Derivation of Non-structural Invariants of Petri Nets Using Abstract Interpretation

Robert Clarisó, Enric Rodríguez-Carbonell, and Jordi Cortadella

Universitat Politècnica de Catalunya,
Barcelona, Spain

Abstract. Abstract interpretation is a paradigm that has been successfully used in the verification and optimization of programs. This paper presents a new approach for the analysis of Petri Nets based on abstract interpretation. The main contribution is the capability of deriving non-structural invariants that can increase the accuracy of structural methods in calculating approximations of the reachability space. This new approach is illustrated with the verification of two examples from the literature.

1 Introduction

The analysis of the state space of a Petri Net can be done by using different methods. Traditionally, three types of methods have been proposed [24]:

– Enumeration techniques, which provide an exact characterization for bounded systems and partial approximations for unbounded systems. These techniques suffer from the state explosion problem that often appears in highly concurrent systems.
– Transformation techniques, which alleviate the previous problem by reducing the system into a smaller one that still preserves the properties under analysis.
– Structural techniques, which provide information of the system based on the underlying graph structure of the net. Structural techniques typically compute upper approximations of the state space that can be effectively used for the verification of safety properties.

Structural techniques provide linear descriptions of the state space by exploiting the information given by the state equation [19]. As an example, the following invariant characterizes the markings that fulfill the state equation for the Petri Net in Fig. 1(a):

$$2p_1 + p_2 + p_3 + p_4 + p_5 = 2. \tag{1}$$

The reachability graph is depicted in Fig. 1(b), in which the shadowed states represent spurious (unreachable) markings. The presence of spurious markings is the cost that must be paid when using approximation techniques to calculate the state space.

This paper presents an attempt to explore a different analysis approach that lives between the accuracy of the enumeration methods and the efficiency of structural techniques. The goal is to reduce the set of spurious markings by generating more accurate

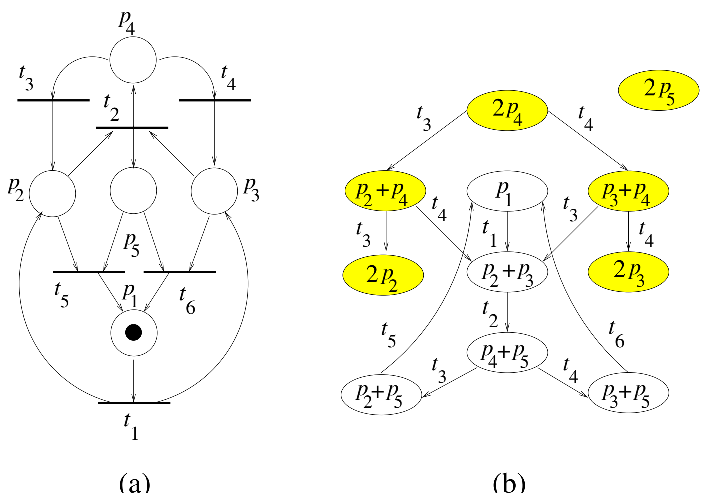


Fig. 1. (a) Petri Net (from [24]), (b) reachability graph

Table 1. Non-structural invariants for the Petri Net in Fig. 1(a)

Linear inequalities		Polynomial equalities	
$p_2 + p_4 \leq p_3 + p_5$	(2)	$p_5^2 - p_5 = 0$	(5)
$p_3 + p_4 \leq p_2 + p_5$	(3)	$p_4 p_5 - p_4 = 0$	(6)
$p_5 \leq p_2 + p_3 + p_4$	(4)	$2p_3 p_5 + p_2 + p_4 = p_3 + p_5$	(7)

characterizations of the state space taking into account both the initial marking and the structure of the net.

The approach is based on the paradigm of abstract interpretation [9], successfully used in different areas for the verification and optimization of systems [7]. In this paper we present two abstract domains that are able to derive non-structural invariants for Petri Nets: linear inequalities and polynomial equalities.

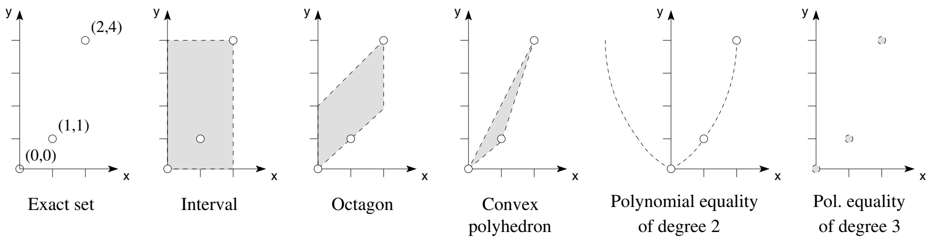
As an example, abstract interpretation has been able to obtain the invariants in Table 1 for the previous Petri net.[1]

Some observations on the new invariants:

– The invariants (1)-(4) represent the exact reachability graph.
– The invariants (1) and (5)-(7) also represent the exact reachability graph.

Even though in this case abstract interpretation can characterize the reachability graph exactly, in general it provides an upper approximation, which may be more accurate than the one defined by the structural invariants. Nevertheless, these conservative invariants can be used to prove safety properties of the Petri Net, like boundedness and

[1] The invariant (1) is also obtained in both abstract domains.



Interval	Octagon	Convex polyhedron	Polynomial equalities of degree ≤ 2	Polynomial equalities of degree ≤ 3
$0 \leq x \leq 2$	$0 \leq x \leq 2$	$y - x \geq 0$	$y = x^2$	$y = x^2$
$0 \leq y \leq 4$	$0 \leq y - x \leq 2$	$x - 2y \geq 0$		$x \cdot (x-1) \cdot (x-2) = 0$
		$3x - y \leq 2$		

Fig. 2. Approximating a set of values (left) with several abstract domains

deadlock freedom (by studying whether the conjunction of the disabling conditions for all the transitions and the invariants is feasible).

Some previous works have also studied the generation of linear inequality invariants. Many of them have been based on the analysis of structural properties and objects [16], including siphons and traps [5]. Another approach, presented in [23], uses Farkas' lemma to generate inductive linear invariants. Finally, Presburger arithmetics [13] and real arithmetics [2] can be used to represent the state space of Petri Nets, providing again linear inequality invariants.

Abstract interpretation offers new chances for the analysis of concurrent systems and, in particular, of Petri Nets. The possibility of deriving non-structural invariants can open the door to a new family of strategies that can better explore the trade-off between accuracy and efficiency in the modeling of concurrent systems. In this paper, the applicability of abstract interpretation is illustrated with the verification of an automated manufacturing system and the alternating bit protocol.

2 Abstract Interpretation

2.1 Fundamentals

Abstract interpretation [9] is a generic approach for the static analysis of complex systems. The underlying notion in abstract interpretation is that of *upper approximation*: to provide an abstraction of a complex behavior with less details. Upper approximations are conservative in the sense that they can be used to prove safety properties, e.g. "no errors in the abstraction" means "no errors in the system". A property about a system such as an invariant is in some way an abstraction: it represents all the states of the system that satisfy the property.

Intuitively, abstract interpretation defines a procedure to compute an upper approximation for a given behavior of a system. This definition guarantees (a) the termination of the procedure and (b) that the result is conservative. An important decision is the choice of the kind of upper approximation to be used, which is called the *abstract domain*. For a given problem, there are typically several abstract domains available. Each abstract domain provides a different trade-off between precision (proximity to the exact result) and efficiency.

There are many problems where abstract interpretation can be applied, several of them oriented towards the compile-time detection of run-time errors in software [6]. For example, some analysis based on abstract interpretation can discover numeric invariants among the variables of a program. Several abstract domains can be used to describe the invariants: intervals [8], octagons [17], convex polyhedra [10] or polynomial equalities [22]. These abstract domains provide different ways to approximate sets of values of numeric variables. For example, Figure 2 shows how these abstract domains can represent the set of values of a pair of variables x and y.

2.2 Application to the Reachability Problem

The reachable markings of a Petri Net can be studied using abstract interpretation. We will consider the classic model of Petri Nets, extended with inhibitor arcs and parameters in the initial marking. A reachable marking of a Petri Net N can be seen as an assignment to k non-negative integer variables, where k is the number of places in N. Therefore, a set of markings can be approximated using the abstract domains from Figure 2. This approach has several benefits. First, the abstract domains can represent large sets of markings compactly. Even infinite sets of markings can be represented efficiently. Moreover, the analysis can work with *parametric* markings where the number of tokens in a place is defined by a parameter. Finally, the approximation leads to a faster generation of the reachable state space. The result may contain some unreachable markings, but all reachable markings will be included in the solution. Thus, all invariants discovered by abstract interpretation hold in all the reachable markings.

We will show the computation of the reachable markings using the convex polyhedra abstract domain, even though other abstract domains could be used. The abstract interpretation procedure applied to the reachability analysis is shown in Figure 3. Intuitively, the algorithm behaves as follows. Initially, only the initial marking is reachable. There may be several enabled transitions, which will discover new reachable markings when they are fired. The algorithm keeps on firing transitions until no more reachable states can be found. The enabling condition can be expressed using linear inequalities, while the effect of firing a transition can be expressed as linear assignments. Both operations are available in the abstract domain of convex polyhedra. Notice that each step deals with sets of reachable markings instead of individual markings.

The algorithm consists in computing the following sequence:

$$\begin{aligned} \text{reachable}^0 &= M_0 \\ \text{reachable}^{i+1} &= \text{reachable}^i \,\cup\, \text{next}(\text{reachable}^i, T)\ . \end{aligned}$$

In this recurrence, M_0 is the initial marking of the Petri Net. The set $\text{next}(M, T)$ represents the markings reached by firing once any transition in $t \in T$ from any marking $m \in M$ such that t is enabled in m. The union operator ($\cup$) is not exact for convex

```
Input: A Petri Net $N = \langle P, T, F, M_0 \rangle$ with a set of places $P$, a set of transitions $T$, a flow-
relation $F$ and an initial marking $M_0$.
Output: An upper approximation of the set of reachable markings of $N$, described in the abstract
domain used in this analysis (e.g. a set of linear inequalities or polynomial equalities).

reachable = { $M_0$ }                                   # Start from the initial marking
do {
  old := reachable
  for each transition $t \in T$ {
    enabling := enablingCondition( $t$, $F$ )            # Enabling condition of t defined by F
    enabled := reachable $\cap$ enabling                  # Reachable markings where t is enabled
    if ( enabled = $\emptyset$ ) continue                # Check if t is not enabled yet
    next := fire( $t$, enabled, $F$ )                    # Fire t from the enabled markings
    reachable := reachable $\nabla$ ( reachable $\cup$ next )   # Accumulate the new markings
  }
} while ( reachable $\neq$ old );
```

Fig. 3. Abstract interpretation algorithm used to compute the set of reachable markings

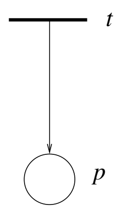
polyhedra, so some degree of approximation is introduced in this way. However, solving this recurrence has a problem: there is no guarantee that the algorithm will terminate. If the Petri Net is unbounded, the algorithm might iterate an infinite number of times. So instead of this recurrence, the algorithm solves another recurrence that relies on a *widening* operator (∇). Widening extrapolates the effect of repeating a computation an unbounded number of times. Its definition ensures the termination of the analysis after a finite number of steps. Given A, the states before the computation, and B the states after the computation, the widening is denoted as $A \nabla B$. A possible high-level definition of widening can be "keep the constraints from A that also hold in B", considering that any property modified during the computation might be further modified in later iterations. Using this operator, the recurrence can be rewritten as:

$$\begin{aligned} \text{reachable}^0 &= M_0 \\ \text{reachable}^{i+1} &= \text{reachable}^i \ \nabla \ (\ \text{reachable}^i \ \cup \ \text{next}(\text{reachable}^i, T) \ ) \ . \end{aligned}$$

Figure 4 illustrates the effect of the widening on the computation. On the top, we show the computation of the reachable markings if the widening is not used. This computation does not terminate. On the bottom, we see the same computation using widening. In this case, the computation terminates quickly. Notice that, between $(p = 0)$ and $(0 \leq p \leq 1)$, the only common property is $(p \geq 0)$.

2.3 An Example

The example used to present the abstract interpretation algorithm is shown in Figure 5. This Petri Net is modeling a producer-consumer system that communicates through a lossy channel. The left subnet, the producer, generates tokens while the right subnet, the consumer, removes these tokens. The place p_4 models the communication channel



$$\begin{array}{l}
(p = 0) \\
\quad \downarrow t \\
(p = 1) \cup (p = 0) = (0 \le p \le 1) \\
\qquad\qquad\qquad\qquad\qquad \downarrow t \\
\qquad\qquad\qquad\qquad (1 \le p \le 2) \cup (0 \le p \le 1) = (0 \le p \le 2) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow t \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ldots
\end{array}$$

$$\begin{array}{l}
(p = 0) \\
\quad \downarrow t \\
(p = 1) \cup (p = 0) = (0 \le p \le 1) \\
(p = 0) \; \nabla \; (0 \le p \le 1) = (p \ge 0) \\
\qquad\qquad\qquad\qquad\qquad \downarrow t \\
\qquad\qquad\qquad\qquad (p \ge 1) \cup (p \ge 0) = (p \ge 0) \\
\qquad\qquad\qquad\qquad (p \ge 0) \; \nabla \; (p \ge 0) = (p \ge 0) \quad \text{Fixpoint}
\end{array}$$

Fig. 4. Justification of the necessity of a widening operator. On the top, computation of the reachable markings without any widening. On the bottom, computation of the reachable markings using widening

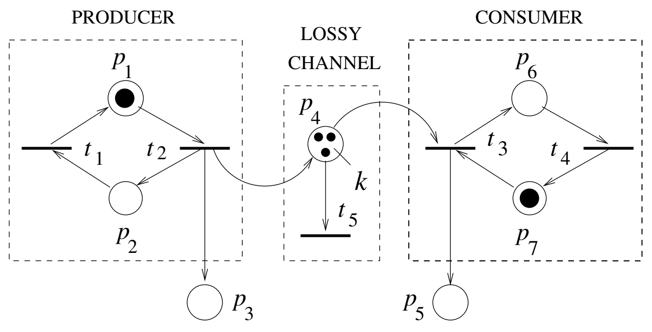


Fig. 5. Petri Net model of a producer-consumer system

between the producer and the consumer, while places p_3 and p_5 count the number of executions of the producer and the consumer respectively. The transition t_5 models the possible loss of data in the channel: some tokens generated by the producer will disappear before reaching the consumer. Initially, we assume that there are already k tokens in the channel, where k is a parameter of the system.

Several interesting invariants hold in this system. For example, the consumer cannot process more elements than those created by the producer or initially in the channel, i.e. $(p_5 \le p_3 + k)$. It should also be noted that the places p_3, p_4 and p_5 are not bounded, so the set of reachable markings is infinite.

Let us discuss a part of the execution of the algorithm in this example. For the sake of brevity, the constraints of the form $(p_i \ge 0)$ and $(k \ge 0)$ will not be shown. The initial markings, parametrized by the value of k, are the following:

$$(p_1 = 1) \wedge (p_2 = 0) \wedge (p_3 = 0) \wedge (p_4 = k) \wedge (p_5 = 0) \wedge (p_6 = 0) \wedge (p_7 = 1).$$

In these markings, there are three transitions enabled: t_2, t_3 and t_5. When t_2 is fired, the following markings are reached:

$$(p_1 = 0) \wedge (p_2 = 1) \wedge (p_3 = 1) \wedge (p_4 = k + 1) \wedge (p_5 = 0) \wedge (p_6 = 0) \wedge (p_7 = 1).$$

These new markings can be combined with the initial markings using union and widening, producing the invariant:

$$\begin{gathered}(p_1 + p_4 = k + 1) \ \wedge \ (p_2 + k = p_4) \ \wedge \ (p_3 + k = p_4) \ \wedge \ (p_5 = 0) \ \wedge \\ (p_6 = 0) \ \wedge \ (p_7 = 1) \ \wedge \ (p_4 \geq k) \ \wedge \ (p_4 \leq k + 1).\end{gathered}$$

The algorithm does the same computation for the enabled transitions t_3 and t_5. After this step, new transitions become enabled, which once they are fired, increase again the set of reachable markings. The procedure is repeated until the set of markings does not change. When the fixpoint is found in this example, the set of reachable markings is defined by:

$$(p_1 + p_2 = 1) \ \wedge \ (p_6 + p_7 = 1) \ \wedge \ (p_2 \leq p_3) \ \wedge \ (p_5 \geq p_6) \ \wedge (p_3 + k \geq p_4 + p_5).$$

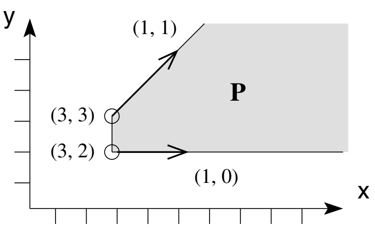
The most interesting invariant in this result is $(p_3 + k \geq p_4 + p_5)$. This property is stating that any element that is consumed (p_5) or remains in the channel (p_4) was either produced (p_3) or initially available (k). Note that this invariant implies the one presented previously, $(p_5 \leq p_3 + k)$.

The following sections will present in detail two abstract domains that are suitable for the discovery of Petri Net invariants. *Convex polyhedra* and *polynomial equalities* can both represent a large class of interesting invariants. On one hand, convex polyhedra describe the set of reachable markings as a system of linear inequalities, so properties like $(p = 1)$ or $(p_1 \leq p_2)$ are easy to represent. The weakness of convex polyhedra is the loss of precision in the union, e.g. $(p_1 = 3) \cup (p_2 = 3)$ can only be approximated as $(p_1 + p_2 \geq 3)$. On the other hand, polynomial equalities are very precise in terms of describing disjunctions, e.g. $(p_1 = 3) \cup (p_2 = 3)$ can be represented exactly as $((p_1 - 3) \cdot (p_2 - 3) = 0)$. However, the description of inequality properties is more difficult: it is only possible when an upper bound is known, e.g. $(p_1 \leq 2)$ can be represented exactly as $(p_1 \cdot (p_1 - 1) \cdot (p_1 - 2) = 0)$; but for instance, that is not possible with $(p_1 \leq p_2)$.

3 Linear Inequality Invariants

3.1 Convex Polyhedra

Convex polyhedra can be described as the set of solutions of a conjunction of *linear inequality constraints* with rational ($\mathbb{Q}$) coefficients. Let P be a polyhedron over $\mathbb{Q}^n$, then it can be represented as the solution to the system of m inequalities $P = \{X | AX \geq B\}$ where $A \in \mathbb{Q}^{m \times n}$ and $B \in \mathbb{Q}^m$. The set of variables X contains the counters of the number of tokens in each place and the parameters of the initial marking. Convex polyhedra can also be characterized in a *polar* representation by means of a *system of generators*, i.e. as a linear combination of a set of vertices V (points) and a set



System of generators

$$P=\{\lambda_1 \cdot (3,3) + \lambda_2 \cdot (3,2) + \mu_1 \cdot (1,1) + \mu_2 \cdot (1,0) | \\ \lambda_1 \geq 0, \lambda_2 \geq 0, \mu_1 \geq 0, \mu_2 \geq 0, \lambda_1 + \lambda_2 = 1\}$$

System of constraints

$$P = \{(x,y) \mid (y \geq 2) \wedge (x \geq 3) \wedge (x - y \geq 0)\}$$

Fig. 6. An example of a convex polyhedron (shaded area) and its double description

of rays R (vectors). Formally, the convex polyhedron P can also be represented as $P = \{\sum_{v_i \in V} \lambda_i \cdot v_i + \sum_{r_j \in R} \mu_j \cdot r_j \mid \lambda_i \geq 0, \mu_j \geq 0, \sum_i \lambda_i = 1\}$. Figure 6 shows an example of a convex polyhedron and its double description.

The fact that there are two representations is important, because several of the operations for convex polyhedra are computed very efficiently when the proper representation is available. There are efficient algorithms [3, 10] that translate one representation into the other. Also, the dual representations can be used to keep a *minimal* description, removing redundant constraints and generators.

In the remaining of the paper, we will denote the number of tokens in a place p_i as x_i.

3.2 Abstract Semantics

The abstract domain of convex polyhedra provides all the operations required in abstract interpretation. This section will describe the implementation of the operations used in our problem: guards, assignments, test for inclusion $(\subseteq)$, union $(\cup)$ and widening (∇).

Initial Marking. The initial marking defines the number of tokens in each place, and therefore, the value of all token counter variables. Therefore, the convex polyhedron that represents the initial marking has the following system of constraints: $\wedge_i(x_i = m_i)$, where m_i is the initial number of tokens in place i.

Guards. There are two kinds of guards that arise in our analysis: guards testing the presence of tokens in the input nodes, of the form $(x_1 \geq 1)$; and guards describing inhibitor arcs, of the form $(x_2 = 0)$. In any case, these guards are linear inequalities, so the resulting convex polyhedron only adds these guards to its system of constraints.

Assignments. The assignments that appear in our analysis increase or decrease counter variables by a constant value, e.g. $(x_i := x_i \pm c)$. These assignments can be applied to a convex polyhedron by changing its system of generators: each vertex is modified by increasing variable x_i by the constant c.

Test for Inclusion. In order to decide whether a fixpoint has been reached, the convex polyhedra approximating the reachability set must be compared with the one computed in the previous iteration. This comparison is made using the test for inclusion $(P \subseteq Q)$, which requires both representations of polyhedra. A convex polyhedron P, whose system of generators is the set V of vertices and the set R of rays, is included in a polyhedron Q, whose system of constraints is $AX \geq B$, if and only if $\forall v \in V, Av \geq B$ and $\forall r \in R, Ar \geq 0$.

Union. The new markings discovered when a transition is fired must be added to the previously known set of markings using the union operator. In the convex polyhedra domain, the union of convex polyhedra is not necessarily a convex polyhedron. Therefore, the union of two convex polyhedra is approximated by the *convex hull*, the smallest convex polyhedron that includes both operands. The system of generators of the convex hull can be computed by joining the systems of generators of the operands.

Widening. The extrapolation operator on convex polyhedra works on the system of constraints. The widening $P \nabla Q$ can be simply defined as the inequalities from P that are also satisfied by Q. More complex definitions of widening may provide a better precision in the analysis [1, 15].

The firing of a transition can be modeled as a sequence of these operations. First, testing if the transition is enabled can be performed by guard operations that check the number of tokens in the input places, e.g. $(x_1 \geq 1)$? or $(x_2 = 0)$? for inhibitor arcs. Then, the changes in the number of tokens in a place are modeled as linear assignments, e.g. $(x_1 := x_1 + 1)$. The new reachable markings will be added to the current reachable set using the union and widening operator, as it was described in Section 2.2. Figure 8 shows an example of this computation.

Figure 7 shows several examples of the operations described in this section. Notice that the intersection and linear assignment are exact, while the union and the widening operations are approximate.

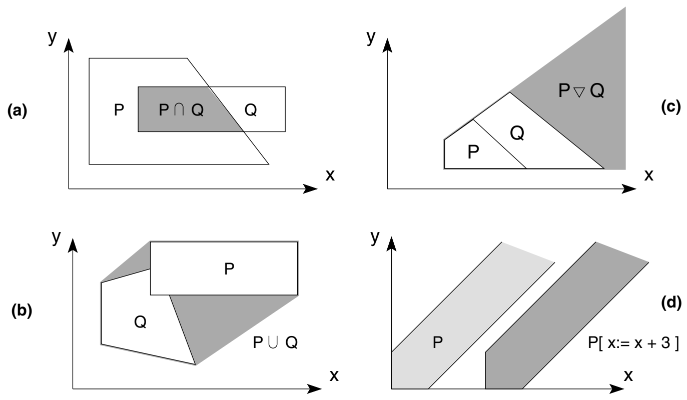
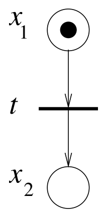


Fig. 7. Example of the operations on convex polyhedra: (a) intersection, (b) union, (c) widening and (d) linear assignment



$$(x_1 = 1) \wedge (x_2 = 0) \rightsquigarrow (x_1 + x_2 = 1) \wedge (0 \leq x_1 \leq 1) \wedge (0 \leq x_2 \leq 1)$$

(a)

$$(x_1 = 1) \wedge (x_2 = 0) \rightsquigarrow (x_1 + x_2 = 1) \wedge (x_2(x_2 - 1) = 0)$$

(b)

Fig. 8. Computation of the invariants when a transition is fired, in the case of (a) linear inequalities and (b) polynomial equalities

4 Polynomial Equality Invariants

4.1 Ideals of Polynomials

In the abstract domain of convex polyhedra (Section 3), we have represented states as solutions of a system of *linear inequalities*; now, in the domain of ideals of polynomials, we will consider states as solutions of a system of *polynomial equalities*.

Namely, we abstract as follows: given a set of states S, regarded as points in $\mathbb{Q}^n$, the corresponding abstraction is a set of polynomials P with rational coefficients such that $P(\sigma) = 0 \; \forall \sigma \in S$, i.e. all points in S are zeroes of P. This set of polynomials has the algebraic structure of an *ideal*: by definition, an ideal I is a set of polynomials such that it a) contains 0, b) is closed under addition, and c) for any polynomial P, if $Q \in I$ then $P \cdot Q \in I$. Thus, we take ideals as our abstract values: an ideal I is an abstraction of the common zeroes of its polynomials, $\{\sigma \in \mathbb{Q}^n | \; P(\sigma) = 0 \; \forall P \in I\}$, which we call the *variety* of I and denote by $\mathbf{V}(I)$.

The set of polynomials with rational coefficients is denoted as $\mathbb{Q}[X]$. Given a subset $S \subseteq \mathbb{Q}[X]$, the *ideal generated by* S is

$$\langle S \rangle = \{ f \in \mathbb{Q}[X] \mid \exists k \geq 1 \; f = \sum_{j=1}^{k} P_j Q_j \text{ with } P_j \in \mathbb{Q}[X], Q_j \in S \}.$$

For an ideal I, a set of polynomials S such that $I = \langle S \rangle$ is called a *basis* of I. By Hilbert's basis theorem, all ideals of polynomials admit a finite basis. Therefore any ideal is associated to a *finite* system of polynomial equality constraints: the ideal $I = \langle P_1(X), ..., P_k(X) \rangle$ corresponds to the system $\{P_1(X) = 0, ..., P_k(X) = 0\}$, or equivalently to the formula $\bigwedge_{j=1}^{k} P_j(X) = 0$.

For example, the ideal $\langle x(x^2 + y^2 - 16), \; y(x^2 + y^2 - 16) \rangle$ is associated to the system $\{x(x^2 + y^2 - 16) = 0 \, , y(x^2 + y^2 - 16) = 0\}$. Its solutions, which form the variety $\mathbf{V}(\langle x(x^2 + y^2 - 16) \; , y(x^2 + y^2 - 16) \rangle)$, are the union of a circle and a point, pictured in Figure 9. Notice that this set, unlike convex polyhedra, is not convex or even connected.

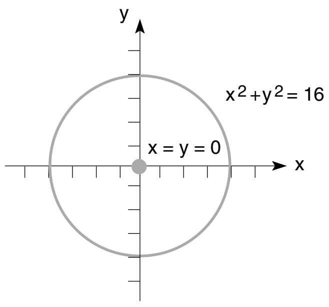


Fig. 9. An example of variety of an ideal

4.2 Abstract Semantics

This section shows the operations required to perform the abstract interpretation of Petri Nets using ideals of polynomials. For a more detailed description, see [22].[2]

Initial Marking. If we are given an initial marking $(m_1, m_2, ..., m_n)$ for the Petri Net (where each m_i may be a constant or a parameter), at first we know that $(x_i = m_i)$ for every place of the net, and so we take the ideal

$$\langle x_1 - m_1, x_2 - m_2, ..., x_n - m_n \rangle$$

as initial ideal.

Guards: Inhibitor Arcs. A guard describing an inhibitor arc, i.e. checking that there are no tokens in the inhibitor place p_i, is of the form $(x_i = 0)$. Similarly as we did with convex polyhedra, we have to add the guard to the system of constraints. In this case, we just need to add the polynomial x_i to the list of generators of the input ideal.

Guards: Presence of Tokens. Testing the presence of tokens is conservatively translated into polynomial disequality guards: checking that there are at least C tokens at place p_i is expressed as $(x_i \neq 0 \wedge \cdots \wedge x_i \neq C - 1)$. Given an input ideal I, for each of these disequalities $(x_i \neq c)$ we want to represent the points that belong to $\mathbf{V}(I)$ but not to $\mathbf{V}(\langle x_i - c \rangle)$, in other words $\mathbf{V}(I) \setminus \mathbf{V}(\langle x_i - c \rangle)$. The polynomials in the *quotient ideal* $I : \langle x_i - c \rangle$ [12] evaluate to 0 at this difference of sets, and therefore abstract the states we are interested in; so we take this quotient as output ideal.

Assignments. The assignments that appear in our analysis are of the form $x_i := x_i \pm c$, as they express the change in the number of tokens at place p_i after firing a transition. Given an ideal $I = \langle P_1, ..., P_k \rangle$, we want to compute the effect of applying the assignment $x_i := x_i + c$ on I (in case of a subtraction, we may take c as a negative value). In terms of formulas, we need to express the following assertion using ideals:

$$\exists x_i' (x_i = x_i' + c \wedge (\bigwedge_{j=1}^{k} P_j(x_i \leftarrow x_i') = 0)) \, ,$$

[2] The abstract domain of ideals of polynomials and its semantics have been simplified in this paper with respect to [22] for the sake of clearness and efficiency.

where x'_i stands for the value of the assigned variable previous to the assignment, and $\leftarrow$ denotes substitution of variables. In this case, the auxiliary variable x'_i can be easily eliminated by substitution, as $x'_i = x_i - c$. So we get the formula $\wedge_{j=1}^{k} P_j(x_i \leftarrow x_i - c) = 0$, which translated into ideals yields $I(x_i \leftarrow x_i - c)$.

Test for Inclusion. In order to check whether a fixpoint has been reached, we need to test if the newly computed reachable states, represented by the ideal I, are already included in our previous approximation given by I_{prev}. So we need to see if $\mathbf{V}(I) \subseteq \mathbf{V}(I_{prev})$, which can be done by duality by checking that $I \supseteq I_{prev}$.

Union. Unlike with convex polyhedra, we can perform exact unions of states in the domain of ideals of polynomials. Let I, J be ideals corresponding to the sets of states $\mathbf{V}(I)$ and $\mathbf{V}(J)$ respectively, and assume that we want to represent $\mathbf{V}(I) \cup \mathbf{V}(J)$. In this case the abstraction is given by the *intersection ideal* $I \cap J$, which satisfies that $\mathbf{V}(I \cap J) = \mathbf{V}(I) \cup \mathbf{V}(J)$.

Widening. In order to get termination if the initial marking has parameters or the Petri Net is not bounded, we need to introduce a widening operator. Similarly as we did with convex polyhedra, given the ideals I and J we have to perform an upper approximation of $\mathbf{V}(I) \cup \mathbf{V}(J)$. By duality, we have to compute a lower approximation of $I \cap J$; i.e., we need to sieve the polynomials in the intersection so that the result is still sound and also the analysis terminates in a finite number of steps without much loss of precision.

Given a degree bound $d \in \mathbb{N}$ and a graded term ordering $\succ$, our widening operator $I \nabla_d J$ is defined as the ideal generated by the polynomials of a Gröbner basis of $I \cap J$ (with respect to $\succ$) of degree at most d; more formally,

$$I \nabla_d J = \langle \{ P \in GB(I \cap J, \succ) \mid \text{degree}(P) \leq d \} \rangle,$$

where $GB(\cdot, \succ)$ stands for a Gröbner basis of an ideal with respect to the term ordering $\succ$. For definitions of Gröbner basis, graded term ordering and related concepts, we refer the reader to [12].

Figure 10 shows several examples of the operations described in this section. Contrary to convex polyhedra, the union operator is exact. Widening can be seen as a parametrized union, where any polynomial in the basis with a degree higher than the bound is abstracted. Varying this bound achieves different levels of precision in the result. For instance, Figures 10(b) and (c) show two widenings with degree bounds 2 and 3 respectively; notice that the latter represents exactly the union of states.

5 Examples

The techniques presented in the previous sections have been implemented and applied to several Petri Net examples from the literature. The linear inequality analysis has been implemented as a C program using the New Polka convex polyhedra library [21]. On the other hand, the polynomial equality analysis is performed by means of the algebraic geometry tool Macaulay 2 [14]. When it is not computationally feasible to work with polynomials over the rationals, we heuristically employ coefficients in a finite field instead. As the invariants obtained in the finite field might not necessarily be invariants in $\mathbb{Q}$, the polynomials thus generated are finally checked to be truly invariants of the system.

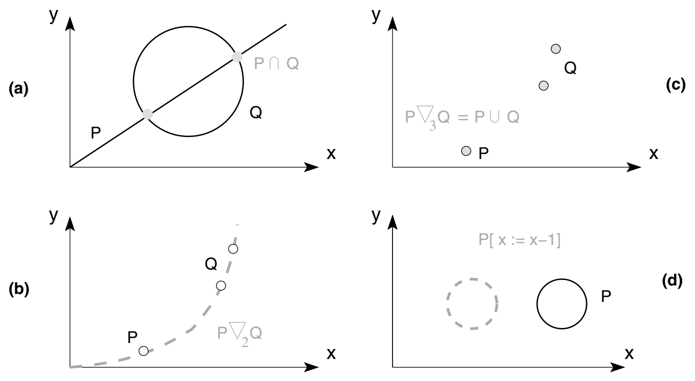


Fig. 10. Example of the operations on ideals of polynomials: (a) intersection, (b) widening with degree bound 2, (c) widening with degree bound 3 (union) and (d) linear assignment

5.1 Comparison with Structural Invariants

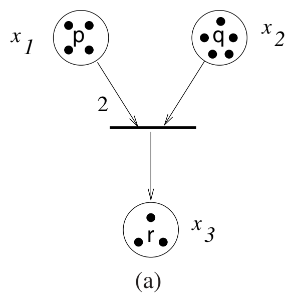
Structural invariants [20] are properties of the Petri Net structure, so they are independent of the initial marking. For instance, a Petri net may be *bounded* for a given initial marking, i.e. the number of tokens in each place is bounded in all reachable markings. Moreover, a net may be bounded for any initial marking, i.e. *structurally bounded*. Any structurally bounded net is also bounded, but the reverse does not necessarily hold. The approaches presented in this paper can be used to detect structural properties of a Petri Net: a parameter defines the initial marking of each place, hence the invariants obtained describe properties that are independent on the initial marking. For example, Figure 11(a) shows a Petri Net whose initial marking is defined by the parameters p, q and r: $x_1 = p$, $x_2 = q$, $x_3 = r$. In this Petri Net, the linear inequality invariants that can be computed with our approach appear in Figure 11(b). Notice that these invariants are structural as they are satisfied by any initial marking; for instance, $(r + p \geq x_3)$ meansthat place x_3 is structurally bounded. Invariant polynomial equalities can be similarly obtained using the same concept.

If a net is bounded, the analysis with polynomial invariants discovers the exact state space, for a sufficiently large degree. Thus, if a net is bounded but not structurally bounded, the analysis with polynomial invariants obtains a description of the state space which is more precise than structural invariants.

Regarding the analysis with convex polyhedra, or Petri nets with an infinite state space, using a widening adds some approximation. This approximation may make us fail to discover some structural invariants. In practice, in all the examples that we have studied, the state space computed by abstract interpretation satisfies all the structural invariants.

Furthermore, invariants describing properties which depend on the initial marking can also be computed with our approach. For example, Figure 12 shows a Petri Net where a place p is bounded for the initial marking depicted in the figure, while it is not structurally bounded. Abstract interpretation analysis discovers this property, encoded as the linear inequalities $(0 \leq p \leq 1)$ or the polynomial equality $p \cdot (p - 1) = 0$.

Another example of a non-structural property discovered by these invariants appears in Figure 13. This Petri Net can have a deadlock depending on the initial marking. For



$$
\begin{aligned}
2x_3 + x_1 &= 2r + p \\
x_3 + x_2 &= r + q \\
x_3 &\geq r \\
r + p &\geq x_3 \\
r + q &\geq x_3
\end{aligned}
$$

(b)

Fig. 11. (a) Petri Net with a parametric initial marking and (b) the computed linear inequality invariants

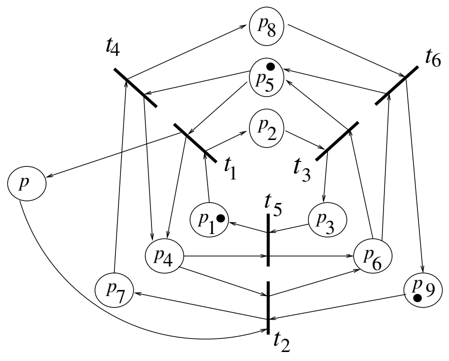


Fig. 12. Petri Net with a bounded place p which is not structurally bounded

instance, for the initial marking p_1p_8 there is no deadlock, while for the initial marking p_1 the deadlocks p_3p_8 and p_5p_8 are reachable. The invariants produced by both linear inequalities and polynomial equalities are sufficient to prove the absence of deadlocks for the former initial marking, as the conjunction of these invariants with the disabling conditions of all the transitions is unfeasible.

5.2 Automated Manufacturing System

Figure 14 shows a Petri Net model of an automated manufacturing system [25]. This manufacturing system consists of several elements: four machines $(M_1 - M_4)$, two robots $(R_1 - R_2)$, two buffers with capacity 3 $(B_1 - B_2)$ and an assembly cell. The place x_1 models the entry point for raw material, while the place x_{10} (x_{15}) represents the availability of the buffer B_1 (B_2). The place x_{12} (x_{13}) models the availability of the robot R_1 (R_2), whereas the place x_{25} represents the delivery point for the final product. Finally, the places x_4, x_7, x_{16} and x_{19} model the availability of the machines M_1 to M_4.

The initial marking of this Petri Net is as follows. The entry point x_1 has an undetermined number of tokens p, as we want to study the behavior of the system depending on the quantity of available raw materials. The capacities of the buffers, x_{10} and x_{15}, have 3 tokens as the buffers have size 3. Finally, places x_2, x_4, x_7, x_{12}, x_{13}, x_{16}, x_{19} and x_{24} have one token, and the rest of places have no tokens in the initial marking.

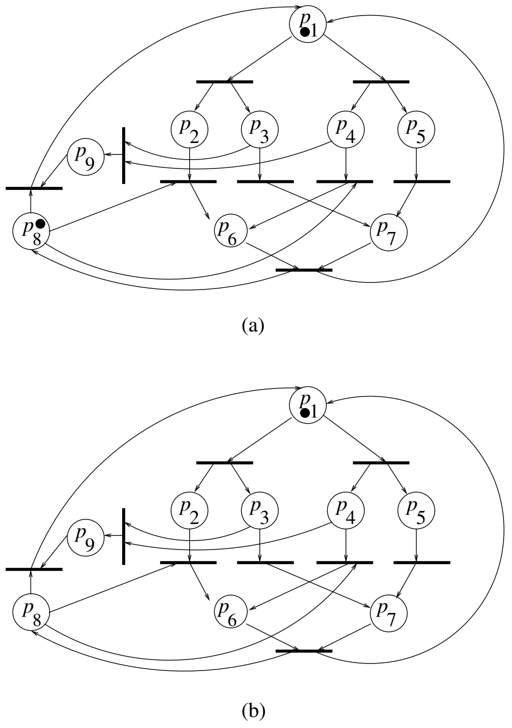


Fig. 13. Petri Net with a non-structural deadlock: (a) no deadlock, (b) potential deadlock

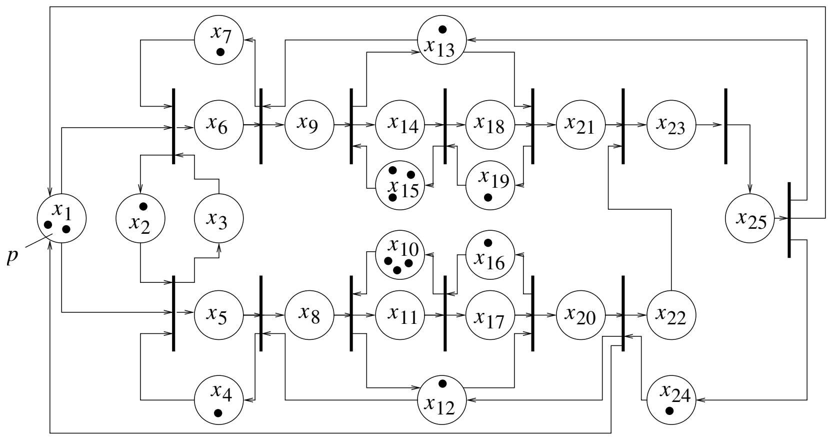


Fig. 14. Petri Net model of an automated manufacturing system

Some relevant properties in this system are *boundedness* and *liveness*, i.e. deadlock freedom. In previous work, these properties have been studied in detail. In [25], it was proven that the system is bounded, and that it is live only for some values of p, namely $2 \leq p \leq 4$. A different approach based on integer programming [4] managed to prove liveness for a wider interval of values, $1 \leq p \leq 8$. Also, a sequence of firings leading to a deadlock when $p > 8$ was shown. Later work has revisited these results using other techniques such as Presburger arithmetics [13], real arithmetics [2] and inductive linear inequalities based on Farkas' lemma [23].

We have analyzed the manufacturing system using both linear inequality invariants and polynomial equality invariants. Using linear inequalities, the computation takes 2.2 seconds and 68 Mb of memory. The resulting invariants are the following (we show the equalities and the inequalities separatedly):

$$\begin{array}{ll}
x_{18} + x_{19} = 1 & x_2 + x_3 = 1 \\
x_8 + x_{12} + x_{20} = 1 & x_4 + x_5 = 1 \\
x_{22} + x_{23} + x_{24} + x_{25} = 1 & x_6 + x_7 = 1 \\
x_9 + x_{13} + x_{21} + x_{23} + x_{25} = 1 & x_{10} + x_{11} = 3 \\
x_{19} + x_{17} + x_{15} + x_{13} + x_{11} + x_7 + x_5 + x_2 - x_{24} - x_{12} = 5 & x_{14} + x_{15} = 3 \\
x_{19} + x_{17} + x_{15} + x_{13} + x_{11} + x_7 + x_5 - x_{24} - x_{12} - x_3 = 4 & x_{16} + x_{17} = 1 \\
x_{19} + x_{15} + x_{13} + x_{12} + x_7 + p - x_{17} - x_{11} - x_5 - x_1 = 7 & \\
\hline
x_{25} + x_{24} + x_{23} \le 1 & x_7 \le 1 \\
x_{25} + x_{23} + x_{21} + x_{13} \le 1 & x_{15} \le 3 \\
x_{19} + x_{15} + x_{13} + x_7 - x_{24} - x_1 \le 5 & x_{19} \le 1 \\
x_{19} + x_{17} + x_{15} + x_{13} + x_{11} + x_7 + x_5 - x_{24} - x_{12} \ge 4 & x_{20} + x_{12} \le 1
\end{array}$$

This set of constraints suffices to prove that the manufacturing system is bounded.

On the other hand, we have applied our analysis with ideals so as to discover polynomial equality invariants of degree at most 2. In order to speed up the computation, we have employed a finite field $\mathbb{Z}_p$ instead of $\mathbb{Q}$, with p a relatively big prime number (in particular, we have taken $p = 32749$, the largest prime allowed in Macaulay 2); the generation of the candidate invariants takes 16 minutes and 304 Mb of memory. After checking that the polynomials obtained are invariant, which requires 6 minutes and 490 Mb, we get (we separate the linear invariants from the quadratic ones):

$$\begin{array}{ll}
x_8 + x_{12} + x_{20} = 1 & x_2 + x_3 = 1 \\
x_9 + x_{13} + x_{21} + x_{23} + x_{25} = 1 & x_4 + x_5 = 1 \\
x_{24} + 2x_{16} + 2x_{12} + 2x_{10} + 2x_4 - x_2 - x_1 + p = 12 & x_6 + x_7 = 1 \\
x_{16} - x_{18} - x_{14} + x_{13} + x_{12} + x_{10} - x_6 + x_4 - x_1 + p = 7 & x_{10} + x_{11} = 3 \\
x_{19} + x_{16} - x_{14} + x_{13} + x_{12} + x_{10} - x_6 + x_4 - x_1 + p = 8 & x_{14} + x_{15} = 3 \\
x_{21} - x_{22} + 2x_{16} + x_{13} + 2x_{12} + 2x_{10} + x_9 + 2x_4 - x_2 - x_1 + p = 12 & x_{16} + x_{17} = 1 \\
\hline
x_2^2 = x_2 & x_8 x_{12} = 0 \\
x_4^2 = x_4 & x_9 x_{13} = 0 \\
x_6^2 = x_6 & x_{13} x_{21} = 0 \\
x_8^2 = x_8 & x_9 x_{21} = 0 \\
x_9^2 = x_9 & x_9 x_{23} = 0 \\
x_{12}^2 = x_{12} & x_{21} x_{23} = 0 \\
x_{13}^2 = x_{13} & x_{13} x_{23} = 0 \\
x_{16}^2 = x_{16} & x_{23}^2 = x_{23} \\
x_{21}^2 = x_{21} &
\end{array}$$

and four other more complex polynomial constraints.

Unlike with linear inequalities, these invariants are not enough to prove that *all* places in the net are bounded; the reason is that polynomial equalities cannot express relations such as $x_1 \le p$, where the bounds are parametric. However, *some* of the places can be shown to be bounded: for instance, $x_2 = x_2^2$ means that either $x_2 = 0$ or $x_2 = 1$. Notice that each family of constraints uses a different approach to represent boundedness: linear inequalities give an upper bound on the number of tokens (for example, the linear invariant $x_{20} + x_{12} \leq 1$ implies $x_{12} \leq 1$), whereas polynomial equalities encode an exact disjunction of the possible values.

Further, by means of these quadratic constraints, together with the implicit invariant that all variables are non-negative, it is also possible to prove that the system is deadlock-free for $1 \leq p \leq 8$; in order to do that, we show that the conjunction of the invariants and the disabling conditions of all the transitions is not satisfiable. Moreover, as in [23], for $p = 9$ we are able to isolate four potential deadlocks, one of which is the deadlock corresponding to the sequence of firings in [4] mentioned above.

5.3 Alternating Bit Protocol

The Petri Net in Figure 15 models the alternating bit protocol for retransmitting lost or corrupted messages. The correctness of the protocol has already been shown in previous work: while in [11] all the proofs were done by hand, in [13] Fribourg and Olsén employed Presburguer arithmetics to automatically characterize the reachable states and thus prove that the system behaves properly.

The initial marking is $x_1 = 1$, $x_{13} = 1$ and $x_i = 0$ for $1 \leq i \leq 16$, $i \neq 1, 13$. Notice that the net has eight inhibitor arcs, linked to the places x_j, $j = 5..12$, which can be proved to be unbounded; these inhibitor arcs are pictured as circle-headed arrows on the figure. Thus, this example cannot be handled by other techniques for generating invariants that do not deal with equality guards, such as [18].

For this Petri Net, by means of convex polyhedra the following linear constraints:

$$x_4 + x_3 + x_2 + x_1 = 1 \qquad x_{16} + x_{15} + x_{14} + x_{13} = 1$$

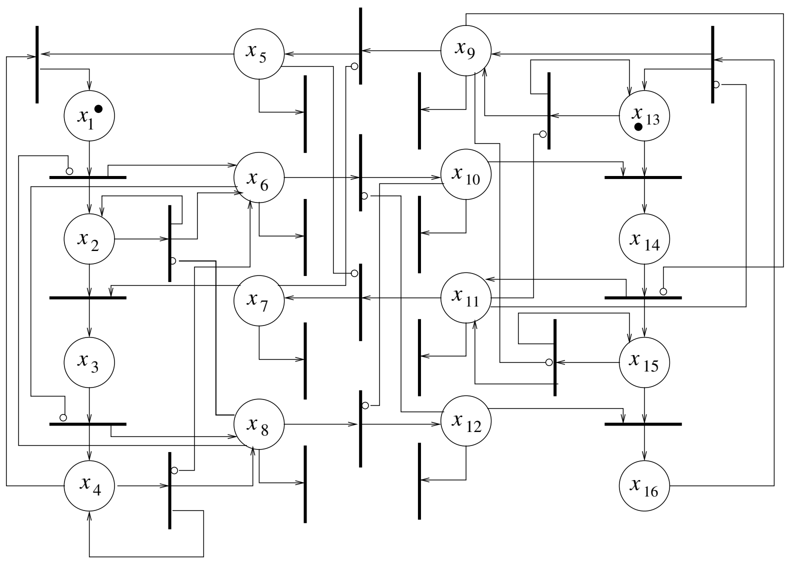


Fig. 15. Petri Net model of the alternating bit protocol

are obtained as invariants in 1.1 seconds using 65 Mb of memory. In this case, these linear invariants are not enough to show correctness, due to the fact that convex polyhedra cannot represent disjunctions in general.

As regards the analysis with polynomials, in 74.7 seconds and using 21 Mb of memory we get the following linear and quadratic constraints:

$$\begin{array}{lll}
\multicolumn{2}{l}{x_4+x_3+x_2+x_1=1} & x_{16}+x_{15}+x_{14}+x_{13}=1 \\
\hline
x_1x_2=0 & x_2x_3=0 & x_1^2=x_1 \\
x_1x_3=0 & x_1x_6=0 & x_2^2=x_2 \\
x_2x_8=0 & x_3x_8=0 & x_3^2=x_3 \\
x_5x_7=0 & x_6x_8=0 & x_{13}^2=x_{13} \\
x_9x_{11}=0 & x_{10}x_{12}=0 & x_{14}^2=x_{14} \\
x_9x_{15}=0 & x_{11}x_{13}=0 & x_{15}^2=x_{15} \\
x_{11}x_{14}=0 & x_{13}x_{14}=0 & x_6x_3+x_6x_2=x_6 \\
x_{13}x_{15}=0 & x_{14}x_{15}=0 & x_{14}x_9+x_{13}x_9=x_9
\end{array}$$

Note that, out of the 26 computed constraints, just the first two are linear and coincide with the linear equalities obtained above using convex polyhedra; the rest of the polynomial constraints are implicitly defining disjunctions. This explains why the linear inequality analysis does not yield enough information to verify the system.

Unfortunately, the quadratic invariants do not suffice to prove the correctness of the Petri Net either. This leads us to generate *cubic* invariant polynomials, i.e. of degree 3; the computation takes 102.9 seconds and 34 Mb of memory and gives

$$\begin{array}{lll}
x_2x_7x_9=0 & x_7x_9x_{10}=0 & x_2x_{12}x_{13}=x_2x_{12} \\
x_2x_7x_{12}=0 & x_7x_{10}x_{13}=0 & x_5x_8x_{13}=x_5x_8 \\
x_2x_7x_{13}=0 & x_8x_9x_{10}=0 & x_5x_{12}x_{13}=x_5x_{12} \\
x_2x_{11}x_{12}=0 & x_2x_5x_6=x_5x_6 & x_6x_{12}x_{13}=x_6x_{12} \\
x_5x_8x_{10}=0 & x_2x_5x_{10}=x_5x_{10} & x_7x_9x_{13}=x_7x_9 \\
x_5x_8x_{11}=0 & x_2x_5x_{11}=x_5x_{11} & x_7x_9x_{13}=x_7x_9 \\
x_5x_{11}x_{12}=0 & x_2x_6x_9=x_6x_9 & x_8x_9x_{13}=x_8x_9 \\
x_6x_7x_9=0 & x_2x_6x_{12}=x_6x_{12} & x_9x_{12}x_{13}=x_9x_{12} \\
x_6x_7x_{12}=0 & x_2x_6x_{13}=x_6x_{13} & x_2x_5x_{13}+x_5=x_5x_{13}+x_2x_5 \\
x_6x_7x_{13}=0 & x_2x_9x_{10}=x_9x_{10} & x_2x_9x_{13}+x_9=x_9x_{13}+x_2x_9 \\
x_6x_{11}x_{12}=0 & x_2x_{10}x_{13}=x_{10}x_{13} &
\end{array}$$

in addition to the quadratic constraints above. Unlike with the quadratic case, the cubic invariants allow us to prove that

$$\begin{cases}
x_1=1 \Longrightarrow (x_2=x_3=x_4=x_6=x_7=x_{10}=x_{11}=x_{14}=x_{15}=x_{16}=0)\wedge(x_{13}=1) \\
x_3=1 \Longrightarrow (x_1=x_2=x_4=x_5=x_8=x_9=x_{12}=x_{13}=x_{14}=x_{16}=0)\wedge(x_{15}=1) \\
x_{14}=1 \Longrightarrow (x_1=x_3=x_4=x_7=x_8=x_{11}=x_{12}=x_{13}=x_{15}=x_{16}=0)\wedge(x_2=1) \\
x_{16}=1 \Longrightarrow (x_1=x_2=x_3=x_5=x_6=x_9=x_{10}=x_{13}=x_{14}=x_{15}=0)\wedge(x_4=1)
\end{cases}$$

which implies that the system is correct (see [11]).

6 Conclusions

The applicability of abstract interpretation can be extended to the analysis of Petri Nets. This paper has presented an approach that can generate a rich set of invariants using this paradigm.

Abstract interpretation is a general approach that accepts different algorithmic techniques to calculate approximations. We believe that different strategies can be studied to explore the trade-off between efficiency and accuracy in analyzing concurrent systems.

Acknowledgements. This work has been partially funded by CICYT TIC2001-2476-C03-01, CICYT TIN 2004-07925, the LogicTools project (CICYT TIN 2004-03382), and the FPU grants AP2002-3862 and AP2002-3693 from the Spanish MECD. We would also like to thank M. Müller-Olm for insightful discussions.

References

1. R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In R. Cousot, editor, *Int. Symp. on Static Analysis*, volume 2694 of *Lecture Notes in Computer Science*, pages 337–354, San Diego, California, USA, 2003. Springer-Verlag.
2. B. Bérard and L. Fribourg. Reachability analysis of (timed) Petri nets using real arithmetic. In *Proc.Int. Conf. on Concurrency Theory*, volume 1664 of *LNCS*, pages 178–193, 1999.
3. N. Chernikova. Algoritm for discovering the set of all solutions of a linear programming problem. *USSR Computational Mathematics and Mathematical Physics*, 6(8):282–293, 1964.
4. F. Chu and X.-L. Xie. Deadlock analysis of Petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*, 13(6):793–804, Dec. 1997.
5. F. Commoner. *Deadlocks in Petri Nets.* Wakefield: Applied Data Research, Inc., CA-7206–2311, 1972.
6. P. Cousot. Abstract interpretation: Achievements and perspectives. In *Proc. of the SSGRR 2000 Computer & eBusiness Int. Conf.* Scuola Superiore G. Reiss Romoli, July 2000.
7. P. Cousot. Abstract interpretation based formal methods and future challenges, invited paper. In R. Wilhelm, editor, *Informatics — 10 Years Back, 10 Years Ahead*, volume 2000 of *Lecture Notes in Computer Science*, pages 138–156. Springer-Verlag, Berlin, Germany, 2001.
8. P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. of the 2nd Int. Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.
9. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, pages 238–252. ACM Press, 1977.
10. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, pages 84–97. ACM Press, New York, 1978.
11. J. M. Couvreur and E. Paviot-Adet. New structural invariants for petri nets analysis. In Valette, R., editor, *Proc. Int. Conf. on Application and Theory of Petri Nets*, volume 815 of *LNCS*, pages 199–218. Springer-Verlag, 1994.
12. D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer-Verlag, 1998.
13. L. Fribourg and H. Olsén. Proving safety properties of infinite state systems by compilation into presburger arithmetics. In *Proc.Int. Conf. on Concurrency Theory*, volume 1243, pages 213–227. Lecture Notes in Computer Science, July 1997.
14. D. R. Grayson and M. E. Stillman. Macaulay 2, a Software System for Research in Algebraic Geometry. Available at `http://www.math.uiuc.edu/Macaulay2/`.
15. N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.

16. G. Memmi and J. Vautherin. Computation of flows for unary-predicates/transition nets. *Lecture Notes in Computer Science: Advances in Petri Nets 1984*, 188:455–467, 1985.
17. A. Miné. The octagon abstract domain. In *Analysis, Slicing and Tranformation (in Working Conference on Reverse Engineering)*, IEEE, pages 310–319. IEEE CS Press, Oct. 2001.
18. M. Müller-Olm and H. Seidl. Computing Polynomial Program Invariants. *Information Processing Letters (IPL)*, 91(5):233–244, 2004.
19. T. Murata. State equation, controllability, and maximal matchings of petri nets. *IEEE Trans. Autom. Contr.*, 22(3):412–416, June 1977.
20. T. Murata. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*, 77(4), 1989.
21. New Polka: Convex Polyhedra Library. http://www.irisa.fr/prive/bjeannet/newpolka.html.
22. E. Rodríguez-Carbonell and D. Kapur. An Abstract Interpretation Approach for Automatic Generation of Polynomial Invariants. In *Int. Symp. on Static Analysis (SAS 2004)*, volume 3148 of *Lecture Notes in Computer Science*, pages 280–295. Springer-Verlag, 2004.
23. S. Sankaranarayanan, H. Sipma, and Z. Manna. Petri net analysis using invariant generation. In *Verification: Theory and Practice*, pages 682–701. Springer-Verlag, 2003.
24. M. Silva, E. Teruel, and J. M. Colom. Linear algebraic and linear programming techniques for the analysis of place/transition net systems. *Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models*, 1491:309–373, 1998.
25. M. Zhou, F. DiCesare, and A. Desrochers. A hybrid methodology for synthesis of Petri net models for manufacturing systems. *IEEE Transactions on Robotics and Automation*, 8(3):350–361, June 1992.