Bounded Model Checking
High Level Petri Nets

Xudong He
Florida International University
Miami, FL

Acknowledgements
Funding: partially supported by NSF grants
HRD-0317692 and HRD-0833093
Software Architecture Modeling Methodology (SAM)

- A SAM model \( \{C, h\} \)
  - A set of compositions \( C \)
  - A hierarchical mapping \( h \)
- Dual formalisms
  - Petri nets (behavior \( B \))
  - Temporal logic (property \( S \))
- Correctness
  - \( B \models S \)
SAM Modeling & Analysis Tools

**Modeling**
- Modeling Behavior in HLPNs
- Specifying Properties in FOLTL

**Analysis**
- Simulation (PIPE+)
- Explicit State Model Checking (SAMAT)
- Bounded Model Checking (PIPE+Verifier)
High Level Petri Nets (HLPNs)

High level Petri nets (1980s’):

• Syntax (net structure): \( N = (P, T, F) \), \( P \cap T = \emptyset \), \( P \cup T \neq \emptyset \), \( F \subseteq P \times T \cup T \times P \)

• Static Semantics (net inscription): \( \varphi: P \rightarrow \text{Types} \), \( L: F \rightarrow \text{Labels} \),

  \( R: T \rightarrow \text{Logic Formulas} \) (can be normalized as \( \text{pre-cond} \wedge \text{post-cond} \))

• Dynamic Semantics:

  \text{Initial Marking: } M_0: P \rightarrow \text{Tokens},

  \text{Transition enabling: } \forall p: p \in P. (\overline{L}(p, t): \alpha) \subseteq M(p)) \wedge R(t): \alpha

  \text{Transition firing: } M'(p) = M(p) - \overline{L}(p, t): \alpha \cup \overline{L}(t, p): \alpha

  \text{Execution sequence: } M_0[(T_1, \alpha_1)] > M_1[(T_2, \alpha_2)] > \cdots M_n[(T_{n+1}, \alpha_{n+1})] > \cdots

• Expressive power: control structure and flow, data structure and flow, functional processing
A HLPN model of the five dining philosophers’ problem
The new marking after firing t1 twice with substitutions \( \alpha_1 = \{ x \leftarrow 0, y \leftarrow 1 \} \) and \( \alpha_2 = \{ x \leftarrow 3, y \leftarrow 4 \} \) concurrently.
High Level Petri Nets – An Example

The new marking after firing t2 with substitution $\alpha_3 = \{ x \leftarrow 3, y \leftarrow 4\}$
Bounded Model Checking Process of High Level Petri Nets

A HLPN Model $M$

A Safety Property $\Box f$

A Given Bound $k$

First Order Logic Formula

Counter Example

Sat

Not Sat

$M \models^k f$

$Z3$

$\text{Sat}$

$\text{Not Sat}$
Encoding HLPNs for SMT Solver Z3

DEF

\[ s : \text{STATETUPLE} \]

ASSERT

\[ \text{Initial\_marking}(s_0) \]
\[ \bigwedge_{i=0}^{k-1} \text{Transition}(s_i, s_{i+1}) \]
\[ \bigwedge_{i=0}^{k} \neg \text{Negated\_property}(s_i) \]

CHECK

• DEF defines the global state \( s \) of a HLPN model;
• ASSERT is a first order logic formula encoding an execution sequence of a HLPN up to \( k+1 \) states, and the negation of a safety property
Defining STATETUPLE

<table>
<thead>
<tr>
<th>HLPN Elements</th>
<th>SMT Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLPN Model</td>
<td>Tuple (Places)</td>
</tr>
<tr>
<td>Place Type</td>
<td>Set Type (Tokens)</td>
</tr>
<tr>
<td>Token Type</td>
<td>Tuple (Integer or String Values)</td>
</tr>
<tr>
<td>Primitive Data</td>
<td>Integer or String (Mapping to Integer)</td>
</tr>
</tbody>
</table>

### Dining Philosopher Problem in HLPN model

- **Tuple of Places**
  - Place Thinking
  - Place Chopsticks
  - Place Eating

- **Set of Tokens**

- **Tuple of Integers**
  - Token Type [int]
  - {<0>,<1>,<2>,<3>,<4>}
Defining Transition Formula

- Transition\((s_i, s_{i+1})\) – a disjunction of the transitions \(t\) in the HLPN model (assume the HLPN model has \(n\) transitions):

\[
\text{Transition}(s_i, s_{i+1}) = \bigvee_{j=1}^{n} t_j(s_i, s_{i+1})
\]

- \(t_j(s_i, s_{i+1})\) is defined using an if \(c_0\) then \(c_1\) else \(c_2\) structure, where \(c_0\) is the precondition, \(c_1\) is the post-condition and \(c_2\) updates nothing \(s_{i+1} = s_i\);

- The above naïve translation captures all possible interleaving, and results in exponential formula size growth;

- By exploring net structure and transition dependencies, we can reduce the size of resulting formula.
Reducing the Size of Transition Formula

- When $P_h$ is neither an initial marking place nor property identified place:

$$T(s, s') = (t_{i0}(s, s') \lor t_{i1}(s, s') \lor \ldots) \land (t_{o0}(s, s') \lor t_{o1}(s, s') \lor \ldots)$$
Experiments in PIPE+Verifier

• PIPE+Verifier is to check the first three high level Petri net models from the annual Model Checking Contest @ Petri Nets:
  – Dining Philosophers
  – Shared Memory
  – Token Ring
  – Mondex smart card system (the first pilot project of the International Grand Challenge on Verified Software).

• The detailed experiment results are in the Proc. of ICFEM 2014.
## Related Work

<table>
<thead>
<tr>
<th>Name</th>
<th>Petri Net Type</th>
<th>Analysis Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPiNA</td>
<td>Algebraic Petri Nets</td>
<td>Decision Diagrams</td>
</tr>
<tr>
<td>Cunf</td>
<td>Contextual Nets</td>
<td>Net Unfolding, Satisfiability Solving</td>
</tr>
<tr>
<td>GreatSPN</td>
<td>Stochastic Petri Nets</td>
<td>Decision Diagrams</td>
</tr>
<tr>
<td>ITS-Tools</td>
<td>Timed Petri Nets, ETF, DVE, GAL</td>
<td>Decision Diagrams, Structural Reduction</td>
</tr>
<tr>
<td>LoLA</td>
<td>Place/Transition Nets</td>
<td>Decision Diagrams</td>
</tr>
<tr>
<td>Marcie</td>
<td>Stochastic Petri Nets</td>
<td>Decision Diagrams</td>
</tr>
<tr>
<td>Neco</td>
<td>High Level Petri Nets</td>
<td>Explicit Model Checking</td>
</tr>
<tr>
<td>PNXDD</td>
<td>Place/Transition Nets</td>
<td>Net Unfolding, Decision Diagrams, Topological</td>
</tr>
<tr>
<td>Sara</td>
<td>Place/Transition Nets</td>
<td>Satisfiability Solving, Stubborn Sets, Topological</td>
</tr>
<tr>
<td>CPN Tools</td>
<td>Colored Petri Nets</td>
<td>Explicit Model Checking</td>
</tr>
</tbody>
</table>
Concluding Remarks

• Preliminary results on bounded model checking of HLPNs;

• More Research Issues:
  – How to use net structural patterns to reduce the size of the encoded formula?
  – How to determine the bound $k$?
  – How to deal with more complex transition constraints that contain quantifiers?
Thank you!