Constraint-based Analysis of Broadcast Protocols

Giorgio Delzanno[1], Javier Esparza[2], and Andreas Podelski[1]

[1] Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany
e-mail: {delzanno | podelski}@mpi-sb.mpg.de
[2] Technische Universität Munchen, Arcisstr. 21, 80290 München, Germany
esparza@informatik.tu-muenchen.de

Abstract. Broadcast protocols are systems composed of a finite but arbitrarily large number of processes that communicate by rendezvous (two processes exchange a message) or by broadcast (a process sends a message to all other processes). The paper describes an optimized algorithm for the automatic verification of safety properties in broadcast protocols. The algorithm checks whether a property holds for any number of processes.

1 Introduction

Broadcast protocols [EN98] are systems composed of a finite but arbitrarily large number of processes that communicate by rendezvous (two processes exchange a message) or by broadcast (a process sends a message to all other processes). They are a natural model for problems involving readers and writers, such as cache-coherence problems.

From a mathematical point of view, broadcast protocols can be regarded as an extension of vector addition systems or Petri nets. Their operational semantics is a transition system whose states are tuples of integers. Moves between transitions are determined by a finite set of affine transformations with guards. Vector Addition Systems correspond to the particular case in which the matrix of the affine transformation is the identity matrix.

In [EFM99], Esparza, Finkel and Mayr show that the problem of deciding whether a broadcast protocol satisfies a safety property can be reduced to a special reachability problem, and using results by Abdulla *et al.*, [ACJ+96] (see also [FS98]), they prove that this problem is decidable. They propose an abstract algorithm working on infinite sets of states. The algorithm starts with the set of states to be reached, and repeatedly adds to it the set of its immediate predecessors until a fixpoint is reached.

As shown e.g. in [Kin99,DP99], linear arithmetic constraints can be used to finitely represent infinite sets of states in integer valued systems. Symbolic model checking algorithms can be defined using the 'satisfiability' and the 'entailment' test to symbolically compute the transitive closure of the *predecessor* relation defined over sets of states. However, in order to obtain an efficient algorithm it is crucial to choose the right format for the constraints.

In this paper we discuss different classes of constraints, and propose *linear constraints with disjoint variables* as a very suitable class for broadcast protocols. We show that the operations of computing the immediate predecessors and checking if the fixpoint has been reached can both be efficiently implemented. We also propose a compact data structure for these constraints.

We have implemented a specialized checker based on our ideas, and used it to define a symbolic model checking procedure for broadcast protocols. As expected, the solver leads to a significant speed-up with respect to procedures using general purpose constraint solvers (HyTech [HHW97] and Bultan, Gerber and Pugh's model checker based on the Omega library [BGP97]). We present some experimental results for both broadcast protocols and weighted Petri Nets.

2 Broadcast Protocols: Syntax and Semantics

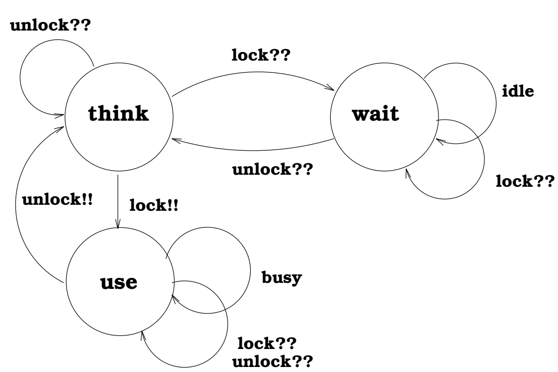
2.1 Syntax

A *broadcast protocol* is a triple (S, L, R) where

- S is a finite set of *states*.
- L is a set of *labels*, composed of a set Σ_l of *local* labels, two sets $\Sigma_r \times \{?\}$ and $\Sigma_r \times \{!\}$ of *input* and *output rendez-vous* labels, and two sets $\Sigma_b \times \{??\}$ and $\Sigma_b \times \{!!\}$ of *input* and *output broadcast* labels, where $\Sigma_l, \Sigma_r, \Sigma_b$ are disjoint finite sets. The elements of $\Sigma = \Sigma_l \cup \Sigma_r \cup \Sigma_b$ are called *actions*.
- $R \subseteq S \times L \times S$ is a set of *transitions* satisfying the following property: for every $a \in \Sigma_b$ and every state $s \in S$, there exists a state $s' \in S$ such that $s \xrightarrow{a??} s'$. Intuitively, this condition guarantees that a process is always willing to receive a broadcasted message.

We denote $(s, l, s) \in R$ by $s \xrightarrow{l} s'$. The letters $a, b, c, \ldots$ denote actions. Rendezvous and broadcast labels like $(a, ?)$ or $(b, !!)$ are shortened to $a?$ and $b!!$. We restrict our attention to broadcast protocols satisfying the following additional conditions: (i) for each state s and each broadcast label $a??$ there is exactly one state s' such that $s \xrightarrow{a??} s'$ (determinism); (ii) each label of the form a, $a!$, $a?$ and $a!!$ appears in exactly one transition.

Consider the following example:



The finite-state automata in the figure models the behaviour of a system of identical processes that race for using a shared resource. Initially, all processes are in the state **think**. Before accessing its own critical section, a process broadcasts the request `lock!!`. In reply to the broadcast (`lock??`) the remaining processes are forced to move to the state **wait** (an abstraction of a queue). After using the resource, the process in the critical section broadcasts the message `unlock!!` in order to restore the initial configuration. The key point here is that the description of the protocol is independent of the number of processes in the network.

2.2 Semantics

Let $B = (S, L, R)$ be a broadcast protocol, and let $S = \{s_1, \ldots, s_n\}$. A *configuration* is a vector $\mathbf{c} = \langle c_1, \ldots, c_n \rangle$ where c_i denotes the number of processes in state s_i for $i : 1, \ldots, n$.

Moves between configurations are either local (a process moves in isolation to a new state), rendezvous (two processes exchange a message and move to new states), or broadcasts (a process sends a message to all other processes; all processes move to new states). Formally, the possible moves are the smallest subset of $\mathbb{N}^n \times \Sigma \times \mathbb{N}^n$ satisfying the three conditions below, where u_i denotes the configuration such that $u_i(s_i) = 1$ and $u_i(s_j) = 0$ for $j \neq i$, and where $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ denotes $(\mathbf{c}, a, \mathbf{c}') \in R$.

- If $s_i \xrightarrow{a} s_j$, then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) \geq 1$ and $\mathbf{c}' = \mathbf{c} - \mathbf{u}_i + \mathbf{u}_j$.
  I.e. one process is removed from s_i, and one process is added to s_j.
- If $s_i \xrightarrow{a!} s_j$ and $s_k \xrightarrow{a?} s_l$, then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) \geq 1$, $\mathbf{c}(s_k) \geq 1$ and $\mathbf{c}' = \mathbf{c} - \mathbf{u}_i - \mathbf{u}_k + \mathbf{u}_j + \mathbf{u}_l$.
  I.e. one process is removed from s_i and s_k, and one process is added to s_j and s_l.
- If $s_i \xrightarrow{a!!} s_j$, then $\mathbf{c} \xrightarrow{a} \mathbf{c}'$ for every $\mathbf{c}$, $\mathbf{c}'$ such that $\mathbf{c}(s_i) \geq 1$ and $\mathbf{c}'$ can be computed from $\mathbf{c}$ in the following three steps:

$$\mathbf{c}_1 = \mathbf{c} - \mathbf{u}_i \tag{1}$$

$$\mathbf{c}_2(s_k) = \sum_{\{s_l \mid s_l \xrightarrow{a??} s_k\}} \mathbf{c}_1(s_l) \tag{2}$$

$$\mathbf{c}' = \mathbf{c}_2 + \mathbf{u}_j \tag{3}$$

  I.e. the sending process leaves s_i (1), all other processes receive the broadcast and move to their destinations (2), and the sending process reaches s_j (3).

Thanks to the conditions (i) and (ii) of Section 2.1, the configuration $\mathbf{c}'$ is completely determined by $\mathbf{c}$ and the action a.

We denote by $\preceq$ the pointwise order between configurations, i.e. $\mathbf{c} \preceq \mathbf{c}'$ if and only if $\mathbf{c}(s_i) \leq \mathbf{c}'(s_i)$ for every $i : 1, \ldots, n$. A *parameterized configuration* is a *partial* function $\mathbf{p}: S \rightarrow \mathbb{N}$. Loosely speaking, $\mathbf{p}(s) = \perp$ denotes that the number of processes on state s is arbitrary. Formally, a parameterised configuration denotes a set of configurations, namely those extending $\mathbf{p}$ to a total function.

2.3 Checking safety properties

In this paper we study the *reachability* problem for broadcast protocols, defined as follows:

> Given a broadcast protocol B, a parameterized initial configuration $\mathbf{p}_0$ and a set of configurations C, can a configuration $\mathbf{c} \in C$ be reached from one of the configurations of $\mathbf{p}_0$?

In [EFM99] this problem is shown to be decidable for upwards-closed sets C.[1] A set C is *upwards-closed* if $\mathbf{c} \in C$ and $\mathbf{c}' \succeq \mathbf{c}$ implies $\mathbf{c}' \in C$. The mutual exclusion property of the example in the introduction can be checked by showing that no configuration satisfying $Use \geq 2$ (an upwards-closed set) is reachable from an initial configuration satisfying $Wait = 0, Use = 0$. It is shown in [EFM99] that the model-checking problem for safety properties can be reduced to the reachability problem for upwards-closed sets. (Here we follow the automata-theoretic approach to model-checking [VW86], in which a safety property is modelled as a regular set of dangerous sequences of actions the protocol should *not* engage in.)

The algorithm of [EFM99] for the reachability problem in the upwards-closed case is an "instantiation" of a general backwards reachability algorithm presented in [ACJ+96] (see also [FS98]). Define the *predecessor* operator as follows:

$$pre(C) = \{\mathbf{c} \mid \mathbf{c} \xrightarrow{a} \mathbf{c}',\ \mathbf{c}' \in C\}.$$

I.e., *pre* takes a set of configurations C_0, and delivers its set of *immediate* predecessors. The algorithm repeatedly applies the predecessor operator until a fixpoint is reached, corresponding to the set of all predecessors of C_0. If this set contains some initial configurations, then C_0 is reachable.

Proc Reach(C_0 : upwards-closed set of configurations)
  $C := C_0$;
  repeat
    $old\_C := C$;
    $C := old\_C \cup pre(old\_C)$;
  until $C = old\_C$;
  return C

The algorithm works because of the following properties: (i) if C is upwards-closed, then so is $pre(C)$; (ii) the set of minimal elements of an upwards-closed set with respect to the pointwise order is finite (see also Section 4); (iii) the repeat loop terminates. To prove property (i), we observe that we can associate to each label $a \in \Sigma$ [EFM99]:

- The set of configurations Occ_a from which a can occur.
  In the case of local moves and broadcasts there is a state s_i such that $Occ_a = \{\mathbf{c} \mid \mathbf{c}(s_i) \geq 1\}$. In the case of rendezvous there are states s_i, s_j such that $Occ_a = \{\mathbf{c} \mid \mathbf{c}(s_i) \geq 1 \text{ and } \mathbf{c}(s_j) \geq 1\}$.

[1] On the other hand, the problem is undecidable for singleton sets!.

– An affine transformation $\mathbf{T_a}(\mathbf{x}) = \mathbf{M_a} \cdot \mathbf{x} + \mathbf{b}_a$ such that if $\mathbf{c} \xrightarrow{a} \mathbf{c}'$, then $\mathbf{c}' = \mathbf{T_a}(\mathbf{c})$.
  $\mathbf{M_a}$ is a matrix whose columns are unit vectors, and $\mathbf{b}$ is a vector of integers. (Actually, the components of $\mathbf{b}$ belong to $\{-1, 0, 1\}$, but our results can be extended without changes to the case in which they are arbitrary integer numbers. An example is discussed in Section 8.)

It follows that $pre(C)$ can be computed by the equation

$$pre(C) = \bigcup_{a \in \Sigma} (Occ_a \cap \mathbf{T_a^{-1}}(C)) \tag{4}$$

Hence if C is upwards-closed then so is $pre(C)$. Properties (ii) and (iii) are an immediate consequence of the well-known

Lemma 1 (Dickson's Lemma). *Let $\mathbf{v}_1, \mathbf{v}_2, \ldots$ be an infinite sequence of elements of $I\!N^k$. There exists $i < j$ such that $\mathbf{v}_i \preceq \mathbf{v}_j$ (pointwise order).*

The only known upper-bound for the number of iterations until termination is non-primitive recursive [McA84]. However, despite this result, the algorithm can still be applied to small but interesting examples.

3 Symbolic Representation via Constraints

A *linear arithmetic constraint* (or *constraint* for short) is a (finite) first-order formula $\phi_1 \wedge \ldots \wedge \phi_n$. with free variables (implicitly existentially quantified), and such that each ϕ_i is an atomic formula (constraint) built over the predicates $=, \geq, \leq, >, <$ and over arithmetic expressions (without multiplication between variables) built over $+, -, *, 0, 1$, etc.

The solutions (assignments of values to the free variables that make the formula true) of a constraint ϕ over the domain $\mathcal{D}$ are denoted by $[\![\phi]\!]_{\mathcal{D}}$. In the sequel we always take $\mathcal{D} = \mathbb{Z}$, and abbreviate $[\![\phi]\!]_{\mathbb{Z}}$ to $[\![\phi]\!]$. We often represent the *disjunction* of constraints $\phi_1 \vee \ldots \vee \phi_n$ as the *set* $\{\phi_1, \ldots, \phi_n\}$.

Constraints can be used to symbolically represent sets of configurations of a broadcast protocol. Given a protocol with states $\{s_1, \ldots, s_n\}$, let $\mathbf{x} = x_1, \ldots, x_n$ be a vector of variables, where x_i is intended to stand for the number of processes currently in state s_i. We assume that variables range over positive values (i.e., each variable x_i comes with an implicit constraint $x_i \geq 0$). A configuration $\mathbf{c} = \langle c_1, \ldots, c_n \rangle$ is simply represented as the constraint $\bigwedge_{i=1}^n x_i = c_i$. A parametric configuration $\mathbf{p} = \langle p_1, \ldots, p_n \rangle$ is represented as the constraint $\bigwedge_{i=1}^n \phi_i$ where: if $p_i \in I\!N$ then ϕ_i is the atomic constraint $x_i = c_i$, and if $p_i = \bot$ then ϕ_i is the atomic constraint $x_i \geq 0$.

As an example, the flow of processes caused by the *lock* broadcast in the protocol of the introduction is described by the inequality below (where, for clarity, we use *Think*, *Wait*, *Use* instead of x_1, x_2, x_3 and we omit the equalities of the form $x_i' = x_i$).

$$Think \geq 1 \wedge Think' = 0 \wedge Wait' = Think + Wait - 1 \wedge Use' = Use + 1$$

Let $\mathcal{C}$ be a class of constraints denoting exactly the upwards-closed sets, i.e., if a set S is upwards-closed then there is a set of constraints $\Phi \subseteq \mathcal{C}$ such that $[\![\Phi]\!] = S$, and viceversa. We can use any such class $\mathcal{C}$ to derive a symbolic version Symb-Reach$_\mathcal{C}$ of the procedure Reach:

Proc Symb-Reach$_\mathcal{C}$(Φ_0 : set of constraints of $\mathcal{C}$)
  $\Phi := \Phi_0$;
  repeat
    $old\_\Phi := \Phi$;
    $\Phi := old\_\Phi \cup \mathbf{pre}_\mathcal{C}(old\_\Phi)$;
  until Entail$_\mathcal{C}(\Phi, old_\Phi)$;
  return Φ

where (a) $\mathcal{C}$ is closed under application of $\mathbf{pre}_\mathcal{C}$, (b) $[\![\mathbf{pre}_\mathcal{C}(\Phi)]\!] = pre([\![\Phi]\!])$, and (c) Entail$_\mathcal{C}(\Phi, \Psi) = true$ if and only if $[\![\Phi]\!] \subseteq [\![\Psi]\!]$.

Condition (b) on $\mathbf{pre}_\mathcal{C}$ can be reformulated in syntactic terms. Let Φ be a set of constraints, and for each action a let G_a be a constraint such that $[\![G_a]\!] = Occ_a$ (we call G_a the *guard* of the action a). We have $\mathbf{T_a^{-1}}([\![\Phi]\!]) = [\![\Phi[\mathbf{x}/\mathbf{T_a}(\mathbf{x})]]\!]$. By equation (4) we obtain

$$\mathbf{pre}_\mathcal{C}(\Phi) \equiv \bigvee_{a \in \Sigma, \phi \in \Phi} G_a \wedge \phi[\mathbf{x}/\mathbf{T_a}(\mathbf{x})] \tag{5}$$

where $\equiv$ denotes logical equivalence of constraints.

In the next sections we investigate which classes of constraints are suitable for Symb-Reach$_\mathcal{C}$. We consider only classes $\mathcal{C}$ denoting exactly the upwards-closed sets. In this way, the termination of Symb-Reach$_\mathcal{C}$ follows directly from the termination of Reach, under the proviso that there exist procedures for computing $\mathbf{pre}_\mathcal{C}(\Phi)$ and for deciding Entail$_\mathcal{C}(\Phi, \Psi)$.

The suitability of a class $\mathcal{C}$ is measured with respect to the following parameters:

(1) The computational complexity of deciding Entail$_\mathcal{C}(\Phi, \Psi)$.
(2) The size of the set $\mathbf{pre}_\mathcal{C}(\Phi)$ as a function of the size of Φ.

A note about terminology. Given two sets of constraints Φ, Ψ, we refer to the *containment problem* as the decision problem Entail$(\Phi, \Psi) = true$ for two sets of constraints Φ, Ψ, whereas we refer to the *entailment problem* as the decision problem Entail$(\{\phi\}, \{\psi\}) = true$ for constraints ϕ and ψ.

4 NA-constraints: No Addition

A NA-constraint is a conjunction of atomic constraints of the form $x_i \geq k$, where $x_i \in \{x_1, \ldots, x_n\}$ and k is a positive integer.

The class of NA-constraints denotes exactly the upwards closed sets. If Φ is a set of NA-constraints then $[\![\Phi]\!]$ is clearly upwards-closed. For the other direction, observe first that an upwards-closed set C is completely characterised by its set of minimal elements M, where minimality is taken with respect the pointwise order $\prec$. More precisely, we have $C = \cup_{\mathbf{m}\in M} Up(\mathbf{m})$, where $Up(\mathbf{m}) = \{\mathbf{c} \mid \mathbf{c} \succeq \mathbf{m}\}$. The set M is finite by Dickson's lemma, and $Up(\mathbf{m})$ can be represented by the constraint $x_1 \geq \mathbf{m}(s_1) \wedge \ldots \wedge x_n \geq \mathbf{m}(s_n)$. So the set C can be represented by a set of NA-constraints.

4.1 Complexity of the containment problem in NA

The containment problem can be solved in polynomial time. In fact, the following properties hold. Let Φ, Ψ be sets of NA-constraints. Then,

- Φ entails Ψ if and only if for every constraint $\phi \in \Phi$ there is a constraint $\psi \in \Psi$ such that ϕ entails ψ.
- $\bigwedge_{i=1}^{n} x_i \geq k_i$ entails $\bigwedge_{i=1}^{n} x_i \geq l_i$ if and only if $k_i \geq l_i$ for $i : 1, \ldots, m$.

Thus, the worst-case complexity of the test 'Φ entails Ψ' is $O(|\Phi| * |\Psi| * n)$, where n is the number of variables in Φ and Ψ.

4.2 Size of the set $\mathbf{pre}_{\mathrm{NA}}(\Phi)$

Let Φ be a set of NA-constraints. By equation (5), $\mathbf{pre}_{\mathrm{NA}}(\Phi)$ must be equivalent to the set $\bigvee_{a\in\Sigma,\phi\in\Phi} G_a \wedge \phi[\mathbf{x}/\mathbf{T_a}(\mathbf{x})]$. Unfortunately, we cannot choose $\mathbf{pre}_{\mathrm{NA}}(\Phi)$ *equal* to this set, because it may contain constraints of the form $x_{i_1} + \ldots + x_{i_m} \geq k$. However, when evaluating variables on positive integers, a constraint of the form $x_{i_1} + \ldots + x_{i_m} \geq k$ is equivalent to the following set (disjunction) of NA-constraints:

$$\bigvee_{\langle k_1,\ldots,k_m\rangle} x_{i_1} \geq k_1 \wedge \ldots \wedge x_{i_m} \geq k_m,$$

where each tuple of positive integers $\langle k_1, \ldots, k_m\rangle$ represents an ordered partition of k, i.e. $k_1 + \ldots + k_m = k$. (Moreover, it is easy to see that this is the smallest representation of $x_{i_1} + \ldots + x_{i_m} \geq k$ with NA-constraints.) We define the operator $\mathbf{pre}_{\mathrm{NA}}$ as the result of decomposing all constraints with additions of (5) into NA-constraints.

The cardinality of $\mathbf{pre}_{\mathrm{NA}}(\Phi)$ depends on the number of ordered partitions of the constants appearing in constraints with additions. For $x_1 + \ldots + x_m \geq k$, this number, denoted by $\rho(m, k)$, is equal to the number of subsets of $\{1, 2, \ldots, k + m - 1\}$ containing $m - 1$ elements, i.e.,

$$\rho(m, k) = \binom{k+m-1}{n-1} = \binom{k+m-1}{k}.$$

If c is the biggest constant occurring in constraints of Φ, and n, a are the number of states and actions of the broadcast protocol, we get $|\mathbf{pre}_{\mathrm{NA}}(\Phi)| \in O(|\Phi| * a * \rho(n, c))$. This makes NA-constraints inadequate for cases in which the constants $c \approx n$, initially or during the iteration of algorithm Symb-Reach$_{\mathrm{NA}}$. In this case we get $\rho(n, c) \approx \frac{4^n}{\sqrt{\pi n}}$, which leads to an exponential blow-up.

4.3 Conclusion.

NA-constraints have an efficient entailment algorithm, but they are inadequate as data structure for Symb-Reach. Whenever the constants in the constraints reach values similar to the number of states, the number of constraints grows exponentially.

The blow-up is due to the decomposition of constraints with additions into NA-constraints. In the following section we investigate whether constraints with additions are a better data structure.

5 AD-constraints: With Addition

An AD-constraint is a conjunction of atomic constraints $x_{i_1} + \ldots + x_{i_m} \geq k$ where $x_{i_1}, \ldots, x_{i_m}$ are *distinct* variables of $\{x_1, \ldots, x_n\}$, and k is a positive integer. A constraint in AD can be characterized as the system of inequalities $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ where $\mathbf{A}$ is a 0-1 matrix.

It is easy to see that AD-constraints denote exactly the upwards-closed sets. Since AD-constraints are equivalent to disjunctions of NA-constraints, they only denote upwards-closed sets, and since they are more general than NA-constraints, they denote them all.

5.1 Complexity of the containment problem in AD.

The following result shows that even the entailment test between two AD-constraints is difficult to decide.

Proposition 1 (Entailment in AD is co-NP complete). *Given two* AD-*constraints ϕ and ψ, the problem 'ϕ entails ψ' is co-NP complete.*

Proof. By reduction from HITTING SET [GJ78]. An instance of HITTING SET consists of a finite set $S = \{s_1, \ldots, s_n\}$, a finite family $S_1, \ldots, S_m$ of subsets of S, and a constant $k \leq n$. The problem is to find $T \subseteq S$ of cardinality at most k that *hits* all the S_i, i.e., such that $S_i \cap T \neq \emptyset$.

Take a collection of variables $X = \{x_1, \ldots, x_n\}$. Let ϕ be a conjunction of atomic constraints ϕ_i, one for each set S_i, given by: If $S_i = \{s_{i_1}, \ldots, s_{i_{n_i}}\}$, then $\phi_i = x_{i_1} + \ldots + x_{i_{n_i}} \geq 1$. Let $\psi = x_1 + \ldots + x_n \geq k + 1$.

If ϕ does not entail ψ, then there is a valuation $V : X \to I\!\!N$ that satisfies ϕ but not ψ. Let T be the set given by: $s_i \in T$ if and only $V(x_i) > 0$. Since V satisfies ϕ, T is a hitting set. Since V does not satisfy ψ, it contains at most k elements.

If T is a hitting set with at most k elements, then the valuation $V : X \to I\!\!N$ given by $V(x_i) = 1$ if $s_i \in T$, and 0 otherwise, satisfies ϕ but not ψ.

This implies that entailment of AD-constraints is co-NP-hard. Completeness follows by noting that the containment problem for sets of linear arithmetics constraints is co-NP complete [Sri92]. $\square$

The following corollary immediately follows.

Corollary 1 (Containment in AD is co-NP complete). *Given two sets of* AD*-constraints Φ and Ψ, the problem 'Φ entails Ψ' is co-NP complete.*

5.2 Size of the set $\mathbf{pre}_{\mathbf{AD}}(\Phi)$

We can define

$$\mathbf{pre}_{\mathrm{AD}}(\Phi) = \bigvee_{a \in \Sigma, \phi \in \Phi} G_a \wedge \phi[\mathbf{x}/\mathbf{T_a}(\mathbf{x})]$$

since the right hand side is a set of AD-constraints whenever Φ is. If a is the number of actions of the broadcast protocol, then $|\mathbf{pre}_{\mathrm{AD}}(\Phi)| \in O(|\Phi| * a)$.

5.3 Conclusion

AD-constraints are not a good data structure for Symb-Reach either, due to the high computational cost of checking containment and entailment. This result suggests to look for a class of constraints between NA and AD.

6 DV-constraints: With Distinct Variables

DV-constraints are AD-constraints of the form

$$x_{1,1} + \ldots + x_{1,n_1} \geq k_1 \quad \wedge \ldots \wedge \quad x_{m,1} + \ldots + x_{m,n_m} \geq k_m ,$$

where $x_{i,j}$ and $x_{i',j'}$ are distinct variables (DV) for all i, j, i', j'. In other words, a DV-constraint can be represented as $\mathbf{A} \cdot \mathbf{x} \geq \mathbf{b}$ where $\mathbf{A}$ is a 0-1 matrix with unit vectors as columns.

Since DV-constraints are more general than NA-constraints, but a particular case of AD-constraints, they denote exactly the upwards-closed sets.

6.1 Complexity of the containment problem in DV.

Entailment between sets of DV-constraints can still be very expensive, as shown by the following result.

Proposition 2 (Containment in DV is co-NP complete). *Given two sets of* DV*-constraints Φ and Ψ, the problem 'Φ entails Ψ' is co-NP complete.*

Proof. By reduction from INDEPENDENT SET [GJ78]. An instance of INDEPENDENT SET consists of a finite graph $G = (V, E)$ and a constant $k \leq |V|$. The problem is to find $I \subseteq V$ of cardinality at most k such that for every $u, v \in I$ there is no edge between u and v.

Assume $V = \{v_1, \ldots, v_n\}$. Take a collection of variables $X = \{x_1, \ldots, x_n\}$. The set Φ contains a constraint $x_i \leq 1$ for $i : 1 \ldots n$, and $x_i + x_j \leq 1$ for every edge $(v_i, v_j) \in E$. The set Ψ is the singleton $\{\psi\}$, where $\psi = x_1 + \ldots + x_n \geq k + 1$.

If Φ does not entail ψ, then there is a valuation $V : X \to I\!N$ that satisfies Φ but not ψ. Let I be the set given by: $s_i \in I$ if and only $V(x_i) > 0$. Since V

satisfies Φ, I is an independent set. Since V does not satisfy ψ, it contains at most k elements.

If I is an independent set with at most k elements, then the valuation $V: X \rightarrow \mathbb{N}$ given by $V(x_i) = 1$ if $s_i \in I$, and 0 otherwise, satisfies ϕ but not ψ. $\square$

However, and differently from the AD-case, checking entailment between two AD-constraints can be done in polynomial time. Let $Var(\phi)$ denote the set of free variables occurring in the constraint ϕ, and let $Cons(\gamma)$ denote the constant occurring in the *atomic* constraint γ. We have the following result:

Proposition 3. *Let ϕ and γ be an arbitrary and an atomic* DV*-constraint, respectively. Let Δ be the largest set of atomic constraints δ in ϕ such that $Var(\delta) \subseteq Var(\gamma)$. Then, ϕ entails γ if and only if $\Sigma_{\delta \in \Delta} Cons(\delta) \geq Cons(\gamma)$.*

Proof. ($\Rightarrow$): Assume $\Sigma_{\delta \in \Delta} Cons(\delta) < Cons(\gamma)$. Then, any valuation that assigns $Cons(\delta)$ to one variable in δ and 0 to the others, and 0 to the remaining variables of $Var(\gamma)$, satisfies ϕ but not γ.
($\Leftarrow$): Clearly ϕ entails Δ. Since ϕ is a DV-constraint, Δ entails the constraint $\sum_{x_i \in Var(\delta)} x_i \geq \Sigma_{\delta \in \Delta} Cons(\delta)$. Since $Var(\delta) \subseteq Var(\gamma)$ and $\sum_{\delta \in \Delta} Cons(\delta) \geq Cons(\gamma)$, it also entails $\sum_{x_i \in Var(\gamma)} x_i \geq Cons(\gamma)$, which is the constraint γ. $\square$

For instance, we have that $x_1 + x_2 \geq a \wedge x_3 \geq b$ entails $x_1 + x_2 + x_3 + x_4 \geq c$ if and only if $a + b \geq c$.

Since ϕ entails ψ if and only if ϕ entails each atomic constraint of ψ, we get the following

Corollary 2 (Entailment in DV **is in P).** *Given two* DV*-constraints ϕ and ψ, it can be checked in polynomial time whether ϕ entails ψ.*

Since the symbolic procedure for the reachability problem requires to check containment, and not entailment, Corollary 2 does not seem to be of much use at first sight. However, it allows to define a new reachability procedure by replacing the $\text{Entail}_{\mathcal{C}}(\Phi, old\_\Phi)$ test in Symb-Reach by the *local* containment test:

$$\textbf{forall } \phi \in \Phi \textbf{ exists } \psi \in old\_\Phi$$

Clearly, the local containment test implies the containment test, and so the new procedure is partially correct. The risk of weakening the fixpoint test is that we may end up with a non-terminating algorithm. Fortunately, this turns out not to be the case, as shown by the following proposition.

Proposition 4. *The procedure Symb-Reach$_{\text{DV}}$ terminates.*

Proof. Let X be a set of variables. Given $Y \subseteq X$, let $Y \geq k$ denote the constraint $\sum_{x_i \in Y} x_i \geq k$.

Let ϕ be a DV-constraint on X. We define the function f_ϕ which assigns to $Y \subseteq X$ a natural number as follows:

$$f_\phi(Y) = \begin{cases} k & \text{if } \Phi \text{ contains the constraint } Y \geq k \\ 0 & \text{otherwise} \end{cases}$$

Observe that f_ϕ is well defined because ϕ is a DV-constraint. Define the pointwise ordering $\preceq$ on these functions, given by $f_\phi \preceq f_\psi$ if $f_\phi(Y) \leq f_\psi(Y)$ for every subset Y of X. We prove that the local containment test corresponds exactly to the pointwise ordering. I.e., for DV-constraints, ϕ entails ψ if and only if $f_\phi(Y) \geq f_\psi(Y)$.

- If $f_\phi \geq f_\psi$, then ϕ entails ψ.
  Let $Y \geq k$ be an atomic constraint of ψ. It follows from $f_\phi(Y) \geq f_\psi(Y)$ that ϕ contains a constraint $Y \geq k'$ such that $k' \geq k$. So every solution of ϕ is a solution of $Y \geq k$.
- If ϕ entails ψ, then $f_\phi \geq f_\psi$.
  We prove the contraposition. Let $Y \subseteq X$ such that $f_\phi(Y) < f_\psi(Y)$. Then ψ contains a constraint $Y \geq k$, and ϕ contains a constraint $Y \geq k'$ such that $k' < k$ (if ϕ contains no constraint $Y \geq k'$ we can assume that it contains the constraint $Y \geq 0$). Since ϕ is a DV-constraint, it has a solution X_0 such that $Y_0 = k'$. So X_0 does not satisfy $Y \geq k$, and so ϕ does not entail ψ.

Assume now that Symb-Reach$_{\mathrm{DV}}$ does not terminate. Then, the i-th iteration of the repeat loop generates at least one constraint ϕ_i such that ϕ_i does not entail ϕ_j for any $i > j$. By the result above, the sequence of functions f_{ϕ_i} satisfies $f_{\phi_i} \not\preceq f_{\phi_j}$ for any $i > j$. This contradicts Dickson's lemma (consider a function f_ϕ as a vector of $\mathbb{N}^{2^{|X|}}$). □

6.2 Size of the set $\mathbf{pre}_{\mathbf{DV}}(\Phi)$

If Φ is a set of DV-constraints, then the set of constraints (5) may contain AD-constraints with shared variables. However, each constraint in set (5) is either a DV-constraint or has one of the two following forms: $\phi \wedge x_i \geq 1$ or $\phi \wedge x_i \geq 1 \wedge x_j \geq 1$, where ϕ is a DV-constraint with at most one occurrence of x_i and x_j. The constraints of the form $x_i \geq 1$ correspond to the 'guards' of the transition rules of the protocol. Thus, in order to maintain constraints in DV-form, all we have to do is to merge the 'guards' and the remaining DV-constraint (i.e. ϕ). The operator $\mathbf{pre}_{\mathrm{DV}}$ is defined as the result of applying the following normalization: Given a constraint $x \geq 1 \ \wedge \ x + y_1 + \ldots + y_m \geq k \wedge \phi$ where, by hypothesis, x does not occur in ϕ, replace it by the equivalent set of constraints

$$\bigvee_{i=0}^{k-1} (x \geq k - i \ \wedge \ y_1 + \ldots + y_m \geq i \wedge \phi)\,.$$

In the worst case, it is necessary to reduce each new constraint with respect to two guards, possibly generating $O(k^2)$ new constraints. Thus, if a is the number of actions of the protocol and c is the maximum constant occurring in the set Φ of DV-constraints, we have $|\mathbf{pre}_{\mathrm{DV}}(\Phi)| \in O(|\Phi| * a * c^2)$.

6.3 Conclusion

DV-constraints are a good compromise between AD and NA-constraints. The application of $\mathbf{pre}_{\mathrm{DV}}$ does not cause an exponential blow up as in the case of NA-constraints. Furthermore, though the containment test is co-NP complete, it can be relaxed to an entailment of low polynomial complexity, unlike the case of AD-constraints. Moreover, as shown in the next section, sets of DV-constraints can be compactly represented.

7 Efficient Representation of Sets of Constraints

DV-constraints can be manipulated using very efficient data-structures and operations. We consider constraints over the variables $\{x_1, \ldots, x_n\}$.

Each *atomic* DV-constraint $\Sigma_{x_i \in Y} x_i \geq k$ can be represented as a pair $\langle \mathbf{b}, k \rangle$, where $\mathbf{b}$ is a *bit-vector*, i.e., $\mathbf{b} = \langle b_1, \ldots, b_n \rangle$ and $b_i = 1$ if $x_i \in Y$, and 0 otherwise. Thus, a DV-constraint can be represented as a set of pairs. Based on this encoding, the decision procedure of Corollary 2 can be defined using bitvector operations *not* and *or*. (**1** denotes the bitvector containing only 1's.)

Proc Entails(*cstr1*, *cstr2*: codings of DV-constraints)
  var s : *integer*
  for all pairs $\langle \mathbf{b}_2, k_2 \rangle$ in *cstr2*
    $s := 0$;
    for all pairs $\langle \mathbf{b}_1, k_1 \rangle$ in *cstr1*
      if $(not(\mathbf{b}_1)\ or\ \mathbf{b}_2) = \mathbf{1}$ **then** $s := s + k_1$ **endif**
    endfor
    if $s < k_2$ **then return false endif**
  endfor;
  return true

8 Examples

In this section we present and discuss some experimental results. We first show some examples of systems and properties that we were able to verify automatically, and then we compare the execution times obtained by using different constraint systems.

The protocol shown in Fig. 1 models a network of processes accessing two shared files (called 'a' and 'b') under the last-in first-served policy. When a process wants to write on one of the files all processes reading it are redirect in the initial state **I**. In the state **I** a process must send a broadcast before starting reading a file: in this case all writers are sent back to the state **I** (last-in first-served). Note that processes operating on 'b' simply skip the broadcast concerning operations on 'a' and vice versa. The protocol must ensure mutual

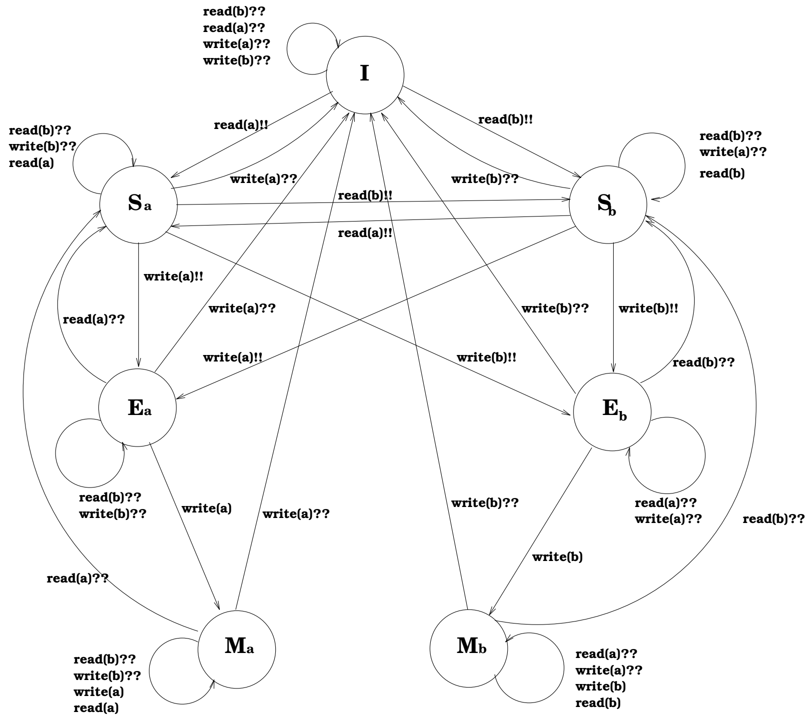


Fig. 1. Last-in first-served access to two resources.

exclusion between readers and writers. The initial parameterized configuration of the protocol is

$$I \geq 1, S_a = 0, S_b = 0, E_a = 0, E_b = 0, M_a = 0, M_b = 0\,.$$

We prove that the unsafe configurations $Sa \geq 1, Ma \geq 1$ are not reachable.

In Fig. 2, we describe a central server model [ABC+95]. Processes in state **think** represent thinking clients that submit jobs to the CPU. A number of processes may accumulate in state $\mathbf{wait_{cpu}}$. The first job requesting the CPU finds it idle and starts using it. A job that completes its service proceeds to a selection point where it continues requesting the I/O subsytem or leaves the central system. No specific policy is specified for the queues of waiting jobs. In the initial state of the broadcast protocol in Fig. 2 an arbitrary number of processes are in state **think**, whereas one process is respectively in state $\mathbf{idle_{cpu}}$, $\mathbf{idle_{disk}}$, $\mathbf{no_{int}}$. The protocol must ensure that only one job at a time can use the CPU and the I/O subsytem. The flow of processes is represented by a collection of rules over 17 variables (one for each state). The initial parameterized configuration of the protocol is

$$\mathit{Think} \geq 1,\ \mathit{Idle}_{cpu} = 1,\ \mathit{Idle}_{disk} = 1,\ \mathit{No\text{-}int} = 1,$$

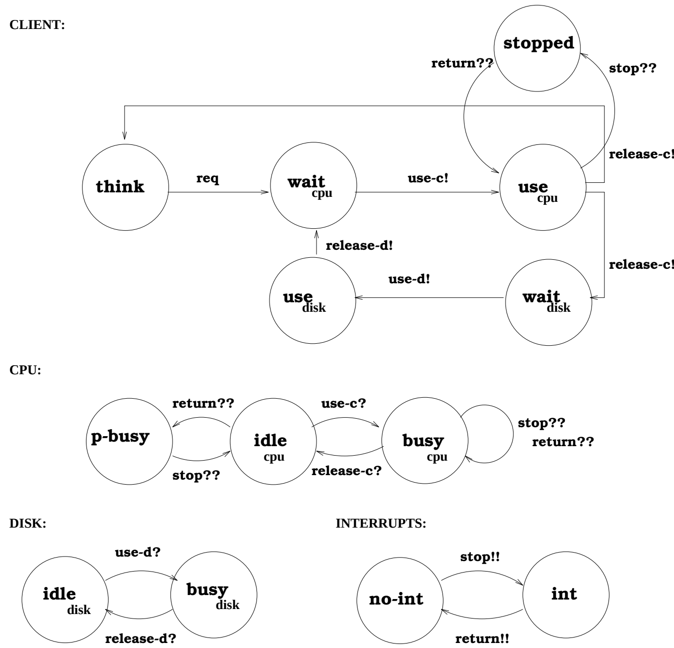


Fig. 2. Central Server System.

with all other variables equal to zero. We prove that the unsafe configurations $Use_{cpu} \geq 2$ is not reachable.

Petri Nets can be seen as a special case of broadcast protocols where the constraints generated during the analysis are in NA-form. Consider the Petri net of [Ter94] shown in Fig. 3, which describes a system for manufacturing tables (for instance, transition t_4 assembles a table by taking a board from the place p_6 and four legs from the place p_5). The constraint-based representation introduces a variable for each place and for each transition. The variables corresponding to transitions count the number of times a transition is fired during the execution. There is a rule for each transition. For instance, the rule corresponding to transition t_4 is

$$P_6 \geq 1,\ P_5 \geq 4,\ P_6' = P_6 - 1,\ P_5' = P_5 - 4,\ P_7' = P_7 + 1,\ T_4' = T_4 + 1$$

In [Ter94] it is shown that an initial marking of this is deadlock-free (i.e., no sequence of transition occurrences can lead to a deadlock) if and only if it enables a sequence of transition occurrences containing t_1 at least three times and all other transitions at least twice. Based on this preliminary result we can then compute all deadlock-free initial states. They are exactly the predecessors states of the states

$$T_1 \geq 3,\ T_2 \geq 2,\ T_3 \geq 2,\ T_4 \geq 2,\ T_5 \geq 2,\ T_6 \geq 2$$

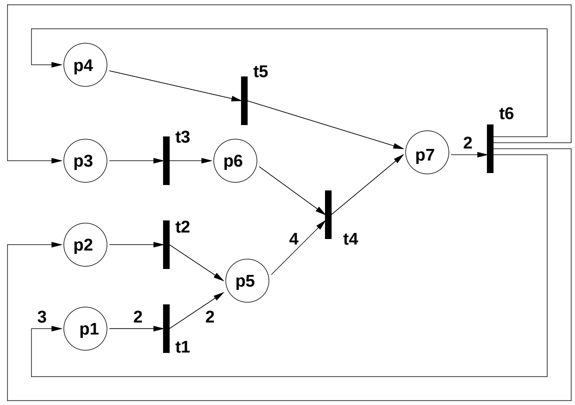


Fig. 3. Manufacturing System modeled as a Choice-free Petri Net.

intersected with the initial states of the system, i.e., those such that $T_i = 0$ for all i and $P_5 = P_6 = P_7 = 0$. The result of the fixpoint computation is given by the following set of constraints

$$
\begin{array}{lll}
P_1 \geq 10, P_2 \geq 1, P_3 \geq 2 & P_1 \geq 8, P_2 \geq 3 & P_1 \geq 12, P_3 \geq 2 \\
P_1 \geq 6, P_2 \geq 5, P_3 \geq 2 & P_1 \geq 8, P_3 \geq 1, P_4 \geq 1 & P_1 \geq 6, P_4 \geq 2 \\
P_1 \geq 6, P_2 \geq 1, P_3 \geq 1, P_4 \geq 1 & &
\end{array}
$$

8.1 Comparison of execution times

We have tested the previous examples on HyTech (polyhedra representation of sets of configurations, full entailment test), on Bultan, Gerber and Pugh's model checker based on the Omega library for Presburger arithmetic [BGP97], and on the specialized model checker we have introduced in the paper (DV-constraint representation of sets of states, local entailment test). HyTech works on real arithmetic, i.e., it employs efficient constraint solving for dealing with linear constraints. The results are shown in the following table, where 'Presb' refers to the model checker of [BGP97], and 'BitVector' to our checker.

Fig	Rules	Unsafe States	Steps	BitVector[1]	HyTech[1]	Presb[2]
1	21	$S_a \geq 1, M_a \geq 1$	2	<1s	<1s	not tested
2	9	$Use_{cpu} \geq 2$	7	<1s	5.5s	40s
		$Use_{cpu} \geq 3$	10	<1s	16s	290s
		$Use_{cpu} \geq 4$	13	<1s	40s	1558s
		$Use_{cpu} \geq 8$	25	15s	578s	not tested
		$Use_{cpu} \geq 10$	31	76s	1738s	not tested
3	6	$T_1 \geq 3, \wedge_{i>1} T_i \geq 2$	24	1090s	>6h	19h50m

[1] On a Sun Sparc 5.6. [2] On a Sun Ultra Sparc.

9 Related work

The first algorithm for testing safety properties of broadcast protocols was proposed by Emerson and Namjoshi in [EN98]. Their approach is based on an extension of the Karp and Miller's cover graph construction (used for Petri Nets) [KM69]. In [EFM99], Esparza, Finkel and Mayr show that the algorithm may not terminate and propose a backwards-reachability procedure. The correctness of the procedure follows from general results on the decidability of infinite state systems by Abdulla *et al.* [ACJ+96]. In [Kin99], Kindahl uses constraints as symbolic representation of upwards-closed sets for Petri Nets and lossy channel systems, but does not discuss the issue of finding adequate classes of constraints. Finally, Delzanno and Podelski [DP99], and Bérard and Fribourg [BF99] have recently applied real-arithmetics to model checking of integer systems.

10 Conclusion

We have proposed linear constraints with disjoint variables as a good symbolic representation for upwards-closed sets of configurations of broadcast protocols. Experimental results shown that even a prototype implementation can beat tools for more general constraints.

Acknowledgements We thank Tevfik Bultan for the experiments using his model checker based on Presburger Arithmetics [BGP97].

References

[ACJ+96] P. A. Abdulla, K. Cerāns, B. Jonsson and Y.-K. Tsay. General Decidability Theorems for Infinite-State Systems. In *Proc. 10th IEEE Int. Symp. on Logic in Computer Science*, pp. 313–321, 1996.

[ABC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* Series in Parallel Computing. John Wiley & Sons, 1995.

[BF99] B. Bérard, and L. Fribourg. Reachability analysis of (timed) Petri nets using real arithmetic. In *Proc. .10th Int. Conf. on Concurrency Theory (CONCUR'99)*, Eindhoven, The Netherlands, August, 1999. To appear.

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic Model Checking of Infinite-state Systems using Presburger Arithmetics. In Orna Grumberg, editor, *Proc. 9th Conf. on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 400–411. Springer-Verlag, 1997.

[DP99] G. Delzanno and A. Podelski, Model Checking in CLP. In W. R. Cleaveland, editor, *Proc. 5th Int. Con. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99).*, LNCS 1579, pp. 223–239. Springer-Verlag, 1999.

[EN98] E. A. Emerson and K. S. Namjoshi. On Model Checking for Non-Deterministic Infinite-State Systems. In *Proc. 13th IEEE Int. Symp. on Logic in Computer Science*, 1998.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *Proc. 14th IEEE Int. Symp. on Logic in Computer Science*, 1998.

[FS98] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! Technical Report LSV-98-4, Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan. April 1998. To appear in *Theoretical Computer Science*, 1999.

[GJ78] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman, 1978.

[HHW97] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: a Model Checker for Hybrid Systems. In Orna Grumberg, editor, *Proc. 9th Conf. on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 460–463. Springer-Verlag, 1997.

[Kin99] M. Kindahl. *Verification of Infinite State Systems: Decision Problems and Efficient Algorithms.* Ph.D. thesis, Department of Computer Systems, Uppsala University, June 1999. Available as report DoCS 110.

[KM69] R. M. Karp and R. E. Miller. Parallel Program Schemata. *Journal of Computer and System Sciences*, 3, pp. 147-195, 1969.

[McA84] K. McAloon. Petri Nets and Large Finite Sets. *Theoretical Computer Science* 32, pp. 173–183, 1984.

[Sri92] D. Srivistava. Subsumption and indexing in constraint query languages with linear arithmetic constraints. In *2nd Int. Symp. on Artificial Intelligence and Mathematics, Fort Lauderdale*, 1992.

[Ter94] E. Teruel. *Structure Theory of Weighted Place/Transition Net Systems: The Equal Conflict Hiatus.* Ph.D. Thesis, University of Zaragoza, 1994.

[VW86] M. Y. Vardi and P. Wolper. Automata-Theoretic Techniques for Modal Logics of Programs. *Journal of Computer and System Sciences.* 32(2), pp. 183–221, April, 1986.