Better Quasi-Ordered 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

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

1 Introduction

A major current challenge in automatic program verification is to extend model checking [CES86, QS82] to transition systems with infinite state

*Parts of this paper have appeared in Proc. LICS'2000, 14th IEEE Int. Symp. on Logic in Computer Science, and Proc. ICATPN'2001, 22nd Int. Conf. on application and theory of Petri nets.

spaces. Standard techniques such as reachability analysis and tableau procedures can be adapted, by using *constraints* to represent (potentially infinite) sets of states. These algorithms are based on two operations, namely that of computing predecessors or successors of sets of states (represented by constraints), and that of checking for termination (formulated as entailment between constraints). Since the number of constraints is not *a priori* bounded, a key problem when applying the algorithms, is to guarantee termination. A general powerful tool which can be applied for proving termination is to show that the set of constraints is *well quasi-ordered* under entailment. In [AČJYK96, AJ01], a constraint based backward reachability algorithm is presented. Furthermore, a methodology is defined for inventing well quasi-ordered constraint systems. The key idea is to start from a set of "basic" constraints, and repeatedly derive new ones, using the fact that well quasi-orderings are closed under certain operations on constraints such as building finite trees, words, vectors, multisets, sets, etc. The methodology has been applied both to unify earlier existing results for Petri nets, timed automata, lossy channel systems, completely specified protocols, relational automata, etc, and to design verification algorithms for new classes of systems such as timed networks [AJ98], broadcast protocols [EFM99, DEP99], and cache coherence protocols [Del00]. However many of the constraint systems constructed according to this method, suffer from a "constraint explosion" problem, as the number of constraints generated when computing predecessors (successors) grows exponentially with the number of components.

In this work, we demonstrate that a refinement of the theory of well quasi-orderings, called the theory of *better quasi-orderings* [Mil85, Pou85] is more appropriate for symbolic model checking, as it allows for constraint systems which are more compact and hence less prone to constraint explosion. More precisely, better quasi-orderings offer two advantages: (i) better quasi-ordering implies well quasi-ordering; hence all the verification algorithms originally designed for well quasi-ordered constraint systems are also applicable to better quasi-ordered ones; and (ii) better quasi-orderings are more "robust" than well quasi-orderings. For instance, in addition to the above mentioned operations, better quasi-ordered constraint systems (in contrast to well quasi-ordered ones) are closed under disjunction: if a set of constraints is better quasi-ordered under entailment, then the set of finite disjunctions of these constraints is also better quasi-ordered under entailment. In this paper, we provide several examples which show that using disjunction often leads to very compact constraint systems.

First, we propose a new constraint system, which we call *existential zones*, for verification of systems with unboundedly many clocks such as timed networks [AJ98] and timed Petri nets (Section 5). Such systems cannot be modelled as real-time automata, since the latter operate on a finite set of clocks. An existential zone specifies a minimal required behaviour, typically of the form $\exists x_1 x_2 :\ 3 \leq x_2 - x_1 \leq 8$, characterizing the set of configurations in which there exist *at least* two clocks whose values differ by at least 3 and at most 8. Existential zones are related to *existential regions*, which are used in [AJ98] for verification of timed networks. Existential regions are better quasi-ordered since they are constructed by repeatedly building words, multisets, and sets. Each existential zone is equivalent to the disjunction of a finite number of existential regions. Since better quasi-orderings are closed under disjunction, it follows that existential zones are better quasi-ordered (and hence well quasi-ordered). The well quasi-ordering of existential zones cannot be shown using the approach of [AČJYK96, AJ01, FS98], since well quasi-orderings in general are not closed under disjunction. In fact, an existential zone is often equivalent to the disjunction of an exponential number of existensial regions, thus offering a much more compact representation (in the same manner that *zones* are more efficient than *regions* in verification tools for real-time automata [LPY97, Yov97]). We can extend the results further and consider "existential variants" of CDDs [BLP$^+$99] and DDDs [MLAH99]– constraint systems which are even more compact than zones. We have implemented a prototype based on existential DDDs, and carried out a verification of a parametrized version of Fischer's protocol. While the set of constraints explodes when using existential regions, our tool performs the verification in a few seconds.

We also consider broadcast protocols, which consist of an arbitrary number of finite-state processes, communicating through rendezvous or through broadcast. In [EFM99] safety properties are checked, using constraints which characterize upward closed sets of vectors of natural numbers. In [DEP99] several new constraint systems are proposed, represented by different forms of linear inequalities over natural numbers. Since the new constraint systems cannot be constructed from upward closed sets using the previously mentioned constraint operations, 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, the upward closed sets are characterized by constraints which are vectors of natural numbers and hence are better quasi-ordered. Using the fact that each inequality is a finite union (disjunction) of upward closed sets, we conclude that they are also better quasi-ordered (and hence well quasi-ordered).

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

Related Work The first work which applies well quasi-orderings in symbolic model checking is reported in [AJ93]. The main contribution was an algorithm for checking safety properties for *lossy channel systems*. The idea of the algorithm is to perform *backward* reachability analysis using the fact that the underlying transition relation is *monotonic* under a given well quasi-ordering.

Independently, Finkel [Fin94] used well quasi-orderings for checking termination properties. This algorithm uses forward analysis and is therefore not sufficiently powerful for verification of safety properties.

The method of [AJ93] was extended in [AČJYK96] into a general framework for verification of relational automata, (Timed) Petri nets, timed networks, etc. In [FS98] the monotonicity conditions of [AČJYK96] were further relaxed extending applicability to new classes of systems.

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 [AJ03]. 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, RFC99], 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.

Outline In the next Section we introduce the notions of constraints and well quasi-orderings. In Section 3 and Section 4 we give the basics of the theory of better quasi-orderings and its application in model checking. Timed Petri nets are defined in Section 5 and in Section 6 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 7. In Section 8, Section 9 and Section 10 we provide better quasi-ordered constraint systems for the verification of broadcast protocols, lossy channel systems and integral relational automata, respectively. Finally in Section 11 we give some conclusions and directions for future work.

2 Constraints and WQOs

In this section, we introduce the notions of *constraints* and *well quasi-orderings*, and describe how to use them for performing symbolic model checking. We assume a transition system $(\Gamma, \longrightarrow)$, where Γ is a potentially infinite set of *configurations*, and $\longrightarrow$ is a transition relation on Γ whose reflexive transitive closure is denoted by $\xrightarrow{*}$.

Constraints We use *constraints* ϕ for representing sets $[\![\phi]\!]$ of configurations. We define an *entailment relation* $\preceq$ on constraints, where $\phi_1 \preceq \phi_2$ iff $[\![\phi_2]\!] \subseteq [\![\phi_1]\!]$, and let $\equiv$ be the equivalence relation induced by $\preceq$. We sometimes write disjunctions $\phi_1 \vee \cdots \vee \phi_n$ of constraints as $\vee\{\phi_1, \ldots, \phi_n\}$. For sets Φ_1, Φ_2 of constraints, we let $\Phi_1 \sqsubseteq \Phi_2$ denote that for each $\phi_2 \in \Phi_2$ there is a $\phi_1 \in \Phi_1$ with $\phi_1 \preceq \phi_2$. Notice that $\Phi_1 \sqsubseteq \Phi_2$ implies $\vee\Phi_1 \preceq \vee\Phi_2$.

Reachability In the sequel, we concentrate on the *reachability problem*: given a configuration γ_{init} and a constraint ϕ_F, is there $\gamma_F \in [\![\phi_F]\!]$ such that $\gamma_{init} \xrightarrow{*} \gamma_F$? We perform a backward reachability analysis, generating a sequence $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots$ of finite sets of constraints where $\Phi_0 = \{\phi_F\}$ and $\Phi_{j+1} = \Phi_j \cup Pre(\Phi_j)$. Here $Pre(\Phi) = \cup_{\phi \in \Phi} Pre(\phi)$, where $Pre(\phi)$ is a finite set of constraints, such that $[\![\vee Pre(\phi)]\!] = \{\gamma' |\ \exists \gamma \in [\![\phi]\!].\ \gamma' \longrightarrow \gamma\}$. For all the constraint systems we consider in this paper, the set $Pre(\phi)$ exists and is computable. Since $\Phi_0 \sqsupseteq \Phi_1 \sqsupseteq \Phi_2 \sqsupseteq \cdots$, the algorithm terminates when we reach a point j where $\Phi_j \sqsubseteq \Phi_{j+1}$ (implying $\vee\Phi_{j+1} \equiv \vee\Phi_j$). Then, Φ_j characterizes the set of all predecessors of ϕ_F (sometimes written as $Pre^*(\phi_F)$). This means that the answer to the reachability question is equivalent to whether $\gamma_{init} \in [\![\vee\Phi_j]\!]$. We observe that, in order to be able to implement the algorithm for a given class of systems, the constraint system should allow (i) computing $Pre(\phi)$, (ii) checking entailment between constraints, and satisfiability of a constraint by a configuration.

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

Lemma 2.1. A constraint system is well quasi-ordered iff for each infinite ($\subseteq$-increasing) sequence $\Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \cdots$ of constraint sets, there is a j such that $\Phi_j \sqsubseteq \Phi_{j+1}$.

Remark on Well Quasi-Ordered Transition Systems Alternatively, we can consider transition systems $(\Gamma, \longrightarrow)$ which are *well quasi-ordered* [AČJYK96, AJ01, FS98]. This means that the set Γ of configurations is equipped with a well quasi-ordering $\preceq_\Gamma$ such that the transition relation is *monotonic* with respect to $\preceq_\Gamma$. In other words, for configurations $\gamma_1, \gamma_1', \gamma_2$, if $\gamma_1 \preceq_\Gamma \gamma_1'$ and $\gamma_1 \longrightarrow \gamma_2$ then there is a configuration γ_2' such that $\gamma_2 \preceq_\Gamma \gamma_2'$ and $\gamma_1' \longrightarrow \gamma_2'$. We can now develop a theory based on well quasi-ordered transitions systems rather than well quasi-ordered constraint systems. The two theories are intimately related and yield identical model checking algorithms [AČJYK96, AJ01]. All constraints which we we will consider in this paper characterize sets of configurations which are upward closed with respect to $\preceq_\Gamma$. This means that the reachability problem described above in fact asks about reachability of sets of states which are upward closed sets of states rather than that of a single state. This offers two advantages:

- Checking safety properties amounts to upward closed set reachability. More precisely, the states in $[\![\gamma_F]\!]$ are usually taken to be bad states that we do not want to occur during an execution. Using standard techniques [GW93, VW86], we can reduce several classes of safety properties to the reachability problem.
- Single state reachability is more difficult to solve. For instance, in the context of Petri nets, upward closed set reachability amounts to *coverability*. In *timed Petri nets*, single state reachability is undecidable [RFC99], while we show in this paper that coverability is decidable.

3 Basics of BQOs

In this section, we introduce the basic definitions and properties of better quasi-orderings. We let $\mathcal{N}$ denote the set of natural numbers, and 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*$(s_1) \ll s_2$. An infinite set $\beta \subseteq \mathcal{N}^{<\omega}$ is said to be a barrier if the following two conditions are satisfied.

- There are no $s_1, s_2 \in \beta$ such that $\lambda(s_1) \subsetneq \lambda(s_2)$.

- For each $s_2 \in \mathcal{N}^\omega$ there is $s_1 \in \beta$ with $s_1 \ll s_2$.

Let $(A, \preceq)$ be a quasi-ordering, i.e., $\preceq$ is a reflexive and transitive relation on A. 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 (bqo)* if each A-pattern is good.

We use A^ω to denote the set of infinite sequences over A. For $w \in A^\omega$, 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 monotone[1] injection $h : \mathcal{N} \to \mathcal{N}$ such that $w_1(i) \preceq w_2(h(i))$, for each $i \in \mathcal{N}$.

We shall use the following two properties (from [Mil85])

Lemma 3.1.

- 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 bqo then $(A^\omega, \preceq^\omega)$ is bqo

4 Application of BQOs

As evident from Lemma 2.1, well quasi-ordering is crucial for termination of the symbolic algorithm presented in Section 2. Furthermore, three other properties of a given constraint system decide how efficient the algorithm may run in practice. These properties are the size of the set $Pre(\phi)$, the cost of checking entailment and membership, and the number of iterations needed before achieving termination. In [AČJYK96, AJ01, FS98], a methodology is defined for inventing well quasi-ordered constraint systems, based on the fact that all finite domains are well quasi-ordered under equality, and that well quasi-orderings are closed under a basic set of operations including building finite trees, words, vectors, multisets, sets, etc. This means that we can start from a set of constraints over finite domains, and then repeatedly generate new constraints by building more compound data structures. A typical application of this approach is a constraint system, called *existential regions*, introduced in [AJ98] for verification of systems with unboundedly many clocks. However, constraints developed according to the above methodology suffer from "constraint explosion" caused by the size of the set $Pre(\phi)$. For instance, using existential regions, the set of generated constraints explodes even for very small examples. Often, the constraint explosion can be much reduced, by considering new constraint systems, which are disjunctions of the ones derived using the above mentioned set of operations. In Section 6 we present *existential zones* each of which corresponds to the disjunction of

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

a (sometimes exponential) number of existential regions. Thus, existential zones offer a much more compact representation, allowing us to verify a parameterized version of Fischer's protocol in a few seconds. As we show in this section, well quasi-ordered constraint systems are not closed under disjunction, and hence we cannot prove well quasi-ordering of existential zones within the framework of [AČJYK96, AJ01, FS98].

Instead of wqos, we propose here to use an alternative approach based on bqos. In Theorem 4.1, we state some properties of bqos which make them attractive for symbolic model checking. In the rest of this section we write $(A, \preceq)$ to denote a quasi-ordering $\preceq$ on a set A. Let A^* denote the set of finite words over A, and let $A^{\oplus}$ denote the set of finite multisets over A. For a natural number n, let $\widehat{n}$ denote the set $\{1, \ldots, n\}$. An element w of A^* and of $A^{\oplus}$ can be represented as a mapping $w\colon \widehat{|w|} \mapsto A$ where $|w|$ is the size of the multiset or the length of the sequence. Given a quasi-order $\preceq$ on a set A, define the quasi-order $\preceq^*$ on A^* by letting $w \preceq^* w'$ if and only if there is a strictly monotone injection $h\colon \widehat{|w|} \mapsto \widehat{|w'|}$ such that $w(j) \preceq w'(h(j))$ for $1 \leq j \leq |w|$. Define the quasi-order $\preceq^{\oplus}$ on $A^{\oplus}$ by $w \preceq^{\oplus} w'$ if and only if there is a (not necessarily monotone) injection $h\colon \widehat{|w|} \mapsto \widehat{|w'|}$ such that $w(j) \preceq w'(h(j))$ for $1 \leq j \leq |w|$.

In the following theorem we state some properties of bqos which we use later in the paper. The proof of property 5 is in [Mar99]. We use $\mathcal{P}(A)$ to denote the powerset of A.

Theorem 4.1. 1. Each bqo is wqo.

2. If A is finite, then $(A, =)$ is bqo.

3. If $(A, \preceq)$ is bqo, then $(A^*, \preceq^*)$ is bqo.

4. If $(A, \preceq)$ is bqo, then $(A^{\oplus}, \preceq^{\oplus})$ is bqo.

5. If $(A, \preceq)$ is bqo, then $(\mathcal{P}(A), \sqsubseteq)$ is bqo [Mar99][2].

Proof. We show properties 1-4.

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 3.1, 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.

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

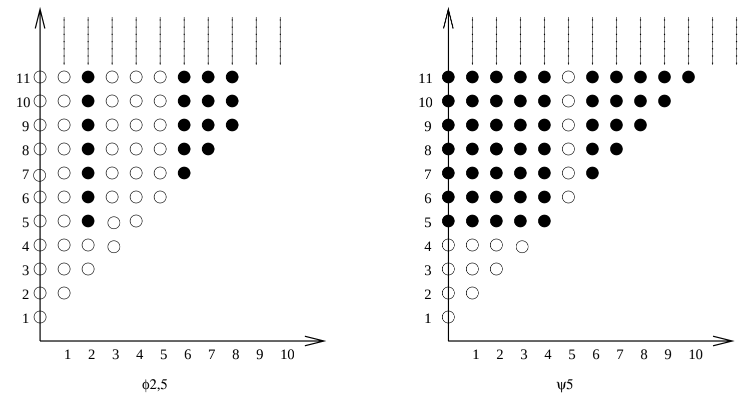

Figure 1: A graphic illustration of $[\![\phi_{2,5}]\!]$ and $[\![\psi_5]\!]$. Filled circles represent points satisfying the corresponding constraint.

3. Suppose that $(A, \preceq)$ is bqo. We show that $(A^*, \preceq^*)$ is bqo. Take any $b \notin A$. For $w \in A^*$, we let w' denote wb^ω (i.e., we add infinitely many b:s to the end of w). It is clear that $w_1 \preceq^* w_2$ if and only if $w_1' \preceq^\omega w_2'$. Let $f : \beta \to A^*$ be an A^*-pattern. We know that $f' : \beta \to A^\omega$, where $f'(s) = w'$ iff $f(s) = w$, is an A^ω-pattern. By Lemma 3.1 it follows that there are $s_1, s_2 \in \beta$ such that $s_1 \ll_* s_2$ and $f'(s_1) \preceq^\omega f'(s_2)$, and hence $f(s_1) \preceq^* f(s_2)$. This means that f is good.

4. Follows from 3. □

Since bqo is a stronger relation than wqo (property 1), it follows by Lemma 2.1 that, to prove termination of the reachability algorithm of Section 2, it is sufficient to prove bqo of constraints under entailment. All constraint systems derived earlier in the literature based on the approach of [AČJYK96, AJ01, FS98]) use properties 2, 3, and 4. This implies that all these constraint systems are also bqos. An immediate consequence of property 5 is that bqo of a set of constraints implies bqo of disjunctions of these constraints.

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

1. We show better quasi-ordering of a constraint system $\mathbb{C}_1$ using properties 2, 3, and 4 in Theorem 4.1 (following a similar methodology to that described in [AČJYK96, AJ01, FS98]).
2. We use property 5 to derive better quasi-ordering of a new more compact constraint system $\mathbb{C}_2$ defined as disjunctions of constraints in $\mathbb{C}_1$.

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

As mentioned earlier, an important difference compared to the approach of [AČJYK96, AJ01, FS98]) is that Step 2 (taking disjunction) cannot be performed within that framework. This is illustrated by the following example, which shows that wqos in general are not closed under disjunction.

Example 4.2. [Rado's Example] Consider the the set $X = \{(a,b) \mid a < b\} \subseteq \mathcal{N}^2$. Define a set $\mathbb{C}_1 = \{\phi_{a,b} \mid (a,b) \in X\}$ of constraints, such that the denotation $[\![\phi_{a,b}]\!] \subseteq X$ of $\phi_{a,b}$ is the set $\{(c,d) \mid (c > b) \vee ((c = a) \wedge (d \geq b))\}$. It is straightforward to check that $\mathbb{C}_1$ is wqo: suppose that we have 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 $(a_1,b_1), (a_2,b_2), \ldots$. First, we show that $a_j \leq b_1$ for all $j \geq 1$. If this is not the case, then let $b_1 < a_j$. We show that this implies $\phi_{a_1,b_1} \preceq \phi_{a_j,b_j}$ which is a contradiction. Take any $(c,d) \in [\![\phi_{a_j,b_j}]\!]$. Then, either $c > b_j$ or $(c = a_j) \wedge (d \geq b_j)$. In both cases, we show that $c > a_1$ and hence $(c,d) \in [\![\phi_{a_1,b_1}]\!]$.

- $c > b_j$. We have $b_j > a_j$ and $b_1 > a_1$ by definition, and $a_j > b_1$ by assumption. It follows that $c > a_1$.
- $(c = a_j) \wedge (d \geq b_j)$. We know that $b_1 > a_1$ by definition, and $a_j > b_1$ by assumption. It follows that $c > a_1$.

Since $a_j \leq b_1$ for all $j \geq 1$, we have a subsequence of the form $(a, b_{i_1}), (a, b_{i_2}), \ldots$, and hence there are k and ℓ such that $b_{i_k} \leq b_{i_\ell}$ which is a contradiction.

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

□

5 Timed Petri Nets

We consider *Timed Petri Nets (TPNs)* where each token is equipped with a real-valued clock representing the "age" of the token. The firing conditions of a transition include the usual ones for Petri nets. Furthermore, each arc between a place and a transition 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 let $\mathcal{Z}$ and $\mathcal{R}^{\geq 0}$ denote the sets of integers, and nonnegative reals respectively. Recall that $\mathcal{N}$ denotes the set of natural numbers.

We also recall that $A^{\oplus}$ denotes the set of finite multisets over A. Often, we view a multiset B over a set A as a mapping from A to $\mathcal{N}$. Sometimes we write multisets as lists, so e.g. $(2.4, 5.1, 5.1, 2.4, 2.4)$ represents a multiset 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 $\left(2.4^3, 5.1^2\right)$. For multisets 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 multiset 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 multiset B where $B(a) = B_2(a) - B_1(a)$, for each $a \in A$. We use $\emptyset$ to denote the empty multiset, i.e., $\emptyset(a) = 0$ for each $a \in A$.

We use a set *Intrv* of *intervals*. An open interval is written as (w, z) where $w \in \mathcal{N}$ and $z \in \mathcal{N} \cup \{\infty\}$. Intervals can also be closed in one or both directions, e.g. $[w, z)$ is closed to the left. 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* and $In, Out : T \times P \mapsto Intrv^{\oplus}$. 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.

Markings A *marking* M of N is a finite multiset over $P \times \mathcal{R}^{\geq 0}$. The marking M defines numbers and ages of the 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 multiset $M(p)$ 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 $[0, \infty)$.

Transition Relation We define two types of transition relations on markings. A *timed transition* increases the age of all tokens by the same real number. Formally $M_1 \longrightarrow_{Time} M_2$ 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* $\longrightarrow_{Disc}$ 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 multisets $B_1 = (x_1, \ldots, x_m)$ 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 $i : 1 \leq i \leq n$.
- $M_2(p) = (M_1(p) - B_1) + B_2$.

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. We define the relation $\longrightarrow$ to be $\longrightarrow_{Time} \cup \longrightarrow_{Disc}$.

For a set M of markings we let $Pre(\mathsf{M})$ denote the set $\{M \mid \exists M' \in \mathsf{M}.\ M \longrightarrow M'\}$, i.e., $Pre(\mathsf{M})$ is the set of markings from which we can reach a marking in M through the application of a single (timed or discrete) transition.

A set M of markings is said to be *upward closed* if it is the case that $M \in \mathsf{M}$ and $M \leq M'$ imply $M' \in \mathsf{M}$.

Coverability The *coverability problem* is defined as follows: Given a TPN N, a marking M_{init} of N, and an upward closed set M_{fin} of markings of N, is there an $M \in \mathsf{M}_{fin}$ such that $M_{init} \xrightarrow{*} \mathsf{M}$?

Using standard techniques [VW86, GW93], we can show that checking several classes of safety properties for TPNs can be reduced to the coverability problem. In the next section, we define constraints called *existential zones*. In our reachability algorithm, we use an existential zone to characterize the set M_{fin}.

Example 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.

Remark 1 For simplicity of presentation we use only non-strict inequalities. All the results can be generalized in a straightforward manner to include the more general case, where we also allow strict inequalities.

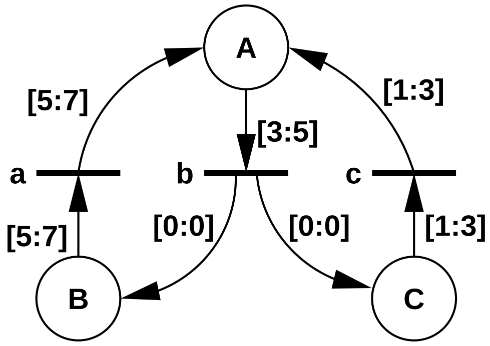

Figure 2: A small timed Petri net.

Remark 2 Notice that, in our definition of the operational behaviour of TPNs, we assume a lazy (non-urgent) behaviour of the net. This means that we may choose to "let time pass" instead of firing enabled transitions, even if that makes transitions disabled due to some of the needed tokens becoming "too old". Tokens that are too old to participate in firing transitions are usually called dead tokens. In an urgent TPN, timed transitions that cause dead tokens are not allowed. This means that the set of transitions of an urgent TPN is a subset of the set of transitions of the corresponding lazy TPN. Therefore, if a set of markings is not reachable in the lazy TPN it is not reachable in the urgent TPN either. In other words safety properties that hold for the lazy TPN also hold for the urgent TPN.

6 Existential Zones

In this section we introduce a constraint system called *existential zones*. Intuitively, an existential zone characterizes an upward closed set of markings. An existential zone Z represents minimal conditions on markings. More precisely, Z specifies a minimum number of tokens which should be in the marking, and then imposes certain conditions on these tokens. The conditions are formulated as specifications of the places in which the tokens should reside and restrictions on their ages. The age restrictions are stated as bounds on values of clocks, and bounds on differences between values of pairs of clocks. A marking M which satisfies 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 by the marking).

For a natural number n, we let $\underline{n}^0$ denote the set $\{0, 1, 2, \ldots, n\}$, and let $\underline{n}^1$

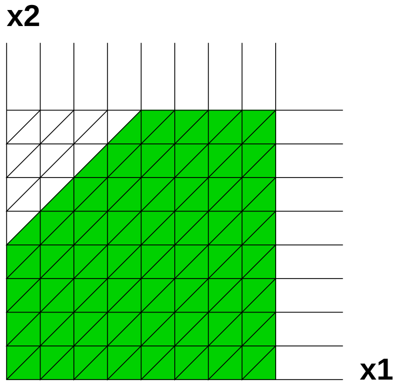

Figure 3: Example of restrictions on ages of tokens.

denote the set $\{1, 2, \ldots, n\}$. We assume a TPN (P, T, In, Out).

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

Consider the example from Section 5. 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 e.g. $D(0, i) = 0$ 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 an illustration of the age restrictions of Z.

Consider a marking $M = ((p_1, x_1), \ldots, (p_n, x_n))$ and an injection $h : \underline{m}^1 \to \underline{n}^1$ (called 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 $i, j \in \underline{m}^1$ with $i \neq j$.
- $x_{h(i)} \leq D(i,0)$ and $-D(0,i) \leq x_{h(i)}$, for each $i \in \underline{m}^1$.

We say that M *satisfies* Z, written $M \models Z$, if $M, h \models Z$ for some h. 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 $[\![Z]\!] = \{M \mid M \models Z\}$. Notice that the value of $D(i,i)$ is irrelevant for the satisfiability of Z.

Membership From the above definitions the following lemma is straightforward.

Lemma 6.1. For an existential zone Z and a marking M, it is decidable whether $M \models Z$.

Upward Closedness 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 $M' \models Z$.

Normal and Consistent Existential Zones An existential zone $Z = \left(m, \bar{P}, D\right)$ is said to be *normal* if for each $i, j, k \in \underline{m}^0$, we have $D(j,i) \leq D(j,k) + D(k,i)$. It is easy to show the following.

Lemma 6.2. For each existential zone Z there is a unique (up to renaming of the index set) normal existential zone, written $\widetilde{Z}$, such that $[\![\widetilde{Z}]\!] = [\![Z]\!]$.

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

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

6.1 Computing Entailment

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 $Z_1 = \left(m, \bar{P}_1, D_1\right)$ and $Z_2 = \left(n, \bar{P}_2, D_2\right)$. 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 $\left\{v_i \mid i \in \underline{n}^1\right\}$. Let H be the set of injections from $\underline{m}^1$ to $\underline{n}^1$ such that $h \in H$ if and only if $\bar{P}_1(i) = \bar{P}_2(h(i))$ for each $i \in \underline{m}^1$. 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

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

This gives the following.

Lemma 6.3. 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.

6.2 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. 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. In other words $Pre[\![Z]\!] = [\![Z_1]\!] \cup \cdots \cup [\![Z_m]\!]$. We define $Pre = Pre_{Disc} \cup Pre_{Time}$, where Pre_{Disc} corresponds to firing transitions backwards and Pre_{Time} corresponds to running time backwards.

We define $Pre_{Disc} = \cup_{t \in T} Pre_t$, 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 $Z = \left(m, \bar{P}, D\right)$, and a timed Petri net $N = (P, T, In, Out)$. From Lemma 6.2

we know that assuming Z to be normal does not affect the generality of our results.

For an interval $\mathcal{I} = [a, b]$, and $i \in \underline{m}^1$, 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 $j, k \in \underline{m}^1$ 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 $j \in \underline{m}^1$.
- $\bar{P}'(m + 1) = p$.
- $D'(k, j) = D(k, j)$, for each $j, k \in \underline{m}^0$, and $\bar{P}'(j) = \bar{P}(j)$, for each $j \in \underline{m}^1$.

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

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

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

- $D'(j,k) = D(j,k)$, for each $j,k \in \underline{(i-1)}^0$.
- $D'(j,k) = D(j,k+1)$ and $D'(k,j) = D(k+1,j)$, for each $j \in \underline{(i-1)}^0$ 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 $j \in \underline{(i-1)}^0$, 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

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

the abstraction $Z \backslash 2$ is the zone

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

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

Now, we are ready to define Pre.

Lemma 6.4. Consider a TPN $N = (P, T, In, Out)$, a transition $t \in T$, and an existential zone $Z = \big(m, \bar{P}, D\big)$. 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 $h : \underline{m}^1 \longrightarrow \underline{\ell}^1$ 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 $j \in \underline{n}^1$
- $Z \otimes \left(\mathcal{J}_{h(i_1)}, i_1\right) \otimes \cdots \otimes \left(\mathcal{J}_{h(i_n)}, i_n\right)$ 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 6.5. For an existential zone $Z = \left(m, \bar{P}, D\right)$, the set $Pre_{Time}(Z)$ is the existential zone $Z' = \left(m, \bar{P}, D'\right)$,where $D'(0,i) = 0$ and $D'(j,i) = D(j,i)$ if $j \neq 0$, for each $i, j \in \underline{m}^0$, with $i \neq j$.

From Lemma 6.4 and Lemma 6.5 we get the following.

Lemma 6.6. For an existential zone Z, the set $Pre(Z)$ is computable.

6.3 BQO

In order to prove that existential zones are bqo we recall a constraint system related to existential zones, namely that of *existential regions* introduced in [AJ98]. Let *cmax* be the largest natural number which appears in the intervals of the given TPN (excluding ∞). An existential region is a list of multisets $(B_0, B_1, \ldots, B_n, B_{n+1})$ where $n \geq 0$ and B_i is a multiset over $P \times \underline{cmax}^0$. In a similar manner to existential zones, an existential region R defines a set of conditions which should be satisfied by a configuration γ in order for γ to satisfy R. Intuitively B_0 represents tokens with ages which have fractional parts equal to 0. The multisets $B_1, \ldots, B_n$ 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 B_i have a fractional part that is strictly less than the fractional part of the ages of those in B_{i+1}. Finally the multiset B_{n+1} represents tokens with ages greater than *cmax* (regardless of their fractional parts).

BQO of existential zones follows from the following two arguments:

1. Existential regions are built starting from finite domains, and repeatedly building finite words, multisets, and sets. From the properties mentioned above, it follows that existential regions are bqo.

2. For each existential zone Z, there is a finite set R of existential regions such that $Z \equiv \bigvee \mathsf{R}$. Since bqo is closed under union, it follows that existential zones are bqo.

This implies the following:

Lemma 6.7. Existential zones are bqo (and hence wqo).

7 Experimental Results

We have implemented a prototype to perform 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].

7.1 Existential CDDs and DDDs

Clock Difference Diagrams (CDDs) [BLP+99] and *Difference Decision Diagrams (DDDs)* [MLAH99] are constraint systems that have been invented to give representations of real-time systems that 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 6). $(\mathtt{V}, \mathtt{E})$ 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 out-degrees of $\mathtt{v}^0$ and $\mathtt{v}^1$ are zero while the out-degrees of the rest of vertices are two. Each vertex $\mathtt{v} \in \mathtt{V} \setminus \left\{\mathtt{v}^0, \mathtt{v}^1\right\}$ has the following attributes: $pos(\mathtt{v}), neg(\mathtt{v}) \in \underline{m}^0$, $op(\mathtt{v}) \in \{<, \leq\}$, $const(\mathtt{v}) \in \mathcal{Z}$, 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 $\mathtt{v} \in \mathtt{V} - \left\{\mathtt{v}^0, \mathtt{v}^1\right\}$. 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 $Y = \left(m, \bar{P}, \mathtt{V}, \mathtt{E}\right)$, a vertex $\mathtt{v} \in \mathtt{V}$, a marking $M = ((p_1, x_1), \ldots, (p_n, x_n))$ and an injection $h : \underline{m}^1 \rightarrow \underline{n}^1$. 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 $i \in \underline{m}^1$, and either

- $\mathtt{v} = \mathtt{v}^1$; or

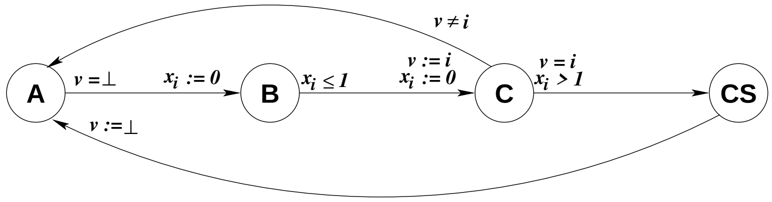

Figure 4: Fischer's Protocol for Mutual Exclusion

- $\left(\left(\begin{array}{c} x_{h(pos(\mathtt{v}))} \\ - \\ x_{h(neg(\mathtt{v}))} \end{array}\right) \sim const(\mathtt{v})\right) \rightarrow \left(\begin{array}{c} M, h \models (Y, high(\mathtt{v})) \\ , \\ M, h \models (Y, low(\mathtt{v})) \end{array}\right)$

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

As with existential zones, we can modify the operations defined in [MLAH99] to compute predecessors of existential DDDs with respect to transitions of a timed Petri net. 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 Z of existential zones such that $[\![Y]\!] = [\![\bigvee \mathsf{Z}]\!]$. 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. From Lemma 6.7, Theorem 4.1 (Property 5), and the fact that each existential DDD is the disjunction of a finite set of existential zones we get the following result.

Lemma 7.1. Existential DDDs are better quasi-ordered (and hence also well quasi-ordered).

7.2 Fischer's Protocol

We will now describe a timed Petri net model of a parameterized version of Fischer's protocol [SBM92]. 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. [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 that is graphically described in Figure 4. Each process i has a local clock, x_i, and a control state that assumes values in the set $\{A, B, C, CS\}$ where A is the initial state

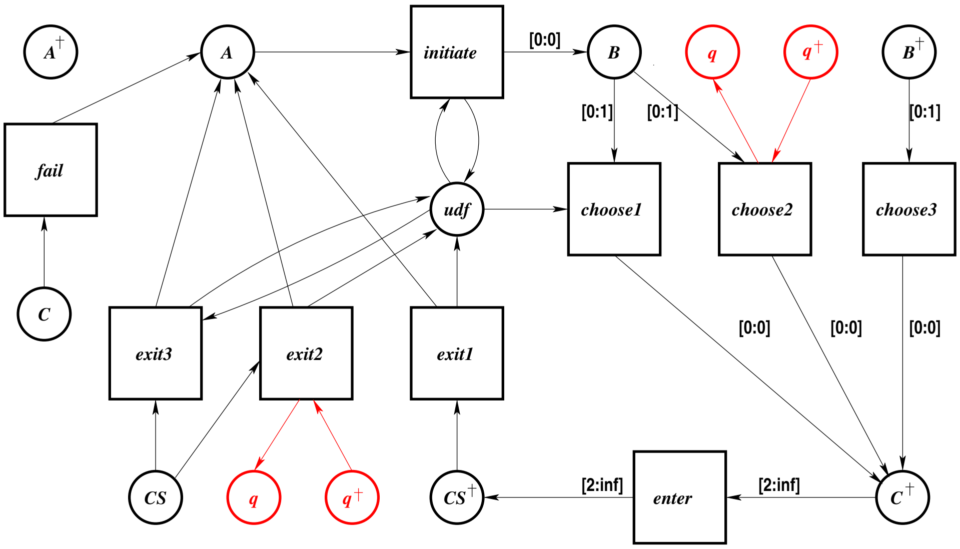

Figure 5: Timed Petri net model of Fischer's Protocol for Mutual Exclusion

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 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 that 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 timed Petri net formalism. The processes running the protocol are modelled 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 modelled 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. The critical section is modelled by the places CS and $CS^\dagger$, so mutual exclusion is satisfied when the number of tokens in those places is less than two.

7.3 Results

We have used our prototype to analyze the parameterized version of Fischer's protocol presented above. 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.

8 Broadcast Protocols

We consider *broadcast protocols*, which consist of an arbitrary number of identical finite-state processes, communicating through rendezvous or through broadcast. We assume a finite set $\{s_1, \ldots, s_n\}$ of *states*, and a set $\{x_1, \ldots, x_n\}$ of variables which range over the natural numbers. A *configuration* γ of a protocol is a tuple $(a_1, \ldots, a_n)$ of natural numbers, where a_i represents the number of processes which are in the state s_i. In [EFM99] a constraint system, which we here refer to as $\mathbb{B}$, is defined, where each constraint is a tuple $(b_1, \ldots, b_n)$ with a denotation $[\![(b_1, \ldots, b_n)]\!]$ which is the upward closed set $\{(a_1, \ldots, a_n) \mid (b_1, \ldots, b_n) \leq (a_1, \ldots, a_n)\}$. In [DEP99] several new constraint systems for broadcast protocols are proposed, and compared with regard to the efficiency parameters mentioned in Section 4. The most general of these constraint systems , called **AD** in [DEP99], consists of conjunctions of constraints each of the form $x_{i_1} + \cdots + x_{i_k} \geq b$, where $x_{i_1}, \ldots, x_{i_k}$ are distinct variables of $\{x_1, \ldots, x_n\}$. Two special cases are considered: **NA** where k is always equal to 1, and **DV** where the set of variables occurring in the different conjuncts are assumed to be disjoint. Since these new constraint systems are not constructed applying the basic set of constraint operations (described in Section 4), a separate proof of termination is required for them.

Applying the method of Section 4 we can show bqo of **AD**, **NA**, and **DV** uniformly as follows. From properties 2 and 3 in Theorem 4.1, it follows that

$\mathbb{B}$ is bqo. Furthermore, it is straightforward to show that each constraint in **AD**, **NA**, and **DV** is equivalent to the disjunction of a finite set of constraints in $\mathbb{B}$. From property 5 of Theorem 4.1, we get

Theorem 8.1. **AD**, **NA**, and **DV** are bqo.

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

9 Lossy Channel Systems

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

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 $[\![a]\!] = \{w |\ a \preceq_w w\}$;

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

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

Example 9.1. In Figure 6, the constraint ϕ_1 is of the form $(a \ \& \ b) \bullet (b + c)$. This means that $[\![\phi_1]\!] = \{w_1 w_2 |\ (a \preceq_w w_1) \text{ and } (b \preceq_w w_1) \text{ and } ((b \preceq_w w_2) \text{ or } (c \preceq_w w_1))\}$. The constraint ϕ_1 is equivalent to the disjunction of the following set of constraints in $\mathbb{L}_1$: $\{abb, abc, bab, bac\}$. □

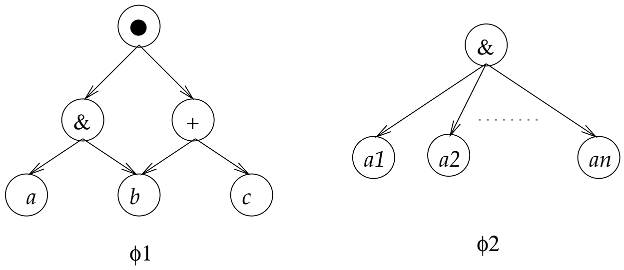

Figure 6: Two constraints in $\mathbb{L}_2$

.

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
$\{b_1 \bullet \cdots \bullet b_n | \ (b_1, \ldots, b_n)$ is a permutation of $(a_1, \ldots, a_n)\}$.

In a similar manner to Section 6 and Section 8 we can use properties of $\mathbb{L}_1$ and $\mathbb{L}_2$ to conclude the following

Theorem 9.2. $\mathbb{L}_2$ is bqo.

10 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 $\mathcal{Z}$ 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 $c \in \mathcal{Z}$; and where the body *stmt* contains, for each $x \in X$, an assignment of one of the forms $x := y$, $x := c$, or $x := \{?\}$, for $y \in X$ and $c \in \mathcal{Z}$. The assignment $x := \{?\}$ is a "read" operation putting an arbitrary integer into the variable x. A *configuration* γ of an IRA is a mapping from X to $\mathcal{Z}$. Sometimes, we write γ as a tuple $(\gamma(x_1), \ldots, \gamma(x_n))$. For $c \in \mathcal{Z}$, we use the convention that $\gamma(c) = c$.

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

Example 10.1. Assume $X = \{x_1, x_2, x_3\}$ and $C = \{5\}$. Consider a constraint $\phi = (10, 5, 12)$, then $\gamma_1 = (12, 5, 17) \in [\![\phi]\!]$, while $\gamma_2 = (8, 5, 16) \notin [\![\phi]\!]$ (since $\phi(x_1) - \phi(x_2) = 5 \not\leq \gamma_2(x_1) - \gamma_2(x_2) = 3$), and $\gamma_3 = (12, 4, 17) \notin [\![\phi]\!]$ (since $\phi(5) = 5 \leq \phi(x_2) = 5$ while $\gamma_3(5) = 5 \not\leq \gamma_3(x_2) = 4$). $\square$

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

Example 10.2. Assume $X = \{x_1, x_2\}$ and $C = \{5\}$. The constraint $5 < x_2$ in $\mathbb{S}_2$ is equivalent to the disjunction of the following set of constraints in $\mathbb{S}_1$:
$\{(4,7), (5,7), (6,7), (7,7), (8,7)\}$. Notice that the constraints correspond to the different relative values which x_1 may have with respect to the constant 5 and the variable x_2. $\square$

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

Theorem 10.3. $\mathbb{S}_2$ is bqo.

11 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 words, 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 the operation of taking infinite sets and infinite words. 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 Pritha Mahata for comments on earlier versions of the paper. Special thanks to Jesper Møller and Henrik Reif Andersen for letting us use their DDD implementation.

References

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

[AJ93] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. In *Proc. LICS' 93 8^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 160–170, 1993.

[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.

[AJ01] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256:145–167, 2001.

[AJ03] Parosh Aziz Abdulla and Bengt Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*, 290(1):241–264, 2003.

[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, Languages, 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.

[Del00] G. Delzanno. Automatic verification of cache coherence protocols. In Emerson and Sistla, editors, *Proc.* 12^{th} *Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 53–68. Springer Verlag, 2000.

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

[dFERA00] D. de Frutos Escrig, V. Valero Ruiz, and O. Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *ICATPN 2000*, volume 1825, pages 187–206, 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.

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

[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.

[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*, number 965 in 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.

[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.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming, Turin*, volume 137 of *Lecture Notes in Computer Science*, pages 337–352. Springer Verlag, 1982.

[RFC99] V. Valero Ruiz, D. De Frutos, and F. Cuartero. On Non-Decidability of reachability for Time Arcs Petri Nets. In *Proceedings . of XXth PNPM.* IEEE. Computer Society, 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.