Towards reasoning about concurrency: a logical approach

XIII Seminário Informal (, mas Formal!)

Bruno Lopes

FRAME lab.
Instituto de Computação
Universidade Federal Fluminense

January, 2016
1 Concurrency

2 Petri nets

3 A logical approach
   PDL
   Petri-PDL
   DS_3

4 Model Checker

5 Examples

6 Ongoing
Towards reasoning about concurrency

Bruno Lopes

Concurrency

Petri nets

A logical approach

Model Checker

Examples

Ongoing

References

Contact

E. Dijkstra & C. A. R. Hoare (1965)
Towards reasoning about concurrency
Bruno Lopes
Concurrent
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

E. Dijkstra & C. A. R. Hoare (1965)
Towards reasoning about concurrency
Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

E. Dijkstra & C. A. R. Hoare (1965)
1 Concurrency

2 Petri nets

3 A logical approach
   PDL
   Petri-PDL
   DS3

4 Model Checker

5 Examples

6 Ongoing
Petri nets

C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.
C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.

Elements

- Place
- Transition
- Tokens
- Edge
C. A. Petri (1939)

A bipartite graph with two types of nodes: places and transitions.

**Elements**

- Place
- Transition
- Tokens
- Edge

**Firing**

A bipartite graph with two types of nodes: places and transitions.
Petri nets

C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.

Elements

- Place
- Transition
- Tokens
- Edge

Firing

\[ p \rightarrow q \]
C. A. Petri (1939)
A bipartite graph with two types of nodes: **places** and **transitions**.

**Elements**
- Place
- Transition
- Tokens
- Edge

**Firing**

\[ p \rightarrow q \]
C. A. Petri (1939)

A bipartite graph with two types of nodes: **places** and **transitions**.

**Elements**

- **Place**
- **Transition**
- **Tokens**
- **Edge**

**Firing**
C. A. Petri (1939)

A bipartite graph with two types of nodes: **places** and **transitions**.

**Elements**

- Place
- Transition
- Tokens
- Edge

**Firing**

\[
\begin{array}{c}
\text{p} \\
\downarrow \\
\text{q}
\end{array}
\]
C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.

Elements

- Place
- Transition
- Tokens
- Edge

Firing

\[ p \rightarrow q \]
C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.

Elements

- □ Place
- □ Transition
- ● Tokens
- → Edge

Firing

- p
- q
- r
**Petri nets**

**C. A. Petri (1939)**

A bipartite graph with two types of nodes: **places** and **transitions**.

**Elements**

- Place
- Transition
- Tokens
- Edge

**Firing**

![Petri net diagram]

**C. A. Petri (1939)**

A bipartite graph with two types of nodes: **places** and **transitions**.
Petri nets

C. A. Petri (1939)
A bipartite graph with two types of nodes: places and transitions.

Elements

- **Place**
- **Transition**
- **Tokens**
- **Edge**

Firing

\[
\begin{align*}
\text{p} & \quad \text{q} \\
\text{r} & 
\end{align*}
\]
Petri nets: usage example

Modelling
Petri nets: usage example

Modelling
Once a coin is inserted in a supposed machine, “Player 1” will able to begin his game.
Petri nets: usage example

Modelling
If the user wins, a token will be placed at “Win₁” and the user will be able to play again.
Petri nets: usage example

Modelling
If he loses, a token will be placed at “Win₂” or the game restarts if there is a draw match.
Petri nets: reasoning challenges

- State explosion
- Undecidability
- Incompleteness
Petri nets model

Type 1:

\[
\begin{array}{c}
\text{X} \\
\text{Y}
\end{array}
\]

Type 2:

\[
\begin{array}{c}
\text{X} \\
\text{Y} \\
\text{Z}
\end{array}
\]

Type 3:

\[
\begin{array}{c}
\text{X} \\
\text{Y} \\
\text{Z}
\end{array}
\]

E. S. de Almeida & E. H. Haeusler (1999)
As an example...

For a Petri net with the three types of transitions
As an example...

We may decompose the Type 1 transition...
As an example...

The Type 2 transition
As an example...

And the Type 3 transition

![Petri Net Diagram](image-url)
1 Concurrency

2 Petri nets

3 A logical approach
   PDL
   Petri-PDL
   $DS_3$

4 Model Checker

5 Examples

6 Ongoing
Propositional Dynamic Logic

**PDL**

Is a multi-modal logic used for specifying and reasoning on sequential programs. It uses one modality $\langle \pi \rangle$ for each program $\pi$. 
Propositional Dynamic Logic

**PDL**

Is a multi-modal logic used for specifying and reasoning on sequential programs. It uses one modality $\langle \pi \rangle$ for each program $\pi$.

**Language**

Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$\varphi ::= p \mid T \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \pi \rangle \varphi$$

$$\pi ::= \alpha \mid \pi; \pi \mid \pi \cup \pi \mid \pi^*$$
Propositional Dynamic Logic

**PDL**

Is a multi-modal logic used for specifying and reasoning on sequential programs. It uses one modality $\langle \pi \rangle$ for each program $\pi$.

**Language**

Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$\varphi ::= p \mid T \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \pi \rangle \varphi \quad \sim \quad \text{“generator”}$$

$$\pi ::= \alpha \mid \pi ; \pi \mid \pi \cup \pi \mid \pi^*$$
Usage example

```latex
if p then
    \alpha;
    \beta;
    \textbf{while} q \textbf{do}
    \mid \beta;
end
end
```

Modelled in PDL

$p \rightarrow [\alpha; \beta] (q \rightarrow [\beta; \star] \neg q)$
Usage example

```plaintext
if p then
    α;
    β;
    while q do
        β;
    end
end

Modelled in PDL
p → [α; β](q → [β*]¬q)
```
**Petri nets**

- Native support to concurrence
- Intuitive graphical interpretation

**Propositional Dynamic Logic**

- Formal system to verify properties in programs
- Deductive systems
Petri nets

- Native support to concurrency
- Intuitive graphical interpretation

Propositional Dynamic Logic

- Formal system to verify properties in programs
- Deductive systems

Our approach
Unify these formalisms!
Petri-PDL

PDL Language

Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$\phi ::= p \mid T \mid \neg \phi \mid \phi \land \phi \mid \langle \pi \rangle \phi$$

$$\pi ::= \alpha \mid \pi ; \pi \mid \pi \cup \pi \mid \pi^*$$
Towards reasoning about concurrency
Bruno Lopes

Concurrency
Petri nets
A logical approach
Petri-PDL
Model Checker
Examples
Ongoing
References
Contact

PDL Language
Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$
\varphi ::= p \mid T \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \pi \rangle \varphi
$$

$$
\pi ::= \alpha \mid \pi; \pi \mid \pi \cup \pi \mid \pi^*
$$
Petri-PDL

Petri-PDL Language

Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$
\varphi ::= p \mid T \mid \neg \varphi \mid \varphi \land \varphi \mid \langle \pi \rangle \varphi \quad \leadsto \text{now a Petri net}
$$

$$
\pi ::= \pi \circ \pi \mid \eta
$$
Petri-PDL

PDL Language

Syntax: Let $p$ be an atomic proposition and $\alpha$ a basic program

$$
\varphi ::= p \mid \top \mid \neg \varphi \mid \varphi \land \varphi \mid \langle s, \pi \rangle \varphi \quad \leadsto \text{marked!}
$$

$$
\pi ::= \pi \odot \pi \mid \eta
$$

$s$ : a sequence of names
Basic Petri nets

Type 1: $at_1b$

Type 2: $a, bt_2c$

Type 3: $at_3b, c$
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Petri-PDL
Model Checker
Examples
Ongoing
References
Contact

Firing function

\[
T_1 \\
\begin{align*}
f(s, at_1 b) &= \begin{cases} 
  s_1 bs_2, & \text{if } s = s_1 as_2 \\
  \epsilon, & \text{if } a \not\prec s
\end{cases}
\end{align*}
\]

\[
T_3 \\
\begin{align*}
f(s, at_3 bc) &= \begin{cases} 
  s_1 s_2 bc, & \text{if } s = s_1 as_2 \\
  \epsilon, & \text{if } a \not\prec s
\end{cases}
\end{align*}
\]

\[
T_2 \\
\begin{align*}
f(s, abt_2 c) &= \begin{cases} 
  s_1 cs_2 s_3, & \text{if } s = s_1 as_2 bs_3 \\
  \epsilon, & \text{if } a, b \not\prec s
\end{cases}
\end{align*}
\]

\[
f(\epsilon, \pi) = \epsilon
\]
Towards reasoning about concurrency
Bruno Lopes

Concurrent
Petri nets
A logical
approach
Petri-PDL
Model
Checker
Examples
Ongoing
References
Contact

Instituto de Computation

Axiomatic System

(PL) Enough propositional logic tautologies

(K) \([s, \pi](p \rightarrow q) \rightarrow ([s, \pi]p \rightarrow [s, \pi]q)\)

(Du) \([s, \pi]p \leftrightarrow \neg \langle s, \pi \rangle \neg p\)

(Sub) If \(\models \varphi\), then \(\models \varphi^\sigma\), where \(\sigma\) uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If \(\models \varphi\) and \(\models \varphi \rightarrow \psi\), then \(\models \psi\).

(Gen) If \(\models \varphi\), then \(\models [s, \pi]\varphi\).

(PC) \(\langle s, \eta \rangle \varphi \leftrightarrow \langle s, \eta_1 \rangle \langle s_1, \eta \rangle \varphi \lor \langle s, \eta_2 \rangle \langle s_2, \eta \rangle \varphi \lor \cdots \lor \langle s, \eta_n \rangle \langle s_n, \eta \rangle \varphi,\)

where \(s_i = f(s, \eta_i)\), for all \(1 \leq i \leq n\)

(R_{\epsilon}) \(\langle s, \eta \rangle \varphi \leftrightarrow \varphi\), if \(f(s, \eta) = \epsilon\)
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Petri-PDL
Model Checker
Examples
Ongoing
References
Contact

Axiomatic System

(PL) Enough propositional logic tautologies

(K) \([s, \pi](p \rightarrow q) \rightarrow ([s, \pi]p \rightarrow [s, \pi]q)\)

(Du) \([s, \pi]p \leftrightarrow \neg\langle s, \pi\rangle \neg p\)

(Sub) If \(\models \varphi\), then \(\models \varphi^\sigma\), where \(\sigma\) uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If \(\models \varphi\) and \(\models \varphi \rightarrow \psi\), then \(\models \psi\).

(Gen) If \(\models \varphi\), then \(\models [s, \pi]\varphi\).

(PC) \(\langle s, \eta \rangle \varphi \leftrightarrow \langle s, \eta_1 \rangle \langle s_1, \eta \rangle \varphi \lor \langle s, \eta_2 \rangle \langle s_2, \eta \rangle \varphi \lor \cdots \lor \langle s, \eta_n \rangle \langle s_n, \eta \rangle \varphi\),

where \(s_i = f(s, \eta_i)\), for all \(1 \leq i \leq n\)

(R\(\epsilon\)) \(\langle s, \eta \rangle \varphi \leftrightarrow \varphi\), if \(f(s, \eta) = \epsilon\)
Towards reasoning about concurrency

Bruno Lopes

Concurrency

Petri nets

A logical approach

Petri-PDL

Model Checker

Examples

Ongoing

References

Contact

Towards reasoning about concurrency

Bruno Lopes

Concurrency

Petri nets

A logical approach

Petri-PDL

Model Checker

Examples

Ongoing

References

Contact

Axiomatic System

(PL) Enough propositional logic tautologies

(K) \([s, \pi](p \rightarrow q) \rightarrow ([s, \pi]p \rightarrow [s, \pi]q)\)

(Du) \([s, \pi]p \leftrightarrow \neg<s, \pi>\neg p\)

(Sub) If \(\vDash \varphi\), then \(\vDash \varphi^\sigma\), where \(\sigma\) uniformly substitutes proposition symbols by arbitrary formulas.

(MP) If \(\vDash \varphi\) and \(\vDash \varphi \rightarrow \psi\), then \(\vDash \psi\).

(Gen) If \(\vDash \varphi\), then \(\vDash [s, \pi]\varphi\).

(PC) \(\langle s, \eta \rangle \varphi \leftrightarrow \langle s, \eta_1 \rangle \langle s_1, \eta \rangle \varphi \lor \langle s, \eta_2 \rangle \langle s_2, \eta \rangle \varphi \lor \cdots \lor \langle s, \eta_n \rangle \langle s_n, \eta \rangle \varphi, \sim \) firing

where \(s_i = f(s, \eta_i)\), for all \(1 \leq i \leq n\)

(RE) \(\langle s, \eta \rangle \varphi \leftrightarrow \varphi\), if \(f(s, \eta) = \epsilon \sim\) stop
Usage example
Usage example

Petri-PDL formula:
Usage example

**Petri-PDL formula:**
\[ \langle (\text{Coin}, \text{Player}_2), \text{Coin} t_1 \text{Player}_1 \circ \land \rangle \psi. \]
Usage example

Petri-PDL formula:
\[
\langle (\text{Coin}, \text{Player}_2), \text{Coint}_1 \text{Player}_1 \rangle \\
\langle f((\text{Coin}, \text{Player}_2), \text{Coint}_1 \text{Player}_1), \text{Coint}_1 \text{Player}_1 \odot \Upsilon \rangle \psi.
\]
Petri-PDL model

Frame: \( \mathcal{F} = \langle W, R_\pi, M \rangle \)

- \( W \) is a non-empty set of states
- \( R_\pi \) is a binary relation on \( W \) for each program \( \pi \)
- \( M \) is a function \( M: W \rightarrow S \) that returns a sequence of names for each state
A labeled Natural Deduction

\[ \text{Peter-PDL} \]

\[ \pi \square_e \frac{\{ w : [s, \pi](p \to q) \}^3, \{ w : [s, \pi]p \}^2}{\frac{\{ w : [s, \pi]q \}^3, \{ w : [s, \pi]p \}^2}{\frac{\{ w : [s, \pi](p \to q) \}^3}{\frac{u : p \to q}{u : p \to e}}\frac{u : p}{u : q}}\frac{u : q}{\pi \square_i}}\frac{\pi \square_i}{\pi \square_i} \]
Anti-Prenex Normal Form

APNF
The modalities are moved inwards a formula and only applied to modal literals.
Anti-Prenex Normal Form

APNF

The modalities are moved inwards a formula and only applied to modal literals.

A formula $\chi$ is in Anti-Prenex Normal Form (APNF) if, and only if

Let $\varphi$ and $\psi$ be formula in the language of Petri-PDL.

1. $\chi$ is a modal term; or
2. $\chi$ is of the form $(\varphi \land \psi)$, $(\varphi \lor \psi)$, or $(\varphi \rightarrow \psi)$, and $\varphi$ and $\psi$ are in APNF;
3. $\chi$ is of the form $[s, \pi] \varphi$, $\varphi$ is disjunctive, and $\varphi$ is in APNF; or
4. $\chi$ is of the form $\langle s, \pi \rangle \varphi$, $\varphi$ is conjunctive, and $\varphi$ is in APNF.
Divided Separated Normal Form for Petri-PDL

**PPDL**
Separates the contexts

- formulae which are true only at the initial state
- formulae which are true in all states
Resolution based calculus

An RPG game
An *RPG* game

A player walks through scenarios, taking a key (token in $K$) in his hand (a token in $H$) to open doors.
Resolution based calculus

An RPG game
If his hand is busy (a token in $B$) he can not open the door.
Resolution based calculus

An RPG game
Is it possible that after three rounds the player has opened one door, has a free hand and still has two keys to continue?
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. $p_0$ $[I]$
2. $\neg p_0 \lor \neg[(KKKH), \pi]\neg p_1$ $[U]$
3. $\neg p_1 \lor \neg[(KKx), \pi]\neg p_2$ $[U]$
4. $\neg p_2 \lor \neg[(KKyO), \pi]\neg p_3$ $[U]$
5. $\neg p_3 \lor \neg p$ $[U]$
6. $\neg p_0 \lor [KKHO, \pi]p$ $[U]$
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \hspace{1cm} [I]
2. \( \neg p_0 \lor \neg[(KKKH), \pi] \neg p_1 \) \hspace{1cm} [U]
3. \( \neg p_1 \lor \neg[(KKx), \pi] \neg p_2 \) \hspace{1cm} [U]
4. \( \neg p_2 \lor \neg[(KKyO), \pi] \neg p_3 \) \hspace{1cm} [U]
5. \( \neg p_3 \lor \neg p \) \hspace{1cm} [U]
6. \( \neg p_0 \lor [KKHO, \pi] p \) \hspace{1cm} [U]
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \hspace{1cm} [I]
2. \( \neg p_0 \lor \neg[(KKKH), \pi] \neg p_1 \) \hspace{1cm} [U]
3. \( \neg p_1 \lor \neg[(KKx), \pi] \neg p_2 \) \hspace{1cm} [U]
4. \( \neg p_2 \lor \neg[(KKyO), \pi] \neg p_3 \) \hspace{1cm} [U]
5. \( \neg p_3 \lor \neg p \) \hspace{1cm} [U]
6. \( \neg p_0 \lor [KKHO, \pi] p \) \hspace{1cm} [U]

\[
\text{ser2} \quad D \lor \neg[s, \pi] l \\
\quad l_1 \lor \cdots \lor l_n \lor l \\
\quad \frac{D \lor \neg[s, \pi] \neg l_1 \lor \cdots \lor \neg[s, \pi] \neg l_n}{\in U}
\]
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. $p_0$  
2. $\neg p_0 \lor \neg [(KKKH), \pi] \neg p_1$ \hspace{1cm} $[I]$  
3. $\neg p_1 \lor \neg [(KKx), \pi] \neg p_2$ \hspace{1cm} $[U]$  
4.  
6. $\neg p_0 \lor [KKHO, \pi] p$ \hspace{1cm} $[U]$  
7. $\neg p_2 \lor \neg [KKyO, \pi] p$ \hspace{1cm} $[U], (ser2), (4, 5)$
An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \([I]\)
2. \( \neg p_0 \lor \neg[(KKKH), \pi] \neg p_1 \) \([U]\)
3. \( \neg p_1 \lor \neg[(KKx), \pi] \neg p_2 \) \([U]\)
6. \( \neg p_0 \lor [KKHO, \pi]p \) \([U]\)
7. \( \neg p_2 \lor \neg[KKyO, \pi]p \) \([U], (ser2), (4, 5)\)
Towards reasoning about concurrency

Bruno Lopes

Concurrency

Petri nets

A logical approach

Petri-PDL

Model Checker

Examples

Ongoing

References

Contact

Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \)  \([\mathcal{I}]\)

2. \( \neg p_0 \lor \neg [(KKKH), \pi] \neg p_1 \)  \([\mathcal{U}]\)

3. \( \neg p_1 \lor \neg [(KKx), \pi] \neg p_2 \)  \([\mathcal{U}]\)

6. \( \neg p_0 \lor [KKHO, \pi] p \)  \([\mathcal{U}]\)

7. \( \neg p_2 \lor \neg [KKyO, \pi] p \)  \([\mathcal{U}], (ser2), (4, 5)\)

comp

\[
D \lor \neg [s, \pi] l \in \mathcal{U}
\]

if \( \eta \subseteq \pi \) and for any \( \pi_b \in \pi \),

\[
D' \lor [f(s, \pi_b), \eta] l \in \mathcal{U}
\]

\[
D \lor \neg [s, \pi] \neg D' \in \mathcal{U}
\]
Resolution based calculus

An RPG game
Modelling in Petri-PDL and applying APNF and DSNF we have

1. $p_0$ \hspace{1cm} [I]
2. $\neg p_0 \lor \neg [(KKKH), \pi] \neg p_1$ \hspace{1cm} [U]
6. $\neg p_0 \lor [KKHO, \pi] p$ \hspace{1cm} [U]
8. $\neg p_1 \lor \neg [KKx, \pi] p$ \hspace{1cm} [U], (comp), (7, 3)
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \hspace{1cm} [I]
2. \( \neg p_0 \lor \neg [(KKKH), \pi] \neg p_1 \) \hspace{1cm} [U]
6. \( \neg p_0 \lor [KKHO, \pi]p \) \hspace{1cm} [U]
8. \( \neg p_1 \lor \neg [KKx, \pi]p \) \hspace{1cm} [U], (comp), (7, 3)
Resolution based calculus

An RPG game
Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \[ I \]
2. \( \neg p_0 \lor \neg [KKKH, \pi] \neg p_1 \) \[ U \]
6. \( \neg p_0 \lor [KKHO, \pi] p \) \[ U \]
8. \( \neg p_1 \lor \neg [KKx, \pi] p \) \[ U \], \((\text{comp}), (7, 3)\)

\( \text{comp} \)

if \( \eta \subseteq \pi \) and for any \( \pi_b \in \pi \),

\[ D \lor \neg [s, \pi] l \in U \]
\[ D' \lor [f(s, \pi_b), \eta] l \in U \]
\[ D \lor \neg [s, \pi] \neg D' \in U \]
Resolution based calculus

An RPG game
Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \hspace{2cm} \([I]\)
6. \( \neg p_0 \lor [KKHO, \pi]p \) \hspace{2cm} \([U]\)
9. \( \neg p_0 \lor \neg [KKKH, \pi]p \) \hspace{2cm} \([U], (\text{comp}), (8, 2)\)
Resolution based calculus

An RPG game
Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) [I]
6. \( \neg p_0 \lor [KKHO, \pi]p \) [U]
9. \( \neg p_0 \lor \neg [KKKH, \pi]p \) [U], (comp), (8, 2)
An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) [\( I \)]
6. \( \neg p_0 \lor [KKHO, \pi]p \) [\( U \)]
9. \( \neg p_0 \lor \neg [KKKH, \pi]p \) [\( U \), (comp), (8, 2)]

\[\begin{align*}
\text{ures} \quad D &\lor m \quad \in U \\
D' &\lor \neg m \quad \in U \\
\text{rames} \quad D &\lor D' \quad \in U
\end{align*}\]
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \([I]\)
10. \( \neg p_0 \) \([U],(ures),(9,6)\)
An *RPG* game

Modelling in Petri-PDL and applying APNF and DSNF we have

\[
\begin{array}{ll}
1. & p_0 \quad [I] \\
10. & \neg p_0 \quad [U], (ures), (9, 6)
\end{array}
\]
Resolution based calculus

An RPG game

Modelling in Petri-PDL and applying APNF and DSNF we have

1. \( p_0 \) \([I]\)

10. \( \neg p_0 \) \([U], (ures), (9, 6)\)

\[
\text{ires} \quad C \lor l \in I \cup U
\]
\[
C' \lor \neg l \in I
\]
\[
\therefore C \lor C' \in I
\]
Resolution based calculus

An RPG game
Modelling in Petri-PDL and applying APNF and DSNF we have

11. ⊥ [I], (ires), (10, 1)
Marked Petri nets characteristics

- There is no way to control fire rate
- There is no way to model different timings
Marked Petri nets characteristics

- There is no way to control fire rate
- There is no way to model different timings

Extend Petri net model
Stochastic Petri nets
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
An example...

Two processes: I/O bound and CPU bound
Towards reasoning about concurrency
Bruno Lopes
Concurrency
Petri nets
A logical approach
DS3
Model Checker
Examples
Ongoing
References
Contact

An example...

Two processes: I/O bound and CPU bound

\[
\begin{array}{c}
p_1 \\
\downarrow \quad \downarrow \\
T_1 \quad T_2 \\
\downarrow \\
p_2 \\
\downarrow \\
T_2 \\
\downarrow \\
p_3 \\
\downarrow \\
T_3 \\
\downarrow \\
p_4 \\
\downarrow \\
T_3 \\
\downarrow \\
p_5 \\
\end{array}
\]
An example...}

Two processes: I/O bound and CPU bound
Frame: $\mathcal{F} = \langle W, R_\pi, M, (\Pi, \Lambda), \delta \rangle$

- $W$ is a non-empty set of states
- $R_\pi$ is a binary relation on $W$ for each program $\pi$
- $M$ is a function $M : W \rightarrow S$ that returns a sequence of names for each state
- $\Pi$ a stochastic Petri net program
- $\Lambda$ a function $\Lambda : \Pi \rightarrow \mathbb{R}^+$
- $\delta$ a delay function $\delta : W \times \Pi \rightarrow \mathbb{R}^+$
Truth probability of a modality

\[ \mathcal{M}_3, w \models \langle s, \pi_b \rangle \varphi \]

\[
\Pr(\mathcal{M}_3, w \models \langle s, \pi_b \rangle \varphi \mid \delta(w, \Pi)) = \frac{\delta(w, \pi_b)}{\sum_{\pi_b \in \Pi : f(s, \pi_b) \neq \epsilon} \delta(w, \pi_b)}
\]
1 Concurrency

2 Petri nets

3 A logical approach
   PDL
   Petri-PDL
   DS_3

4 Model Checker

5 Examples

6 Ongoing
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

Rock-Paper-Scissors

\[ \Pi : \]
\[ ct_3g_1g_2 \circ g_1t_1r_1 \circ g_1t_1s_1 \circ \]
\[ g_1t_1p_1 \circ g_2t_1r_2 \circ g_2t_1s_2 \circ \]
\[ g_2t_1p_2 \circ r_1s_2t_2w_1 \circ r_1p_2t_2w_2 \circ \]
\[ r_1r_2t_2d \circ s_1r_2t_2w_2 \circ s_1s_2t_2d \circ \]
\[ s_1p_2t_2w_1 \circ p_1r_2t_2w_1 \circ \]
\[ p_1s_2t_2w_2 \circ p_1p_2t_2d. \]
Towards reasoning about concurrency
Bruno Lopes
Concurrency Petri nets
A logical approach Model Checker Examples Ongoing References Contact

Rock-Paper-Scissors

\[ \Pi: \text{Does it always have a winner?} \]
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

Instituto de Computação

Petri-PDL model checker

```
mod PETRI-PDL is
  sort Place Places BasicProg Prog Net .
  subset Place < Places .
  subset BasicProg < Prog .

  op _t1_ : Place Place -> BasicProg [prec 30] .
  op __t2__ : Place Place Place -> BasicProg [prec 30] .
  op _t3__ : Place Place Place -> BasicProg [prec 30] .
  op _+_ : Prog Prog -> Prog [assoc comm prec 40] .
  op _-_- : Places Prog -> Net .

  vars A B C : Place . var W : Places . var P : Prog .

  rl [t1] : A W , A t1 B => B W , A t1 B .

  rl [t1] : A W , A t1 B + P => B W , A t1 B + P .

endm
```
Model checking
“Rock-Paper-Scissors”

1 mod VALUATION is
2 inc PETRI–PDL–MODEL–CHECKER.
3 ops c g1 g2 s1 s2 r1 r2 p1 p2 w1 w2 d : \rightarrow \text{Place}.
4 ops p q : \rightarrow \text{Formula}.
5 eq valuation(w1) = p. eq valuation(w2) = q. eq valuation(d) = ((\neg p)(\neg q)).
6 endm

1 reduce in VALUATION : modelCheck(\neg < c,(g1 t1 r1 + g1 t1 p1 + g1 t1 s1 +
2 g2 t1 r2 +
3 g2 t1 p2 + g2 t1 s2 + ((((((((
4 s1 s2 t2 d + s1 p2 t2 w1) + s1 r2 t2 w2) + p1 s2 t2 w2) + p1 p2 t2 d) +
5 p1 r2 t2 w1) + r1 s2 t2 w1) + r1 p2 t2 w2) + r1 r2 t2 d) + c t3 g1 g2 > (\neg (p \lor q)), 4, mt–placeslistset).
2 rewrites: 1139 in 24ms cpu (25ms real) (45942 rewrites/second)
3 result PPDLModel: ppdlModel(false, c \rightarrow g1 g2 \rightarrow g1 s2 \rightarrow s1 s2 \rightarrow d)
Model checking

“Rock-Paper-Scissors”

```
mod VALUATION is
  ops c g1 g2 s1 s2 r1 r2 p1 p2 w1 w2 d : --> Place .
  ops p q : --> Formula .
  eq valuation(w1) = p . eq valuation(w2) = q . eq valuation(d) = ((¬ p) (¬ q) ) .
endm

reduce in VALUATION : modelCheck(¬ < c,(g1 t1 r1 + g1 t1 p1 + g1 t1 s1 +
g2 t1 r2 +
g2 t1 p2 + g2 t1 s2 + ((((((s1 s2 t2 d + s1 p2 t2 w1) + s1 r2 t2 w2) + p1 s2 t2 w2) + p1 p2 t2 d) +
p1 r2 t2 w1) + r1 s2 t2 w1) + r1 p2
t2 w2) + r1 r2 t2 d) + c t3 g1 g2 > (¬ (p ∨ q)), 4, mt–placeslistset) .
rewrites: 1139 in 24ms cpu (25ms real) (45942 rewrites/second)
result PPDLModel: ppdlModel(false, c --> g1 g2 --> g1 s2 --> s1 s2 --> d)
```

No!
Gives a counterexample
1 Concurrency

2 Petri nets

3 A logical approach
   PDL
   Petri-PDL
   $DS_3$

4 Model Checker

5 Examples

6 Ongoing
Example: a SPN model for agents
Example: a multi-agent scenario

Scenario
Example: a multi-agent scenario

Scenario

$A_1$, $A_2$, $A_3$ and $A_4$ are agents that must collect and process some data from the resource centre $r$. 
Example: a multi-agent scenario

Scenario

$A_1$ and $A_2$ can not make the full process and needs that $A_3$ or $A_4$ completes the computation.
Example: a multi-agent scenario

Scenario

$A_3$ and $A_4$ have a faster processor than $A_1$ and $A_2$. 
Example: a multi-agent scenario

Scenario

$A_1$ and $A_2$ are in a shared memory system, but the clock of the processor of $A_1$ is faster then $A_2$. 
Example: a multi-agent scenario

Formalizing
Controlling the clock difference:
Example: a multi-agent scenario

Formalizing
Controlling the clock difference: set adequate values to $\lambda$. 
Example: a multi-agent scenario

Formalizing
$\mathcal{D}S_3$ formula:
Example: a multi-agent scenario

Formalizing

$DS_3$ formula: $\langle \{rrrrm\}, rmt_2 A_1 \circ rmt_2 A_2 \circ rt_1 A_3 \circ rt_1 A_4 \circ A_1 t_3 A_3 m \circ A_2 t_3 A_3 m \circ A_1 t_3 A_4 m \circ A_4 t_3 A_4 m \rangle p.$
Example: a multi-agent scenario

Formalizing
Can $A_1$ and $A_2$ compute some data in parallel?
Example: a multi-agent scenario

Formalizing

Can $A_1$ and $A_2$ compute some data in parallel? Look at the result of the firing function.
Example: a multi-agent scenario

Formalizing

From a world \( w \) is it possible that \( A_1 \) collect some data to process?
Example: a multi-agent scenario

Formalizing

From a world $w$ is it possible that $A_1$ collect some data to process? Compute

$$\text{Pr}(\mathcal{M}, w \models \langle s, \text{rmt}_2 A_1 \rangle \top | \delta(w, \text{rmt}_2 A_1 \odot \text{rmt}_2 A_2 \odot \text{rt}_1 A_3 \odot \text{rt}_1 A_4 \odot A_1 t_3 A_3 m \odot A_2 t_3 A_3 m \odot A_1 t_3 A_4 m \odot A_4 t_3 A_4 m)),$$
Example: a multi-agent scenario

Formalizing

From a world $w$ is it possible that $A_1$ collect some data to process? Compute $\frac{\delta(w, rmt_2 A_1)}{\sum \delta(w, \pi_b)}$.

\[ \pi_b \in \Pi : f(s, \pi_b) \neq \epsilon \]
Example: a multi-agent scenario

Formalizing

Are agents $A_1$ and $A_2$ overheading agents $A_3$ and $A_4$?
Example: a multi-agent scenario

Formalizing
Are agents $A_1$ and $A_2$ overheading agents $A_3$ and $A_4$? Verify if $\sum \delta(v_i, A_1 t_1 A_3 \odot A_1 t_1 A_4 \odot A_2 t_1 A_3 \odot A_2 t_1 A_4)^1 > \sum \delta(v_i, rt_1 A_3 \odot rt_1 A_4)^1$. 

\[ 1 \]
Multi-agent Environment for Reasoning in Logic and Inferring Numerically (MERLIN)

1. a1 ← agent()
2. a2 ← agent()
3. a3 ← agent()
4. a4 ← agent()
5. setDataCenters(1)
6. send(a1, a3)
7. send(a1, a4)
8. send(a2, a3)
9. send(a2, a4)
10. collect(a1, freq=.5, shared=1)
11. collect(a2, freq=.5, shared=1)
12. collect(a3, freq=1)
13. collect(a4, freq=1)

http://github.com/blopesvieira/Merlin
Multi-agent Environment for Reasoning in Logic and Inferring Numerically (MERLIN)

```
1 > setResource(a1, 1)
2 > prToSend("a1", "a3")
3 [1] 0.4
```
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

1 Concurrency

2 Petri nets

3 A logical approach
PDL
Petri-PDL
DS₃

4 Model Checker

5 Examples

6 Ongoing
Towards reasoning about concurrency

Bruno Lopes

Concurrency
Petri nets
A logical approach
Model Checker
Examples
Ongoing
References
Contact

Ongoing...

- Automatic theorem prover
- Model checking framework for Dynamic Logics
- Studies on the computational complexity of the logics
Special thanks
Profa. Daniele Nantes
References I


References II

Bruno Lopes, Mario Benevides, and Edward Hermann Haeusler.
Propositional dynamic logic for Petri nets.

Bruno Lopes, Mario Benevides, and Edward Hermann Haeusler.
Reasoning about Multi-Agent Systems Using Stochastic Petri Nets.
Cláudia Nalon, Bruno Lopes, Edward Hermann Haeusler, and Gilles Dowek.
A calculus for automatic verification of Petri Nets based on Resolution and Dynamic Logics.
Contact information

Bruno Lopes (bruno@ic.uff.br)
http://www.ic.uff.br/~bruno
Instituto de Computação
Universidade Federal Fluminense
Niterói - RJ, Brazil