Petri Nets and Programming: A Survey

Marian V. Iordache  
School of Engineering and Eng. Tech.  
LeTourneau University  
Longview, TX 75607-7001

Panos J. Antsaklis  
Department of Electrical Engineering  
University of Notre Dame  
Notre Dame, IN 46556

June 12, 2009
Introduction

Programs = Specifications

A formal model representation of programs opens the way to

- verification methods
- synthesis methods

Here we are concerned with Petri net (PN) representations of programs.

We show that:

- Various models used in the context of program synthesis and verification can be converted to PNs.
- Program synthesis is related to supervisory control (SC).
Notation: \(\mu\) – the marking, \(\mu_0\) – the initial marking, \(D\) – the incidence matrix, \(q\) – the firing vector, and \(v\) – the Parikh vector. Let \(\mu_i\) denote \(\mu(p_i)\) and \(v_j\) denote \(v(t_j)\).

The state equation: \(\mu = \mu_0 + D(v - v_0)\).

\[
\begin{align*}
\mu_0 &= \begin{bmatrix} 1 & 0 & 2 & 0 \end{bmatrix}^T \\
v_0 &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \\
q &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\
\mu' &= \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix}^T \\
v' &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\
q' &= \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T \\
\mu'' &= \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^T \\
v'' &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T \\
q'' &= \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T
\end{align*}
\]
Supervisory Control

SC input describes:

- System to be controlled (PN, automata).
- Properties to be satisfied during operation.
- Implementation constraints
  - controllability
  - observability
  - decentralization

SC result: *supervisor*

- Represented by a DES (PN, automata).
- Implemented in computer code.

Example: $\mu_1 + \mu_2 \leq 1$

• **Fairness constraints.** *In [Li 93] for manufacturing application. In [Genrich 80] for a communication protocol.*

Example: $v_1 - v_3 \leq 0$

• **Enabling constraints.** *In [Yamalidou 91], for chem. process control. In [Giua and Seatzu 01], for railway networks.*

Example: $q_1 \leq \mu_2$
Supervisory Control

\[ \mu_1 + \mu_2 \leq 1 \]

\[ v_1 - v_3 \leq 0 \]

\[ q_1 \leq \mu_2 \]
Supervisory Control

Monitors implement specs $L\mu + Hq + Cv \leq b$ [Iordache and Antsaklis 2003].

Not all interesting specs have this form.

Example: In a single track segment all trains must go in the same direction.

Thus, $[\mu_1 \leq 0] \lor [\mu_2 \leq 0]$.

Example: Readers/writers problem.

Disjunctive constraints: $\bigvee_i L_i \mu + H_i q + C_i v \leq b_i$.

Under certain boundedness assumptions they can be implemented by a PN supervisor.
Supervisory Control

Specifications implemented by PN supervisors: prefix type PN languages.

PLANT

SPECIFICATION
Supervisory Control

Petri Nets and Programming: A Survey
Application

- Extracting PN Models
  - Esterel and CFSMs
  - Condition systems
  - PEP
  - Representing Control Flow

- PN Based Design
Esterel is a programming language for reactive systems [Halbwachs, 1993].

Programs written in Esterel can be represented by FSMs.

Note that FSMs are PNs.

Codesign Finite State Machines (CFSMs): a FSM based model.

Polis: Translates Esterel specifications to Codesign Finite State Machines (CFSMs).

CFSM references include [Chiodo et al, 1993], [Balarin et al, 1997], ...

Note that CFSM networks can be converted to safe PNs.
In CFSM networks, components interact asynchronously.

For instance, the transition $\eta/\beta$ may take place if $\eta$ is present.

When $\eta/\beta$ takes place, the event $\beta$ is generated.

$\eta/\beta$ takes place after $\eta$ is generated, not at the same time.

When $\eta/\beta$ takes place, $\eta$ is consumed.
CFSM networks can be converted to PNs.

For consistency with CFSM networks, $p_\beta$ and $p_\gamma$ should have no more than one token.
$p_\beta$ marked when $\beta$ present; $p_{\overline{\beta}}$ marked otherwise. $p_\gamma$ and $p_{\overline{\gamma}}$ used in the same manner.
• Condition systems: high level PNs in which places output conditions and transitions are labeled with conditions.

• Applied to the synthesis and specification of control software in Spectool.

• References include [Holloway, 2000], [Ashley, 2007], ...

• Condition systems to PNs ...
Petri Net Modeling

Condition Systems

Places output conditions and transitions are labeled with conditions.

A condition is true when a place generating the condition is marked.
Conditions can be represented by linear constraints.

Here: $q_3 + q_9 + \mu_1 \leq 1$ and $q_{10} + q_{11} \leq \mu_4 + \mu_5$. 
Conditions can be represented by linear constraints.

Add \( p_{10} \) for \( q_3 + q_9 + \mu_1 \leq 1 \) and \( p_{11} \) for \( q_{10} + q_{11} \leq \mu_4 + \mu_5 \).
PEP: software for the development, verification, and simulation of parallel programs [Best et al].

1. The language B(PN)$^2$ was developed.

2. Specifications can be written in B(PN)$^2$, SDL, or given as parallel finite automata with B(PN)$^2$ instructions.

3. Specifications are converted to high level PNs (M-nets).

4. When the variables are defined on finite domains, M-nets can be converted to finite safe PNs.
Petri Net Modeling

- PNs are not as expressive as Turing machines.

- However, they easily represent the control flow of a program.

Example: The figure shows a “for loop”.

- Such PN models have been used to detect deadlocks in Ada programs [Murata et al, 1989], [Shatz et al, 1996], [Barkaoui et al, 1998], ...

- Such PN models also applied for synthesis ...

Control Flow
PNs for Program Synthesis

SC of PNs and related methods have been used for control logic design of concurrent programs.

- Predicate Control
- Schedules
- GADARA
- ACTS
Predicate Control [Taraftdar and Garg, 2004].

Objective: Ensure that distributed computations satisfy given predicates.

- A distributed system consists of several processes $P_1, P_2, \ldots$
- $P_i$ is modeled by a sequence of events $e_{i,0}, e_{i,1}, \ldots$
- Synchronization represented graphically by arrows.

```
P_1: e_0 \rightarrow e_1 \rightarrow e_2 \rightarrow e_3
P_2: f_0 \rightarrow f_1 \rightarrow f_2 \rightarrow f_3
```
• Distributed computations must satisfy given predicates.

• Predicates are enforced by means of synchronizations.

• Example: Enforcing $|\epsilon_1 + \epsilon_2 + \epsilon_3 - \phi_1 - \phi_2| \leq 1$. 
The problem could be approached using monitor based supervision.
Scheduling: Deals with generating the correct operation sequence in a multitasking environment.

- Offline scheduling: of interest in certain embedded applications.

- Schedules should satisfy certain properties of interest in spite of decisions made at run time.

- Properties: liveness, execution with bounded memory, ...

- PN models of the software: used in various papers on scheduling, including [Cortadella, 2005], [Hsiung, 2001], [Liu, 2006], [Lin, 1998], and [Sgroi, 1999].
Example: Scheduling the operations of two communicating processes [Cortadella et al, 2003].
PN and possible sequential schedule (one computing resource).
PN and possible concurrent schedule (two computing resources).

Note: The schedules are PN supervisors!
PN Based Design

Simpler Supervisors

Petri Nets and Programming: A Survey 29
The GADARA Project [Y. Wang et al].

Objective: Control the execution of multithreaded programs for deadlock prevention.

- Source code $\rightarrow$ Control Flow Graph.
- Control Flow Graph $\rightarrow$ Petri Nets.
- Monitor places added for deadlock prevention.
- Control logic and source code $\rightarrow$ instrumented executable.
ACTS – A Concurrency Tool Suite; work in progress [Iordache and Antsaklis].
Final Remarks

- Various models used in the context of program synthesis and verification can be converted to PNs.

- Program synthesis methods resemble SC.

- SC can be applied to program synthesis.

- Some enhancements of current SC methods are needed.