Better-Structured Transition Systems

Parosh Aziz Abdulla   Aletta Nylén

Dept. of Information Technology, Uppsala University
P.O. Box 337, SE-751 05 Uppsala, Sweden
Email:{parosh, aletta}@it.uu.se

Abstract

In automated verification of infinite-state systems, a variety of algorithms that operate on constraints representing sets of states have been developed. Many of these algorithms rely on well quasi-ordering of the constraint system for proving termination. A number of methods for generating new well quasi-ordered constraint systems have been proposed. However, many of these constraint systems suffer from constraint explosion as the number of constraints generated during analysis grows exponentially with the size of the problem. We suggest using the theory of better quasi-ordering to prove termination since that will allow generation of constraint systems that are less prone to constraint explosion. We also present a method to derive such constraint systems. We introduce existential zones, a constraint system for verification of systems with an unbounded number of clocks and use our methodology to prove that existential zones are better quasi-ordered. We show how to use existential zones in verification of timed Petri nets and present some experimental results. Finally, we present several other constraint systems which have been derived using our methodology.

1 Introduction

Model checking [CES86] infinite state systems is a major current challenge in automatic program verification. In the literature there is a wide variety of model checking techniques for finite state systems. By using *symbolic methods*, i.e., performing the analysis on *constraints* representing (potentially infinite) sets of states rather than on individual states, finite state techniques can be adopted to work in the analysis of infinite state systems. In the literature, a variety of symbolic algorithms have been presented for analyzing *well-structured transition systems* [FS98, AČJYK00, AJ01a], transition systems where the state space is equipped with a well quasi-ordering and where the transition relation is monotonic with respect to that ordering.

In this work, we concentrate on reachability analysis, a technique that can be used to analyze several classes of safety properties [VW86, GW93]. An instance of the reachability problem consists of a well-structured transition system, an initial state and an upward closed set of bad states. Intuitively, the bad states are states that we do not want to occur during an execution of the system so we are interested in showing that no bad state is reachable from the initial state.

To solve the reachability problem we apply a general algorithm for reachability analysis of infinite-state systems described in [AČJYK00, AJ01a]. Given a constraint representing the set of bad states, the algorithm performs a fixpoint iteration generating successively larger sets of constraints representing the set of states from which a bad state can be reached. A main issue when using the algorithm is to show that the fixpoint iteration always terminates. A result from [AČJYK00] is that the algorithm is guaranteed to terminate exactly when the constraint system is *well quasi-ordered*, i.e., for each infinite sequence $\phi_0, \phi_1, \phi_2, \ldots$ of constraints there are i and j with $i < j$ where the set of states represented by ϕ_j is a subset of the set of states that ϕ_i characterizes. Thus, an important part in using the algorithm is to develop methods for generating new well quasi-ordered constraint systems. One such methodology that is based on Higman's theorem [Hig52], is described in [AČJYK00, AJ01a, FS98]. The main idea is to start from a set of basic constraints and repeatedly derive new ones using the fact that well quasi-ordering is closed under certain operations on constraints such as building finite strings, vectors, sets, multisets, trees, etc.

Many constraint systems based on the theory of well quasi-ordering suffer from a constraint explosion problem as the number of constraints generated during the analysis grows exponentially with the size of the problem. In order to make the analysis feasible in practice we need to develop more compact constraint systems, systems that increase the number of states represented by a single constraint. We suggest ordering state spaces by a stronger relation than well quasi-ordering, namely that of *better quasi-ordering* [Mil85, Pou85]. We call such systems *better-structured transition systems*. We also present a methodology to develop better quasi-ordered constraint systems which can be used together with existing algorithms designed for well-quasi ordered constraints.

There are two properties of better quasi-orderings that make them appropriate for symbolic model checking. (i) Better quasi-ordering implies well quasi-ordering, and (ii) in addition to the operations mentioned above, better quasi-ordering is closed under disjunction (in contrast to well quasi-ordering). The first property implies that all verification algorithms designed for well quasi-ordered constraint systems are also applicable for better quasi-ordered ones. The second property allows us to derive new constraint systems where each constraint is a finite disjunction of old constraints, thus making the new system more compact than the old one.

We introduce a new constraint system called *existential zones* for verification of systems with unboundedly many clocks. Existential zones are related to existential regions which were used in [AJ01b] for verification of timed networks. Using our methodology we can show better quasi-ordering of existential regions since they are constructed by repeatedly building finite strings and multisets. Each existential zone is equal to the disjunction of a set of existential regions. Since better quasi-ordering is closed under disjunction it follows that existential zones are better quasi-ordered (and hence well quasi-ordered).

We demonstrate practical feasibility of existential zones by using them to verify safety properties for timed Petri nets. In timed Petri nets, each token is equipped with a real-valued clock representing the age of the token. Furthermore, each arc is equipped with an interval defined by two natural numbers restricting the ages of the tokens travelling the arc. Timed Petri nets cannot be modelled as timed automata [AD90] since the latter operate on a finite number of clocks while timed Petri nets in general may generate an unbounded number of clocks. We have implemented a prototype for verifying safety properties for timed Petri nets using existential zones. The tool allows us to verify a parameterized version of Fischer's protocol [SBM92] within a few seconds.

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

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

Existential zones are variants of zones, a symbolic representation used in several tools for verification of timed automata, e.g. KRONOS [Yov97] and UPPAAL [LPY97]. However, zones characterize finite sets of clocks and therefore cannot be used to analyze timed Petri nets.

A model close to timed Petri nets, timed networks, was considered in [AJ01b]. A timed network consists of an arbitrary number of timed processes and hence contain an unbounded number of clocks. The constraint system used in that work was that of existential regions, a constraint system that is far less efficient than existential zones and the number of existential regions generated during analysis explode even on small applications.

Most earlier work on studying decidability issues for timed Petri nets, e.g. [RP85, BD91, GMMP91, RGdFE99], either report undecidability results or decidability under the assumption that the net is bounded. A work closely related to ours is [dFERA00]. The authors consider the coverability problem for a class of timed Petri nets similar to our model. The main difference is that in [dFERA00], it is assumed that the ages of the tokens are natural numbers. Furthermore, it is not evident how efficient the constraint system is in practical applications.

In this paper, we consider only lazy timed Petri nets. In fact it can be shown [JLL77] that very simple classes of timed Petri nets with urgent behaviours can simulate two-counter machines, and hence almost all verification problems are undecidable for them. This is not a problem when checking reachability since the set of transitions of an urgent net is a subset of the set of transitions of the corresponding lazy net. This means that if a set of markings is not reachable in the lazy net, then it is certainly not reachable in the urgent net.

Outline After some preliminary definitions in Section 2, we devote Section 3 to recalling the notion of constraints and describing the reachability algorithm. In Section 5 we introduce better-structured transition systems and describe how they can be used in verification. Timed Petri nets are defined in Section 6 and in Section 7 we introduce existential zones and show how they can be used in the analysis of timed Petri nets. The results of our experiments are described in Section 8. In Section 9, Section 10 and Section 11 we provide better quasi-ordered constraint systems for the verification of broadcast protocols, lossy channel systems and integral relational automata.

2 Preliminaries

We let $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}^{\geq 0}$ denote the set of natural numbers, integers and non-negative reals respectively. A *multiset* $\mathcal{M}$ over a set A is a mapping from A to $\mathbb{N}$ such that for an element $a \in A$, $\mathcal{M}(a)$ is the number of occurrences of a in $\mathcal{M}$. We sometimes write multisets as lists, so e.g. $\mathcal{M} = (a, a, b)$ means that $\mathcal{M}(a) = 2$, $\mathcal{M}(b) = 1$ and $\mathcal{M}(x) = 0$ for $x \notin \{a, b\}$. We also sometimes write $\mathcal{M}$ as $\left(a^2, b\right)$.

$A^{\mathcal{M}}$ is used to denote the set of multisets over A. For two multisets $\mathcal{M}_1$ and $\mathcal{M}_2$ we define $\mathcal{M}_1 \leq \mathcal{M}_2$ if $\mathcal{M}_1(a) \leq \mathcal{M}_2(a)$ for each $a \in A$. We also define the addition $\mathcal{M}_1 + \mathcal{M}_2$ to be the multiset $\mathcal{M}$ where $\mathcal{M}(a) = \mathcal{M}_1(a) + \mathcal{M}_2(a)$ and, assuming $\mathcal{M}_2 \leq \mathcal{M}_1$, the subtraction $\mathcal{M}_1 - \mathcal{M}_2$ is the multiset $\mathcal{M}$ where $\mathcal{M}(a) = \mathcal{M}_1(a) - \mathcal{M}_2(a)$ for each $a \in A$. We use $\emptyset$ to denote the empty multiset, i.e., $\emptyset(a) = 0$ for each $a \in A$.

A *quasi-order* is a pair $(A, \preceq)$ where $\preceq$ is a reflexive and transitive binary relation on a set A. A quasi-order is said to be a *well-quasi order* if for any infinite sequence $a_0, a_1, a_2, \ldots$ of elements of A, there are i and j with $i < j$ and $a_i \preceq a_j$. Hence there is no infinite strictly decreasing sequence of elements of A.

Given a quasi-order $(A, \preceq)$, we say that a set $B \subseteq A$ is *upward closed* if $a \in B$ and $a \preceq b$ imply $b \in B$. We let $\min(B)$ be the set of minimal elements in B, i.e., the set $\{a \mid a \in B \text{ and } \nexists a' \in B \,.\, a' \preceq a\}$. If $(A, \preceq)$ is a well quasi-order we know that $\min(B)$ is finite. Notice that if B is upward closed, then B is completely characterized by its minimal elements.

We use A^* (A^ω) to denote the set of finite (infinite) strings over A. For a string w we let $w(i)$ be the i^{th} element of w. Given a quasi-order $(A, \preceq)$ we define a quasi order $(A^*, \preceq^*)$ such that $a_1 \bullet \cdots \bullet a_m \preceq^* b_1 \bullet \cdots \bullet b_n$ if and only if there is an injection $h : \{1, \ldots, m\} \to \{1, \ldots, n\}$ such that $a_i \preceq b_{h(i)}$ for each $i \in \{1, \ldots, m\}$ and h is strictly monotone, meaning that $h(i) < h(j)$ if and only if $i < j$. A quasi-order $\left(A^{\mathcal{M}}, \preceq^{\mathcal{M}}\right)$ on the set of multisets over A can be defined in a similar manner. Note that for multisets, the injection h is not necessarily monotone. We also define the quasi-order $(A^\omega, \preceq^\omega)$ where $w_1 \preceq^\omega w_2$ if and only if there is a strictly monotone injection $h : \mathbb{N} \to \mathbb{N}$ where $w_1(i) \preceq w_2(h(i))$ for each $i \in \mathbb{N}$. We let $\mathcal{P}(A)$ denote the powerset of A and define a relation $\sqsubseteq$ on $\mathcal{P}(A)$ so that $A_1 \sqsubseteq A_2$ if and only if $\forall b \in A_2 \ : \ \exists a \in A_1 \ : \ a \preceq b$.

2.1 Well-Structured Transition Systems

A *transition system* is a tuple $(S, \longrightarrow)$ where S is a potentially infinite set of states and $\longrightarrow$ is a transition relation. We use $\xrightarrow{*}$ to denote the transitive closure of $\longrightarrow$. Assume that the set S of states is equipped with a quasi-order $\preceq$. We say that the transition system is *monotonic* with respect to $\preceq$ if for each $s_1, s_1', s_2 \in S$ such that $s_1 \preceq s_1'$ and $s_1 \longrightarrow s_2$ there is a state s_2' such that $s_2 \preceq s_2'$ and $s_1' \longrightarrow s_2'$.

A *well-structured transition system* is a triple $(S, \longrightarrow, \preceq)$ where $(S, \preceq)$ is a well quasi-ordering and $(S, \longrightarrow)$ is a transition system which is monotonic with respect to $\preceq$.

3 Reachability Analysis

We concentrate on the *reachability problem*: given a well-structured transition system $(S, \longrightarrow, \preceq)$, an initial state $s_0 \in S$ and a set of states $S_F \subseteq S$ which is upward closed with respect to $\preceq$, we want to find out whether S_F can be reached from the initial state, or more precisely we want to find out whether there is a state $s_F \in S_F$ such that $s_0 \xrightarrow{*} s_F$. The states in S_F are usually bad states, i.e., states that we do not want to occur during an execution. Several classes of safety properties can be reduced to the reachability problem using standard techniques [VW86, GW93].

The reachability analysis is performed on constraints representing sets of states that are upward closed with respect to $\preceq$. A constraint ϕ characterizes a potentially infinite set of states $[\![\phi]\!]$. We define an entailment relation $\preceq$ on constraints where $\phi_1 \preceq \phi_2$ if and only if $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$. We sometimes write disjunctions $\phi_1 \vee \cdots \vee \phi_n$ of constraints as $\bigvee \{\phi_1, \ldots, \phi_n\}$. For a finite set of constraints Φ, the disjunction $\bigvee \Phi$ denotes the union of the denotations of its elements, i.e., $[\![\bigvee \Phi]\!] = \bigcup_{\phi \in \Phi} [\![\phi]\!]$. We recall the relation $\sqsubseteq$ defined in Section 2 and note that for sets Φ_1, Φ_2 of constraints, $\Phi_1 \sqsubseteq \Phi_2$ implies $\bigvee \Phi_1 \preceq \bigvee \Phi_2$.

3.1 The Verification Algorithm

For the verification, we use a general algorithm for symbolic reachability analysis [AJ01a]. The algorithm performs a backward reachability analysis where a key ingredient is computing the pre-image of a constraint. That is, given a constraint ϕ, we compute a finite set of constraints $Pre(\phi)$ such that $[\![\bigvee Pre(\phi)]\!]$ is the set $\{s \mid \exists s' \in [\![\phi]\!] \,.\, s \longrightarrow s'\}$ of states from which a state in $[\![\phi]\!]$ is reachable through a single transition. For a set Φ of constraints $Pre(\Phi)$ is the union $\bigcup_{\phi \in \Phi} Pre(\phi)$.

Given a constraint ϕ_F characterizing the set S_F of bad states, the reachability algorithm computes a set of constraints characterizing the states from which $[\![\phi_F]\!]$ can be reached in at most i steps for successively larger i. More precisely, we compute the sequence $\Phi_0, \Phi_1, \Phi_2, \ldots$ where each Φ_i characterizes the states from which $[\![\phi_F]\!]$ can be reached in i steps or less. That is, $\Phi_{i+1} = \Phi_i \cup Pre(\Phi_i)$. Note that the sequence is increasing, i.e., $\Phi_0 \subseteq \Phi_1 \subseteq \cdots$. This implies that the procedure can be terminated if and when we reach a point j where $\Phi_j \sqsubseteq \Phi_{j+1}$. In such a case we have reached a fixpoint and $\bigvee \Phi_j$ characterizes the set of all states from which S_F is reachable.

Consequently, the reachability of S_F from s_0 is equivalent to whether $s_0 \in [\![\bigvee \Phi_j]\!]$.

4 Well Quasi-Ordered Constraint Systems

In order to be able to implement the reachability algorithm we need the following four properties of the constraint system.

1. Computability of the membership relation. We need this property to check whether the initial state satisfies the fix point.
2. Computability of the entailment relation between constraints. This property is needed to check the termination condition.
3. Computability of the function *Pre* on constraints. *Pre* is used to derive each element of the sequence from the previous one.
4. Termination. We need to be able to guarantee that the algorithm will always terminate.

The fourth property is the least obvious one. However, the class of constraint systems for which termination is guaranteed is characterized by the following lemma.

Lemma 4.1 ([AČJYK00]). *A constraint system is well quasi-ordered if and only if for each infinite ($\subseteq$-increasing) sequence $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots$ of constraints, there is an i where $\Phi_j \sqsubseteq \Phi_{j+1}$ for all $j \geq i$.*

The lemma shows that the algorithm will terminate exactly in the case where the set of constraints is well quasi-ordered under the entailment relation. In [AČJYK00, AJ01a, FS98] a methodology for deriving new well quasi-ordered constraint systems from existing ones is described. The methodology is based on the fact that all finite domains are well quasi-ordered under equality and that well quasi-orderings are closed under a set of basic operations including building finite trees, strings, vectors, sets, multisets, etc. This means that we can start from a set of constraints over finite domains and then repeatedly generate new constraints by building more and more compound data structures. A typical application of this approach is a constraint system called *existential regions* which is used for verification of systems with unboundedly many clocks. Existential regions were introduced in [AJ98].

There are three properties of a given constraint system that decide how efficient the reachability analysis will be in practice. The properties are the size of the set $Pre(\phi)$, the cost of checking entailment and the number of iterations needed before termination. Constraint systems developed according to the above methodology are often infeasible for practical use due to a constraint explosion problem caused by the set $Pre(\phi)$ being too large. This problem can often be reduced be by considering new, more compact constraint system which are disjunctions of the ones derived using the operations above. In Section 7 we present a constraint system called *existential zones* where each zone corresponds to the disjunction of a (sometimes exponential) number of existential regions.

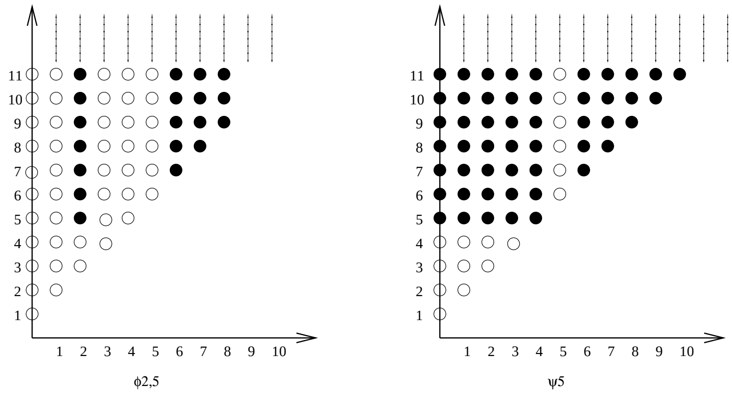


Figure 1: A graphic illustration of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$. Filled circles represent points satisfying the corresponding constraint.

Using disjunctions of constraints allows us to develop much more compact constraint systems. Existential zones are compact enough to allow us to verify a parameterized version of Fischer's protocol in a few seconds, but as we show in Example 4.2, well quasi-ordered constraint systems are not closed under disjunction. In fact, we cannot prove well quasi-ordering of existential zones within the framework of [AČJYK00, AJ01a, FS98].

Example 4.2 (Rado's Example). *We consider the set $\boldsymbol{R} = \{(a,b) \in \mathbb{N} \mid a < b\}$ and a relation $\leq_{rado}$ on $\boldsymbol{R}$ defined by $(a_1,b_1) \leq_{rado} (a_2,b_2)$ if and only if $b_1 < a_2$ or $a_1 = a_2$ and $b_1 \leq b_2$. We define a set $\mathbb{C}_1 = \{\phi_{a,b} \mid (a,b) \in \boldsymbol{R}\}$ of constraints such that the denotation $[\![\phi_{a,b}]\!]$ is the upward closure of (a,b) with respect to $\leq_{rado}$, i.e., the set $\{(a',b') \in \boldsymbol{R} \mid (a,b) \leq_{rado} (a',b')\}$. Recall that we define an entailment relation $\preceq$ on constraints where $\phi_{a_1,b_1} \preceq \phi_{a_2,b_2}$ if $[\![\phi_{a_2,b_2}]\!] \subseteq [\![\phi_{a_1,b_1}]\!]$. In fact, it follows that $\phi_{a_1,b_1} \preceq \phi_{a_2,b_2}$ if and only if $(a_1,b_1) \leq_{rado} (a_2,b_2)$.*

We will show that $(\mathbb{C}_1, \preceq)$ is a well quasi-order and define a new constraint system $\mathbb{C}_2$ where each constraint is a disjunction of constraints in $\mathbb{C}_1$. We will then show that $(\mathbb{C}_2, \preceq)$ is not a well quasi-order and hence that well quasi-ordering is not closed under disjunction.

We start by showing that $(\mathbb{C}_1, \preceq)$ is a well quasi-order. Suppose that we have an infinite sequence $\phi_{a_1,b_1}, \phi_{a_2,b_2}, \ldots$ where $\phi_{a_i,b_i} \not\preceq \phi_{a_j,b_j}$ if $i < j$. Consider the sequence $(a_1,b_1), (a_2,b_2), \ldots$. We know that $a_j \leq b_1$ for all $j > 1$. This means that we have an infinite subsequence of the form $(a,b_{i_1}), (a,b_{i_2}), \ldots$ and hence there are $k < l$ such that $b_{i_k} < b_{i_l}$, but then $\phi_{a,b_{i_k}} \preceq \phi_{a,b_{i_l}}$ which is a contradiction.

Now, we define a set $\mathbb{C}_2 = \{\psi_j \mid \psi_j = \phi_{0,j} \vee \cdots \vee \phi_{j-1,j}\}$ of constraints. The sequence $\psi_1, \psi_2, \ldots$ violates the well quasi-ordering property since for each $k, l : k < l$ we have $(k,l) \in [\![\psi_l]\!]$ but $(k,l) \notin [\![\psi_k]\!]$ and hence $[\![\psi_l]\!] \not\subseteq [\![\psi_k]\!]$, i.e., there are no $k < l$ with $\psi_k \preceq \psi_l$.

We have shown that $(\mathbb{C}_2, \preceq)$ is not a well quasi-order and that well quasi-ordering is not closed under disjunction. In Figure 1, we give graphic illustrations of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$.

5 Better-Structured Transition Systems

In the previous section we saw that in order to perform reachability analysis on disjunctions of constraints we need an ordering that is stronger than well quasi-ordering. We suggest using *better quasi-ordering*, a notion introduced by Nash-Williams in the 1960's [NW65, Mil85].

We let $\mathbb{N}^{<\omega}$ ($\mathbb{N}^{\omega}$) be the set of finite (infinite), strictly increasing sequences over $\mathbb{N}$. For a finite sequence $s \in \mathbb{N}^{<\omega}$ we let $\lambda(s)$ denote the set of natural numbers occurring in s and if s is not empty, we let $tail(s)$ be the result of deleting the first element of s. For $s_1 \in \mathbb{N}^{<\omega}$ and $s_2 \in \mathbb{N}^{<\omega} \cup \mathbb{N}^{\omega}$ we write $s_1 \ll s_2$ to denote that s_1 is a proper prefix of s_2 and if s_1 is not empty we write $s_1 \ll_* s_2$ to denote that $tail(s_1) \ll s_2$.

An infinite set $\beta \subseteq \mathbb{N}^{<\omega}$ is said to be a *barrier* if the following conditions are satisfied.

- there are no $s_1, s_2 \in \beta$ such that $\lambda(s_1) \subset \lambda(s_2)$
- for each $s_2 \in \mathbb{N}^{\omega}$ there is $s_1 \in \beta$ with $s_1 \ll s_2$

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

A *better-structured transition system* is a triple $(S, \longrightarrow, \preceq)$ where $(S, \preceq)$ is a better quasi-ordering and $(S, \longrightarrow)$ is a transition system which is monotonic with respect to $\preceq$. In verification of better-structured constraint systems, we need to be able to construct better quasi-ordered constraint systems. We will now state two properties of better quasi-orders from [Mil85] which are essential for construction of constraint systems.

Lemma 5.1 ([Mil85]).

- *If β is a barrier and $\beta = \beta_1 \cup \beta_2$ 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 a better quasi-order then $(A^{\omega}, \preceq^{\omega})$ is a better quasi-order.*

Using the properties above, we will derive some properties that allow using better quasi-ordered constraint systems in verification and which we will use to form a methodology for developing new constraint systems.

Theorem 5.2.

1. *Each better quasi-order is a well quasi-order.*
2. *If A is finite, then $(A, =)$ is a better quasi-order.*
3. *If $(A, \preceq)$ is a better quasi-order, then $(A^*, \preceq^*)$ is a better quasi-order.*
4. *If $(A, \preceq)$ is a better quasi-order, then $(A^{\mathcal{M}}, \preceq^{\mathcal{M}})$ is a better quasi-order.*

5. If $(A, \preceq)$ is a better quasi-order, then $(\mathcal{P}(A), \sqsubseteq)$ is a better quasi-order.

Proof.

1. Define an order $\sqsubset$ on sequences such that $s_1 \sqsubset s_2$ if

- $hd(s_i) < hd(s_j)$, or
- $hd(s_i) = hd(s_j)$ and $tail(s_1) \sqsubset tail(s_2)$

Note that $s_i \ll_* s_j$ imply $s_i \sqsubset s_j$. Assume a better quasi-order $(A, \preceq)$ and a barrier β. Let $\beta = s_1, s_2, \ldots$ such that $s_i \sqsubset s_{i+1}$ for all i. Now, look at the sequence $a_1, a_2, \ldots$ where $a_i = f(s_i)$ and f is an A-pattern. Since $(A, \preceq)$ is a better quasi-order there are i and j such that $s_i \ll_* s_j$ and $f(s_i) \preceq f(s_j)$ for each A-pattern f. That is, for each sequence $a_1, a_2, \ldots$ we have that there are i and j such that $i < j$ and $a_i \preceq a_j$, i.e., $(A, \preceq)$ is a well quasi-order.

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 $i \in \{1, \ldots, n\}$. By Lemma 5.1 there is a barrier $\alpha \subseteq \beta_i$ for some $i : 1 \leq i \leq n$. Take any $s_1 \in \alpha$ and any $s_2 \in \mathbb{N}^\omega$ 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 and $(A, =)$ is a better quasi-order.

3. Assume that $(A, \preceq)$ is a better quasi-order. We will show that $(A^*, \preceq^*)$ is also a better quasi-order. Take any $b \notin A$. For $w \in A^*$, we let w' denote $w \bullet b^\omega$ (i.e., we add infinitely many bs 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 $f'(s) = w'$ if and only if $f(s) = w$, is an A^ω-pattern. By Lemma 5.1 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 and that $(A^*, \preceq^*)$ is a better quasi-order.

4. Follows from 3.

5. This property was proved by Marcone in [Mar99]. In [Jan99], Jančar proved a weaker version of the theorem, namely that better quasi-ordering of $(A, \preceq)$ is sufficient for well quasi-ordering of $(\mathcal{P}(A), \sqsubseteq)$.

$\square$

Since better quasi-ordering is a stronger relation than well quasi-ordering it is sufficient to prove better quasi-ordering of a constraint system under entailment to prove termination of the reachability algorithm of Section 3. All constraint systems derived earlier in the literature based on the approach of [AČJYK00, AJ01a, FS98] use properties 2, 3 and 4. This implies that all these constraint systems arte also better quasi-ordered. A direct consequence of the property 5 is that better quasi-ordering of a set of constraints implies better quasi-ordering of disjunctions of these constraints.

In the following sections we shall introduce several constraint systems which have been derived by 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 5.2 (following a methodology similar to that described in [AČJYK00, AJ01a, FS98]).

2. We use property 5 to derive better quasi-ordering of a new, more compact constraint system $\mathbb{C}_2$ where the constraints are 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 3 in general needs more 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 efficiency of the algorithm.

As mentioned earlier, an important difference compared to the approach of [AČJYK96, AJ01a, FS98] is that the second step of our methodology cannot be performed within that framework since well quasi-orderings in general are not closed under disjunction (see Example 4.2).

6 Timed Petri Nets

Petri nets is a powerful model widely used to model resource and synchronization issues in concurrent systems. Over the years several extensions of Petri nets have been proposed. One such extension is *timed Petri nets* which allows modelling real time systems. Timed Petri nets cannot be modelled as timed automata [AD90] since timed automata operate on a finite number of clocks while a timed Petri net may generate an unbounded number of clocks. In fact, a timed Petri net has an infinite control part, as does an untimed Petri net, and it also operates on infinite data structures, namely the clocks.

A timed Petri net (TPN) is a generalization of a Petri net where each token is equipped with a real-valued clock. The value of the clock is usually called the *age* of the token. Furthermore, each arc of the net is labeled by an interval defined by two natural numbers. This interval restricts the ages of the tokens travelling the arcs.

Let *Intrv* be the set of intervals of the form $[a : b]$ where $a \in \mathbb{N}$ and $b \in \mathbb{N} \cup \{\infty\}$. For a real number x we say that $x \in [a : b]$ if $a \leq x \leq b$. A timed Petri net $\mathcal{N}$ is a tuple (P, T, In, Out) where P is a finite set of *places* and T is a finite set of *transitions*. In and Out are mappings from $P \times T$ to $Intrv^{\mathcal{M}}$ describing the input and output to the transitions. That is, for a place p and a transition t, $In(p, t)$ is a multiset of intervals where each interval is the age restriction of an arc from p to t and $Out(p, t)$ is the multiset of intervals labeling arcs leading from t to p. If $In(p, t) \neq \emptyset$ ($Out(p, t) \neq \emptyset$) we say that p is an *input (output) place* of t.

A state of a timed Petri net is called a *marking* and defines the number and ages of the tokens in each place. More precisely, a marking M is a multiset over $P \times \mathbb{R}^{\geq 0}$ where $M(p, x)$ is the number of tokens with age x in place p. We define $M(p)$ to be a multiset over $\mathbb{R}^{\geq 0}$ satisfying $(M(p))(x) = M(p, x)$.

A transition t is *enabled*, that is, it may be *fired* in a marking M if for each input arc, there is a token in the corresponding input place whose age is in the interval labeling the arc. We call these tokens the input tokens. When the transition is fired, the input tokens are removed and new tokens

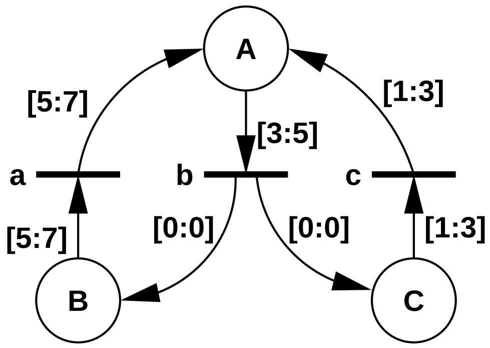


Figure 2: A small timed Petri net.

are added to the output places of the transition, one for each output arc. The ages of the new tokens are in the intervals of the corresponding arcs.

Example 6.1. *Figure 2 shows an example of a TPN where* $P = \{A, B, C\}$ *and* $T = \{a, b, c\}$*. For instance,* $In(a) = ((B, [5 : 7]))$ *and* $Out(b) = ((B, [0 : 0]), (C, [0 : 0]))$*. The initial marking of this net is the marking* $M_{init} = ((A, 0.0))$ *with only one token with age 0 in place A. None of the transitions is enabled since the age of the only token is not in the interval labeling the input arc to transition b, namely* $[3 : 5]$*. However, if we let time pass for* 3 *time units transition b will be enabled.*

We define two types of transition relations on markings. The set of *discrete transitions* $\longrightarrow_D$ which is the union $\bigcup_{t \in T} \longrightarrow_t$ where $\longrightarrow_t$ correspond to firing transition t and *timed transitions* $\longrightarrow_\delta$ that correspond to letting time pass. More precisely, we define $M_1 \longrightarrow_t M_2$ if, for each place p with $In(p, t) = (\mathcal{I}_1, \dots, \mathcal{I}_m)$ and $Out(p, t) = (\mathcal{J}_1, \dots, \mathcal{J}_n)$, there are multisets $\mathcal{M}_{In} = (x_1, \dots, x_m)$ and $\mathcal{M}_{Out} = (y_1, \dots, y_n)$ over $\mathbb{R}^{\geq 0}$ such that

- $\mathcal{M}_{In} \leq M_1(p)$
- $x_i \in \mathcal{I}_i$ for $i \in \{1, \dots, m\}$
- $y_i \in \mathcal{J}_i$ for $i \in \{1, \dots, n\}$
- $M_2(p) = M_1(p) - \mathcal{M}_{In} + \mathcal{M}_{Out}$

A timed transition lets time pass by increasing the age of all tokens by the same real number. Formally, $M_1 \longrightarrow_\delta M_2$ if $M_1 = ((p_1, x_1), \dots, (p_n, x_n))$ and there is $\delta \in \mathbb{R}^{\geq 0}$ such that $M_2 = ((p_1, x_1 + \delta), \dots, (p_n, x_n + \delta))$.

We define the relation $\longrightarrow$ to be the union $\longrightarrow_D \cup \longrightarrow_\delta$ and define $\xrightarrow{*}$ to be the reflexive transitive closure of $\longrightarrow$. For markings M_1 and M_2 we say that M_2 is reachable from M_1 if $M_1 \xrightarrow{*} M_2$. For a marking M and a set of markings $\boldsymbol{M}$ we write $M \xrightarrow{*} \boldsymbol{M}$ to denote that there is a marking $M' \in \boldsymbol{M}$ with $M \xrightarrow{*} M'$.

Notice that we assume that the timed Petri nets are non-urgent. In other words, we may choose to let time pass instead of firing an enabled transition even though the transition may be disabled due to tokens becoming too old. In an urgent timed Petri net, timed transitions that disable discrete transitions are not allowed. This means that the set of transitions of an urgent timed Petri net is a subset of the set of transitions of the corresponding non-urgent net. Therefore, if a set of markings is not reachable in the non-urgent net it is not reachable in the urgent net either. In other words, safety properties that hold for the non-urgent net also hold for the urgent net.

It has been shown [JLL77] that even simple classes of timed Petri nets with urgent behaviours can simulate two-counter machines, and thus almost all verification problems are undecidable for them.

7 Existential Zones

We will now introduce existential zones, a constraint system developed for verification of systems with an unbounded number of clocks, which we will use for verifying timed Petri nets. An existential zone characterizes an upward closed set of markings by specifying a minimum number of tokens that should be in a marking satisfying the zone and then imposing conditions on these tokens. The conditions specifies the places in which the tokens should reside and restricts their ages. The age restrictions are stated as bounds on values of clocks and bounds on differences between pairs of clock values. A marking M satisfying a zone Z should have at least the number of tokens specified by Z. Furthermore, the places and ages of these tokens should satisfy the conditions imposed by Z. In such a case M may have any number of additional tokens whose places and ages are irrelevant for the satisfiability of the zone.

An existential zone Z is a triple $\left(m, \bar{P}, D\right)$ where m is a natural number, the *placing* $\bar{P}$ is a mapping $\bar{P} : \{1, \ldots, m\} \to P$ and the *difference bound matrix)* D is a mapping $D : \{0, \ldots, m\} \times \{0, \ldots, m\} \to \mathbb{N} \cup \{\infty\}$. Intuitively, m denotes the minimum number of tokens in a marking satisfying Z, $\bar{P}$ maps each token to a place and D defines restrictions on ages of the tokens and on differences between the ages of the tokens. Difference bound matrices, or DBMs, are widely used in verification of timed automata, e.g. [Dil89, LPY95].

Example 7.1. *Consider again the timed Petri net in Figure 2. Assume that we are interested in checking the coverability of markings with at least two tokens, one in place B and one in place C, such that the ages of the tokens are at most 8 and the token in B is at most 4 time units older than the one in C. The markings satisfying these constraints can be described by the existential zone $Z = \left(2, \bar{P}, D\right)$ where $\bar{P}(1) = B$, $\bar{P}(2) = C$ and D is described by the following table where eg. $D(0, i) = 0$ and $D(2, 1) = 4$.*

$$D = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array}$$

Figure 3 shows a graphical illustration of the age restrictions of Z.

Consider a zone $Z = \left(m, \bar{P}, D\right)$, a marking $M = ((p_1, x_1), \ldots, (p_n, x_n))$ and an injection h :

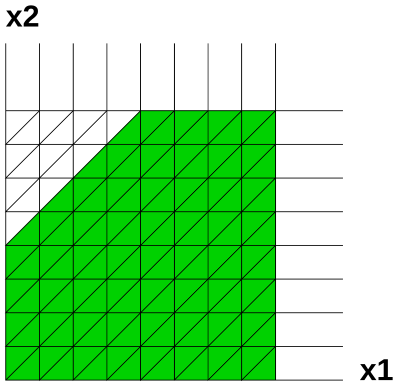


Figure 3: Example of restrictions on ages of tokens.

$\{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ which we call a *witness*. We say that M *satisfies* Z *with respect to* h, written $M, h \models Z$, if the following conditions are satisfied.

- $\bar{P}(i) = p_{h(i)}$, for each $i : 1 \leq i \leq m$.
- $x_{h(j)} - x_{h(i)} \leq D(j, i)$, for each $i, j \in \{1, \ldots, m\}$ with $i \neq j$.
- $x_{h(i)} \leq D(i, 0)$ and $-D(0, i) \leq x_{h(i)}$, for each $i \in \{1, \ldots, m\}$.

We say that M *satisfies* Z, written $M \models Z$ if there is a witness h such that $M, h \models Z$. Notice that if M satisfies Z then $m \leq n$ (since h is an injection), i.e., M has at least the number of tokens required by Z, and furthermore, the places and ages of the tokens satisfy the requirements of Z. We define $[\![Z]\!] = \{M \mid M \models Z\}$. Note also that the value of $D(i, i)$ is irrelevant for the satisfiability of Z.

From the above definitions the following lemma is straightforward.

Lemma 7.2. *For an existential zone Z and a marking M, it is decidable whether $M \models Z$.*

We observe that Z defines a number of minimal requirements on M, in the sense that M should contain at least m tokens whose places and ages are constrained by the functions $\bar{P}$ and D respectively. This means the set $[\![Z]\!]$ is upward closed since $M \models Z$ and $M \leq M'$ implies $M' \models Z$.

An existential zone $Z = \left(m, \bar{P}, D\right)$ is said to be *consistent* if $[\![Z]\!] \neq \emptyset$. We say that Z is *normal* if for each $i, j, k \in \{0, \ldots, m\}$, we have $D(i, j) \leq D(i, k) + D(k, j)$. It is easy to show the following.

Lemma 7.3. *For each existential zone Z there is a unique (up to renaming of the index set) normal existential zone, written $\widetilde{Z}$, such that $[\![\widetilde{Z}]\!] = [\![Z]\!]$.*

This means that we can assume without loss of generality that all existential zones we work with are normal.

7.1 Entailment

By the definitions in Section 3 we get that entailment between existential zones is defined by $Z_1 \preceq Z_2$ if $[\![Z_2]\!] \subseteq [\![Z_1]\!]$. We reduce checking entailment between zones into validity of formulae in a logic which we call *Difference Bound Logic (DBL)*. Atomic DBL-formulae are of either of the forms $v \leq c$ or $v - u \leq c$, where v and u are variables interpreted over $\mathbb{R}^{\geq 0}$ and $c \in \mathbb{N}$. Furthermore the set of formulae is closed under the propositional connectives. It is easy to see that validity of DBL-formulae is NP-complete.

Given two existential zones $Z_1 = (m_1, \bar{P}_1, D_1)$ and $Z_2 = (m_2, \bar{P}_2, D_2)$ we translate the relation $Z_1 \preceq Z_2$ into validity of a DBL-formula F. Intuitively, each token specified by Z_2 is represented by a free variable in F. The value of the variables represent the ages of the tokens. The idea is that the formula F is valid if and only if whenever the variables have values satisfying the age restrictions imposed by Z_2, there is a mapping h from indices of Z_1 to indeces of Z_2 such that token i in Z_1 reside in the same place as token $h(i)$ in Z_2, and the value of variable $v_{h(i)}$ satisfy the age restrictions that Z_1 imposes on token i.

We construct the formula F according to the following. We let $\{v_i \mid i \in \{1, \ldots, m_2\}\}$ be the set of free variables in F. We define a set H of witnesses, injections from $\{1, \ldots, m_1\}$ to $\{1, \ldots, m_2\}$, where a witness $h \in H$ if and only if $\bar{P}_1(i) = \bar{P}_2(h(i))$ for each $i \in \{1, \ldots, m_1\}$. We define $F = \left(F_1 \implies \left(\bigvee_{h \in H} F_2\right)\right)$ where $F_1 = F_{11} \wedge F_{12} \wedge F_{13}$ and $F_2 = F_{21} \wedge F_{22} \wedge F_{23}$ and

- $F_{11} = \bigwedge_{i,j \in \{1,\ldots,m_2\}, i \neq j} (v_i - v_j \leq D_2(i,j))$.
- $F_{12} = \bigwedge_{i \in \{1,\ldots,m_2\}} (v_i \leq D_2(i,0))$.
- $F_{13} = \bigwedge_{i \in \{1,\ldots,m_2\}} (-D_2(0,i) \leq v_i)$.
- $F_{21} = \bigwedge_{i,j \in \{1,\ldots,m_1\}, i \neq j} \left(v_{h(i)} - v_{h(j)} \leq D_1(i,j)\right)$.
- $F_{22} = \bigwedge_{i \in \{1,\ldots,m_1\}} \left(v_{h(i)} \leq D_1(i,0)\right)$.
- $F_{23} = \bigwedge_{i \in \{1,\ldots,m_1\}} \left(-D_1(0,i) \leq v_{h(i)}\right)$.

Example 7.4. *Consider the zones $Z_1 = (1, \bar{P}_2, D_1)$ and $Z_2 = (2, \bar{P}_2, D_2)$ where $\bar{P}_1(1) = \bar{P}_2(1) = \bar{P}_2(2) = C$ and D_1 and D_2 are described by the tables below.*

$$D_1 = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & - & 0 \\ 1 & 8 & - \end{array} \qquad D_2 = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & -2 \\ 1 & 5 & - & 3 \\ 2 & 8 & 4 & - \end{array}$$

The relation $Z_1 \preceq Z_2$ is equivalent to validity of the following formula

$$F = \left(\begin{array}{ccc} v_1 - v_2 \leq 3 \wedge v_2 - v_1 \leq 4 & & \\ \wedge & & v_1 \leq 8 \wedge 0 \leq v_1 \\ v_1 \leq 5 \wedge v_2 \leq 8 & \implies & \vee \\ \wedge & & v_2 \leq 8 \wedge 0 \leq v_2 \\ 0 \leq v_1 \wedge 2 \leq v_2 & & \end{array} \right)$$

We now have the following.

Lemma 7.5. *The entailment relation is decidable for existential zones.*

Notice that in contrast to zones for which entailment can be checked in polynomial time, the entailment relation for existential zones can be checked only in nondeterministic polynomial time (as we have to consider exponentially many witnesses). This is the price we pay for working with an unbounded number of clocks. On the other hand, when using zones, the size of the problem grows exponentially with the number of clocks inside the system.

7.2 Computing Predecessors

We define a function Pre such that for a zone Z, $Pre(Z)$ characterizes the set of markings from which we can reach a marking satisfying Z through the performance of a single discrete or timed transition. The value of $Pre(Z)$ is a finite set $\{Z_1, \ldots, Z_m\}$ of zones. In other words $[\![Pre(Z)]\!] = [\![Z_1]\!] \cup \cdots \cup [\![Z_m]\!]$. We define $Pre = Pre_D \cup Pre_\delta$, where Pre_D corresponds to firing transitions backwards and Pre_δ corresponds to running time backwards.

We define $Pre_D = \bigcup_{t \in T} Pre_t$, where Pre_t characterizes the effect of running the transition t backwards. To define Pre_t, we need the following operations on zones. In the rest of the section we assume a normal existential zone $Z = \left(m, \bar{P}, D\right)$, and a timed Petri net $\mathcal{N} = (P, T, In, Out)$. From Lemma 7.3 we know that assuming Z to be normal does not affect the generality of our results.

For an interval $\mathcal{I} = [a : b]$, and $i \in \{1, \ldots, m\}$, we define the *conjunction* $Z \otimes (\mathcal{I}, i)$ of Z with $\mathcal{I}$ at i to be the existential zone $Z' = \left(m, \bar{P}, D'\right)$, where

- $D'(i,0) = \min(b, D(i,0))$.
- $D'(0,i) = \min(-a, D(0,i))$.
- $D'(k,j) = D(k,j)$, for each $j,k \in \{1, \ldots, m\}$ with $k \neq j$, $(k,j) \neq (i,0)$, and $(k,j) \neq (0,i)$.

Intuitively, the operation adds an additional constraint on the age of token i, namely that its age should be in the interval $\mathcal{I}$. For example, for a zone

$$Z = \left(2, \bar{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array} \right)$$

the conjunction $Z \otimes ([1:6], 1)$ is the zone

$$\left(2, \bar{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & -1 & 0 \\ 1 & 6 & - & 8 \\ 2 & 8 & 4 & - \end{array} \right)$$

while the conjunction $Z \otimes ([0:10], 1) = Z$

For a place p and an interval $\mathcal{I} = [a:b]$, we define the *addition* $Z \oplus (p, \mathcal{I})$ of $(p, \mathcal{I})$ to Z to be the existential zone $Z' = \left(m+1, \bar{P}', D'\right)$, and

- $D'(m+1, 0) = b$, and $D'(0, m+1) = -a$.
- $D'(m+1, j) = \infty$, and $D'(j, m+1) = \infty$, for each $j \in \{1, \ldots, m\}$.
- $\bar{P}'(m+1) = p$.
- $D'(k, j) = D(k, j)$, for each $j, k \in \{0, \ldots, m\}$, and $\bar{P}'(j) = \bar{P}(j)$, for each $j \in \{1, \ldots, m\}$.

Intuitively, the new existential zone Z' requires one additional token to be present in place p such that the age of the token is in the interval $\mathcal{I}$. For example, for a zone

$$Z = \left(2, \begin{array}{l} \bar{P}(1) = B \\ \bar{P}(2) = C \end{array}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array}\right)$$

the addition $Z \oplus (A, [1:2])$ is the zone

$$\left(3, \begin{array}{l} \bar{P}(1) = B \\ \bar{P}(2) = C \\ \bar{P}(3) = A \end{array}, \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 8 & \infty \\ 2 & 8 & 4 & - & \infty \\ 3 & 2 & \infty & \infty & - \end{array}\right)$$

For $i \in \{1, \ldots, m\}$, we define the *abstraction* $Z \backslash i$ of i in Z to be the zone $Z' = \left(m-1, \bar{P}', D'\right)$, where

- $D'(j, k) = D(j, k)$, for each $j, k \in \{0, \ldots, (i-1)\}$.
- $D'(j, k) = D(j, k+1)$ and $D'(k, j) = D(k+1, j)$, for each $j \in \{0, \ldots, (i-1)\}$ and $k \in \{i, \ldots, m-1\}$.
- $D'(j, k) = D(j+1, k+1)$, for each $j, k \in \{i, \ldots, m-1\}$.
- $\bar{P}'(j) = \bar{P}(j)$, for each $j \in \{0, \ldots, (i-1)\}$, and $\bar{P}'(j) = \bar{P}(j+1)$, for $j \in \{i, \ldots, m-1\}$.

Intuitively, the operation removes all constraints related to token i from Z, so the number of required tokens is reduced by 1 and the restrictions related to the age and place of the token disappear. For example, for a zone

$$Z = \left(3, \begin{array}{l} \bar{P}(1) = B \\ \bar{P}(2) = C \\ \bar{P}(3) = A \end{array}, \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 6 & 7 \\ 2 & 8 & 4 & - & 7 \\ 3 & 2 & 2 & 2 & - \end{array}\right)$$

the abstraction $Z\backslash 2$ is the zone

$$\left(2, \begin{array}{l} \bar{P}(1) = B \\ \bar{P}(2) = A \end{array}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & -1 \\ 1 & 8 & - & 7 \\ 2 & 2 & 2 & - \end{array} \right)$$

Notice that the existential zones we obtain as a result of performing the three operations above need not be normal.

We are now ready to define *Pre*.

Lemma 7.6. *Consider a TPN $\mathcal{N} = (P, T, In, Out)$, a transition $t \in T$, and an existential zone $Z = (m, \bar{P}, D)$. Let $In(t) = ((p_1, \mathcal{I}_1), \ldots, (p_k, \mathcal{I}_k))$, and $Out(t) = ((q_1, \mathcal{J}_1), \ldots, (q_\ell, \mathcal{J}_\ell))$. Then $Pre_t(Z)$ is the smallest set containing each existential zone Z' such that there is a partial injection $h : \{1, \ldots, m\} \longrightarrow \{1, \ldots, \ell\}$ with a domain $\{i_1, \ldots, i_n\}$, and an existential zone Z_1 satisfying the following conditions.*

- $\bar{P}(i_j) = q_{h(i_j)}$, *for each* $j \in \{1, \ldots, n\}$
- $Z \otimes (\mathcal{J}_{h(i_1)}, i_1) \otimes \cdots \otimes (\mathcal{J}_{h(i_n)}, i_n)$ *is consistent.*
- $Z_1 = Z\backslash i_1 \backslash \cdots \backslash i_n$.
- $Z' = Z_1 \oplus (p_1, \mathcal{I}_1) \oplus \cdots \oplus (p_k, \mathcal{I}_k)$.

Lemma 7.7. *For an existential zone $Z = (m, \bar{P}, D)$, the set $Pre_\delta(Z)$ is the existential zone $Z' = (m, \bar{P}, D')$, where $D'(0, i) = 0$ and $D'(j, i) = D(j, i)$ if $j \neq 0$, for each $i, j \in \{0, \ldots, m\}$, with $i \neq j$.*

From Lemma 7.6 and Lemma 7.7 we get the following.

Lemma 7.8. *For an existential zone Z, the set $Pre(Z)$ is computable.*

7.3 Termination

As stated in section 3, the reachability algorithm is guaranteed to terminate if the constraint system, existential zones, is well quasi-ordered under entailment. To prove well quasi-ordering of zones we prove better quasi-ordering using the methodology presented in Section 5. Since each better quasi-order is a well quasi-order Theorem 5.2 (Property 1) it then follows that zones are well quasi-ordered and the algorithm terminates.

In order to prove that existential zones are better quasi-ordered we recall a constraint system related to existential zones, namely that of *existential regions* introduced in [AJ98]. An existential region is a list of multisets $(\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_n, \mathcal{M}_{n+1})$ where $n \geq 0$ and $\mathcal{M}_i$ is a multiset over $P \times \mathbb{N}$. In a similar manner to existential zones, an existential region R defines a set of conditions which should be satisfied by a marking M in order for M to satisfy R. Intuitively $\mathcal{M}_0$ represents tokens with ages which have fractional parts equal to 0. The multisets $\mathcal{M}_1, \ldots, \mathcal{M}_n$ represent tokens whose ages have increasing fractional parts where ages of tokens belonging to the same multiset have the same fractional part and ages of tokens belonging to $\mathcal{M}_i$ have a fractional part that is strictly less than the fractional part of the ages of those in $\mathcal{M}_{i+1}$. Finally the multiset $\mathcal{M}_{n+1}$ represents tokens with ages greater than the maximum natural number occuring in the enabling conditions of a given TPN (regardless of their fractional parts).

We show better quasi-ordering of existential zones by showing better quasi-ordering of existential regions and then .... that each zone is equal to the disjunction of a finite number of existential regions. By Theorem 5.2 (Property 5) we have that better quasi-ordering is closed under disjunction, hence existential zones are better quasi-ordered.

Lemma 7.9. *Existential zones are better quasi-ordered (and hence well quasi-ordered).*

1. *Existential regions are built starting from finite domains, and repeatedly building finite strings, bags, and sets. From the properties mentioned above, it follows that existential regions are better quasi-ordered.*

2. *For each existential zone Z, there is a finite set $\boldsymbol{R}$ of existential regions such that $Z \equiv \bigvee_{R \in \boldsymbol{R}} R$. Since better quasi-ordering is closed under disjunction, it follows that existential zones are better quasi-ordered.*

8 Experimental Results

We have implemented a prototype to perform reachability (coverability) analysis for timed Petri nets. In our experimentation we use a constraint system called existential DDDs which is described below. The implementation is based on a DDD package developed at Technical University of Denmark [ML98]. We have used the tool to verify a parameterized version of Fischer's protocol [SBM92].

8.1 Existential CDDs and DDDs

CDDs [BLP+99] and DDDs [MLAH99] are constraint systems that have been invented to give representations of real-time systems which are more compact than zones. In the same manner as zones were modified into existential zones, we modify the definitions of CDDs (DDDs) into *existential CDDs (DDDs)* to make them suitable for verifying systems with an unbounded number of clocks. Below we give the definition of existential DDDs. The definition of existential CDDs can be stated in a similar manner.

An *existential DDD* Y is a tuple $\left(m, \bar{P}, \mathtt{V}, \mathtt{E}\right)$, where m is a natural number denoting the minimum number of tokens in a marking satisfying Y and the placing $\bar{P}$ maps each token to a place in the same manner as in an existential zone (Section 7). $(\mathtt{V}, \mathtt{E})$ is a finite directed acyclic graph where $\mathtt{V}$ is the set of vertices and $\mathtt{E}$ is the set of edges. The set $\mathtt{V}$ contains two special elements $\mathtt{v}^0$ and $\mathtt{v}^1$. The outdegrees of $\mathtt{v}^0$ and $\mathtt{v}^1$ are zero while the outdegrees of the rest of vertices are two. Each vertex $\mathtt{v} \in \mathtt{V} \setminus \{\mathtt{v}^0, \mathtt{v}^1\}$ has the following attributes: $pos(\mathtt{v}), neg(\mathtt{v}) \in \{0, \dots, m\}$, $op(\mathtt{v}) \in \{<, \leq\}$, $const(\mathtt{v}) \in \mathbb{Z}$, and $high(\mathtt{v}), low(\mathtt{v}) \in \mathtt{V}$. The set $\mathtt{E}$ contains the edges $(\mathtt{v}, low(\mathtt{v}))$ and $(\mathtt{v}, high(\mathtt{v}))$, where $\mathtt{v} \in \mathtt{V} - \{\mathtt{v}^0, \mathtt{v}^1\}$. In a similar manner to BDDs, the internal nodes of Y correspond to the if-then-else operator $\phi \rightarrow \phi_1, \phi_2$, defined as $(\phi \wedge \phi_1) \vee (\neg\phi \wedge \phi_2)$. Intuitively, the attributes of the node represent the DBL-formula $\phi = x_{pos(\mathtt{v})} - x_{neg(\mathtt{v})}\ op(\mathtt{v})\ const(\mathtt{v})$, and $high(\mathtt{v})$ and $low(\mathtt{v})$ are children of $\mathtt{v}$ corresponding to ϕ_1 and ϕ_2 respectively. The special vertices $\mathtt{v}^0$ and $\mathtt{v}^1$ correspond to *false* and *true*.

Consider an existential DDD $Y = (m, \bar{P}, \mathtt{V}, \mathtt{E})$, a vertex $\mathtt{v} \in \mathtt{V}$, a marking $M = ((p_1, x_1), \dots, (p_n, x_n))$ and an injection $h : \{1, \dots, m\} \rightarrow \{1, \dots, n\}$. We say that M *satisfies* Y *at* $\mathtt{v}$ *with respect to* h, written $M, h \models (Y, \mathtt{v})$, if $\bar{P}(i) = p_{h(i)}$, for each $i \in \{1, \dots, m\}$, and either

- $\mathtt{v} = \mathtt{v}^1$; or
- $\left(\begin{pmatrix} x_{h(pos(\mathtt{v}))} \\ - \\ x_{h(neg(\mathtt{v}))} \end{pmatrix} \sim const(\mathtt{v}) \right) \rightarrow \begin{pmatrix} M, h \models (Y, high(\mathtt{v})) \\ , \\ M, h \models (Y, low(\mathtt{v})) \end{pmatrix},$

  where $\sim = op(\mathtt{v})$.

In a similar manner to existential zones, we can modify the operations defined in [MLAH99] to compute predecessors of existential DDDs with respect to transitions of a TPN. To check entailment we must, as we did for existential zones, take into consideration all variable permutations.

For each existential DDD Y there is a finite set $\boldsymbol{Z}$ of existential zones such that $[\![Y]\!] = [\![\bigvee \boldsymbol{Z}]\!]$. Intuitively this means that an existential DDD can replace several existential zones, and hence existential DDDs give a more compact (efficient) representation of sets of states. Note that each existential DDD is a disjunction of existential zones. This together with Lemma 7.9 and Theorem 5.2 (Property 5) gives us the following result.

Lemma 8.1. *Existential DDDs are better quasi-ordered (and hence also well quasi-ordered).*

8.2 Fischer's Protocol

The purpose of Fischer's protocol is to guarantee mutual exclusion in a concurrent system consisting of an arbitrary number of processes. The example was suggested by Schneider et al. [SBM92]. The protocol analyzed here is in fact a weakened version of Fischer's protocol but since the set of reachable states of the weakened version is a superset of the reachable states of the original protocol, the results of our analysis are still valid.

The protocol consists of each process running the code which is graphically described in Figure 4. Each process i has a local clock, x_i, and a control state, which assumes values in the set $\{A, B, C, CS\}$ where A is the initial state and CS is the critical section. The processes read from and write to a shared variable v, whose value is either $\perp$ or the index of one of the processes.

All processes start in state A. If the value of the shared variable is $\perp$, a process wishing to enter the critical section can proceed to state B and reset its local clock. From state B, the process can

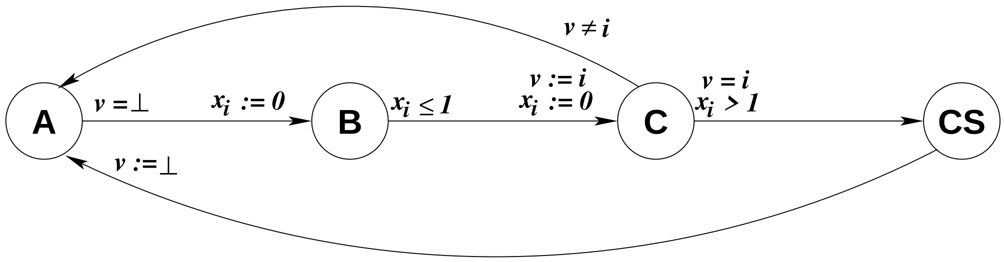


Figure 4: Fischer's Protocol for Mutual Exclusion

proceed to state C within one time unit or get stuck in B forever. When making the transition from B to C, the process resets its local clock and sets the value of the shared variable to its own index. The process now has to wait in state C for more than one time unit, a period of time which is strictly greater than the one used in the timeout of state B. If the value of the shared variable is still the index of the process, the process may enter the critical section, otherwise it may return to state A and start over again. When exiting the critical section, the process resets the shared variable to $\perp$.

We will now make a model of the protocol in our TPN formalism. The processes running the protocol are modeled by tokens in the places A, B, C, CS, $A^\dagger$, $B^\dagger$, $C^\dagger$ and $CS^\dagger$. The places marked with † represent that the value of the shared variable is the index of the process modeled by the token in that place. We use a place *udf* to represent that the value of the shared variable is $\perp$. A straightforward translation of the description in Figure 4 yields the Petri net model in Figure 5. q is used to denote an arbitrary process state.

8.3 Results

In order to prove mutual exclusion we examine the reachability of the existential zones stating that at least two processes are in the critical section, i.e., the following zones:

- $Z_1 = \big(2, \bar{P}_1, D\big)$ where $\bar{P}_1(1) = \bar{P}_1(2) = CS$
- $Z_2 = \big(2, \bar{P}_2, D\big)$ where $\bar{P}_2(1) = CS$ and $\bar{P}_2(2) = CS^\dagger$
- $Z_3 = \big(2, \bar{P}_3, D\big)$ where $\bar{P}_3(1) = \bar{P}_3(2) = CS^\dagger$

For all three zones $D(0,i) = 0$, $D(i,j) = \infty$ for $i \neq j$.

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.

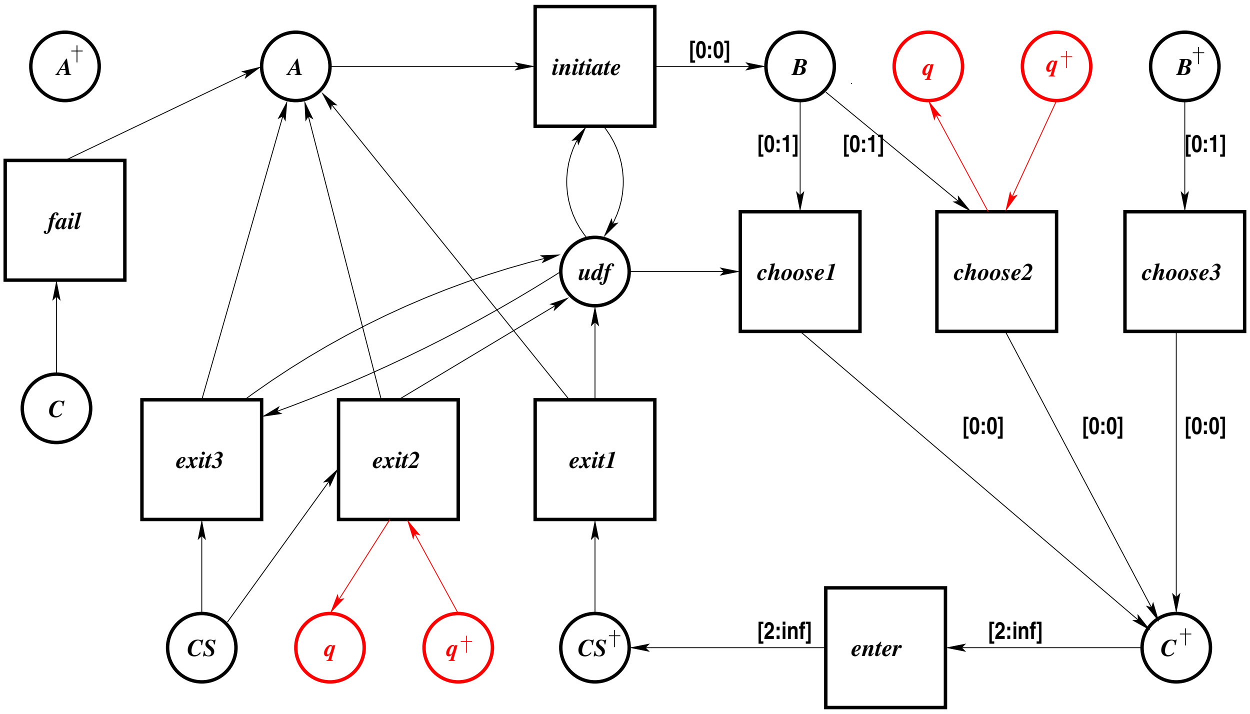


Figure 5: TPN model of Fischer's Protocol for Mutual Exclusion

9 Broadcast Protocols

A *broadcast protocol* consists 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 $\{x_1, \ldots, x_m\}$ of variables ranging over the natural numbers. A global state of a protocol is called a *configuration*. A configuration γ 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. [EFM99] defines a constraint system, which we here refer to as $\mathbb{B}$, 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 4. The most general of these constraint systems is called **AD** in [DEP99] and 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 4), a separate proof of termination is required for them.

Applying the method of Section 5 we can show better quasi-ordering of **AD**, **NA**, and **DV** uniformly as follows. From properties 2 and 3 in Theorem 5.2 it follows that $\mathbb{B}$ is better quasi-ordered. 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 5.2 we get

Theorem 9.1. **AD**, **NA**, *and* **DV** *are better quasi-ordered.*

In fact we can derive the better quasi-ordering property for a more general constraint system than **AD**, namely that consisting of basic constraints of the form $a_1 x_1 + \cdots + a_k x_k \geq b$ combined through conjunction and disjunction.

10 Lossy Channel Systems

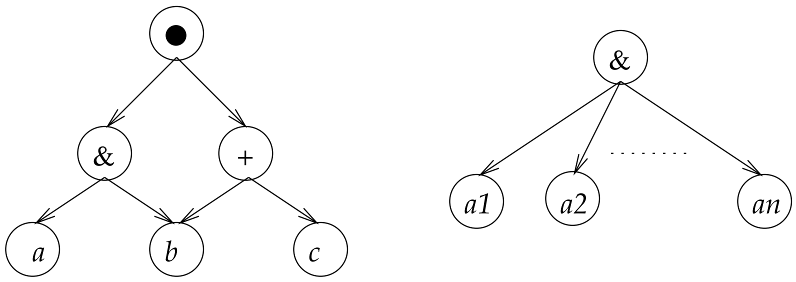
A *lossy channel system* is a finite state machine communicating over unbounded and unreliable FIFO buffers. In [AJ96] we present a constraint system, here denoted $\mathbb{L}_1$, for verification of such systems. The constraints in $\mathbb{L}_1$ represent upward closed sets of strings over a finite 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 continuous) substring of w_2. A constraint in $\mathbb{L}_1$ is represented by a string $w \in \Sigma^*$ 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:

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

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

- concatenation:



$\phi1$ $\phi2$

Figure 6: Two constraints in $\mathbb{L}_2$

.

$[\![\phi_1 \bullet \phi_2]\!] = \{w_1 w_2 \mid w_1 \in [\![\phi_1]\!] \text{ and } w_2 \in [\![\phi_2]\!]\}$

- conjunction:

  $[\![\phi_1 \& \phi_2]\!] = \{w \mid w \in [\![\phi_1]\!] \text{ and } w \in [\![\phi_2]\!]\}$

- disjunction:

  $[\![\phi_1 + \phi_2]\!] = \{w \mid w \in [\![\phi_1]\!] \text{ or } w \in [\![\phi_2]\!]\}$

Example 10.1. *ϕ_1 in Figure 6 is the constraint $(a \ \& \ b) \bullet (b + c)$. This means that $[\![\phi_1]\!] = \{w_1 w_2 \mid (a \preceq w_1)$ and $(b \preceq w_1)$ and $((b \preceq w_2)$ or $(c \preceq w_2))\}$. The constraint ϕ_1 is equivalent to the disjunction of the following set of constraints in $\mathbb{L}_1$: $\{abb, abc, bab, bac\}$.*

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 6) 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 7 and Section 9 we can use properties of $\mathbb{L}_1$ and $\mathbb{L}_2$ to conclude the following

Theorem 10.2. *$\mathbb{L}_2$ is better quasi-ordered.*

11 Integral Relational Automata

An *Integral Relational Automaton (IRA)* operates on a set $X = \{x_1, \ldots, x_n\}$ of *variables* assuming values from the set $\mathbb{Z}$ of integers. The transitions of the automaton are labeled by guarded commands of the form $g \to stmt$ 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 $c \in \mathbb{Z}$ 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 $c \in \mathbb{Z}$. The assignment $x := \{?\}$ is a "read" operation putting an arbitrary integer into the variable x. A *state* s of an IRA is a mapping from X to $\mathbb{Z}$. Sometimes, we write s as a tuple $(s(x_1), \ldots, s(x_n))$. For $c \in \mathbb{Z}$ we use the convention that $s(c) = c$.

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 $\mathbb{Z}$. In a similar manner to states we assume $\phi(c) = c$ for $c \in \mathbb{Z}$. A state s satisfies ϕ if and only if for each $x, y \in X \cup C$, we have (i) $s(x) \leq s(y)$ if and only if $\phi(x) \leq \phi(y)$ and (ii) if $\phi(x) \leq \phi(y)$ then $\phi(y) - \phi(x) \leq s(y) - s(x)$.

Example 11.1. *Assume $X = \{x_1, x_2, x_3\}$ and $C = \{5\}$. Consider a constraint $\phi = (10, 5, 12)$, then $s_1 = (12, 5, 17) \in [\![\phi]\!]$ while $s_2 = (8, 5, 16) \notin [\![\phi]\!]$ (since $\phi(x_1) - \phi(x_2) = 5 \not\leq s_2(x_1) - s_2(x_2) = 3$) and $s_3 = (12, 4, 17) \notin [\![\phi]\!]$ (since $\phi(5) = 5 \leq \phi(x_2) = 5$ while $s_3(5) = 5 \not\leq s_3(x_2) = 4$).*

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 $c \in \mathbb{Z}$. The satisfiability of ϕ by a state s is defined in the obvious way.

Example 11.2. *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.*

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 11.3. *$\mathbb{S}_2$ is better quasi-ordered.*

12 Conclusions and Future Work

We have proposed *better quasi-orderings*, a refinement of the theory of *well quasi-ordering*, as a 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 and for 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.

We have introduced a new constraint system, *existential zones* for verification of real-time systems with an unbounded number of clocks. Using and modifying efficient data structures for verification of real-time automata, we have obtained some encouraging experimental results. One direction for future work is to design efficient data structures for manipulating the new constraint systems. It would also be interesting to investigate the feasibility of defining a general framework for implementation of better quasi-ordered constraint systems.

Furthermore, in addition to disjunction, better quasi-orderings are closed under several other operations which do not preserve well quasi-ordering. An example is that better quasi-orderings 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.

Acknowledgments

We are grateful to Alberto Marcone and Petr Jančar for many interesting discussions on the theory of better quasi-orderings. 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.

This work is supported by TFR – The Swedish Research Council for Engineering Sciences.

References

[AČJYK96] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. LICS' 96* 11^{th} *IEEE Int. Symp. on Logic in Computer Science*, pages 313–321, 1996.

[AČJYK00] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160:109–127, 2000.

[AD90] R. Alur and D. Dill. Automata for modelling real-time systems. In *Proc. ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335, 1990.

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

[AJ98] 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.

[AJ01a] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256(1-2):145–167, 2001.

[AJ01b] Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes, 2001. To appear in the journal of Theoretical Science.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Engineering*, 17(3):259–273, 1991.

[BLP+99] Gerd Behrmann, Kim G. Larsen, Justin Pearson, Carsten Weise, and Wang Yi. Efficient Timed Reachability Analysis Using Clock Difference Diagrams. In Nicolas Halbwachs and Doron Peled, editors, *Proc.* 11^{th} *Int. Conf. on Computer Aided Verification*, number 1633 in Lecture Notes in Computer Science, pages 341–353. Springer Verlag, July 1999.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In Abiteboul and Shamir, editors, *Proc. ICALP '94,* 21^{st} *International Colloquium on Automata, Lnaguages, and Programming*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer Verlag, 1994.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specification. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, April 1986.

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

[dFERA00] David de Frutos Escrig, Valentín Valero Ruiz, and Olga Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *Proc.* 21^{th} *Int. Conf. on Application and Theory of Petri Nets*, volume 1825 of *Lecture Notes in Computer Science*, pages 187–206. Springer Verlag, 2000.

[Dil89] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite-State Systems*, volume 407 of *Lecture Notes in Computer Science.* Springer Verlag, 1989.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. LICS' 99* 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.

[GMMP91] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzè. A unified high-level Petri net formalism for time-critical systems. *IEEE Trans. on Software Engineering*, 17(2):160–172, 1991.

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, 1993.

[Hig52] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 2:326–336, 1952.

[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.

[LPY95] Kim G. Larsen, Paul Pettersson, and Wang Yi. Model-Checking for Real-Time Systems. In *Proc. of Fundamentals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 62–88, August 1995.

[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. D. Reidel Publishing Company, 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.

[NW65] C. Nash-Williams. On well quasi-ordering infinite trees. *Proc. Cambridge Phil. Soc.*, 61:697–720, 1965.

[Pou85] M. Pouzet. Applications of well quasi-orderings and better quasi-orderings. In I. Rival, editor, *Graphs and Orders*, pages 503–519. D. Reidel Publishing Company, 1985.

[RGdFE99] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proceedings of the 8th Int. Workshop on Petri Net and Performance Models (PNPM'99)*, pages 188–196, 1999.

[RP85] R. Razouk and C. Phelps. Performance analysis using timed Petri nets. In *Protocol Testing, Specification, and Verification*, pages 561–576, 1985.

[SBM92] F. B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. In de Bakker, Huizing, de Roever, and Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, 1992.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. LICS '86, 1^{st} IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

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