Empirically Efficient Verification for a Class of Infinite-State Systems*

Jesse Bingham and Alan J. Hu
Department of Computer Science, University of British Columbia

Abstract. *Well-structured transition systems* (WSTS) are a broad and well-studied class of infinite-state systems, for which the problem of verifying the reachability of an upward-closed set of error states is decidable (subject to some technicalities). Recently, Bingham proposed a new algorithm for this problem, but applicable only to the special cases of broadcast protocols and petri nets. The algorithm exploits finite-state symbolic model checking and was shown to outperform the classical WSTS verification algorithm on a contrived example family of petri nets.
In this work, we generalize the earlier results to handle a larger class of WSTS, which we dub *nicely sliceable*, that includes broadcast protocols, petri nets, context-free grammars, and lossy channel systems. We also add an optimization to the algorithm that accelerates convergence. In addition, we introduce a new reduction that soundly converts the verification of parameterized systems with unbounded conjunctive guards into a verification problem on nicely sliceable WSTS. The reduction is complete if a certain decidable side condition holds. This allows us to access industrially relevant challenge problems from parameterized memory system verification. Our empirical results show that, although our new method performs worse than the classical approach on small petri net examples, it performs substantially better on the larger examples based on real, parameterized protocols (e.g., German's cache coherence protocol, with data paths).

1 Introduction

The widespread practical success of finite-state model checking [9, 29] has stimulated interest in the algorithmic verification of infinite-state systems. The goal is to verify systems that are naturally modelled as infinite state as well as systems that might be finite-state in practice, but that are too large to be verified via finite-state methods in the foreseeable future (e.g., pushdown automata to model a program's call stack, parameterized memory system protocols to model a realistically-sized memory system).

Well-structured transition systems (WSTS) [19, 2, 20] are a broad class of infinite-state systems, for which an extensive and elegant body of research has developed. In particular, the verification problem of determining the reachability of an upward-closed set of error states is decidable (provided some side conditions are satisfied) via an algorithmic framework we call the *classical approach* [2, 20, 18].

* This is an an extended version of [5] that differs from [5] in two ways. First, an appendix with proofs is added. Second, two minor typographical errors in Sect. 6 and one in Def. 10 are corrected. This work was supported in part by a UBC Li Tze Fong Memorial Fellowship and a grant from the Natural Sciences and Engineering Research Council of Canada.

Recently, Bingham proposed a new algorithm for this problem [4]. Unlike the classical approach, the new algorithm works by computing fix-points over a series of finite-state systems of increasing size, allowing the leveraging of sophisticated techniques from finite-state model checking. However, the theory was developed only for a special case of WSTS, namely, broadcast protocols (which subsume petri nets). Using finite-state symbolic model checking [7], Bingham demonstrated a contrived family of petri nets for which the new algorithm substantially outperformed the classical approach.

This paper generalizes and extends the earlier work in several ways. We introduce a new subclass of WSTS and generalize the earlier theory and algorithms to apply to the subclass. We show how the new subclass subsumes petri nets, broadcast protocols, lossy channel systems, and context-free grammars. We introduce an optimization to the algorithm that accelerates convergence. We also provide a new reduction that allows soundly applying our verification method to certain protocols with unbounded conjunctive guards, which are not WSTS, as commonly occurs in memory system protocols. Finally, we give experimental evidence on a variety of infinite-state systems, including German's parameterized cache coherence protocol [23], a widely cited verification challenge problem.

Proofs of all theorems in this paper can be found in the appendix.

2 Preliminaries

Let $\mathbb{N}$ denote the natural numbers. We use various notations for orderings: $\preceq$ will denote an arbitrary reflexive and transitive relation (which may satisfy stronger requirements depending on context), and we write $x \prec y$ to mean $x \preceq y \wedge y \not\preceq x$. The symbol $\leq$ will denote the usual ordering on the reals and subsets thereof, and for any positive dimension m, we extend $\leq$ to be the usual point-wise vector ordering over $\mathbb{N}^m$ defined by $v \leq u$ iff $v_i \leq u_i$ for all $1 \leq i \leq m$. We also employ $\leq$ as the covering relation between petri net markings.

The systems we consider are a certain type of *well-structured transition system*, and the "bad" states will be characterized by an *upward-closed* set. These and other relevant notions are now defined, mostly following the terminology of [20].

Definition 1 (upward-closure,basis,upward-closed set). *Let $\preceq$ be a reflexive and transitive relation over a set X. For $Y \subseteq X$, the* upward-closure *of Y is the set $\uparrow Y = \{x \mid \exists y \in Y : y \preceq x\}$. When $U = \uparrow Y$ we say that Y is a* basis *for U. A set U is said to be $\preceq$-*upward-closed *(or simply* upward-closed *if $\preceq$ is clear from context) if $U = \uparrow U$.*

Definition 2 (well-quasi-ordering). *A* well-quasi-ordering (wqo) *is a reflexive and transitive relation $\preceq$ over a set X such that for any infinite sequence $x_0, x_1, x_2, \ldots$ over X, there exists $i, j \in \mathbb{N}$ such that $i < j$ and $x_i \preceq x_j$.*

Lemma 1. *[25] If $\preceq$ is a wqo, then any $\preceq$-upward-closed set has a unique finite basis B such that for all $x, y \in B$ we have $x \not\preceq y \wedge y \not\preceq x$.*

Given upward-closed U, we let $\mathsf{basis}(U)$ denote the unique finite basis of U, the existence of which is guaranteed by Lemma 1.

Definition 3 (well-structured transition system). *A* well-structured transition system *(WSTS) is a triple $(S, \rightarrow, \preceq)$ such that*

```
previous_reach, reach : finite subset of S
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 classical algorithm

1. *S is a (possibly infinite)* state space
2. *$\rightarrow \subseteq S \times S$ is called the* transition relation
3. *$\preceq$ is a wqo over S*
4. *For all $x, x', y \in S$ such that $x \rightarrow x'$ and $x \preceq y$, there exists $y' \in S$ such that $y \rightarrow y'$.*[1]

The covering relation $\leq$ between petri net markings is a wqo. Given a finite set of markings M, the set $\uparrow M$ includes all markings that cover at least one $m \in M$. Petri nets are WSTS (with respect to $\leq$) [20].

The decision problem regarding WSTS we aim to solve is as follows.

Definition 4 (WSTS Safety Problem). *Given a WSTS $\mathcal{S} = (S, \rightarrow, \preceq)$, an $\preceq$-upward-closed set $U \subseteq S$, and a set of* initial states *$I \subseteq S$, does there exists a sequence $x_0 \rightarrow \cdots \rightarrow x_\ell$ such that $x_0 \in I$ and $x_\ell \in U$? We write $\mathsf{Safe}(\mathcal{S}, I, U)$ (resp., $\neg\mathsf{Safe}(\mathcal{S}, I, U)$) if the answer is "no" (resp. "yes").*

We have intentionally omitted any restrictions on the initial state set I to avoid needlessly complicating this paper. In general I can be infinite, hence a symbolic representation is necessary; for example, [2,4] require that I be a so-called *parametric set*. Decidability of the WSTS Safety Problem depends (in part) on the form of I.

The *classical approach* to this problem is given in Fig. 1 [2,18,20]. On the surface, this algorithm resembles the well-known finite-state backward reachability analysis, i.e. *least fix-point computation*, the difference being that the involved sets are upward-closed (and hence infinite), so a symbolic representation (i.e. finite basis) is necessary. For the approach to work, the following conditions are necessary:

- Given finite $reach \subseteq S$, we must be able to compute another finite set X such that $\uparrow X = \{x \mid \exists y \in \uparrow reach : x \rightarrow y\}$. We denote X by $\mathsf{Pred}(\uparrow reach)$.
- I must be represented in a form that permits the intersection checks of the `if` conditional.

Necessary for practical implementation of the classical algorithm is an efficient representation of *reach*, since this set can become very large. Delzanno et al. propose using a data structure called *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.

[1] This requirement is called *monotonicity* in [2] and *strong compatibility* in [20]. The latter paper gives a slightly weaker definition of WSTS, requiring that y' only satisfy $y \rightarrow^* y'$.

3 Nicely Sliceable WSTS

Our algorithm works on a subclass of WSTS we call *nicely sliceable WSTS* (NSW). To be deemed a NSW, a WSTS must satisfy three properties. We first describe each intuitively and provide some motivation for why they are required, and then we present the formal definitions.

- **Discrete:** The wqo must be discrete, meaning that for any element x, there is a bound on the length of any strictly decreasing sequence starting with x. We call the length of the longest such sequence x's *weight*. Furthermore, discreteness requires that the number of elements of a given weight be finite. Discreteness allows for finite-state model checking to be applied to the subsystem formed by bounding the weight of states.
- **Weight-respecting:** When a transition changes the weight, the same change in weight can be effected by the transition relation for elements greater than the starting state of the transition. Weight-respectfulness is a technical requirement needed for the proof of the Convergence Theorem, which gives a termination condition for our algorithm.
- **Deflatable:** Whenever we have a transition from outside an upward-closed set U to a state in U, deflatability asserts the existence of a similar transition involving states of bounded weight. Deflatability is similar to downward compatibility [20], though the two are incomparable. Deflatability, like weight-respectfulness, is essential in the proof of our Convergence Theorem.

Definition 5 (dwqo,weight function,base weight). *A wqo is a* discrete wqo *(dwqo) over X if for all $x \in X$ there exists $k \in \mathbb{N}$ such for any sequence $x_0 \prec x_1 \prec \cdots \prec x_\ell = x$ we have $\ell \leq k$. Associated with a dwqo $\preceq$ is the* weight function $w : X \to \mathbb{N}$ *that maps each x to the minimum such k. We also require that $\{x \in X \mid w(x) = i\}$ be finite for each $i \in \mathbb{N}$. For $\preceq$-upward-closed U, the* base weight *of U is* $\mathsf{bw}(U) = \max(\{w(x) \mid x \in \mathsf{basis}(U)\})$.

Example 1. For $m \geq 1$, the point-wise vector ordering $\leq$ over $\mathbb{N}^m$ is a dwqo, and for each $v \in \mathbb{N}^m$ we have $w(v) = \sum_{i=1}^{m} v_i$. The set $\{0, 1/2, 2/3, 3/4, \ldots\} \cup \{1\}$ along with $\leq$ is an example of a wqo that is not a dwqo, since taking $x = 1$ violates Def. 5.

Definition 6 (discrete WSTS). *A* discrete WSTS *(DWSTS) is a WSTS $(S, \to, \preceq)$ where $\preceq$ is a dwqo.*

In a DWSTS, the weight function slices the state space into a countable number of finite partitions $S_0, S_1, S_2, \ldots$, where $S_i = \{x \in S \mid w(x) = i\}$.

Example 2. Petri nets along with the marking dominance relation $\leq$ are an example of DWSTSs; the induced weight function simply counts the number of tokens.

Definition 7 (weight respecting DWSTS). *A DWSTS is said to be* weight respecting *if we may strengthen condition 4 of Def. 3 to require that $w(x') - w(x) = w(y') - w(y)$.*

Example 3. Petri nets are weight respecting DWSTSs. Suppose $x \to x'$ by firing transition t, and $x \preceq y$. Since firing t always changes the total number of tokens by the same amount, we can obtain the appropriate y' by firing t from y.

Definition 8 (δ-deflatable DWSTS). *A DWSTS $(S, \rightarrow, \preceq)$ is said to be* δ-deflatable *for $\delta \in \mathbb{N}$ if whenever $x \rightarrow x'$ and $z \preceq x'$, there exists y and y' such that all of the following hold: 1) $y \preceq x$, 2) $y \rightarrow y'$, 3) $z \preceq y'$, 4) $w(y) \leq w(z) + \delta$, and 5) $w(y') \leq w(z) + \delta$. (See Fig. 2.)*

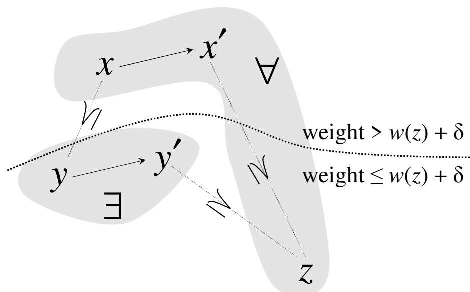

Fig. 2. A diagrammatic presentation of Def. 8. A DWSTS $(S, \rightarrow, \preceq)$ is said to be δ-deflatable (for $\delta \in \mathbb{N}$) if for all $x, x', z \in S$ that satisfy the depicted relations, there exists $y, y' \in S$ that satisfy the depicted relations, and also both $w(y)$ and $w(y')$ are not greater than $w(z) + \delta$.

Example 4. Petri nets are δ-deflatable, where δ is the maximum over all in-degrees and out-degrees of the petri net transitions. A suitable $y \rightarrow y'$ can be constructed by taking only the tokens involved in the firing that takes $x \rightarrow x'$ and adding them to z.

Definition 9 (NSW). *A* δ-NSW *is a DWSTS that is weight-respecting and δ-deflatable. A* NSW *is a δ-NSW for some δ.*

We now give three examples of systems that are NSW.

Example 5. Broadcast protocols (BP), which model the composition of identical finite-state processes, are 2-NSW. Here we roughly follow the definition of [18, 17]. A BP is a triple (L, Σ, R), where L is the set of *local states*, Σ is the set of *labels*, and $R \subseteq L \times \Sigma \times L$. Σ is required to be of the form $\Sigma_l \cup \Sigma_r \times \{!, ?\} \cup \Sigma_b \times \{!!, ??\}$, where Σ_l, Σ_r, and Σ_b are disjoint sets of *actions*, respectively called *local*, *rendez-vous*, and *broadcast* actions. Labels of the form (a, d) are written simply as ad, i.e. $(a, ??)$ is written $a??$. Intuitively, labels of the form $a!$ and $a!!$, are outputs, while those of the form $a?$ and $a??$ are inputs. We make the following restriction on R: for any $a!! \in \Sigma$ and any $s \in L$, there exists $s' \in L$ such that $(s, a??, s') \in R$.

The semantics of a BP (L, Σ, R) is the transition system $(S, \rightarrow)$ where the state space S is the set of all nonempty finite words over L, and $s \rightarrow s'$ iff $s = \ell_1 \ldots \ell_n$ and $s' = \ell'_1 \ldots \ell'_n$, and one of the following hold.

- *local transition*: there exists $1 \leq i \leq n$ and an action $a \in \Sigma_l$ such that $(\ell_i, a, \ell'_i) \in R$, and $\ell'_j = \ell_j$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$.
- *rendez-vous transition*: there exists distinct $i, k \in \{1, \ldots, n\}$ and an action $a \in \Sigma_r$ such that $(\ell_i, a!, \ell'_i) \in R$ and $(\ell_k, a?, \ell'_k) \in R$, and $\ell'_j = \ell_j$ for all $j \in \{1, \ldots, n\} \setminus \{i, k\}$.

– *broadcast transition*: there exists $1 \leq i \leq n$ and an action $a \in \Sigma_b$ such that $(\ell_i, a!!, \ell'_i) \in R$ and, for each $j \in \{1,\ldots,n\} \setminus \{i\}$, $(\ell_j, a??, \ell'_j) \in R$.

The weight of a BP state is simply its length (i.e. the number of processes involved). Weight respectfulness of a BP follows from the fact that $s \to s'$ implies that s and s' are of the same weight. BPs are 2-deflatable; here 2 arises from the fact that a rendez-vous transition is guarded by 2 processes.

Example 6. Lossy Channel Systems (LCS) [1] are 1-NSW. The state of a lossy LCS is a pair[2] (s,σ), where s an element in a finite state space, and $\sigma \in \Sigma^*$ is a string over the channel alphabet Σ. The usual wqo defined by $(s_1,\sigma_1) \preceq (s_2,\sigma_2)$ if $s_1 = s_2$ and σ_1 is a (not necessarily contiguous) substring of σ_2 is a dwqo[3]. The associated weight function is $w((s,\sigma)) = \mathit{length}(\sigma)$. A transition of a LCS can manipulate the channel string by appending a symbol to the tail, removing a symbol from the head, or nondeterministically deleting a symbol from anywhere in the string. The reader may verify that these systems are 1-deflatable and weight-respecting.

Example 7. Context-free grammars (CFG) are NSW. A CFG is a triple $G = (N,T,R)$ where N and T are disjoint, finite sets of *nonterminal symbols* and *terminal symbols*, respectively, and $R \subseteq N \times \Sigma^*$ is a finite set of *production rules*, where $\Sigma = N \cup T$. A CFG corresponds to the NSW $(\Sigma^*, \to, \preceq)$, where $x \to y$ iff there exist $x_1, x_2 \in \Sigma^*$ and $(\alpha,\beta) \in R$ such that $x = x_1 \alpha x_2$ and $y = x_1 \beta x_2$. The dwqo $\preceq \subseteq \Sigma^* \times \Sigma^*$ is such that $x \preceq y$ iff x can be obtained by deleting zero or more symbols from[3] y. The weight function is $w(x) = \mathit{length}(x)$. The system is δ-deflatable, where $\delta = \max(\{\mathit{length}(x) \mid \exists y \in N : (y,x) \in R\})$. Weight respectfulness comes from the fact that each production rule induces a fixed weight change.

4 Our Algorithm

This section develops our algorithm, which is shown in Fig. 3. The inputs are a δ-NSW $(S, \to, \preceq)$, a set of initial states I, and an $\preceq$-upward-closed set of target states U. For each $i = i_0, i_0+1, i_0+2, \ldots$, (where $i_0 = \mathsf{bw}(U)$) the algorithm computes the backward reachable set $\mathsf{br}(U,i)$, which is the set of states from which U is reachable along a path that never exceeds weight i. Formally, we have the following definition.

Definition 10 (br)**.** *Given a WSTS $(S, \to, \preceq)$, a set $Y \subseteq S$, and $i \in \mathbb{N}$ we let $\mathsf{br}(Y,i)$ denote the set of all $x \in S$ such that there exists a sequence $x_0 \to x_1 \to \cdots \to x_\ell$ such that $x_0 = x$, $x_\ell \in Y$, and for all $0 \leq j \leq \ell$ we have $w(x_j) \leq i$. We also define $\mathsf{br}(Y) = \bigcup_{i=0}^{\infty} \mathsf{br}(Y,i)$.*

Since $\mathsf{br}(U,i)$ is necessarily finite for all $i \geq 0$, this set can be computed using classical finite-state symbolic model checking [7] based on BDDs [6]. The algorithm terminates upon either of the following events:

[2] For simplicity we include only a single channel, the usual definition allows for an arbitrary (but finite) number of channels.
[3] That this relation is a wqo is known as *Higman's Lemma*[25].

– Convergence occurs. By *convergence*, we mean that we have reached an n such that $\uparrow \mathsf{br}(U,n) = \mathsf{br}(U)$. How this is done is articulated in our Theorem 1 below. The existence of such an n is guaranteed by Theorem 2.
– Intersection with the initial states is detected. Since we have left the requirements of the initial states undefined, we have necessarily left this check undefined in our algorithm. In general, for this check to be computable, we must be able to decide if $I \cap \mathsf{br}(U) = \emptyset$, given $\mathsf{br}(U,n)$, where n is as in the previous item.

We now present two theorems. Theorem 1 gives us a necessary and sufficient condition for convergence, while Theorem 2 guarantees that our algorithm will always terminate.

Theorem 1 (Convergence). *For a* δ*-NSW, an upward-closed set* U*, and* $n \geq \mathsf{bw}(U)$*,*

$$\mathsf{br}(U, n+\delta) \subseteq \uparrow \mathsf{br}(U,n) \tag{1}$$

if and only if

$$\mathsf{br}(U) = \uparrow \mathsf{br}(U,n) \tag{2}$$

Theorem 2. *For any DWSTS and upward-closed set* U*, there exists an* n *satisfying (2).*

In order to use Theorem 1 in our algorithm, we must have a means to decide (1). Our approach requires the use of a computable *lifting operator*, which intuitively "lifts" a set $\mathsf{br}(U,i)$ to a truncated version of its upward-closure. The truncation omits everything with weight strictly greater than some given $d \in \mathbb{N}$; hence finiteness is preserved.

Definition 11 (lifting operator). *Given a dwqo* $\preceq$ *over a set* X*, the associated* lifting operator *is the function* $\mathsf{Lift} : X \times \mathbb{N} \to 2^X$ *defined by*

$$\mathsf{Lift}(x,d) = \{y \mid x \preceq y \wedge w(y) \leq d\}$$

We extend Lift *to act on sets by decreeing* $\mathsf{Lift}(Y,d) = \bigcup_{y \in Y} \mathsf{Lift}(y,d)$.

The following theorem explains how the lifting operator is relevant to deciding containments along the lines of (1). For a finite set X, let $\mathsf{maxw}(X) = \max(\{w(x) \mid x \in X\})$

Theorem 3. *Let* $\preceq$ *be a dwqo over a set* X*, and let* X_{i-1} *and* X_i *be finite subsets of* X *such that* $\mathsf{maxw}(X_{i-1}) \leq i-1$ *and* $\mathsf{maxw}(X_i) \leq i$*. Then* $X_i \subseteq \uparrow X_{i-1}$ *if and only if* $X_i \subseteq \mathsf{Lift}(X_{i-1}, i)$.

4.1 An Optimization

In this section we propose an optimization to the algorithm of Fig. 3. Note that in Fig 3, the computation of $\mathsf{br}(U,i)$ involves an iterative fix-point computation, starting with set $U_{\leq i} = \{x \in U \mid w(x) \leq i\}$. In some sense, much of the work of this computation was already performed when computing $\mathsf{br}(U,i-1)$; since this is a subset of $\mathsf{br}(U,i)$, it is redundant to "rediscover" these states. Also note that

$$U_{\leq i} \;\subseteq\; \mathsf{Lift}(\mathsf{br}(U,i-1),i) \;\subseteq\; \mathsf{br}(U)$$

1 $i := \mathsf{bw}(U)$
2 $n := i$
3 $\Gamma_{i-1} := \emptyset$
4 `while` $(n \geq i - \delta)$ `do`
5   `compute` $\Gamma_i := \mathsf{br}(U, i)$
6   `if intersection`(Γ_i, I) `then`
7   `exit with verification failure`
8   `if` $(\Gamma_i \not\subseteq \mathsf{Lift}(\Gamma_{i-1}, i))$ `then`
9         $n := i$
10     $i := i + 1$
11 `exit with verification success`

Fig. 3. Our algorithm, which, given a δ-NSW $\mathcal{S}$, an upward-closed set U, and a set of initial states I, decides $\mathsf{Safe}(\mathcal{S}, I, U)$ using finite-state model checking. The variable i represents the maximum weight of the states computed in each iteration of the while-loop. i is initially the base weight of U and is incremented each iteration. The variable n tracks the last value of i for which "new" states were found in $\mathsf{br}(U,i)$ (see Def. 10), i.e. states x that weren't already "covered" by the existence of $y \in \mathsf{br}(U, i-1)$ such that $y \preceq x$. The condition of the while loop (line 4) will only fail when (1) holds, which by Theorem 1 indicates convergence. Each iteration of the loop involves computing $\mathsf{br}(U,i)$, which is done in a nested backward reachability loop (implicit in line 5). Line 6 tests to see if the initial states have been reached, and line 7 terminates if so. Line 8 determines if something "new" was found this iteration, if so n is updated to be i. If the condition of line 8 fails δ times consecutively, by Theorem 3 we have $\Gamma_{n+\delta} \subseteq \Gamma_{n+\delta-1} \subseteq \cdots \subseteq \Gamma_n$ and thus (1) holds and verification is successful. Theorem 2 guarantees that this will eventually happen.

It follows that we can eliminate the unnecessary overhead by starting the fix-point computation from $\mathsf{Lift}(\mathsf{br}(U, i-1), i)$, a set which we need to compute anyway for the containment check of line 8. Our optimization involves replacing lines 3 and 5 of Fig. 3 with the following, respectively:

$$\begin{array}{ll} 3' & \Gamma_{i-1} := \mathsf{basis}(U) \\ 5' & \quad \texttt{compute } \Gamma_i := \mathsf{br}(\mathsf{Lift}(\Gamma_{i-1}, i), i) \end{array} \tag{3}$$

This optimization has the potential to greatly reduce the number of iterations performed in the fix-point computations. As an extreme example, in an iteration of the outer loop for which $\mathsf{br}(U,i) \subseteq \mathsf{Lift}(\mathsf{br}(U, i-1), i)$ holds, the computation of Γ_i will involve only a single backward image computation.

Theorem 4. *The optimization (3) preserves correctness of our algorithm.*

5 Implementation Using Symbolic Model Checking

Given an NSW $(S, \rightarrow, \preceq)$, our algorithm manipulates finite subsets of S, so, in theory, we can directly apply standard finite-state symbolic model checking. In practice, we must provide a state encoding for the the finite-state subsets and a way to compute the tasks needed by our algorithm:

– the fix-point computation of line 5 or 5′

– the intersection check of line 6
– the lifting operation of line 8
– the containment check of line 8

This section sketches how we implemented the algorithm for various types of NSW. Our current implementation uses a very straightforward BDD-based approach, but our algorithm should be able to harness the many advances in symbolic model checking.

5.1 Parameterized Protocols

For petri nets and extensions such as broadcast protocols, there is a natural notion of *local state*, i.e., the (finite) state of each process in the broadcast protocol, or the place (out of a finite number) occupied by each token in a petri net. Our encoding follows [4] and uses *concrete global states*, i.e. tuples over local states. The weight is simply the number of processes, so we can represent subsets of S_i by sets over L^i, where L is the local state space. It is straightforward to construct a BDD for the transition relation in this framework, and hence the fix-point computation. The lifting operation is called *existential lifting* and can be computed using standard BDD operations. Finally, [4] shows that when I is a so-called *parametric set*, the intersection check of line 6 can also be performed using standard operations.

5.2 Lossy Channel Systems

As explained in Example 6, the infinite state space of a LCS is $S = C \times \Sigma^*$, where C is the finite state space of the control, and Σ is the channel alphabet. Let $S_{\leq i} = \{(c,\sigma) \in S \mid \mathit{length}(\sigma) \leq i\}$. Similarly to our encoding for parameterized protocols, we represent a subset of $S_{\leq i}$ by a collection of tuples of the form $(s, c_1, c_2, \ldots, c_i)$, where $s \in S$ and each $c_j \in \Sigma \cup \{\mathit{empty}\}$. Here $c_1, \ldots, c_i$ stores the contents of the channel, and the new symbol *empty* indicates that the channel "slot" does not contain a message. The lifting operator simply inserts an element of $\Sigma \cup \{\mathit{empty}\}$ nondeterministically into the channel, hence (possibly) increasing the number of non-*empty* slots by 1. The intersection and containment checks are also straightforward in this representation.

5.3 Comparison to Standard Approach

Comparing our approach to the classical approach (i.e., CSTs) provides intuition about when each approach is likely to perform better.

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 classical algorithm (cf. the `while` condition in Fig. 1). To combat this problem, Delzanno et al. develop a sophisticated heuristic solution in which certain 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 [6]. In fact, we can correctly replace the containment of line 8 of Fig. 3 with an equality test: $\Gamma_i \neq \mathsf{Lift}(\Gamma_{i-1}, i)$. This test can be done *in constant time* using a reasonable BDD library, such as CUDD [30].

Data Structure Size The main efficiency difference is likely to derive from the sizes 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). Similarly, although bounds on the size of CST have not been derived in the literature (to our knowledge), any such bound is at least exponential in the height of the structure. Here, we consider data structure height as a coarse measure of worst-case size.

The CST-based approach is applicable to both petri nets and broadcast protocols. Let L be the set of local states (i.e. petri net places). Then we call $|L|$ the *dimensionality* of a parameterized protocol. The height of the CSTs is fixed and equal to the dimensionality, while the height of the BDDs is at most $(n_f + \delta)\lceil \log_2 |L| \rceil$, where n_f is the final value of n in our algorithm. This suggests that our approach might be superior when $(n_f + \delta)\lceil \log_2 |L| \rceil$ is much less than the dimensionality, since under such circumstances the CSTs are more likely to blow-up.

For other NSW, such as LCS and CFG, we expect our ability to encode large control states spaces and/or large alphabets compactly using BDDs to provide our approach with an advantage for systems with these characteristics.

6 Conjunctive Guard Reduction

Though WSTS (and indeed NSW) encompass a broad and important class of infinite state systems, there are common system attributes that preclude well-structuredness. An example of such an attribute is the so-called *conjunctive guard* (CG). CG are used in parameterized systems of processes when a transition is to be enabled only if the local states of *all* processes satisfy some predicate. This contrasts with petri net or broadcast protocols, in which only a fixed, finite number of processes may guard a transition. Unfortunately, endowing petri nets or broadcast protocols with CG renders even safety property verification undecidable [15]. In this section we develop a sound reduction that reduces a BP with conjunctive guards with to a BP.

Emerson and Kahlon have proposed a sound *and complete* verification technique for a class of protocols with CG [16], however it is unclear if the approach will scale beyond systems with small local state. For example, their subsequent treatment of German's protocol requires a nontrivial amount of manual reasoning [14].

BPs were defined formally in Example 5; here we extend that definition to define *conjunctively guarded broadcast protocols* (CGBP). A CGBP is a tuple (L, Σ, R, g), where (L, Σ, R) is a BP, and $g : \Sigma_l \to 2^L$. For each action $a \in \Sigma_l$, $g(a)$ is called the *conjunctive guard*. The semantics are changed so that a local transition a may occur only if all other processes are in states that satisfy the conjunctive guard of a[4]. Formally, we conjoin the following condition to the local transition semantics presented in Example 5: $\ell_j \in g(a)$ for all $j \in \{1, \ldots, n\} \setminus \{i\}$.

[4] Our definition of CGBP allows only local actions to have conjunctive guards. The definition and the reduction can be generalized to support conjunctively guarded rendez-vous and broadcasts.

Local action a is said to be *conjunctively guarded* if $g(a) \neq L$. Hence a BP is a CGBP in which no action is conjunctively guarded, since in this case the additional requirement on each local transition is tautological.

Our reduction transforms a CGBP $\mathcal{B} = (L, \Sigma, R, g)$ into a BP $\mathcal{B}' = (L', \Sigma', R')$. Intuitively $\mathcal{B}'$ replaces conjunctively guarded local actions with broadcasts. These new broadcasts allow all processes to check if they *would have* permitted the transition in $\mathcal{B}$, i.e. if their local state satisfies the CG. Whenever a process detects a violation of a CG in this manner, it refuses to participate in any future actions by "resigning"; resigned processes are stuck in that state forever.

Formally, we define $\mathcal{B}'$ as follows. We denote by Σ_{cg} the set of conjunctively guarded actions in $\mathcal{B}$, i.e. $\Sigma_{cg} = \{a \mid a \in \Sigma_l \wedge g(a) \neq L\}$.

- $L' = L \cup \{\mathit{resigned}\}$, where *resigned* is a new local state not in L. A process will enter *resigned* if it notices (through a broadcast) that a conjunctive guard was violated.
- Σ' is defined by $\Sigma'_l = \Sigma_l \setminus \Sigma_{cg}$, $\Sigma'_r = \Sigma_r$, and $\Sigma'_b = \Sigma_b \cup \Sigma_{cg}$, i.e. all conjunctively guarded local actions are replaced with broadcasts.
- R' contains exactly the following transitions
  - for each $(\ell, \alpha, \ell') \in R$ such that $\alpha \in \{a!, a?, a!!, a?? \mid a \in \Sigma_r \cup \Sigma_b\} \cup \Sigma'_l$ we have $(\ell, \alpha, \ell') \in R'$. Hence all rendez-vous, broadcast, and non-conjunctively guarded local transitions are unchanged.
  - for each $(\ell, a, \ell') \in R$ such that $a \in \Sigma_{cg}$ we have $(\ell, a!!, \ell') \in R'$. Hence conjunctively guarded local actions become broadcasts.
  - for each $a \in \Sigma_{cg}$ we have $(\ell, a??, \ell') \in R'$, where $\ell' = \ell$ if $\ell \in g(a)$ otherwise $\ell' = \mathit{resigned}$. Hence, upon receiving a broadcast corresponding to a CG transition, a process is unaffected if it satisfied the conjunctive guard, otherwise it enters *resigned*.
  - for each $a \in \Sigma'_b$ we have $(\mathit{resigned}, a??, \mathit{resigned}) \in R'$. These transitions serve only to satisfy the restriction that broadcasts must always be received.

The following theorem states that $\mathcal{B}'$ is a sound reduction of $\mathcal{B}$, and can be proved by observing that any reachable state of $\mathcal{B}$ corresponds to a reachable state of $\mathcal{B}'$ in which no process is in local state *resigned*.

Theorem 5. *For CGBP $\mathcal{B}$, $\mathsf{Safe}(\mathcal{B}', I, U)$ implies $\mathsf{Safe}(\mathcal{B}, I, U)$.*

This reduction is "complete" if a certain decidable side condition holds. For each conjunctively guarded local transition a of $\mathcal{B}$, let $\widehat{g}(a) \subseteq L$ be the set of local states ℓ such that there exists a sequence of zero or more non-conjunctively guarded local transitions taking ℓ to a state $\ell' \in g(a)$; note that $g(a) \subseteq \widehat{g}(a)$. We construct a broadcast protocol $\mathcal{B}''$ that modifies $\mathcal{B}'$ as follows. A new local state *error* is added, and when a process in local state ℓ receives broadcast a (corresponding to a conjunctively guarded local action in $\mathcal{B}$), its next state is ℓ', defined by

$$\ell' = \begin{cases} \mathit{error} & \text{if } \ell \notin \widehat{g}(a) \\ \mathit{resigned} & \text{if } \ell \in \widehat{g}(a) \wedge \ell \notin g(a) \\ \ell & \text{if } \ell \in g(a) \end{cases}$$

Petri net	Our runtime	CST runtime	Max BDD height	CST height (dimensionality)
Multipool	3010	2.09	50	18
CSM	95	0.06	36	14
Mesh(2×2)	$>$1300	1.30	$>$40	32

Table 1. Experiments involving selected petri nets from [13]. For Mesh(2×2), our tool spaced out.

Let *Error* be the set of all broadcast protocol states such that at least one process is in local state *error*. We note that *Error* is upward-closed. The following theorem states that if *Error* is unreachable in $\mathcal{B}''$, then the conjunctive guard reduction is both sound and complete.

Theorem 6. *For CGBP $\mathcal{B}$, suppose* $\mathsf{Safe}(\mathcal{B}'', I, Error)$. *Then* $\mathsf{Safe}(\mathcal{B}, I, U)$ *if and only if* $\mathsf{Safe}(\mathcal{B}', I, U)$.

7 Experiments

In this section, we present experimental results for several petri nets, a MESI cache protocol, a lossy channel system, and a more elaborate caching protocol. All experiments were run on a machine with an Intel Pentium 4 at 2.6GHz and 4GB total memory. The implementation of the classical approach we compare against is based on an extension of CSTs called *interval sharing trees* [21].

7.1 Petri Nets

In [13], Delzanno et al. run their CST-based implementation of the classical approach against several petri nets. These nets have small dimensionality, so, as discussed in Sect. 5.3, we do not expect our approach to perform well. Indeed, Table 1 shows that the CST-based implementation outperforms our approach by several orders of magnitude. Recall from Sect. 5.3 that we anticipated that our approach would have an advantage when the height of our BDDs is dwarfed by the height of the CSTs, which is not the case here. In fact, for all three petri nets, the CSTs enjoy a shorter height than the BDDs.

7.2 MESI Protocol

MESI is a common variety of cache coherence protocol. In a MESI protocol, each client has a cache block in one of four states: *modified* (M), *exclusive* (E), *shared* (S), or *invalid* (I). Although many MESI protocols are conceivable, here we use the standard version used by computer architects (e.g., [10]), which has a conjunctive guard, so we use our reduction from Sect. 6.

We wanted a “knob” that would give us some control over the size of the local state space. Since cache protocols are typically used to orchestrate the sharing of multiple blocks, we instantiated the MESI protocol over m blocks[5], for $m \in \{1,2,3,4\}$.

[5] In this case, since the “sub-protocols” controlling each block are independent, correctness for $m = 1$ entails correctness for all $m \geq 1$. In practice, however, such a simplification is often not

# of blocks	Our runtime	CST runtime	Hytech runtime	Max BDD height	dimension
1	0.0	0.0	0.0	9	5
2	0.1	0.2	380.0	18	25
3	0.7	131.9	>7989.0	27	125
4	4.6			36	625

Table 2. Results for the MESI protocol with conjunctive guards over multiple blocks. Run times are in seconds. The column *dimension* indicates the height of the CST data structures in the CST approach, and also the width of the real vectors processed by Hytech in Delzanno's polyhedral approach.

Results are given in Table 2. Our CG reduction allowed the verification to succeed, so there was no need to verify the side condition. We compare our result against both the CST-based classical approach and another variant of the classical approach based on the polyhedral model checker hytech [11, 24]. The results clearly indicate the superior scalability of our approach as the local state grows. The Hytech-based approach aborts even for $m = 3$, reporting "Out of memory". The parser of the CST tool cannot handle the size of the description of MESI with 4 blocks, which is 5.9 MBs. This large size arises because the broadcast matrices used by the classical approach grow quadratically in the dimension of the problem. This contrasts with the mere 30 KB of SMV that constitutes our tool's input.

7.3 Alternating Bit Protocol

As an experiment with lossy channel systems, we selected the alternating bit protocol (ABP). ABP involves two unbounded, lossy channels, one that carries data and sequence bits from the sender to the receiver, and another that carries acknowledgements from the receiver to the sender. Our ABP model is based on the presentation in [27], and we verify that whenever the sender receives an acknowledgement, the previously sent data (a copy of which is saved by the sender) matches the receiver's data buffer. As a complexity knob, we vary the number *data_count*, which specifies the number of different data values that may be sent. Results are shown in Table 3.

7.4 German's Protocol

German's protocol [23] is a challenge problem for parameterized verification that has been previously tackled in several papers [28, 26, 8]. As in [8], we include a one bit data path. The original description [23] is a Murϕ model, and is almost a CGBP: the Murϕ description involves a variable of type *client ID*. We've encoded this variable by simply giving each process an extra bit, which is true iff the original variable would point to the process. This system is a CGBP, and our CG reduction is applied.

As mentioned in Sect. 7.2, even *describing* a BP in a format suitable for the classical approach is problematic when the dimension is large. Due to its various channels and

possible, because real protocols can exhibit nontrivial interactions between different blocks. This experiment measures how our approach handles the explosive growth in local state resulting from analyzing multiple blocks.

data_count	our runtime
3	0.2
7	0.6
15	1.9
31	5.7
63	23.1
127	90.0
255	340.7

data_count	TReX runtime
1	0.01
2	0.02
3	0.05
4	0.08
5	0.15

Table 3. Alternating Bit Protocol Results. As a rough comparison, we ran preliminary experiments with TReX, a state-of-the-art verification tool for lossy channel systems [3]. We do not intend a direct comparison, because of our inexperience with TReX (e.g., an internal data structure overflowed when we tried $data\_count = 6$). However, the pattern is clear: TReX is faster when the alphabet (analogous to dimensionality in broadcast protocols) is small, but the run time is growing exponentially; our tool scales more gracefully.

Property (all passed)	Runtime (sec)
Encoding of curPtr	3
Conjunctive guard reduction	214
Data coherence	63

Table 4. Results for German's Protocol. To convert the protocol to a CGBP, we needed to re-encode the curPtr variable. Although this encoding was straightforward, we verified the encoding as a sanity check. The main verification task was "data coherence", which verified that the value of each read is the most recently written data value. Since the CG reduction is sound and the verification succeeded, we actually did not need to run the "conjunctive guard reduction" verification task. We have provided the run time simply to illustrate that the side condition is verifiable in practice.

the presence of data variables, our model of German's protocol has a dimensionality of 6144. For this reason, we were unable even to run the CST-tool against this example. The results for our tool are given in Table 4.

8 Conclusions and Future Work

We have introduced the concept of NSW and provided a new algorithm for verification of these transition systems. The algorithm harnesses the power of finite-state symbolic model checking. We have also introduced a new reduction for systems with unbounded conjunctive guards. As predicted by our theory, experimental results show that our new verification algorithm greatly outperforms existing approaches for systems that involve large local state spaces, control state spaces, channel alphabets, etc. We attribute this to the ability of BDDs to encode such sets succinctly.

Our current implementation is fairly naive. We believe more sophisticated symbolic model checking techniques can produce still better results. Other avenues for future work include computing bounds on BDD sizes, finding additional NSW applications, and finding ways to apply our method semi-algorithmically to systems that are not NSW.

Very recently, Geeraerts et al. [22] have proposed a compelling approach to verification of WSTS based on forward reachability. This is the first sound and complete algorithm that performs forward analysis of WSTS. Similar to our approach, theirs is based on a framework in which a sequence of finite-state subsystems of increasing size are examined until either a counterexample is found, or a certain convergence condition is reached. Convergence occurs when an abstraction, which becomes more and more precise, is tight enough to verify non-reachability. Obvious directions for future work include comparing our approach with that of Geeraerts et al., and investigating the possibility of employing BDDs as we do for backward reachability in their forward framework.

References

1. P. Abdulla and B. Jonsson. Verifying programs with unreliable channels. In *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 160–170, 1993.
2. 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.
3. A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: A tool for reachability analysis of complex systems. In *Proc. 13th Intern. Conf. on Computer Aided Verification (CAV'01)*, 2001.
4. J. Bingham. A new approach to upward closed set backward reachability analysis. In *6th International Workshop on Verification of Infinite-State Systems (INFINITY)*, 2004.
5. J. Bingham and A. J. Hu. Empirically efficient verification for a class of infinite-state systems. In *11th International Conference Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, 2005.
6. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
7. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2), 1992.
8. C.T. Chou, P. K. Mannava, and S. Park. A simple method for parameterized verification of cache coherence protocols. In *Formal Methods in Computer-Aided Design*, 2004.
9. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Workshop on Logics of Programs*, pages 52–71, May 1981. Published 1982 as Lecture Notes in Computer Science Number 131.
10. D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1998.
11. G. Delzanno. Automatic verification of parameterized cache coherence protocols. In *Proceedings of the 12th International Conference on Computer Aided Verification*, July 2000.
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. 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.

15. 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.
16. E. A. Emerson and V. Kahlon. Rapid parameterized model checking of snoopy cache protocols. In *9th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, pages 144–159, April 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. G. Geeraerts, J.-F. Raskin, and L. Van Begin. Expand, Enlarge and Check: new algorithms for the coverability problem of WSTS. In *Proceedings of FSTTCS'04, 24th International Conference on Foundations of Software Technology and Theoretical Computer Science, Chennai, India*, pages 287–298, 2004.
23. S. German. Personal correspondence. 2003.
24. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1:110–122, 1997.
25. G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society (3)*, 2(7):326–336, 1952.
26. S. K. Lahiri and R. E. Bryant. Constructing quantified invariants via predicate abstraction. In *Proc. of 5th Intl. Conference on Verification, Model Checking and Abstract Interpretation (VMCAI)*, pages 267–281, 2004. LNCS 2937.
27. L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addision-Wesley, 2002.
28. A. Pnueli, S. Ruah, and L. Zuck. Automatic deductive verification with invisible invariants. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 82–97, 2001.
29. J.-P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *5th International Symposium on Programming*, pages 337–351. Springer, 1981. Lecture Notes in Computer Science Number 137.
30. F. Somenzi. Colorado university decision diagram package (CUDD) webpage. http://vlsi.colorado.edu/∼fabio/CUDD/cuddIntro.html.

A Proofs

The proof of Theorem 1 requires the following lemma.

Lemma 2. *For a weight respecting DWSTS and an upward-closed set U, $z \in \mathsf{br}(U,n)$ and $z \preceq y'$ imply $y' \in \mathsf{br}(U, n + w(y') - w(z))$.*

Proof: Let $d = w(y') - w(z)$. $z \in \mathsf{br}(U,n)$ implies there exists a sequence $z_0 \to z_1 \to \cdots \to z_\ell$ where $z_0 = z$ and $z_\ell \in U$, and $w(z_j) \leq n$ for all $0 \leq j \leq \ell$. We show by induction on ℓ that $y' \in \mathsf{br}(U, n+d)$. If $\ell = 0$, then $z \in U$ and thus $y' \in U$ and we are done since

$w(y') \leq n+d$. For the inductive step, we note that $z \to z_1$ implies that there exists y_1 such that $w(y_1) = w(z_1)+d$, $y' \to y_1$, and $z_1 \preceq y_1$ since the system is weight respecting. Note that $z_1 \in \mathsf{br}(U,n)$, thus, by our inductive hypothesis, $y_1 \in \mathsf{br}(U, n+w(y_1)-w(z_1)) = \mathsf{br}(U,n+d)$, which implies $y' \in \mathsf{br}(U,n+d)$.

$\square$

Proof of Theorem 1: ($\Leftarrow$) Trivial. ($\Rightarrow$) $\mathsf{br}(U) \supseteq \uparrow\mathsf{br}(U,n)$ holds trivially. To prove the converse containment, suppose that (1) holds, but there exists $i \geq 1$ such that there exists $x \in \mathsf{br}(U,i)$ such that $x \notin \uparrow\mathsf{br}(U,n)$. Since $\mathsf{br}(U,j) \subseteq \mathsf{br}(U,k)$ whenever $j \leq k$, $i \leq n+\delta$ implies $x \in \mathsf{br}(U,n+\delta) \subseteq \uparrow\mathsf{br}(U,n)$, thus we need only consider $i > n+\delta$.

Let $x = x_0 \to x_1 \to \cdots \to x_\ell \in U$ be a path from x to U. Since $U \subseteq \uparrow\mathsf{br}(U,n)$, there must exist $k \in \{1,\ldots,\ell\}$ such that $x_k \in \uparrow\mathsf{br}(U,n)$ and $x_{k-1} \notin \uparrow\mathsf{br}(U,n)$. Then there exists $z \in \mathsf{br}(U,n)$ such that $z \preceq x_k$. Now because the system is δ-deflatable, there exists y and y' satisfying the five conditions of Def. 8, i.e. $y \preceq x_{k-1}$, $y \to y'$, $z \preceq y'$, $w(y) \leq w(z)+\delta$, and $w(y') \leq w(z)+\delta$. From Lemma 2 we have that $y' \in \mathsf{br}(U,n+\delta)$ which implies $y \in \mathsf{br}(U,n+\delta)$ since $w(y) \leq n+\delta$. Since $y \preceq x_{k-1}$, this implies $x_{k-1} \in \uparrow\mathsf{br}(U,n)$, which is a contradiction.

$\square$

Proof of Theorem 2: For any $x \in \mathsf{br}(U)$, let $g(x)$ denote the minimum j such that $x \in \mathsf{br}(U,j)$. Since $\mathsf{br}(U)$ is upward-closed [20], there exists a finite set $B = \mathsf{basis}(\mathsf{br}(U))$. Now take $n = \max(\{g(x) \mid x \in B\})$. To see that this n satisfies (2), note that for any $i \geq n$, if there exists $x \in \mathsf{br}(U,i)$ such that $x \notin \uparrow\mathsf{br}(U,n)$, then this contradicts B being a basis for $\mathsf{br}(U)$, since $B \subseteq \mathsf{br}(U,n)$.

$\square$

Proof of Theorem 3: ($\Leftarrow$) Trivial, since $\mathsf{Lift}(X_i, i+1) \subseteq \uparrow X_i$. ($\Rightarrow$) Suppose $X_i \subseteq \uparrow X_{i-1}$, and let $x \in X_i$. Then there exists $y \in X_{i-1}$ such that $y \preceq x$. Since $w(x) \leq i$, this implies $x \in \mathsf{Lift}(X_{i-1}, i)$.

$\square$

Proof of Theorem 4: For a set X denote $X_{\leq i} = \{x \in X \mid w(x) \leq i\}$. Let $b = \mathsf{bw}(U)$, and let $\Gamma_{b-1}, \Gamma_b, \Gamma_{b+1}, \ldots$ and $\Gamma'_{b-1}, \Gamma'_b, \Gamma'_{b+1}, \ldots$ be the sequences of values assigned to the variables Γ_i by the algorithm of Fig. 3 and the optimized version, respectively. We first show that for all $i \geq b$ we have

$$\Gamma_i \subseteq \Gamma'_i \subseteq \mathsf{br}(U) \tag{4}$$

(4) clearly holds when $i = b$, since $\Gamma_b = \Gamma'_b$. Assume that (4) holds for $i \geq b$. $\Gamma'_i \subseteq \mathsf{br}(U)$ implies that $\mathsf{Lift}(\Gamma'_i, i+1) \subseteq \mathsf{br}(U)$ (since $\mathsf{br}(U)$ is upward-closed [20]), and hence $\Gamma'_{i+1} = \mathsf{br}(\mathsf{Lift}(\Gamma'_i, i+1), i+1) \subseteq \mathsf{br}(U)$. Thus the second containment of (4) holds for $i+1$. Also, since $U_{\leq i} \subseteq \Gamma_i$ and $i \geq \mathsf{bw}(U)$, we have

$$\begin{aligned}
\Gamma_{i+1} &= \mathsf{br}(U_{\leq i+1}, i+1) \\
&= \mathsf{br}(\mathsf{Lift}(U_{\leq i}, i+1), i+1) \\
&\subseteq \mathsf{br}(\mathsf{Lift}(\Gamma_i, i+1), i+1) \\
&\subseteq \mathsf{br}(\mathsf{Lift}(\Gamma'_i, i+1), i+1) \\
&= \Gamma'_{i+1}
\end{aligned}$$

Therefore the first containment of (4) holds for $i+1$.

Now let N be the final value of the variable n in the unoptimized algorithm. Then, by Theorem 1, we have $\uparrow\Gamma_N = \mathsf{br}(U)$, which, along with (4) implies $\uparrow\Gamma'_N = \mathsf{br}(U)$. Since $\Gamma'_k \subseteq \Gamma'_j$ when $k \leq j$, it follows that for each $i \in \{N+1,\ldots,N+\delta\}$ we have

$$\Gamma'_i = \mathsf{Lift}(\Gamma'_{i-1}, i) = \mathsf{br}(U)_{\leq i} \tag{5}$$

Therefore the optimized version also terminates with the final value of n being N, and computes Γ'_N such that $\uparrow\Gamma'_N = \mathsf{br}(U)$.

□

The proofs of Theorems 5 and 6 use the following terminology. We call a sequence of states $x_0 \to x_1 \to \cdots \to x_\ell$ a *path* of $\mathcal{B}'$ (resp. of $\mathcal{B}$) if $\to$ is the transition relation of $\mathcal{B}'$ (resp. $\mathcal{B}$).

Lemma 3. σ *is a path of* $\mathcal{B}$ *if and only if* σ *is a path of* $\mathcal{B}'$ *in which no process is ever in the local state resigned.*

Proof: Follows from the simple observations that any transition of $\mathcal{B}$ is a transition of $\mathcal{B}'$, and, conversely, any transition $\mathcal{B}'$ in which no process starts or finishes in *resigned* is a transition of $\mathcal{B}$.

□

Proof of Theorem 5: By Lemma 3, the set of states reachable from I in $\mathcal{B}'$ over-approximates the set of states reachable in $\mathcal{B}$.

□

Proof of Theorem 6: ($\Leftarrow$) Follows by Theorem 5.

($\Rightarrow$) We assume $\neg\mathsf{Safe}(\mathcal{B}', I, U)$ and $\mathsf{Safe}(\mathcal{B}'', I, \mathit{Error})$, and show that $\neg\mathsf{Safe}(\mathcal{B}, I, U)$ follows. Call a path *unsafe* if it starts in I and ends in U. For the remainder of the proof, $\to$ will denote the transition relation of $\mathcal{B}'$. For a state x of a (CG) broadcast protocol (i.e. a finite word over the local state space) and a process p involved in this state (i.e. an index $p \in \{1,\ldots,n\}$, where n is the length of x), we denote by $x(p)$ the local state of p in x (i.e. $x(p)$ is the pth symbol in x).

For a path σ of $\mathcal{B}'$ and a process p, let $\mathsf{rsl}(\sigma, p)$ be the length of the suffix of σ in which p is in state *resigned*. Note that $\mathsf{rsl}(\sigma, p) = 0$ precisely in the case that p never resigns. We show how, given an unsafe path σ of $\mathcal{B}'$, we can construct another unsafe path τ of $\mathcal{B}'$ such that

Property 1 for all $p \in \{1,\ldots,n\}$ we have $\mathsf{rsl}(\tau, p) \leq \mathsf{rsl}(\sigma, p)$, and
Property 2 there exists $q \in \{1,\ldots,n\}$ such that $\mathsf{rsl}(\tau, q) < \mathsf{rsl}(\sigma, q)$,

where n is the number of processes involved in σ. By iterating this construction, one can transform any unsafe path of $\mathcal{B}'$ into an unsafe path of $\mathcal{B}'$ in which no process resigns, which, by Lemma 3, is an unsafe path of $\mathcal{B}$.

We now describe the construction. Suppose we have an unsafe path of $\mathcal{B}'$

$$x_0 \to x_1 \to \cdots \to x_\ell \tag{6}$$

in which some process resigns. Let q be an earliest resigning process, i.e. q is such that there exists $i \geq 1$ where $x_{i-1}(q) \neq \mathit{resigned} = x_i(q)$, and further no process is resigned in x_{i-1}. Then there exists $a \in \Sigma_{cg}$ (i.e. a is a CG local action of $\mathcal{B}$ and hence a is a broadcast action of $\mathcal{B}'$) such that $x_{i-1} \rightarrow x_i$ is a broadcast transition on the action a. Since $\mathsf{Safe}(\mathcal{B}'', I, \mathit{Error})$, we have that $x_{i-1}(q) \in \widehat{g}(a)$. Thus there exists a path of $\mathcal{B}'$

$$x_{i-1}^0 \rightarrow x_{i-1}^1 \rightarrow \cdots \rightarrow x_{i-1}^m \tag{7}$$

such that

- $x_{i-1}^0 = x_{i-1}$,
- for each $j \in \{1, \ldots, m\}$, $x_{i-1}^{j-1} \rightarrow x_{i-1}^j$ involves a local transition of process q, and
- $x_{i-1}^m(q) \in g(a)$

Hence x_{i-1} and x_{i-1}^m differ only in the local state of q, and therefore the broadcast of a may occur from x_{i-1}^m, yielding a state y_i which differs from x_i only in that $y_i(q) = x_{i-1}^m(q) \neq \mathit{resigned} = x_i(q)$. Note that no process is resigned in any state of (7). Starting from y_i, there exists a path of $\mathcal{B}'$

$$y_i \rightarrow y_{i+1} \rightarrow \cdots \rightarrow y_\ell \tag{8}$$

such that for each $j \in \{i, \ldots, \ell\}$, y_j differs from x_j in (at most) the local state of process q. For each $j \in \{i+1, \ldots, \ell\}$, the transition $y_{j-1} \rightarrow y_j$ involves the same local, rendez-vous, or broadcast action as $x_{j-1} \rightarrow x_j$, which is always possible because $x_{j-1}(q) = \mathit{resigned}$ and thus q never performs an output in these transitions. Intuitively, in (8) q is passive in the sense that it only receives inputs from rendez-vous and broadcasts, possibly resigning at some point, but is definitely not resigned in y_i. Also, since $x_\ell \in U$ and no processes are resigned in any state of $\mathsf{basis}(U)$, we have that $y_\ell \in U$.

We may construct the desired path τ by concatenating the first i states of (6), all of (7), and all of (8), as follows.

$$x_0 \rightarrow x_1 \rightarrow \cdots \rightarrow x_{i-1} \rightarrow x_{i-1}^1 \rightarrow \cdots \rightarrow x_{i-1}^m \rightarrow y_i \rightarrow \cdots \rightarrow y_\ell \tag{9}$$

It is easy to see that (9) is unsafe and satisfies properties 1 and 2 above, hence this completes the proof.

□