contains the following edges:
  - $([u,q',j,P],[v,q'',j,P])$, for each $(u,v) \in E_i$,
  - $(([u,q_t,j,P],[z,q',t,P \cup trueAP_{\mathcal{M}}(u)]),[v,q'',j,P])$, for each $((u,z),v) \in E_i$ and $q_t \in Q$

We can now give the SHSM $\mathcal{M}' = \mathcal{M} \bigotimes A$ as follows:

- $M_{(k,1,\emptyset)}$ is the top-level machine of $\mathcal{M}'$, (see Definition 3);
- Let $M_{(i,j,P)}$ be a machine of $\mathcal{M}'$, and $[u,q_t,j,P]$ be a vertex of $M_{(i,j,P)}$.
  - If $expn_{\mathcal{M}}(u) = 0$ then $expn_{\mathcal{M}'}([u,q_t,j,P]) = 0$;
  - If $expn_{\mathcal{M}}(u) = h > 0$ and $P' = P \cup trueAP_{\mathcal{M}}(u)$ then $M_{(h,t,P')}$ is a machine of $\mathcal{M}'$ and $expn_{\mathcal{M}'}([u,q_t,j,P])$ is the index of $M_{(h,t,P')}$;
- The labeling of $\mathcal{M}'$ is defined as follows:
  - $trueAP_{\mathcal{M}'}([u,q,j,P]) = \{trgt\}$ for all $[u,q,j,P]$ such that $q \in T$ and
  - $trueAP_{\mathcal{M}'}([u,q,j,P]) = \emptyset$ otherwise.

An upper bound on the size of $\mathcal{M}'$ is given in the following lemma.

Lemma 6. *Given an* SHSM $\mathcal{M}$ *and a Büchi automaton* A *with set of states* Q, $\mathcal{M}' = \mathcal{M} \bigotimes A$ *has size* $O(|Q|^2 \cdot |\mathcal{M}| \cdot |A| \cdot |2^{AP}|)$. *Moreover, if* $\mathcal{M}$ *is restricted, then the size of* $\mathcal{M}'$ *is* $O(|Q|^2 \cdot |\mathcal{M}| \cdot |A|)$.

Proof. Let $\mathcal{M} = (M_1,M_2,\ldots,M_k)$ be an SHSM and $A = (Q,q_1,\Delta,L,T)$ be a Büchi automaton. Let us first consider the size of each machine $M_{(i,j,P)}$. There is at most one edge in $M_{(i,j,P)}$ for any $(q',q'') \in \Delta$ and $(u,v) \in E_i$ and there are at most $|Q|$ edges in $M_{(i,j,P)}$

[3] Informally a *run* of an SHSM $\mathcal{M}$ is a path of $\mathcal{M}^F$ starting from in_k.

for any $(q',q'') \in \Delta$ and $((u,z),v) \in E_i$, thus an upper bound to the size of $M_{(i,j,P)}$ is given by $(|Q| \cdot |M_i| \cdot |A|)$ and an upper bound for the size of $\mathcal{M}'$ is $\sum_{P \subseteq AP} \sum_{j=1}^{|Q|} \sum_{i=1}^{k} (|Q| \cdot |M_i| \cdot |A|) = O(2^{|AP|} \cdot |Q|^2 \cdot |\mathcal{M}| \cdot |A|)$.

Now, let $\mathcal{M}$ be a restricted Shsm. Given a machine $M_{(i,j,P)}$ of $\mathcal{M}'$, for each $p \in P$ there exists a vertex u in $\mathcal{M}$ such that u is an ancestor of the vertices of M_i and $p \in \mathit{trueAP}(u)$ (this trivially descends from the definition of $\mathcal{M}'$). From Definition 2, it follows that $P \cap \mathit{trueAP}_{\mathcal{M}}(in_i) = \emptyset$. Moreover, recall that $P \cup \mathit{trueAP}_{\mathcal{M}}(in_i) = L(q_j)$. Thus, given i and j, P is uniquely determined as the set of the atomic proposition belonging to $L(q_j) - \mathit{trueAP}_{\mathcal{M}}(in_i)$ and then at most one machine $M_{(i,j,P)}$ is in $\mathcal{M}$. Therefore, an upper bound on the size of $\mathcal{M}'$ is $\sum_{j=1}^{|Q|} \sum_{i=1}^{k} (|Q| \cdot |M_i| \cdot |A|) = O(|Q|^2 \cdot |\mathcal{M}| \cdot |A|)$. □

Given an Shsm $\mathcal{M}$, we define the language $\mathcal{L}(\mathcal{M})$ as the set of the infinite traces of $\mathcal{M}^F$ starting from its initial state. The language $\mathcal{L}(A)$ accepted by a Büchi automaton A is the set of all the infinite traces corresponding to paths visiting infinitely often a state of T. The following lemma shows that we can reduce the problem of checking for emptiness the intersection of $\mathcal{L}(\mathcal{M})$ and $\mathcal{L}(A)$ to solving the cycle detection problem on $\mathcal{M} \bigotimes A$ and the formula constituted by the only atomic proposition *trgt*.

Lemma 7. *Given an* Shsm $\mathcal{M}$ *and a Büchi automaton* A*, the problem* **Cycle** *of* $\mathcal{M} \bigotimes A$ *and formula trgt holds true if and only if* $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A) \neq \emptyset$.

Proof. Denote with $\mathcal{M}'$ the Shsm $\mathcal{M} \bigotimes A$. For the "only if" part observe that, by a simple induction on the length of runs, it is possible to show that each run of $\mathcal{M}'$ can be simulated by both $\mathcal{M}$ and A. More precisely, given a run of $\mathcal{M}'$, the corresponding run of $\mathcal{M}$ can be obtained by projecting for each state the first component of all its *vertices*, and the corresponding run of A can be obtained by projecting the second component of the *nodes* of each state. Also, observe that if a run of $\mathcal{M}'$ has a cycle over a state labeled with *trgt* we can construct a run of $\mathcal{M}'$ which definitely loops within such cycle by simply pumping it. Since each vertex $[u,q,j,p]$ is labeled with *trgt* if and only if q is accepting, we can simulate the resulting run with an accepting run of A, and therefore, $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A) \neq \emptyset$ holds.

Consider now the "if" part. Let w be a trace over sets of atomic propositions. By a simple induction on the length of w, we can show that for each run r_A of A and $r_{\mathcal{M}}$ of $\mathcal{M}$ over w there is a run r of $\mathcal{M}'$ which simulates both r_A and $r_{\mathcal{M}}$. Moreover, if $w \in \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A)$, such run r also loops forever within a cycle which has a state labeled with *trgt* (for the above observation on the labeling of $\mathcal{M}'$ vertices). Therefore, the lemma is proved. □

Since the formula consisting of the sole atomic proposition *trgt* has only a partial evaluation, that is *trgt* itself, from Theorem 3 and the above result we get the following lemma.

Lemma 8. *There exists an algorithm checking whether* $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A) = \emptyset$ *in time linear in the size of* $\mathcal{M} \bigotimes A$.

As a consequence of the above lemmas, we obtain an algorithm to solving the Ltl model checking for Shsms.

Theorem 5. *The* Ltl *model checking problem on an* Shsm $\mathcal{M}$ *and a formula* φ *can be solved in* $O(|\mathcal{M}| \cdot 16^{|\varphi|})$ *time. Moreover, if* $\mathcal{M}$ *is a restricted* Shsm *the problem can be solved in* $O(|\mathcal{M}| \cdot 8^{|\varphi|})$ *time.*

Proof. From [19], we can construct a Büchi automaton $A_{\neg\varphi}$ of size $O(2^{|\varphi|})$ accepting the set $\mathcal{L}(A_{\neg\varphi})$ of all the sequences that do not satisfy φ, and thus, φ is satisfied on all paths of $\mathcal{M}$ if and only if $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A_{\neg\varphi})$ is empty. According to the construction given in [19], each state of the automaton $A_{\neg\varphi}$ corresponds to a set of sub-formulas of $\neg\varphi$ which are logically consistent with each other, and such that each accepting run rewrites the input sequence w with the sets of φ sub-formulas which are satisfied at each position of w. Therefore, the initial states of $A_{\neg\varphi}$ correspond to all the sets of consistent sub-formulas of $\neg\varphi$ containing $\neg\varphi$. From $A_{\neg\varphi}$ we can construct a Büchi automaton A' with a single initial state such that $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A') = \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A_{\neg\varphi})$: we just add to $A_{\neg\varphi}$ a new state (the initial state of A') along with the transitions from it which simulate the transitions from the initial states of $A_{\neg\varphi}$ which are labeled with the same atomic propositions as the initial node of $\mathcal{M}$. (Observe that we cannot just collapse all the initial states of $A_{\neg\varphi}$ because of the labeling on the states.) From Lemma 6, we can construct $\mathcal{M} \bigotimes A'$, whose size is $O(|Q|^2 \cdot |\mathcal{M}| \cdot |A_{\neg\varphi}| \cdot 2^{|AP|}) = O(|\mathcal{M}| \cdot 16^{|\varphi|})$ (since $|Q| = |A_{\neg\varphi}| = O(2^{|\varphi|})$ and $2^{|AP|} \leqslant 2^{|\varphi|}$). Moreover, this size reduces to $O(|Q|^2 \cdot |\mathcal{M}| \cdot |A_{\neg\varphi}|) = O(|\mathcal{M}| \cdot 8^{|\varphi|})$, when $\mathcal{M}$ is a restricted Shsm. Hence, by Lemma 8 we obtain the theorem. □

The best known upper bound on the time complexity of Ltl model checking on Hsms is $O(|\mathcal{M}| \cdot 8^{|\varphi|})$ (see [6]). Thus, from the above theorem and since restricted Shsms can be exponentially more succinct than Hsms (see Proposition 1), we obtain that Ltl model checking could be solved more efficiently if we choose to model the system as a restricted Shsm instead of as a Hsm.

5.2. Ctl *model checking*

To solve Ctl model checking for Shsm we reduce it to the same problem for Hsm solved in [6].

Theorem 6 ([16]). *The* CTL *model checking of an* HSM $\mathcal{M}$ *can be solved in* $O(|\mathcal{M}| \cdot 2^{|\varphi| \cdot d})$, *where d is the maximum number of exit nodes of* $\mathcal{M}$.

We fix an SHSM $\mathcal{M} = (M_1, M_2, \ldots, M_k)$ and a CTL formula φ. Let AP_φ be the set of atomic propositions that occur in φ. The first step of our algorithm consists of constructing an HSM $\mathcal{M}_\varphi$ such that $\mathcal{M}_\varphi^F$ is isomorphic to $\mathcal{M}^F$.

Let $index : \{1, \ldots, k\} \times 2^{AP_\varphi} \to \{1, \ldots, k\, 2^{|AP_\varphi|}\}$ be a bijection such that $index(i,P) < index(j,P')$ whenever $i < j$. Clearly, *index* maps (i,P) into a strictly increasing sequence of consecutive naturals starting from 1. For a machine $M_i = (V_i, in_i, \text{OUT}_i, trueAP_i, expn_i, E_i)$, $1 \le i \le k$ and $P \subseteq AP_\varphi$, define M_i^P as the machine $(V_i^P, in_i^P, \text{OUT}_i^P, trueAP_i^P, expn_i^P, E_i^P)$ where:

- $V_i^P = \{u^P \mid u \in V_i\}$, and $\text{OUT}_i^P = \{u^P \mid u \in \text{OUT}_i\}$;
- $trueAP_i^P(u^P) = trueAP_i(u)$ if u is a node and $trueAP_i^P(u^P) = \emptyset$, otherwise;
- $expn_i^P(u) = 0$ if u is a node and $expn_i^P(u) = index(expn_i(u), P \cup trueAP_i(u))$, otherwise;
- $E_i^P = \{(u^P, v^P) \mid (u,v) \in E_i\} \cup \{((u^P, z^{P \cup trueAP_i(u)}), v^p) \mid ((u,z),v) \in E_i\}$.

Let $k' = k\, 2^{|AP_\varphi|}$. We define $\mathcal{M}_\varphi$ by the tuple of machines $(M'_1, \ldots, M'_{k'})$ such that for $j = 1, \ldots, k'$, $M'_j = M_i^P$ where $j = index(i,P)$. From the definition of M_i^P it is simple to verify that $\mathcal{M}_\varphi$ is a HSM and $|\mathcal{M}_\varphi|$ is $O(|\mathcal{M}|\, 2^{|AP_\varphi|})$. Moreover, $\mathcal{M}_\varphi^F$ and $\mathcal{M}^F$ are identical up to a renaming of the states. Therefore, from Theorem 6, we get the following theorem (where $\mathcal{M}$ is an SHSM, φ is a formula, AP_φ is the set of atomic propositions that occur in φ, and d is the maximum number of exit nodes of a machine of $\mathcal{M}$).

Theorem 7.

The CTL *model checking of* SHSMs *can be solved in* $O(|\mathcal{M}|\, 2^{|\varphi|\, d + |AP_\varphi|})$ *time*.

6. Conclusions

In this paper, we have introduced the scope-dependent hierarchic state machines as a succinct model of systems and analyzed the computational complexity of verification.

We have considered SHSMs composed of machines M_i's with a single entry. We can generalize the obtained results to machines with multiple entries. The semantics of this model is the natural extension of that given for single-entries machines.

Given a multiple-entry SHSM $\mathcal{M}$, an equivalent single-entry SHSM $\mathcal{M}'$ can be obtained by replacing each machine M which has, say $m > 1$ entry nodes, with a set of m single-entry machines. Thus, each box expanding into M is replaced with m boxes, and edges and expansions are updated consistently. This translation causes a quadratic blow-up in the size of the model. More precisely, given a multi-entry SHSM $\mathcal{M}$ we can construct an equivalent single-entry SHSM $\mathcal{M}'$ of $O(e^2|\mathcal{M}|)$ size, where e is the maximum number of entries to each machine of $\mathcal{M}$. To solve the reachability and cycle-detection problems on multi-entry SHSMs, we can use such a construction along with our algorithms. This approach leads to time complexities that are quadratic in the maximum number of entries to a machine.

Slightly more efficient algorithms can be obtained by adapting our algorithms to work directly on multi-entry SHSMs. In fact, we can let our algorithms to visit each machine starting either from the entry nodes or the exit nodes depending on which are fewer in number. Thus, obtaining complexities that increases (with respect to the single-entry case) only by a factor θ^2, where θ is the maximum over all machines M of the minimum between the number of entry nodes and the number of exit nodes of M (see also [12]).

Scope properties are also interesting for software verification. An SHSM can be seen as an abstract model capturing the flow of control of a computer program, where each machine corresponds to a program routine. Then, the atomic propositions can be used to model predicates over the infinite states of the program (atomic propositions on our machines can be used to model the boolean variables of boolean programs [20]). In particular, scope properties could be used to succinctly express predicates which involve the global environment.

In our framework, the value of the atomic propositions can be checked along the executions but cannot be explicitly used to allow/disallow transitions. We could extend our model by allowing the edges to be guarded by boolean conditions over the atomic propositions and with the meaning that an edge can be crossed if the truth values of the atomic propositions in the current state satisfy the condition.

Such an extension has been already considered in verification for finite state automata, called *finite state automata with boolean variables* (see [21]). It would be interesting to study the combined effects of hierarchy and boolean variables on the succinctness of system models and the complexity of the main decision problems. As a first interesting property, it is easy to see that, for the resulting model, the transformation from multiple entries to a single entry would be linear.

Acknowledgment

We thank P. Madhusudan for helpful discussions.

References

[1] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching-time temporal logic, in: D. Kozen (Ed.), Logic of Programs, Lecture Notes in Computer Science, vol. 131, 1981, pp. 52–71.
[2] E.M. Clarke, R.P. Kurshan, Computer-aided verification, IEEE Spectrum, 33 (6) (1996) 61–67.
[3] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, 1998.
[4] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.
[5] B. Selic, G. Gullekson, P.T. Ward, Real-Time Object Oriented Modeling and Design, John Wiley, 1994.
[6] R. Alur, M. Yannakakis, Model checking of hierarchical state machines, ACM Trans. Program. Lang. Syst. 23 (3) (2001) 273–303.
[7] R. Alur, S. Kannan, M. Yannakakis, Communicating hierarchical state machines, in: J. Wiedermann, P. van Emde Boas, M. Nielsen (Eds.), ICALP, Lecture Notes in Computer Science, vol. 1644, Springer, 1999, pp. 169–178.
[8] S. La Torre, M. Napoli, M. Parente, G. Parlato, Hierarchical and recursive state machines with context-dependent properties, in: J.C.M. Baeten, J.K. Lenstra, J. Parrow, G.J. Woeginger (Eds.), ICALP, Lecture Notes in Computer Science, vol. 2719, Springer, 2003, pp. 776–789.
[9] A. Pnueli, The temporal logic of programs, In: FOCS, IEEE, 1977, pp. 46–57.
[10] R. Alur, R. Grosu, M. McDougall, Efficient reachability analysis of hierarchical reactive machines, in: E. Allen Emerson, A. Prasad Sistla (Eds.), CAV, Lecture Notes in Computer Science, vol. 1855, Springer, 2000, pp. 280–295.
[11] R. Alur, M. McDougall, Z. Yang, Exploiting behavioral hierarchy for efficient model checking, in: E. Brinksma, K.G. Larsen (Eds.), CAV, Lecture Notes in Computer Science, vol. 2404, Springer, 2002, pp. 338–342.
[12] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T.W. Reps, M. Yannakakis, Analysis of recursive state machines, ACM Trans. Program. Lang. Syst. 27 (4) (2005) 786–818.
[13] R. Alur, K. Etessami, P. Madhusudan, A temporal logic of nested calls and returns, in: K. Jensen, A. Podelski (Eds.), TACAS, Lecture Notes in Computer Science, vol. 2988, Springer, 2004, pp. 467–481.
[14] R. Lanotte, A. Maggiolo-Schettini, A. Peron, Structural model checking for communicating hierarchical machines, in: J. Fiala, V. Koubek, J. Kratochvíl (Eds.), MFCS, Lecture Notes in Computer Science, vol. 3153, Springer, 2004, pp. 525–536.
[15] L. Bozzelli, S. La Torre, A. Peron, Verification of well-formed communicating recursive state machines, in: E. Allen Emerson, K.S. Namjoshi (Eds.), VMCAI, Lecture Notes in Computer Science, vol. 3855, Springer, 2006, pp. 412–426.
[16] R. Alur, S. La Torre, P. Madhusudan, Modular strategies for recursive game graphs, in: H. Garavel, J. Hatcliff (Eds.), TACAS, Lecture Notes in Computer Science, vol. 2619, Springer, 2003, pp. 363–378.
[17] R. Alur, S. La Torre, P. Madhusudan, Modular strategies for infinite games on recursive graphs, in: W.A. Hunt Jr., , F. Somenzi (Eds.), CAV, Lecture Notes in Computer Science, vol. 2725, Springer, 2003, pp. 67–79.
[18] E.A. Emerson, Temporal and modal logic, in: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), 1990, pp. 995–1072.
[19] M.Y. Vardi, P. Wolper, Automata-theoretic techniques for modal logics of programs, J. Comput. Syst. Sci. 32 (2) (1986) 183–221.
[20] T. Ball, S.K. Rajamani, Bebop: A symbolic model checker for boolean programs, in: K. Havelund, J. Penix, W. Visser (Eds.), SPIN, Lecture Notes in Computer Science, vol. 1885, Springer, 2000, pp. 113–130.
[21] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, P. Schnoebelen, Model-Checking Techniques and Tools, Springer, 2001.