Verification by Network Decomposition*

Edmund Clarke[1], Muralidhar Talupur[1], Tayssir Touili[1], and Helmut Veith[2]

[1] Carnegie Mellon University
[2] Technische Universität München

Abstract. We describe a new method to verify networks of homogeneous processes which communicate by token passing. Given an arbitrary network graph and an indexed $LTL \setminus X$ property, we show how to decompose the network graph into multiple constant size networks, thereby reducing one model checking call on a large network to several calls on small networks. We thus obtain cut-offs for arbitrary classes of networks, adding to previous work by Emerson and Namjoshi on the ring topology. Our results on $LTL \setminus X$ are complemented by a negative result which precludes the existence of reductions for $CTL \setminus X$ on general networks.

1 Introduction

Despite the big success of model checking in hardware and software verification, the classical approach to model checking can handle only finite state systems. Consequently, applying model checking techniques to systems involving unlimited concurrency, unlimited memory, or unlimited domain sizes, is a major challenge. Researchers have sought to address these issues by different verification methods including, among others, abstraction, regular model checking, static analysis and theorem proving.

Many software and hardware systems however are described in terms of natural parameters, and for each concrete value of the parameters, the systems have finite state space. A system model involving such parameters is called a parameterized system. Verifying a property of a parameterized system amounts to verifying this property for all values of the parameters. Examples of parameterized systems include, mutual exclusion protocols, cache coherence protocols and multi-threaded systems.

* This research was sponsored by the Semiconductor Research Corporation (SRC) under contract no. 99-TJ-684, the National Science Foundation (NSF) under grants no. CCR-9803774 and CCR-0121547, the Office of Naval Research (ONR) and the Naval Research Laboratory (NRL) under contract no. N00014-01-1-0796, and the Army Research Office (ARO) under contract no. DAAD19-01-1-0485 and by the European Community Research Training Network GAMES. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of SRC, NSF, ONR, NRL, ARO, the U.S. Government or any other entity.

P. Gardner and N. Yoshida (Eds.): CONCUR 2004, LNCS 3170, pp. 276–291, 2004.

In a seminal paper, Emerson and Namjoshi [17] consider systems composed of identical asynchronous processes which are arranged in a ring topology and communicate by passing a boolean token. For several classes of indexed $CTL^* \setminus X$ properties [9] they provide *cutoffs*, i.e., reductions to single systems of constant small size. Consequently, $CTL^* \setminus X$ properties over an *infinite class of networks* can be reduced to a *single model checking call*.

In this paper, we extend the results of Emerson and Namjoshi from rings to *arbitrary classes of networks*. There are two modifications, however: first, our results hold true only for LTL$\setminus X$, and second, we introduce a more refined notion of cut-offs. The first restriction is necessary: We show in Section 4 that with $CTL \setminus X$ it is *impossible* to obtain cut-offs for arbitrary networks.

The second modification actually provides an interesting new view on the notion of cut-offs: in order to verify the parametrized system, we are allowed to model check a *constant number* c of small systems whose network graphs have sizes bounded by a *constant* s. Then, the verification result for the parametrized system is a Boolean combination of the collected results for the small systems. We call such a reduction to a finite case distinction a (c, s)-bounded reduction.

Our main results can be summarized as follows:

- **Verification by Network Decomposition:** Verifying systems with fixed large network graphs G (e.g., concrete instantiations of a parametrized system) can be as challenging as verifying parameterized systems. Note that when $|Q|$ is the state space of the individual processes, then the state space of the whole network can be as high as $|Q|^n$, where n is the number of nodes. We show that the verification of an indexed LTL$\setminus X$ property φ for a system with network graph G can be achieved by an *efficiently computable* (c, s)-bounded reduction. For the important case of 2-indexed properties, it is sufficient to model check at most 36 networks of size 4.
- **Offline Verification:** In a scenario where φ is known in advance and the network G can change for different applications, we can first verify a constant number of small systems *offline*. Later, when we get to know the network graph G, the correctness of G with respect to specification φ can be verified *online* by simply evaluating a constant size Boolean function, regardless of the size of the processes.

  Again, for 2-indexed properties, the offline computation involves at most 36 calls to the model checker for networks of size 4.
- **Cut-Offs:** For every class of networks $\mathbb{T}$ and k-indexed LTL$\setminus X$ property φ one can verify if φ holds on *all* networks in $\mathbb{T}$ by a (c, s)-bounded reduction, where c and s depend only on k.

  Depending on the complexity of the networks in $\mathbb{T}$, finding a suitable (c, s)-bounded reduction will in general still involve manual algorithm design. Similar to famous results about linear time algorithms for bounded tree-width [11], our proofs just guarantee the existence of small reductions.

This paper is organized as follows: this section concludes with related work. In Section 2, we describe the system model in detail. Section 3 contains the main results of the paper. Section 4 describes the impossibility of cut-offs for $CTL \setminus X$. Finally, the conclusion in Section 5 briefly considers further performance enhancements for practical applications of our method.

Related Work. Verification of parameterized systems is well known to be undecidable [2, 25]. Many interesting approaches to this problem have been developed over the years, including the use of symbolic automata-based techniques [22, 6, 26, 7, 1, 4], network invariants [3, 24], predicate abstraction [23], or symmetry [10, 16, 20, 18, 19]. In [5], cut-offs were used for the verification of systems sharing common resources, where the access to the resources is managed according to a FIFO-based policy.

In addition to [17] mentioned above, Emerson et al. have shown a large number of fundamental results involving cut-offs. The paper [13] by Emerson and Kahlon also considers LTL\X cut-offs for arbitrary network topologies with multiple tokens, but each of them is *confined to two processes* which renders their model incomparable to ours. Other previous work by Emerson and Kahlon [12, 15, 14] consider other restricted forms of process interaction. [21] considers the verification of single index properties for systems with multiple synchronous processes.

Indexed temporal logic was introduced in [9]. This paper also considers identical processes arranged in ring topology.

The work that is closest in spirit to our negative results on $CTL^* \setminus X$ logic is the work by Browne, Clarke and Grumberg in [8] which shows how to characterize Kripke structures up to bisimilarity using fragments of CTL*. Our results show that even $CTL^* \setminus X$ with only two atomic propositions is sufficient to describe an infinite class of Kripke structures which are not bisimilar to each other. In other words, bisimilarity over the class of Kripke structures with two labels gives rise to an infinite number of equivalence classes.

2 Computation Model

Network Topologies. A *network graph* is a finite directed graph $G = (S, C)$ without self-loops, where S is the set of *sites*, and C is the set of *connections*. Without loss of generality we assume that the sites are numbers, i.e., $S = \{1, 2, \ldots, |S|\}$. A (network) *topology* $\mathbb{T}$ is a class of network graphs.

Token Passing Process. A single token passing process P (process) is a *labeled transition system* (Q, Σ, δ, I) such that:

- $Q = \widehat{Q} \times B$, where $\widehat{Q}$ is a finite, nonempty set and $B = \{0, 1\}$. Elements of Q will be called *local states*. The boolean component of a local state indicates the possession of the token. We say that a local state (q, b) holds the token if $b = 1$.
- $\Sigma = \Sigma_f \cup \Sigma_d \cup \{\mathsf{rcv}, \mathsf{snd}\}$ is the set of actions. The actions in Σ_d are token dependent actions, those of Σ_f are called token independent actions, and $\{\mathsf{rcv}, \mathsf{snd}\}$ are actions to receive and send the token. The sets Σ_f, Σ_d are mutually exclusive.

- $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, such that every $((q,b),a,(q',b')) \in \delta$ fulfills the following conditions:
  (a) A free transition does not change token possession: $a \in \Sigma_f \Rightarrow b = b'$
  (b) A dependent transition can execute only if the process possesses the token: $a \in \Sigma_d \Rightarrow b = b' = 1$
  (c) A receive establishes possession of token: $a = \mathsf{rcv} \Rightarrow b = 0, b' = 1$
  (d) A send revokes the possession of token: $a = \mathsf{snd} \Rightarrow b = 1, b' = 0$
- $I \subseteq Q$ is the set of initial states.

Topological Composition. Let $G = (S,C)$ be a network graph and $P = (Q,\Sigma,\delta,I)$ be a single token process. Then P^G denotes the concurrent system containing $n = |S|$ instances of P denoted by P_s, $s \in S$. The only synchronization mechanism between the processes is the passage of a token according to the network graph G. Formally, the system P^G is associated with a transition system $(\mathcal{Q},\Delta,\mathcal{I})$ defined as follows:

- $\mathcal{Q} = \{(q_1,\ldots,q_n) \in Q^n \mid \text{exactly one of the } q_i \text{ holds the token}\}$.
- $\Delta \subseteq \mathcal{Q}^{2n}$ is defined as follows: a transition $(q_1,q_2,\ldots,q_n) \to (q'_1,q'_2,\ldots,q'_n)$ is in Δ in one of two cases:
  (a) **Asynchronous Transition:** there exist an index $j \in \{1,\ldots,n\}$ and an action $a \in \Sigma_f \cup \Sigma_d$ such that $(q_j,a,q'_j) \in \delta$, and for all indices $i \neq j$ we have $q_i = q'_i$. In other words, only process P_j makes a transition (different from a send or receive).
  (b) **Token Transition:** there exist a network connection $(j,k) \in C$ in the network graph, such that $(q_j,\mathsf{snd},q'_j) \in \delta$, $(q_k,\mathsf{rcv},q'_k) \in \delta$, and $q_i = q'_i$ for all indices i different from j,k.
- $\mathcal{I} = \{(q_1,\ldots,q_n) \in I^n \mid \text{exactly one of the } q_i \text{ holds the token}\}$.

An *execution path* is considered fair if and only if every process P_i receives and sends the token infinitely often. We assume that every system P^G that we consider has fair paths. An immediate consequence of the fairness condition is that a system P^G can have fair paths only if G is strongly connected.

We shall use indexed temporal logics, which can refer explicitly to the atomic propositions of each process P_i, to specify properties of the compound systems. For each local state q in Q we introduce propositional variables $q(1),\ldots,q(n)$. The atomic proposition $q(i)$ says that process P_i is in state q. Thus, for a global state g we define

$$g \models q(i) \qquad \text{iff in global state } g\text{, process } P_i \text{ is in state } q.$$

Starting from this definition for atomic propositions, we can easily define common temporal logics such as CTL or LTL in a canonical way. Throughout this paper, we will assume that the path quantifiers **A** and **E** quantify over fair paths. Further we assume that LTL formulas are implicitly quantified by **E**. This restriction simplifies our proofs but does not restrict generality.

Example 1. The formula $\mathbf{G}(q(1) \Rightarrow \mathbf{F}q(2))$ says that whenever process P_1 is in state q then process P_2 will be in state q sometime in the future.

For increased expressibility we permit that in an atomic formula $q(x)$ the process index x is a variable (called *index variable*) which can take any value from 1 to $|S|$, the total number of processes. Thus, x can refer to arbitrary processes. We shall write $\varphi(x_1, \ldots, x_n)$ to indicate that the temporal formula φ depends on the index variables $x_1, \ldots x_n$. We can substitute the index variables in a formula $\varphi(x_1, \ldots, x_k)$ by integer values $i_1, \ldots, i_k$ in the natural way, and denote the resulting formula by $\varphi(i_1, \ldots, i_k)$.

In addition to substitution by constants, we can also quantify over the index variables $x_1, \ldots x_n$ using a prefix of existential and universal quantifiers with the natural semantics. Such formulas are called quantified temporal formulas. For example, the formula $\forall x \exists y . \varphi(x, y)$ means "For all processes x there exists a process y, such that the temporal formula $\varphi(x, y)$ holds." A formula without quantifier prefix is called *quantifier-free*. If all index variables in a formula are bound by quantifiers we say that the formula is *closed*, and *open* otherwise. The quantifier-free part of a quantified formula is called the *matrix* of a formula.

Example 2. The formula $\exists x, y . \mathbf{G}(q(x) \Rightarrow \mathbf{F}q(y))$ says that there exist two processes P_x and P_y, such that whenever process P_x is in state q then process P_y will be in state q some time in future.

The formal semantics of this logic is straightforward and is omitted for the sake of brevity.

Definition 1 (k-Indexed Temporal Formula). *Let $\mathcal{L}$ be a temporal logic. A k-indexed temporal formula is a formula whose matrix refers to at most k different processes, i.e., there are at most k different constant indices and index variables.*

3 Reductions for Indexed LTL\X Specifications

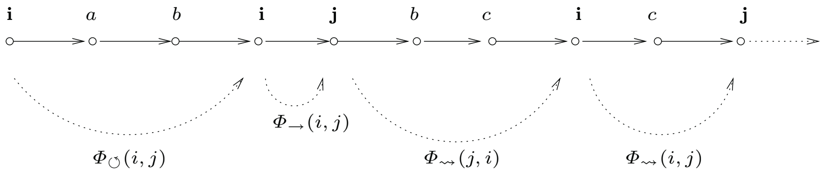
In this section, we will show how to reduce the model checking question $P^G \models \varphi$ to a series of model checking questions on smaller systems P^{G_i}'s where we can bound the size of the network graphs G_i as well as the number of the G_i's. For the sake of simplicity, we will start with the special case of 2-indexed existential LTL\X specifications, which can be readily generalized to the full case.

3.1 Existential 2-Indexed $LTL \setminus X$ Specifications

In this section we show how to verify simple 2-indexed LTL\X properties of the form $\exists i, j . \varphi(i, j)$, where $i \neq j$. We will use the combinatorial insights we obtain from this case to obtain the more general results later on.

Recall that 2-indexed properties are concerned only with properties of two processes in a given system. Our process communication model implies that two processes P_i and P_j can only affect each other by passing or receiving a token. Consequently, the synchronization between P_i and P_j crucially depends on the paths between sites i and j in the network graph. The following example is crucial to understanding the intuition behind our approach:

Example 3. The Figure below shows one path $\pi = i, a, b, i, j, b, c, i, c, j, \ldots$ in a network graph.



Suppose that we are only interested in properties concerning the processes P_i and P_j, but not in processes P_a, P_b, P_c. Then only the sequence of the i's and j's in the path are of interest. Looking at π from left to right, we see four possibilities for what can happen between i and j: (1) P_i sends a token, and receives it back without P_j seeing it (formally, we will write $\Phi_{\circlearrowleft}(i,j)$ to denote this); (2) P_i passes the token *directly* to P_j ($\Phi_{\rightarrow}(i,j)$); (3) P_j sends the token to P_i through several intermediate sites ($\Phi_{\rightsquigarrow}(j,i)$); and (4) P_i sends the token back to P_j through several intermediate sites ($\Phi_{\rightsquigarrow}(i,j)$). There are two more possibilities which do not occur in π: (5) $\Phi_{\rightarrow}(j,i)$ and (6) $\Phi_{\circlearrowleft}(j,i)$. The important insight is the following: **If we know which of these 6 cases can occur in a network graph G, then we have all information needed to reason about the communication between P_i and P_j.**

We will later construct small network graphs with 4 nodes where the sites i and j are represented by two distinguished nodes $site_1$ and $site_2$, while *all other* sites are represented by two "hub" nodes hub_1 and hub_2.

This example motivates the following definitions:

Definition 2 (Free Path). *Let I be a set of indices, and π be a path in a network graph G. We say that π is I-free, if π does not contain a site from I.*

We now define three kinds of path types which will be shown to capture all relevant token paths between two processes P_i and P_j.

Definition 3 (Connectivity, Characteristic Vectors). *Let i, j be indices in a network graph G. We define three connectivity properties of the indices i, j:*

$G \models \Phi_{\circlearrowleft}(i,j)$	*"There is a* $\{j\}$-free *path from i to itself."*
$G \models \Phi_{\rightsquigarrow}(i,j)$	*"There is a path from i to j via a third node not in $\{i,j\}$."*
$G \models \Phi_{\rightarrow}(i,j)$	*"There is a direct edge from i to j."*

Using the connectivity properties, we define an equivalence relation $\sim_2$ on network graphs: Given two network graphs G_1 and G_2 along with two pairs of indices a_1, b_1 and a_2, b_2, we define

$$(G_1, a_1, b_1) \sim_2 (G_2, a_2, b_2)$$

iff for every $\Phi \in \{\Phi_{\circlearrowleft}, \Phi_{\rightsquigarrow}, \Phi_{\rightarrow}\}$,

$$G_1 \models \Phi(a_1, b_1) \iff G_2 \models \Phi(a_2, b_2) \text{ and}$$
$$G_1 \models \Phi(b_1, a_1) \iff G_2 \models \Phi(b_2, a_2)$$

If $(G_1, a_1, b_1) \sim_2 (G_2, a_2, b_2)$ *we say that the indices* a_1, b_1 *in* G_1 *have the same connectivity as the indices* a_2, b_2 *in* G_2.

The characteristic vector $v(G_1, a_1, b_1)$ *is the* 6*-tuple containing the truth values of* $G_1 \models \Phi_{\circlearrowleft}(a_1, b_1)$, $G_1 \models \Phi_{\rightsquigarrow}(a_1, b_1)$, $G_1 \models \Phi_{\rightarrow}(a_1, b_1)$ $G_1 \models \Phi_{\circlearrowleft}(b_1, a_1)$, $G_1 \models \Phi_{\rightarrow}(b_1, a_1)$, *and* $G_1 \models \Phi_{\rightsquigarrow}(b_1, a_1)$.

By definition it holds that $(G_1, a_1, b_1) \sim_2 (G_2, a_2, b_2)$ iff they have the same characteristic vectors, i.e., $v(G_1, a_1, b_1) = v(G_2, a_2, b_2)$. Since the number of characteristic vectors is constant, it follows that $\sim_2$ has finite index. The characteristic vectors can be viewed as representatives of the equivalence classes.

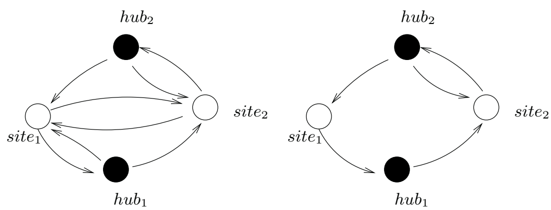


Fig. 1. Network Graphs A, B, realizing two different characteristic vectors

Example 4. Consider the network graphs A, B of Figure 1. It is easy to see that $(A, site_1, site_2)$ has characteristic vector $(1, 1, 1, 1, 1, 1)$, i.e.,

$$A \models \Phi_{\circlearrowleft}(site_1, site_2) \wedge \Phi_{\rightsquigarrow}(site_1, site_2) \wedge \Phi_{\rightarrow}(site_1, site_2) \wedge \Phi_{\circlearrowleft}(site_2, site_1) \wedge \Phi_{\rightsquigarrow}(site_2, site_1) \wedge \Phi_{\rightarrow}(site_2, site_1)$$

and $(B, site_1, site_2)$ has characteristic vector $(0, 1, 0, 1, 1, 0)$, i.e.,

$$B \models \neg\Phi_{\circlearrowleft}(site_1, site_2) \wedge \Phi_{\rightsquigarrow}(site_1, site_2) \wedge \neg\Phi_{\rightarrow}(site_1, site_2) \wedge \Phi_{\circlearrowleft}(site_2, site_1) \wedge \Phi_{\rightsquigarrow}(site_2, site_1) \wedge \neg\Phi_{\rightarrow}(site_2, site_1).$$

Note that a network graph will in general have several characteristic vectors depending on the indices we consider. The set of characteristic vectors of a graph G can be efficiently computed from G in quadratic time. The crucial insight in our proof is that for two processes P_i and P_j, the connectivity between their indices i, j in the network graph determines the satisfaction of quantifier-free LTL\X properties $\varphi(i, j)$ over P^G:

Lemma 1 (2-Index Reduction Lemma). *Let* G_1, G_2 *be network graphs,* P *a process, and* $\varphi(x, y)$ *a* 2*-indexed quantifier-free LTL\X property. Let* a_1, b_1 *be a pair of indices on* G_1*, and* a_2, b_2 *a pair of indices on* G_2*. The following are equivalent:*

(a) $(G_1, a_1, b_1) \sim_2 (G_2, a_2, b_2)$*, i.e.,* a_1, b_1 *and* a_2, b_2 *have the same connectivity.*
(b) $P^{G_1} \models \varphi(a_1, b_1)$ *iff* $P^{G_2} \models \varphi(a_2, b_2)$.

The lemma motivates the following model checking strategy: Given a (possibly complicated) network graph G_1 and two of its sites i, j, we can try to obtain a simpler network $G_2 := G_{(i,j)}$, with two special nodes $site_1$ and $site_2$ that have the same connectivity in G_2 as the indices i and j in G_1, and thus satisfies condition (a) of the lemma. For the case of two indices, we can always find such a network graph $G_{(i,j)}$ with at most 4 sites.

Proposition 1. *For each graph G and indices i, j there exists a 4-node graph $G_{(i,j)}$ called the* connection topology *of i, j, having two special sites $site_1$ and $site_2$ such that*

$$(G, i, j) \sim_2 (G_{(i,j)}, site_1, site_2).$$

In other words, the indices i and j in G have the same connectivity as the indices $site_1$ and $site_2$ in $G_{(i,j)}$.

Since $G_{(i,j)}$ satisfies condition (a) of Lemma 1, we obtain the following important consequence:

Corollary 1. *Let $\varphi(i, j)$ be a 2-indexed quantifier-free LTL\X property. Then*

$$P^G \models \varphi(i, j) \qquad \text{iff} \qquad P^{G_{(i,j)}} \models \varphi(site_1, site_2).$$

Thus, we have achieved a reduction from a potentially large network graph G to a 4-node network graph $G_{(i,j)}$. We will now show how to actually construct the connection topology $G_{(i,j)}$.

Construction of $G_{(i,j)}$. We construct the reduction graphs as follows. $G_{(i,j)}$ has four sites: $site_1, site_2, hub_1$, and hub_2. The sites $site_1$ and $site_2$ are called *primary sites*. They represent the sites of interest i and j. The other sites are called *hubs*, and they represent the other nodes of the graph G. Let us describe in more detail the role of these different nodes. Recall that to satisfy Proposition 1, the sites $site_1$ and $site_2$ in $G_{(i,j)}$ should have the same connectivity as i, j in G. Therefore:

- If $\Phi_{\leadsto}(i, j)$ holds in G (i.e., there exists a path from i to j in G that goes through a third node), then $\Phi_{\leadsto}(site_1, site_2)$ has also to hold in $G_{(i,j)}$, i.e., there should exist in $G_{(i,j)}$ a path from $site_1$ to $site_2$ that goes through a third node. The site hub_1 will play the role of this "third node". Therefore, in this case, $G_{(i,j)}$ contains an edge from $site_1$ to hub_1, and from hub_1 to $site_2$.
- In the same manner, if $\Phi_{\circlearrowleft}(i, j)$ holds in G (i.e., there exists a path from i to itself in G that does not go through j), then $\Phi_{\circlearrowleft}(site_1, site_2)$ should also be true in $G_{(i,j)}$. As previously, this is ensured by considering the following edges: $(site_1, hub_1)$ and $(hub_1, site_1)$.
- Finally, if $\Phi_{\rightarrow}(i, j)$ holds in G (i.e., there exists a direct edge in G from i to j), then $G_{(i,j)}$ should also contain the edge $(site_1, site_2)$.
- The paths from j to i are treated in a symmetrical way.

For example, let H be a graph having as sites i, j, k, and l (among others), such that $v(H, i, j) = (1, 1, 1, 1, 1, 1)$, and $v(H, k, l) = (0, 1, 0, 1, 1, 0)$; then the graphs A and B of Example 4 correspond respectively to the reduction graphs $H_{(i,j)}$ and $H_{(k,l)}$.

Since our fairness assumption implies that the network is strongly connected, not all characteristic vectors actually occur in practice. A closer analysis yields the following bound:

Proposition 2. *For* 2 *indices, there exist at most* 36 *connection topologies.*

Proof. By our fairness assumption, every connection topology must be strongly connected. This implies that the following conditions must hold:

- At least one of $\Phi_{\rightarrow}(i, j)$ or $\Phi_{\rightsquigarrow}(i, j)$ must be true.
- At least one of $\Phi_{\rightarrow}(j, i)$ or $\Phi_{\rightsquigarrow}(j, i)$ must be true.

Consequently a detailed counting shows that the number of different possible characteristic vectors is $3 \times 3 \times 4 = 36$. □

Let us now return to the question of verifying properties of the form $\exists x, y.\varphi(x, y)$. Note that Corollary 1 only provides us with a way to verify one quantifier-free formula $\varphi(i, j)$. Given a system P^G, we define its 2-topology, denoted by $T_2(G)$, as the collection of all different connection topologies appearing in G. Formally,

Definition 4. *Given a network graph $G = (S, C)$ the 2-topology of G is given by*

$$T_2(G) = \{G_{(i,j)} \mid i, j \in S, i \neq j\}.$$

By Proposition 2, we know that $|T_2(G)| \leq 36$. Since we can express $\exists x, y.\varphi(x, y)$ as a disjunction $\bigvee_{i,j \in S} \varphi(i, j)$ we obtain the following result as a consequence of Corollary 1:

Theorem 1. *The following are equivalent:*

(i) $P^G \models \exists x, y.\varphi(x, y)$
(ii) *There exists a connection topology $T \in T_2(G)$, such that $P^T \models \varphi(site_1, site_2)$.*

Thus, we obtain the following reduction algorithm for model checking $P^G \models \exists x, y.\varphi(x, y)$:

1: Determine $T_2(G)$.
2: For each $T \in T_2(G)$, model check $P^T \models \varphi(site_1, site_2)$.
3: If one of the model checking calls is successful then output "true" else output "false".

3.2 Existential k-Indexed LTL\X Specifications

We will now show how to generalize the results of the previous section to k-indexed properties. Throughout this section, we will write expressions such as $\bar{i}$ to denote k-tuples of indices, and $\bar{x}$ to denote k-tuples of variables. We will first adapt the notion of connectivity as follows. Let $\bar{i} = i_1, i_2 \ldots i_k$ be a sequence of indices, and $I = \{i_1, i_2 \ldots i_k\}$. Then we define the following connectivity properties:

$G \models \Phi_{\circlearrowleft}(x, I)$ "There is an $(I \setminus \{x\})$-*free* path from x to itself."
$G \models \Phi_{\rightsquigarrow}(x, y, I)$ "There is a path from x to y via a third node not in I."
$G \models \Phi_{\rightarrow}(x, y)$ "There is a direct edge from x to y."

By instantiating the variables x and y by the indices $i_1, \ldots, i_k$ in all possible ways, we obtain a finite number of different conditions which will describe all possible connectivities between the indices $i_1, \ldots, i_k$.

As in the previous section, we can define an equivalence relation $\sim_k$, where $(G_1, \bar{i}) \sim_k (G_2, \bar{j})$ iff the indices $\bar{i}$ have the same connectivity in G_1 as the indices $\bar{j}$ in G_2. Since the number of conditions is bounded, $\sim_k$ is an equivalence relation of finite index, and we can describe each equivalence class by a characteristic vector $v(G, \bar{v})$. Like in the previous section, we define the k-connection topologies, $G_{(i_1, i_2 \ldots i_k)}$ of the processes $P_{i_1}, P_{i_2} \ldots P_{i_k}$ in G as the smallest graphs that preserve all the connectivity properties between the processes $P_{i_1}, P_{i_2} \ldots P_{i_k}$. The construction of the topology graphs is illustrated in Figure 2.

The unfilled nodes $site_1, \ldots, site_k$ in the graph are the primary sites. There is a *hub* site associated with each primary site. Moreover, there is an edge from each hub hub_j back to its primary $site_j$ if there is an $(I \setminus \{i_j\})$-*free* path from i_j to itself. There is an edge from hub_j to $site_l$ if there is a path from i_j to i_l in G via a third node not in I, and there is an edge from $site_j$ to $site_l$ if there exists a direct edge (i_j, i_l) in G.

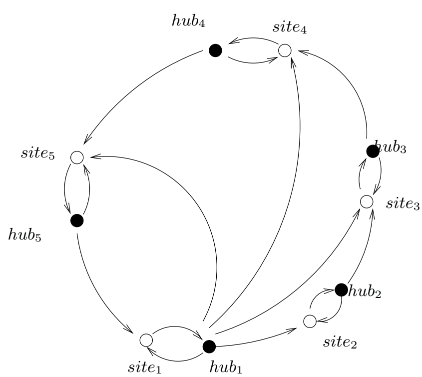


Fig. 2. An example of a 5-index connection topology

Analogous to the bounds on 2-connection topologies it can be shown that each k-connection topology has at most $2k$ processes and that there are at most $3^{k(k-1)}2^k$ distinct k-connection topologies. By an argument analogous to that of the previous section, we obtain the following corollary

Corollary 2. *Let $\varphi(\bar{x})$ be a k-indexed quantifier-free LTL\X property. Then*

$$P^G \models \varphi(\bar{i}) \qquad \text{iff} \qquad P^{G_{(\bar{i})}} \models \varphi(site_1, site_2, \ldots, site_k).$$

The notion of k-topology is also defined completely analogously:

Definition 5. *Given a network graph $G = (S, C)$ the k-topology of G is given by*

$$T_k(G) = \{G_{(\bar{i})} \mid \bar{i} \in S^k, \textit{all indices in } \bar{i} \textit{ are distinct}\}.$$

Consequently, we obtain a model checking procedure from the following theorem, similar to the case of 2-indices:

Theorem 2. *The following are equivalent:*

(i) $P^G \models \exists \bar{x}.\varphi(\bar{x})$
(ii) *There exists a connection topology $T \in T_k(G)$, such that $P^T \models \varphi(site_1, site_2, \ldots, site_k)$.*

As mentioned before $|T_k(G)| \leq 3^{k(k-1)}2^k$.

3.3 Specifications with General Quantifier Prefixes

In this section we will show how to obtain reductions for k-indexed specifications with first order prefixes.

Let us for simplicity consider the 2-indexed formula $\Phi := \forall x \exists y.\varphi(x, y)$. Over a network graph $G = (S, C)$, $|S| = n$ it is clear that Φ is equivalent to $\wedge_{1\leq i\leq n} \vee_{1\leq j\leq n} \varphi(i, j)$. A naive application of Corollary 2 would therefore require n^2 calls to the model checker which may be expensive for practical values of n. In practice, however, we can bound the number of model checker calls by $|T_2(G)|$ since this is the maximum number of *different* connection topologies. We conclude that the n^2 model checker calls must contain repetitions. In the program, we can make sure that at most 36 calls to the model checker are needed. We obtain the following algorithm:

1: Determine $T_2(G)$.
2: For each $T \in T_2(G)$
3: model check $P^T \models \varphi(site_1, site_2)$
4: $g[T] := 1$ iff model checking successful, and 0 otherwise
5: Output $\bigwedge_{1\leq i\leq n} \bigvee_{1\leq j\leq n} g[G_{(i,j)}]$.

By simplifying the formula in line 5, we may further increase performance. The algorithm can be adapted for k indices in the obvious way. To state the main theorem of this section, we define (c, s)-bounded reductions, where c bounds the number of **c**alls to the model checker, and s bounds the **s**ize of the network graph.

Definition 6 ((c, s)-Bounded Reduction). *Let G, P be as above, and φ a closed k-indexed formula with matrix $\varphi'(x_1, \ldots, x_k)$. Let Ψ denote a property of interest (e.g., the model checking property "$P^G \models \varphi$"). A (c, s)-bounded reduction of property Ψ is given by:*

- *a sequence of c* reduced *network graphs $G_i = (S_i, C_i), 1 \leq i \leq c$ such that $|S_i| \leq s$. called reduction graphs.*
- *a boolean function B over c variables $g_1, \ldots, g_c$, such that*

$$\Psi \quad \textit{iff} \quad B(g_1, \ldots, g_c) = 1 \quad \text{where} \quad g_i := 1 \textit{ iff } G_i^P \models \varphi'(site_1, \ldots, site_k)$$

In other words, property Ψ is decided by c calls to the model checker, where in each call the network graph is bounded by s.

Further, we say that a class $\mathcal{L}$ of specifications has (c, s) bounded reduction if for all network graphs G and any $\varphi \in \mathcal{L}$, the property $P^G \models \varphi$ has (c, s)-bounded reduction. We can now state our main result:

Theorem 3. *Let φ be any k-indexed LTL\X specification. Then the model checking problem "$P^G \models \varphi$" has polynomial-time*[1] *computable $(3^{k(k-1)}2^k, 2k)$-bounded reductions.*

In fact, the sequence of reduced network graphs is just the different k-connection topologies occurring in G. This implies that given k and network graph G, all k-indexed LTL\X specifications have the same reduction. Stated another way, LTL\X has $(3^{k(k-1)}2^k, 2k)$-bounded reduction.

3.4 Cut-Offs for Network Topologies

In this section, we prove the existence of cutoffs for network topologies, i.e., (infinite) classes of network graphs. We say that a class of network graphs has cutoff (c, s), if the question whether **all** the network graphs in this topology satisfy the specification has a (c, s)-bounded reduction.

Definition 7 (Cut-Off). *Let $\mathbb{T}$ be a network topology, and $\mathcal{L}$ a class of specifications. $\mathbb{T}$ has a cut-off (c, s) for $\mathcal{L}$ if for all specifications $\varphi \in \mathcal{L}$ the property*

$$\Psi := \quad \text{“} \forall G \in \mathbb{T} \;.\; P^G \models \varphi \text{”}$$

has a (c, s)-bounded reduction.

It is not hard to prove that a (c, s)-bounded reduction for a network graph translates to a cut-off for a network topology:

Theorem 4. *For k-indexed specifications, all network topologies $\mathbb{T}$ have $(2k, 3^{k(k-1)}2^k)$-bounded reductions.*

Note that the theorem does not provide us with an *effective* means to find the reduction; it does however guarantee that at least in principle we can always find a cutoff by investigating the topology $\mathbb{T}$.

[1] In the size of the network graph G.

4 Bounded Reductions for $CTL \setminus X$ Are Impossible

In this section, we show that indexed CTL\ X formulas over two indices don't have (c, s)-bounded reductions. We will first show the following generic result about CTL\ X:

Theorem 5. *For each number i there exists an CTL\ X formula φ_i with the following properties:*

- *φ_i is satisfiable (and has a finite model).*
- *φ_i uses only two atomic propositions l and r.*
- *Every Kripke structure K where φ_i is true has at least i states.*
- *φ_i has the form $\mathbf{EF}\varphi_i'$.*

The result is true even when the Kripke structure is required to have a strongly connected transition relation.

Proof. Our goal is to describe a formula φ_i using atomic propositions l and r whose models must have at least i states. We will construct a large conjunction $\bigwedge_{\psi \in \Gamma} \psi$, and describe which formulas to put in Γ. The idea is simple: Γ needs to contain i CTL\X formulas which describe the existence of i different states. Then the formula $\mathbf{EF} \bigwedge_{\psi \in \Gamma} \psi$ will be the sought for φ_i.

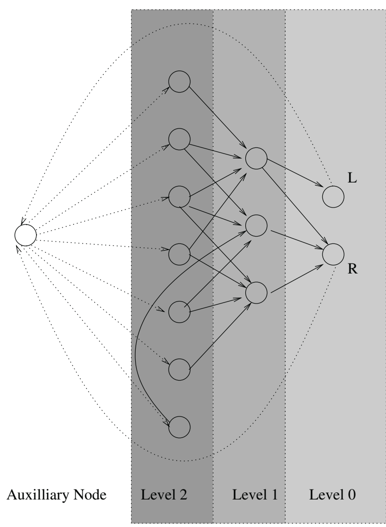


Fig. 3. The Kripke structure K, constructed for three levels. The dashed lines indicate the connections necessary to achieve a strongly connected graph

Consider a Kripke structure K as in Figure 3:

- In Level 0, it contains two distinct states L, R labelled with l and r respectively. To express the presence of these states, we include the formulas, let $\psi_0^1 := (l \wedge \neg r)$ and $\psi_0^2 := (r \wedge \neg l)$, and include $\mathbf{EF}\psi_0^1$ and $\mathbf{EF}\psi_0^2$ into Γ. It is clear that $\mathbf{EF}\psi_0^1$ and $\mathbf{EF}\psi_0^2$ express the presence of two mutually exclusive states.
- In Level 1, K contains $2^2 - 1 = 3$ states, such that the first one has $\{L, R\}$-free paths to L and R, the second one an $\{L, R\}$-free path only to L, and the third one an $\{L, R\}$-free path only to R. The characteristic properties of level 1 states are expressed by formulas
  $\psi_1^1 := \mathbf{EF}^- \psi_0^1 \wedge \mathbf{EF}^- \psi_0^2$
  $\psi_1^2 := \mathbf{EF}^- \psi_0^1 \wedge \neg \mathbf{EF}^- \psi_0^2$
  $\psi_1^3 := \neg \mathbf{EF}^- \psi_0^1 \wedge \mathbf{EF}^- \psi_0^2$
  where $\mathbf{EF}^- x$ denotes $\mathbf{E}(\neg l \wedge \neg r)\mathbf{U}x$, i.e., a variant of $\mathbf{EF}$ which forbids paths through L and R. To enforce the existence of the Level 1 states in the Kripke structure, we include $\mathbf{EF}\psi_1^1, \mathbf{EF}\psi_1^2, \mathbf{EF}\psi_1^3$ into Γ.
- In general, each Level k has at least $2^{k+1} - 1$ states which differ in their relationship to the states in Level $k - 1$. The presence of such states is expressed by formulas $\mathbf{EF}\psi_k^x$.

All these formulas are included into Γ until the requested number i of different states is reached. By construction, all properties required in the theorem statement are trivially fulfilled. In particular, Figure 3 demonstrates that there always exists a strongly connected model. □

Remark 1. This result is closely related to early results about characterizing Kripke structures up to bisimulation in [8]. The results in [8] give rise to the following proof idea for Theorem 5: Let $K_1, \ldots, K_n$ be all Kripke structures with 2 labels of size $\leq i$, and let $f_1, \ldots, f_n$ be CTL\ X formulas which characterize them up to stuttering bisimulation. Consider now the formula $\varphi_i := \bigwedge_{1 \leq j \leq n} \neg f_j$. By construction every model of φ_i must have $> i$ states. At this point, however, the proof breaks down, because we do not know from the construction if φ_i is satisfiable at all. The natural way to show that φ_i has a model would be to prove that stuttering bisimulation over a 2-symbol alphabet has infinite index. This property however is a corollary to Theorem 5, and we are not aware of a proof in the literature.

For *properties involving only the presence of the token*, a system P^G, where $G = (S, C)$ essentially behaves like a Kripke structure with set of states S and transition relation C. The proof of this assertion is not given here.

Now we can show by contradiction that indexed CTL\ X cannot have bounded reductions. Suppose CTL\X did have (c, s)-bounded reduction for some s. Then, by Theorem 5, we can always find a CTL\X formula Φ such that the network graph underlying any system that satisfies Φ must have size at least $c + 1$. Thus CTL\X does not have bounded reductions. Consequently, we also have the following corollary:

Corollary 3. *There exists a network topology* $\mathbb{T}$ *for which* 2*-indexed CTL\ X does not have cut-offs.*

5 Conclusion and Future Work

In this paper, we have described a systematic approach for reducing the verification of large and parameterized systems to the verification of a sequence of much smaller systems. The current paper is primarily concerned with the algorithmic and logical concepts underlying our approach. We will conclude this paper with further considerations concerning the practical complexity of model checking.

For simplicity, let us again consider the case of 2-indexed properties. Suppose the processes P in our network have state space $|Q|$. Then our reduction requires to model check up to 36 network graphs with 4 sites, resulting in a state space of $|Q|^4$. Even this model checking problem may be expensive in practice. By a close analysis of our proofs, it is however possible to reduce the state space even further to $O(|Q|^2)$.

It is easy to show that Lemma 1 will hold even when the *processes at the hubs* are simple *dummy processes* containing two states whose mere task is to send and receive the token infinitely often. Consequently, the systems $P^{G_{(i,j)}}$ will have state space of size $2^2 \times |Q|^2$.

The results in this paper on LTL\X were derived assuming fairness condition on the systems. We can obtain similar reductions by removing this assumption. Doing away with fairness necessitates the consideration of two more path types other than the ones described in Section 3.1. Consequently, the topology graphs have more than 4 sites and also the number of different topology graphs increases. Reductions in non-fair case will be described in a future work.

References

1. P. A. Abdulla, B. Jonsson, M. Nilsson, and J. d'Orso. Regular model-checking made simple and efficient. In *In Proceedings 13th International Conference on Concurrency Theory (CONCUR)*, volume 2421 of *Lecture Notes in Computer Science*, pages 116–130. Springer-Verlag, 2002.
2. K. Apt and D. Kozen. Limits for automatic verification of finite state concurrent systems. *Information Processing Letters*, 15:307–309, 1986.
3. T. Arons, A. Pnueli, S. Ruah, and L. Zuck. Parameterized verification with automatically computed inductive assertions. In *Proc. 13th Intl. Conf. Computer Aided Verification (CAV)*, 2001.
4. B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large. In *15th Intern. Conf. on Computer Aided Verification (CAV'03)*. LNCS, Springer-Verlag, 2003.
5. A. Bouajjani, P. Habermehl, and T. Vojnar. Verification of Parametric Concurrent Systems with Prioritized FIFO Resource Management. In *Proceedings of CONCUR'03*, 2003.

6. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *12th Intern. Conf. on Computer Aided Verification (CAV'00)*. LNCS, Springer-Verlag, 2000.
7. A. Bouajjani and T. Touili. Extrapolating tree transformations. In *14th Intern. Conf. on Computer Aided Verification (CAV'02)*. LNCS, Springer-Verlag, 2002.
8. M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.
9. M. C. Browne, E. M. Clarke, and O. Grumberg. Reasoning about networks with many identical finite state processes. *Information and Computation*, 81:13–31, 1989.
10. E. M. Clarke, T. Filkorn, and S. Jha. Exploiting symmetry in temporal model checking. In *Proc. 5th Intl. Conf. Computer Aided Verification (CAV)*, 1993.
11. B. Courcelle. Graph rewriting: An algebraic and logic approach. B:459–492, 1990.
12. A. E. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *17th International Conference on Automated Deduction*, pages 236–254, 2000.
13. A. E. Emerson and V. Kahlon. Model checking larage-scale and parameterized resource allocation systems. In *Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, pages 251–265, 2002.
14. A. E. Emerson and V. Kahlon. Model checking guarded protocols. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 361–370, 2003.
15. A. E. Emerson and V. Kahlon. Rapid parameterized model checking of snoopy cache protocols. In *Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, pages 144–159, 2003.
16. E. A. Emerson, J. Havlicek, and R. Trefler. Virtual symmetry. In *LICS*, 2000.
17. E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *ACM Symposium on Principles of Programming Languages (POPL'95)*, 1995.
18. E. A. Emerson and A. Sistla. Utlizing symmetry when model-checking under fairness assumptions: An automata theoretic approach. *TOPLAS*, 4, 1997.
19. E. A. Emerson and A. P. Sistla. Symmetry and model checking. In *Proc. 5th Intl. Conf. Computer Aided Verification (CAV)*, 1993.
20. E. A. Emerson and R. Trefler. From asymmetry to full symmetry. In *CHARME*, 1999.
21. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of ACM*, 39, 1992.
22. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In O. Grumberg, editor, *Proc. CAV'97*, volume 1254 of *LNCS*, pages 424–435. Springer, June 1997.
23. S. K. Lahiri and R. E. Bryant. Indexed predicate discovery for unbounded system verification". In *Proc. CAV'04*. To appear.
24. A. Pnueli, S. Ruah, and L. Zuck. Automatic deductive verification with invisible invariants. In *Lecture Notes in Computer Science*, 2001.
25. I. Suzuki. Proving properties of a ring of finite state machines. *Information Processing Letters*, 28:213–214, 1988.
26. T. Touili. Widening Techniques for Regular Model Checking. In *1st vepas workshop*. Volume 50 of Electronic Notes in Theoretical Computer Science, 2001.