1991

Logical Inference of Horn Clauses in Petri Net Models

C. Lin

A. Chaudhury

A. B. Whinston

Dan C. Marinescu

Report Number:
91-031

LOGICAL INTERFERENCE OF HORN CLAUSES IN PETRI NET MODELS

C. Lin
A. Chaudhury
A. B. Whinston
Dan C. Marinescu

CSD-TR-91-031
April 1991

LOGICAL INFERENCE OF HORN CLAUSES IN PETRI NET MODELS

C. Lin*
Information Science Institute
State Information Center, Beijing, China

A. Chaudhury and A. B. Whinston
Department of Management Science and Information Systems
Graduate School of Business
University of Texas, Austin

Dan. C. Marinescu†
Department of Computer Science
Purdue University
West Lafayette, IN 47907

April 8, 1991

Abstract

This paper studies Petri net models for the Horn clause form of propositional logic and of first order predicate logic. A net model for logical inconsistency check is proposed. Algorithms for computing T-invariants of Petri net models of logical inference systems are investigated. The algorithms are based on the idea of resolution and exploit the presence of one-literal, pure-literal and splitting clauses to lead to faster computation. Algorithms for computing T-invariants of High Level Petri net models of predicate logic are presented.

Keywords: Logical Inference, Horn Clause, Resolution, Petri Nets, T-invariants, High Level Petri nets.

---

*Work partially supported by IC^2 Institute, University of Texas, Austin

†Work partially supported by SDI under ARO contract DAAL03-86-K-0106, by the NATO grant 891007 and by the NSF grant NCR-8702115

1 Introduction

The problem of inference is fundamental in Artificial Intelligence, AI. The role of inference is central in various applications such as data bases, expert systems, decision support systems, and logic programming. The process of inference determines whether a given proposition is implied by a massive collection of data (rules) and, furthermore, whether a proof procedure and answer extraction can be drawn from these data to answer a particular question.

Different models for the representation of knowledge inference systems are known. For example directed and acyclic networks are used as a syntactic device for representing facts in a first order logic system [Nil 80]. Directed networks are also used to represent belief-networks and probabilistic dependencies [Pea 86]. Petri nets are chosen to model logical inference because Petri nets are themselves good models for describing parallelism, nondeterminism and asynchronous characteristics; in addition, there is a well-developed net theory. Transforming logical inference into Petri net models and using existing Petri net analysis methods to handle logical inference enhance the chance of treating problems of inference in an efficient manner.

Petri nets have been proposed to represent first order propositional logic [Rei 85], modal logic [Gen 79], predicate calculus [Gio 85], and linear logic [GuG 89]. For modeling logical inference Lautenbach [Lau 85], Sinachopoulos [Sin 87], and Murata and his co-workers [PeM 89] give the transformation procedure from a set of clauses to a Petri net model, and a necessary net theoretical condition for a set of clauses and a sufficient net theoretical condition for a set of Horn clauses to be unsatisfiable. It is shown that the goal transition of the Petri net model of a set of Horn Clauses is potentially firable iff there exists a non-negative T-invariant which includes the goal transition in its support.

The focus of this paper is the treatment of Horn clauses. Horn clauses are an important subset of clausal form because any problem which can be expressed in logic can be re-expressed by means of Horn clauses [Kow 79]. A contribution of this paper is to show that the inference methods for the Horn clauses have their counterparts in the structural methods used to analyze Petri net models. The main motivation of this work is to provide new insights for computation of T-invariants for logical inference in Petri net models and to reveal analogies among logical inference and T-invariants methods in Petri net analysis. These relationships show that various techniques can be applied to solve large inference problems and they can lead to fast inference methods. The linear representation and invariant techniques of Petri net plays a central role in logical inference models and inference methods as described in this paper.

This paper is organized as follows. A brief review of Petri nets, Petri net models of propositional logic, and the mapping of a set of Horn clauses to the incidence matrix of a Petri net are the topics of Section 2. The application of the logical concepts of resolution, one-literal, pure-literal and splitting in computing T-invariants of first order predicate logic models are covered in Section 3. Section 4 discusses High Level Petri Net models of first order predicate logic. An algorithm for computing T-invariants in High Level Petri nets, HLPNs, models of predicate logic is introduced in Section 5. Section 6 concludes the paper and discusses future research directions.

2 Petri Nets and Models of Propositional Logic

Petri nets and related concepts, such as marking, incidence matrix and T-invariants are introduced. Then Horn clauses and their representation as Petri nets are discussed. In this paper N denotes the set of natural numbers and Z the set of integers.

2.1 Definitions and Terminology

Definition 2.1. A tuple $PN = (S,T;F,M_0,W)$ is called a Petri net iff:

(a) $(S,T;F)$ is a finite net with S the set of places, T the set of transitions, F the set of arcs. The following properties hold:

- $S \cap T = \phi$ (duality between places and transitions);
- $F \subseteq (S \times T) \cup (T \times S)$ (the flow relation holds only between places and transition or vice versa);
- $S \neq \emptyset$ and $T \neq \emptyset$ (no empty net);
- $\mathrm{Dom}(F) \cup \mathrm{Cod}(F) = S \cup T$ (no isolated elements).

(b) $M_0 : S \rightarrow N$ is the *initial marking.*

(c) $W : F \rightarrow \{1\}$ is a *weight function* associated with each arc of the net.

Let $X = S \cup T$ be the set of elements of the net. The *pre-set (post-set) of an element* $x \in X$ is denoted by ${}^\circ x = \{y | y \in X, (y,x) \in F\}$ ($x^\circ = \{y | y \in X, (x,y) \in F\}$).

Definition 2.2. (Marking, Follower Marking and Firing Sequences). Let $PN = (S,T;F,M_0,W)$ be a Petri net.

(a) A *marking* M of PN is a mapping $M : S \rightarrow N$, for all $s \in S$. It represents the distribution of tokens over places.

(b) A transition $t \in T$ is *enabled* at marking M iff $\forall s \in {}^\circ t$, $M(s) \geq W(s,t)$.

(c) If $t \in T$ is enabled at marking M and t fires, we call M' the *follower marking* of M, denoted by $M[t > M'$ iff for each $s \in S, M'(s) = M(s) - W(s,t) + W(t,s)$.

(d) A *firing sequence* $\sigma = < t_{i1}, t_{i2}, \ldots, t_{in} >$ is said to be executable from M_0 if t_{i1} is fireable from M_0 and leads to the follower marking M_1, then t_{i2} is firable from M_1 and leads to M_2, and so on, for all transitions in σ.

Definition 2.3. (Incidence Matrix). Let $PN = (S,T;F,M_0,W)$ be a Petri net with n transitions and m places. A matrix $\mathbf{C} = [C_{ij}]$ is a $n \times m$ matrix of integers so that $C_{ij} = W(t_i,s_j) - W(s_j,t_i)$, where t_i represents transition i and s_j is place j. $\mathbf{C}_{ij}$ is the weight of the arc from transition i to place j minus the weight of the arc from place j to transition i. $\mathbf{C}$ is called the *incidence matrix* of PN. $\mathbf{C}^T$ is the transpose of the incidence matrix.

Denote the t^{th} row of C by $C^{(t)}$ and note that $C^{(t)}$ is associated with transition t. The definition of the follower marking can be rewritten as

$$M'(s) = M(s) + C^{(t)}.$$

Definition 2.4. (T-invariants). Let C be the incidence matrix of a net. An n-vector of integers X, is called a T-invariant if $\mathbf{C}^T X = 0$. The i^{th} entry of a vector X is denoted by $X(i)$. The support of a T-invariant X, denoted by $\|X\|$, is the set of transitions whose components in X are strictly positive. A support is minimal iff it does not contain the support of another invariant except itself and the empty set. A T-invariant $X \geq 0$, is said to be executable from marking M if there exists a firing sequence σ executable from the marking M, such that its firing count vector $\overline{\sigma} = X$. Since $\mathbf{C}^T X = 0$, it follows that an n-vector $X \geq 0$ is a T-invariant iff there exist a marking M and a firing sequence σ from M back to M such that $\overline{\sigma} = X$.

Definition 2.5. (Multisets, Sequences, Parikh mapping of a multiset). A multi-set (or a bag) P' is a function defined on a non-empty set P, $P' \in [P \rightarrow N]$ where $[P \rightarrow N]$ is the space of all functions from P to N. Intuitively, a multi-set is a set which can contain multiple occurrences of the same element. For example, $B = \{a, c, c\}$ is a multiset over the domain $\{a, b, c\}$. Call $\#(a, B)$ the number of occurrences, of the element a in the multiset B. Then in the previous example $\#(a, B) = 1$ and $\#(c, B) = 2$. An ordered multiset is called a *sequence*. A sequence σ can be converted to corresponding multiset B by $B = \text{Bag}(\sigma)$.

The Parikh mapping of a multiset B is denoted by $\psi(B)$ and is defined by

$$\psi(B) = (\#(t_1, B), \ldots, \#(t_n, B))$$

where $\{t_1, \ldots, t_n\}$ is the domain for B and $\#(t_i, B)$ is the number of t_i in the multiset B. $\psi(\text{Bag}(\sigma))$ the Parikh mapping of a firing sequence σ is denoted by $\overline{\sigma}$.

2.2 Modeling Horn Clauses

A Horn clause of propositional logic has the form

$$B \leftarrow A_1 \wedge A_2, \ldots, \wedge A_n.$$

This notation means that holding of all conditions A_1 to A_n implies the conclusion B. Logical connectiveness is expressed using the $\leftarrow$ (implication) and $\wedge$ (conjunction) symbols. A Horn clause is a clause in which the conjunction of zero or more conditions implies at most one conclusion. There are four different forms of Horn clauses. The Petri net representation of Horn clauses are:

1. The Horn clause with non-empty condition(s) and conclusion

   $$B \leftarrow A_1 \wedge A_2, \ldots, \wedge A_n \text{ with } n \geq 1.$$

   For example, the clause $C \leftarrow A \wedge B$ is represented by the Petri net in Figure 1a. When the conditions A and B are true, the corresponding places A and B hold tokens, transitions t fires, and a token is deposited in place C, i.e., the conclusion C is true.

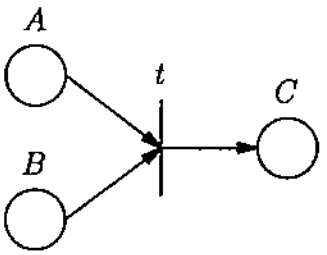

Figure 1a. A Horn clause with two condition and a conclusion.

2. The Horn clause with empty condition(s)

$$B \leftarrow .$$

This type of Horn clause is interpreted as an assertion of facts. A fact B can be represented in a Petri net model as a transition system with a *source* transition, as shown in Figure 1b.

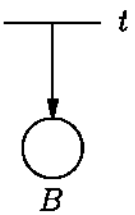

Figure 1b. A Horn clause with no condition.

The source transition t is always enabled and this means that the formula B is always true.

3. The Horn clause with empty conclusion

$$\leftarrow A_1 \wedge A_2, \ldots, A_n \text{ with } n \geq 1.$$

This type of Horn clause is interpreted as the goal statement which is in the negation form of what is to be proven. In a Petri net model a condition like 'A_1 and A_2' is represented as a goal transition system with a *sink transition*, as shown in Figure 1c.

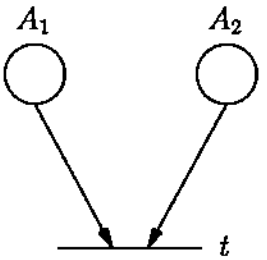

Figure 1c. A Horn clause with empty conclusion.

4. The null clause, which is interpreted as a contradiction. There is no representation of such clause, the *empty net* is not defined in the net theory, see Definition 2.1.

2.3 Mapping of a Set of Horn Clauses into a Petri Net

Given a set of Horn clauses consisting of n clauses and m distinct symbols, the $n \times m$ incidence matrix $\mathbf{C} = [C_{ij}]$ of a Petri net corresponding to the set of clauses can be obtained by the following procedure given by Murata and Zhang [MuZ 88].

Step 1: Denote the n *clauses* by $t_1, \ldots, t_n$. The clause t_i represents the i^{th} row of $\mathbf{C}$.

Step 2: Denote the m *predicate symbols* by $P_1, \ldots, P_m$. The symbol P_j represents the j^{th} column of $\mathbf{C}$.

Step 3: The $(i,j)^{th}$ entry of $\mathbf{C}$, C_{ij}, is the sum of the arguments in the i^{th} clause and the j^{th} symbol. The sum is taken over all the j^{th} symbols appearing in the i^{th} clause. All the arguments to the left hand side of the $\leftarrow$ operator are taken as positive, and all the arguments to the right hand side of it are taken as negative. Thus the elements C_{ij} can be either '0', or '1' or '-1'.

The following example shows the translation procedure.

Example 1: (based on [PeM 89]).

Given the following set of Horn clauses represented in the conventional way

1) A	2) B
3) $A \wedge B \rightarrow C$	4) $C \wedge B \rightarrow D$
5) $D \rightarrow A$	6) $D \rightarrow C$

To prove that $D \wedge C$ is true, one can apply the satisfiability principle. Let S be a set of first order formula and G be a first order formula. G is a *logic consequence* of S iff $S \cup (\neg G)$ is unsatisfiable. The following result is obtained by adding the negation of $D \wedge C$ to the set of clauses

1) A	2) B
3) $C \vee \neg A \vee \neg B$	4) $D \vee \neg B \vee \neg C$
5) $A \vee \neg D$	6) $C \vee \neg D$
7) $\neg D \vee \neg C$	

The Petri net representation of this set of Horn clauses and its incidence matrix are shown in Figure 2.

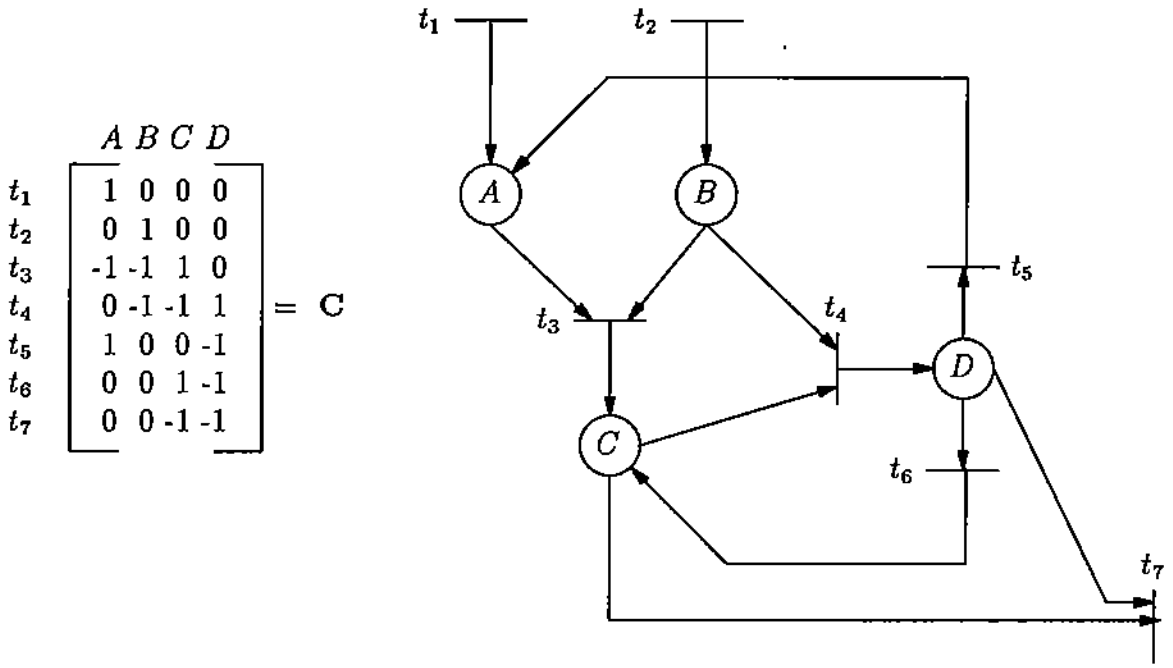

Figure 2. The Incidence Matrix and the Petri Net for the set of Horn clauses in Example 1.

3 An Algorithm for Logical Inference in a Petri Net Model of Propositional Logic

Sinachopoulos [Sin 87], Lautenbach [Lan 85] and Murata [PeM 89] have investigated the necessary and sufficient conditions for a set of Horn clauses to contain a contradiction based on analysis of the Petri net model of such a clauses. These conditions are:

Theorem 3.1 [Sin 87]. A necessary net theoretical condition for a set of clauses J, to be unsatisfiable is that the net representation of J has a non-negative T-invariant.

Theorem 3.2 [Sin 87]. A sufficient net theoretical condition for a set of Horn clauses J, to be unsatisfiable is that J contains at least one source transition, at least one sink transition, and has a non-zero T-invariant.

Theorem 3.3. Let $PN = (S, T; F, M_0, W)$ be a Petri net representation of a set of Horn clauses. Let t_g be a goal transition in T. There exists a firing transition sequence which reproduces the empty marking ($M = 0$) and fires the goal transition t_g in PN iff PN has a T-invariant X such that $X \geq 0$ and $X(t_g) \neq 0$. X is a vector and the value of its t_g^{th} element is given by $X(t_g)$.

The algorithm for computing the T-invariants of a Petri net due to Martinez and Silva [MaS 82] is discussed next. Consider a net with n transitions and m places. The algorithm starts with a $n \times (n+m)$ matrix consisting of an $n \times n$ identity matrix and the $n \times m$ incidence matrix. The algorithm consists of m steps. At each step one colomn of the incidence matrix is eliminated by performing a set of linear combinations. If the colomn has n^+ positive and n^- negative elements then $n^+ + n^-$ rows are eliminated and $n^+ \times n^-$ rows are created

at that particular step of the algorithm. If the algorithm computes the minimum support invariants then every time a new row is created a test is performed to determine if the new row is covered by a previous one. If it is then the new row is omitted. When the algorithm completes, the $n \times n$ left sub-matrix contains the T-invariants of the net or the minimum support T-invariants of the net.

Algorithm 1 (Martinez-Silva): Let **C** be the incidence matrix of a Petri net and $\mathbf{I}_n$ be the identity matrix.

1. $\mathbf{A} := \mathbf{C}$; $\mathbf{D} := \mathbf{I}_n$ (n is the number of transitions);
2. Repeat for $i = 1$ until $i = m$ (m is the number of places);
   - 2.1 Append to the matrix $[\mathbf{D}|\mathbf{A}]$ every row resulting as a non-negative linear combination of row pairs from $[\mathbf{D}|\mathbf{A}]$ that annul the i^{th} column of **A**.
   - 2.2 Eliminate from $[\mathbf{D}|\mathbf{A}]$ the rows in which the i^{th} column of **A** is non null.

The algorithm is applied to the net in Example 1 as shown in Figure 2. For instance, column A of the incidence matrix has terms 1 and -1. Linear combinations of rows t_1 and t_3 and of rows t_3 and t_5 annul the elements in column A of the incidence matrix. The two new rows are added to the matrix and rows t_1, t_3 and t_5 are removed.

$$
\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \end{array}
\begin{array}{c} \begin{array}{ccccccc} & & & & & & \end{array} \\
\left(\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right.
\begin{array}{c} : \\ : \\ : \\ : \\ : \\ : \\ : \end{array}
\left.\begin{array}{cccc} A & B & C & D \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & -1 \end{array}\right) \end{array}
\Rightarrow
\begin{array}{c} \\ t_1+t_3 \\ t_3+t_5 \\ t_2 \\ t_4 \\ t_6 \\ t_7 \end{array}
\begin{array}{c} \begin{array}{ccccccc} & & & & & & \end{array} \\
\left(\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right.
\begin{array}{c} : \\ : \\ : \\ : \\ : \\ : \end{array}
\left.\begin{array}{ccc} B & C & D \\ -1 & 1 & 0 \\ -1 & 1 & -1 \\ 1 & 0 & 0 \\ -1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & -1 & -1 \end{array}\right) \end{array}
$$

$$
\Rightarrow
\begin{array}{c} \\ t_1+t_2+t_3 \\ t_2+t_3+t_5 \\ t_2+t_4 \\ t_6 \\ t_7 \end{array}
\begin{array}{c} \begin{array}{ccccccc} & & & & & & \end{array} \\
\left(\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right.
\begin{array}{c} : \\ : \\ : \\ : \\ : \end{array}
\left.\begin{array}{cc} C & D \\ 1 & 0 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{array}\right) \end{array}
$$

$$
\Rightarrow
\begin{array}{c} \\ t_1+2t_2+t_3+t_4 \\ 2t_2+t_3+t_4+t_5 \\ t_2+t_4+t_6 \\ t_1+t_2+t_3+t_7 \\ t_2+t_3+t_5+t_7 \\ t_6+t_7 \end{array}
\begin{array}{c} \begin{array}{ccccccc} & & & & & & \end{array} \\
\left(\begin{array}{ccccccc} 1 & 2 & 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right.
\begin{array}{c} : \\ : \\ : \\ : \\ : \\ : \end{array}
\left.\begin{array}{c} D \\ 1 \\ 0 \\ 0 \\ -1 \\ -2 \\ -2 \end{array}\right) \end{array}
$$

$$
\Rightarrow \quad
\begin{array}{r}
2t_2 + t_3 + t_4 + t_5 \\
t_2 + t_4 + t_6 \\
2t_1 + 3t_2 + 2t_3 + t_7 \\
2t_1 + 5t_2 + 3t_3 + 2t_4 + t_5 + t_7 \\
2t_1 + 4t_2 + 3t_3 + 2t_4 + t_6 + t_7
\end{array}
\begin{bmatrix}
0 & 2 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 \\
2 & 3 & 2 & 1 & 0 & 0 & 1 \\
2 & 4 & 2 & 2 & 0 & 1 & 1 \\
2 & 5 & 3 & 2 & 1 & 0 & 1
\end{bmatrix}
\begin{array}{l}
X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5
\end{array}
$$

The set of clauses (1) to (6) implies $D \wedge C$ because the T-invariants X_3, X_4 and X_5 contain the goal transition t_7.

Consider the interpretation of resolution in terms of the algorithm presented above. The process of row combination, row removal and row addition corresponds directly to the process of resolution. A non-negative linear combination of row pairs to cancel the element of the i^{th} column is in effect the cancellation rule of resolution. For example, initially the t_1 row is A and t_3 row is $\neg A \vee \neg B \vee \neg C$. The result of a non-negative linear combination of $t_1 + t_3$ is the same as the one obtained by the resolution rule. For instance the clause $\neg B \vee C$ is derived from two separate disjunctive clauses A and $\neg A \vee \neg B \vee C$.

The mapping from a set of Horn clauses to a Petri net proposed in [PeM 89] is based upon a one to one mapping from the set of clauses to the set of transition vectors (rows of the incidence matrix). A refinement of this method, namely to transform a set of inference rules into a set L' such that $|L'| < |L|, L'$ has fewer inference rules is proposed in this paper. The method presented reduces the effort to compute the T-invariants and the resulting algorithm is susceptible to parallelization.

Inference rules to reduce the effort to compute T-invariants are discussed next. A strategy to reduce the size of the incidence matrix is presented in [MuM 88]. This method is refined here and applied to the previous algorithm for computing T-invariants using the following three rules [DaP 60].

Let J be a set of Horn clauses in propositional logic.

1. *One-literal rule.* This rule applies when the set J of clauses contains a single literal say L. J' is obtained by deleting the clauses of J containing L. J is inconsistent iff J' is inconsistent. In example 1, t_1 is a unit clause and row t_5 can be deleted before starting the procedure for determining the T-invariants, because t_1 is a unit clause containing literal A and t_5 also contains A.

2. *Pure-literal rule.* A literal L is called pure if the literal $\neg L$ does not appear in J. If a literal L is pure in J, then J' is obtained by deleting all clauses containing L. If J' is empty, J is consistent. J is inconsistent iff J' is inconsistent. For example the column P_2 of the following matrix contains two 1's in rows t_1 and t_2. The pure-literal rule can be applied to get matrix C' by deleting rows t_1 and t_2 and column P_2.

$$
\mathbf{C} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array}
\begin{array}{c}
\begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\
\begin{pmatrix}
0 & 1 & -1 & 0 \\
-1 & 1 & 1 & 0 \\
1 & 0 & -1 & 1 \\
0 & 0 & 1 & -1 \\
0 & 0 & 1 & 1
\end{pmatrix}
\end{array}
\quad \Rightarrow \quad
\mathbf{C}' = \begin{array}{c} \\ t_3 \\ t_4 \\ t_5 \end{array}
\begin{array}{c}
\begin{array}{ccc} P_1 & P_3 & P_4 \end{array} \\
\begin{pmatrix}
1 & -1 & 1 \\
0 & 1 & -1 \\
0 & 1 & 1
\end{pmatrix}
\end{array}
$$

From Algorithm 1 it is known that the T-invariants, including support $X(t_1)$ or support $X(t_2)$ cannot be obtained. If rows t_1 and t_2 and column P_2 are deleted before computing the T-invariants, the computational effort is reduced.

3. *Splitting rule.* This rule can be used when neither the one-literal rule nor the pure-literal rule apply. J can be written as

$$(A_1 \vee \neg L) \wedge ... \wedge (A_m \vee \neg L) \wedge (B_1 \vee L) \wedge ... \wedge (B_n \vee L) \wedge G$$

where A_i and B_i are literals or disjunctions of literals, G is a set of clauses, and L is a literal. It is required that A_i, B_i and G contain neither L nor $\neg L$. Obtain $J_1 = A_1 \wedge ... \wedge A_m \wedge G$ and $J_2 = B_1 \wedge ... \wedge B_n \wedge G$. J is inconsistent iff both J_1 and J_2 are inconsistent. In Example 1 after applying the one -literal rule, the matrix $\mathbf{C_1}$ is obtained from matrix C.

$$\mathbf{C_1} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_6 \\ t_7 \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ 0 & -1 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & -1 \end{pmatrix} \end{array} \Rightarrow$$

$$\mathbf{C_2} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_6 \end{array} \begin{array}{c} \begin{array}{ccc} A & B & D \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \end{array} \Rightarrow \mathbf{C_2'} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cc} A & B \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{pmatrix} \end{array}$$

$$\mathbf{C_3} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_4 \\ t_7 \end{array} \begin{array}{c} \begin{array}{ccc} A & B & D \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{pmatrix} \end{array} \Rightarrow \mathbf{C_3'} = \begin{array}{c} \\ t_2 \\ t_4 \\ t_7 \end{array} \begin{array}{c} \begin{array}{cc} B & D \end{array} \\ \begin{pmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{pmatrix} \end{array}$$

The choice to delete column C and split the remaining matrix into two matrices $\mathbf{C_2}$ and $\mathbf{C_3}$ such that $\mathbf{C_2}$ consists of rows having 1 or 0 in column C, $\mathbf{C_3}$ consists of rows having -1 or 0 in column C, was made.

The one-literal rule and the pure-literal rule are applied to the matrices $\mathbf{C_2}$ and $\mathbf{C_3}$ to obtain the matrices $\mathbf{C_2'}$ and $\mathbf{C_3'}$. After applying Algorithm 1, two T-invariants X_1 and X_2 are obtained separately from $\mathbf{C_2'}$ and $\mathbf{C_3'}$.

$$\begin{array}{ccccccc} & & t_1 & t_2 & t_3 & & \\ X_1 & = & <1 & 1 & 1> & X_2 & = \end{array} \begin{array}{ccc} t_2 & t_4 & t_7 \\ <1 & 1 & 1> \end{array}$$

The T-invariants for the incidence matrix C are obtained from the T-invariants for the matrices $\mathbf{C_2}$ and $\mathbf{C_3}$ following the sequence of steps described below. Let A be a place and let $\| X_1(i) \|$, and $\| X_2(i) \|$ be the two subsets of transitions in the preset of A, ${}^\circ A$, and in the postset of A, A°, respectively. Let $a = \sum_{(i)} X_1(i)$ for all $\| X_1(i) \| \subseteq {}^\circ A$ and let $b = \sum_{(i)} X_2(i)$ for all $\| X_2(i) \| \subseteq A^\circ$. Let d be the least common multiple of a and b. X', a T-invariant of the initial matrix C is given by $X' = d/a * X_1 + d/b * X_2$

As an example consider the place C in Figure 2. ${}^\circ C = \{t_3, t_6\}$ and $C^\circ = \{t_4, t_7\}$. In this case $a = \sum X_1(i) = 1$, $b = \sum X_2(i) = 2$ and $d = 2$. Call X_1' and X_2' the extended versions of X_1 and X_2.

$$\begin{array}{ccccccccccccccc} & & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & & & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ X_1' & = & <1 & 1 & 1 & 0 & 0 & 0 & 0> & X_2' & = & <0 & 1 & 0 & 1 & 0 & 0 & 1> \end{array}$$

Then

$$X' = d/a * X'_1 + d/b * X'_2 \quad = \quad \begin{array}{ccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \\ <2 & 3 & 2 & 1 & 0 & 0 & 1> \end{array}$$

The T-invariant X' is identical with X_3 obtained from Algorithm 1. By reducing the size of the matrices, the splitting rule reduces not only the computational effort, but also the memory requirements for computing the T-invariants of a set of Horn clauses. Note also that the algorithm which exploits the splitting rule is susceptible to parallelization, the invariants for C_2 and C_3 can be computed in parallel [MaB 91].

Algorithm 2 in now introduced. The algorithm takes advantage of the three rules discussed above. It uses two new procedures called Procedure 2 and Procedure 3 to reduce the size of the incidence matrix based upon the one-literal rule and the pure-literal rule respectively and calls Procedure 1 which implements the original algorithm for computing place or transition invariants (Algorithm 1).

Procedure 2: given an $n \times m$ incidence matrix **C** as input, a $n \times m$ output matrix **C'** is produced.

procedure2 (C,C')

1. Repeat for $k := 1$ until $k = n$. If row k of matrix **C** consists of 0's and a unique non-zero entry 1 (or -1), say in the column P_i, then mark all rows whose value in column P_i is 1 (or -1) except row k itself.

2. Delete all rows marked and update as follows $n := n$ - the number of rows marked.

endprocedure2

Procedure 3: given an $n \times m$ incidence matrix **C** as input, an output matrix **C'** with a reduced number of rows and columns is produced.

procedure3(C,C')

1. $i := m$

2. Repeat for $k := 1$ until $k = i$. If column k consists of 1's and 0's or -1's and 0's, then
   begin

   delete all rows having a non-zero entry in column k and set
   $n := n$ - the number of rows deleted;
   delete the column k and set $m := m - 1$;

   end

endprocedure3

Algorithm 2: given an $n \times m$ incidence matrix **A** as input, the T-invariants are produced

1. call procedure2(**A**,**C**);
2. call procedure3(**C**,**C**′);
3. choose column P_i of **C**′ with a minimum number of 0's;
4. delete column P_i and split **C**′ into $\mathbf{C_1}$ and $\mathbf{C_2}$ so that $\mathbf{C_1}$ consists of rows having 1 or 0 in the column P_i, and $\mathbf{C_2}$ consists of rows having -1 or 0 in the column P_i;
5. call procedure2($\mathbf{C_1}$,$\mathbf{C_1}'$);
6. call procedure3($\mathbf{C_1}'$,$\mathbf{C_1}''$);
7. call procedure 1 to compute $Y_j, 1 \leq j \leq a$, the T-invariants of $\mathbf{C_1}''$;
8. call procedure2($\mathbf{C_2}$,$\mathbf{C_2}'$);
9. call procedure3($\mathbf{C_2}'$,$\mathbf{C_2}''$);
10. call procedure 1 to compute $X_k, 1 \leq k \leq b$, the T-invariants of $\mathbf{C_2}''$;
11. combine T-invariants of $\mathbf{C_1}''$ and $\mathbf{C_2}''$); for $1 \leq j \leq a$, $1 \leq k \leq b$. If $\exists Y_j(1) \neq 0 \wedge \|Y_j(1)\| \subseteq {}^\circ P_i$ and $\exists X_k(h) \neq 0 \wedge \|X_k(h)\| \subseteq P_i^\circ$, then
    begin for $\forall \|Y_j(1)\| \in {}^\circ P_i \;\; \sum Y_j(1) = d$;

        for $\forall \|X_k(h)\| \in P_i \;\; \sum X_k(h) = q$;
        Obtain the least common multiple of d and q, say $e = lcm(d,g)$;
        $X = \frac{e}{d} Y_j + \frac{e}{q} X_k$;
        delete Y_j and X_k;

    end

For the sake of clarity the input and the output of different algorithms and procedures presented above are explicit. In an actual program the output may overwrite the input data structures.

Every T-invariant of a Petri net can be expressed as a linear combination of minimum support T-invariants with positive coefficients [MaS 82]. Only minimal support T-invariants which include the goal transition in their support are of interest in this paper. For instance, in Example 1, only the T-invariant X_3 is needed. X_1 and X_2 are minimal support T-invariants but do not contain the goal transition t_7 in their support. X_4 and X_5 include the goal transition in their support but they are non-minimal support invariants.

4 High Level Petri Net Models of First Order Predicate Logic

The following definition of High Level Petri Nets, (HLPNs) is based upon [Jen 86] and [Gen 86].

Definition 4.1. A HLPN is an 8-tuple $H = (S, T; F, A, V, X, W, M_0)$ where:

- $N = (S, T; F)$ is the underlying net of H.
- A is a finite set of atomic colors. A^k denotes the set of all k-tuples $< a_1, \ldots, a_k >$, with $a_i \in A$.
- V is a finite set of variable over A. V^k denotes the set of all k-tuples $< v_1, \ldots, v_k >$, with $v_i \in V$.
- X: $S \rightarrow \cup_{0 \leq k \leq n} A^k$ with $A^\circ = \{<>\}$ and $T \rightarrow \cup_{0 \leq k \leq n} (V \leftarrow A)^k$ is called a color function. n is a given maximal arity of predicates. $X(S)$ represents the set of predicates. X attaches to each place a set of possible token colors. $X(T)$ represents the set of transition colors. X attaches to each transition a set of possible occurrence colors, i.e., $\forall t \in T$, $X(t)$ is the set of substitutions of all variables appearing free in the arc labels connected t.
- W: $F \rightarrow [\cup_{0 \leq k \leq n} (A \cup V)^k \rightarrow N]$ is an arc label function. It indexes a family of multi-sets over $\cup_{0 \leq k \leq n} (A \cup V)^k$, i.e., $\forall (x, y) \in F$, $W_{x,y}$: $[(A \cup V)^k \rightarrow N]$.
- M_0 is initial marking of H. Marking M is an S-indexed family of multi-sets over $X(S)$: $\forall s \in S$, $M(s)$: $X(s) \rightarrow Z$.

In a HLPN, each place is a predicate which describes a relation among individuals, i.e., $\forall s \in S$, s has a certain subset of A^k. Each $< a_1, \ldots, a_k > \in A^k$ has a value of either true or false. An arc label specifies a variable extension of a predicate to which the arc is connected. A transition defines a logical implication among its input and its output predicates. When the input predicates are satisfied, the output predicate yields a prescribed conclusion.

Let σ be a transition color and $W_{x,y}(\sigma)$ be the multi-set obtained from $W_{x,y}$ by substituting the free variables by atomic colors according to σ. For instance, if $\sigma = (u \leftarrow b, v \leftarrow a)$ and $W_{x,y} =< a, u > + < u, v >$, then $W_{x,y}(\sigma) =< a, b > + < b, a >$.

Definition 4.2. The incidence matrix of a HLPN is the matrix $\mathbf{C} = (C_{t,s})$ for all $t \in T$, $s \in S$ with $C_{t,s}$ defined as

$$C_{t,s} = W_{t,s} - W_{s,t}.$$

Thus $C_{t,s} \in [A \cup V)^k \rightarrow Z]$.

Definition 4.3. The transition t is enabled at marking M iff $\exists \sigma \in X(t)$, such that $\forall s \in^\circ t$, $W_{s,t}(\sigma) \leq M(s)$.

We will say that a step (t, σ), rather than transition t, is enabled if a transition color function σ exists, i.e., $\exists \sigma \in X(t)$. The step (t, σ) is not enabled if a colors function σ does not exist.

Definition 4.4. When a step (t, σ) is enabled at M, it can fire and transform M into a directly reachable marking M' defined for $\forall s \in S$ by

$$M'(s) = M(s) - W_{s,t}(\sigma) + W_{t,s}(\sigma).$$

Or

$$M'(s) = M(s) + C^t(\sigma).$$

Any HLPN with a finite set of colors, admits an equivalent Place/Transition net obtained through unfolding each place s into the set of places $\{(s,a)|a \in X(s)\}$ and unfolding each transition t into the set of transitions $\{(t,\sigma)|\sigma \subset X(t)\}$.

A Horn clause is defined as

$$B \leftarrow A_1 \wedge A_2, \ldots, \wedge A_n \tag{4.1}$$

A first order predicate logic atomic formula is $f(a_1, \ldots, a_k)$ where f is a k-element predicate symbol and $a_1, \ldots, a_k$ are terms. A term is defined inductively as follows: a variable is a term, a constraint is a term, if f is an n-ary function symbol and $t_1, t_2, \ldots, t_n$ are terms, then $f(t_1, t_2, \ldots, t_n)$ is a term.

If there are variables $x_1, \ldots x_j$ in (4.1), it is to be interpreted as "for all $x_1, \ldots, x_j$, if A_1 and ..., and A_n then B". The transformation procedure from the first order predicate logic to the incidence matrix of HLPN is almost the same as the procedure in Section 2. The only difference is that an entry of the matrix $C_{i,j}$ is the formal sum of arguments in the j^{th} clause and in the j^{th} predicate symbol where the sum is taken over, all the j^{th} predicate symbols appearing in the i^{th} clause.

Example 2: Consider the following set of Horn clauses from [PeM 89].

C1: ancestor (x,y): $\leftarrow$ parent (x,y)

C2: ancestor (x,y): $\leftarrow$ parent $(x,z) \wedge$ ancestor (z,y)

C3: parent (D,J): $\leftarrow$

C4: parent (J,M): $\leftarrow$

If a query (or a goal statement) to the set of clauses is as follows:

C5: $\leftarrow$ ancestor (x,M)

then the HLPM net model and the incidence matrix for the set of clauses and the query are obtained through the transformation procedure shown in Figure 3.

$$C = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \text{ancestor(a)} \\ \begin{pmatrix} < x,y > \\ < x,y > - < z,y > \\ \phi \\ \phi \\ - < x,M > \end{pmatrix} \end{array} \begin{array}{c} \text{parent(P)} \\ \begin{pmatrix} - < x,y \\ - < x,z > \\ < D,J > \\ < J,M > \\ \phi \end{pmatrix} \end{array}$$

A set of clauses can be represented as a HLPN. The main problem is that a pair of unifiable literals is not explicitly represented in the set of clauses when the resolution rule is applied. So we cannot explicitly obtain the colors function X.

5 Algorithms for Logical Inference in a HLPN Model of First Order Predicate Logic

To discuss inference in first order logic, the concept of unification is needed. In order to represent unifiable relations explicitly, a unifiable relation set $U(p)$ for each place p is introduced. A member of $U(p)$, $u(\theta) = (W_{t_i,p}, W_{p,t_j}, \theta)$ represents a unifiable pair $(W_{t_i,p}, W_{p,t_j})$ with a mgu (most general unifier) θ where $W_{t_i,p}$ is the label on the input arc to p and W_{p,t_j} is

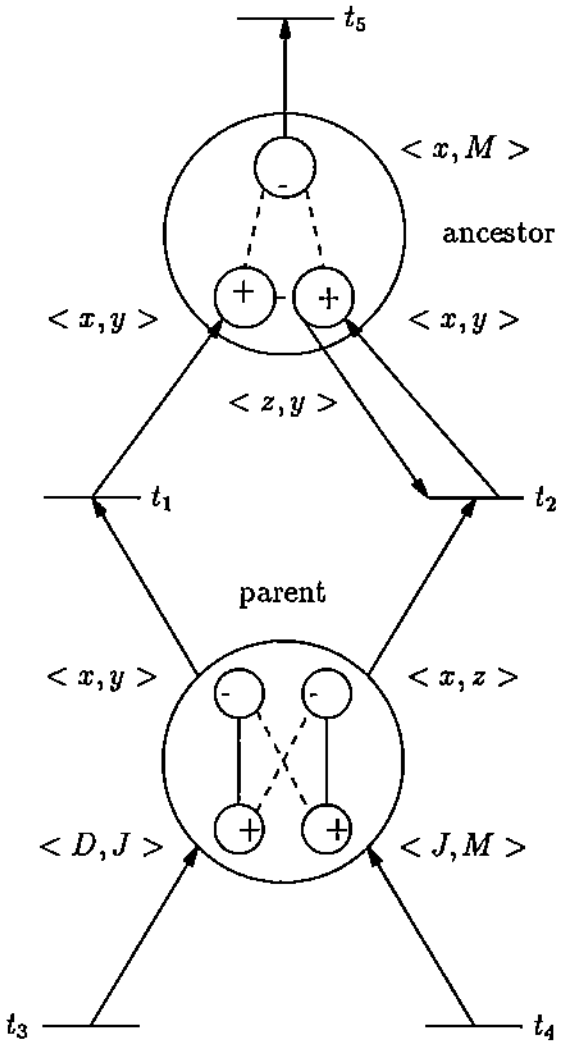

Figure 3. The High Level Petri Net for Example 3

the label on the output arc from p to the transition t_j. In special cases, the label is a formal sum of multi-atomic formulas. Multiple labels are needed in a unifiable pair. In Example 2, $W_{t_2,a} = < x,y > - < z,y >$. In this example, $W_{t_1,a}$, $W_{t_5,a}$ and $W_{t_2,a}$ can be in one unifiable pair, $W_{t_1,a}$ and $W_{t_3,a}$ are in another unifiable pair. To facilitate the resolution rule (non-negative linear combination of row pairs), unification can only be carried out between a pair of complementary signed occurrences. In Figure 3, dashed lines among the occurrences indicate all the unification possibilities. In fact, the unifiable pairs show the possible paths of the token flow.

The following unifiable relation sets for Example 2 can be obtained.

$$\begin{aligned} U \text{ (parent)} &= \{\theta_1 = (< D,J >, < x,y >, (D/x, J/y)), \\ &\quad\ \theta_2 = (< D,J >, < x,z >, (D/x, J/z)), \\ &\quad\ \theta_3 = (< J,M >, < x,y >, (J/x, M/y)), \\ &\quad\ \theta_4 = (< J,M >, < x,z >, (J/x, M/z))\} \\ U \text{ (ancestor)} &= \{\theta_5 = (< x,y >, < x,M >, (M/y)) \\ \theta_6 &= ((< x,y >, < x,M >), (< x,y >, < z,y >)(M/y, z/x))\} \end{aligned}$$

An algorithm for computing the T-invariants of HLPNs is now discussed. Each element in a T-invariant vector of a HLPN generally consists of two components: one is the coefficient which denotes the number of firing in the firing sequence; another is the substitution which is related to the transition colors.

The algorithm for S-invariants of HLPNs [LiM 91] is updated and the following proposition for T-invariants of HLPNs is derived.

Proposition 1. Let $T' = (Z_{t,\theta} | t \in T, \theta$ is a substitution in the unifications) be a solution of

$$\forall s \in S,\ \forall u(\theta) \in U(s) \text{ then } \sum_{t \in T} Z_{t,\theta}(\theta) * C_{t,s} = 0$$

The corresponding T-invariant is an n vector T

$$T = [Z_{t_1}(\beta_1), \ldots, Z_{t_n}(\beta_n)]$$

where β_i is a constant '1' or a substitution or a composition of multiple substitutions according to the connection between $Z_{t,\theta}$ and θ.

Intuitive Explanation of Proposition 1. The n vector T satisfies $\mathbf{C}^T T = 0$ because each possible unification of any column in the incidence matrix $\mathbf{C}$ is zero.

Applying Proposition 1 to Example 2, the following equation groups are obtained:

$$\begin{cases} Z_{t_4,\theta_3} * < J,M > + Z_{t_1,\theta_3} * (- < x,y >: (J/x, M/y)) = 0 \\ Z_{t_1,\theta_6} * (< x,y >: (M/y)) + Z_{t_5,\theta_6} * (- < x,M >) = 0 \end{cases} \tag{1}$$

$$\begin{cases} Z_{t_4,\theta_3} * < J,M > + Z_{t_1,\theta_3} * (- < x,y >: (J/x, M/y)) = 0 \\ Z_{t_1,\theta_7} * (< x,y >: (z/x, M/y)) + Z_{t_5,\theta_7}(- < x,M >) \\ \quad + Z_{t_2,\theta_7} * ((< x,y > - < z,y >) : (M/y)) = 0 \\ \\ Z_{t_2,\theta_2} * (- < x,y >: (D/x, J/z)) + Z_{t_3,\theta_2} * < D,J >= 0 \end{cases} \tag{2}$$

The two equation groups can be written as

$$\begin{cases} Z_{t_4,\theta_3} - Z_{t_1,\theta_3} = 0 \\ Z_{t_1,\theta_6} - Z_{t_5,\theta_6} = 0 \end{cases} \quad (1')$$

$$\begin{cases} Z_{t_4,\theta_3} - Z_{t_1,\theta_3} = 0 \\ (Z_{t_1,\theta_7} - Z_{t_2,\theta_7}) - (Z_{t_5,\theta_7} + Z_{t_2,\theta_7}) = 0 \\ -Z_{t_2,\theta_2} + Z_{t_3,\theta_2} = 0 \end{cases} \quad (2')$$

Solving (1'),

Z_{t_1}	Z_{t_2}	Z_{t_3}	Z_{t_4}	Z_{t_5}
1	0	0	1	1

From Z_{t_1,θ_3} and Z_{t_1,θ_6}, it follows that $\beta_1 = \theta_3\theta_6 = (J/x, M/y)$. The term Z_{t_4} does not need a substitution. β_5 is identical with β_1, i.e., $\beta_5 = (J/x)$ since Z_5 and Z_{t_1} appear in one equation.

Finally, a T-invariant T_1 is obtained

$$T_1 = \begin{array}{ccccc} Z_{t_1} & Z_{t_2} & Z_{t_3} & T_{t_4} & Z_{t_5} \\ 1 & 0 & 0 & 1 & 1 \\ (J/x, M/y) & & & & (J/x) \end{array}$$

Solving (2'), the T-invariant T_2 is obtained

$$T_2 = \begin{array}{ccccc} Z_{t_1} & Z_{t_2} & Z_{t_3} & T_{t_4} & Z_{t_5} \\ 1 & 1 & 1 & 1 & 1 \\ (J/x, M/y) & (D/x, J/z, M/y) & & & (D/x) \end{array}$$

Algorithm 1 can be extended to have a uniform way to compute T-invariants for various classes of Petri net models. The "non-negative combination of row pairs" in Step (2.1) of Algorithm 1 becomes "non-negative combination of unifiable row pairs in the i^{th} column". In addition, the unifier is attached to each row of the row pairs.

Algorithm 3: Let **C** be the incidence matrix of a HLPN and $\mathbf{I}_n$ be the identity matrix.

1. $\mathbf{A} := \mathbf{C}$; $\mathbf{D} := \mathbf{I}_n$ (n is the number of transitions)
2. Repeat for $i = 1$ unti $i = m$ (m is the number of places)
   - 2.1 Append to the matrix $[\mathbf{D}|\mathbf{A}]$ every row which resulted from a non-negative linear combination of unifiable row pairs in the i^{th} column from $[\mathbf{D}|\mathbf{A}]$ that annul the i^{th} column of **A** and attach the corresponding unifier to each row of the pair.
   - 2.2 Eliminate the rows in which the i^{th} column of **A** is non null from $[\mathbf{D}|\mathbf{A}]$.

The T-invariants for Example 2 are now computed using Algorithm 3.

$$\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{ccccc} & & & & \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{array}{c:} \\ \\ \\ \\ \\ \end{array} \begin{array}{cc} A & P \\ < x, y > & - < x, y > \\ < x, y > - < z, y > & - < x, z > \\ 0 & < D, J > \\ 0 & < J, M > \\ - < x, M > & 0 \end{array}$$

$$\Downarrow$$

$$\begin{array}{r} \\ t_1(J/x,M/y)+t_4 \\ t_2(J/x,M/z)+t_4 \\ t_1(D/x,J/y)+t_3 \\ t_2(D/x,J/z)+t_3 \\ t_5 \end{array} \begin{pmatrix} (J/x,M/y) & 0 & 0 & 1 & 0 & : & A \\ & & & & & & < J,M > \\ 0 & (J/x,M/z) & 0 & 1 & 0 & : & < J,y > - < M,y > \\ (D/x,J/y) & 0 & 1 & 0 & 0 & : & < D,J > \\ 0 & (D/x,J/z) & 1 & 0 & 0 & : & < D,y > - < J,y > \\ 0 & 0 & 0 & 0 & 1 & : & - < x,M > \end{pmatrix}$$

$$\Downarrow$$

$$\begin{array}{r} \\ t_1(J/x,M/y)+t_4+t_5(J/x) \\ t_1(J/x,M/y)+t_4+t_2(D/x,J/z)(M/y) \\ +t_3+t_5(D/x) \\ t_1(D/x,J/y)+t_3 \\ t_2(J/x,M/y)+t_4 \end{array} \begin{pmatrix} (J/x,M/y) & 0 & 0 & 1 & (J/x) & : & A \\ & & & & & & 0 \\ (J/x,M/y) & (D/x,J/z)(M/y) & 1 & 1 & (D/x) & : & 0 \\ & & & & & & \\ (D/x,J/y) & 0 & 1 & 0 & 0 & : & < D,J > \\ 0 & (J/x,M/z) & 0 & 1 & 0 & : & < J,y > - < M,y > \end{pmatrix}$$

Two T-invariants are obtained:

$$T_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} (J/x,M/y) \\ \\ \\ \\ (J/x) \end{matrix} \qquad T_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} (J/x,M/y) \\ (D/x,J/z,M/y) \\ \\ \\ (D/x) \end{matrix}$$

Two T-invariants finally yield two solutions for the querry: "J is an ancestor of M" and "D is an ancestor of M".

The unifier is identified when the non-negative linear combination is computed. The unifier leads to the variants being replaced uniformly by constants.

In computing a HLPN model of a first order predicate logic inference rules can be used to reduce the computations required by the algorithm. The basic idea is to delete rows which are not unifiable. Due to the limited space, this is not described in detail.

6 Conclusions

This paper studies Petri net models for the Horn clause form of propositional logic and first order predicate logic. The paper proposes an algorithm for finding the T-invariants of a Petri net model of a Horn clause system. This algorithm is based on the idea of resolution and exploits the presence of one-literal, pure-literal and splitting clauses to lead to faster computation. The algorithm is then extended for computing T-invariants of High Level Petri net models of predicate logic.

This paper does not provide a quantitative analysis of the new algorithms. The computational complexity of Petri net analysis algorithms is a neglected area of research. Only qualitative arguments supporting the advantages of the algorithms proposed in this paper can be given. Reducing the size of the incidence matrix always reduces the storage requirements and *may* reduce the computations required by structural analysis.

Another neglected area of research in Petri nets is the investigation of parallel algorithms and methods in net analysis. The size of the Petri net models of realistic applications makes

the sequential analysis algorithms impractical in terms of storage and computing time. A parallel algorithm to compute place and transition invariants and its implementation on a distributed memory multiprocessor system is discussed in [MaB 91].

As a future research agenda at the top of our list is extending the applications of Petri net methods, such as computation of T-invariants to the domain of non-monotonic logic systems. Non-monotonic logic systems are coming to play an increasingly important role in artificial intelligence research. We are exploring the idea of using a Petri net formalism with inhibitor arcs and multi-valued logic as a means of representation of such systems.

Acknowledgements

The authors acknowledge the constructive comments and suggestions of the anonymous reviewers. They have a substantial contribution to improve the quality of the paper.

7 References

[DaP 60] M. Davis and H. Putnum, "A Computing Procedure for Quantification Theory", JACM, Vol. 7, No. 3, 1980, pp. 201–215.

[Gen 86] H.J. Genrich, "Predicate/Transition Nets", Lecture Notes in Computer Science, Vol. 254, 1986, pp. 207–247.

[GEN 79] H.J. Genrich, et al., "Elements of General Net Theory", Lecture Notes in Computer Science, Vol. 85, 1979, pp. 21–164.

[Gio 85] A. Giordana, et al., "Modeling Production Rules by Means of Predicate Transition Network", Information Sciences, 35, 1985, pp. 1–41.

[GuG 89] C. Gunter and V. Gehlot, "Nets as Tensor Theories", Proceedings of the 10^{th} International Conference on Applications and Theory of Petri Nets, June 1989.

[Hoo 88] J.N. Hooker, "A Quantitative Approach to Logical Inference", North-Holland, Decision Support Systems, Vol. 4, 1988, pp. 45–69.

[JeW 87] R.G. Jeroslow and J. Wang, "Solving Propositional Satisfiability Problems", Working Paper, Georgia Institute of Technology, Atlanta 1987.

[Jen 86] K. Jensen, "Coloured Petri Nets", Lecture Notes in Computer Science, Vol. 254, 1986, pp. 248–299.

[Kow 79] R. Kowalski, "Logic for Problem Solving", New York: Elsevier Science, 1979.

[Lau 85] K. Lautenbach, "On Logical and Linear Dependencies", St. Augustin, Germany, GMD Rep. 147, 1985.

[LiM 91] C. Lin and D.C. Marinescu, "On the Analysis of Stochastic High Level Petri Nets", International Journal of Microelectronics and Reliability, 1991 (to appear). Also CSD-TR 860, February 1989.

[MaB 91] D.C. Marinescu, M. Beaven, B. Elmore and R. Stansifer, "A Parallel Algorithm for Computing Place/Transition Invariants of a Petri Net on a Distributed Memory Multiprocessor System", CSD-TR 91-024, Computer Science, March 1991.

[MaS 82] J. Martinez and M. Silva, "A Simple and Fast Algorithm to Obtain all Invariants of Generalized Petri Nets", Informatik-Fachbrichte 52 (C. Girrault and W. Reisig, eds), Springer-Verlag, 1982, pp. 301–303.

[MuM 88] T. Murata and K. Matsuyama, "Inconsistency Check of a Set of Clauses using Petri Net Reductions", The Franklin Institute, 1988.

[MuZ 88] T. Murata and D. Zhang, "A Predicate-Transition Net Model for Parallel Interpretation of Logic Programs", IEEE Trans. on Software Engineering, Vol. 14, No. 4, April 1988, pp. 481–497.

[Nil 80] N.J. Nilsson, "Principles of Artificial Intelligence", Tioga Publishing Co., 1980.

[Pea 86] J. Pearl, "Fusion, Propagation and Structuring in Belief Networks", Artificial Intelligence, 29, 1986, pp. 241–288.

[PeM 89] G. Peterka and T. Murata, "Proof Procedure and Answer Extraction in Petri Net Model of Logic Programs", IEEE Trans. on Software Engineering, Vol. 15, No. 2, February 1989, pp. 209–217.

[Rei 85] W. Reisig, "Petri-Nets – An Introduction", Springer-Verlag, 1985.

[Sin 87] A. Sinachopoulos, "Derivation of a Contradiction by Resolution Using Petri Nets", Petri Net Newsletter, Vol. 26, April 1987, pp. 16–29.