Petri nets for modeling problem
of logic programming and knowledge representation

Perspective for modeling Fluent Calculus

Doan Trung Son

12th of July, 2012
...what in a computational architecture of [Barret 2010]

- Combining interacting procedural model (based on Petri net) and inferential model (called the Probabilistic Relational Model) to capture structure of actions.

...but

- Inference cannot be implemented on achieved Petri nets
Question?

How to apply Petri nets to Fluent Calculus and vice versa?

Similar thing...

- Fluent Calculus consider state as multiset of fluent and represent state on the term level
- High-level Petri nets has state representing the distribution of multiset tokens. Each places stand for predicate
- Mathematical foundation: Multiset satisfies foundational axiom (AC1)
- Relation between transition rule in Petri nets with update axioms of fluent in Fluent Calculus.
Structure

Part 1. Definitions of Petri nets

Part 2. Applications of Petri nets

Problems:
- Robot planning
- Well-founded semantics
What kind of Petri nets for investigation:

1. High-level Petri nets
   - Robot planning
     
     \(... for modeling Fluent Calculus in High-level Petri nets\)

2. Extended Petri nets
   - Well-founded semantics
     
     \(... for modeling Well-founded model in Extended Petri nets\)
Definition: Place/Transition nets

Place/Transition nets is 5-tuple: \( \langle P, T, F, W, M_0 \rangle \)

- \( P \) is a finite set of places
- \( T \) is a finite set of transitions
- \( F \subseteq (P \times T) \cup (T \times P) \) is set of arcs
- \( W: F \to \{\mathbb{N}/0\} \) is weight function, assigning a positive integer to each arc
- \( M_0: P \to \mathbb{N} \) is the initial marking representing the initial distribution of tokens
Place/Transition nets example

The Place/Transition nets above is defined as follows:

- \( P = \{p_1, p_2, p_3, p_4, p_5, p_6\} \) and \( T = \{t_1, t_2, t_3, t_4\} \) where:

- \( F = \{\langle p_1, t_1 \rangle, \langle t_1, p_2 \rangle, \langle t_1, p_3 \rangle, \langle p_2, t_2 \rangle, \langle t_2, p_4 \rangle, \langle p_4, t_4 \rangle, \langle p_3, t_3 \rangle, \langle t_3, p_5 \rangle, \langle p_5, t_4 \rangle, \langle t_4, p_6 \rangle\} \)

- \( W = \{\langle p_1, t_1 \rangle : 2, \langle t_1, p_2 \rangle : 1, \langle t_1, p_3 \rangle : 1, \langle p_2, t_2 \rangle : 2, \langle t_2, p_4 \rangle : 1, \langle p_4, t_4 \rangle : 1, \langle p_3, t_3 \rangle : 1, \langle t_3, p_5 \rangle : 1, \langle p_5, t_4 \rangle : 1, \langle t_4, p_6 \rangle : 1\} \)

- \( M_0 = \{p_1 \rightarrow 3, p_2 \rightarrow 0, p_3 \rightarrow 0, p_4 \rightarrow 1, p_5 \rightarrow 0, p_6 \rightarrow 0\} = \langle 3, 0, 0, 1, 0, 0 \rangle \)
Transition Rule

Let \( \langle P, T, F, W, M_0 \rangle \) be a Petri net and marking \( M : P \to \mathbb{N} \)

**Firing condition:**

- Transition \( t \in T \) is **enabled** iff every input place \( p \) of transition \( t \) satisfy: \( M(p) \geq W(p, t) \)

**Firing rule:**

- Transition \( t \in T \) may **fire** and produce the successor making \( M' \) written by \( M \xrightarrow{t} M' \)

\[
\forall p \in P : M'(p) = M(p) - W(p, t) + W(t, p)
\]

**NOTE:** if \( \langle p, t \rangle \notin F \) or \( \langle t, p \rangle \notin F \) then \( W(p, t) = 0 \) or \( W(t, p) = 0 \) correspondingly.
Definition: High level Petri nets

High-level Petri nets is 8-tuple: \( \langle P, T, F, \text{Sig}, V, \text{Sort}, \text{AN}, M_0 \rangle \) where:

- **P**: set of **Places**
- **T**: set of **Transitions**
- **F**: set of **Arcs**. Each arc is pair of \((Arc, TypeArc)\). \(TypeArc : Arc \rightarrow N\) is a function which assigns types to arcs.
- **Sig**: \(\text{Sig} = (S, O)\) is a **Boolean signature**
- **V**: set of **Variables**
- **Sort**: \(\text{Sort} : P \rightarrow S\) is a function which assigns sorts to places
- **AN**: \(\text{AN} = (A, TC)\): a pair of **net annotations**
  - \(A : F \rightarrow \text{TERM}(O \cup V)\): function that annotates each arc with a term of the same sort as that of the associated place
  - \(TC : T \rightarrow \text{TERM}(O \cup V)_{\text{Bool}}\): function that annotates transitions with Boolean terms
- **\(M_0\)**: **Initial marking** \(M_0 : P \rightarrow \text{TERM}(O)\): function which associates a ground term with each place or \(\forall p \in P, M_0(p) \in \text{TERM}(O)_{\text{Sort}(p)}\).
High level Petri nets

∇ Enabling transition:
The marking of each input place of the transition satisfies the demand imposed on it by its arc annotation evaluated for the particular transition mode.

∇ Transition rule:
1. **For each input places of the transition:** the enabling tokens of the input arc annotation in the transition mode are substracted from the input places’s marking
2. **For each output place of the transition:** the multiset of tokens of the evaluated output arc annotation is added to the output place’s marking.
**Extended Petri nets (EPN)**

**EPN:** Place/Transition nets with an extended set of inhibitor arcs

\[ \nabla \text{Siphon (deadlock):} \]

A nonempty set of places \( S \) in an ordinary Petri net is called a siphon if every transition having an output places in \( S \) has an input place in \( S \).

\[ \star \text{Siphon in EPN is a siphon when we consider all inhibitor arcs that are deleted } \]

\[ \star \text{Greatest siphon in EPN is the union of all possible siphons in the net} \]
I. Blocks World Problem

**INITIAL STATE:** OnTable(A), OnTable(B), Clear(B), Clear(C), HandEmpty, On(C,A)

**OPERATORS:**

- **PICKUP**($x$) Pre-conditions and delete condition: OnTable($x$), Clear($x$), HandEmpty
  Add condition: Holding($x$)

- **PUTDOWN**($x$) Pre-condition and delete condition: Holding($x$)
  Add conditions: OnTable($x$), Clear($x$), HandEmpty

- **UNSTACK**($x,y$) Pre-condition and delete condition: Clear($x$), On($x,y$)
  Add conditions: Holding($x$), Clear($y$)

- **STACK**($x,y$) Pre-condition and delete condition: Holding($x$), Clear($y$)
  Add conditions: HandEmpty, On($x,y$), Clear($x$)

**GOAL:** OnTable(C), Clear(A), On(B,C), On(A,B)
Modeling for single-hand robot planning

The high level Petri net model for single-robot planning problem

**Initial state:** Ontable(A) • Ontable(B) • Clear(B) • Clear(C) • Handempty • On(C,A)

\[
M_0 = (\{\mathcal{A}, \mathcal{B}\}, \{\mathcal{B}, \mathcal{C}\}, \{\mathcal{A}\}, \{\mathcal{A}\}, \{\mathcal{C}, \mathcal{A}\})
\]

**Goal state:** Ontable(C) • Clear(A) • Handempty • On(A,B) • On(B,C)

\[
M_\mathcal{G} = (\{\mathcal{C}\}, \{\mathcal{A}\}, \{\mathcal{A}\}, \{\mathcal{A}\}, \{\mathcal{A}, \mathcal{B} = \mathcal{B}, \mathcal{C}\})
\]
Modeling for multi-hand robot planning

The high Petri net model for multi-robot planning problem
II. Representation of normal logic program in Petri net

Why Fix-point semantics is defined for Petri nets?

- Logical equivalence
- Alternative theorem proving technique
Transformation Procedure 1

Instantiated normal logic program clause $C$ in form of:

$$ A \leftarrow B_1, \ldots, B_n, \neg D_1, \ldots, \neg D_m $$

Transformation 1:

1. Draw a transition and label it with $C$
2. Draw a place and arc from transition $C$ to the place and label the place with $A$
3. If $n \neq 0$ the repeat the following for $i = 1$ to $n$:
   - Draw a place and a normal arc from the place to the transition $C$ and label the place with $B_i$
4. If $m \neq 0$ the repeat the following for $i = 1$ to $m$:
   - Draw a place and an inhibitor arc from the place to the transition $C$ and label the place with $D_i$
Transformation 2:

1. For each clause, do the Transformation 1

2. For each label of the place produced by the above step, superimpose all the places that have the same label

Let the normal logic program $P$ be:

$C1: b \leftarrow \neg a$

$C2: c \leftarrow \neg b$

$C3: c \leftarrow a, \neg p$

$C4: p \leftarrow \neg q$

$C5: q \leftarrow \neg p, b$
Notes:

\( \triangledown PL \): set of all places
\[ PL = \{ a, b, c, p, q \} \]
\( \triangledown TR \): set of all transitions
\[ TR = \{ C1, C2, C3, C4, C5 \} \]
\( \triangledown ST \): set of all source transition
\[ ST = \{ \emptyset \} \]
\( \triangledown GS_1 \): a greatest siphon
\[ GS_1 = \{ a \} \]
\( \triangledown B_P \): the Herbrand base of a program P
We have $I_0 = \langle \emptyset, \emptyset \rangle$ and $I_i = W_p(I_{i-1}) = \langle T_i, F_i \rangle$ for $i \geq 1$:

∇ Algorithm:

▷ Step 1: For $i = 0$:
\[
\begin{align*}
T_0 &= \emptyset \\
F_0 &= \emptyset
\end{align*}
\]

▷ Step 2: For $i = 1$:
\[
\begin{align*}
T_1 &= \text{all output places of source transitions} \\
F_1 &= GS_1 \cup (B_p - PL)
\end{align*}
\]
Algorithm (2): 3 steps compute the well founded model

▶ **Step 3:** For \( i = 2, 3, 4 \ldots \)

1. \( T_i = T_{i-1} \cup T'_i \)

where:

\( T'_i = \{ x \in PL \mid \text{for some input transition } t \text{ of } x \} \)

if there exists input place \( u \in F_{i-1} \) of an inhibitor arc of \( t \) (**Form 1**)

or if there exists input place \( u \in T_{i-1} \) of a non-inhibitor arc of \( t \) (**Form 2**)

2. \( F_i = GS_i \cup (B_p - PL) \)

where:

\( GS_i \): the greatest siphon in the net obtained by deleting all transitions of \( TR_i \)

\( TR_i = \{ t \in TR \mid \text{there exists either:} \} \)

a non-inhibitor arc from \( u \in F_{i-1} \) to \( t \) (**Form 3**)

or an inhibitor arc from \( u \in T_{i-1} \) to \( t \) (**Form 4**)


Graphical interpretation for Step 3

Computing $T_i$

$T_i = T_{i-1} \cup T_i'$

Form 1

and $u \in F_{i-1}$ then $T_i = T_{i-1} \cup \{x\}$

Form 2

and $u \in T_{i-1}$ then $T_i = T_{i-1} \cup \{x\}$

Form 3

and $u \in F_{i-1}$ then Deleting transition $t$

Form 4

and $u \in T_{i-1}$ then Deleting transition $t$

Computing $GS_i$

$F_i = GS_i \cup (B_p - PL)$
Graphical interpretation for Step 3

- Computing $T_i$
- $T_i = T_{i-1} \cup T'_i$

- Form 1
  - $u \rightarrow x$ and $u \in F_{i-1}$ then $T_i = T_{i-1} \cup \{x\}$

- Form 2
  - $u \rightarrow x$ and $u \in T_{i-1}$ then $T_i = T_{i-1} \cup \{x\}$

- Form 3
  - $u \rightarrow x$ and $u \in F_{i-1}$ then Deleting transition $t$

- Form 4
  - $u \rightarrow x$ and $u \in T_{i-1}$ then Deleting transition $t$

- Computing $GS_i$
- $F_i = GS_i \cup (B_p - PL)$
Example 1:

Steps of Implementation

1. For $i = 0$: $T_0 = \{\emptyset\}$  $F_0 = \{\emptyset\}$
2. For $i = 1$: $T_1 = \{\emptyset\}$  $F_1 = \{a\}$
3. For $i = 2$: $T_2 = \{b\}$  $F_2 = \{a\}$
4. For $i = 3$: $T_3 = \{b\}$  $F_3 = \{a, c\}$
5. For $i = 4$: $T_4 = \{b\}$  $F_4 = \{a, c\}$

∴ Well-founded model is $< \{b\}, \{a, c\} >$
Example 2:

Let the normal logic program P be:

\[
C_1 : q(a) \leftarrow r(a), \neg q(b) \\
C_2 : q(b) \leftarrow \\
\]

where \( q, r \): are predicate symbols and \( a, b \) are constant symbols.

Steps of Implementation

1. For \( i = 0 \):
   \( T_0 = \{ \emptyset \} \) and \( F_0 = \{ \emptyset \} \)
2. For \( i = 1 \):
   \( T_1 = \{ q(b) \} \) and \( F_1 = \{ r(a), r(b), q(a) \} \)
3. For \( i = 2 \):
   \( T_2 = T_1 \) and \( F_2 = F_1 \)

\[\therefore\] Well-founded model is

\[< \{ q(b) \}, \{ r(a), r(b), q(a) \} >\]
Example 3:

Let the normal logic program P be:

\[ C_1 : \ b \leftarrow \neg a \]
\[ C_2 : \ c \leftarrow b \]
\[ C_3 : \ b \leftarrow c \]
\[ C_4 : \ c \leftarrow \neg d \]
\[ C_5 : \ e \leftarrow c \]
\[ C_6 : \ a \leftarrow \]
\[ C_7 : \ d \leftarrow \]

Steps of Implementation

1. For \( i = 0 \):
   \( T_0 = \{\emptyset\} \) and \( F_0 = \{\emptyset\} \)
2. For \( i = 1 \):
   \( T_1 = \{a, d\} \) and \( F_1 = \{\emptyset\} \)
3. For \( i = 2 \):
   \( T_2 = \{a, d\} \) and \( F_2 = \{b, c, e\} \)

\[ \therefore \text{Well-founded model is } < \{a, d\}, \{b, c, e\} > \]
Summary

1. Planning problem
   - Robot planning

2. Logic program
   - Well-founded semantics
Further investigation

- Equivalent components with Fluent Calculus.
- Search Algorithms in High-level Petri-nets.
- Real-time systems.
- Correlation with Neural-Symbolic systems.
Petri nets for modeling problems
logic program and knowledge representation

Application Artificial Intelligent Neural Network
Algorithm State Discuss Agent Action language Fluent

Petri nets for modeling problems
logic program and knowledge representation

P E T R I N E T

Semantic Operator
Representation Combination Feature Graph Space
First order logic Flux player Goal Syntax
Logic Solution Rule Transition Planing Well-founded semantic
Interpretation Fix point semantics Model checking Lattice theory
Reference (1)

[1]. High level Petri nets - Concepts, Definitions and Graphical Notation.


Tadao Mutara and Du Zhang.

[2]. A predicate transition net model for parallel interpretation of logic programs.

Tadao Mutara, Peter C. Nelson and Jacgeol Yim.

[3]. A predicate transition net model for multi agent planning.

Reference (2)

Teruhiro Shimura, Jorge Lobo and Tadao Mutara.

[4]. An extended Petri net model for normal logic programs.  

Tadao Mutara, V.S. Subrahmanian and Toshiro Wakayama.

[5]. A Petri net model for reasoning in the presence of inconsistency.  

Tadao Mutara.

[6]. Some recent application of high level Petri nets.
Reference (3)

Tadao Mutara fellow IEEE.

[7]. Petri nets: properties, analysis and applications.  
Proceedings of the IEEE.

Michael Thielscher.

[8]. Introduction to the Fluent Calculus.

Leon Rubin Barrett.

[9]. An Architecture for Structured, Concurrent, Real-time Action  
Doctor dissertation in Computer Science.
Multiset

**Definition Multiset**

Multiset $B$ over non empty basic set $A$ is a function $B : A \rightarrow N$ which associates a multiplicity with each of the basic elements. Multiplicity of $a \in A$ in $B$, is given by $B(a)$. The set of multisets over $A$ is denoted by $\mu A$.

**Sum Representation**

A multiset may be represented by their multiplicities as symbolic sum of basic elements: $B = \sum_{a \in A} B(a) \cdot a$
Multiset

Membership
Given a multiset $B \in \mu A$, $a \in A$ is a member of $B$, denoted $a \in B$ if $B(a) \geq 0$ and conversely if $B(a) = 0$ then $a \notin B$.

Empty multiset
The empty multiset $\emptyset$ has no member: $\forall a \in A, \emptyset(a) = 0$.

Multiset equality and comparison
Two multiset $B_1$ and $B_2 \in \mu A$ are equal $B_1 = B_2$ iff $\forall a \in A, B_1(a) = B_2(a)$.

$B_1$ is less than or equal to $B_2$, $B_1 \leq B_2$ iff $\forall a \in A, B_1(a) \leq B_2(a)$.
Multiset Operation

The addition operation and subtraction partial operation on two multiset $B_1, B_2 \in \mu A$:

$B = B_1 + B_2$ iff $\forall a \in A : B(a) = B_1(a) + B_2(a)$

$B = B_1 - B_2$ iff $\forall a \in A$ if $B_1(a) \geq B_2(a)$ then

$B(a) = B_1(a) - B_2(a)$

Scalar multiplication of a multiset $B_1 \in \mu A$ by a natural number $n \in N$ is defined as: $B = nB_1$ if $\forall a \in A$,

$B(a) = n \times B_1(a)$ where $\times$ is arithmetic multiplication.
II. Horn clause logic program

Let the Horn clause logic program be:

\[ C_1 : Parent(David, Mary) \leftarrow \]
\[ C_2 : Parent(Mary, Tom) \leftarrow \]
\[ C_3 : Ancestor(x, y) \leftarrow Parent(x, y) \]
\[ C_4 : Ancestor(x, z) \leftarrow Parent(x, y), Ancestor(y, z) \]
\[ C_5 : \leftarrow Ancestor(x, Tom) \]

There are 2 firing sequences \( \delta_1, \delta_2 \) which start from the empty marking, fire the goal transition \( C_5 \) and end at the empty marking:

\[ \delta_1 = \langle C_2, C_3, C_5 \rangle \text{ with substitution } \{Mary|x\} \text{ then } x = Mary \]
\[ \delta_2 = \langle [C_1, C_2], C_3, C_4, C_5 \rangle \text{ with substitution } \{David|x\} \text{ then } x = David \]
III. Non-monotonic reasoning

Let the non-monotonic logic program be:

\[ C1 : a \leftarrow \]
\[ C2 : b \leftarrow, unless(c) \]
\[ C3 : a \leftarrow, unless(b) \]

◊ Fix point of non-monotonic theory can be represented as the support of a firing sequence which is maximal and consistent.

◊ There are two fix point \{a, b\} and \{a, c\}
IV. Example of annotated predicate logic program

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $d_1(&lt;x,t&gt;) ← s_1(&lt;x,t&gt;) &amp; s_2(&lt;x,t&gt;)$</td>
<td>9. $s_2(&lt;John,f&gt;) ←$</td>
</tr>
<tr>
<td>2. $d_2(&lt;x,t&gt;) ← s_1(&lt;x,t&gt;) &amp; s_3(&lt;x,t&gt;)$</td>
<td>10. $s_2(&lt;Bill,f&gt;) ←$</td>
</tr>
<tr>
<td>3. $d_1(&lt;x,t&gt;) ← d_2(&lt;x,t&gt;)$</td>
<td>11. $s_3(&lt;John,t&gt;) ←$</td>
</tr>
<tr>
<td>4. $d_2(&lt;x,t&gt;) ← d_1(&lt;x,t&gt;)$</td>
<td>12. $s_3(&lt;Bill,t&gt;) ←$</td>
</tr>
<tr>
<td>5. $d_1(&lt;x,t&gt;) ← s_1(&lt;x,t&gt;) &amp; s_4(&lt;x,t&gt;)$</td>
<td>13. $s_4(&lt;John,t&gt;) ←$</td>
</tr>
<tr>
<td>6. $d_2(&lt;x,t&gt;) ← s_1(&lt;x,t&gt;) &amp; s_3(&lt;x,t&gt;)$</td>
<td>14. $s_4(&lt;Bill,f&gt;) ←$</td>
</tr>
<tr>
<td>7. $s_1(&lt;John,t&gt;) ←$</td>
<td>15. $← d_1(&lt;x,t&gt;)$</td>
</tr>
<tr>
<td>8. $s_1(&lt;Bill,f&gt;) ←$</td>
<td>16. $← d_1(&lt;x,f&gt;)$</td>
</tr>
</tbody>
</table>

**Interpretation:**

Two doctors DOC1 and DOC2 have given their rules to diagnose two diseases $d_1, d_2$ from four symptoms $s_1, s_2, s_3, s_4$.
The high level Petri net of mini expert system

Three firing sequences which reproduce the empty initial marking and fire a goal $t_{15}, t_{16}$:

$$\delta_1 = \langle t_7, t_{13}, t_5, t_{15} \rangle$$
$$\delta_2 = \langle t_7, t_{11}, t_2, t_3, t_{16} \rangle$$
$$\delta_3 = \langle t_8, t_{12}, t_6, t_3, t_{16} \rangle$$

Remark:

It is possible to perform reasoning or inference even if knowledge bases are inconsistent.
Theorem

Necessary and sufficient conditions for the goal transition satisfirability.

Let N be the high level representation of a finite horn clause logic program P. Let G be a goal clause and $t_g$ be the corresponding goal transition in N. Then the following three statements are equivalent:

1. $P \cup \neg G$ has a contradiction
2. There exists a firing sequence which reproduces the empty marking and fires the goal transition $t_g$ in N
3. N has non-negative T-invariant X such that $X(t_g) \neq \emptyset$. 
Thank you for your attention