Encoding Reversing Petri Nets in Answer Set Programming

Kyriaki Psara

Joint work with Yiannis Dimopoulos, Eleftheria Kouppari, and Anna Philippou
Our goal

• To develop concise and efficient logical representation of reversible system behaviour
• To implement a systematic way for the automatic analysis and reasoning about Reversing Petri Nets (RPNs)
• To highlight how an Answer Set Programming (ASP) can be used to reason about the behaviour of RPN models
Answer Set Programming

• A novel paradigm for applying declarative logic programming techniques
• ASP is a set of rules of the form:
  \[ A_0 \leftarrow A_1, \ldots, A_m, \text{not} A_{m+1}, \ldots, \text{not} A_n \]
• Models a system as well as a query about the system by devising a logic program
  – clingo (https://potassco.org/clingo/)
• ASP has been used to model
  – 1-safe Place/Transition nets, basic Petri Nets, Coloured Petri nets, Petri net extensions
Reversing Petri nets

• A variation of Petri nets a reversible approach to Petri nets, which allows the transitions of a net to be reversed [Philippou & Psara 2018]

• A Reversing Petri net is a tuple \((P, T, A, B, F)\) where

  – \(P\) is a finite set of places
  – \(T\) is a finite set of transitions
  – \(A\) is a finite set of bases or tokens
  – \(B \subseteq A \times A\) is a set of bonds
  – \(F : (P \times T \cup T \times P) \rightarrow 2^{A \cup A \cup B \cup B}\) is a set of directed arcs
RPNs to ASP

- The basic predicates that represent the input network are:
  - $trans(T)$, $token(Q)$, $place(P)$,
    $ptarc(P,T,Q)$, $tparc(T,P,Q)$,
    $ptarcbond(P,T,Q1,Q2)$, $tparcbond(P,T,Q1,Q2)$
Marking and States

- **Marking**: A distribution of tokens/bonds on places:
  \[ M : P \rightarrow 2^{A \cup B} \]

- **History**: assigns a memory to a transition
  \[ H : T \rightarrow 2^N \]

- **State**: a pair of a marking and a history
  \[ \langle M, H \rangle \]
From RPNs to ASP

• **Marking:**
  \[ \text{holds}(P, Q1, TS), \text{holds bonds}(P, Q1, Q2, TS) \]

• **History:** $TS$ is the step of the simulation
  – the simulation length is encoded by the last argument $TS$ of the predicates of our model

```
1 holds(u, b, 0). holds(w, a, 0). holds(u, c, 0).
2 holds bonds(u, b, c, 0)
```
**Definition:** A transition $t$ is *forward enabled* in an RPN state $\langle M, H \rangle$ if:

- All tokens/bonds required for the transition are available on its incoming places
- Tokens/bonds cannot be cloned
- Bonds cannot be recreated

|   |   |
|---|---|
| 1 | `notenabled(T,TS):-ptarc(P,T,Q),not holds(P,Q,TS).` |
| 2 | `notenabled(T,TS):-ptarcbond(P,T,Q1,Q2),` |
| 3 | `    not holdsbonds(P,Q1,Q2,TS).` |
| 4 | `notenabled(T,TS):-tparc(T,P1,Q1),tparc(T,P2,Q2),P1!=P2,` |
| 5 | `    connected(P,Q1,Q2,TS),ptarc(P,T,\_).` |
| 6 | `notenabled(T,TS):-ptarcbond(T,TP,Q1,Q2),ptarc(PT,T,\_),` |
| 7 | `    holdsbonds(PT,Q1,Q2,TS),` |
| 8 | `    not pttarcbond(PT,T,Q1,Q2).` |
| 9 |   |
| 10|   |
| 11|   |
| 12| `enabled(T,TS):-not notenabled(T,TS).` |
| 13| `{fires(T,TS)}:-enabled(T,TS).` |
Forward Execution

[Diagram of a forward execution process with labeled states and transitions]

- From state a, proceed to t₁ with input a.
- From t₁, proceed to a state with input a.
- From this state, proceed to t₃ with input a.
- From t₃, proceed to an output with a-b.

- From state b, proceed to t₂ with input b.
- From t₂, proceed to a state with input b.
- From this state, proceed to t₃ with input b.
- From t₃, proceed to an output with a-b.
Forward Execution

[1]

July 2020 12th Conference on Reversible Computation 2020
Forward Execution

[1] \( t_1 \)

[2] \( t_2 \)

\( a \rightarrow t_1 \rightarrow a \)

\( b \rightarrow t_2 \rightarrow b \)

\( a \rightarrow a \)

\( b \rightarrow b \)

\( a-b \rightarrow t_3 \rightarrow a-b \)
Forward Execution

1. a → t_1 → a
2. b → t_2 → b
3. a → t_3 → a-b
   b → t_3 → a-b
Forward Execution

**Definition:** If a transition $t$ is forward enabled in an RPN then the marking is updated as follows:

```
1  addBond(TP,Q1,Q2,TS+1):=-fires(T,TS),tparcbond(T,TP,Q1,Q2).
2  addBond(TP,Q1,Q2,TS+1):=-fires(T,TS),tparc(T,TP,Q),
3     pparc(P,T,Q),connected(P,T,Q1,TS),
4     holdsbonds(P,T,Q1,Q2,TS).
5
6  delBond(P,T,Q1,Q2,TS+1):=-fires(T,TS),tparcbond(P,T,Q1,Q2).
7  delBond(P,T,Q1,Q2,TS+1):=-fires(T,TS),tparc(P,T,Q),
8     connected(P,T,Q1,TS),holdsbonds(P,T,Q1,Q2,TS).
```

```
1  holds(P,Q,TS):-holds(P,Q,TS-1),not del(P,Q,TS).
2
3  holdsbonds(P,Q1,Q2,TS):-holdsbonds(P,Q2,Q1,TS).
4  holdsbonds(P,Q1,Q2,TS):-addBond(P,Q1,Q2,TS).
5  holdsbonds(P,Q1,Q2,TS):-holdsbonds(P,Q1,Q2,TS-1),
6     not delBond(P,Q1,Q2,TS).
```
Definition: A transition $t$ is \textit{causally enabled} in an RPN state $\langle M, H \rangle$ if:

- There are no transitions causally dependent on $t$
  - Two transitions are causally dependent if they manipulate the same tokens
- The transition has been executed

\begin{verbatim}
1 dependent(T2,T1,TS):-tparc(T1,_ ,Q),ptarc(_,T2,Q),
                   H2=#max{H:transHistory(T2,H,TS),history(H)},
                   H1=#max{H:transHistory(T1,H,TS),history(H)},
                   H2>H1,H1>0.
2 dependent(T2,T1,TS):-tparc(T1,_ ,Q),ptarc(_,T2,Q1),
                   connected(_,Q,Q1,TS),
                   H2=#max{H:transHistory(T2,H,TS),history(H)},
                   H1=#max{H:transHistory(T1,H,TS),history(H)},
                   H2>H1,H1>0.
3 notenedabledC(T,TS):-dependent(T1,T,TS),trans(T1),trans(T).
4 notenedabledC(T,TS):-irreversible(T).
5 enabledC(T,TS):-trans(T),time(TS),not notenedabledC(T,TS),
                   transHistory(T,H,TS),H>0.
6 {reversesC(T,TS)}:-enabledC(T,TS),trans(T),time(TS).
\end{verbatim}
Causal Execution

Diagram:

- \( t_1 \) with inputs \( a \) and \( b \)
- \( t_2 \) with inputs \( b \)
- \( t_3 \) with inputs \( a \) and \( b \)
- Connections: \( a \to t_1 \), \( a \to t_3 \), \( b \to t_2 \), \( b \to t_3 \), \( a \to a-b \)
Causal Execution

[1] t_1

[2] t_2

a → t_1 → a → t_3 → a-b

b → t_2 → b → t_3

a
Causal Execution

\[ a \rightarrow t_1 \rightarrow a \rightarrow t_3 \rightarrow a-b \]

\[ b \rightarrow t_2 \rightarrow b \rightarrow t_3 \]

[2]
Causal Execution
Causal Execution

Definition: If a transition $t$ is causally reversible in an RPN then the marking is updated as follows

```prolog
breakBond(P,Q1,Q2,TS):-breakBond(P,Q2,Q1,TS).
breakBond(P,Q1,Q2,TS):-reversesC(T,TS),tparcbond(T,P,Q1,Q2).

addBond(PT,Q1,Q2,TS):-reversesC(T,TS),tparcbond(PT,T,Q1,Q2).
addBond(PT,Q1,Q2,TS):-reversesC(T,TS),tparc(PT,T,Q),
                     tparc(T,TP,Q),holdsbonds(TP,Q1,Q2,TS),
                     not breakBond(TP,Q1,Q2,TS),
                     connected(TP,Q,Q1,TS).
addBond(PT,Q1,Q2,TS):-reversesC(T,TS),tparc(PT,T,Q),
                     tparc(T,TP,Q),holdsbonds(TP,Q1,Q2,TS),
                     not breakBond(TP,Q1,Q2,TS),
                     connected(TP,Q,Q2,TS).

delBond(TP,Q1,Q2,TS):-delBond(TP,Q2,Q1,TS).
delBond(TP,Q1,Q2,TS):-reversesC(T,TS),tparcbond(T,TP,Q1,Q2).
delBond(TP,Q1,Q2,TS):-reversesC(T,TS),tparc(PT,T,Q),
                     tparc(T,TP,Q),holdsbonds(TP,Q1,Q2,TS),
                     connected(TP,Q,Q1,TS).
delBond(TP,Q1,Q2,TS):-reversesC(T,TS),tparc(PT,T,Q),
                     tparc(T,TP,Q),holdsbonds(TP,Q1,Q2,TS),
                     connected(TP,Q,Q2,TS).
delBond(TP,Q1,Q2,TS):-breakBond(TP,Q1,Q2,TS).
```
Queries

A reachable state where some place holds the bond $a - c$

goal:- connected(P,a,c,T),place(P),time(T).
:- not goal.
fires(t1,0) fires(t3,1)
Queries

1. A reachable state where some place holds a bond with at least three tokens

```
1 goal: ¬C > 1, C = #count{K2:connected(P,K1,K2,T),token(K2)},
2  holds(P,K1,T).
3  :- not goal.
4  fires(t1,0) fires(t2,1) fires(t4,2)
```

A reachable state where some place holds a bond with at least three tokens
A reachable state that first creates a bond with at least three tokens, and a bond with a and c but without b

 Queries

1. goal1(T):= C>1, C=#count{K2:connected(P,K1,K2,T),token(K2)}, holds(P,K1,T).
2. goal2(T):= connected(P,a,c,T), not connected(P,a,b,T), time(T).
3. goal:=- goal1(T1),goal2(T2),T2>T1, time(T1), time(T2).
4. :- not goal.
5. fires(t1,0) fires(t2,1) fires(t4,2) reversesC(t4,3)
6. reversesC(t2,4) fires(t3,5)
Concluding Remarks

• Presented a methodology for analysing reversible systems modelled as RPNs based on ASP

• We argue that ASP:
  – allows an expressive and flexible methodology for defining models and their properties
  – can handle difficult queries on complex models efficiently
Current and Future Work

- Extend our translation to out-of-causal reversibility
- Capture a variety of RPN properties
- Allow multiple tokens of the same base/type to occur in a model
- Use the graphical interface of existing RPN tools to model ASP
  - Colored Petri Nets:
  - Customised RPN tool
Thank you for your attention!
References

[Heljanko and Niemela 2003] K. Heljanko and I. Niemelä. Bounded LTL model checking with stable models. TPLP, 3(4-5):519–550, 2003.

[Anward et al. 2013] S. Anwar, C. Baral, and K. Inoue. Encoding Petri nets in answer set programming for simulation based reasoning. TPLP, 13(4-5-Online-Supplement), 2013.

[Anward et al. 2013] S. Anwar, C. Baral, and K. Inoue. Encoding higher level extensions of Petri nets in answer set programming. In Proceedings of LPNMR 2013, LNCS 8148, pages 116–121. Springer, 2013.

[Anward et al. 2014] S. Anwar, C. Baral, and K. Inoue. Simulation-based reasoning about biological pathways using Petri nets and ASP. Logical Modeling of Biological Systems, pages 207–243, 2014.

[Philippou & Psara 2018] A. Philippou and K. Psara. Reversible computation in Petri nets. In Proceedings of RC 2018, LNCS 11106, pages 84–101. Springer, 2018.