A New Approach to Upward-Closed Set Backward Reachability Analysis*

Jesse Bingham

Department of Computer Science, University of British Columbia, Canada
jbingham@cs.ubc.ca

Abstract. In this paper we present a new framework for computing the backward reachability from an upward-closed set in a class of parameterized (i.e. infinite state) systems that includes broadcast protocols and petri nets. In contrast to the standard approach, which performs a single least fixpoint computation, we consecutively compute the finite state least fixpoint for constituents of increasing size, which allows us to employ binary decision diagram (BDD)-based symbolic model checking. In support of this framework, we prove necessary and sufficient conditions for convergence and intersection with the initial states, and provide an algorithm that uses BDDs as the underlying data structure. We give experimental results that demonstrate the existence of a petri net for which our algorithm is two orders of magnitude faster than the standard approach, and speculate properties that might suggest which approach to apply.

1 Introduction

The successes of finite state model checking techniques have motivated much research on automatic verification of infinite state systems. Recent works have investigated a class of infinite systems called *well-structured transition systems* and provided several positive decidability results [1, 18, 20]. Examples of well-structured transition systems include: basic process algebra, lossy channel systems, timed automata, petri nets, and various species of parameterized finite state systems. An important class of decidable problems, with variants variously called *safety property verification*, *control state reachability*, and *coverability*, ask if a specified set of target (i.e. violating) states is reachable from any of the designated *initial* states.

The standard approach to solving these problems is based on *backward reachability analysis*, in which, starting from the set of violating states, preimages are iteratively computed until a fixpoint is reached. For well-structured systems, convergence is guaranteed, and an abstract algorithm is given in several papers on the topic [1, 18, 20]; we will call this the *standard algorithm*. Necessary for practical implementation of the standard algorithm is an efficient representation of so-called *upward-closed sets*. Delzanno et al. propose using *covering sharing trees* (CST) for this purpose [12]. One drawback of this technique is that checking for convergence is co-NP hard in the size of the involved CSTs.

* UBC Computer Science Tech Report TR-2004-07, version 2.0

In this paper we propose an alternative to the standard algorithm. We focus on a generalization of *broadcast protocols* [18, 17]. A broadcast protocol represents the composition of an unbounded number of identical finite state processes that communicate in certain ways. The infinite state space arises because a broadcast protocol consists of an infinite family of systems; for each positive n the system of *size* n involves the composition of n processes. The paramount difference between our approach and the standard is that rather than compute the transitive preimage of the entire set of violating states in one fell swoop, we iteratively compute the backward reachability set for constituent systems of increasing size, until we reach a certain convergence condition. Since the constituent systems are each finite state, we can leverage well-known finite state symbolic model checking [6] and binary decision diagram (BDD) [5] techniques. A primary advantage of our approach is that the necessary convergence checks can be done efficiently. A possible disadvantage is that in some sense we undo a symmetry reduction inherent in the standard approach.

That our procedure eventually covers all elements of the transitive preimage follows trivially from the theory of well-structured transition systems. However, *determining when* we have reached full coverage is unobvious[1]. A key result in this paper is a theorem that gives us a necessary and sufficient condition for detecting this convergence. Through our experimental results we exhibit the existence of a family of petri nets for which our algorithm is two orders of magnitude faster than a state-of-the-art implementation of the standard approach. We emphasize that our approach does not always outperform the standard approach, but there are examples for which each is superior. In our discussion of Sect. 6 we speculate the system properties that might suggest which technique to apply.

The paper is organized as follows. Preliminary definitions are given in Sect. 2. Section 3 develops our approach and highlights the differences between the standard approach. In Sect. 4 we show that our technique can be applied to petri nets, and in Sect. 5 experimental results for an example petri net are presented. Our discussion of Sect. 6 attempts to explain the strengths of each approach and outlines future work. Finally, related work is outlined in Sect. 7.

2 Preliminaries

A *transition system* T is a pair $(S, \rightarrow)$, where S is the *state space* and $\rightarrow \subseteq S \times S$ is the *transition relation*. Associated with T is the *predecessor function* $Pred : 2^S \rightarrow 2^S$ defined by $Pred(X) = \{y \mid \exists x \in X : y \rightarrow x\}$. A *path of* T is a sequence $x_0, \ldots, x_\ell$ over S such that for each $1 \leq i \leq \ell$ we have $x_{i-1} \rightarrow x_i$. The function $Pred^* : 2^S \rightarrow 2^S$ is defined by $Pred^*(X) = \{y \mid$ there exists a path from y to some $x \in X\}$.

Let $\mathbb{N}$ and $\mathbb{Z}$ denote the natural numbers and the integers, respectively. For a vector $v \in \mathbb{Z}^m$ we let $v[i]$ denote the ith component of v for each $1 \leq i \leq m$. The *weight* of $v \in \mathbb{Z}^m$ is $|v| = \sum_{i=1}^{m} v[i]$, and for any $X \subseteq \mathbb{N}^m$, denote $\{x \in X \mid |x| = n\}$ by $[X]_n$. We define a reflexive and transitive relation $\preceq$ on $\mathbb{N}^m$ by $x \preceq y$ iff for all $1 \leq i \leq m$ we have

[1] In particular, convergence between sizes n and $n+1$ is in general insufficient for full coverage, as shown by our Theorem 2.

$x[i] \leq y[i]$, and we write $x \prec y$ iff $x \preceq y$ and $x \neq y$. For $1 \leq j \leq m$, let inc^j be the vector of weight 1 such that $\text{inc}^j[i]$ is 1 if $i = j$ and 0 otherwise.

A *broadcast protocol* B is a finite set of pairs $\{(M_1, c_1), \ldots, (M_{|B|}, c_{|B|})\}$ where each M_i is a $m \times m$ binary matrix with all columns being unit vectors, and each c_i is an integer vector of height m such that $|c_i| = 0$. The semantics of B is the transition system $(\mathbb{N}^m, \rightarrow)$, where $\rightarrow \subseteq \mathbb{N}^m \times \mathbb{N}^m$ is such that $u \rightarrow v$ iff $v = Mu + c$ for some $(M, c) \in B$.

It follows that whenever $u \rightarrow v$ in a broadcast protocol we have $|u| = |v|$. Intuitively, a broadcast protocol models a parameterized system consisting of an unbounded number of identical finite state processes, where each process has m states. A vector $v \in \mathbb{N}^m$ represents any state in which there are $v[i]$ processes in local state i for each $1 \leq i \leq m$. Communication between processes can occur as a broadcast, in which all components change state based on the type of broadcast and the local state, along with a rendezvous-like synchronization in which a bounded number of processes collaborate to change state. These two aspects of a transition are respectively modelled by the matrix M and the vector c of each pair (M, c) in the broadcast protocol.

In our verification problem the initial states and target (i.e. violating) states are respectively required to be sets of the following forms. A *parametric set* is $I \subseteq \mathbb{N}^m$ such that $I = \{x \mid x[1] \sim_1 a[1] \wedge \cdots \wedge x[m] \sim_m a[m]\}$, for some $a \in \mathbb{N}^m$ and each $\sim_i$ is either $=$ or $\geq$. The vector a is called the *root vector* of I. An *upward-closed set* [20, 1] is a set $U \subseteq \mathbb{N}^m$ such that $x \in U$ and $x \preceq y$ implies $y \in U$. For $X \subseteq \mathbb{N}^m$, the *upward-closure* of X is $\uparrow X = \{y \mid \exists x \in X : x \preceq y\}$. If U is upward-closed, a *basis* for U is a set U^b such that $U = \uparrow U^b$.

Lemma 1 *For broadcast protocols, if U is upward-closed then $Pred^*(U)$ is upward-closed.*

Proof: Proved in [18] for a slightly less general notion of broadcast protocol. However, the proof is easily extended to handle our notion. □

A set $X \subset \mathbb{N}^m$ is *canonical* if for all distinct $x, y \in X$ we have that x and y are incomparable under $\preceq$. It is well-known that any upward-closed set has a canonical finite basis, and that this basis is unique. Given an upward-closed set U, we let $gen(U)$ denote this basis. The *base-weight* of an upward-closed set U is $\max(\{|u| \mid u \in gen(U)\})$, denoted $bw(U)$.

The verification problem we tackle is now defined.

Definition 1 (Broadcast Protocol Reachability Problem). *The* Broadcast Protocol Reachability Problem *(BPRP) asks, given a broadcast protocol B, a parametric set I, and an upward-closed set U, does there exist $v \in I$ and $u \in U$ such that there is a path of B from v to u?*

BPRP is decidable; a decision procedure based on the standard algorithm is given by Esparza, Finkel, and Mayr [18].[2] Our notion of broadcast protocols subsumes petri nets in that any algorithm to solve BPRP can be harnessed to solve a similar problem on petri

[2] In fact, our definition of broadcast protocols and our admissible initial state sets both modestly generalize those of [18], however it is clear that their algorithm could be extended to handle our version of BPRP.

```
previous_reach := ∅
reach := gen(U)
while ¬(↑reach ⊆ ↑previous_reach) do
     if (I ∩ ↑reach ≠ ∅) then
          exit with verification failure
     previous_reach := reach
     reach := reach ∪ Pred(reach)
exit with verification success
```

Fig. 1. The standard algorithm

nets called the *coverability problem* (it is well-known that broadcast protocols subsume petri nets in this sense).

3 Our Approach

A rendition of the standard algorithm is given in Fig. 1. On the surface, this algorithm resembles the well-known finite state backward reachability analysis, i.e. *least fixpoint computation*, the difference being that the involved sets are upward-closed and hence infinite. The algorithm is guaranteed to converge for well-structured transition systems such as broadcast protocols. For details on the standard approach we refer the reader to [1, 18, 20].

Our approach contrasts with the standard approach and is based on the following observation. A broadcast protocol consists of the disjoint union of a countably infinite number of finite transition systems (this follows from the fact that $u \to v$ implies $|u| = |v|$). For each i, one such subsystem is obtained by restricting $\to$ to $[\mathbb{N}^m]_i$, and intuitively corresponds to the instance of with i processes. Starting with $i = 1$, we analyze each of these subsystems by computing $Pred^*([U]_i)$ for successive values of i and checking if this set allows us to determine a nonempty intersection with the initial states I. If so, there exists a path from I to U and the procedure terminates. Otherwise i is incremented and the process repeats until we reach a certain convergence condition.

The skeleton of our algorithm is given in Fig. 2. Omitted are the definitions of the termination condition *converged*, and the function *intersect_check*. Note that implicit in the line "`compute` $\Gamma_i := Pred^*([U]_i)$" is an inner loop that performs a least fixpoint computation. However, unlike the fixpoint computation of the standard approach (i.e. the `while` loop of Fig. 1), our fixpoint computations take place in finite domains (namely $2^{[\mathbb{N}^m]_i}$) and are hence amenable to the well-known techniques of finite state symbolic model checking [6].

In Sect. 3.1 we develop a theorem that gives a necessary and sufficient condition for *converged*, while in Sect. 3.2 we present a theorem that conveys the appropriate definition of *intersect_check*. Section 3.3 shows how BDDs can be employed as the underlying data structure in our algorithm. For the remainder of this section we fix an BPRP instance (B, I, U) with corresponding transition system $(\mathbb{N}^m, \to)$.

```
i := 1
while (¬converged) do
     compute Γ_i := Pred*([U]_i)
     if intersect_check(I,Γ_i) then
          exit with verification failure
     i := i+1
exit with verification success
```

Fig. 2. The skeleton of our algorithm

3.1 A Convergence Condition

In this section we develop Theorem 1, which gives us a convergence condition for the algorithm of Fig. 2. Central to the proof is the notion of *eager descent*, explained here.

Define[3] $\mu : Pred^*(U) \to Pred^*(U)$ to be such that $\mu(x)$ is a $\preceq$-minimal element of the set $\{t \mid t \preceq x\} \cap Pred^*(U)$. Suppose $v \in Pred^*(U)$ for some upward-closed set U. Then there exists a path $\tau = t_0, \ldots, t_q$ in $(\mathbb{N}^m, \to)$ such that $t_0 = \mu(v)$ and $t_q \in U$. Note that for each $1 \leq i \leq q$, there exists $(M_i, c_i) \in B$ such that $t_i = M_i t_{i-1} + c_i$. Letting $d = v - \mu(v)$, consider the sequence

$$\sigma = t_0 + d,\ t_1 + M_1 d,\ t_2 + M_2 M_1 d,\ \ldots,\ t_q + \prod_{j=0}^{q-1} M_{q-j} d$$

which is also a path in $(\mathbb{N}^m, \to)$; let σ_i denote the $(i+1)$th entry in σ. This path starts at $v = t_0 + d$ and terminates at a state in U (since U is upward-closed). If we view σ as a *witness* to the assertion $v \in Pred^*(U)$, then we can view τ as a "light-weight witness" of $v \in Pred^*(U)$. σ and τ are similar in that both paths follow the same sequence of broadcast "firings", i.e. for each $1 \leq i \leq q$, there we have both $\sigma_i = M_i \sigma_{i-1} + c_i$. and $t_i = M_i t_{i-1} + c_i$. Further, the sequence $v, t_0, t_1, \ldots, t_q$, which is a path in the digraph $(\mathbb{N}^m, \to \cup \preceq^{-1})$, in some sense expresses the connection between v and its light-weight witness: we start at v, then jump down through $\preceq^{-1}$ to t_0, then follow the transitions through a smaller system to get to U. An eager descent takes this concept a step further by iteratively identifying light-weight witnesses for each state reached in a path to U, hence we alternate between jumping down to a smaller system and following system transitions; these jumps always land as low as possible.

Definition 2 (eager descent). *An* eager descent *from $v \in Pred^*(U)$ to an upward-closed set U is a path $e = s_0, t_0, s_1, t_1, \ldots, s_q, t_q$ through the digraph $(\mathbb{N}^m, \to \cup \preceq^{-1})$ such that*

1. *$v = s_0$, and*
2. *$t_i = \mu(s_i)$ for all $0 \leq i \leq q$, and*
3. *$t_i \to s_{i+1}$ for all $0 \leq i < q$, and*
4. *$s_i \neq s_j$ for all $0 \leq i < j \leq q$, and*
5. *$t_q \in U$*

Figure 3 gives a graphical depiction of an eager descent.

[3] μ is not uniquely defined, but that's okay; any μ satisfying the definition will suffice.

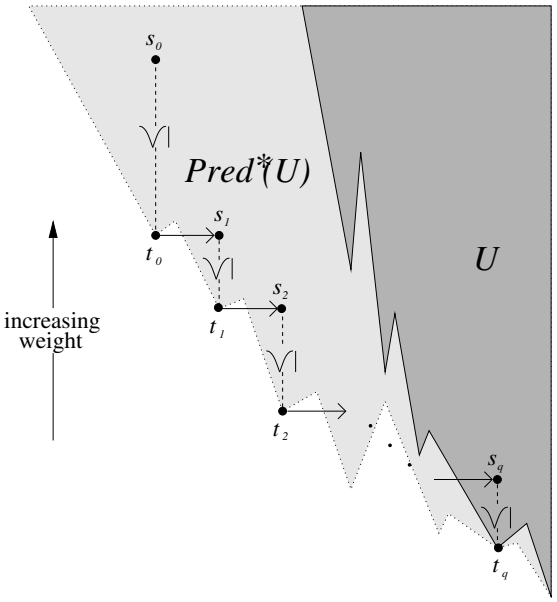

Fig. 3. Eager Descent

Lemma 2 *For all $s \in Pred^*(U)$ there exists an eager descent from s to U.*

Proof: Given $s \in Pred^*(U)$, let $dist(s,U)$ denote the length of a shortest path through $\rightarrow$ from s to a state in U. We define an eager descent from s to U as follows. Let $s_0 = s$ and $t_0 = \mu(s_0)$. If $t_0 \in U$, we are done. Otherwise, let s_1 be such that $t_0 \rightarrow s_1$ and $dist(s_1,U) = dist(t_0,U) - 1$; such a state must always exist. Continuing in this manner, we iteratively define $t_i = \mu(s_i)$ and choose s_{i+1} such that $t_i \rightarrow s_{i+1}$ and $dist(s_{i+1},U) = dist(t_i,U) - 1$. We terminate when we reach some $t_q \in U$[4]. Clearly the sequence generated in this way satisfies conditions 1-3 of definition 2.

To see that condition 4 is satisfied, suppose there exists $0 \leq i < j$ such that $s_i = s_j$. Note that the sequence $|s_0|, |s_1|, \ldots$ is non-increasing. Hence $|s_i| = |s_{i+1}| = \cdots = |s_j|$, and also $t_k = s_k$ for all $k \in \{i, \ldots, j-1\}$. It follows that $dist(s_j,U) < dist(s_i,U)$, which contradicts $s_i = s_j$. Using similar reasoning we can show that the sequence $t_0, t_1, \ldots$ never repeats an element.

It remains to show that this procedure will reach $t_q \in U$ (condition 5 of Def. 2). Consider the set $A = \bigcup_{i=1}^{|s|} Pred^*([U]_i)$. Since $|s|$ is finite and $Pred^*([U]_i)$ is finite for any i, it follows that A is finite. The sequence $t_0, t_1, \ldots$ never repeats an element (as argued above), and all elements are in A. Finally, as long as $t_i \in A \setminus U$ we have that s_{i+1} and t_{i+1} are both well-defined. Hence the procedure will eventually reach $t_q \in U$.

□

[4] Note that we could very well reach i such that $s_i \in U$ and $t_i \notin U$. This is certainly allowable under Def. 2.

Lemma 3 *Let $e = s_0, t_0, \ldots, s_q, t_q$ be an eager descent to upward-closed set U. Then $t_q \in gen(U)$.*

Proof: From Def. 2 we have $t_q \in U$. Now suppose there exists $x \in U$ such that $x \prec t_q$. Then $x \preceq s_q$, which contradicts the requirement of Def. 2 that $t_q = \mu(s_q)$. □

Definition 3 (maximum displacement). *For $v \in \mathbb{Z}^m$ with weight 0, the* displacement *of v, denoted $dis(v)$ is the sum of the positive components of v. The* maximum displacement *$maxdis(B)$ of a broadcast protocol B is defined to be* $\max(\{dis(c) \mid (M,c) \in B\})$.

Lemma 4 *Let $e = s_0, t_0, \ldots, s_q, t_q$ be an eager descent from s_0 to upward-closed set U. Then $|s_i| - |t_i| \leq maxdis(B)$ for each $1 \leq i \leq q$.*

Proof: Let us abbreviate t_{i-1}, s_i, and t_i by r, s, and t respectively, and let $u = s - t$, and hence u is nonnegative and $|s| - |t| = |u|$. We have that $s = Mr + c$ for some $(M,c) \in B$, and we note that $dis(c) \leq maxdis(B)$. We will prove that for each $1 \leq k \leq m$, $u[k] \leq c[k]$ if $c[k] > 0$, or $u[k] = 0$ otherwise. It follows that $|u| \leq dis(c)$.

Suppose there exists k such that $u[k] > c[k]$ and $u[k] > 0$. Then we have that[5]

$$0 \;\leq\; t[k] \;=\; s[k] - u[k] \;=\; (Mr + c)[k] - u[k] \;<\; (Mr)[k] \;=\; \sum_{j=1}^{m} M[k,j]r[j]$$

Hence, since M is a binary matrix and r is nonnegative, this implies that there exists j such that $M[k,j] = 1$ and $r[j] \geq 1$. Define the vector $r' = r - \mathrm{inc}^j$. Then $Mr' + c = M(r - \mathrm{inc}^j) + c = Mr - M\mathrm{inc}^j + c = Mr - \mathrm{inc}^k + c = s - \mathrm{inc}^k$, where the second to last equality follows from the fact that M has unit vector columns. Since $s[k] - u[k] \geq 0$, we have $s[k] > 0$ which implies $s - \mathrm{inc}^k$ is a nonnegative vector. Therefore we have that $r' \prec r$, $r' \rightarrow s - \mathrm{inc}^k$, and $t \preceq s - \mathrm{inc}^k$. Since $t \in Pred^*(U)$ and by Lemma 1 $Pred^*(U)$ is upward-closed, we have $s - \mathrm{inc}^k \in Pred^*(U)$. Hence $r' \in Pred^*(U)$. This contradicts e being an eager descent, since if that were the case, then $r = \mu(s_{i-1})$, however we have $r' \prec r$ which implies $\mu(s_{i-1}) \neq r$. □

Theorem 1 (convergence). *Let U be an upward-closed set and let $n \geq bw(U)$. Then*

$$Pred^*(U) = \uparrow \bigcup_{i=1}^{n} Pred^*([U]_i) \tag{1}$$

if and only if

$$\uparrow Pred^*([U]_{n+maxdis(B)}) \subseteq \uparrow Pred^*([U]_{n+maxdis(B)-1}) \subseteq \cdots \subseteq \uparrow Pred^*([U]_n) \tag{2}$$

[5] Here $M[k,j]$ denotes the entry of M at row k and column j.

Proof: (If) Clearly $Pred^*(U) \supseteq \uparrow \bigcup_{i=1}^{n} Pred^*([U]_i)$, therefore we focus on the other containment. Let s be a state of $Pred^*([U]_\ell)$ for some $\ell > n+\delta$, and let $e = s_0, t_0, \ldots, s_q, t_q$ be an eager descent from s to U, which exists by Lemma 2. We prove by induction on q that $|t_0| \leq n$. The result follows since $|t_0| \leq n$ and $t_0 \preceq s$ imply that $s \in \uparrow \bigcup_{i=1}^{n} Pred^*([U]_i)$. If $q = 0$, then by Lemma 3 $t_0 \in gen(U)$, and hence $|t_0| \leq bw(U) \leq n$. For the inductive step, consider the sequence $e' = s_1, t_1, \ldots, s_q, t_q$. e' is an eager descent of length $q-1$, hence $|t_1| \leq n$. Now by Lemma 4 we have that $|t_0| - |t_1| \leq maxdis(B)$, implying that $|t_0| \leq n + maxdis(B)$. But if $|t_0| \geq n$ then by (2) we have $t_0 \in \uparrow Pred^*([U]_n)$, which implies $|t_0| \leq n$ since e is an eager descent.

(Only if) Choose $v' \in \uparrow Pred^*([U]_j)$ for some $j \geq n$. Then there exists $v \in Pred^*([U]_j)$ such that $v \preceq v'$. From (1) it follows that there exists $u \in Pred^*([U]_i)$ for some $1 \leq i \leq n$ such that $u \preceq v$. Let u' be any vector such that $u \preceq u' \preceq v$ and $|u'| = n$. From Lemma 1, we have that $u' \in Pred^*([U]_n)$ and hence $v' \in \uparrow Pred^*([U]_n)$. □

Corollary 1 *The least n such that (2) holds is* $\max(bw(Pred^*(U)), bw(U))$.

Proof: Follows from Theorem 1. □

Theorem 1 tells us that we can terminate our algorithm (Fig. 2) once we arrive at a value of i such that (2) holds for $n = i - maxdis(B)$. In Sect. 3.3 we will show how the set containments of (2) can be checked efficiently.

The reader may wonder if the $maxdis(B)$ trace containments of (2) are overkill in the sense that convergence is implied by fewer containments. For example, perhaps $\uparrow Pred^*([U]_{n+1}) \subseteq \uparrow Pred^*([U]_n)$ implies (2), and hence this single containment is necessary and sufficient for convergence. Unfortunately this is not true, as the following theorem formalizes.[6]

Theorem 2. *For each $k \geq 1$, there exists a broadcast protocol B such that $maxdis(B) = k$, an upward-closed set U, and $n \geq bw(U)$ such that*

$$\uparrow Pred^*([U]_{n+k-2}) \subseteq \uparrow Pred^*([U]_{n+k-3}) \subseteq \cdots \subseteq \uparrow Pred^*([U]_n) \tag{3}$$

and

$$\uparrow Pred^*([U]_{n+k-1}) \not\subseteq \uparrow Pred^*([U]_{n+k-2}) \tag{4}$$

Proof: Taking k to be fixed but arbitrary, define $B = \{(M, c)\}$ where M is the 3×3 identity matrix and $c = \langle k-1, -k, 1 \rangle^T$. Defined U to be $\uparrow \{\langle 0, k-1, 1 \rangle^T\}$, then $bw(U) = k$, and let $n = k$. The reader may verify that $Pred^*([U]_j) = [U]_j$ for each $j \in \{n, \ldots, n+k-2\}$, and hence for each such j we have $\uparrow Pred^*([U]_j) = \uparrow [U]_j$. Thus (3) holds. However, $Pred^*([U]_{n+k-1}) = [U]_{n+k-1} \cup \{\langle 0, 2k-1, 0 \rangle^T\}$ (note that this is a disjoint union). Hence (4) is also satisfied. □

[6] We note that Theorem 2 only shows the necessity of checking $maxdis(B) - 1$ containments. Whether the $maxdis(B)$ containments of (2) are necessary is open. Theorem 2 does demonstrate, however, that $maxdis(B)$ is asymptotically tight.

3.2 Checking for Intersection with I

Another aspect of our algorithm that is yet to be defined is the function *intersection_check* (cf. Fig. 2). The goal of this function is to return true if we can ascertain that $Pred^*(U) \cap I \neq \emptyset$ by examining $Pred^*([U]_n)$. In this section we provide a necessary and sufficient condition for this intersection being nonempty that is fit for use in our algorithm. For $Y \subseteq \{1,\ldots,m\}$, define the partial order $\preceq_Y$ on $\mathbb{N}^m$ such that $v \preceq_Y u$ iff for all $i \in Y$ we have $v[i] \leq u[i]$.

Theorem 3 (intersection check). *Let $P = \{x \mid x[1] \sim_1 a[1]) \wedge \cdots \wedge x[m] \sim_m a[m]\}$ be a parametric set, let X be an upward-closed set, and let $r \geq bw(X)$. Then $P \cap X = \emptyset$ if and only if $\bigcup_{i=1}^{r} [X]_i$ does not contain v such that $v \preceq_E a$, where $E = \{i \mid \sim_i \text{ is } =\}$ and a is the root vector of P.*

Proof: Let $Y = \bigcup_{i=1}^{r} [X]_i$. **(Only if)** Suppose Y contains $v \preceq_E a$. Then let v' be defined by

$$v'[i] = \begin{cases} a[i] & \text{if } i \in E \\ \max(a[i], v[i]) & \text{otherwise} \end{cases}$$

Then $v' \in X$, since $v \in X$ and $v \preceq v'$ and X is upward-closed, and also $v' \in P$.

(If) Suppose there exists $v \in P \cap X$, then $v \preceq_E a$. If $1 \leq |v| \leq r$ we are done. Otherwise, there exists $u \in gen(X)$ such that $u \preceq v$, which implies $u \preceq_E a$ since $\preceq_E$ is transitive and $\preceq$ is stronger than $\preceq_E$. But since $|u| \leq r$, we have $u \in Y$. □

Since I is parametric and $Pred^*(U)$ is upward-closed (by Lemma 1), we can apply Theorem 3 to determine if the intersection of these sets is empty. The suggested implementation of *intersection_check* takes $Pred^*([U]_n)$ and simply tests if this set has a nonempty intersection with $[\{v \mid v \preceq_E a\}]_n$. Decidability of this problem follows from the fact that both of these sets are finite.

3.3 Using BDDs

Our discourse so far has dealt with the semantics of a broadcast protocol as a transition system $(\mathbb{N}^m, \rightarrow)$. Broadcast protocols were originally devised to model the composition of an unbounded number of identical communicating finite state processes, i.e. a parameterized family of finite state systems [17, 18]. We let $L = \{\ell_1, \ldots, \ell_m\}$ denote the local states of an individual process. For the system instance with n processes, a *global state* is a vector $g \in L^n$, where g_i gives the local state of the ith process. A vector $v \in \mathbb{N}^m$ is said to *abstract* a global state $g \in L^{|v|}$ if $\left|\{j \mid g_j = \ell_i\}\right| = v_i$ for each $1 \leq i \leq m$. Of course g is not in general unique; v in fact abstracts a set of global states that are equivalent under a symmetry relation.

This distinction between abstract and concrete states must be made; our theory of the previous sections pertains to the former while in this section we will instantiate our algorithm to manipulate sets of the latter. The concretization function γ maps abstract states to sets of concrete states: given $v \in \mathbb{N}^m$, $\gamma(v)$ is the subset of $L^{|v|}$ consisting of all concrete states abstracted by v. We extend γ to work on a set A of abstract states by $\gamma(A) = \bigcup_{v \in A} \gamma(v)$. The abstraction function α will take a concrete state c to the abstract

state v such that $\gamma(v) = c$; we extend α to work on sets in the obvious way. The set of all concrete states is $C = \gamma(\mathbb{N}^m)$.

Binary decision diagrams (BDDs) are a popular data structure for representing and manipulating boolean functions [5]. BDDs can be used to store arbitrary finite sets by encoding elements using boolean variables and building a BDD for the characteristic function. BDDs can also represent relations over finite sets. We employ BDDs to represent sets of tuples of concrete states and binary relations over such sets. Subsequently, we identify all sets in the concrete domain with the corresponding BDD, which is unique up to isomorphism, and assume some reasonable boolean encoding of the elements of L. We use $\Rightarrow$ and $\Leftrightarrow$ to respectively denote set containment and set equivalence between BDDs (i.e. logical implication and logical equivalence).

Definition 4 (existential lifting). *Let $S \subseteq L^n$ be a set of concrete states. Then the* existential lifting S^{el} *of S is the subset of L^{n+1} such that*

$$\langle c_1, \ldots, c_{n+1} \rangle \in S^{\mathrm{el}} \;\Leftrightarrow\; \exists i \in \{1, \ldots, n+1\} : \langle c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n+1} \rangle \in S$$

Intuitively, the states of S^{el} are precisely those that can be transformed into a state of S by deleting one component. The BDD for S^{el} can be computed from the BDD for S.

Definition 5 (symmetric). *A set $S \subseteq L^n$ is* symmetric *if $c \in S$ implies $\gamma(\alpha(c)) \subseteq S$.*

Theorem 4. *For symmetric sets $X \subseteq L^n$ and $Y \subseteq L^{n+1}$, $\uparrow\alpha(Y) \subseteq \uparrow\alpha(X)$ if and only if $Y \Rightarrow X^{\mathrm{el}}$.*

Proof: (If) Suppose $Y \Rightarrow X^{\mathrm{el}}$, and let $y \in \uparrow\alpha(Y)$. Then there exists $c \in Y$ such that $\alpha(c) \preceq y$. We have that $c \in X^{\mathrm{el}}$, hence by Def. 4 there exists $i \in \{1, \ldots, n+1\}$ such that $c' = \langle c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n+1} \rangle \in X$. Let j be such that $c_i = \ell_j$. Then $\alpha(c') = \alpha(c) - \mathrm{inc}^j$, hence $\alpha(c') \preceq y$, and $y \in \uparrow\alpha(X)$. **(Only if)** Let $c = \langle c_1, \ldots, c_{n+1} \rangle \in Y$. Then $\alpha(c) \in \uparrow\alpha(Y)$, and thus $\alpha(c) \in \uparrow\alpha(X)$. Hence there exists $x \in \alpha(X)$ such that $x \prec \alpha(c)$. In particular, $x = \alpha(c) - \mathrm{inc}^j$ for some $1 \leq j \leq m$, since $|\alpha(c)| = |x| + 1$. There must exist $1 \leq i \leq n+1$ such that $c_i = \ell_j$, since $\alpha(c)[j] \geq 1$. Let $c' = \langle c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{n+1} \rangle$. We note that $\alpha(c') = x$, thus $c' \in X$ since X is symmetric. Finally, $c' \in X$ implies $c \in X^{\mathrm{el}}$, thus $Y \Rightarrow X^{\mathrm{el}}$. □

A detailed version of our algorithm is given in figure 4. This version elaborates on the skeleton of figure 2 in three ways: 1) it explicates the convergence condition, 2) it explicates the initial states intersection check, and 3) it processes BDDs representing concrete state sets. Application of the concretization operator γ indicates a BDD representing the resulting set of concrete states, which is easy to construct. The algorithm employs the *concrete predecessor function* $Pred_\gamma : 2^C \to 2^C$ defined by $Pred_\gamma(S) = \gamma(Pred(\alpha(S)))$. Intuitively, $Pred_\gamma$ mimics $Pred$ in the concrete domain. In order to compute the BDD Γ_n, a BDD representing $Pred_\gamma$ is needed. This BDD is easily built by inverting the transition relation from the definition of the broadcast protocol B.

Theorem 5. *The algorithm of figure 4 solves BPRP.*

```
i := 1
n := 1
Γ_0 := ∅
while (n ≥ i − maxdis(B) ∨ i ≤ bw(U)) do
    compute Γ_i := Pred*_γ(γ([U]_i))
    if (γ([{v | v ⪯_E a}]_i) ∩ Γ_i ≠ ∅) then
        exit with verification failure
    if (¬(Γ_i ⇒ Γ^el_{i−1})) then
        n := i
    i := i + 1
exit with verification success
```

Fig. 4. Our algorithm, version 2

Proof: Note that $\gamma([U]_i)$ is always symmetric, and hence so too is Γ_i. Thus, by Theorem 4 the condition of the second `if` statement is equivalent to checking $\uparrow Pred^*([U]_i) \subseteq \uparrow Pred^*([U]_{i-1})$. Suppose the algorithm exits with successful verification. Let n_f be the final value of n. Then the condition of the second `if` statement was false for each of the final $maxdis(B)$ iterations, i.e. the above containment held for each $i \in \{n_f+1,\ldots,n_f+maxdis(B)\}$. Hence we have (2), and by the Convergence Theorem, $Pred^*(U) = \uparrow \bigcup_{j=1}^{n_f} Pred^*([U]_j)$ and by Corollary 1, $n_f \geq bw(Pred^*(U))$. Therefore, by the Intersection Check Theorem, since the condition of the first `if` was never satisfied, we can conclude $Pred^*(U) \cap I = \emptyset$. Now suppose the algorithm exits with verification failure. Then by the Intersection Check Theorem, we can conclude $Pred^*(U) \cap I \neq \emptyset$. Finally, termination is guaranteed by Corollary 1. □

3.4 An Optimization

In this section we propose an optimization to the algorithm of Fig. 4. Consider the computation of Γ_i. This involves an iterative fixpoint computation, starting with the BDD for $\gamma([U]_i)$. In some sense, much of the work of this computation was already performed when computing Γ_{n-1}. This is articulated (in the abstract domain) by noting that $v \in Pred^*([U]_{i-1})$ implies $[\uparrow\{v\}]_i \subseteq Pred^*([U]_i)$. Hence, we already "know" that $[\uparrow Pred^*([U]_{i-1})]_i$ is contained in $Pred^*([U]_i)$. When we start the fixpoint computation from $[U]_i$, we are in effect doing redundant work to "rediscover" these elements. We let Y_i denote $[\uparrow Pred^*([U]_{i-1})]_i \cup [U]_i$.

Lemma 5 $Pred^*(Y_i) = Pred^*([U]_i)$.

Proof: The $\supseteq$ direction is obvious. Conversely, choose $v \in Pred^*(Y_i)$. Then there exists a path from v to $y \in Y_i$. If $y \in [U]_i$ we are done. Otherwise there exists $u \in Pred^*([U]_{i-1})$ such that $y = u + \mathrm{inc}^j$ from some $1 \leq j \leq m$. Then there exists a path from u to $x \in [U]_{i-1}$. Since broadcast protocols are well-structured transition systems, this implies that there exists $z \in [\uparrow[U]_{i-1}]_i \subseteq [U]_i$ such that there is a path from y to z. Hence there is a path from v to z, and $v \in Pred^*([U]_i)$.

```
i := 1
n := 1
Γ_0 := ∅
while (n ≥ i − maxdis(B) ∨ i ≤ bw(U)) do
    if (i > bw(U)) then
        compute Γ_i := Pred*_γ(Γ^el_{i−1})
    else
        compute Γ_i := Pred*_γ(Γ^el_{i−1} ∪ γ([U]_i))
    if (γ([{v | v ⪯_E a}]_i) ∩ Γ_i ≠ ∅) then
        exit with verification failure
    if (¬(Γ_i ⇒ Γ^el_{i−1})) then
        n := i
    i := i + 1
exit with verification success
```

Fig. 5. Our algorithm with optimization of Sect. 3.4. The changed lines are indicated by the vertical bar.

□

Lemma 6 $\gamma(Y_i) = \Gamma^{\mathrm{el}}_{i-1} \cup \gamma([U]_i)$.

Proof: Follows from definition 4 and the definition of Y_i. □

Lemma 7 *If* $i > bw(U)$, *then* $\gamma(Y_i) = \Gamma^{\mathrm{el}}_{i-1}$.

Proof: Follows from Lemma 6 and the fact that when $i > bw(U)$ we have $[U]_i \subset \uparrow[U]_{i-1}$.
□

The optimized algorithm is given in Fig. 5, the only change from Fig. 4 being the computation of Γ_i. This optimization has the potential to greatly reduce the number of iterations performed in the fixpoint computations. As an extreme example, in an iteration of the outer loop for which $\Gamma_i \Rightarrow \Gamma^{\mathrm{el}}_{i-1}$ holds, the computation of Γ_i will only ever involve a single iteration of the fixpoint routine.

Theorem 6. *The optimization of Fig. 5 preserves correctness.*

Proof: From Lemmas 5, 6, and 7, both algorithms compute the same Γ_i for each i. □

4 Petri Nets

In this section we show how we leverage any algorithm that solves BPRP to solve a similar problem regarding petri nets. A *petri net* is a quadruple (P,T,F,W) where $P = \{p_1,\ldots,p_m\}$ is a finite set of *places* which is disjoint from the finite set of *transitions* T. $F \subseteq (P \times T) \cup (T \times P)$ is called the *flow relation*, and $W : F \to \mathbb{N} \setminus \{0\}$ is the *weighting function* (not related to the notion of vector weight). A *marking* of a net is a vector $v \in \mathbb{N}^m$. Intuitively, $v[i]$ gives the number of *tokens* that marking v puts at place p_i for

each $1 \leq i \leq m$; hence in a slight abuse we identify v with the function $P \to \mathbb{N}$ such that $v(p_i) = v[i]$. The *pre-vector* of $t \in T$ is the vector ${}^\bullet t$ defined by ${}^\bullet t[i] = W((p_i,t))$ if $(p_i,t) \in F$ and ${}^\bullet t[i] = 0$ otherwise. Analogously, the *post-vector* of t is $t^\bullet$ defined by defined by $t^\bullet[i] = W((t,p_i))$ if $(t,p_i) \in F$, otherwise $t^\bullet[i] = 0$.

Similar to a broadcast protocol, the semantics of a petri net (P,T,F) is a transition system $(\mathbb{N}^m, \to)$, where $m = |P|$. The transition relation $\to$ is the greatest subset of $\mathbb{N}^m \times \mathbb{N}^m$ such that $v \to v'$ implies there exists $t \in T$ such that ${}^\bullet t \preceq v$ and $v' = v - {}^\bullet t + t^\bullet$.

Definition 6 (petri net coverability problem). *The* petri net coverability problem *(PNCP) asks, given a petri net N, a parametric set of initial markings I, and an upward-closed set U of markings, is there an element of U reachable from an element of I in N?*

The following Theorem 7 allows us to apply our algorithm for BPRP to PNCP.

Theorem 7. *There exists an effective reduction from the petri net coverability problem to the broadcast protocol reachability problem.*

Proof: (sketch) We say that a place/transition pair (p,t) is a *self-loop* if both $(p,t) \in F$ and $(t,p) \in F$; a petri net is called *pure* if it has no self-loops. A transition is called *token-preserving* if $|{}^\bullet t| = |t^\bullet|$; we say a net is token-preserving if all its transitions are. There is a straightforward reduction when the petri net is both pure and weight preserving; the resulting broadcast protocol B has an element (I,x) for each transition, where the vector x performs the effect of the transition and I is the identity matrix. We can "purify" an instance of PNCP by using a simple refinement on the petri net [27] and updating I and U in accord with this refinement. Finally, weight preservation can be achieved by adding an auxiliary place p_{nw} to which destroyed token are deposited and from which created tokens are obtained. The new versions of both I and U place an arbitrary number of tokens on p_{nw}. □

5 Experimental Results

As a proof of concept, we have implemented our algorithm as a pair of tools `translate` and `bucsub`. `bucsub` implements our algorithm using the CUDD BDD package [30], and requires its input to be in SMV format. `translate` takes a simple petri net description (extended to handle broadcast transitions) and produces the necessary SMV files for `bucsub`; these SMV files describe the concrete transitions systems.

To demonstrate the existence of examples for which our approach outperforms the standard approach, we constructed the "parameterized petri net" $ME(h)$ depicted in Fig. 6. The parameter h dictates the width of the chain of places $x_1, \ldots, x_h$ along the top of the figure. h allows us to control the size of the local state space in the resulting broadcast protocol. For any h, the initial marking of $ME(h)$ places a single token at the place *notin*, and an arbitrary number k at place x_0. A token can move from x_0 to x_1 when there is a token at *notin* by firing transition a. This will move the token at *notin* to *in*, which will disallow any more tokens from entering x_1. The token that is in the chain can

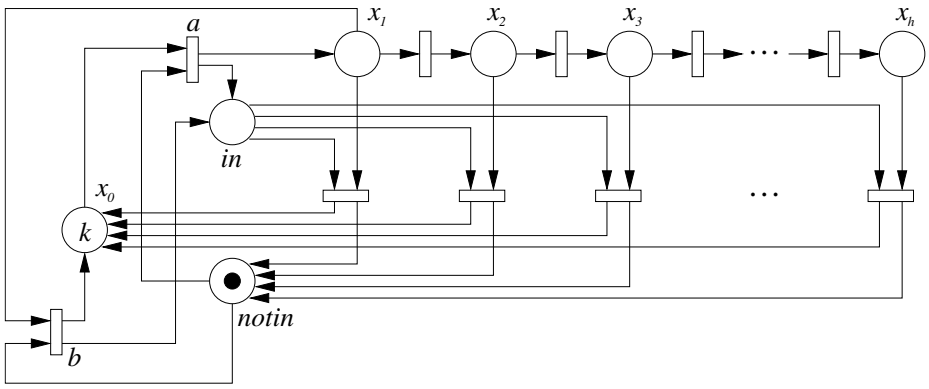

Fig. 6. The petri net $ME(h)$ used in the experiments

move along the chain, and at any point it may hop back to x_0, which will result in the token at *in* being moved back to *notin*. Hence $ME(h)$ implements mutual exclusion in that at most one token can be in the chain at any time. For our experiments, we verified that there can only be at most one token at x_h; i.e. the upward-closed set of markings $\{m \mid m(x_h) \geq 2\}$ is not reachable.

We measured execution times[7] for $h = 25, 50, \ldots, 250$; the results are plotted in Fig. 7. The plot labelled "CST" gives the runtime for the Delzanno et al.s CST approach[8] [12, 13]. The plot labelled "BDD" gives the total runtime for both tools `translate` and `bucsub`, while "BDD minus translation" provides the runtime of `bucsub` only. The latter is presented because the current `translate` phase is suboptimal; in a sophisticated implementation there would be no need for the intermediate SMV files and hence the time in this phase would be highly mitigated.

From Fig. 7 we see that for $ME(h)$ our approach is two orders of magnitude faster than the state-of-the art implementation of the standard approach (for large h).

6 Discussion

In this section we compare our approach with the standard approach (i.e. CSTs) and outline future research directions.

Convergence Given two CSTs C_1 and C_2, the problem of checking if C_1 *subsumes* C_2 (i.e. if the upward-closed set represented by C_1 is a superset of that of C_2) is co-NP hard in the size of the involved CSTs [12]. Unfortunately, checking subsumption is an integral part of the standard algorithm (cf. the `while` condition in Fig. 1). To combat this problem, Delzanno et al. develop a sophisticated heuristic solution in which certain

[7] Experiments were executed on a 2.6 GHz Intel Pentium 4 machine running Red Hat Linux 9.

[8] In fact, the reported run-times are those of software based on *interval sharing trees* (IST) which are an extension of CSTs to handle two-sided constraints [21]. The IST software used is a constant 3 or 4 times slower than a pure CST implementation.

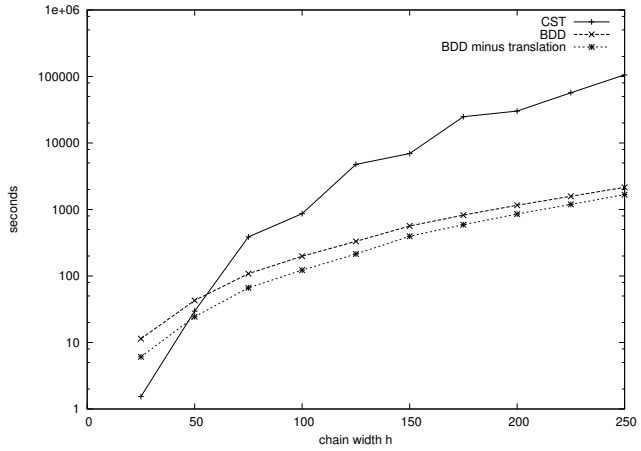

Fig. 7. Experiment execution times for the two approaches. The vertical axis gives the runtime is seconds; the horizontal axis gives the petri net parameter h. Our approach is labelled "BDD", while the CST approach of Delzanno et al. is labelled "CST".

CST simulation relations facilitate pruning of an (exponential time) exact subsumption check [13].

In contrast, subsumption between two BDDs can be decided in time proportional to the product of their sizes [5]. In fact, we can correctly replace the condition of the second `if` statement of Fig. 4 with a bidirectional subsumption: $\neg(\Gamma_i \Leftrightarrow \Gamma^{\mathrm{el}}_{i-1})$. This test can be done *in constant time* for BDDs [5].

Data structure size Another indication of the efficiency of the two algorithms is the size of the underlying data structures. Predicting the dynamics of the sizes is a complex problem. Though BDDs compactly represent many practical boolean functions, the worst case size is exponential in their height (i.e. the number of boolean variables). To the author's knowledge, bounds on the size of CST have not been derived in the literature, however, any such bound is at least exponential in the height of the structure. Here we consider data structure height as a course measure of worst-case size.

The height of the CSTs is fixed at $|L|$, while the height of the BDDs is at most $(n_f + \mathit{maxdis}(B))\lceil \log_2 |L| \rceil$, where n_f is the final value of n in our algorithm (which, by Corollary 1 is $\max(\mathit{bw}(\mathit{Pred}^*(U)), \mathit{bw}(U))$). For example, consider our petri net $ME(250)$, where $|L| = 253$, $n_f = 4$, and $\mathit{maxdis}(B) = 2$. In this case our BDDs have height at most 48, while the CSTs have height 253. This rudimentary analysis suggests that our approach appears to have an advantage when $|L|$ is large and $n_f + \mathit{maxdis}(B)$ is modest. Unfortunately, only the lower bound $\mathit{bw}(U)$ of n_f is known a priori.

Future work Clearly further comparison between our approach and Delzanno et al.'s implementation of the standard algorithm is necessary. We also plan on deducing the bound on the size of a BDD for S^{el} in terms of the size of S for symmetric S as such a result would help further quantify the complexity of our algorithm. It is likely that our method can be applied to other discrete well-structured systems, hence we intend to pursue such extensions.

7 Related Work

BPRP is an instance of the *parameterized model checking problem* (PMCP). PMCP is an umbrella term that refers to any verification problem of the form

$$\forall n \geq 1 : P(n) \vDash \phi$$

where ϕ is some property, and for each $n \geq 1$, $P(n)$ is a (usually finite state) system. The $P(n)$s are typically related in some fundamental way, for example $P(n)$ is often the composition of n identical process templates. A general statement of PMCP was shown to be undecidable in [3].

Well-structured transition systems (WSTS) were first proposed by Finkel [19]. The upward-closed set backward reachability algorithm for WSTS (i.e. what we dub the standard approach) was first articulated by Abdulla et al. [1], in which the application to petri nets is observed. Finkel and Schnoebelen generalize the notion and provide assorted examples in [20].

Broadcast protocols and a problem closely resembling our BPRP were introduced by Emerson and Namjoshi [17]. Their procedure, which is based on the Karp-Miller petri net covering graph [24], was shown to not necessarily terminate by Esparza et al. [18]. Decidability of safety properties for broadcast protocols was established in [18] by showing how the backward reachability algorithm of [1] could applied to this problem. Delzanno et al. have investigated means of symbolically representing upward-closed sets in the backward reachability algorithm [11–13] with encouraging results. In [10], Delzanno advances the standard algorithm for broadcast protocols in two ways: 1) the permitted transition guards are generalized (though this generalization causes the approach to be only semi-algorithmic) , and 2) he shows how efficient real (as opposed to integer) constraint solvers can be used to implement the algorithm.

So-called *cut-off* results are those that reduce the correctness of a class of parameterized system to the correctness of some of the first n constituents, where n is typically a function of the size of a single process. German and Sistla consider CCS style processes and prove exponentially sized cut-offs for regular properties among other PMCP results [22]. For protocols with either conjunctive or disjunctive guards, Emerson and Kahlon give cut-offs for checking LTL\X formula [14]. Our approach can be viewed as computing the minimal cut-off for the broadcast protocol being model checked, whereas these previous works infer cut-offs for entire classes of systems. In [28, 4], Pnueli et al. give smaller cut-off results for a more expressive class of systems, but this benefits come at a cost since completeness is sacrificed.

Emerson and Kahlon also provide a thorough study of PMCP decidability for various combinations of properties and communication primitives, which includes broadcasts [16]. The same authors have developed an efficient approach to PMCP for a class of cache coherence protocols (with broadcasts) [15].

Network invariants are a non-automatic approach to PMCP [7, 26, 31, 25]. Here the user provides a process I called the network invariant. I attempts to abstract the composition of an unbounded number of system processes $P^n = P||P|| \cdots ||P$ (with n Ps). Proof obligations typically involve verifying that $P||I$ is abstracted by I, that P is abstracted by I, and that $I \vDash \phi$, which can all be dispatched to finite state model checking. From these

premises we can conclude $\forall n : P^n \models \phi$. A drawback of using network invariants is that they are often a challenge to produce. Further, network invariants are not guaranteed to exist in general [31, 2].

Abstraction [8] has been used to tackle PMCP, such approaches are are usually incomplete. Several works use replication abstraction to reduce an infinite state system to finite state [29, 23]. These involve abstracting global states by tracking if there are zero, exactly one, or some multiplicity of processes in each local state. Predicate abstraction has also been successfully applied to PMCP, for example [9].

Acknowledgement

I thank my supervisors Anne Condon and Alan J. Hu for their valuable feedback on this paper. I thank Pierre Ganty, Laurent Van Begin, and Giorgio Delzanno for equipping me with the IST/CST software used in the experiments and for their useful responses to my emails. I thank Armin Biere for writing and making public the tools `SmvFlatten` and `FlatSMV`, which were integral to the implementation of our algorithm. Finally, I thank an anonymous CONCUR 2004 reviewer who gave many useful suggestions.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. In *10th Annual IEEE Symp. on Logic in Computer Science (LICS'96)*, pages 313–321, 1996.
2. P. A. Abdulla and B. Jonsson. On the existence of network invariants for verifying parameterized systems. In *Correct System Design – Recent Insights and Advances, LNCS*, volume 1710, 1999.
3. K. Apt and D. Kozen. Limits for automatic verification of finite-state concurrent systems. *Information Processing Letters*, 15:307–309, 1986.
4. T. Arons, A. Pnueli, S. Ruah, J. Xu, and L. Zuck. Parameterized verification with automatically computed inductive assertions. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV 2001)*, volume 2102, pages 221–234, 2001. Lecture Notes in Computer Science.
5. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
6. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Conference on Logic in Computer Science*, pages 428–439, 1990. An extended version appeared in *Information and Computation*, Vol. 98, No. 2, June 1992.
7. E. M. Clarke, O. Grumberg, and M. Browne. Reasoning about networks with many identical finite state processes. In *Proceedings of the 5th ACM Symposium on Principles of Distributed Computing*, pages 240–248, New York, 1986. ACM.
8. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, 1977.

9. S. Das, D. L. Dill, and S. Park. Experience with predicate abstraction. In *11th International Conference on Computer Aided Verification*, 1999.
10. G. Delzanno. Automatic verification of parameterized cache coherence protocols. In *Proceedings of the 12th International Conference on Computer Aided Verification*, July 2000.
11. G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In *Proceedings of the Annual Conference of the European Association for Computer Science Logic, LNCS*, volume 1683, pages 50–66, 1999.
12. G. Delzanno and J. F. Raskin. Symbolic representation of upward-closed sets. In *6th International Conference Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, pages 426–440, 2000.
13. G. Delzanno, J. F. Raskin, and L. Van Begin. Attacking symbolic state explosion. In *Proceedings of the 13th International Conference on Computer-Aided Verification (CAV)*, pages 298–310, 2001.
14. E. A. Emerson and V. Kahlon. Reducing model checking of the many to the few. In *17th International Conference on Automated Deduction*, pages 236–254, Pittsburgh PA, July 2000.
15. E. A. Emerson and V. Kahlon. Exact and efficient verification of parameterized cache coherence protocols. In *12th IFIP Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME)*, October 2003.
16. E. A. Emerson and V. Kahlon. Model checking guarded protocols. In *Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 361–370, June 2003.
17. E. A. Emerson and K. S. Namjoshi. On model checking for non-deterministic infinite-state systems. In *Proceedings of LICS 1998*, pages 70–80, 1998.
18. J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proceedings of LICS '99*, pages 352–359, 1999.
19. A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.
20. A. Finkel and Ph. Schnoebelen. Well structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
21. P. Ganty and L. Van Begin. Non deterministic automata for the efficient verification of infinite-state. presented at: CP+CV Workshop at European Joint Conferences on Theory and Practice of Software (ETAPS), 2004.
22. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, July 1992.
23. C. N. Ip and D. L. Dill. Verifying systems with replicated components in murphi. In *International Conference on Computer-Aided Verification*, 1996.
24. R. M. Karp and R. E. Miller. Parallel Program Schemata. *Journal of Computer and System Sciences*, 3:147–195, 1969.
25. Y. Kesten, A. Pnueli, E. Shahar, , and L. Zuck. Network invariants in action. In *Proc. 13th Conference on Concurrency Theory*, 2002.
26. R. P. Kurshan and K. L. McMillan. A Structural Induction Theorem for Processes. *Information and Computation*, 117(1), Feb 1995.
27. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), April 1989.
28. A. Pnueli, S. Ruah, and L. Zuck. Automatic deductive verification with invisible invariants. In Tiziana Margaria and Wang Yi, editors, *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, (TACAS 2001)*, volume 2031 of *Lecture Notes in Computer Science*, pages 82–97, 2001.
29. F. Pong and M. Dubois. A new approach for the verification of cache coherence protocols. *IEEE Trans. on Parallel and Distributed Systems*, 6(8), August 1995.
30. F. Somenzi. Colorado university decision diagram package (CUDD) webpage. http://vlsi.colorado.edu/∼fabio/CUDD/cuddIntro.html.

31. P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *Automatic Verification Methods for Finite State Systems, Proc. Int. Workshop*, pages 68–80, 1989. volume 407 of LNCS.