On Model Checking for Non-Deterministic Infinite-State Systems *

E. Allen Emerson   Kedar S. Namjoshi

Department of Computer Sciences,
The University of Texas at Austin.

Abstract

We demonstrate that many known algorithms for model checking infinite-state systems can be derived uniformly from a reachability procedure that generates a "covering graph", a generalization of the Karp-Miller graph for Petri Nets. Each node of the covering graph has an associated non-empty set of reachable states, which makes it possible to model check safety properties of the system on the covering graph. For systems with a well-quasi-ordered simulation relation, each infinite fair computation has a *finite* witness, which may be detected using the covering graph and combinatorial properties of the specific infinite state system. These results explain many known decidability results in a simple, uniform manner. This is a strong indication that the covering graph construction is appropriate for the analysis of infinite state systems. We also consider the new application domain of parameterized broadcast protocols, and indicate how to apply the construction in this domain. This application is illustrated on an invalidation-based cache coherency protocol, for which many safety properties can be proved fully automatically for an *arbitrary* number of processes.

1 Introduction

Model Checking refers to a collection of algorithms for automatically checking temporal properties of *finite* state systems [CE 81, QS 82, CES 86, LP 85]. Model Checking is well established as a verification method in large part because it is fully automated, and because most model checking algorithms produce a counter-example if the correctness property does not hold of the system. Inspired by the success of Model Checking, there is now increased attention to developing such algorithms and procedures for *infinite* state systems.

*This work is supported in part by NSF Grant CCR-9415496 and SRC Grant 97-DP-388. The authors may be reached at `{emerson,kedar}@cs.utexas.edu` and at `http://www.cs.utexas.edu/users/{emerson,kedar}`

This effort is motivated by two tasks : the verification of systems parameterized by the number of processes, such as distributed protocols, and the verification of a fixed set of processes communicating over unbounded channels. Such systems are commonplace and generate an infinite state space, in the first case from the infinite number of instances, each with a fixed number of processes, and in the second case from the unboundedness of the communication channels. State explosion, the limiting factor to the practical application of model checking algorithms, often arises in such parameterized systems for large instances. Thus, a solution for the general problem also helps ameliorate state explosion.

Algorithms are known for model checking many types of infinite state systems : Petri Nets [Es 94], asynchronous parameterized protocols [GS 92], timed automata [AD 91], hybrid systems [H 95], parameterized token rings [EN 95], and parameterized synchronous protocols [EN 96]. For other types of systems, semi-algorithmic procedures have been proposed [PD 95, ID 96, BG 96, BGWW 97].

In an interesting recent paper [ACJT 96] (cf. [Fi 90]), it is pointed out that programs in these formalisms induce "well-quasi-ordered" (*wqo*) transition systems. This condition, together with additional properties, is shown to make the model checking of *safety* properties decidable. The question of deciding general liveness properties is, however, left open. The algorithm for safety properties in [ACJT 96] computes $\mathsf{EF}\mathit{bad}$ as a fixpoint, i.e., in a "backward" direction, starting with the set of states where *bad* holds. More recently, [KMMPS 97] explores the use of automata as representations for infinite sets of states in general fixpoint computations.

Many of the known algorithms, however, are based on a "forward" search, starting with the set of initial states [GS 92, Es 94, BG 96, EN 96]. Algorithms based on quotients w.r.t. a bisimulation equivalence [AD 91, H 95] are also forward searches as the incremental computation of the quotient structure proceeds in a forward direction.

In this paper, we propose a new type of "covering graph" construction as a general method of forward search for Model Checking. The covering graph construction for Petri Nets is developed by Karp and Miller [KM 69], where it is used to decide covering and boundedness questions. Finkel [Fi 90] generalized this construction to arbitrary deterministic *wqo* systems to decide the same properties.

We show that neither the well-quasi-ordering, nor the restriction to deterministic systems is necessary for checking *safety* properties. The essential feature of the covering graph construction is the use of a simulation preorder on the infinite state space. The simulation relation is used to detect potentially infinite paths, and "compress" them by replacing the path with the least upper bound (w.r.t. the simulation preorder) of the set of states occuring on the path. We prove that each node of the covering graph has an associated set of reachable states, which makes it possible to model check safety properties. For liveness properties, well-quasi-ordering of the simulation relation has an important consequence : we show that there is a *finite* witness for the satisfaction of $\mathsf{E}h$ formulas, where h is a linear-time temporal property. The finite witness can be searched for in the covering graph, given an algorithm for determining if a strongly connected component is "good"; i.e., has the "tail" of the witness path embedded in it. Both results apply to general *non-deterministic* systems, thus providing a framework under which to explore the decidability of model checking for non-deterministic infinite state systems.

Although termination is not guaranteed in general (cf. [Fi 90]), many of the forward search algorithms [AD 91, GS 92, EN 96] can be developed in a simple, uniform fashion from the new construction. Furthermore, the construction exposes the key ideas common to these algorithms and other procedures [PD 95, BG 96]. Despite the wide variety among these formalisms, these model checking procedures are based on common high-level ideas. This is a strong indication that the covering graph construction is appropriate for analysis of infinite state systems, even for computation models that are Turing-powerful for which termination cannot be guaranteed. We also consider the new application domain of parameterized broadcast protocols. The new approach is illustrated on the verification of an invalidation-based (MESI) cache coherency protocol, which is shown, fully automatically, to satisfy correctness properties for an *arbitrary* number of processes.

The rest of the paper is structured as follows. Section 2 contains preliminaries; Section 3 introduces the covering graph construction and the model checking procedure for safety properties. Section 4 deals with liveness properties. In Section 5, we describe how many known algorithms may be derived uniformly from the construction, and introduce the application domain of parameterized broadcast protocols. Section 6 concludes the paper with a discussion of related work and future directions.

2 Preliminaries

Infinite state systems are represented as Labeled Transition Systems and linear temporal properties by automata on infinite strings. This section contains the definitions of these concepts and their basic properties. Quantified expressions are written in the format $(\mathbf{Q}x : r : p)$, where $\mathbf{Q}$ is the quantifier, x the bound variable, r the range, and p the expression being quantified. The powerset of a set S is denoted by $\mathcal{P}(S)$.

2.1 Quasi orders and Partial orders

A binary relation $\preceq$ on set S is said to be a *quasi-order* (or *preorder*) if it is reflexive and transitive. The pair $(S, \preceq)$ is called a *preset* (for preordered set). If $\preceq$ is symmetric it is an equivalence relation, and if it is antisymmetric it is a *partial order* and $(S, \preceq)$ is called a *poset*.

In a poset $(S, \preceq)$, an element c is an *upper bound* of a subset X iff $(\forall x : x \in X : x \preceq c)$. The *least upper bound* (*lub*) of X, if it exists, is the upper bound of X that is the minimum w.r.t. $\preceq$ among the set of upper bounds of X. A subset X is *directed* iff any pair of elements in X has an upper bound in X. A a function $C : \mathbf{N} \rightarrow S$ is a chain iff for every $i \in \mathbf{N}$, $C(i) \preceq C(i+1)$. The set of elements of a chain C, $\hat{C}$, equals $\{C(i) | i \in \mathbf{N}\}$. The poset is a *complete partial order* (*cpo*) iff every directed subset has a least upper bound.

Definition 1 (Well-Partial-Order, *wpo***)** *For a poset $(S, \preceq)$, $\preceq$ is a well-partial-order iff for every infinite sequence σ of elements of S, there exist positions $i, j \in \mathbf{N}$ such that $i < j$ and $\sigma_i \preceq \sigma_j$.*

A preorder $(S, \preceq)$ is a *well-quasi-order* iff the induced partial order is a *wpo*.

Proposition 1 *(cf. [Fr 86]) For a preset $(S, \preceq)$, $\preceq$ is a wqo iff every infinite sequence of elements of S contains an infinite sub-sequence that is a chain.*

2.2 Ordered transition systems

A *labeled transition system* [Kel 76] (LTS) A is a structure (S, Σ, R, I, AP, L) where S is a non-empty set of *states*, Σ is a *finite*, nonempty set of *labels*, $R \subseteq S \times L \times S$ is the *transition relation*, $I \subseteq S$ is a nonempty set of *initial states*, AP is a *finite* set of atomic propositions, and $L : S \to \mathcal{P}(AP)$ is the *labeling function*, which assigns to each state the set of atomic propositions true at that state. We write $s \xrightarrow{a} t$ instead of $(s, a, t) \in R$, $s \to t$ for $(\exists a : a \in \Sigma : s \xrightarrow{a} t)$, $s \xrightarrow{*} t$ if t is reachable from s, and $s \xrightarrow{+} t$ if t is reachable in at least one step from s. For sequences of symbols from Σ, the function $\overline{} : \mathcal{P}(S) \to \mathcal{P}(S)$ is defined recursively by $\overline{\epsilon}(X) = X$, $\overline{a}(X) = \{t | (\exists s : s \in X : s \xrightarrow{a} t)\}$, and $\overline{\alpha; a}(X) = \overline{a}(\overline{\alpha}(X))$. Informally, $\overline{\alpha}(X)$ is the set of states reachable from states in X by performing the actions of α in order.

Definition 2 (Simulation) *A relation $\lhd$ on S is a simulation on A iff for any states s, t such that $s \lhd t$, $L(s) = L(t)$, and for every a, u such that $s \xrightarrow{a} u$, there is v such that $t \xrightarrow{a} v$ and $u \lhd v$.*

Definition 3 (Ordered LTS) *The pair $(A, \preceq)$ is called an ordered LTS iff $\preceq$ is a* simulation *on A and a wqo on S.*

Several important ways of specifying systems give rise to ordered LTS's:

1. Finite LTS's with the identity preorder, as any infinite path contains a repeated state.
2. Vector Addition Systems (VAS) [KM 69], as the component-wise ordering of vectors over $\mathbf{N}^k$ (the state space of a VAS), given by $u \preceq v$ iff $(\forall i : i \in [0, k) : u_i \leq v_i)$, is a *wpo*.
3. Petri Nets and Vector addition systems with states (VASS) [Rei 85] are equivalent to VAS's.
4. Real-Time Automata [AD 91], as the bisimulation equivalence on clock values has finite index.
5. Finite state machines communicating over restricted FIFO channels [FR 88] or lossy channels [AJ 93], as the orderings on channel words are *wqo*'s.
6. Parameterized protocols [GS 92, EN 96], where the state space is encoded either as a VASS [GS 92] or by constraints [EN 96].

Definition 4 (Fair LTS) *A fair LTS $\mathcal{A}$ is a structure (A, F), where $A = (S, \Sigma, R, I, AP, L)$ is an LTS and $F \subseteq S$ is a non-empty set of fair states.*

The set of computations of $\mathcal{A}$ is restricted to those paths through A where some state in F occurs infinitely often.

2.3 Temporal Logic and Büchi automata

A Büchi automaton $\mathcal{B}$ is given by a structure $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*, q_0 is the *initial state*, and F is a nonempty subset of *accepting states*. Büchi automata recognize infinite strings over Σ. A *run* of $\mathcal{B}$ over an infinite string w is a function $r : \mathbf{N} \to Q$ such that $r(0) = q_0$, and $(r(i), w_i, r(i+1)) \in \delta$ for every $i \in \mathbf{N}$. Let $\mathit{inf}(r) = \{q | q \in Q \wedge |r^{-1}(q)| = \omega\}$ denote the states from Q that appear infinitely often in r. A run r is *accepting* iff $\mathit{inf}(r) \cap F \neq \emptyset$; i.e., a state from F appears infinitely often along r. A word w is accepted by the automaton iff there is an accepting run over w. The *language* of $\mathcal{B}$, $\mathcal{L}(\mathcal{B})$, is the set of words accepted by it.

Büchi automata are used to specify temporal correctness properties. Propositional linear temporal formulae can be translated into equivalent Büchi automata (cf. [Th 90]). We adopt the automata-theoretic approach to model checking [VW 86], in which the negation of the correctness property is expressed by a Büchi automaton $\mathcal{B}$, and every computation of an LTS A satisfies the correctness property iff the set of computations of the "product" $\mathcal{C} = \mathcal{B} \times A$ is empty. Given a simulation preorder $\preceq$ on A, define $\preceq'$ on C by $(b, s) \preceq' (c, t)$ iff $b = c$ and $s \preceq t$.

Proposition 2 *If $(A, \preceq)$ is an ordered LTS, then $(\mathcal{C}, \preceq')$ is an ordered fair LTS.*

Proposition 3 *Accepting runs of $\mathcal{B}$ over A correspond to computations of $\mathcal{C}$.*

3 Model Checking Safety Properties

The negation of a linear-time safety property can be expressed as an automaton on *finite* strings, instead of a Büchi automaton. The system satisfies the safety property iff no finite path of the system is accepted by the automaton, which holds iff in the combined system there is no path to an accepting state.

The Karp-Miller construction for Petri Nets [KM 69] is a sophisticated version of the naive reachability procedure. The key idea is to "compress" paths that are potentially infinite, using a simulation preorder. We develop below a new generalization of the Karp-Miller construction geared towards Model Checking.

We work with an LTS $A = (S, \Sigma, R, I, AP, L)$, where Σ and AP are both *finite*. As Σ and AP are finite, one can augment the edge labeling to $\Sigma \times AP$, where $s \overset{(a,l)}{\to} t$ in the new labeling iff $s \overset{a}{\to} t$ and $L(t) = l$. This relabeling induces the following *labeling property* : For any subset X of states, and a new action label (a, l), states in $\overline{(a,l)}(X)$ have identical state labels. For the rest of the paper, the system is assumed to have the labeling property.

Let a *uniform* subset of S be a set where any two members have the same state label. The set of uniform subsets of S is denoted by $\mathcal{U}(S)$. The label of a uniform subset X, denoted by $\Lambda(X)$, is the common label of its members if the set is non-empty and a special value $\perp$ if the set is empty. Let $\sqsubseteq$ be a relation on $\mathcal{U}(S)$ with the following properties :

1. $\sqsubseteq$ is a *pre-order*, such that if $X \sqsubseteq Y$ then $\Lambda(X) = \Lambda(Y)$. Let $\approx$ denote the equivalence generated by $\sqsubseteq$. I.e., $X \approx Y$ iff $X \sqsubseteq Y$ and $Y \sqsubseteq X$.

2. For any action symbol a, $\overline{a}$ is monotonic w.r.t. $\sqsubseteq$. I.e., for any X, Y, $X \sqsubseteq Y$ implies $\overline{a}(X) \sqsubseteq \overline{a}(Y)$.

3. $(\mathcal{U}(S)/\approx, \sqsubseteq)$ is a *complete partial order* (*cpo*).

4. For any chain C over $\mathcal{U}(S)$, *lub* $\hat{C} \approx \bigcup \hat{C}$.

5. For any s and any uniform subset X, if $s \in X$ then $\{s\} \sqsubseteq X$. If $(A, \preceq)$ is an ordered LTS and $\{s\} \sqsubseteq X$, there exists $t \in X$ such that $s \preceq t$.

Lemma 1 *$X \approx Y$ implies $\overline{a}(X) \approx \overline{a}(Y)$.*

Lemma 2 *Let C and D be chains such that for every i in $\mathbf{N}$, $C(i) \approx D(i)$. Then lub $\hat{C} \approx$ lub $\hat{D}$.*

The proofs of these Lemmas are quite straightforward and are deferred to the appendix.

3.1 The Covering Graph Procedure

The procedure constructs a covering graph incrementally. Each node n of the graph is labeled by a non-empty uniform subset, which is denoted by $\mathsf{L}(n)$. The subsets are usually infinite so, in practice, finite representations and methods of manipulating such representations are needed. We describe some such representations in Section 5; the properties of the algorithm are independent of the representation method.

The function *rep* maps a uniform subset to its representative, so that for any subset X, $rep(X) \approx X$. The graph is constructed by the following nondeterministic procedure. *New* is the set of unexamined (node, edge label) pairs.

Begin

- Choose a *finite* partition of the set of initial states into uniform subsets. For each set X in the partition, an initial node n of the covering graph is created with $\mathsf{L}(n) = rep(X)$. For each $a \in \Sigma$, add (n, a) to *New*.

- Repeat the following process as long as *New* is non-empty :

  Choose and remove a pair (n, a) from *New*. Let $Y = \overline{a}(\mathsf{L}(n))$. If $Y \not\approx \emptyset$, perform these actions in order:

(Cover) If there is a node m such that $Y \sqsubseteq \mathsf{L}(m)$: make m the a-successor of n.

(Limit) If there is a predecessor k of n on a path γ, with $\mathsf{L}(k) = Z$, such that $\mathsf{L}(n) \approx \overline{\gamma}(Z)$ and $Z \sqsubseteq Y$:
Let $\beta = \gamma; a$. Define C by $C(i) = \overline{\beta^i}(Z)$ for $i \in \mathbf{N}$. Create a node m labeled with $rep(W)$ as the a-successor of n, where $W =$ *lub* $\hat{C}$. For each $a \in \Sigma$, add (m, a) to *New*.

(Step) Create a node m labeled with $rep(Y)$ as the a-successor of n. For each $a \in \Sigma$, add (m, a) to *New*.

End

The Covering Graph Construction Procedure.

In the second alternative, note that $\overline{\beta^0}(Z) = Z \sqsubseteq Y \approx \overline{\beta^1}(Z)$. From this initial condition and the monotonicity of $\overline{\beta}$ (property (2) of $\sqsubseteq$), it follows that C is indeed a chain, and by property (3), *lub*'s of chains exist. The construction procedure is nondeterministic, so several possible covering graphs may be generated. The theorems below hold for every such graph.

Theorem 1 *For every node n, there is a non-empty reachable set of states R such that $\mathsf{L}(n) \approx R$.*

Proof. We prove that this property is an invariant of the procedure. It holds of the initial nodes by their definition and the property of *rep*.

Assume that the property holds at the beginning of an iteration. Let (n, a) be the choice from *New*.

(1) The first alternative is taken. As the set of nodes of the covering graph is not changed, the invariant holds.

(2) The second alternative is taken and a new node is added with label $rep(W)$. From the invariant, there is a non-empty, reachable subset of states R such that $Z \approx R$. So for the new node m,

$$
\begin{aligned}
& \mathit{rep}(W) \\
\approx & \text{ (by definition of } C \text{ and property of } \mathit{rep} \text{)} \\
& \mathit{lub}\ \{\overline{\beta^i}(Z) | i \in \mathbf{N}\} \\
\approx & \text{ (from Lemma 1 } \overline{\beta^i}(Z) \approx \overline{\beta^i}(R)\text{; Lemma 2)} \\
& \mathit{lub}\ \{\overline{\beta^i}(R) | i \in \mathbf{N}\} \\
\approx & \text{ (property (4) of } \sqsubseteq \text{)} \\
& \bigcup \{\overline{\beta^i}(R) | i \in \mathbf{N}\}
\end{aligned}
$$

$\bigcup\{\overline{\beta^i}(R)|i \in \mathbf{N}\}$ is a set of reachable states by definition, and is non-empty as it has R as a subset.

(3) The third alternative is taken and a new node is added with label $\mathit{rep}(Y)$. From the invariant, there is a non-empty reachable subset of states R such that $\mathsf{L}(n) \approx R$. So for the new node m,

$$
\begin{aligned}
& \mathit{rep}(Y) \\
\approx & \text{ (by property of } \mathit{rep} \text{)} \\
& Y \\
\approx & \text{ (by definition of } Y \text{)} \\
& \overline{a}(\mathsf{L}(n)) \\
\approx & \text{ (from Lemma 1)} \\
& \overline{a}(R)
\end{aligned}
$$

$\overline{a}(R)$ is a reachable set of states as R is reachable. As $Y \not\approx \emptyset$, $\overline{a}(R)$ is non-empty. $\square$

Definition 5 (Covering simulation) *Let $\lhd$ be the relation defined between the LTS A and the covering graph by $s \lhd n \equiv \{s\} \sqsubseteq \mathsf{L}(n)$.*

Theorem 2 *Every Covering Graph simulates the underlying LTS by $\lhd$.*

Proof.

Suppose $s \lhd n$ and $\mathsf{L}(n) = X$. By property (1) of $\sqsubseteq$, $L(s) = \Lambda(X)$. Let t be any state such that $s \xrightarrow{a} t$. Then $t \in \overline{a}(\{s\})$, hence $\overline{a}(\{s\})$ is non-empty. By property (5) of $\sqsubseteq$, $\{t\} \sqsubseteq \overline{a}(\{s\})$. By property (2) of $\sqsubseteq$, $\overline{a}(\{s\}) \sqsubseteq \overline{a}(X)$. As $\{t\} \sqsubseteq \overline{a}(X)$, $\overline{a}(X) \not\approx \emptyset$, so n has a successor m on action a.

An invariant of the procedure is that for any edge (k, a, l) in the graph, $\overline{a}(\mathsf{L}(k)) \sqsubseteq \mathsf{L}(l)$. It follows that $\overline{a}(\{s\}) \sqsubseteq \mathsf{L}(m)$, so $\{t\} \sqsubseteq \mathsf{L}(m)$ and $t \lhd m$. This proves that $\lhd$ is a simulation relation. $\square$

Several choices for $\sqsubseteq$ have the necessary properties:

- $\sqsubseteq$ is the subset relation.
- Let $\preceq$ be a simulation relation on the LTS that is a pre-order. Then, $X \sqsubseteq Y$, defined as $(\forall s : s \in X : (\exists y : y \in Y : x \preceq y))$, is a pre-order that satisfies the conditions.
- The same preorder restricted to *directed* subsets, is appropriate for deterministic LTS's. For non-deterministic LTS's, a stronger form of directedness is needed, which is discussed in the full paper [EN 98].

Theorem 3 (Model Checking Safety Properties) *There is a reachable accepting state of the combined system iff there is a covering graph node labeled as accepting.*

Proof.

For any reachable accepting state s, by Theorem 2, there is a node n in the covering graph such that $\{s\} \sqsubseteq \mathsf{L}(n)$. As $\Lambda(\{s\}) = \Lambda(\mathsf{L}(n))$ by property (1) of $\sqsubseteq$, n is an accepting node.

Conversely, by Theorem 1, for every reachable node n of the Covering Graph, there is a non-empty reachable subset R such that $\mathsf{L}(n) \approx R$. By property (1) of $\sqsubseteq$, $\Lambda(\mathsf{L}(n)) = \Lambda(R)$, so that every state in R is accepting. $\square$

4 Model Checking Liveness Properties

Checking if a finite-state fair LTS has a computation is straightforward : the NLOGSPACE algorithm searches for a "looping" path of the form $s_0 \xrightarrow{\alpha} s \xrightarrow{\beta} s$, where s is a fair state (cf. [SVW 87]). The following theorem shows that the concept of a finite "looping" path is easily extended to the concept of a finite "self-covering" path for an ordered fair infinite-state system. Put differently, ordered fair LTS's have a *finite* witness for non-emptiness of the set of computations, even though the LTS may have an infinite number of states.

Definition 6 (Self-covering fair path) *A self covering fair path in an ordered fair LTS $(A, F, \preceq)$ is a finite path of the form $s \xrightarrow{*} t \xrightarrow{+} u$, where s is an initial state, $t \preceq u$, and $t \in F$.*

For the rest of this section, let $(A, \preceq)$ be an ordered LTS, and $\mathcal{B}$ a Büchi automaton. Let $\mathcal{C}$ represent the product of $\mathcal{B}$ and A.

Theorem 4 *There is an accepting run of $\mathcal{B}$ on A iff there is a self-covering fair path in $\mathcal{C}$.*

Proof.

($\Rightarrow$) By Proposition 2, $\mathcal{C}$ is an ordered fair LTS, with the induced preorder $\preceq'$ defined by $(b, s) \preceq' (c, t)$ iff $b = c$ and $s \preceq t$.

Let σ be an accepting run of $\mathcal{B}$ over A. From Proposition 3, σ is a computation of $\mathcal{C}$. As $\mathcal{B}$ is finite-state, some accepting state b from $\mathcal{B}$ appears infinitely often along σ. Let δ be the infinite subsequence of σ obtained by retaining those states with automaton component b.

As $\mathcal{C}$ is ordered, by Proposition 1, δ has an infinite subsequence γ that is a chain w.r.t. $\preceq'$. Thus, there exist distinct states t, u on δ such that $t \preceq' u$ and t is a fair state. Since t is reachable from the initial state s of σ, $s \xrightarrow{*} t \xrightarrow{+} u$ forms a self-covering fair path in $\mathcal{C}$.

($\Leftarrow$) Let $s \xrightarrow{x} t \xrightarrow{y} u$ be a self covering fair path in $\mathcal{C}$. As $t \preceq' u$ and $\preceq'$ is a simulation, for some v, $u \xrightarrow{y} v$ with $u \preceq' v$. Continuing in this manner, define an infinite path labeled by y^ω from $w_0 = t$, where for every i, $w_i \xrightarrow{y} w_{i+1}$ and $w_i \preceq' w_{i+1}$. As w_0 is a fair state, and $w_0 \preceq' w_i$ for every i, it follows from the definition of $\mathcal{C}$ that each w_i is a fair state. Hence the sequence of transitions $x; y^\omega$ induces an infinite path from s that is infinitely often fair. By Proposition 3 this computation is an accepting run of $\mathcal{B}$ on A. $\square$

Theorem 4, when combined with the automata theoretic approach to Model Checking [VW 86], transforms the model checking problem for an ordered LTS to determining if a self-covering fair path exists in the ordered fair LTS formed by the product of the LTS with the Büchi automaton for the negation of the correctness property.

Definition 7 (Positive sequence) *A finite sequence of transitions σ of A is called* positive *for s iff $(\exists t : s \xrightarrow{\sigma} t : s \preceq t)$. An ordered LTS has the* positive path *property iff whenever σ is positive for s, for any u such that $s \preceq u$, there exist v and j such that $j > 0$, $u \xrightarrow{*} v$, σ^j is positive for v, and v is fair if u is fair.*

Note that every VASS has the positive path property. Any sequence σ that is positive for a state s has non-negative vector sum. Hence, for every u such that $s \preceq u$, σ is positive for u; i.e., $j = 1$ and $u = v$ in the definition above.

Definition 8 (Good SCC) *A strongly connected component (SCC) of the covering graph is* good *iff it contains a fair node n such that there is a finite path in the component from n which is positive for a state s such that $s \lhd n$.*

Theorem 5 *For any finite covering graph of C, any self-covering fair path in C induces a good SCC in the covering graph.*

Proof.
Let $s \xrightarrow{*} t \xrightarrow{\sigma} u$ be a self covering path in C. As $t \preceq u$, σ is a positive path for t.

Consider an infinite path ρ labeled with σ^ω starting at t (such a path exists; cf. the proof of Theorem 4). Let m be a node in the covering graph such that $t \lhd m$ (m exists from Theorem 2). By Theorem 2, σ^ω induces an infinite path η through the covering graph from m. Consider the set of nodes of the covering graph occuring on η after each prefix σ^i for $i \in \mathbf{N}$. Since the covering graph is finite, there is a repeated node n in this set. Let $m \xrightarrow{\sigma^l} n \xrightarrow{\sigma^k} n$ be the prefix of η up to the second occurence of n. Let u be the state on ρ after the prefix σ^l. From these definitions $u \lhd n$, hence n is a fair node of the covering graph. From the construction of ρ, σ^k is positive for u. The cycle induced by σ^k in the covering graph from n defines a good SCC of the covering graph. $\square$

Theorem 6 *If C has the positive path property, then a good SCC in the covering graph induces a self-covering fair path in C.*

Proof. Let n be the state in a good SCC from which there is a finite path σ that is positive for a state s such that $s \lhd n$. From Theorem 1, there is a non-empty reachable set of states R such that $\mathsf{L}(n) \approx R$. By the definition of $\lhd$ and the transitivity of $\sqsubseteq$, $\{s\} \sqsubseteq R$.

By Property (5) of $\sqsubseteq$, there is a state t in R such that $s \preceq t$. By the positive path property, there exist u, j such that $t \xrightarrow{*} u$ and σ^j $(j > 0)$ is positive for u, which implies that there is a state v such that $u \xrightarrow{\sigma^j} v$ with $u \preceq v$. Since t is fair, so is u. Since t is reachable from some initial state w, $w \xrightarrow{*} t \xrightarrow{*} u \xrightarrow{+} v$ forms a self-covering fair path in C. $\square$

Specific choices for the simulation relation, and the representation of subsets for the construction are discussed in the following section. To check if a property specified by a Büchi automaton $\mathcal{B}$ for its negation holds for an ordered LTS A, one must

1. Define the product $\mathcal{C} = \mathcal{B} \times A$.

2. Pick an appropriate relation $\sqsubseteq$, and construct a finite covering graph.

3. As the covering graph simulates $\mathcal{C}$, one may determine if the property holds by checking it on the covering graph. If this fails, an algorithmic test to determine if an SCC of the covering graph is good is required, provided that $\mathcal{C}$ has the positive path property.

Theorem 7 (Model Checking Liveness Properties) *Let A be an ordered LTS and $\mathcal{B}$ be a Büchi automaton for the negation of the correctness* *property such that $\mathcal{B} \times A$ has a finite covering graph. If $\mathcal{B} \times A$ has the positive path property, A is correct iff the covering graph does not contain a good SCC.*

5 Applications

5.1 Parameterized Systems

Many distributed protocols are specified as a system parameterized by the number of instances of identical processes. The processes are usually finite-state so that each instance is finite, but there is an infinite number of instances whose disjoint union forms an infinite-state system. Model checking a parameterized system is undecidable in general [AK 86].

A commonly studied type is a control-user system, where each instance contains a single copy of the control process and a specified number of user process copies. A state of an instance is represented by a vector, with the first component being the control state and the other components indicating the number of user processes in each user state. Vectors are ordered by the usual component-wise partial ordering. For the parameterized systems studied in [GS 92] and [EN 96], this ordering is a simulation relation. In [GS 92] the parameterized system is modeled by a VASS and the model checking algorithm is based on Rackoff's [Ra 78] near-optimal algorithm for detecting self-covering paths. As the covering graph construction is effective for VASS's, it provides an alternative algorithm, although of higher worst-case complexity. In [EN 96] a synchronous composition operation is defined, which makes it impossible to model the system as a VASS. The analysis is performed with a finite "abstract graph" which is, in fact, a specialization of the covering graph construction presented in this paper.

In both cases, it is possible to recognize good SCC's algorithmically. For the reduction to a VASS, Rackoff's procedure [Ra 78] may be used. The algorithm for detecting good SCC's in [EN 96] uses a threading construction, which resolves a cycle in the covering graph into "threads" that indicate how processes move from one local state to another. Analysis of this threaded cycle can determine whether the cycle represents a positive path for a state covered by the initial node.

It is usually the case that correctness properties for parameterized systems are of the forms : "every process i satisfies $f(i)$" or "every distinct pair of processes (i,j) satisfies $f(i,j)$". Such properties may be reduced to checking properties of the control process of a modified system using symmetry results from [ES 93, CFJ 93].

5.1.1 Broadcast Protocols

The broadcast model is appropriate for analyzing bus-based hardware protocols such as those for cache coherency. For simplicity, we consider protocols where the state change in response to a broadcast is deterministic.

The system is defined as a control-user system with an interleaving composition rule. As in [GS 92, EN 96], the global state is represented by a vector. Local transitions of a process and synchronizations between pairs of processes can be represented as vector additions [GS 92], while broadcast moves are represented as matrix transforms. For instance, consider a broadcast specified by

- The broadcast send $(a!)$: $s \stackrel{a!}{\rightarrow} t$, and
- The corresponding deterministic broadcast receptions $(a?)$: $s \stackrel{a?}{\rightarrow} u$, $t \stackrel{a?}{\rightarrow} u$, and $u \stackrel{a?}{\rightarrow} s$.

This synchronized broadcast action may be represented by a set of simultaneous equations defining the number of processes in each local state after the broadcast. For a global state G, let $G.s$ represent the number of processes in local state s in G. For a transition from G to H with the broadcast action specified above, the equations are : $H.s = G.u; H.t = 1; H.u = (G.s - 1) + G.t$. Informally, the single process that broadcasts moves from s to t; the processes receiving in state s move to state u. Such transformations may in general be represented by $H = T(G)$, where $T(X) = M(X) + C$ for a 0-1 matrix M with unit vectors as columns. M has this special structure as each state occurs on the r.h.s. in exactly one equation. For local transitions and pairwise synchronizations M is the identity matrix, so that T reduces to a vector addition. The guard of a transform is given by the conjunction of terms $x > 0$ for each variable x that is decremented by the transform; e.g., the guard for the transform above is $s > 0$. The usual component-wise ordering on vectors is easily shown to be a simulation relation for such transforms.

Lemma 3 *For any matrix M of the form above, there exist $m, n \in \mathbf{N}$ such that $m < n$ and $M^m = M^n$.*

Proof. For matrices M, N of this type, every column of MN is a column of M. Thus every column of M^i, for any $i > 0$, is a column of M. Since there are only finitely many distinct arrangements of columns of M into matrices of the same size, there must exist m, n such that $m < n$ and $M^m = M^n$. □

With this Lemma, we can devise an effective procedure for computing the *lub*'s of the chains that arise in

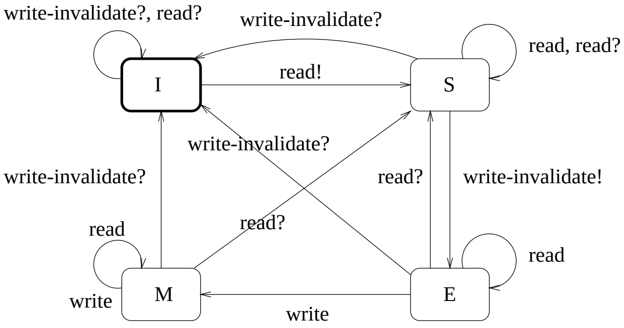

Figure 1: The MESI protocol

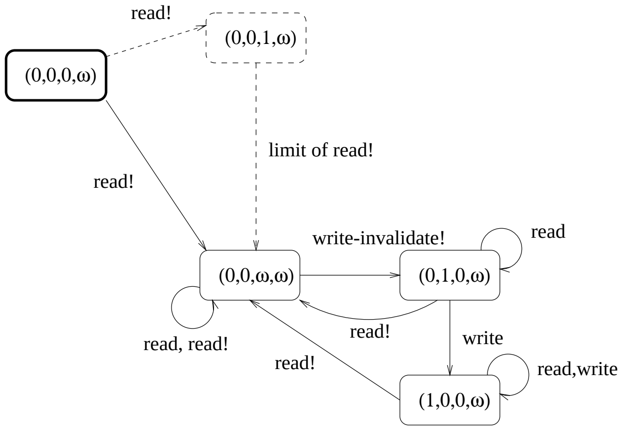

Figure 2: The covering graph. State vector has the form (M, E, S, I).

the covering graph construction. Let $T(X) = M(X) + C$ be a transform and v a vector. For any i, $T^i(X)$ equals (using distributivity of matrix application over vector sum) $M^i(X) + \Sigma_{j\in[0,i)} M^j(C)$. Let m, n be as in the lemma above, and let $\Delta = n - m$. For any k, $M^{m+k*\Delta} = M^m$. Hence, for $i = m + k*\Delta$, $T^i(X)$ equals $M^m(X) + \Sigma_{j\in[0,m)} M^j(C) + k * \Sigma_{j\in[m,n)} M^j(C)$.

Now suppose v is a vector such that $v \leq T(v)$. The set $\{T^i(v) | i \in \mathbf{N}\}$ forms a chain (with $T^0(v) = v$). The set $\{T^i(v) | i \in \mathbf{N} \wedge (i \ mod\ \Delta \equiv m)\}$ is an infinite subchain of this chain, so it has the same *lub* . By the argument above, this set equals $\{u + k * w | k \in \mathbf{N}\}$, where $u = M^m(v) + \Sigma_{j\in[0,m)} M^j(C)$, and $w = \Sigma_{j\in[m,n)} M^j(C)$. For the non-$\omega$ components of v the values in w must be non-negative, as the set is a chain. The representation of the *lub* is given by changing u_i to ω, for every i such that w_i is finite and non-zero. This procedure generalizes the standard limit construction for VASS's (where M is the identity, so $m = 0, n = 1$, which implies that $u = v$ and $w = C$.).

The protocol in Fig. 1 is a variation on the MESI protocol for cache coherency. We have modeled the synchronization mechanisms of a single address (cache line), ignoring the data stored at the address. The covering graph has initial state $(0, 0, 0, \omega)$ representing the set of initial states of the parameterized system, which has an arbitrary number of processes in state I. The covering graph may be used to prove several invariants of the protocol for every instance. For instance the readers-writers exclusion of S (shared) and M (modified) states, which may be written as $\mathsf{AG}(\#M * \#S = 0)$. Similarly mutual exclusion holds between the M and E states, in that $\mathsf{AG}(\#M + \#E \leq 1)$.

Pong and Dubois [PD 95] have analyzed several cache coherency protocols with an abstraction that keeps track of whether there is zero or at least one process in a given local state. The abstraction loses information in the sense that a violation of a safety property in the abstract graph is not necessarily a violation in the concrete system. The covering graph construction, on the other hand, is exact by Theorem 3.

5.2 Real-Time Systems

In the parameterized systems discussed above, the equivalence relation induced by the simulation partial order does not have finite index. For some classes of systems, such as real-time and hybrid systems [AD 91, H 95], there is a bisimulation or simulation equivalence of finite index. In this case, the function *rep* of the covering graph procedure may be chosen so that *rep*(X), for a subset X of equivalent states, is the equivalence class that the states in X belong to. With this definition, since there is a finite number of equivalence classes, the covering graph construction terminates.

In [BCG 89, EN 95], a method for verification of parameterized systems is proposed, which is to set up a family of bisimulations $\{B_n | n \geq m\}$ between instances of size $n \geq m$ and the instance of size m. If this is possible, then the correctness property holds of all instances iff it holds on instances of sizes at most m. Clearly, $B = \bigcup_{n\geq m} B_n$ is a bisimulation over the family of instances. In these papers, B is also an equivalence relation, and by the properties of B_n above, has finite index.

5.3 Communication protocols

[FR 88] consider systems of processes communicating with FIFO channels. They show that if the set of channel contents considered as words over an alphabet are prefixes of $u; v^*$ for some words u, v, a finite covering graph can be constructed for the protocol so that boundedness of channel contents and deadlock-freedom are decidable. Using the results in this paper, general safety properties of the finite state control for these protocols are also decidable. A related approach for dealing with communication protocols is proposed in [BG 96], where sets of reachable states are represented by deterministic automata over finite words.

6 Related Work and Conclusions

Among related work, Finkel [Fi 90] generalizes the construction of Karp and Miller to *wqo* deterministic systems to solve boundedness and covering questions. As noted in the introduction, to ensure the relevant properties of every covering graph neither the restriction to deterministic systems nor the *wqo* property is essential.

Bradfield and Stirling [BS 90, Br 92] use a tableau-based procedure for determining whether a μ-calculus property holds of an infinite state system. They consider the use of the Karp-Miller graph for checking Petri Nets. As noted in [Br 92], a property which is true of the Petri Net may not hold over a Karp-Miller graph for the Petri Net. The method presented here avoids this problem by adopting the automata theoretic approach and constructing the covering graph of the *product* of the system with the property automaton.

[ACJT 96] propose an interesting approach to the decidability of safety properties and the liveness property $\mathsf{AF}p$ for ordered LTS's. This is based on an abstract interpretation of the infinite state space in terms of upward closed subsets. Using their procedure, they can derive uniformly algorithms for deciding safety properties of many types of systems. It is not, however, possible to derive methods for deciding general liveness properties from their procedure.

Solutions to the problem of model checking systems with infinite state spaces are the key to extending the applicability of model checking procedures to parameterized systems, real-time systems, and communication protocols. General approaches to the problem have not been very well-studied, although many decidability results for specific classes are known. This paper provides such a unifying framework by demonstrating that a covering graph construction generates a graph which if finite, allows the checking of safety properties. In addition, we show that the decidability of general liveness properties in *wqo* systems, is linked to an algorithm for deciding the existence of finite self-covering fair paths. The covering graph may be used to search for such paths. We also show that many of the known decidability results for infinite state systems can be cast in these terms. This is a strong indication that the covering graph construction is appropriate for the analysis of infinite-state systems. We also consider a new application domain, that of parameterized broadcast protocols, and indicate how to apply the construction in this domain. This application is demonstrated on an invalidation based cache coherency protocol. These results, we hope, will motivate further applications of this procedure to a wide class of systems.

References

[AJ 93] Abdulla, P., Jonsson, B. Verifying programs with unreliable channels. LICS, 1993. (full version in *Information and Computation*, 127(2):91-101, 1996)

[AD 91] Alur, R., Dill, D. Automata for modeling Real-Time Systems, ICALP 1991.

[AK 86] Apt, K., Kozen, D. Limits for automatic verification of finite-state concurrent systems. *IPL* 15.

[ACJT 96] Abdulla, P., Cerans, K., Jonsson B., Tsay Y-K. General Decidability Theorems for Infinite-State Systems, LICS, 1996.

[Br 92] Bradfield, J. *Verifying Temporal Properties of Systems*, in Progress in Theoretical Computer Science, Birkhäuser, 1992.

[BCG 89] Browne, M. C., Clarke, E. M., Grumberg, O. Reasoning about Networks with Many Identical Finite State Processes, *Information and Computation*, vol. 81, 1989.

[BG 96] Boigelot, B., Godefroid, P., Symbolic verification of communication protocols with infinite state spaces using QDD's, in CAV 1996.

[BGWW 97] Boigelot, B., Godefroid, P., Willems, B., Wolper, P. The power of QDD's, Proceedings of the Fourth International Static Analysis Symposium, Paris, September 1997. LNCS, Springer-Verlag.

[BS 90] Bradfield, J., Stirling, C. Local Model Checking for Infinite State Spaces, *Theoretical Computer Science*, 1990.

[CE 81] Clarke, E.M., Emerson, E.A. Design and Synthesis of Synchronization Skeletons Using Branching Time Temporal

Logic, *Proc. of the Workshop on Logics of Programs*, LNCS#131, Springer–Verlag, 1981.

[CES 86] Clarke, E.M., Emerson, E. A., Sistla, A. P. Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach, *ACM Transactions on Programming Languages and Systems*, **vol.** 8, 1986.

[CFJ 93] Clarke, E.M., Filkorn, T., Jha, S. Exploiting Symmetry in Temporal Logic Model Checking, 5th CAV, LNCS 697.

[ES 93] Emerson, E.A., Sistla, A.P. Symmetry and Model Checking, 5th CAV, LNCS 697.

[EN 95] Emerson, E.A., Namjoshi, K.S. Reasoning about Rings. POPL, 1995.

[EN 96] Emerson, E.A., Namjoshi, K.S. Automatic Verification of Parameterized Synchronous Systems, 8th CAV, LNCS 1102.

[EN 98] Emerson, E.A., Namjoshi, K.S. On Model Checking for Non-deterministic Infinite-State Systems, Manuscript.

[Es 94] Esparza, J. On the decidability of model checking for several μ-calculi and Petri nets, in CAAP 94, LNCS 787.

[Fi 90] Finkel, A. Reduction and Covering of Infinite Reachability Trees, *Information and Computation* vol. 89, 1990.

[Fr 86] Fraisse, R. *Theory of Relations*, Studies in Logic and the Foundations of Mathematics, vol. 118, North-Holland, 1986.

[FR 88] Finkel, A., Rosier, L. A survey on the decidability questions for classes of fifo nets. in Advances in Petri Nets, 1988, LNCS 340.

[GS 92] German, S.M., Sistla, A.P. Reasoning about Systems with Many Processes, *J.ACM*, Vol. 39, No. 3, 1992.

[H 95] Henzinger, T. A. Hybrid automata with finite bisimulations, in ICALP 1995.

[KM 69] Karp, R., Miller, R. Parallel Program Schemata, JCSS, vol. 3., 1969.

[KMMPS 97] Kesten, Y., Maler, O., Marcus, M., Pnueli, A., Shahar, E., Symbolic model checking with rich assertional languages, in CAV, 1997.

[ID 96] Ip, N., Dill, D. Verifying Systems with Replicated Components in Murphi, CAV 1996.

[Kel 76] Keller, R. M. Formal verification of parallel programs. CACM, July 1976.

[LP 85] Lichtenstein, O., A. Pnueli, Checking That Finite State Concurrent Programs Satisfy Their Linear Specification, POPL, 1985.

[PD 95] Pong, F., Dubois, M. A new approach for the verification of cache coherence protocols, *IEEE Trans. Parallel and Distr. Syst.*, vol.6, no.8.

[QS 82] Queille, J.P., Sifakis, J. Specification and Verification of Concurrent Systems in CESAR, *Proc. of the 5th International Symposium on Programming*, LNCS#137, Springer–Verlag, pp. 337–350, April 1982.

[Ra 78] Rackoff, C. The covering and boundedness problem for Petri Nets and Vector Addition Systems. *TCS*, 1978.

[Rei 85] Reisig, W. Petri Nets. An Introduction, volume 4 of Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[SE 89] Streett, R.S., Emerson, E.A. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81, 1989.

[Th 90] Thomas, W. Automata on Infinite Objects, Handbook of Theoretical Computer Science, vol. B.

[Ver 93] Vernier, I. Specification and Verification of Parameterized Parallel Programs, Proc. 8th Intl. Symp. on Comp. and Inf. Sciences, Istanbul, Turkey.

[SVW 87] Sistla, P., Vardi, M., Wolper, P. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. TCS 49: 217-237 (1987)

[VW 86] Vardi, M., Wolper, P. An Automata-Theoretic approach to automatic program verification, LICS, 1986.

A Appendix

A.1 Products with Büchi automata

The product $\mathcal{C}$ for LTS $A = (S, \Sigma, R, I, AP, L)$ and Büchi automaton $\mathcal{B} = (Q, \mathcal{P}(AP), \delta, q_0, F)$ is a fair LTS, defined as $\mathcal{C} = (A', F')$, for $A' = (S', \Sigma', R', I', AP', L')$, where :

1. $S' = Q \times S$,
2. $\Sigma' = \Sigma$,
3. $((b, s), a, (c, t)) \in R'$ iff $(s, a, t) \in R$ and $(b, L(s), c) \in \delta$,
4. $I' = \{q_0\} \times I$,
5. $AP' = AP$,
6. $L'(b, s) = (b, L(s))$, and
7. $F' = \{(b, s) | b \in F, s \in S\}$.

A.2 Properties of the covering graph

Lemma 1 *$X \approx Y$ implies $\hat{a}(X) \approx \hat{a}(Y)$.*

Proof.

$\quad\quad \overline{a}(X) \approx \overline{a}(Y)$
iff (definition of $\approx$)
$\quad\quad \overline{a}(X) \sqsubseteq \overline{a}(Y) \wedge \overline{a}(Y) \sqsubseteq \overline{a}(X)$
$\Leftarrow$(property (2) of $\sqsubseteq$)
$\quad\quad X \sqsubseteq Y \wedge Y \sqsubseteq X$
iff (definition of $\approx$)
$\quad\quad X \approx Y$
$\square$

Lemma 2 *Let C and D be chains such that for every i in $\mathbf{N}$, $C(i) \approx D(i)$. Then lub $\hat{C} \approx$ lub $\hat{D}$.*

Proof.

For any Z,

$\quad\quad$ *lub* $\hat{C} \sqsubseteq Z$
$\Rightarrow$ (definition of *lub* ; transitivity of $\sqsubseteq$)
$\quad\quad (\forall i : i \in \mathbf{N} : C(i) \sqsubseteq Z)$
$\Rightarrow$ ($D(i) \sqsubseteq C(i)$ for every i in $\mathbf{N}$; transitivity of $\sqsubseteq$)
$\quad\quad (\forall i : i \in \mathbf{N} : D(i) \sqsubseteq Z)$
$\Rightarrow$ (definition of *lub*)
$\quad\quad$ *lub* $\hat{D} \sqsubseteq Z$

From this proof, and the reflexivity of $\sqsubseteq$, we can conclude that *lub* $\hat{C} \sqsubseteq$ *lub* $\hat{D}$. A symmetric proof establishes the other direction. Hence, *lub* $\hat{C} \approx$ *lub* $\hat{D}$.
$\square$