Encoding Petri Nets in Answer Set Programming for Simulation Based Reasoning

Saadat Anwar, Chitta Baral (ASU)
Katsumi Inoue (NII)
Presented by Gauvain Bourgne (UPMC)

ICLP 2013
August 27, 2013
Outline

• Motivation
• Introduction
• Encoding Petri Nets in ASP
• Changing the Firing Semantics
• Adding Petri Net Extensions
• Reasoning Example
• Conclusion
Motivation – 1/2

• **Objective:** Understand biological pathways and answer realistic questions about them
  – Questions that a biologist may ask to test students’ understanding of biological systems
  – We found such questions in college level text books
• Many such questions require simulation based reasoning, requiring formalisms that can:
  – Model and simulate pathways
  – Have the ability to model interventions to pathways, and
  – Compare simulations under different circumstances
Motivation – 2/2

• We found Petri Nets to be a suitable starting point for modeling and simulation
  – Petri Net graphical representation and semantics closely matches biological pathway diagrams

• Answering questions requires
  – Certain extensions to the Petri Net model, and
  – Reasoning with multiple simulations and parallel state evolutions

• Existing Petri Net modeling systems do not offer
  – Easy extendibility,
  – Different firing semantics, or
  – Exploring all possible state evolutions

• We use Answer Set Programming (ASP) to address these
Introduction – Petri Nets

A transition $t$ is enabled when all its input places $p_i$ have at least the number of tokens equal to the arc-weight $w_i$ between $(p_i, t)$.

Any number of enabled transitions may fire.

When a transition fires, it consumes $w_i$ tokens from each of its input places $p_i$ and produces $w_j$ tokens to each of its output places $p_j$, where $w_i$ is the arc-weight for $(p_i, t)$ and $w_j$ is the arc-weight for $(t, p_j)$.
Introduction – Pathway as Petri Net

- Places represent substances
- Transitions represent reactions / processes
- Tokens represent substance quantities
- Arc-weights on (place,transition)-arcs represent reactant quantities consumed during the reaction execution
- Arc-weights on (transition,place)-arcs represent product quantities produced at the end of the reaction execution
- Simultaneously firing transitions capture the the parallelism that exists in biological systems
Encoding Petri Nets in ASP

Given a Petri Net \( PN=(P,T,E,W) \)

- \( E^+ \) : \( T \times P \), \( E^- \) : \( P \times T \), \( E = E^+ \cup E^- \)
- \( P \) : finite set of places
- \( T \) : finite set of transitions

<table>
<thead>
<tr>
<th>Petri net representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>f1</strong>: Facts ( \text{place}(p_i) ). where ( p_i \in P ) is a place.</td>
</tr>
<tr>
<td><strong>f2</strong>: Facts ( \text{trans}(t_j) ). where ( t_j \in T ) is a transition.</td>
</tr>
<tr>
<td><strong>f3</strong>: Facts ( \text{ptarc}(p_i,t_j,W(p_i,t_j)) ). where ( (p_i,t_j) \in E^- ) with weight ( W(p_i,t_j) ).</td>
</tr>
<tr>
<td><strong>f4</strong>: Facts ( \text{tparc}(t_i,p_j,W(t_i,p_j)) ). where ( (t_i,p_j) \in E^+ ) with weight ( W(t_i,p_j) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial marking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>i1</strong>: Facts ( \text{holds}(p_i,M_0(p_i),0) ) for every place ( p_i \in P ) with initial marking ( M_0(p_i) ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of token domain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x1</strong>: Facts ( \text{num}(n) ). where ( 0 \leq n \leq \text{ntok} )</td>
</tr>
</tbody>
</table>
Encoding Petri Nets in ASP

• Enabling transitions

  e1: notenabled(T,TS):-ptarc(P,T,N),holds(P,Q,TS),Q<N, place(P), trans(T), time(TS), num(N), num(Q).
  e2: enabled(T,TS) :- trans(T), time(TS), not notenabled(T, TS).

• Firing some enabled transitions

  a1: {fires(T,TS)} :- enabled(T,TS), trans(T), time(TS).
Encoding Petri Nets in ASP

- **Fired transition**: adjusting quantities
  
  **r1**: \(\text{add}(P, Q, T, TS) : - \text{fires}(T, TS), \text{tparc}(T, P, Q), \text{time}(TS).\)
  
  **r2**: \(\text{del}(P, Q, T, TS) : - \text{fires}(T, TS), \text{ptarc}(P, T, Q), \text{time}(TS).\)
  
  **r3**: \(\text{tot_incr}(P, QQ, TS) : \quad \text{QQ} = \#\text{sum}[\text{add}(P, Q, T, TS) = Q : \text{num}(Q) : \text{trans}(T)], \text{time}(TS), \text{num}(QQ), \text{place}(P).\)
  
  **r4**: \(\text{tot_decr}(P, QQ, TS) : \quad \text{QQ} = \#\text{sum}[\text{del}(P, Q, T, TS) = Q : \text{num}(Q) : \text{trans}(T)], \text{time}(TS), \text{num}(QQ), \text{place}(P).\)
  
  **r5**: \(\text{holds}(P, Q, TS + 1) : - \text{holds}(P, Q1, TS), \text{tot_incr}(P, Q2, TS), \text{time}(TS + 1), \text{tot_decr}(P, Q3, TS), Q = Q1 + Q2 - Q3, \text{place}(P), \text{num}(Q; Q1; Q2; Q3), \text{time}(TS).\)

- **Avoiding overconsumption**
  
  **a2**: \(\text{consumesmore}(P, TS) : - \text{holds}(P, Q, TS), \text{tot_decr}(P, Q1, TS), Q1 > Q.\)
  
  **a3**: \(\text{consumesmore} : - \text{consumesmore}(P, TS).\)
  
  **a4**: \(- \text{consumesmore}.\)
Enclosing Petri Nets in ASP

- Given a Petri Net PN=(P,T,E,W)
  - P : finite set of places
  - T : finite set of transitions

\[ \text{f1: } \text{Facts place}(p_i) \text{. where } p_i \in P \text{ is a place.} \]
\[ \text{f2: } \text{Facts trans}(t_j) \text{. where } t_j \in T \text{ is a transition.} \]
\[ \text{f3: } \text{Facts ptarc}(p_i,t_j,W(p_i,t_j)) \text{. where } (p_i,t_j) \in E^- \text{ with weight } W(p_i,t_j). \]
\[ \text{f4: } \text{Facts tparc}(t_i,p_j,W(t_i,p_j)) \text{. where } (t_i,p_j) \in E^+ \text{ with weight } W(t_i,p_j). \]
\[ \text{f5: } \text{Facts time}(ts_i) \text{ where } 0 \leq ts_i \leq k. \]
\[ \text{i1: } \text{Facts holds}(p_i,M_0(p_i),0) \text{ for every place } p_i \in P \text{ with initial marking } M_0(p_i). \]
\[ \text{x1: } \text{Facts num}(n) \text{. where } 0 \leq n \leq ntok \]
\[ \text{e1: } \text{notenabled}(T,TS):-\text{ptarc}(P,T,N),\text{holds}(P,Q,TS),Q<N, \text{place}(P),\]
\[ \text{trans}(T), \text{time}(TS), \text{num}(N), \text{num}(Q). \]
\[ \text{e2: } \text{enabled}(T,TS) :- \text{trans}(T), \text{time}(TS), \text{not notenabled}(T, TS). \]
\[ \text{a1: } \{\text{fires}(T,TS)\} :- \text{_enabled}(T,TS), \text{trans}(T), \text{time}(TS). \]
\[ \text{r1: } \text{add}(P,Q,T,TS) :- \text{fires}(T,TS), \text{tparc}(T,P,Q), \text{time}(TS). \]
\[ \text{r2: } \text{del}(P,Q,T,TS) :- \text{fires}(T,TS), \text{tparc}(P,T,Q), \text{time}(TS). \]
\[ \text{r3: } \text{tot_incr}(P,Q,Q,TS) :- QQ=\#\text{sum}[\text{add}(P,Q,T,TS)=Q:\text{num}(Q):\text{trans}(T)],\]
\[ \text{time}(TS), \text{num}(QQ), \text{place}(P). \]
\[ \text{r4: } \text{tot_decr}(P,Q,Q,TS) :- QQ=\#\text{sum}[\text{del}(P,Q,T,TS)=Q:\text{num}(Q):\text{trans}(T)],\]
\[ \text{time}(TS), \text{num}(QQ), \text{place}(P). \]
\[ \text{r5: } \text{holds}(P,Q,TS+1) :- \text{holds}(P,Q1,TS), \text{tot_incr}(P,Q2,TS), \text{time}(TS+1),\]
\[ \text{tot_decr}(P,Q3,TS), Q=Q1+Q2-Q3, \text{place}(P), \text{num}(Q,Q1;Q2;Q3), \text{time}(TS). \]
\[ \text{a2: } \text{consumesmore}(P,TS) :- \text{holds}(P,Q,TS), \text{tot_decr}(P,Q1,TS), Q1 > Q. \]
\[ \text{a3: } \text{consumesmore} :- \text{consumesmore}(P,TS). \]
\[ \text{a4: } :- \text{consumesmore}. \]
Changing the Firing Semantics

• In biological systems simulation, many times we are only interested in the maximum parallel activity
• Our ASP encoding supports set firing semantics, which goes through all possible combinations of Petri Net state evolutions
• We add the following rules to implement maximal firing semantics:

  a5: could_not_have(T,TS) :- enabled(T,TS), not fires(T,TS),
      ptarc(S,T,Q), holds(S,QQ,TS), tot_decr(S,QQQ,TS), Q > QQ - QQQ.
  a6: :-not could_not_have(T,TS), enabled(T,TS), not fires(T,TS),
       trans(T), time(TS).

• If an interleaved (one at a time) semantics is desired, we add the following rules instead:

  a5’: more_than_one_fires :- fires(T1,TS), fires(T2, TS), T1 != T2,
    time(TS).
  a6’: :- more_than_one_fires.
Adding Petri Net Extensions – 1/2

- **Reset arcs** remove all tokens from their input places when their transition fires.
- It is used to model removal of all quantity of a substance in a biological system.
- To add reset transitions we make the transition weight dependent upon the current time-step. Reset arcs are then added as having arc weight based on the current marking.
- We augment ptarc and tparc predicates with a time parameter replacing f3,f4,e1,r1,r2,a5 with f6,f7,e3,r6,r7,a6 respectively and add f8 for each reset arc in our encoding to implement reset arcs:

  - **f6:** Rules \( \text{ptarc}(p_i,t_j,W(p_i,t_j),ts_k) :- \text{time}(ts_k) \). for each non-reset arc \((p_i,t_j) \in E^-\)
  - **f7:** Rules \( \text{tparc}(t_i,p_j,W(t_i,p_j),ts_k) :- \text{time}(ts_k) \). for each non-reset arc \((t_i,p_j) \in E^+\)
  - **e3:** \( \text{notenabled}(T,TS) :- \text{ptarc}(P,T,N,TS), \text{holds}(P,Q,TS), Q < N, \text{place}(P), \text{trans}(T), \text{time}(TS), \text{num}(N), \text{num}(Q) \).\)
  - **r6:** \( \text{add}(P,Q,T,TS) :- \text{fires}(T,TS), \text{tparc}(T,P,Q,TS), \text{time}(TS) \).
  - **r7:** \( \text{del}(P,Q,T,TS) :- \text{fires}(T,TS), \text{ptarc}(P,T,Q,TS), \text{time}(TS) \).
  - **f8:** Rules \( \text{ptarc}(p_i,t_j,X,ts_k) :- \text{holds}(p_i,X,ts_k), \text{num}(X), X > 0. \) for each reset arc between \( p_i \) and \( t_j \) using \( X = M_k(p_i) \) as arc-weight at time step \( ts_k \).
  - **a6:** \( \text{could_not_have}(T,TS) :- \text{enabled}(T,TS), \text{not fires}(T,TS), \text{ptarc}(S,T,Q,TS), \text{holds}(S,QQ,TS), \text{tot_decr}(S,QQQ,TS), Q > QQ-QQQ. \)
Adding Petri Net Extensions – 2/2

- **Inhibitor arcs** inhibit a transition from firing as long as its source place has any tokens.
- It models biological process inhibition.
- These are added to our encoding with the following additional rules:
  1. **Rules** \( \text{iptarc}(p_i,t_j,l,ts_k) :- \text{time}(ts_k). \) for each inhibitor arc between \( p_i \in I(t_j) \) and \( t_j \).
  2. **Rules** \( \text{notenabled}(T,TS) :- \text{iptarc}(P,T,N,TS), \text{holds}(P,Q,TS), \text{place}(P), \text{trans}(T), \text{time}(TS), \text{num}(N), \text{num}(Q), Q \geq N. \)
- **Read arcs** inhibit a transition from firing as long as its source place has less than the number of tokens as the arc-weight between them.
- It models biological processes that require substances in different quantity to initiate a reaction than what is consumed during the reaction.
- These are added to our encoding with the following additional rules:
  1. **Rules** \( \text{tptarc}(p_i,t_j,QW(p_i,t_j),ts_k) :- \text{time}(ts_k). \) for each read arc \( (p_i,t_j) \in Q \).
  2. **Rules** \( \text{notenabled}(T,TS) :- \text{tptarc}(P,T,N,TS), \text{holds}(P,Q,TS), \text{place}(P), \text{trans}(T), \text{time}(TS), \text{num}(N), \text{num}(Q), Q < N. \).
Reasoning Example – 1/2

• “At one point in the process of glycolysis, both dihydroxyacetone phosphate (DHAP) and glyceraldehyde 3-phosphate (G3P) are produced. Isomerase catalyzes the reversible conversion between the two isomers. The conversion of DHAP to G3P never reaches equilibrium and G3P is used in the next step of glycolysis. What would happen to the rate of glycolysis if DHAP were removed from the process of glycolysis as quickly as it was produced?” [Campbell Biology]

• Construct Petri Net model of the normal pathway (shown in solid lines) as the relevant sub-portion of the glycolysis pathway

• Extend the model with a reset arc to a new transition “tr” (shown in dotted lines) that models the intervention to remove DHAP as soon as it is produced

• Simulate both for 15-time steps using maximal firing set semantics

• Compute the rate of glycolysis as the amount of BPG13 produced over the simulation length
Reasoning Example – 2/2

- Average quantity of BPG13 produced over simulation of length 15.
- The quantity is higher for the normal pathway, hence the rate will also be higher, since rate is quantity produced over time.
- Hence immediate removal of DHAP will result in lower rate of glycolysis.

- Spread of unique quantity of BPG13 produced by various simulation runs.
- Higher variation in BPG13 production when DHAP is removed immediately after production.
- The rate of production of BPG13 and hence glycolysis can reduce to zero in some cases.
Conclusion

• Petri Nets are an appropriate formalism for modeling biological pathways and answering questions about such pathways
• Petri Nets can be easily implemented in ASP
• ASP provides an elaboration tolerant method to encode various Petri Net extensions and firing semantics
• Our encoding has a low specification-implementation gap
• An ASP implementation explores all possible state evolutions and has the capability for additional reasoning
• Our focus has been less on performance and more on ease of encoding, extensibility, exploring all possible state evolutions, and strong reasoning abilities not available in other Petri Net implementations examined
• We will extend this work in the future to include other Petri Net extensions
Questions?