Better is Better than Well: On Efficient Verification of Infinite-State Systems

(Extended Abstract)

Parosh Aziz Abdulla Aletta Nylén

Department of Computer Systems, Uppsala University
P.O.Box 325, 751 05 Uppsala, Sweden
phone: +46 18 471 31 63, fax: +46 18 55 02 25,
email: {parosh, aletta}@docs.uu.se

Abstract

Many existing algorithms for model checking of infinite-state systems operate on *constraints* which are used to represent (potentially infinite) sets of states. A general powerful technique which can be employed for proving termination of these algorithms is that of *well quasi-orderings*. Several methodologies have been proposed for derivation of new well quasi-ordered constraint systems. However, many of these constraint systems suffer from a "constraint explosion problem", as the number of the generated constraints grows exponentially with the size of the problem. In this paper, we demonstrate that a refinement of the theory of well quasi-orderings, called the theory of *better quasi-orderings* is more appropriate for symbolic model checking, since it allows inventing constraint systems which are both well quasi-ordered and compact. We apply our methodology to derive new constraint systems for verification of systems with unboundedly many clocks, broadcast protocols, lossy channel systems, and integral relational automata. We show that the new constraint systems are exponentially more succinct than existing ones, and that their well quasi-ordering cannot be shown by previous methods in the literature.

1 Introduction

A considerable amount of research has been devoted to extend the applicability of model checking from the context of finite to that of infinite-state systems. Standard techniques such as reachability analysis and tableau procedures can be adapted, by using *constraints* to represent (potentially infinite) sets of states. These algorithms are based on two operations, namely that of computing predecessors or successors of sets of states (represented by constraints), and that of checking for termination (formulated as entailment between constraints). Since the number of constraints is not *a priori* bounded, a key problem when applying the algorithms, is to guarantee termination. A general powerful tool which can be applied for proving termination is to show that the set of constraints is *well quasi-ordered* under entailment. In [AČJYK96, AJ98a, FS98] a methodology is defined for inventing well quasi-ordered constraint systems. The key idea is to start from a set of "basic" constraints, and repeatedly derive new ones, using the fact that well quasi-orderings are closed under certain operations on constraints such as building finite trees, strings, vectors, bags, sets, etc. The methodology has been applied both to unify earlier existing results for Petri nets, timed automata, lossy channel systems, completely specified protocols, relational automata, etc, and to design verification algorithms for new classes of systems such as timed networks [AJ98b] and broadcast protocols [EFM99, DEP99]. However many of the constraint systems constructed according to this method, suffer from a "constraint explosion" problem, as the number of constraints generated when computing predecessors (successors) grows exponentially with the number of components.

In this work, we demonstrate that a refinement of the theory of well quasi-orderings, called the theory of *better quasi-orderings*, is more appropriate for symbolic model checking, as it allows for constraint systems which are more compact and hence less prone to constraint explosion. More precisely, better quasi-orderings offer two advantages: (i) better quasi-ordering implies well quasi-ordering; hence all the verification algorithms originally designed for well quasi-ordered constraint systems are also applicable to better quasi-ordered ones; and (ii) better quasi-orderings are more "robust" than well quasi-orderings. For instance, in addition to the above mentioned operations, better quasi-ordered constraint systems (in contrast to well quasi-ordered ones) are closed under disjunction: if a set of constraints is better quasi-ordered under entailment, then the set of finite disjunctions of these constraints is also better quasi-ordered under entailment. In this paper, we provide several examples which show that using disjunction often leads to very compact constraint systems.

First, we propose a new constraint system, which we call *existential zones*, for verification of systems with unboundedly many clocks such as timed networks [AJ98b] and timed Petri nets (Section 4). Such systems cannot be modelled as real-time automata, since the latter operate on a finite set of clocks. An existential zone specifies a minimal required behaviour, typically of the form $\exists x_1 x_2 : \ 3 \leq x_2 - x_1 \leq 8$, characterizing the set of configurations in which there exist *at least* two clocks whose values differ by at least 3 and at most 8. Existential zones are related to *existential regions*, which are used in [AJ98b] for verification of timed networks. We can show better quasi-ordering of existential regions, since they are constructed by repeatedly building strings, bags, and sets. Each existential zone is equivalent to the disjunction of a finite number of existential regions. Since better quasi-orderings are closed under disjunction, it follows that existential zones are better quasi-ordered (and hence well quasi-ordered). We observe that well quasi-ordering of existential zones cannot be shown using the approach of [AČJYK96, AJ98a, FS98], since well quasi-orderings in general are not closed under disjunction. In fact, an existential zone is often equivalent to the disjunction of an exponential number of regions, thus offering a much more compact representation (in the same manner that *zones* are more efficient than *regions* in verification tools for real-time automata [LPY97, Yov97]). We can extend the results further and consider "existential variants" of CDDs [LPWY99] and DDDs [MLAH99], constraint systems which are even more compact than zones. We have implemented a prototype based on existential DDDs, and carried out a verification of a parametrized version of Fischer's protocol. While the set of constraints explodes when using existential regions, our tool performs the verification in a few seconds.

We also consider broadcast protocols, which consist of an arbitrary number of finite-state processes, communicating through rendezvous or through broadcast. In [EFM99] safety properties are checked, using constraints which characterize upward closed sets of vectors of natural numbers. In [DEP99] several new constraint systems are proposed, represented by different forms of linear inequalities over natural numbers. Since the new constraint systems cannot be constructed from upward closed sets using the previously mentioned constraint operations, each class requires an explicit termination proof for the underlying reachability algorithm. Applying our methodology we are able to prove well quasi-ordering of the different constraint systems in a uniform manner. More precisely, the upward closed sets are characterized by constraints which are vectors of natural numbers and hence are better quasi-ordered. Using the fact that each inequality is a finite union (disjunction) of upward closed sets, we conclude that they are also better quasi-ordered (and hence well quasi-ordered).

Finally, we provide new better quasi-ordered constraint systems for verification of lossy channel systems [AJ96] and integral relational automata [Čer94]. The new constraint systems are exponentially more succinct than existing ones.

To our knowledge this work is the first application of the theory of better quasi-orderings in the context of symbolic model checking.

Outline In the next section we recall how constraints and well quasi-orderings can be applied in symbolic model checking. In Section 3 we introduce the properties of better quasi-orderings which make them attractive for the design of constraint systems. In Section 4 we introduce existential zones and show how they can be used for verification of (unbounded) timed Petri nets. In sections 5 we show how to derive the constraints in [DEP99] within our framework. In Section 6 and Section 7 we provide new better quasi-ordered constraint systems for lossy channel systems and integral relational automata.

2 Constraints and WQOs

In this section, we introduce the notions of *constraints* and *well quasi-orderings*, and describe how to use them for performing symbolic model checking. We assume a transition system $(\Gamma, \longrightarrow)$, where Γ is a potentially infinite set of *configurations*, and $\longrightarrow$ is a transition relation on Γ whose reflexive transitive closure is denoted by $\xrightarrow{*}$. We use *constraints* ϕ for representing sets $[\![\phi]\!]$ of configurations. We define an *entailment relation* $\preceq$ on constraints, where $\phi_1 \preceq \phi_2$ iff $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$, and let $\equiv$ be the equivalence relation induced by $\preceq$. We sometimes write disjunctions $\phi_1 \vee \cdots \vee \phi_n$ of constraints as $\vee\{\phi_1, \ldots, \phi_n\}$. For sets Φ_1, Φ_2 of constraints, we let $\Phi_1 \sqsubseteq \Phi_2$ denote that for each $\phi_2 \in \Phi_2$ there is a $\phi_1 \in \Phi_1$ with $\phi_1 \preceq \phi_2$. Notice that $\Phi_1 \sqsubseteq \Phi_2$ implies $\vee\Phi_1 \preceq \vee\Phi_2$.

In the sequel, we concentrate on the *reachability problem*: given a configuration γ_{init} and a constraint ϕ_F, is there $\gamma_F \in [\![\phi_F]\!]$ such that $\gamma_{init} \xrightarrow{*} \gamma_F$? We perform a backward reachability analysis, generating a sequence $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots$, of finite sets of constraints where $\Phi_0 = \{\phi_F\}$ and $\Phi_{j+1} = \Phi_j \cup Pre(\Phi_j)$. Here $Pre(\Phi) = \cup_{\phi \in \Phi} Pre(\phi)$, where $Pre(\phi)$ is a finite set of constraints, such that $[\![\vee Pre(\phi)]\!] = \{\gamma' |\ \gamma' \longrightarrow \gamma\}$. For all the constraint systems we consider in this paper, this set exists and is computable. Since $\Phi_0 \sqsupseteq \Phi_1 \sqsupseteq \Phi_2 \sqsupseteq \cdots$, the algorithm terminates when we reach a point j where $\Phi_j \sqsubseteq \Phi_{j+1}$ (implying $\vee\Phi_{j+1} \equiv \vee\Phi_j$). Then, Φ_j characterizes the set of all predecessors of ϕ_F (sometimes written as $Pre^*(\phi_F)$).

This means that the answer to the reachability question is equivalent to whether $\gamma_{init} \in [\![\vee\Phi_j]\!]$. We observe that, in order to be able to implement the algorithm for a given class of systems, the constraint system should allow (i) computing $Pre(\phi)$, (ii) checking entailment between constraints, and satisfiability of a constraint by a configuration.

To show termination we rely on the theory of *well quasi-orderings (wqo)*. A constraint system is said to be *well quasi-ordered* if for each infinite sequence $\phi_0, \phi_1, \phi_2, \ldots$ of constraints, there are $i < j$ with $\phi_i \preceq \phi_j$. The following lemma (from [AČJYK96]) characterizes the class of constraint systems for which termination is guaranteed.

Lemma 2.1 A constraint system is well quasi-ordered iff for each infinite ($\subseteq$-increasing) sequence $\phi_0 \subseteq \phi_1 \subseteq \phi_2 \subseteq \cdots$ of constraints, there is a j such that $\phi_j \sqsubseteq \phi_{j+1}$.

3 BQOs

As evident from Lemma 2.1, well quasi-ordering is crucial for termination of the symbolic algorithm presented in Section 2. Furthermore, three other properties of a given constraint system decide how efficient the algorithm may run in practice. These properties are the size of the set $Pre(\phi)$, the cost of checking entailment and membership, and the number of iterations needed before achieving termination. In [AČJYK96, AJ98a, FS98], a methodology is defined for inventing well quasi-ordered constraint systems, based on the fact that all finite domains are well quasi-ordered under equality, and that well quasi-orderings are closed under a basic set of operations including building finite trees, strings, vectors, bags, sets, etc. This means that we can start from a set of constraints over finite domains, and then repeatedly generate new constraints by building more compound data structures. A typical application of this approach is a constraint system, called *existential regions*, introduced in [AJ98b] for verification of systems with unboundedly many clocks. However, constraints developed according to the above methodology suffer from "constraint explosion" caused by the size of the set $Pre(\phi)$. For instance, using existential regions, the set of generated constraints explodes even for very small examples. Often, the constraint explosion can be much reduced, by considering new constraint systems, which are disjunctions of the ones derived using the above mentioned set of operations. In Section 4 we present *existential zones* each of which corresponds to the disjunction of a (sometimes exponential) number of existential regions. Thus, existential zones offer a much more compact representation, allowing us to verify a parametrized version of Fischer's protocol in a few seconds. As we show in this section, well quasi-ordered constraint systems are not closed under disjunction, and hence we cannot prove well quasi-ordering of existential zones within the framework of [AČJYK96, AJ98a, FS98]. Here, we propose an alternative approach namely that of *better quasi-orderings (bqo)*.

In the rest of this section we write $(A, \preceq)$ to denote a quasi-ordering $\preceq$ on a set A. We will not use the (rather technical) definition of bqos in the remaining sections, so we move the definition to the appendix. Instead we state (Theorem 3.1) some properties of bqos which make them attractive for symbolic model checking. Let A^* denote the set of finite strings over A, and let A^B denote the set of finite bags over A. For a natural number n, let $\widehat{n}$ denote the set $\{1, \ldots, n\}$. An element of A^* and of A^B can be represented as a mapping $w \colon \widehat{|w|} \mapsto A$ where $|w|$ is the size of the bag or the length of the sequence. Given a quasi-order $\preceq$ on a set A, define the quasi-order $\preceq^*$ on A^* by letting $w \preceq^* w'$ if and only if there is a strictly monotone[1] injection $h \colon \widehat{|w|} \mapsto \widehat{|w'|}$ such that $w(j) \preceq w'(h(j))$ for $1 \leq j \leq |w|$. Define the quasi-order $\preceq^B$ on bags of A by $w \preceq^* w'$ if and only if there is a (not necessarily monotone) injection $h \colon \widehat{|w|} \mapsto \widehat{|w'|}$ such that $w(j) \preceq w'(h(j))$ for $1 \leq j \leq |w|$.

In the following theorem we state some properties of bqos which we use later in the paper. In the appendix, we show the first four properties stated in the theorem, using results from [Mil85]. The proof of property 5 is in [Mar99]. We use $\mathcal{P}(A)$ to denote the power set of A.

Theorem 3.1

1. Each bqo is wqo.
2. If A is finite, then $(A, =)$ is bqo.
3. If $(A, \preceq)$ is bqo, then $(A^*, \preceq^*)$ is bqo.
4. If $(A, \preceq)$ is bqo, then $(A^B, \preceq^B)$ is bqo.
5. If $(A, \preceq)$ is bqo, then $(\mathcal{P}(A), \sqsubseteq)$ is bqo [Mar99][2].

Since bqo is a stronger relation than wqo (property 1), it follows by Lemma 2.1 that, to prove termination of the reachability algorithm of Section 2, it is sufficient to prove bqo of constraints under entailment. All constraint systems derived earlier

[1] meaning that $h(j_1) < h(j_2)$ if and only if $j_1 < j_2$

[2] [Jan99] provides a proof for a weaker version of the theorem, namely that bqo of $(A, \preceq)$ is sufficient for wqo of $(\mathcal{P}(A), \sqsubseteq)$.

in the literature using the approach of [AČJYK96, AJ98a, FS98]) use properties 2, 3, and 4. This implies that all these constraint systems are also bqos. An immediate consequence of property 5 is that bqo of a set of constraints implies bqo of disjunctions of these constraints.

In the next sections, we introduce several constraint systems applying the following two steps.

1. We show better quasi-ordering of a constraint system $\mathbb{C}_1$ using properties 2, 3, and 4 in Theorem 3.1 (following a similar methodology to that described in [AČJYK96, AJ98a, FS98]).

2. We use property 5 to derive better quasi-ordering of a new more compact constraint system $\mathbb{C}_2$ defined as disjunctions of constraints in $\mathbb{C}_1$.

We notice that, although $\mathbb{C}_2$ is more compact, the computational complexity for checking membership and entailment may be higher for $\mathbb{C}_2$ than for $\mathbb{C}_1$. Furthermore, the reachability algorithm of Section 2 needs in general a higher number of iterations in case $\mathbb{C}_2$ is employed. However, in almost all cases, the compactness offered by $\mathbb{C}_2$ is the dominating factor in the application of the algorithm.

As mentioned earlier, an important difference compared to the approach of [AČJYK96, AJ98a, FS98]) is that Step 2 cannot be performed within that framework. This is illustrated by the following example, which shows that wqos in general are not closed under disjunction.

Example 3.2 [Rado's Example] Consider the the set $X = \{(a,b) \mid a < b\} \subseteq \mathcal{N}^2$. Define a set $\mathbb{C}_1 = \{\phi_{i,j} \mid \ (i,j) \in X\}$ of constraints, such that the denotation $[\![\phi_{i,j}]\!] \subseteq X$ of $\phi_{i,j}$ is the set $\{(a,b) \mid (j < a) \vee ((i = a) \wedge (j \leq b))\}$. It is straightforward to check that $\mathbb{C}_1$ is wqo: suppose that we have a bad sequence $\phi_{a_1,b_1}, \phi_{a_2,b_2}, \ldots$. Consider the sequence $(a_1,b_1), (a_2,b_2), \ldots$. It follows that $a_j \leq b_1$ for all $j \geq 1$. This means that we have a subsequence of the form $(a, b_{i_1}), (a, b_{i_2}), \ldots$, and hence there are k and ℓ such that $b_{i_k} \leq b_{i_\ell}$ which is a contradiction.

Now, we consider a set $\mathbb{C}_2$ constraints of the form ψ_j, where $\psi_j \equiv \phi_{0,j} \vee \cdots \vee \phi_{j-1,j}$. The sequence $\psi_1, \psi_2, \ldots$ is a bad sequence, since for each $k, \ell : k < \ell$, we have $(k,\ell) \in [\![\psi_\ell]\!]$, but $(k,\ell) \notin [\![\psi_k]\!]$, and hence $[\![\psi_\ell]\!] \not\subseteq [\![\psi_k]\!]$. In Figure 1, we give graphic illustrations of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$.

□

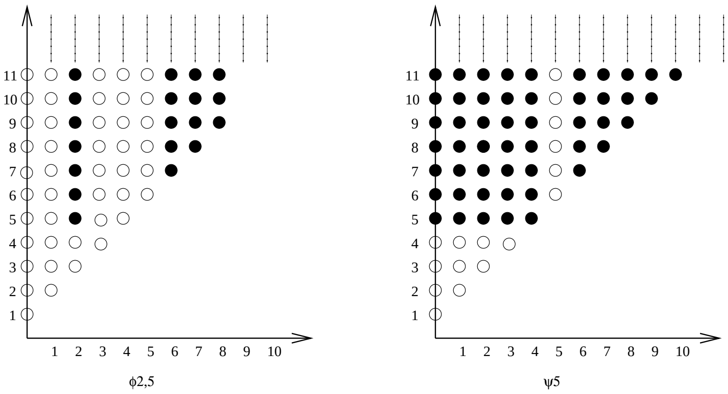

Figure 1: A graphic illustration of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$. Filled circles represent points satisfying the corresponding constraint.

4 Existential Zones and Timed Petri Nets

In this section we introduce a constraint system, called *existential zones* $\mathbb{Z}$, for systems with unboundedly many clocks, and apply it for verification of Timed Petri Nets.

We let $\mathcal{N}$, $\mathcal{Z}$, and $\mathcal{R}^{\geq 0}$ denote the sets of natural numbers, integers, and nonnegative reals, respectively. Sometimes we write bags as lists, so e.g. $(2.4, 5.1, 5.1, 2.4, 2.4)$ represents a bag B over $\mathcal{R}^{\geq 0}$ where $B(2.4) = 3$, $B(5.1) = 2$, and $B(x) = 0$ for $x \neq 2.4, 5.1$.

An *existential zone* Z is of the form $\exists x_1, \ldots, x_n.\ \phi_1 \wedge \cdots \wedge \phi_m$, where each ϕ_i is of one of the forms $y_2 - y_1 \leq \alpha$, $y \leq \alpha$, $\alpha \leq y$, $\alpha \in \mathcal{Z}$ and $y, y_1, y_2 \in \{x_1, \ldots, x_n\}$. The existential zone Z characterizes an infinite set of *clock configurations* each of which is a bag over $\mathcal{R}^{\geq 0}$. More precisely, for a clock configuration $\gamma = (c_1, \ldots, c_k)$ and an injection $h\colon \widehat{n} \mapsto \widehat{k}$, called a *witness*, we have $\gamma, h \models Z$ iff $\phi_i[c_{h(1)}/x_1, \ldots, c_{h(n)}/x_n]$ holds for each $i : 1 \leq i \leq m$. Also, $\gamma \models Z$ (or equivalently $\gamma \in [\![Z]\!]$) iff $\gamma, h \models Z$, for some witness h.

We observe that checking membership for existential zones is of exponential complexity, as we have to consider exponentially many witnesses. This is in contrast to *zones* (from the theory of real-time automata) which has a membership problem of polynomial complexity. However, in practice, the performance of existential zones will be comparable to that of zones, since the size of the problem, in the case of zones, grows exponentially with the number of clocks inside the system. In a similar manner to membership, we can use witnesses to extend existing procedures for checking entailment from zones to existential zones.

4.1 Existential Zones are BQO

We apply the methodology of Section 3 to show that $\mathbb{Z}$ are bqo.

1. In [AJ98b] we consider a constraint system, which we here call *existential regions* $\mathbb{R}$. Existential regions are built starting from finite domains, and repeatedly building finite strings, bags, and sets. From properties 2, 3, and 4 in Theorem 3.1, it follows that $\mathbb{R}$ is bqo.

2. For each existential zone Z, there is a finite set *Regions* of existential regions such that $Z \equiv \vee Regions$.

From the above, together with property 5 of Theorem 3.1, we get

Theorem 4.1 $\mathbb{Z}$ is bqo.

4.2 Timed Petri Nets

Each token in a *Timed Petri Net (TPN)*, is equipped with a clock representing the "age" of the token. The firing conditions of a transition include the usual ones for Petri nets. Furthermore, each arc, leading from a place to a transition or conversely, is labeled with a subinterval of the natural numbers. When a transition is fired, the tokens removed from the input places of the transition, and the tokens added to the output places, should have ages lying in the intervals of the corresponding arcs. TPNs cannot be modelled in the framework of timed automata, since they operate on an unbounded number of clocks. Due to space limitations, we do not give the formal definition of TPNs here. The definition can be found in [AN99], where we also show how to compute the set of predecessors of an existential zone with respect to the transitions of a TPN. From this, together with Theorem 4.1, and the fact that existential zones allow checking membership and entailment, we have the following theorem (Recall that $Pre^*(Z)$ characterizes the set of all configurations which can reach a configuration in $[\![Z]\!]$ in finitely many steps).

Theorem 4.2 For an existential zone Z, the set $Pre^*(Z)$ is computable as a finite set of existential zones. Hence the reachability problem is decidable for TPNs.

Although the main contribution of this paper is not showing new decidability results, it is worth mentioning that the above result has not previously appeared in the literature.

Remark Our model assumes a lazy (non-urgent) behaviour of TPNs. In other words a transition may choose to "let time pass" and not fire, even if that implies that the transition becomes disabled (due to the tokens becoming "too old"). In fact it can be shown [JLL77] that very simple classes of TPNs with urgent behaviours can simulate two-counter machines, and hence almost all verification problems are undecidable for them.

4.3 Implementation Effort

We can extend the results on existential zones further and consider "existential variants" of CDDs [LPWY99] and DDDs [MLAH99], constraint systems which have recently appeared in the literature of real-time automata. We use the fact that each existential CDD (DDD) is equivalent to a disjunction of existential zones. We have implemented a prototype to perform reachability analysis for TPNs, based on a DDD package described in [ML98]. We used the tool to verify a parametrized version of Fischer's protocol (the details can be found in [AN99]). The reachable state space, represented by 45 existential DDDs, takes 3.5 seconds to compute on a Sun Ultra 60 with 512 MB memory and a 360 MHz UltraSPARC-II processor. In the process, *Pre* was computed for 51 existential DDDs. Using existential regions [AJ98b], the analysis immediately explodes, making the verification infeasible to carry out. In fact existential zones (CDDS, DDDs) offer the same advantages over existential regions as *zones* over *regions* in verification tools for real-time automata [LPY97, Yov97].

5 Broadcast Protocols

We consider *broadcast protocols*, which consist of an arbitrary number of identical finite-state processes, communicating through rendezvous or through broadcast. We assume a finite set $\{s_1, \ldots, s_n\}$ of *states*, and a set $\{x_1, \ldots, x_n\}$ of variables which range over the natural numbers. A *configuration* γ of a protocol is a tuple $(a_1, \ldots, a_n)$

of natural numbers, where a_i represents the number of processes which are in the state s_i. In [EFM99] a constraint system, which we here refer to as $\mathbb{B}$, is defined, where each constraint is a tuple $(b_1, \ldots, b_n)$ with a denotation $[\![(b_1, \ldots, b_n)]\!]$ which is the upward closed set $\{(a_1, \ldots, a_n) \mid (b_1, \ldots, b_n) \leq (a_1, \ldots, a_n)\}$. In [DEP99] several new constraint systems for broadcast protocols are proposed, and compared with regard to the efficiency parameters mentioned in Section 3. The most general of these constraint systems , called **AD** in [DEP99], consists of conjunctions of constraints each of the form $x_{i_1} + \cdots + x_{i_k} \geq b$, where $x_{i_1}, \ldots, x_{i_k}$ are distinct variables of $\{x_1, \ldots, x_n\}$. Two special cases are considered: **NA** where k is always equal to 1, and **DV** where the set of variables occurring in the different conjuncts are assumed to be disjoint. Since these new constraint systems are not constructed applying the basic set of constraint operations (described in Section 3), a separate (different) proof of termination is required for each one.

Applying the method of Section 3 we can show bqo of **AD**, **NA**, and **DV** uniformly as follows. From properties 2 and 3 in Theorem 3.1, it follows that $\mathbb{B}$ is bqo. Furthermore, it is straightforward to show that each constraint in **AD**, **NA**, and **DV** is equivalent to the disjunction of a finite set of constraints in $\mathbb{B}$. From property 5 of Theorem 3.1, we get

Theorem 5.1 **AD**, **NA**, and **DV** are bqo.

In fact we can derive the bqo property for a more general constraint system than **AD**, namely that consisting of basic constraints of the form $a_1x_1 + \cdots + a_kx_k \geq b$, combined through conjunction and disjunction.

6 Lossy Channel Systems

In [AJ96], we present a constraint system, here denoted $\mathbb{L}_1$, for representing upward closed sets of strings. The constraints in $\mathbb{L}_1$ are used in [AJ96] for verification of *lossy channel systems*: finite state machines communicating over unbounded and unreliable FIFO buffers. We assume a finite alphabet Σ. For strings $w_1, w_2 \in \Sigma^*$, we let $w_1 \preceq w_2$ denote that w_1 is a (not necessarily contiguous) substring of w_2. A constraint in $\mathbb{L}_1$ is represented by a string w, where $[\![w]\!] = \{w' \mid w \preceq w'\}$.

Here, we introduce a new constraint system $\mathbb{L}_2$, defined as the smallest set such that $\mathbb{L}_2$ contains:

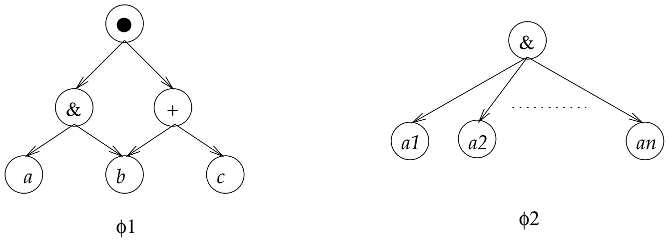

Figure 2: Two constraints in $\mathbb{L}_2$

.

- a, for each $a \in \Sigma$, where $[\![a]\!] = \{w \mid a \preceq w\}$;

and $\mathbb{L}_2$ is closed under:

- concatenation: $[\![\phi_1 \bullet \phi_2]\!] = \{w_1 w_2 \mid w_1 \in [\![\phi_1]\!] \text{ and } w_2 \in [\![\phi_2]\!]\}$;
- conjunction: $[\![\phi_1 \& \phi_2]\!] = \{w \mid w \in [\![\phi_1]\!] \text{ and } w \in [\![\phi_2]\!]\}$; and
- disjunction: $[\![\phi_1 + \phi_2]\!] = \{w \mid w \in [\![\phi_1]\!] \text{ or } w \in [\![\phi_2]\!]\}$.

Example 6.1 In Figure 2, the constraint ϕ_1 is of the form $(a \ \& \ d) \bullet (b + c)$. This means that $[\![\phi_1]\!] = \{w_1 w_2 \mid (a \preceq w_1) \text{ and } (b \preceq w_1) \text{ and } ((b \preceq w_2) \text{ or } (c \preceq w_1))\}$. The constraint ϕ_1 is equivalent to the disjunction of the following set of constraints in $\mathbb{L}_1$: $\{aba, abc, baa, bac\}$. $\square$

The constraint system $\mathbb{L}_2$ is exponentially more succinct than $\mathbb{L}_1$. More precisely, each constraint $\phi_1 \in \mathbb{L}_1$ has a linear-size translation (through the concatenation operator) into an equivalent constraint $\phi_2 \in \mathbb{L}_2$. On the other hand a constraint of the form $a_1 \& \cdots \& a_n$ (ϕ_2 in Figure 2) can only be represented in $\mathbb{L}_1$ by the disjunction of a set of constraints of size $n!$; namely the set
$\{b_1 \bullet \cdots \bullet b_n \mid (b_1, \ldots, b_n) \text{ is a permutation of } (a_1, \ldots, a_n)\}$.

In a similar manner to Section 4 and Section 5 we can use properties of $\mathbb{L}_1$ and $\mathbb{L}_2$ to conclude the following

Theorem 6.2 $\mathbb{L}_2$ is bqo.

7 Integral Relational Automata

An *Integral Relational Automaton (IRA)* operates on a set of $X = \{x_1, \ldots, x_n\}$ of *variables* assuming values from the set $\mathcal{Z}$ of integers. The transitions of the automaton are labeled by guarded commands of the form $g \rightarrow stmt$ in which the guard g is a boolean combination of inequalities of form $x < y$, $c < x$, or $x < c$, for $x, y \in X$ and $c \in \mathcal{Z}$; and where the body *stmt* contains, for each $x \in X$, an assignment of one of the forms $x := y$, $x := c$, or $x := \{?\}$, for $y \in X$ and $c \in \mathcal{Z}$. The assignment $x := \{?\}$ is a "read" operation putting an arbitrary integer into the variable x. A *configuration* γ of an IRA is a mapping from X to $\mathcal{Z}$. Sometimes, we write γ as a tuple $(\gamma(x_1), \ldots, \gamma(x_n))$. For $c \in \mathcal{Z}$, we use the convention that $\gamma(c) = c$.

A constraint system, called the *sparser than* system $\mathbb{S}_1$, is defined in [Čer94], for verification of IRAs as follows. Let c_{min} (c_{max}) be the smallest (largest) constant occurring syntactically in the IRA. Define $C = \{c_{min}, \ldots, c_{max}\}$ to be the set of integers between c_{min} and c_{max}. A constraint ϕ in $\mathbb{S}_1$ is a mapping from X to $\mathcal{Z}$. In a similar manner to configurations, we assume $\gamma(c) = c$ for $c \in \mathcal{Z}$. A configuration γ satisfies ϕ iff for each $x, y \in X \cup C$, we have (i) $\gamma(x) \leq \gamma(y)$ iff $\phi(x) \leq \phi(y)$; and (ii) if $\phi(x) \leq \phi(y)$ then $\phi(y) - \phi(x) \leq \gamma(y) - \gamma(x)$.

Example 7.1 Assume $X = \{x_1, x_2, x_3\}$ and $C = \{5\}$. Consider a constraint $\phi = (10, 5, 12)$, then $\gamma_1 = (12, 5, 17) \in [\![\phi]\!]$, while $\gamma_2 = (8, 5, 16) \notin [\![\phi]\!]$ (since $\phi(x_1) - \phi(x_2) = 5 \not\leq \gamma_2(x_1) - \gamma_2(x_2) = 3$), and $\gamma_3 = (12, 4, 17) \notin [\![\phi]\!]$ (since $\phi(5) = 5 \leq \phi(x_2) = 5$ while $\gamma_3(5) = 5 \not\leq \gamma_3(x_2) = 4$). $\square$

We introduce a new constraint system $\mathbb{S}_2$, such that a constraint ϕ in $\mathbb{S}_2$ is a conjunction of conditions of the forms $c \leq x$, $x \leq c$, and $c \leq y - x$, where $x, y \in X$ and $c \in \mathcal{Z}$. The satisfiability of ϕ by a configuration γ is defined in the obvious way.

Example 7.2 Assume $X = \{x_1, x_2\}$ and $C = \{5\}$. The constraint $5 < x_2$ in $\mathbb{S}_2$ is equivalent to the disjunction of the following set of constraints in $\mathbb{S}_1$:
$\{(4,7), (5,7), (6,7), (7,7), (8,7)\}$. Notice that the constraints correspond to the different relative values which x_1 may have with respect to the constant 5 and the variable x_2. $\square$

In a similar manner to the constraint systems in the previous sections, we can show that $\mathbb{S}_2$ is exponentially more succinct than $\mathbb{S}_1$ and that the following theorem holds.

Theorem 7.3 $\mathbb{S}_2$ is bqo.

8 Conclusions and Future Work

We have proposed *better quasi-orderings*, a refinement of the theory of *well quasi-ordering*, as a more suitable framework for symbolic model checking, since they allow us to build constraint systems, which are more compact than previous ones. For instance, we show better quasi-ordering of complex expressions for upward closed sets of strings, used for verification of lossy channel systems, arbitrary boolean combinations of linear inequalities, used for verification of broadcast protocols. We also achieve similar results for binary constraints which can be applied for model checking of real-time systems and relational automata.

One direction for future work is to design efficient data structures for manipulating the new constraint systems. For example, in this paper we show that efficient data structures for verification of real-time automata can be modified so that they become applicable for verification of systems with unbounded numbers of clocks. In [DEP99] efficient representations are described for verification of broadcast protocols. It will be interesting to investigate the feasibility of defining a general framework for implementation of better quasi-ordered constraint systems.

Furthermore, in addition to disjunction, bqos are closed under several other operations which do not preserve wqo. For instance, in contrast to wqos, bqos are closed under taking infinite sets and infinite strings. This means that we can consider much richer structures for building constraints. Therefore, although the main concern of this work is that of efficiency, we believe that the approach will eventually also lead to decidability results for new classes of infinite-state systems.

Acknowledgements

We are grateful to Alberto Marcone and Petr Jančar for many interesting discussions on the theory of bqos. Many thanks to Ahmed Bouajjani, Purushothaman Iyer, and Bengt Jonsson for comments on earlier versions of the paper. Special thanks to Jesper Møller and Henrik Reif Andersen for letting us use their DDD implementation.

References

[AČJYK96] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 313–321, 1996.

[AJ96] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AJ98a] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems, 1998. To appear in the journal of Theoretical Computer Science.

[AJ98b] Parosh Aziz Abdulla and Bengt Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Proc. TACAS '98, 4^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 298–312, 1998.

[AN99] Parosh Aziz Abdulla and Aletta Nylén. BQOs and timed Petri nets. Technical Report DoCS 99/102, Department of Computer Systems, Uppsala University, 1999. Available at `http://www.docs.uu.se/~aletta/bqo/tpn.ps`.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In Abiteboul and Shamir, editors, *Proc. ICALP '94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer Verlag, 1994.

[DEP99] G. Delzanno, J. Esparza, and A. Podelski. Constraint-based analysis of broadcast protocols. In *Proc. CSL'99*, 1999.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. 14^{th} IEEE Int. Symp. on Logic in Computer Science*, 1999.

[FS98] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere. Technical Report LSV-98-4, Ecole Normale Supérieure de Cachan, April 1998.

[Jan99] Petr Jančar. ω^2-well quasi-orderings and reachability analysis. Technical Report 158, Department of Computing Systems, Uppsala University, 1999.

[JLL77] N. D. Jones, L. H. Landweber, and Y. E. Lyen. Complexity of some problems in Petri nets. *Theoretical Computer Science*, (4):277–299, 1977.

[LPWY99] K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using clock difference diagrams. In *Proc. 11^{th} Int. Conf. on Computer Aided Verification*, 1999.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1-2), 1997.

[Mar99] A. Marcone. Fine and axiomatic analysis of the quasi-orderings on $\mathcal{P}(q)$. Technical Report 17/99/RR, *Rapporto di Ricerca del Dipartimento di Matematica e Informatica dell'Università di Udine*, 1999.

[Mil85] E. C. Milner. Basic wqo- and bqo-theory. In I. Rival, editor, *Graphs and Orders*, pages 487–502. 1985.

[ML98] Jesper Møller and Jakob Lichtenberg. Difference decision diagrams. Master's thesis, Department of Information Technology, Technical University of Denmark, Building 344, DK-2800 Lyngby, Denmark, August 1998.

[MLAH99] Jesper Møller, Jakob Lichtenberg, Henrik R. Andersen, and Henrik Hulgaard. Difference decision diagrams. Technical Report IT-TR-1999-023, Department of Information Technology, Technical University of Denmark, February 1999.

[Yov97] S. Yovine. Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1-2), 1997.

A Appendix

Basics of the Theory of Better Quasi-Orderings

We let $\mathcal{N}^{<\omega}$ ($\mathcal{N}^{\omega}$) denote the set of finite (infinite) strictly increasing sequences over the set $\mathcal{N}$ of natural numbers. For $s \in \mathcal{N}^{<\omega}$, we let $\lambda(s)$ be the set of natural numbers occurring in s, and if s is not empty then we let $tail(s)$ be the result of deleting the first element of s. For $s_1 \in \mathcal{N}^{<\omega}$ and $s_2 \in \mathcal{N}^{<\omega} \cup \mathcal{N}^{\omega}$, we write $s_1 \ll s_2$ to denote that s_1 is a proper prefix of s_2. If s_1 is not empty then we write $s_1 \ll_* s_2$ to denote that $tail(s_1) \ll s_2$. An infinite set $\beta \subseteq \mathcal{N}^{<\omega}$ is said to be a barrier if the following two conditions are satisfied.

- There are no $s_1, s_2 \in \beta$ such that $\lambda(s_1) \subsetneq \lambda(s_2)$.
- For each $s_2 \in \mathcal{N}^{\omega}$ there is $s_1 \in \beta$ with $s_1 \ll s_2$.

Let $(A, \preceq)$ be a quasi-ordering. An *A-pattern* is a mapping $f : \beta \to A$, where β is a barrier. We say that f is *good* if there are $s_1, s_2 \in \beta$ such that $s_1 \ll_* s_2$ and $f(s_1) \preceq f(s_2)$. We say that $(A, \preceq)$ is a *better quasi-ordering* if each A-pattern is good.

We use A^{ω} to denote the set of infinite sequences over A. For $w \in A^* \cup A^{\omega}$, we let $w(i)$ be the the i^{th} element of w. For a quasi-ordering $(A, \preceq)$, we define the quasi-ordering $(A^{\omega}, \preceq^{\omega})$ where $w_1 \preceq^{\omega} w_2$ if and only if there is a strictly increasing injection $h : \mathcal{N} \to \mathcal{N}$ such that $w_1(i) \preceq w_2(h(i))$, for each $i \in \mathcal{N}$.

We use the following two properties (from [Mil85]) to prove Theorem 3.1.

Lemma A.1
- If β is a barrier and $\beta = \beta_1 \cup \beta_2$, then then there is a barrier α such that $\alpha \subseteq \beta_1$ or $\alpha \subseteq \beta_2$. (using induction on n we can generalize this property to $\beta = \beta_1 \cup \cdots \cup \beta_n$).
- If $(A, \preceq)$ is bqo then $(A^{\omega}, \preceq^{\omega})$ is bqo

Proof of Theorem 3.1 We show properties 1-4. Proof of property 5 can be found in [Mar99].

1. Follows immediately from definitions of bqo and wqo.

2. Consider $(A, =)$ where $A = \{a_1, \ldots, a_n\}$ is finite. Let $f : \beta \to A$ be an A-pattern. Define $\beta_i = f^{-1}(a_i)$, for $i : 1 \leq i \leq n$. By Lemma A.1, there is a barrier $\alpha \subseteq \beta_i$, for some $i : 1 \leq i \leq n$. Take any $s_1 \in \alpha$ and any $s_2 \in \mathcal{N}^\omega$, where $s_1 \ll_* s_2$. Since α is a barrier, we know that there is $s_3 \in \alpha$ such that $s_3 \ll s_2$, and that $s_3 \not\subseteq s_1$. It follows that $s_1 \ll_* s_3$ and hence f is good.

3. Suppose that $(A, \preceq)$ is bqo. We show that $(A^*, \preceq^*)$ is bqo. Take any $b \notin A$. For $w \in A^*$, we let w' denote wb^ω (i.e., we add infinitely many b:s to the end of w). It is clear that $w_1 \preceq^* w_2$ if and only if $w'_1 \preceq^\omega w'_2$. Let $f : \beta \to A^*$ be an A^*-pattern. We know that $f' : \beta \to A^\omega$, where $f'(s) = w'$ iff $f(s) = w$, is an A^ω-pattern. By Lemma A.1 it follows that there are $s_1, s_2 \in \beta$ such that $s_1 \ll_* s_2$ and $f'(s_1) \preceq^\omega f'(s_2)$, and hence $f(s_1) \preceq^* f(s_2)$. This means that f is good.

4. Follows from 3.