

Hierarchical and Recursive State Machines with Context-Dependent Properties [★]

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

Dipartimento di Informatica e Applicazioni
Università degli Studi di Salerno

Abstract. Hierarchical and recursive state machines are suitable abstract models for many software systems. In this paper we extend a model recently introduced in literature, by allowing atomic propositions to label all the kinds of vertices and not only basic nodes. We call the obtained models *context-dependent hierarchical/recursive state machines*. We study on such models cycle detection, reachability and LTL model-checking. Despite of a more succinct representation, we prove that LTL model-checking can be done in time linear in the size of the model and exponential in the size of the formula, as for standard LTL model-checking.

Reachability and cycle detection become NP-complete, and if we place some restrictions on the representation of the target states, we can decide them in time linear in the size of the formula and the size of the model.

Keywords: Model Checking, Automata, Temporal Logic.

1 Introduction

Due to their complexity, the verification of the *correctness* of many modern digital systems is infeasible without suitable automated techniques. Formal verification has been very successful and recent results have led to the implementation of powerful design tools (see [CK96]). In this area one of the most successful techniques has been *model checking* [CE81]: a high-level specification is expressed by a formula of a logic and this is checked for fulfillment on an abstract model (state machine) of the system. Though model checking is linear in the size of the model, it is computationally hard since the model generally grows exponentially with the number of variables used to describe a state of 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. Hierarchical structures are also typical of object-oriented paradigms [BJR97,RBP⁺91,SGW94]. We consider systems modeled as *hierarchical finite state machines*, that is, finite state machines where a vertex can either expand to another hierarchical state machine or be a basic vertex (in the former case we call the vertex a *supernode*, in the latter simply a *node*).

The model we consider in this paper generalizes instead the model studied in [AY01]. There the authors consider the model checking on Hierarchical State Machines (HSM)

[★] This research was partially supported by the MIUR in the framework of the project “Metodi Formali per la Sicurezza e il Tempo” (MEFISTO) and MIUR grant 60% 2002.

where only the nodes are labeled with atomic propositions (AP). We relax this constraint and thus we allow to associate atomic propositions also with vertices that expand to a machine. Expanding a supernode v to a machine M , all vertices of M *inherit* the atomic propositions of v (*context*), so that different vertices expanding to M can place M into different contexts. For this reason, we call such a model a *hierarchical state machine with context-dependent properties* (in the following denoted by **Context-dependent Hierarchical State Machine**). The semantics of a **CHSM** is given by the corresponding natural flat model which is a Kripke structure.

By allowing this more general labeling, for a given system it is possible to obtain very succinct abstract models. In the following example, we show that the gain of succinctness can be exponential compared to the models used in [AY01]. Consider a digital clock with hours, minutes, and seconds. We can construct a hierarchical finite state machine \mathcal{M} composed of three machines M_1 , M_2 , and M_3 such that the supernodes of M_3 expands to M_2 and the supernodes of M_2 expands to M_1 . Machine M_1 is a chain of nodes. Machines M_2 and M_3 are chains of supernodes except for the initial and the output vertices that are nodes. In M_3 each supernode corresponds to a hour and they are linked accordingly to increasing time. Analogously, M_2 models minutes and M_1 seconds. A flat model for the digital clock has at least $24 \cdot 60 \cdot 60 = 86,400$ vertices, while the above hierarchical model has only $24 + 60 + 60 + 6 = 150$ vertices (6 are simply initial and output nodes). Assume that we are interested in checking properties that refer to a precise time expressed in hours, minutes and seconds. Clearly, it is not sufficient to label only the nodes (we would be able to capture only that an event happens at a certain second, but we would have no clue of the actual hour and minute). In the model defined in [AY01], at least 86,400 nodes are needed, that is, there would be no gain with respect to a minimal flat model. In our model, we are able to label each supernode in M_3 with atomic propositions encoding the corresponding hour. Analogously we can use atomic propositions to encode minutes and seconds on M_2 and M_1 , respectively. This way, each state of the corresponding flat model is labeled with the encoding of a hour, a minute and a second in a day and vertices are linked by increasing time.

A simple way of analyzing hierarchical systems is first to flatten them into equivalent non-hierarchical systems and then apply existing verification techniques on finite state systems. The drawback of such an approach is that the size of the flat system can be exponential in the hierarchical depth. In many recent papers, it has been shown that it is possible to reduce the complexity growth caused by handling large systems, by performing verification in a hierarchical manner [AGM00,AG00,BLA⁺99,AY01]. We follow this approach and study on **CHSMs** standard decision problems which are related to system verification, such as reachability, cycle detection, and model checking. In this paper, we also consider **Context-dependent Recursive State Machines (CRSM)** which generalize **CHSMs** by allowing recursive expansions and we study on them the verification-related problems listed above. Recursive generalizations of the hierarchical model presented in [AY01] are studied in [AEY01,BGR01]. Recursive machines can be used to model the control flow of programs with recursive calls and thus are suitable for abstracting the behavior of reactive software systems.

Results. Given a transition system, a state s and a set of *target states* T , (usually expressed by a propositional boolean formula), the *reachability problem* is the problem of determining whether a state of T can be reached from s on a run 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*).

We prove that reachability on **CRSMs** is NP-complete, and NP-hardness still holds if we restrict to **CHSMs**. We then give an algorithm to decide reachability on **CRSMs** that runs in time linear in the size of the model and exponential in the size of the formula. Finally, given a **CHSM** \mathcal{M} , we show effective sufficient conditions for solving reachability in time linear in both the size of the formula and the size of the model. Let us remark that these conditions are satisfied when we consider an instance of the reachability problem where the model is given by a Hierarchical State Machine (HSM) as defined in [AY01].

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*: “some good thing will eventually happen”.

We also consider the model checking of LTL formulas on **CRSMs**. Given a set of atomic propositions AP , a *linear temporal logic* (LTL) formula is built up in the usual way from atomic propositions, the boolean connectives, the temporal operators *next* and *until*. An LTL formula is interpreted over an infinite sequence over 2^{AP} . A **CRSM** satisfies a formula φ if every run in the corresponding flat model satisfies φ . Given an LTL formula φ and a **CRSM** \mathcal{M} , the *model checking problem* for \mathcal{M} and φ is the problem to determine whether \mathcal{M} satisfies φ . We give a decision algorithm that runs in $O(|\mathcal{M}| \cdot 8^{|\varphi|})$ time for **CHSMs** and an algorithm in $O(|\mathcal{M}| \cdot 16^{|\varphi|})$ time for **CRSMs**. Our algorithms do not need to flatten the system and mainly consist of reducing the model checking problem to the emptiness problem of recursive Büchi automata [AEY01].

The rest of the paper is organized as follows. In the next section definitions and notation are given. The NP-completeness of the cycle detection and of the reachability problems is shown in section 3 (actually the proofs for the cycle detection problems are omitted in this version, due to the lack of space). In section 4 we give the linear time algorithms for **CHSMs** and **CRSMs**. In Section 5, we discuss the model checking of LTL formulas. We conclude with few remarks in Section 6.

2 Context-dependent State Machines

In this section we introduce the definitions and the notation we will use in the rest of the paper. We consider *Kripke structures*, that is, state-transition graphs where each state is labeled by a subset of a finite set of atomic propositions (AP). A *Context-dependent Recursive State Machine* (**CRSM**) over AP is a tuple $\mathcal{M} = (M_1, \dots, M_k)$ of Kripke structures with:

- a set of vertices N , split into disjoint sets N_1, \dots, N_k ; a set $IN = \{in_1, \dots, in_k\}$ of initial vertices, where $in_i \in N_i$, and a set of output vertices OUT split into OUT_1, \dots, OUT_k , with $OUT_i \subseteq N_i$;
- a mapping $expand : N \longrightarrow \{0, 1, \dots, k\}$ such that $expand(u) = 0$, for each $u \in IN \cup OUT$. We define the closure of $expand$, $expand^+ : N \longrightarrow 2^{\{0, 1, \dots, k\}}$, as: $h \in expand^+(u)$ if either $h = expand(u)$ or $u' \in N_{expand(u)}$ exists such that $h \in expand^+(u')$.
- the sets of edges E_i , for $1 \leq i \leq k$, such that each edge in E_i is either a pair (u, v) , with $u, v \in N_i$ and $expand(u) = 0$, or a triple $((u, z), v)$ with $z \in OUT_{expand(u)}$, and $u, v \in N_i$;

- a mapping $true : N \longrightarrow 2^{AP}$, such that $true(u) \cap true(v) = \emptyset$, for $v \in N_h, u \notin N_h$ and $h \in expand^+(u)$.

Informally, a **CRSM** is a collection of graphs which can call each other recursively. Each graph has an initial vertex and some output vertices. The mapping $expand$ gives the recursive-call structure. If $expand(u) = j > 0$, then the vertex u expands to the graph M_j and u is called a *supernode*; when $expand(u) = 0$ the vertex u is called a *node*. The mapping $true$ labels each vertex with a set of atomic propositions holding at that vertex. The *starting* node of a **CRSM** $\mathcal{M} = (M_1, \dots, M_k)$ is the initial node in_k of M_k . **The Semantics of CRSMs.** Every **CRSM** \mathcal{M} corresponds to a flat model \mathcal{M}^F which is a directed graph with (possibly infinite) vertices (*states*) labeled with atomic propositions. Informally speaking, the flat machine \mathcal{M}^F is obtained starting from M_k and iteratively replacing every supernode u in it with the graph $M_{expand(u)}$. The flat machine \mathcal{M}^F is defined as follows. A state of \mathcal{M}^F is a tuple $X = [u_1, \dots, u_m]$ where $u_1 \in N_k, u_{j+1} \in N_{expand(u_j)}$ for $j = 1, \dots, m-1$, and $expand(u_m) = 0$. State X is labeled by a set of atomic proposition $true(X)$, consisting of the union of $true(u_j)$, for $j = 1, \dots, m$. State $[in_k]$ is the initial state of \mathcal{M}^F . The set of transitions E is defined as follows. Let $X = [u_1, \dots, u_m]$ be a state with $u_m \in N_h$ and $u_{m-1} \in N_j$. Then, $(X, X') \in E$ provided that one of the following cases holds:

1. $(u_m, u') \in E_h, u' \in N_h$, and if $expand(u') = 0$ then $X' = [u_1, \dots, u_{m-1}, u']$, otherwise $X' = [u_1, \dots, u_{m-1}, u', in_l]$ for $l = expand(u')$.
2. $u_m \in OUT_h, ((u_{m-1}, u_m), u') \in E_j, u' \in N_j$, and if $expand(u') = 0$ then $X' = [u_1, \dots, u_{m-2}, u']$, otherwise $X' = [u_1, \dots, u_{m-2}, u', in_l]$ for $l = expand(u')$.

Let $[u_1, \dots, u_n]$ be a state of \mathcal{M}^F , a *prefix* of $[u_1, \dots, u_n]$ is u_1, \dots, u_i for $i \leq n$.

A *Context-dependent Hierarchic State Machine* (**CHSM**) is a **CRSM** such that $expand(u) < i$, for every $u \in N_i$. A **CHSM** is a collection of graphs which are organized to form a hierarchy and $expand$ gives the hierarchical structure. The graph M_k is clearly the *top-level* graph of the hierarchy, i.e., no vertices expand to it and, as for **CRSMs**, its initial node in_k is the starting node of the **CHSM**.

3 Reachability and cycle detection problems: computational complexity

In this section we discuss the computational complexity of the reachability and cycle detection problems for **CRSMs** and **CHSMs**. Given a **CRSM** $\mathcal{M} = (M_1, \dots, M_k)$ and a propositional boolean formula φ , the *reachability problem* is the problem of deciding if a path in \mathcal{M}^F exists from $[in_k]$ to a state X on which φ is satisfied. Analogously, the *cycle detection problem* is the problem of deciding if a cycle in \mathcal{M}^F exists containing a reachable state X on which φ is satisfied.

We prove that for **CRSMs** and **CHSMs** these decision problems are NP-complete by showing NP-hardness for **CHSMs** and giving nondeterministic polynomial-time algorithms for **CRSMs**.

Lemma 1. *Reachability and cycle detection for CHSMs are NP-hard.*

Proof : We give a reduction in linear time with respect to the size of the formula from the satisfiability problem SAT. Given a boolean formula φ over the variables x_1, \dots, x_m ,

we construct a **CHSM** $\mathcal{M} = (M_1, M_2, \dots, M_m)$ over $AP = \{P_1, P_2, \dots, P_m\}$, as follows. Each graph M_i has four vertices $in_i, p_i, notp_i, out_i$ forming a chain. 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. Since an atomic proposition P_i does not label vertices in graphs other than M_i , this labeling implicitly corresponds to assigning $\neg P_i$ to $notp_i$. Vertices p_i and $notp_i$, for $i > 1$, are supernodes which expand into M_{i-1} , and p_1 and $notp_1$ are instead nodes.

Thus there are 2^m states of \mathcal{M}^F of type $[u_1, \dots, u_m]$ such that $u_{m-i+1} \in \{p_i, notp_i\}$ for $i = 1, \dots, m$, and it is easy to verify that all these states are reachable from $[in_m]$. Clearly, given a truth assignment ν of x_1, \dots, x_m , a state X of \mathcal{M}^F exists such that ν assigns TRUE to x_i if and only if p_i occurs in X and, in turns, if and only if $P_i \in true(X)$. Thus a reachable state X of \mathcal{M}^F exists whose labeling corresponds to a truth assignment fulfilling φ if and only if φ is satisfiable.

By definition of the cycle detection problem, checking for the existence of a cycle containing a state on which φ is satisfied requires to check for reachability first. Thus, NP-hardness is inherited from reachability. \square

To prove membership to NP of the reachability on **CRSMs**, we need to consider a notion of connectivity of vertices in a **CRSM**. We say that a vertex $u \in N$ is *connected* if a reachable state $[u_1, \dots, u_m]$ of \mathcal{M}^F exists, where $u = u_i$ for some $i = 1, \dots, m$. Observe that the starting node in_k is clearly connected and a vertex $u \in N_j$ is connected if and only if in_j is connected and a path π in M_j from in_j to u exists, such that if π goes through an edge $((v, z), v') \in E_j$ then z is a connected vertex (recall that $z \in \text{OUT}_{\text{expand}(v)}$). From this the following proposition holds.

Proposition 1. *A state $[u_1, \dots, u_m]$ of \mathcal{M}^F is reachable if and only if all the vertices u_i , for $i = 1, \dots, m$, are connected.*

The above observation suggests also an algorithm to determine in linear time the connected vertices. We omit the proof of this result which is given by a rather simple modification of a depth-first search on a graph (see also [AEY01]).

Proposition 2. *Given a **CRSM** \mathcal{M} , the set of connected vertices of \mathcal{M} can be determined in $O(|\mathcal{M}|)$.*

To prove membership to NP of the reachability on **CRSMs**, we need to prove the following technical lemma. Notice that this lemma is not needed for **CHSMs**, where the number of supernodes that compose a state of \mathcal{M}^F is bounded from above by the number of component graphs.

Lemma 2. *Given a **CRSM** \mathcal{M} , for each state $X = [u_1, \dots, u_m]$ such that $m > n^2 + 1$, where n is the number of supernodes of \mathcal{M} , a state $X' = [u'_1, \dots, u'_{m'}]$ exists such that $m' < m$ and $true(X) = true(X')$. Moreover, if X is reachable then also X' is reachable.*

Proof : Consider a sequence $v_1, \dots, v_h \in N$. We say that a sub-sequence $v_i \dots v_j$, $1 \leq i < j$, is a *cycle* if $v_i = v_j$. Moreover, we say that a cycle $v_i \dots v_j$, is *erasable* if $\{v_{i+1}, \dots, v_j\} \subseteq \{v_1, \dots, v_i\}$. It is easy to verify that for a sequence $u_1 \dots u_m$ such that $X = [u_1, \dots, u_m]$ is a state of \mathcal{M}^F and $u_i \dots u_j$ is a cycle, we have that $X' = [u_1, \dots, u_i, u_{j+1}, \dots, u_m]$ is a state of \mathcal{M}^F and if $u_i \dots u_j$ is also erasable then $true(X) = true(X')$. Moreover, by Proposition 1, if X is reachable then also X' is reachable.

To conclude the proof we only need to show that for each state $X = [u_1, \dots, u_m]$ such that $m > n^2 + 1$, where n is the number of supernodes in \mathcal{M} , $u_1 \dots u_m$ contains

an erasable cycle. Notice that $m > n^2 + 1$ implies that a supernode u exists, occurring at least $(n + 1)$ times in $u_1 \dots u_m$. Suppose $u_1 \dots u_m = \alpha_0 u \alpha_1 u \dots \alpha_n u \beta$, where each α_i does not contain occurrences of u . A cycle $u \alpha_i u$ is not erasable only if it contains a supernode that is not in $\alpha_0 u \dots \alpha_{i-1} u$. By a simple count, if $\alpha_0 u \dots \alpha_{n-1}$ does not contain erasable cycles, then all supernodes occur in it. Thus, $u \alpha_n u$ is erasable. \square

Now, we can prove membership to NP of the reachability and the cycle detection problems on **CRSMs**.

Lemma 3. *Reachability and cycle detection for **CRSMs** are decidable in nondeterministic polynomial-time.*

Proof : Consider the instance of the reachability problem given by a **CRSM** \mathcal{M} and a propositional boolean formula φ . By Proposition 2 we can determine in $O(|\mathcal{M}|)$ time the set of the connected vertices, and then, given a state X of \mathcal{M}^F , by Proposition 1 we can check if X is reachable in $O(|\mathcal{M}| + |X|)$ time. Verifying the fulfillment of φ on X takes $O(|\varphi| + |X|)$ time. Moreover, by Lemma 2 we need only to consider states $X = [u_1, \dots, u_m]$ for $m \leq n^2 + 1$, where n is the number of supernodes of \mathcal{M} . Thus, we can conclude that the reachability problem on **CHSMs** is in NP. \square

By Lemmas 1 and 3 we have the following theorem.

Theorem 1. *Reachability and cycle detection for **CRSMs** (**CHSMs**) are NP-complete.*

4 Efficient solutions to reachability and cycle detection problems.

In this section, we give a linear time algorithm that solves reachability and cycle detection problems for **CHSMs** which are related to target sets by a particular condition (specified later). As a corollary we get three consequences: first the results regarding reachability and cycle detection for the model considered in [AY01] are obtained as particular cases, second we characterize a class of formulas guaranteeing that the algorithm works correctly and finally we show that the algorithm works also for DNF formulas, thus obtaining a general solution for any formula with a tight worst case running time of $O(|\mathcal{M}| \cdot 2^{|\varphi|})$.

Finally, we give a linear time reduction from the reachability problem on **CRSMs** for DNF formulas to the corresponding problem on **CHSMs**, thus the above general solution still holds for **CRSMs**.

Consider now **CHSMs**. Clearly a propositional formula φ can be evaluated in a state X of \mathcal{M}^F by instantiating to TRUE the variables corresponding to the atomic propositions in $true(X)$ and to FALSE all the others. Now we wish to evaluate φ without constructing the graph \mathcal{M}^F , to this aim we use a greedy approach in a top-down fashion on the hierarchy: at each supernode we instantiate as many variables as possible. By traversing the hierarchy in a top-down fashion, once a node is reached, φ can only *partially* evaluated. On a supernode u of a **CHSM** all the variables instantiated to TRUE correspond to the atomic propositions in $true(u)$. Determining the variables to instantiate to FALSE is not so immediate. We define $AP(h)$ as the union of the sets labeling either the vertices in N_h or those having an ancestor in N_h , that is, $AP(h) = \bigcup_{v \in N_h} (true(v) \cup AP(expand(v)))$ where $AP(0) = \emptyset$. Moreover, for $u \in N_h$, we define the set $false(u)$ as $AP(h) \setminus (true(u) \cup AP(expand(u)))$. This set contains the atomic propositions that can be instantiated to FALSE at u , since a proposition $p \in false(u)$

Algorithm Reachability (\mathcal{M}, φ) return(Reach (M_k, φ)).
Function Reach (M_h, φ) VISITED[h] \leftarrow <i>MARK</i> ; foreach $u \in N_h$ do $\varphi' = \text{Eval}(\varphi, u)$; if ($\varphi' == \text{TRUE}$) then return TRUE ; if ($\varphi' == \text{FALSE}$) then continue ; if ($\text{expand}(u) > 0$) AND ($\text{VISITED}[\text{expand}(u)] = \text{MARK}$) then if Reach ($M_{\text{expand}(u)}, \varphi'$) then return TRUE ; endfor return FALSE ;

Fig. 1. Algorithm Reachability.

if and only if $p \notin \text{true}(X)$, for every state X of \mathcal{M}^F having the supernode u as a component. It is easy to see that the sets $\text{false}(u)$, $u \in N$, can be preprocessed in time $O(|\mathcal{M}|)$, by visiting \mathcal{M} in a bottom-up way.

For a propositional boolean formula φ we denote by $\text{Eval}(\varphi, u)$ the formula obtained by instantiating φ with $\text{true}(u)$ and $\text{false}(u)$. We generalize this notation to sequences of vertices defining $\text{Eval}(\varphi, u_1, \dots, u_i)$ as $\text{Eval}(\text{Eval}(\varphi, u_1), u_2, \dots, u_i)$. Finally, we will denote by $AP(\varphi)$ the set of atomic propositions corresponding to φ variables.

We consider a condition relating a **CHSM** \mathcal{M} and a target set specified by a formula φ asserting that "when two supernodes expand to the same graph, then any partial evaluation of φ ending on them coincides". Formally, the condition is as follows:

Condition 1 *Let x_1, \dots, x_i and y_1, \dots, y_j be two prefixes of \mathcal{M}^F states such that $\text{expand}(x_i) = \text{expand}(y_j)$. If neither $\text{Eval}(\varphi, x_1, \dots, x_i)$ nor $\text{Eval}(\varphi, y_1, \dots, y_j)$ is one of the constants $\{\text{TRUE}, \text{FALSE}\}$, then $\text{Eval}(\varphi, x_1, \dots, x_i) = \text{Eval}(\varphi, y_1, \dots, y_j)$.*

When reachability and cycle detection become tractable.

Theorem 2. *The reachability and cycle detection problems on a **CHSM** \mathcal{M} and a formula φ satisfying Condition 1 are decidable in time $O(|\mathcal{M}| \cdot |\varphi|)$.*

Proof : Consider a **CHSM** $\mathcal{M} = (M_1, \dots, M_k)$ and without loss of generality assume that all the vertices of \mathcal{M} are connected (see Proposition 2). Algorithm **Reachability**(\mathcal{M}, φ) (Figure 1), returns **TRUE** if and only if φ is evaluated to **TRUE** on a reachable state of \mathcal{M}^F . The function **Reach** uses a global array VISITED (initially unmarked in all the positions) to mark the visited graphs M_h . For each node u of M_h , φ is evaluated on it according to $\text{true}(u)$ and $\text{false}(u)$, call φ' the returned formula. If φ' evaluates **TRUE** on u , then **Reach** stops returning **TRUE**. (and the main algorithm stops too returning **TRUE**). If φ' evaluates **FALSE**, another vertex of M_h which has not yet been explored is processed. In case u is a supernode and $M_{\text{expand}(u)}$ has never been visited, then the function is called on the graph $M_{\text{expand}(u)}$ and φ' . Now note that Condition 1 assures that it is not necessary to visit a graph M_h more than once, thus the overall complexity of the algorithm is linear in $|\mathcal{M}|$ and $|\varphi|$ and clearly returns **TRUE** if and only if a node X in \mathcal{M}^F exists on which φ is **TRUE**. \square

It is easy to see that given any formula φ and a Hierarchical State Machine (HSM) introduced in [AY01] (where only nodes are labeled with the mapping true, see the introduction), Condition 1 always holds, thus the linear time solutions for the reachability and cycle detection problems for HSM given in that paper are here obtained as particular cases.

Now we present a characterization of formulas for which Theorem 2 holds. A propositional boolean formula φ is said to be in \mathcal{M} -normal form if $\varphi = \varphi_1 \wedge \dots \wedge \varphi_m$ and for every φ_i and for every vertex u of \mathcal{M} it holds that either $AP(\varphi_i) \cap (\text{true}(u) \cup \text{false}(u)) = \emptyset$ or $AP(\varphi_i) \cap (\text{true}(u) \cup \text{false}(u)) = AP(\varphi_i)$. It is easy to see that also in this case Condition 1 holds.

Theorem 2 can be generalized for a finite disjunction of formulas satisfying Condition 1. Since a conjunction of literals is in \mathcal{M} -normal form, for all possible \mathcal{M} , then this generalization can be applied to DNF formulas. Thus, as any formula φ can always be transformed in a DNF formula, we have an algorithm for reachability and cycle detection problems whose worst case running time is $O(|\mathcal{M}| \cdot \text{DNF}(\varphi))$, where $\text{DNF}(\varphi)$ is the cost of the transformation of φ in Disjunctive Normal Form. All this yields a tight upper bound of $O(|\mathcal{M}| \cdot 2^{|\varphi|})$.

Reachability and cycle detection are also tractable on **CRSMs** if we restrict to formulas in disjunctive normal form as shown in the following theorem.

Theorem 3. *Reachability and cycle detection problems for a **CRSM** \mathcal{M} and a formula φ in DNF are decidable in time $O(|\mathcal{M}| \cdot |\varphi|)$.*

Proof : Consider a **CRSM** \mathcal{M} and a DNF formula $\varphi = \psi_1 \vee \dots \vee \psi_m$ where each ψ_i is a conjunction of literals. Our algorithm consists of reducing in $O(|\varphi| \cdot |\mathcal{M}|)$ time the reachability problem for \mathcal{M} and ψ_i to the reachability problem for a **CHSM** $\bar{\mathcal{M}}$ and ψ_i , where size of $\bar{\mathcal{M}}$ is $O(|\mathcal{M}|)$. Then the result follows from Theorem 2.

Consider a disjunct clause ψ of φ . We simplify \mathcal{M} using the following two steps.

1. for each graph M_i , delete all the existing edges and insert an edge from in_i to any other connected vertex of M_i ;
2. if u is not an initial node and $\text{true}(u)$ contains an atomic proposition corresponding to a variable which is negated in ψ , then delete u from M_i .

This transformation can be performed in $O(|\psi| \cdot |\mathcal{M}|)$ time and preserves the reachability of the states of $\mathcal{M}^{\mathcal{F}}$ satisfying ψ , thanks to Proposition 2.

Now, define a supernode $u \in N_i$ as *recursively expansible* if $i \in \text{expand}^+(u)$ and a graph M_i as *recursively expansible* if it contains at least a recursively expansible supernode. We define the equivalence relation \approx on the indices of recursively expansible graphs: $i \approx j$ if and only if vertices $u \in N_i$ and $v \in N_j$ exist such that $i \in \text{expand}^+(v)$ and $j \in \text{expand}^+(u)$. We want to define a **CHSM** $\bar{\mathcal{M}} = (\bar{M}_1, \bar{M}_2, \dots, \bar{M}_{k'})$ such that $\bar{\mathcal{M}}$ has a component graph for each equivalence class of the relation \approx . Let $f : \{1, \dots, k\} \rightarrow \{1, \dots, k'\}$ be the function that maps each i to j such that i is in the equivalence class corresponding to \bar{M}_j .

For a graph M_i which is not recursively expansible (i.e., $[i] = \{i\}$), we define $\bar{M}_{f(i)}$ as M_i except for the mapping expand , since $\text{expand}_{\bar{\mathcal{M}}}(u) = f(\text{expand}_{\mathcal{M}}(u))$. For a recursively expansible graph M_i we define $\bar{M}_{f(i)}$ as follows. All vertices $u \in N_j$ which are not recursively expansible, with $j \approx i$, are vertices of $\bar{M}_{f(i)}$, the edges between them in M_i are edges of $\bar{M}_{f(i)}$ as well and $\text{OUT}_{f(i)} = \bigcup_{j, j \approx i} \text{OUT}_j$. Moreover, we add a new initial node $\bar{in}_{f(i)}$ and insert edges from $\bar{in}_{f(i)}$ to all vertices in_j , $j \approx i$. For each

supernode u of $\bar{M}_{f(i)}$ we define $expand_{\bar{\mathcal{M}}}(u) = f(expand_{\mathcal{M}}(u))$. Let $S_{\mathcal{M}}(i)$ be the set of all recursively expansible vertices belonging to all graphs M_j such that $j \approx i$. We define $true_{\bar{\mathcal{M}}}(\bar{in}_{f(i)})$ as $true_{\bar{\mathcal{M}}}(in_j)$ for an arbitrary $j \approx i$, and for each vertex u of $\bar{M}_{f(i)}$, $true_{\bar{\mathcal{M}}}(u)$ as $\bigcup_{v \in S_{\mathcal{M}}(i)} true_{\mathcal{M}}(v) \cup true_{\mathcal{M}}(u)$ (note that no atomic proposition added in this way to the label of u corresponds to a variable which is negated in ψ).

Now observe that, by the above part 2 of the above simplification, if X is a state of \mathcal{M}^F satisfying ψ and Y is a state of $\bar{\mathcal{M}}^F$ such that $true_{\mathcal{M}}(X) \subseteq true_{\bar{\mathcal{M}}}(Y)$ and $true_{\bar{\mathcal{M}}}(Y) \setminus true_{\mathcal{M}}(X)$ does not contain an atomic proposition corresponding to a variable which is negated in ψ , then Y satisfies ψ as well. Since the initial simplification also preserves reachability, we have that if a reachable state of \mathcal{M}^F fulfilling ψ exists, then a state of $\bar{\mathcal{M}}^F$ fulfilling ψ also exists. Since by construction, states of $\bar{\mathcal{M}}^F$ corresponds to states of \mathcal{M}^F , the vice-versa also holds. \square

As a consequence of Theorem 3 and the arguments for **CHSMs** and DNF formulas, the following theorem holds.

Theorem 4. *The reachability and cycle detection problems on a **CRSM** \mathcal{M} and a propositional boolean formula φ are decidable in $O(|\mathcal{M}| \cdot 2^{|\varphi|})$ time.*

5 LTL Model Checking

Here we consider the verification problem of linear-time requirements, expressed by LTL-formulas [Pnu77]. We follow the automata theoretic approach to solving model checking [VW86]: given an LTL formula φ and a Kripke structure M , it is possible to reduce model checking to the emptiness problem of Büchi automata. To use this approach, we extend the Cartesian product between Kripke structures.

Given a transition graph with states labeled by subsets of atomic propositions and a state s , a *trace* is an infinite sequence $\alpha_1 \alpha_2 \dots \alpha_i \dots$ of labels of states occurring in a path starting from s . Moreover, given a **CRSM** \mathcal{M} , we define the language $\mathcal{L}(\mathcal{M})$ as the set of the traces of \mathcal{M}^F starting from its initial state. A Büchi automaton $A = (Q, q_1, \Delta, L, T)$ is a Kripke structure (Q, Δ, L) together with a set of accepting states T and a starting state q_1 . The language $\mathcal{L}(A)$ accepted by A is the set of the traces corresponding to paths visiting infinitely often a state of T .

Let $\mathcal{M} = (M_1, \dots, M_k)$ be a **CRSM** and $A = (Q, q_1, \Delta, L, T)$, for $Q = \{q_1, \dots, q_m\}$, be a Büchi automaton. Let $1 \leq i \leq k$, $1 \leq j \leq m$, and P be such that $P \subseteq AP$ and $P \cup true_{\mathcal{M}}(in_i) = L(q_j)$, we define the graphs $M_{(i,j,P)}$ as follows. Each $M_{(i,j,P)}$ contains vertices $[u, q, j, P]$ such that (u, q) belongs to the standard Cartesian product of M_i and A , and the labeling of q coincides with the labeling of u augmented with the atomic propositions that u inherits from its ancestors in a given context. The inherited set of atomic propositions is given by P . The property $P \cup true_{\mathcal{M}}(in_i) = L(q_j)$ assures that we consider only graphs $M_{(i,j,P)}$ whose initial vertex is compatible with the automaton state. Formally, we have:

- The set $N_{(i,j,P)}$ of the vertices of $M_{(i,j,P)}$ contains quadruples $[u, q, j, P]$, where $u \in N_i$, $q \in Q$, and
 - either $expand_{\mathcal{M}}(u) = 0$ and $L(q) = true_{\mathcal{M}}(u) \cup P$
 - or $expand_{\mathcal{M}}(u) = h > 0$ and $L(q) = true_{\mathcal{M}}(u) \cup true_{\mathcal{M}}(in_h) \cup P$.
- The initial vertex of $M_{(i,j,P)}$ is $[in_i, q_j, j, P]$ and the output nodes are $[u, q, j, P]$ for $u \in OUT_i$ and $q \in Q$;
- $M_{(i,j,P)}$ contains the following edges:

- $([u, q', j, P], [v, q'', j, P])$, with $(q', q'') \in \Delta$ and $(u, v) \in E_i$,
- $(([u, q_j', j, P], [z, q', j', P \cup \text{true}_{\mathcal{M}}(u)]), [v, q'', j, P])$, with $(q', q'') \in \Delta$, $((u, z), v) \in E_i$, and $L(q) = \text{true}_{\mathcal{M}}(u) \cup \text{true}_{\mathcal{M}}(\text{in}_h) \cup P$ for $\text{expand}_{\mathcal{M}}(u) = h$.

From the above definition we observe that if u is a supernode then the labeling of q has to match also with the labeling of $\text{in}_{\text{expand}_{\mathcal{M}}(u)}$ since $[u, q, j, P]$ is a supernode of \mathcal{M}' and one has to assure the correctness, with respect to the labeling, of its expansion. Note that when only the value of j varies, we have graphs which differ one each other only for the choice of the the initial vertex $[in_i, q_j, j, P]$. Moreover, the edges in $M_{(i,j,P)}$ are given by coupling the transitions (q', q'') of A with both kinds of edges (u, v) and $((u, z), v)$ in E_i . For $h = \text{expand}_{\mathcal{M}}(u)$, we have edges $(([u, q, j, P], [z, q', h, P \cup \text{true}_{\mathcal{M}}(u)]), [v, q'', j, P])$ for every $q \in Q$ such that $L(q) = \text{true}_{\mathcal{M}}(u) \cup \text{true}_{\mathcal{M}}(\text{in}_h) \cup P$. Thus, there might be as many as $|Q|$ edges, for every pair of edges $((u, z), v)$ and (q', q'') .

We can now define $\mathcal{M}' = \mathcal{M} \otimes A$ as a **CRSM** constituted by some of the graphs $M_{(i,j,P)}$, and defined inductively as follows:

- $M_{(k,1,\emptyset)}$ is the graph containing the starting node of \mathcal{M}' ;
- Let $M_{(i,j,P)}$ be a graph of \mathcal{M}' , and $[u, q_t, j, P]$ be a vertex of $M_{(i,j,P)}$.
 - If $\text{expand}_{\mathcal{M}}(u) = 0$ then $\text{expand}_{\mathcal{M}'}([u, q_t, j, P]) = 0$;
 - If $\text{expand}_{\mathcal{M}}(u) = h > 0$, and $P' = P \cup \text{true}_{\mathcal{M}}(u)$ then $M_{(h,t,P')}$ is a graph of \mathcal{M}' and $\text{expand}_{\mathcal{M}'}([u, q_t, j, P]) = \langle h, t, P' \rangle$ where $\langle h, t, P' \rangle$ denotes the index of $M_{(h,t,P')}$;
- $\text{true}_{\mathcal{M}'}([u, q, j, P]) = \text{true}_{\mathcal{M}}(u)$, for every $[u, q, j, P]$.

Observe that $\mathcal{M}' = \mathcal{M} \otimes A$ is a **CRSM** and if \mathcal{M} is a **CHSM**, then \mathcal{M}' is a **CHSM**, as well. To determine the size of \mathcal{M}' , first consider the size of each graph $M_{(i,j,P)}$. The number of the edges is bounded by the product of the number of edges in M_i and the number of transitions in A multiplied at most by m , since we have at most $|Q|$ edges 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 $(m \cdot |E_i| \cdot |A|)$. The size of \mathcal{M}' can be obtained now by counting the number of its component graphs.

Lemma 4. *Given a **CRSM** \mathcal{M} , $\mathcal{M}' = \mathcal{M} \otimes A$ is a **CRSM** that can be constructed in $O(m^2 \cdot |\mathcal{M}| \cdot |A| \cdot |2^{AP}|)$ time. Moreover, if \mathcal{M} is a **CHSM**, then \mathcal{M}' is a **CHSM** that can be constructed in $O(m^2 \cdot |\mathcal{M}| \cdot |A|)$ time.*

Proof : First recall that a graph $M_{(i,j,P)}$ of \mathcal{M}' has the property that $P \cup \text{true}_{\mathcal{M}}(\text{in}_i) = L(q_j)$. Therefore, P is the union of two disjoint sets P_1 and P_2 , such that P_1 is the set of the atomic propositions of $L(q_j)$ that do not belong to $\text{true}_{\mathcal{M}}(\text{in}_i)$, and $P_2 = P \cap \text{true}_{\mathcal{M}}(\text{in}_i)$ is a subset of $\text{true}_{\mathcal{M}}(\text{in}_i)$. Thus, for fixed values of i and j , P_1 is fixed and the number of different graphs $M_{(i,j,P)}$ is bounded above by the number of different subsets of $\text{true}_{\mathcal{M}}(\text{in}_i)$. Therefore, the size of \mathcal{M}' is bounded above by $\sum_{j=1}^m \sum_{i=1}^k (2^{|AP|} \cdot m \cdot |M_i| \cdot |A|)$.

Now, let \mathcal{M} be a **CHSM**. Given a graph $M_{(i,j,P)}$ of \mathcal{M}' , P is defined as the set of the propositions that the vertices of M_i inherit. Thus, $P \cap \text{true}_{\mathcal{M}}(u) = \emptyset$, for every vertex u of M_i and then $P \cap \text{true}_{\mathcal{M}}(\text{in}_i) = \emptyset$. Hence, in this case, P_2 is empty and then at most one graph $M_{(i,j,P)}$ exists for fixed values of i and j . Therefore, the size of \mathcal{M}' is bounded above by $\sum_{j=1}^m \sum_{i=1}^k (m \cdot |M_i| \cdot |A|)$. \square

The **CRSM** $\mathcal{M} \otimes A$ can be used to check for the emptiness of the language given by the intersection of $\mathcal{L}(\mathcal{M})$ and $\mathcal{L}(A)$, as shown in the following lemma.

Lemma 5. *There exists an algorithm checking whether $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A) = \emptyset$ in time linear in the size of $\mathcal{M}' = \mathcal{M} \otimes A$.*

Proof : First, observe that if we consider as set of final states the vertices $[u, q, h, P]$ such that $q \in T$, the **CRSM** \mathcal{M}' is a recursive Büchi automaton. Moreover, the set of the traces of \mathcal{M}'^F is the same as the set of traces of the Cartesian product of \mathcal{M}^F and A . Thus $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A) \neq \emptyset$ if and only $\mathcal{L}(\mathcal{M}') \neq \emptyset$. From [AEY01], for recursive Büchi automata with a single initial node for each graph, non-emptiness can be checked in linear time. \square

As a consequence of the above lemmas, we obtain an algorithm to solving the LTL model checking for **CRSMs**. Following the automata theoretic approach, one can construct a Büchi automaton $A_{\neg\varphi}$ of size $O(2^{|\varphi|})$ accepting the set $\mathcal{L}(A_{\neg\varphi})$ of the sequences which do not satisfy φ , and then 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. From Lemma 4, one can now construct $\mathcal{M} \otimes A_{\neg\varphi}$, whose size is $O(m^2 \cdot |\mathcal{M}| \cdot |A_{\neg\varphi}| \cdot 2^{|A^P|}) = O(|\mathcal{M}| \cdot 16^{|\varphi|})$ (since $m = |A_{\neg\varphi}| = O(2^{|\varphi|})$ and $2^{|A^P|} \leq 2^{|\varphi|}$). Moreover, this size reduces to $O(m^2 \cdot |\mathcal{M}| \cdot |A_{\neg\varphi}|) = O(|\mathcal{M}| \cdot 8^{|\varphi|})$, when \mathcal{M} is a **CHSM**. Hence, by Lemma 5 we obtain the main result of this section.

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

6 Discussion

We have proposed new abstract models for sequential state machines: the context-dependent hierarchical and recursive state machines. On these models we have studied reachability, cycle detection and the more general problem of model checking with respect to linear-time specifications. An interesting feature of **CHSMs** is that they allow very succinct representations of systems, and this comes substantially at no cost if compared to analogous hierarchical models studied in the literature. Moreover, we prove that for some particular formulas we improve the complexity of previous approaches.

Several extensions of the introduced models can be considered.

Our models are sequential. If we add concurrency to **CHSMs**, the computational complexity of the considered decision problems grows significantly (we recall that reachability in communicating hierarchical state machines is EXPSpace-complete [AKY99]). While for **CRSMs** with concurrency, reachability becomes undecidable since sequential **CRSMs** are as expressive as pushdown automata [AEY01, BGR01].

We have only considered models where a single entry node is allowed for each component machine. We can relax this limitation allowing multiple entry points. The semantics of this extension naturally follows from the semantics given for the single entry case. In the hierarchic setting, we can translate a multiple-entry **CHSM** \mathcal{M} into an equivalent single-entry **CHSM** \mathcal{M}' of size at most cubic in the size of \mathcal{M} . In fact, each component machine of \mathcal{M} can be replaced in \mathcal{M}' by multiple copies, each copy corresponding to an entry point and having as unique entry point the entry point itself. Expansions are redirected to the proper components in order to match the expansions in \mathcal{M} . Thus, supernodes may need to be replaced by multiple copies each pointing to the proper machine in \mathcal{M}' . If we apply this construction to a multiple-entry **CRSM**, the obtained single-entry **CRSM** does not satisfy the property $true(u) \cap true(v) = \emptyset$, for $v \in N_h, u \notin N_h$ and $h \in expand^+(u)$ (see definition of **CRSM**). This is a consequence of the fact that if

a machine of the multiple-entry **CRSM** can directly or indirectly call itself, then there are two copies of this machine that may call each other recursively. We recall that the above property is sufficient to ensure that Condition 1 holds for conjunctions of literals, and thus is crucial to obtain the results given in Section 4. However, it is possible to prove that Theorem 3 also holds for multiple-entry **CRSMs**. We leave the details of this proof to the full paper.

For modeling purposes it is useful to have variables over a finite domain that can be passed from a component to another. We can extend our models to handle input, output and local variables. Consider a component machine M with h_e entry nodes, h_x exit nodes, and h_t internal vertices. If M is equipped also with k_i input boolean variables, k_o output boolean variables, and k_l local boolean variables, we can model by a machine having $2^{k_i} \cdot h_e$ entry nodes, $2^{k_o} \cdot h_x$ exit nodes, and $2^{k_i+k_l+k_o} \cdot h_t$ internal vertices.

References

- [AEY01] R. Alur, K. Etessami, and M. Yannakakis. Analysis of recursive state machines. In *Proc. of the 13th International Conference on Computer Aided Verification, CAV'01*, LNCS 2102, pages 207–220. Springer, 2001.
- [AG00] R. Alur and R. Grosu. Modular refinement of hierarchic reactive machines. In *Proc. of the 27th Annual ACM Symposium on Principles of Programming Languages*, pages 390–402, 2000.
- [AGM00] R. Alur, R. Grosu, and M. McDougall. Efficient reachability analysis of hierarchical reactive machines. In *Computer Aided Verification, 12th International Conference*, LNCS 1855, pages 280–295. Springer, 2000.
- [AKY99] R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. In *Proc. of the 26-th International Colloquium on Automata, Languages and Programming, ICALP'99*, LNCS 1644, pages 169–178. Springer-Verlag, 1999.
- [AY01] R. Alur and M. Yannakakis. Model checking of hierarchical state machines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 23(3):273 – 303, 2001.
- [BGR01] M. Benedikt, P. Godefroid, and T. W. Reps. Model checking of unrestricted hierarchical state machines. In *Proc. of the 28th International Colloquium Automata, Languages and Programming, ICALP'01*, LNCS 2076, pages 652–666. Springer, 2001.
- [BJR97] G. Booch, I. Jacobson, and J. Rumbaugh. *Unified Modeling Language User Guide*. Addison Wesley, 1997.
- [BLA⁺99] G. Behrmann, K.G. Larsen, H.R. Andersen, H. Hulgaard, and J. Lind-Nielsen. Verification of hierarchical state/event systems using reusability and compositionality. In *Proc. of the Tools and Algorithms for the Construction and Analysis of Systems, TACAS'99*, LNCS 1579, pages 163–177. Springer, 1999.
- [CE81] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. of Workshop on Logic of Programs*, LNCS 131, pages 52 – 71. Springer-Verlag, 1981.
- [CK96] E.M. Clarke and R.P. Kurshan. Computer-aided verification. *IEEE Spectrum*, 33(6):61 – 67, 1996.
- [Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th IEEE Symposium on Foundations of Computer Science*, pages 46 – 77, 1977.
- [RBP⁺91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented Modeling and Design*. Prentice-Hall, 1991.
- [SGW94] B. Selic, G. Gullekson, and P.T. Ward. *Real-time object oriented modeling and design*. J. Wiley, 1994.
- [VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:182 – 211, 1986.