VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MATHEMATICAL MODELING (CO2011)

Assignment

Symbolic and Algebraic Reasoning in Petri Nets

Instructor: Dr. Van-Giang Trinh

Ho Chi Minh City, 12/2025

Members and Workload

STT	Full name	Students ID	Task	Class
1	Lưu Nguyễn Thanh Bình	2310927	Implement the 1st task and 2nd task: *Reading Petri nets from PNML files* and *Explicit computation of reachable markings*	L02
2	Trần Hoàng Bá Huy	2311249	Implement the 3rd task: *Symbolic computation of reachable markings by using BDD*	L01
3	Xà Gia Khánh	2311543	Implement the 4th task: *Deadlock detection by using ILP and BDD*	L04
4	Đoàn Duy Khanh	2311488	Implement the 5th task: *Optimization over reachable markings*	L04
5	Dương Khôi Nguyên	2312333	Conceptualize the structure; writing, editing, formatting the final report and incorporating all co-author feedback and data inputs	L04

Table of Contents

1 Introduction

1.1 Motivation

Petri nets are among the most fundamental and elegant mathematical models for describing concurrent, distributed, and event-driven systems. Since their introduction in the early 1960s by Carl Adam Petri, they have become a cornerstone of formal methods and system verification. They provide a rigorous graphical and formal way to represent the interaction between conditions (places) and events (transitions) through the flow of tokens. This framework is ideal for modeling systems such as manufacturing lines, communication protocols, and complex regulatory networks.

However, the analysis of these systems is often computationally challenging due to the **state space explosion problem**. The concurrency inherent in complex systems can lead to an exponential number of reachable markings (states), making traditional explicit enumeration methods - like Breadth-First Search or Depth-First Search - infeasible for all but small systems.

1.2 The Integration of Symbolic and Algebraic Reasoning

This assignment explores the integration of two powerful computational techniques: symbolic and algebraic reasoning.

- **Symbolic Representation (Binary Decision Diagrams - BDDs):** We leverage BDDs to compactly encode the potentially vast set of reachable markings. By representing the state space symbolically using canonical Boolean functions, BDDs enable efficient memory management and faster formal verification compared to explicit state storage.

- **Algebraic Optimization (Integer Linear Programming - ILP):** We utilize ILP as a flexible, optimization-based framework to reason about non-local properties of the Petri net. Core questions related to structural properties, such as invariant analysis and deadlock detection, can be formulated and solved as systems of linear inequalities based on the net's incidence matrix.

1.3 Project Objectives

The central goal of this assignment is to foster both theoretical insight and hands-on skills in bridging abstract models with algorithmic analysis. We shall build an application integrating these computational ideas to analyze **1-safe Petri nets**. The specific contributions of this report are organized around the following key tasks:

1. **Parsing and Explicit Reachability:** Implementing a parser for the PNML standard to construct the model and explicitly enumerating states for performance baseline.

2. **Symbolic Reachability Analysis:** Using BDDs to symbolically construct the set of reachable markings and comparing its performance against the explicit approach.

3. **Deadlock Detection:** Applying ILP formulations in combination with the BDD representation to detect deadlocks.

4. **Optimization over Reachable Markings:** Solving a final optimization problem, maximizing a linear objective function over the computed set of reachable markings.

1.4 Synthesis

By integrating the structural modeling strength of Petri Nets, the memory efficiency of BDDs, and the analytical power of ILP, this project provides an approach necessary for tackling complex engineering problems in formal verification and optimization.

2 Background Theory

2.1 Petri Net

According to Murata (1989), a Petri net is defined as a tuple $N = (P, T, F, M_0)$, where P is a finite set of places, T is a finite set of transitions, and F represents the flow relation, allowing us to visualize the causal dependencies between events. The state of a Petri net is defined by its **marking** M, which is a vector of non-negative integers of size $|P|$. The dynamic behavior is governed by the firing rule: a transition t is enabled if $M(p) \geq 1$ for all input places $p \in t$. When t fires, the new marking M' is given by the state equation:

$$M' = M + C \cdot u$$

where C is the incidence matrix and u is the firing vector.

This assignment specifically focuses on **1-safe Petri nets**. A net is 1-safe if for every reachable marking $M \in Reach(M_0)$, and for every place $p \in P$, the number of tokens is at most 1:

$$\forall M \in Reach(M_0), \forall p \in P : M(p) \in \{0, 1\}$$

This property allows the global state to be encoded as a Boolean vector, facilitating the use of symbolic logic.

2.2 Binary Decision Diagrams

Binary Decision Diagrams (BDDs) are a canonical directed acyclic graph (DAG) data structure used to represent Boolean functions Bryant (1986). A Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ is represented as a rooted graph where non-terminal nodes are labeled with variables x_i and terminal nodes represent values 0 (False) and 1 (True).

The power of BDDs lies in their **canonical form** (Reduced Ordered BDD - ROBDD). For a fixed variable ordering, any Boolean function has a unique BDD representation. This allows for constant-time equivalence checking and efficient logical operations with complexity proportional to the graph size rather than the exponential state space.

In the context of this assignment, we utilize BDDs to solve the state space explosion problem inherent in concurrent systems.

State Encoding: Since the Petri net is **1-safe** Cheng, Esparza, and Palsberg (1995), the number of tokens in any place p_i is strictly bounded to $\{0, 1\}$. Thus, a global marking M can be encoded as a Boolean vector $(x_1, x_2, \ldots, x_{|P|})$, where $x_i = 1$ if and only if place p_i contains a token. The set of all reachable markings is represented by a characteristic function χ_{Reach}.

Transition Relation: Instead of firing transitions individually, we encode the entire net's behavior into a single Boolean function called the **Transition Relation** $TR(x, x')$. For a transition t, the relation TR_t is defined as:

$$TR_t(x, x') = \left(\bigwedge_{p \in {}^\bullet t} x_p \right) \wedge \left(\bigwedge_{q \in t^\bullet} x'_q \right) \wedge \left(\bigwedge_{r \notin {}^\bullet t \cup t^\bullet} (x'_r \leftrightarrow x_r) \right)$$

The global transition relation is the disjunction of all individual transition relations: $TR(x, x') = \bigvee_{t \in T} TR_t(x, x')$.

Image Computation: We compute the reachability set iteratively using symbolic image computation. The set of next states S_{new} reachable from current states S_{curr} is computed via existential quantification:

$$S_{new}(x') = \exists x.[S_{curr}(x) \wedge TR(x, x')]$$

This process repeats until a fixed point is reached ($S_{new} \subseteq S_{accumulated}$).

2.3 Integer Linear Programming

Integer Linear Programming (ILP) optimizes a linear objective function subject to linear constraints with integer variables (Graver, 1975).

$$\text{Maximize } c^T x \quad \text{subject to } Ax \leq b,\ x \in \mathbb{Z}^n$$

In Petri nets, ILP is fundamentally linked to the state equation $M = M_0 + C \cdot \sigma$, where C is the incidence matrix.

2.3.1 Application: Deadlock Detection (Task 4)

A **deadlock** is formally defined as a reachable marking where no transition is enabled. We employ a hybrid approach combining BDD reachability and ILP logic.

References

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, *35*(8), 677–691.

Cheng, A., Esparza, J., & Palsberg, J. (1995). Complexity results for 1-safe nets. *Theoretical Computer Science*, *147*(1–2), 117–136.

Graver, J. E. (1975). On the foundations of linear and integer linear programming i. *Mathematical Programming*, *9*(1), 207–226.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*(4), 541–580.