BQOs and Timed Petri Nets

Parosh Aziz Abdulla Aletta Nylén
Department of Computer Systems
Uppsala University
Sweden
{parosh, aletta}@docs.uu.se

Abstract

In this paper, we use the theory of *better quasi-orderings* to define a methodology for inventing constraint systems which are both well quasi-ordered and compact. We apply our methodology by presenting new constraint systems for verification of systems with unboundedly many real-valued clocks, and use them for checking safety properties for lazy (non-urgent) timed Petri nets where each token is equipped with a real-valued clock.

1 Introduction

Model checking of infinite-state systems has drawn much attention in the last few years. A program may be infinite-state since it operates on unbounded data structures, e.g. timed automata [ACD90], hybrid automata [Hen95], data-independent systems [JP93, Wol86], relational automata [Čer94], counter machines [Jan97, AČ98], pushdown processes [BS95], lossy channel systems [AJ96], completely specified protocols [Fin94] etc. A program may also be infinite-state since it has an infinite control part, e.g. Petri nets [Esp95, JM95], and parameterized systems [GS92, AJ98b, KMM+97], in which the topology of the system is parameterized by the number of processes inside the system.

In this paper we define a new methodology, based on the theory of *better quasi-orderings*, for efficient verification of infinite-state systems. We consider a standard algorithm for performing symbolic backward reachability analysis. The algorithm uses a symbolic representation, where *constraints* are employed to represent (potentially infinite) sets of states. We start from the set of final states (the states which we want to check for reachability), and then generate a sequence of constraints characterizing sets of states from which we can reach a final state in j or less steps, for increasing values of j. The algorithm terminates when we reach a point j where the constraints at steps j and $j+1$ are equivalent.

In [AČJYK96] we show that termination of the above algorithm is equivalent to well quasi-ordering of the set of constraints under entailment, i.e., for each infinite sequence $\phi_0, \phi_1, \phi_2, \ldots$ of constraints, there are i and j such that $i < j$ and ϕ_i is entailed by ϕ_j. Thus, a crucial part in applying our method for new classes of systems is to define a tool for generating new well quasi-ordered constraint systems. In [AJ98a] we define a methodology, based on Higman's theorem [Hig52], for deriving new well quasi-ordered constraint systems from existing ones. However, several of the constraint systems constructed according to this method, suffer from a "constraint explosion problem", since a large number of constraints is generated during each step of the reachability algorithm.In this paper we consider constraint systems which satisfy a stronger property than well quasi-ordering, namely that of *better quasi-ordering*.

Working with better quasi-orderings offers two advantages: (i) better quasi-ordering implies well quasi-ordering, and hence all the verification algorithms described in [AČJYK96, AJ98a] are still applicable; and (ii) better quasi-orderings are more "robust" than well quasi-orderings, i.e., they are preserved by a larger set of operations. For example, better quasi-ordered constraint systems are closed under the operation of taking finite unions: if a set of "basic constraints" is better quasi-ordered under entailment, then finite unions of "basic constraints" will also be better quasi-ordered under entailment. However, well quasi-ordering is not preserved by the same operation. This operation is important for verification, since it allows to work with more compact (and therefore efficient) constraint systems. For instance, it is well-known from the theory of real-time automata that *zones*, which represent finite sets of *regions*, offer a much more compact representation than *regions*.

We illustrate our methodology by considering verification of real-time sys-

tems with unboundedly many clocks, using a new constraint system, which we call *existential zones*. Existential zones are variants of *zones* used in verification of real-time automata, and are generalizations of the constraints used in [AJ98b] for verification of *timed networks*. Timed networks consist of an arbitrary number of timed processes, and hence contain an unbounded number of clocks. Therefore, they cannot be modelled within the context of real-time automata [AD90], which operate on a finite number of clocks. Instead of letting a region represent states with equivalent behaviours, we use regions to describe minimal required behaviours, so a region denotes the set of all states which can at least simulate the behaviours characterized by the region. Typically, a region states that there *exists* at least two clocks where the values of one clock is equal to 1 while the value of the other clock lies in the interval $(2, 3)$. In the sequel we use *existential regions* to denote regions which are interpreted in the above manner. While in the case of timed automata, termination is guaranteed by finiteness of the number of regions, there is in general no bound on the number of existential regions which can appear during the analysis of timed networks. In [AJ98b] we show termination by proving that existential regions are well quasi-ordered under entailment.

The main problem with regions (and existential regions) is the "constraint space" explosion which occurs since the number of regions increases exponentially with the number of clocks and sizes of constants appearing in the transitions of the system. To avoid this, existing tools for verification of real-time systems, such as KRONOS [Yov97] and UPPAAL [LPY97], employ another constraint system, namely that of *zones*. Each zone is the union of a *set* of regions, yielding a more compact constraint system.

Therefore, we are interested in a similar generalization of existential regions to existential zones. Although existential regions are well quasi-ordered [AJ98b], and each existential zone is a finite union of existential regions, we cannot use the methodology of [AČJYK96, AJ98a] to derive well quasi-ordering of existential zones. This is due to the fact that well quasi-orderings are not closed under the operation of taking finite unions. Instead, we prove that existential regions satisfy the stronger property of being *better quasi-ordered*, and use the fact the better quasi-orderings are closed under taking finite unions, to conclude that existential zones are better quasi-ordered (and hence also well quasi-ordered).

Existential zones can be used in verification of several classes of systems operating on unbounded numbers of clocks. In this paper we show how to use our method for deciding safety properties for lazy (non-urgent) timed Petri nets, where each token is equipped with a clock representing the "age" of the token. This decidability result has not previously appeared in the literature.

We extend the results further and consider "existential variants" of CDDs [LPWY99] and DDDs [MLAH99], constraint systems which have recently appeared in the literature and which give even more compact representations than zones. Better quasi-orderings of existential CDDs (DDDs) follows from the fact that each existential CDD (DDD) is a finite union of existential zones.

To our knowledge this is the first time the theory of better quasi-orderings is used to prove termination when performing symbolic verification.

Outline In Section 2 we describe a general symbolic algorithm for verification of infinite-state systems. In Section 3 we define our methodology using results from the theories of well- and better quasi-orderings. In Section 4 we introduce existential zones. In Section 5 we describe existential regions and some of their properties which lead to well quasi-ordering of existential zones. In Section 6 we show how to check entailment between existential zones. In Section 7 we introduce timed Petri nets. In Section 8 we show how to compute the set of predecessors of an existential zone with respect to the set of transitions in a timed Petri net. In Section 9 we describe existential CDDs and DDDs. In Section 10 we present experimental results from using the method to prove properties of a timed system.

2 Overview of the Verification Algorithm

In this section, we give an overview of a standard symbolic reachability algorithm. We consider a transition system $(\Gamma, \gamma_{init}, \longrightarrow)$, where Γ is a potentially infinite set of *configurations (global states)*, γ_{init} is the *initial configuration*, and $\longrightarrow$ is a binary relation on Γ. We use γ to range over configurations, and use $\stackrel{*}{\longrightarrow}$ to denote the transitive closure of $\longrightarrow$. We perform a standard symbolic backward reachability analysis, using *constraint systems* for repre-

senting sets of configurations. A *constraint system* $\mathbb{C}$ consists of a set[1] $\mathbb{C}$ of *constraints*, and an interpretation function $[\![\cdot]\!]$, where, for each constraint ϕ, $[\![\phi]\!]$ denotes a (possibly infinite) subset of Γ. A finite set $\Phi \subseteq \mathbb{C}$ of constraints denotes the union of the denotations of its elements, i.e., $[\![\Phi]\!] = \bigcup_{\phi \in \Phi} [\![\phi]\!]$. We define an *entailment relation* $\preceq$ on constraints, where $\phi_1 \preceq \phi_2$ if and only if $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$. We want to solve a *reachability problem*, where, given a constraint ϕ_F, we ask the question of whether there is a $\gamma_F \in [\![\phi_F]\!]$ such that $\gamma_{init} \xrightarrow{*} \gamma_F$. We perform a backward reachability analysis as follows. We assume[2] that for each constraint ϕ there is a finite set of constraints, denoted $Pre(\phi)$, such that $[\![Pre(\phi)]\!] = \{\gamma';\ \gamma' \longrightarrow \gamma\}$. This implies that a similar set $Pre(\Phi)$ can be defined for a finite set of constraints Φ. We generate a sequence $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots$, of finite sets of constraints where $\Phi_0 = \{\phi_F\}$ and $\Phi_{j+1} = \Phi_j \cup Pre(\Phi_j)$. Notice that Φ_j denotes the set of configurations from which a configuration in $[\![\phi_F]\!]$ is reachable in j or less steps. The algorithm terminates when we reach point j where for each $\phi \in \Phi_{j+1}$ there is a $\phi' \in \Phi_j$ with $\phi' \preceq \phi$. This implies that $[\![\Phi_{j+1}]\!] \subseteq [\![\Phi_j]\!]$ (and hence $[\![\Phi_{j+1}]\!] = [\![\Phi_j]\!]$). In such a case, $[\![\Phi_j]\!]$ is exactly the set of configurations from which a configuration ϕ_F is reachable, and therefore the answer to reachability question is equivalent to whether $\gamma_{init} \in [\![\Phi_j]\!]$. We observe that, in order to be able to apply the algorithm for a certain class of systems, the constraint system $\mathbb{C}$ should

1. allow to compute $Pre(\phi)$ for any constraint ϕ,
2. allow checking membership of a configuration in the denotation of a constraint,
3. allow checking entailment between constraints (this enables us to decide at each step, whether we can terminate at that step), and
4. guarantee termination, i.e., we always reach a point j where the termination condition is satisfied

Notice that only property 1 is dependent on the transition relation $\longrightarrow$, while properties 2-4 depend only on the definition of the constraint system.

[1] We abuse notation and use $\mathbb{C}$ both to represent the constraint system and the set of constraints.

[2] This condition is satisfied by all the constraint systems we consider in this paper.

In Section 3 (Lemma 3.2) we show that the termination property in 4 is equivalent to well quasi-ordering of the set of constraints under entailment. A potential factor which may make a constraint system infeasible to use in practise is that the size of $Pre(\phi)$ is large for a given constraint ϕ, causing a "constraint explosion problem". In this paper, we define (Section 3) a new methodology based on the theory of *better quasi-orderings* to invent new constraint systems which are both well quasi-ordered and compact (in the sense that the size of $Pre(\phi)$ is small). As an application we consider constraints such as *existential zones* (Section 4) and existential CDDs and DDDs (Section 9), and use them to prove decidability of safety properties for lazy (non-urgent) timed Petri nets. We show (Section 3) that these constraint systems cannot be handled by existing methods for verification of infinite-state systems.

3 Well and Better Quasi-Orderings

In this section, we show some results from the theories of wqo (well quasi-orderings) and bqo (better quasi-orderings), and explain their relation to the reachability algorithm of Section 2.

A *quasi-ordering*, or a *qo* for short, is a pair $(A, \preceq)$, where $\preceq$ is a reflexive and transitive (binary) relation on a set A. We use $a_1 \equiv a_2$ to denote that $a_1 \preceq a_2$ and $a_2 \preceq a_1$. An infinite sequence $a_1, a_2, a_3, \ldots$ of elements of A is called a *bad sequence* iff $\forall i, j : i < j \Rightarrow a_i \not\preceq a_j$. A qo $(A, \preceq)$ is a *well quasi-ordering*, or a *wqo* for short, if there is no bad sequence of elements of A. Given a qo $(A, \preceq)$, we define a qo $(A^*, \preceq^*)$ on the set A^* of finite strings over A, such that $x_1 \bullet \cdots x_m \preceq^* y_1 \bullet \cdots \bullet y_n$ if and only if there is a strictly monotone injection $h : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ where $x_i \preceq y_{h(i)}$ for $i : 1 \leq i \leq m$. We define the relation $\sqsubseteq$ on the set $\mathcal{P}(A)$ of subsets of A, so that $A_1 \sqsubseteq A_2$ if and only if $\forall b \in A_2 : \exists a \in A_1 : a \preceq b$.

Relating to constraints (Section 2), we let A denote the set of constrains, and let $\preceq$ denote entailment, that is $\phi_1 \preceq \phi_2$ if and only if $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$. For a constraint system $\mathbb{C}$, we say that $\mathbb{C}$ is wqo, if $(\mathbb{C}, \preceq)$ is wqo. We say that $\mathbb{C}$ is *canonical* if $\phi_1 \not\preceq \phi, \ldots, \phi_n \not\preceq \phi$ implies $[\![\phi]\!] \not\subseteq [\![\{\phi_1, \ldots, \phi_n\}]\!]$, for any constraints $\phi, \phi_1, \ldots, \phi_n \in \mathbb{C}$. Examples of canonical constraint systems are existential regions $\mathbb{R}$ (Section 5) and the constraint system $\mathbb{C}_1$ in Example 3.3

below. Examples of non-canonical constraint systems are existential zones $\mathbb{Z}$ (Section 4) and existential DDDs $\mathbb{D}$ (Section 9).

For constraint systems $\mathbb{C}_1 = (\mathbb{C}_1, [\![.]\!])$ and $\mathbb{C}_2 = (\mathbb{C}_2, [\![.]\!])$, we say that $\mathbb{C}_2$ is *expressible* in $\mathbb{C}_1$, denoted $\mathbb{C}_1 \lhd \mathbb{C}_2$, if, for each constraint $\psi \in \mathbb{C}_2$, there is a finite set ψ^* of constraints in $\mathbb{C}_1$ such that $[\![\psi]\!] = [\![\psi^*]\!]$. For example, from the literature of real-time automata, we know that *zones* are expressible in *regions*. The relation $\lhd$ is important from the point of view of symbolic verification, since $\mathbb{C}_1 \lhd \mathbb{C}_2$ intuitively means that a constraint in $\mathbb{C}_2$ can "replace" several constraints in $\mathbb{C}_1$, and hence $\mathbb{C}_2$ gives a more compact (efficient) representation of sets of states. In this paper we consider e.g., *existential zones* (Section 4), and prove that they are expressible in *existential regions* (Section 5), and then consider *existential CDDs* and *existential DDDs* which are expressible in existential zones (Section 9).

Proposition 3.1 For constraint systems $\mathbb{C}_1$ and $\mathbb{C}_2$, if $\mathbb{C}_1 \lhd \mathbb{C}_2$, then ${\psi_1}^* \sqsubseteq {\psi_2}^*$ implies $[\![\psi_2]\!] \subseteq [\![\psi_1]\!]$ for each $\psi_1, \psi_2 \in \mathbb{C}_2$. Furthermore, if $\mathbb{C}_1$ is canonical, then $[\![\psi_2]\!] \subseteq [\![\psi_1]\!]$ implies ${\psi_1}^* \sqsubseteq {\psi_2}^*$, for each $\psi_1, \psi_2 \in \mathbb{C}_2$.

According to the following lemma (from [AČJYK96]) the algorithm of Section 2 is guaranteed to terminate exactly in the case where the constraint system is wqo.

Lemma 3.2 A qo $(A, \preceq)$ is wqo if and only if for each infinite ($\subseteq$-increasing) sequence $A_0 \subseteq A_1 \subseteq A_2 \subseteq \cdots$ of subsets of A, there is a j such that $A_j \sqsubseteq A_{j+1}$.

As we see from the above lemma, a crucial part in applying our method for new classes of systems is to define a tool for generating new well quasi-ordered constraint systems. In [AČJYK96, AJ98a] we describe a methodology for deriving well quasi-ordered constraint systems from existing ones. The following example shows that wqos are not robust under the $\lhd$ relation, that is well quasi-ordering of a constraint system $\mathbb{C}_1$ is in general not sufficient for well quasi-ordering of a constraint system $\mathbb{C}_2$ where $\mathbb{C}_1 \lhd \mathbb{C}_2$. Hence the methodology of [AČJYK96, AJ98a] can not be used e.g. for deriving well quasi-ordering of existential zones from well quasi-ordering of existential regions (well quasi-ordering of existential regions was shown in [AJ98b]).

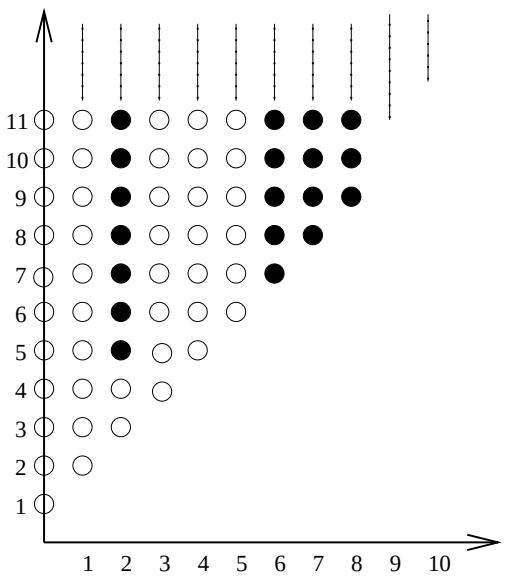

Figure 1: A graphic illustration of $[\![\phi_{2,5}]\!]$. Filled circles represent points belonging to the constraint.

Example 3.3 [Rado's Example] Consider the the set $X = \{(a,b)\,;\ a < b\} \subseteq \mathcal{N}^2$. Define a set $\mathbb{C}_1 = \{\phi_{i,j};\ (i,j) \in X\}$ of constraints, such that the denotation $[\![\phi_{i,j}]\!] \subseteq X$ of $\phi_{i,j}$ is the set $\{(a,b)\,;\ (j < a) \vee ((i = a) \wedge (j \leq b))\}$. It is straightforward to check that $\mathbb{C}_1$ is wqo: suppose that we have a bad sequence $(a_1,b_1), (a_2,b_2), \ldots$. 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 k and ℓ such that $b_{i_k} \leq b_{i_\ell}$ which is a contradiction.

Now, we define the set $\mathbb{C}_2 = \{\psi_Y;\ Y \subseteq X\}$ of constraints, where $[\![\psi_Y]\!] = \cup_{(i,j)\in Y}[\![\phi_{i,j}]\!]$. The sequence $\psi_{Y_1}, \psi_{Y_2}, \ldots$, where $Y_j = \{(i,j)\,;\ (i,j) \in X\}$, is a bad sequence, since for each $k, \ell : k < \ell$, we have $(k,\ell) \in \psi_{Y_\ell}$, but $(k,\ell) \notin \psi_{Y_k}$, and hence $[\![\psi_{Y_\ell}]\!] \not\subseteq [\![\psi_{Y_k}]\!]$. In Figure 1 and Figure 2 we give graphic illustrations of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$, respectively.

To prove that $\mathbb{C}_2$ is wqo, we propose to use a more powerful tool, namely that of better quasi-orderings (bqo). We move the definition of bqos to the appendix. In the following lemma we state some properties of bqos which we use later in the paper. As shown in the appendix, the first three properties stated in the lemma can be derived using results shown in [Mil85].

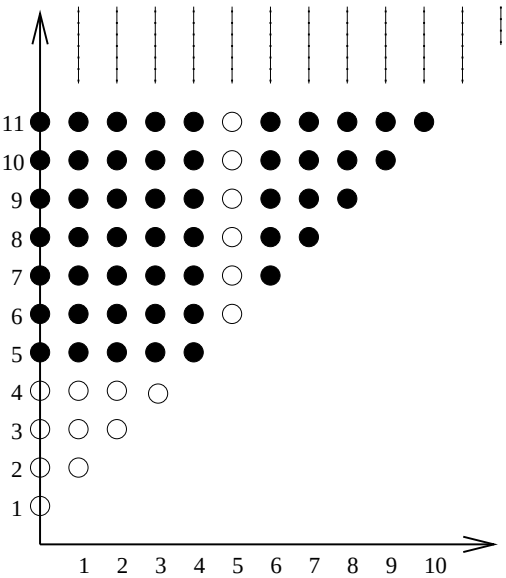

Figure 2: A graphic illustration of $[\![\psi_5]\!]$. Filled circles represent points belonging to the constraint.

Lemma 3.4 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 $(\mathcal{P}(A), \sqsubseteq)$ is bqo [Mar99][3].

Since bqo is a stronger relation than wqo (property 1), it follows by Lemma 3.2 that, to prove termination of the reachability algorithm of Section 2, it is sufficient to prove bqo of constraints under entailment. All constraint systems which have been used earlier in the literature (including those in [AČJYK96, AJ98a] are shown to be wqo, using the facts that $(A, =)$ is wqo if A is finite, and that $(A, \preceq)$ is wqo implies that $(A^*, \preceq^*)$ is wqo. By property 2 and property 3 it follows that all these constraint systems are also bqos. Also, from properties 3, 4, and Example 3.3, we conclude that bqos are "robust" under a larger set of operations than wqos.

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

The following lemma and Corollary 3.6 define our methodology: Given two constraint systems $\mathbb{C}_1$ and $\mathbb{C}_2$, where $\mathbb{C}_1 \lhd \mathbb{C}_2$, we can show that $\mathbb{C}_2$ is wqo by showing that $\mathbb{C}_1$ is bqo. By property 3, showing better quasi-ordering of $\mathbb{C}_1$ is, in most cases, as easy as showing that $\mathbb{C}_1$ is wqo.

Lemma 3.5 For constraint systems $\mathbb{C}_1$ and $\mathbb{C}_2$, where $\mathbb{C}_1 \lhd \mathbb{C}_2$, if $\mathbb{C}_1$ is bqo then $\mathbb{C}_2$ is also bqo.

Proof. From Lemma 3.4 (property 4) we conclude that $\mathcal{P}(\mathbb{C}_1)$ is bqo under $\sqsubseteq$, and hence by Proposition 3.1 and the definition of bqos (see the appendix) we conclude that $\mathbb{C}_2$ is bqo. □

Corollary 3.6 Given two constraint systems $\mathbb{C}_1$ and $\mathbb{C}_2$, where $\mathbb{C}_1 \lhd \mathbb{C}_2$, if $\mathbb{C}_1$ is bqo then $\mathbb{C}_2$ is wqo.

We apply our methodology for showing that existential zones are wqo (Lemma 5.3), by proving (Lemma 5.1) that existential zones are expressible in existential regions, and proving (Lemma 5.2) that existential regions are bqo. In a similar manner we show that existential CDDs (and DDDs) are bqo (Section 9).

4 Existential Zones

In this section we introduce a constraint system $\mathbb{Z}$ called *existential zones*, which can be used for specification of behaviour of systems with unboundedly many clocks. Intuitively, an existential zone Z characterizes a (possibly infinite) set of *(clock) configurations*. A configuration is defined by a finite set of clocks each with a value (real number) and a colour. The colour is used to describe, e.g., the control state of the process, or (in case of timed Petri nets) the place in which the clock resides. An existential zone Z represents minimal conditions on configurations. The conditions are formulated as specifications of colours of clocks, bounds on values of clocks, and bounds on differences between values of pairs of clocks. A configuration γ should have at least the number of clocks specified by Z. Furthermore, the colours and values of these clocks should satisfy the conditions of Z. In such a case, γ is allowed to have any number of additional clocks (whose colours and values are then irrelevant for the satisfiability of the zone by the configuration).

We let $\mathcal{N}$, $\mathcal{Z}$, and $\mathcal{R}^{\geq 0}$ denote the sets of natural numbers, integers, and nonnegative reals, respectively. For a set A, we define the set $Bags(A)$ of *bags* over A to be the set of mappings from A to $\mathcal{N}$. 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 $x \neq 2.4, 5.1$. We may also write B as $(2.4^3, 5.1^2)$. For bags B_1 and B_2 over a set A, we say that $B_1 \leq B_2$ if $B_1(a) \leq B_2(a)$ for each $a \in A$. We define $B_1 + B_2$ to be the bag B where $B(a) = B_1(a) + B_2(a)$, and (assuming $B_1 \leq B_2$) we define $B_2 - B_1$ to be the bag B where $B(a) = B_2(a) - B_1(a)$, for each $a \in A$. For a natural number n we use $\widehat{n}$ to denote the set $\{1, \ldots, n\}$. We assume a finite set $\mathcal{C}$ of *colours*, and a natural number *max*. The colours are used to model, e.g., the set of control states of processes, or the set of places of a (timed) Petri net. The value *max* will denote the maximum natural number which occurs in the enabling conditions of transitions in a given system, and hence differences between clock values which are greater than *max* will be irrelevant. We define the set $Max = \{-max, \ldots, max\} \cup \{\infty\}$.

A *(clock) configuration* γ is a bag $((c_1, x_1), \ldots, (c_n, x_n))$ over $\mathcal{C} \times \mathcal{R}^{\geq 0}$. We assume that we have a finite *index set* I which we use to index clocks in existential zones. We assume that I contains a special index 0, and use I^0 to denote $I - \{0\}$. An *existential zone* Z is a triple (I, C, D), where $C : I^0 \to \mathcal{C}$ define the colours of clocks, and $D : I \times I \to Max$ describes the differences in clock values. Let $h : I^0 \to \widehat{n}$ be an injection. We say that $\gamma = ((c_1, x_1), \ldots, (c_n, x_n))$ *satisfies* Z *with respect to* h, written $\gamma, h \models Z$, if the following conditions are satisfied.

- $C(i) = c_{h(i)}$, for each $i \in I^0$.
- $x_{h(j)} - x_{h(i)} \leq D(j, i)$, for each $i, j \in I^0$, with $i \neq j$.
- $x_{h(i)} \leq D(i, 0)$ and $-D(0, i) \leq x_{h(i)}$, for each $i \in I^0$.

We say that γ *satisfies* Z, written $\gamma \models Z$, if $\gamma, h \models Z$ for some h.

From the above definition we get the following.

Proposition 4.1 For an existential zone Z and a configuration γ, it is decidable whether $\gamma \models Z$.

Notice that the value of $D(i,i)$ is irrelevant for the satisfiability of Z, and that Z defines a number of minimal requirements on γ, in the sense that γ should contain at least $|I|$ clocks whose colours and values are constrained by the functions C and D respectively. The injection h maps each object i in I to a unique *witness* $h(i)$ in γ. We let $[\![Z]\!]$ denote the set $\{\gamma;\ \gamma \models Z\}$, and let $Z_1 \preceq Z_2$ denote $[\![Z_2]\!] \subseteq [\![Z_1]\!]$.

An existential zone $Z = (I, C, D)$ is said to be *normal* if for each $i, j, k \in I$, we have $D(j,i) \leq D(j,k) + D(k,i)$. It is easy to show the following.

Lemma 4.2 For each existential zone Z there is a unique (up to renaming of the index set) normal existential zone, written $\widetilde{Z}$, such that $[\![\widetilde{Z}]\!] = [\![Z]\!]$.

An existential zone Z is said to be *consistent* if $[\![Z]\!] \neq \emptyset$.

Lemma 4.3 Let Z be an existential zone and let $\widetilde{Z} = (I, C, D)$. Then, Z is consistent if and only if $0 \leq D(0,0)$

Remark For simplicity of presentation, we use only non-strict inequalities in the definition of existential zones. All the results can be generalized in a straightforward manner to include the more general case, where we also allow strict inequalities.

5 Well Quasi-Ordering of Existential Zones

We recall a related constraint system to existential zones, namely that of *existential regions* $\mathbb{R}$, introduced in [AJ98b]. An *existential region* R is a list $(B_0, B_1, \ldots, B_m, B_{m+1})$ of bags where $m \geq 0$, and B_i is a bag over $\mathcal{C} \times \mathcal{N}$. Notice that the list is at least of length 2. In a similar manner to existential zones, an existential region R defines a minimal set of conditions which should be satisfied by a configuration γ in order for γ to satisfy R. Intuitively, B_0 represents clocks with values which have fractional parts equal to 0. The bags $B_1, \ldots, B_m$ represent clocks with increasing fractional parts. The clocks corresponding to the same bag should all have the same fractional parts, while clocks belonging to a bag B_i should have fractional parts which are strictly less than those in B_{i+1}. Finally, the bag B_{m+1} represents clocks with values which are strictly greater than *max* (regardless of their fractional parts).

Formally, let B_i be of the form $((c_{i,1}, a_{i,1}), \ldots, (c_{i,\ell_i}, a_{i,\ell_i}))$, and let $\gamma = ((d_1, x_1), \ldots, (d_n, x_n))$ be a configuration. Consider an injection $h : \{(i,j);\ (1 \leq i \leq m+1) \& (1 \leq j \leq \ell_i)\} \mapsto \widehat{n}$. We say that γ *satisfies* R *with respect to* h, written $\gamma, h \models R$, if the following conditions are satisfied.

- $c_{i,j} = d_{h(i,j)}$, for $i : 0 \leq i \leq m+1$, $j : 1 \leq j \leq \ell_i$. That is, each clock should have the same colour as that demanded by the corresponding object in R.
- $a_{i,j} = \lfloor x_{h(i,j)} \rfloor$, for $i : 0 \leq i \leq m$, $j : 1 \leq j \leq \ell_i$; and $x_{h(n+1,j)} > max$, for $j : 1 \leq j \leq \ell_{n+1}$. That is, each clock (except those corresponding to elements in the last bag, i.e., B_{m+1}) should have an integral part which agrees with the value demanded by the corresponding object in R. The actual clock values corresponding to elements in B_{m+1} are not relevant provided that they are greater than max.
- $fract(x_{h(0,j)}) = 0$, for $1 \leq j \leq \ell_0$, i.e. clocks corresponding to elements in B_0 should have fractional parts which are equal to 0.
- $fract(x_{h(i,j)}) = fract(x_{h(i,k)})$, for $0 \leq i \leq m$, $1 \leq j \leq \ell_i$. Clocks corresponding to elements in the same bag (except the last one) should have equal fractional parts.
- $fract(x_{h(i,j)}) < fract(x_{h(i+1,k)})$, for $0 \leq i < m$, $1 \leq j \leq \ell_i$, $1 \leq k \leq \ell_{i+1}$. Clocks corresponding to elements in a bag should have fractional parts which are strictly less than those corresponding to elements in the next bag (except when the next bag is the last bag).

The injection h has the same role as in the interpretation of existential zones, i.e, it maps each "object" in R to a unique clock in γ. Each "object" in R represents a condition to be satisfied by a clock in γ. The injection h finds for each condition in R a *witness* (clock) in γ. We say that γ *satisfies* R, written $\gamma \models R$, if $\gamma, h \models R$ for some h.

As usual, we let $[\![R]\!] = \{\gamma;\ M \models R\}$ and let $R_1 \preceq R_2$ denote $[\![R_2]\!] \subseteq [\![R_1]\!]$. From the definitions of existential regions and zones we conclude the following.

Lemma 5.1 $\mathbb{R}$ is canonical and $\mathbb{R} \lhd \mathbb{Z}$.

From the definition of existential regions and repeated applications of Lemma 3.4 (properties 2 and 3) we get the following.

Lemma 5.2 $\mathbb{R}$ is bqo.

From Lemma 3.5, Lemma 5.1, and Lemma 5.2 it follows that

Lemma 5.3 $\mathbb{Z}$ is bqo (and hence also wqo).

6 Entailment

In this section we describe how to check the entailment relation between zones. Although the entailment relation can be checked in polynomial time for zones, the complexity, in the case of existential zones, is exponential. As explained below, this is due to the fact that we have to have check implication for all variable permutations.

We reduce checking entailment between existential zones into validity of formulas in a logic which we here call *Difference Bound Logic (DBL)*. The atomic formulas are either of the form $v \leq c$ or of the form $v - u \leq c$, where v and u are variables interpreted over $\mathcal{R}^{\geq 0}$ and $c \in \mathcal{N}$. Furthermore the set of formulas is closed under the propositional connectives. It is easy to see that validity of DBL-formulas is NP-complete.

Suppose that we are given two existential zones $Z_1 = (I_1, C_1, D_1)$ and $Z_2 = (I_2, C_2, D_2)$. We translate the relation $Z_1 \preceq Z_2$ into validity of a DBL-formula F as follows. We define the set of free variables in F to be $\{v_i;\ i \in I_2^0\}$. Let H be the set of injections from I_1^0 to I_2^0 such that $h \in H$ if and only if $C_1(i) = C_2(h(i))$ for each $i \in I_1^0$. We define $F = \big(F_1 \implies \big(\bigvee_{h \in H} F_2\big)\big)$, where $F_1 = F_{11} \wedge F_{12} \wedge F_{13}$, and $F_2 = F_{21} \wedge F_{22} \wedge F_{23}$, and

- $F_{11} = \bigwedge_{i,j \in I_2^0, j \neq i} (v_j - v_i \leq D_2(j,i))$.
- $F_{12} = \bigwedge_{i \in I_2^0} (v_i \leq D_2(i,0))$.
- $F_{13} = \bigwedge_{i \in I_2^0} (D_2(0,i) \leq v_i)$.
- $F_{21} = \bigwedge_{i,j \in I_1^0, j \neq i} \big(v_{h(j)} - v_{h(i)} \leq D_1(h(j), h(i))\big)$.

- $F_{22} = \bigwedge_{i \in I_1^0} \big(v_{h(i)} \leq D_1(h(i), 0)\big)$.
- $F_{23} = \bigwedge_{i \in I_1^0} \big(D_1(h(i), 0) \leq v_{h(i)}\big)$.

This gives the following.

Lemma 6.1 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 exponential time. 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 Timed Petri Nets

We consider *Timed Petri Nets (TPNs)* where each token 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.

We use a set *Intrv* of *intervals* of the form $[a, b]$, where $a \in \mathcal{N}$ and $b \in \mathcal{N} \cup \{\infty\}$. For $x \in \mathcal{R}^{\geq 0}$, we write $x \in [a, b]$ to denote that $a \leq x \leq b$.

A *Timed Petri Net (TPN)* is a tuple $N = (P, T, In, Out)$ where P is a finite set of *places*, T is a finite set of *transitions*, $In, Out : T \times P \mapsto Bags(Intrv)$. If $In(t, p)(\mathcal{I}) \neq \emptyset$ ($Out(t, p)(\mathcal{I}) \neq \emptyset$), for some interval $\mathcal{I}$, we say that p is an *input (output) place* of t. Relating to Section 4, the set $\mathcal{C}$ of colours in the definition of existential zones corresponds to the set of P of places in a timed Petri net.

A configuration, called a *marking* M, of N is a bag over $P \times \mathcal{R}^{\geq 0}$. The marking M defines the numbers and ages of tokens in each place in the net. That is, $M(p, x)$ defines the number of tokens with age x in place p. For

example, if $M = ((p_1, 2.5), (p_1, 1.3), (p_2, 4.7), (p_2, 4.7))$, then, in the marking M, there are two tokens with ages 2.5 and 1.3 in p_1, and two tokens each with age 4.7 in the place p_2. Abusing notation, we define, for each place p, a bag $M(p)$ over $\mathcal{R}^{\geq 0}$, where $M(p)(x) = M(p, x)$. Notice that untimed Petri nets are a special case in our model where all intervals are of the form $[0, \infty]$.

We define two types of transition relations on markings. A *Timed transition* increases the age of each token by the same real number. Formally $M_1 \longrightarrow_T M_2$ if M_1 is of the form $((p_1, x_1), \ldots, (p_n, x_n))$, and there is $\delta \in \mathcal{R}^{\geq 0}$ such that $M_2 = ((p_1, x_1 + \delta), \ldots, (p_n, x_n + \delta))$.

We define the set of *discrete transitions* $\longrightarrow_d$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of firing the transition t. More precisely, we define $M_1 \longrightarrow_t M_2$ if, for each place p with $In(t, p) = (\mathcal{I}_1, \ldots, \mathcal{I}_m)$ and $Out(t, p) = (\mathcal{J}_1, \ldots, \mathcal{J}_n)$, there are bags $B_1 = (x_1, \ldots, x_m)$ and $B_2 = (y_1, \ldots, y_n)$ over $\mathcal{R}^{\geq 0}$, such that the following holds.

- $B_1 \leq M_1(p)$.
- $x_i \in \mathcal{I}_i$, for $i : 1 \leq i \leq m$.
- $y_i \in \mathcal{J}_i$, for $i : 1 \leq i \leq n$.
- $M_2 = M_1 - B_1 + B_2$.

Intuitively, a transition t may be fired only if for each incoming arc to the transition, there is a token with the "right" age in the corresponding input place. This token will be removed from the input place when the transition is fired. Furthermore, for each outgoing arc, a token with an age in the interval will be added to the output place. Notice that we assume a lazy (non-urgent) behaviour of the net. 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.

8 Computing Predecessors

We define a function Pre such that, for a zone Z, the value of $Pre(Z)$ is a finite set $\{Z_1, \ldots, Z_m\}$ of zones, where $M \models Z$ if and only if there is a marking M' with $M' \longrightarrow M$ and $M' \models Z_i$ for some $i : 1 \leq i \leq m$. Thus, the set $Pre(Z)$ characterizes the set of markings from which we can reach a marking satisfying Z through the performance of a single discrete or timed transition. 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 $Pre_D = \cup_{t \in T} Pre_t$, 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 $Z = (I, C, D)$, and a timed Petri net $N = (P, T, In, Out)$. From Lemma 4.2 we know that assuming Z to be normal does not affect the generality of our results.

For an interval $\mathcal{I} = [a, b]$, and $i \in I^0$, we define the *conjunction* $Z \otimes (\mathcal{I}, i)$ of Z with $\mathcal{I}$ at i to be the existential zone $Z' = (I, C, D')$, 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 $j, k \in I$, with $k \neq j$, $(j, k) \neq (i, 0)$ and $(j, k) \neq (0, i)$.

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 $Z' = (I \cup \{i\}, C', D')$, where $i \notin I$, and

- $D'(i, 0) = b$, and $D'(0, i) = -a$.
- $D'(i, j) = \infty$, and $D'(j, i) = \infty$, for each $j \in I^0$.
- $C'(i) = p$.
- $D'(k, j) = D(k, j)$, for each $j, k \in I$, and $C'(j) = C(j)$, for each $j \in I^0$.

For $i \in I^0$, we define the *abstraction* $Z \backslash i$ of i in Z to be to be the zone $Z' = (I - \{i\}, C', D')$, where $D'(k, j) = D(k, j)$, for each $j, k \in I - \{i\}$, and $C'(\ell) = C(\ell)$, for each $\ell \in I^0 - \{i\}$.

Lemma 8.1 Consider a TPN $N = (P, T, In, Out)$, a transition $t \in T$, and an existential zone $Z = (I, C, D)$. 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 $h : I^0 \longrightarrow \widehat{\ell}$ with a domain $\{i_1, \ldots, i_n\}$, and an existential zone Z_1 satisfying the following conditions.

- $C(i_j) = q_{h(i_j)}$, for each $j : 1 \leq j \leq n$
- $Z \otimes (\mathcal{J}_{h(i_1)}, i_1) \otimes \cdots \otimes (\mathcal{J}_{h(i_n)}, i_n)$ 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 8.2 For an existential zone $Z = (I, C, D)$, the set $Pre_\delta(Z)$ is the existential zone $Z' = (I, C, D')$,where $D'(0, i) = 0$ and $D'(j, i) = D(j, i)$ if $j \neq 0$, for each $i, j \in I$, with $i \neq j$.

From Lemma 8.1 and Lemma 8.2 we get the following.

Lemma 8.3 For an existential zone Z, the set $Pre(Z)$ is computable.

9 Existential CDDs and DDDs

CDDs [LPWY99] and DDDs [MLAH99] are constraint systems invented recently to give representations of real-time systems which are even more compact than zones. In a similar manner to existential zones, we modify the definitions of CDDs (DDDs) into *existential CDDs (DDDs)*, in order to make them suitable for verifying systems with unboundedly many clocks. Below we give the definition of existential DDDs $\mathbb{D}$. The definition of existential CDDs can be stated in a similar manner.

An *existential DDD* is a tuple $Y = (I, C, \mathtt{V}, \mathtt{E})$, where I and C are defined as for existential zones (Section 4), and $(\mathtt{V}, \mathtt{E})$ is a finite directed acyclic graph where $\mathtt{V}$ is the set of vertices, $\mathtt{E}$ is the set of edges. We assume that $\mathtt{V}$ 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 is two. Each vertex $\mathtt{v} \in \mathtt{V} - \{\mathtt{v}^0, \mathtt{v}^1\}$ has the following attributes: $pos(\mathtt{v}), neg(\mathtt{v}) \in I$, $op(\mathtt{v}) \in \{<, \leq\}$, $const(\mathtt{v}) \in \mathcal{Z}$, 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 $(\phi \wedge \phi_1) \vee (\neg\phi \wedge \phi_2)$.

Consider a clock configuration $\gamma = ((c_1, x_1), \ldots, (c_n, x_n))$, an existential DDD $Y = (I, C, \mathtt{V}, \mathtt{E})$, and an injection $h : I^0 \rightarrow \widehat{n}$. We say that γ *satisfies* Y *at* $\mathtt{v} \in \mathtt{V}$ *with respect to* h, written $\gamma, h \models (Y, \mathtt{v})$, if $C(i) = c_{h(i)}$, for each $i \in I$, and either

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

From the definitions of existential zones and existential DDDs we get the following.

Lemma 9.1 $\mathbb{Z} \lhd \mathbb{D}$.

Lemma 9.1 together with Lemma 5.3 and Lemma 3.5 give us the following.

Lemma 9.2 $\mathbb{D}$ is bqo (and hence also wqo).

10 Experimental Results

We have implemented a prototype to perform reachability analysis for TPNs. The constraints are represented by existential DDDs. The implementa-

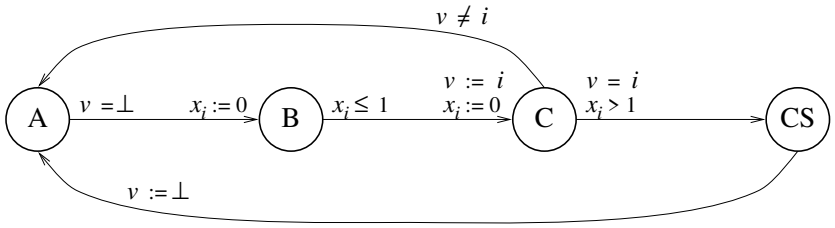

Figure 3: Fischer's Protocol for Mutual Exclusion

tion is based on a DDD package developed at Technical University of Denmark [ML98]. We used the tool to verify a parameterized version of Fischer's protocol.

The purpose of the 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. [SBK92].

The protocol consists of each process running the code which is graphically described in Figure 3. 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 proceed to state C within one time unit or get stuck in B forever. When making the transition from B to C, the process sets the value of the shared variable to its own index and resets its local clock. 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 $\dagger$ represent that the value of the shared variable is the index of the process modeled by the token in that

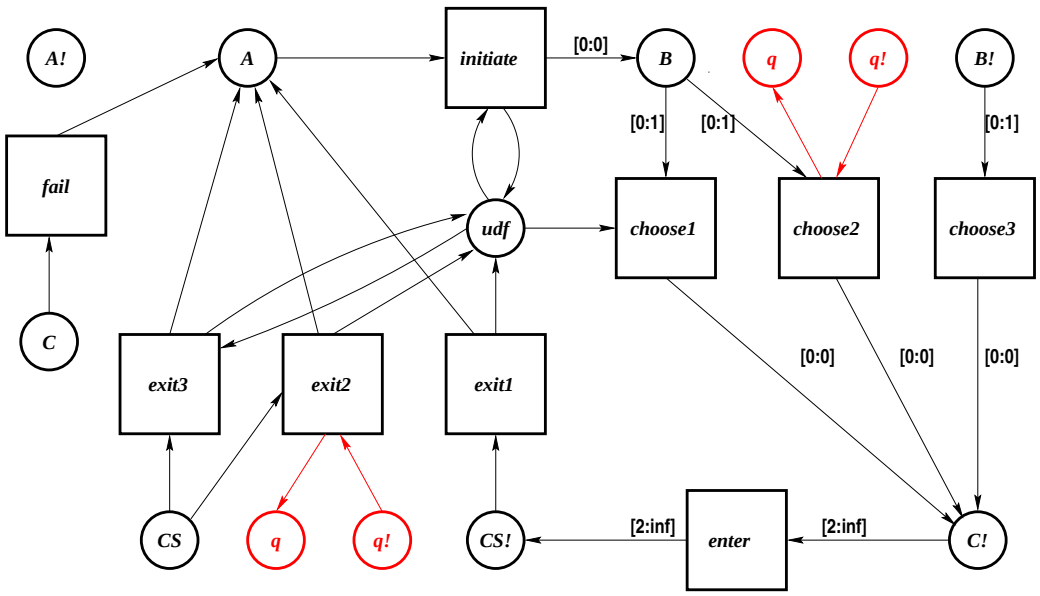

Figure 4: TPN model of Fischer's Protocol for Mutual Exclusion

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 3 yields the Petri net model in Figure 4. q is used to denote an arbitrary process state.

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:

- $(CS\ [0:\infty], CS\ [0:\infty])$
- $\left(CS\ [0:\infty], CS^{\dagger}\ [0:\infty]\right)$
- $\left(CS^{\dagger}\ [0:\infty], CS^{\dagger}\ [0:\infty]\right)$

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.

Acknowledgments

We are grateful to Alberto Marcone for pointing out the potential of using better quasi-ordered relations. Petr Jančar and Alberto Marcone provided the proof of Lemma 3.4 (property 4), which is central for developing the methodology described in this paper. Jančar [Jan99] provides a simple and elegant proof of a slightly weaker version of the lemma, namely that bqo of $(A, \preceq)$ is sufficient for wqo of $(\mathcal{P}(A), \sqsubseteq)$. Our proof for TPNs carries through even using this weaker version of the lemma. Many thanks to Ahmed Bouajjani, Jens Gustedt, Thomas Forster, Tero Harju, Bengt Jonsson, Joseph Sifakis Wolfgang Thomas and Stefano Varricchio for helpful comments and discussions. Special thanks to Jesper Møller and Henrik Reif Andersen for letting us use their DDD implementation and for helping us solve the problems that arose in the process.

References

[AČ98] Parosh Aziz Abdulla and Karlis Čerāns. Simulation is decidable for one-counter nets. In *Proc. CONCUR '98, 9^{th} Int. Conf. on Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science*, pages 253–268, 1998.

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 414–425, Philadelphia, 1990.

[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.

[AD90] R. Alur and D. Dill. Automata for modeeling 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.

[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.

[BS95] O. Burkart and B. Steffen. Composition, decomposition, and model checking of pushdown processes. *Nordic Journal of Computing*, 2(2):89–125, 1995.

[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.

[Esp95] J. Esparza. Petri nets, commutative context-free grammers, and basic parallel processes. In *Proc. Fundementals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232, 1995.

[Fin94] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.

[GS92] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In *Proc. ICALP '95*, 1995.

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

[Jan97] P. Jančar. Bisimulation equivalence is decidable for one-counter processes. In *Proc. ICALP '97*, pages 549–559, 1997.

[Jan99] Petr Jančar. ω^2-well quasi-orderings and reachability analysis. Technical Report 158, Deaprtment 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.

[JM95] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proc. CONCUR '95, 6^{th} Int. Conf. on Concurrency Theory*, volume 962 of *Lecture Notes in Computer Science*, pages 348–362. Springer Verlag, 1995.

[JP93] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 107(2):272–302, Dec. 1993.

[KMM+97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In O. Grumberg, editor, *Proc. 9^{th} Int. Conf. on Computer Aided Verification*, volume 1254, pages 424–435, Haifa, Israel, 1997. Springer Verlag.

[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] Alberto Marcone, 1999. private communication.

[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.

[SBK92] F. B. Schneider, Bloom B, and Marzullo K. 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.

[Wol86] Pierre Wolper. Expressing interesting properties of programs in propositional temporal logic (extended abstract). In *Proc. 13^{th} ACM Symp. on Principles of Programming Languages*, pages 184–193, Jan. 1986.

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

A 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 $\mathcal{N}$. 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 $tail(s_1) \ll s_2$. A set $\beta \subseteq \mathcal{N}^{<\omega}$ is said to be a barrier if the following three conditions are satisfied.

- β is infinite.
- 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 $i \in \mathcal{N}$.

Below we describe two properties of bqos, shown in [Mil85], which we use to derive the properties stated in Lemma 3.4.

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

From Lemma A.1 we derive the first three properties stated in Lemma 3.4.

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 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 $f'(s) = w'$ 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.