Efficient Unfolding of Coloured Petri Nets using Interval Decision Diagrams

Martin Schwarick, Christian Rohr, Fei Liu, George Assaf, Jacek Chodak and Monika Heiner

Brandenburg Technical University
Petri Nets 2020 - Paris

25 June 2020
Outline

1. Why coloured Petri Nets?
2. State of the Art
3. The Problem
4. What are IDD?
5. IDD basic principles
6. IDD-based unfolding algorithm
7. Experiments
Coloured PNs - Powerful modelling

Color definitions:

\[
\text{\textbf{Simple CS}}
\]

\[
\text{enum Nodes = \{}\text{A, B, C, D, F, G, H, I, J} \text{;}
\]

\[
\text{\textbf{Product CS}}
\]

\[
\text{Matrix = Prod(Nodes,Nodes);}
\]

\[
\text{\textbf{Subset CS}}
\]

\[
\text{Connections = Matrix[(a = A \& (b = B \| b = C \| b = D \| b = G \| b = I)) \| (a = B \& (b = A \| b = G \| b = H \| b = J)) \| (a = C \& (b = F \| b = H))];}
\]

variables:

Nodes : a;
Nodes : b;

functions:

bool IsConnected(Node a1, Node b1)

\{(a1, b1) elemOf Connections\};
Coloured PNs - Powerful modelling

Color definitions:

```plaintext
/// Simple CS
enum Nodes = {A,B,C,D,F,G,H,I,J};

/// Product CS
Matrix = Prod(Nodes,Nodes);

/// Subset CS
Connections = Matrix[(a=A & (b=B | b=C | b=D | b=G | b=I)) | (a=B & (b=A | b=G | b=H | b=J)) | (a=C & (b=F | b=H))];
```

variables:
Nodes : a;
Nodes : b;

functions:
bool isConnected(Node a1, Node b1)
{(a1,b1) elemOf Connections};
Coloured PNs - Powerful modelling

Color definitions:

```plaintext
\begin{itemize}
  \item Simple CS
    \begin{verbatim}
    enum Nodes = {A,B,C,D,F,G,H,I,J};
    \end{verbatim}
  \item Product CS
    \begin{verbatim}
    Matrix = Prod(Nodes,Nodes);
    \end{verbatim}
  \item Subset CS
    \begin{verbatim}
    Connections = Matrix[(a=A & (b=B | b=C | b=D | b=G | b=I)) | (a=B & (b=A | b=G | b=H | b=J)) | (a=C & (b=F | b=H))];
    \end{verbatim}
\end{itemize}

variables:
Nodes : a;
Nodes : b;
functions:
bool IsConnected(Node a1, Node b1)
{(a1,b1) elemOf Connections};
```
Coloured PNs - Powerful modelling

Color definitions:

\[\text{\texttt{\textcolor{red}{\textbackslash\textbackslash Simple CS}}}\]
\[\text{\texttt{\textcolor{blue}{\texttt{enum Nodes = \{A,B,C,D,F,G,H,I,J \};}}}}\]
\[\text{\texttt{\textcolor{red}{\textbackslash\textbackslash Product CS}}}\]
\[\text{\texttt{\textcolor{blue}{\texttt{Matrix = Prod(Nodes,Nodes);}}}}\]
\[\text{\texttt{\textcolor{red}{\textbackslash\textbackslash Subset CS}}}\]
\[\text{\texttt{\textcolor{blue}{\texttt{Connections = Matrix[(a=A \& (b=B | b=C | b=D | b=G | b=I)) | (a=B \& (b=A | b=G | b=H | b=J)) | (a=C \& (b=F | b=H))];}}}}\]

\[\text{\texttt{\textcolor{red}{\texttt{variables:}}}}\]
\[\text{\texttt{Nodes : a;}}\]
\[\text{\texttt{Nodes : b;}}\]

\[\text{\texttt{\textcolor{red}{\texttt{functions:}}}}\]
\[\text{\texttt{\textcolor{blue}{\texttt{bool IsConnected(Node a1, Node b1)}}}}\]
\[\text{\texttt{\{ (a1,b1) elemOf Connections \};}}\]
Coloured PNs - Powerful modelling

- Colour definitions:
  - Simple CS
    ```
    enum Nodes = {A,B,C,D,F,G,H,I,J};
    ```
  - Product CS
    ```
    Matrix = Prod(Nodes, Nodes);
    ```
  - Subset CS
    ```
    Connections = Matrix[(a=A & b=B | b=C | b=D | b=G | b=I) | (a=B & b=A | b=G | b=H | b=J) | (a=C & b=F | b=H)];
    ```

- Variables:
  - Nodes : a;
  - Nodes : b;

- Functions:
  ```
  bool IsConnected(Node a1, Node b1) {
  (a1,b1) elemOf Connections;
  ```
Powerful modelling - Coloured Petri nets

coloured continuous Petri net

[IsNeighbour_4(x, y, a, b)]
(x, y)

100*(x=MID & y=MID)

Grid2D
diff

(a, b)

coloured stochastic Petri net

Edibles
4`b++
2`m++
2`p

y=t1]2`p++
y=t2]1`p++
3`b++
2`m++
y=t3]1`m

Guard

Guard

8

preps

[y=t1]1`ps++
y=t2]1`fs++
y=t3]1`ms
Coloured Petri nets are in use for a wide range of applications, covering natural/engineering/life sciences.
State of the Art

- Coloured Petri nets are in use for a wide range of applications, covering natural/engineering/life sciences.
- Currently, most analysis and simulation techniques require unfolding: coloured Petri net $\rightarrow$ plain Petri net.
Coloured Petri nets are in use for a wide range of applications, covering natural/engineering/life sciences.

Currently, most analysis and simulation techniques require unfolding: coloured Petri net $\rightarrow$ plain Petri net.

Unfolding tends to be time consuming.
State of the Art

- Example of a scaleable model to adjust grid size.
State of the Art

- Example of a scaleable model to adjust grid size.
- 2D Diffusion in space.
The core problem of efficient unfolding is to determine the transition instances, e.g., all bindings of the involved variables.
The core problem of efficient unfolding is to determine the transition instances, e.g., all bindings of the involved variables.
The core problem of efficient unfolding is to determine the transition instances, e.g., all bindings of the involved variables.

\[ \text{CSP} \]

Range of Variables \( x, y, a \) and \( b \)

\( 1 \ldots D \)
Interval decision diagrams $\rightarrow$ CSP.
What are IDD?

- Directed acyclic graphs (DAGs) to encode interval logic functions in the form of **symbolic data structure**.
What are IDD?

- They have two types of nodes: non-terminal (ellipses) and terminal ones (boxes).
What are IDD?

- They have two types of nodes: non-terminal (ellipses) and terminal ones (boxes).
What are IDD?

- Non-terminal nodes may have an arbitrary number of outgoing arcs labelled with intervals of natural numbers in the form \([a,b)\).
The set of all paths going from: the root $\rightarrow$ the terminal node 1 describes all solutions of the given constraint problem.
The set of all paths going from: the root $\rightarrow$ the terminal node 1 describes all solutions of the given constraint problem.

Typically, one path encodes more than one solution.
IDD basic algorithm

- The set of all paths going from: the root → the terminal node 1 describes all solutions of the given constraint problem.
- Typically, one path encodes more than one solution.
- Thus, we can easily pick all CSP solutions from the constraint IDD.
Example: \((x_1 \geq 8) \lor (x_1 \in [6, 8) \land x_2 > 0)\)

- Variable ordering

\[ x_1 \]

\[ x_2 \]

\[ x_2 \]

\[ 0 \]

\[ 1 \]
Example \((x_1 \geq 8) \lor (x_1 \in [6, 8) \land x_2 > 0)\)

- \(x_1 \geq 8\).
Example $(x_1 \geq 8) \lor (x_1 \in [6,8) \land x_2 > 0)$

- some intermediate screenshots.
Example \((x_1 \geq 8) \lor (x_1 \in [6, 8) \land x_2 > 0)\)

- One final solution.
Reducing IDD

- Interval partitions labelling the outgoing arcs of each non-terminal node are reduced. For example, $[6,7)$ and $[7,8]$
Interval partitions labelling the outgoing arcs of each non-terminal node are reduced. For example, [6, 7) and [7, 8]

Each non-terminal node has at least two different children.
Reducing IDD

- Interval partitions labelling the outgoing arcs of each non-terminal node are reduced. For example, [6, 7) and [7, 8]
- Each non-terminal node has at least two different children.
- There exist no two nodes with isomorphic subgraphs.

Not Reduced

Reduced
**Preparation step:** registration of constants, color sets, variables and functions.
IDD-based unfolding algorithm

- **Preparation step**: registration of constants, color sets, variables and functions.

- **Unfold places and determine initial marking**: this may involve CSP if we have subset mechanism and to determine initial marking.
IDD-based unfolding algorithm

- **Preparation step**: registration of constants, color sets, variables and functions.
- **Unfold places and determine initial marking**: this may involve CSP if we have subset mechanism and to determine initial marking.
- **Unfold transitions**: and their adjacent arcs and add the result to the unfolded net.
IDD-based unfolding algorithm

- **Preparation step**: registration of constants, color sets, variables and functions.

- **Unfold places and determine initial marking**: this may involve CSP if we have subset mechanism and to determine initial marking.

- **Unfold transitions**: and their adjacent arcs and add the result to the unfolded net.

- **Add all unfolded places** which are involved in the transition unfoldings to the unfolded net. This implicitly removes isolated places.
IDD-based unfolding algorithm

- **Preparation step**: registration of constants, color sets, variables and functions.
- **Unfold places and determine initial marking**: this may involve CSP if we have subset mechanism and to determine initial marking.
- **Unfold transitions**: and their adjacent arcs and add the result to the unfolded net.
- **Add all unfolded places** which are involved in the transition unfoldings to the unfolded net. This implicitly removes isolated places.
- **The entire pseudo code of the algorithm is given in the paper.**
Color definitions:

\[ \text{cs} = \{1,8,3..6,10,9,11,20..23\}; \]
\[ \text{enum ab} = \{A,C,D\}; \]
\[ \text{variables:} \]
\[ \text{cs} : x; \]
\[ \text{ab} : y; \]
Example (no constraints)

Color definitions:

\[ cs = \{1,8,3..6,10,9,11,20..23\}; \]
\[ \text{enum } ab = \{A,C,D\}; \]
variables:
\[ cs : x; \]
\[ ab : y; \]
Example (with Guard)

Color definitions:

```plaintext
cs = \{1,8,3..6,10,9,11,20..23\};
enum ab = \{A,C,D\};
variables:
  cs : x;
  ab : y;
```

Schwarick, Rohr, Liu, Assaf, Chodak, Heiner

25 June 2020 19 / 37
Encoding the entire $\mathbf{cs} = \{1, 8, 3..6, 10, 9, 11, 20..23\}$
Constraining the color set $cs$ to $6 \leq x$
Constraining the color set $cs$ to $x \leq 10$
Combining $6 \leq x$ and $x \leq 10$ using & operator.
Encoding the entire color set $ab = \{A, C, D\}$
Constraining the color set \( ab \) to \( y = A \)
Merging the result of \((6 \leq x \text{ and } x \leq 10)\) and \((y = A)\) using & operator

Two-path solution:

1st path: \((y = A, x = 6)\)
2nd path: \((y = A, x = 8 \ldots 10)\)
We compared our IDD unfolding with an unfolding employing the popular constraint solver library Gecode.
We compared our IDD unfolding with an unfolding employing the popular constraint solver library Gecode.

22 MCC models (PNML format) → https://mcc.lip6.fr/models.php
- 1st group: requires no substantial unfolding time.
- 2nd group: requires substantial unfolding time.
Experiments

- We compared our IDD unfolding with an unfolding employing the popular constraint solver library Gecode.

  - 1st group: requires no substantial unfolding time.
  - 2nd group: requires substantial unfolding time.

- We used also two biological test cases from our own collection:
  https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Examples?dir=lddUnfolding
  - 3D Diffusion.
  - Brusselator.
the model

- a lane bridge with limited capacity.
- used by two types of vehicles.
- coloured model has 15 places, 11 transition and 57 arcs.
Figure: Bridges and vehicles (MCC); requires no substantial unfolding time
Family reunion (MCC)

the model

- reunification process.
- the coloured model has 104 places, 66 transition and 198 arcs.
- it is scaled by the number of legal residents.
**Figure**: Family Reunion (MCC); requires substantial unfolding time
Diffusion in space (3D)

\[ \text{IsNeighbour}(x, y, z, a, b, c) \]

\( (x, y, z) \)

\( (a, b, c) \)

\( 10 \cdot (\text{MID, MID, MID}) \)

- diffusion
- Grid3D
- pos
### Diffusion in space (3D)

#### Figure: Diffusion (3D); N – Grid size

<table>
<thead>
<tr>
<th>N</th>
<th>P</th>
<th>T</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>125</td>
<td>600</td>
<td>1200</td>
</tr>
<tr>
<td>10</td>
<td>1000</td>
<td>5400</td>
<td>10800</td>
</tr>
<tr>
<td>15</td>
<td>3375</td>
<td>18900</td>
<td>37800</td>
</tr>
<tr>
<td>20</td>
<td>8000</td>
<td>72998</td>
<td>145996</td>
</tr>
<tr>
<td>25</td>
<td>15625</td>
<td>90000</td>
<td>180000</td>
</tr>
<tr>
<td>30</td>
<td>27000</td>
<td>156600</td>
<td>313200</td>
</tr>
<tr>
<td>35</td>
<td>42875</td>
<td>249900</td>
<td>499800</td>
</tr>
<tr>
<td>40</td>
<td>64000</td>
<td>374400</td>
<td>748800</td>
</tr>
<tr>
<td>45</td>
<td>91125</td>
<td>534600</td>
<td>1069200</td>
</tr>
<tr>
<td>50</td>
<td>125000</td>
<td>735000</td>
<td>1470000</td>
</tr>
</tbody>
</table>

Schwarick, Rohr, Liu, Assaf, Chodak, Heiner
Figure: Brusselator; $N$ – Grid size of a 2D square

<table>
<thead>
<tr>
<th>$N$</th>
<th>$P$</th>
<th>$T$</th>
<th>$A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1250</td>
<td>11 908</td>
<td>23 191</td>
</tr>
<tr>
<td>50</td>
<td>5000</td>
<td>48 808</td>
<td>95 116</td>
</tr>
<tr>
<td>75</td>
<td>11 250</td>
<td>110 708</td>
<td>215 791</td>
</tr>
<tr>
<td>100</td>
<td>20 000</td>
<td>197 608</td>
<td>385 216</td>
</tr>
<tr>
<td>125</td>
<td>31 250</td>
<td>309 508</td>
<td>603 391</td>
</tr>
<tr>
<td>150</td>
<td>45 000</td>
<td>446 408</td>
<td>870 316</td>
</tr>
<tr>
<td>175</td>
<td>61 250</td>
<td>608 308</td>
<td>1 185 991</td>
</tr>
<tr>
<td>200</td>
<td>80 000</td>
<td>795 208</td>
<td>1 550 416</td>
</tr>
<tr>
<td>225</td>
<td>101 250</td>
<td>1 007 108</td>
<td>1 963 591</td>
</tr>
<tr>
<td>250</td>
<td>125 000</td>
<td>1 244 008</td>
<td>2 425 516</td>
</tr>
</tbody>
</table>
THE WINNER IS...
And the winner is . . .

- Gecode, when:
  - models with a few guards or no guards.
  - models with simple colour sets.
  - e.g, 12 MCC models (no substantial time).
And the winner is . . .

- **Gecode**, when:
  - models with *a few guards or no guards*.
  - models with simple colour sets.
  - e.g, 12 MCC models (*no substantial time*).

- **IDDs**, when:
  - parametrized models with a large scaling factor, e.g, Diffusion in space.
  - models with sophisticated guards.
  - e.g, most models in our own collection.

The complete performance report is available: https://www-dssz.infotu-cottbus.de/DSSZ/Software/Examples?dir=IddUnfolding
And the winner is . . .

- **Gecode**, when:
  - models with *a few guards or no guards*.
  - models with simple colour sets.
  - e.g., 12 MCC models (*no substantial time*).

- **IDDs**, when:
  - parametrized models with a large scaling factor, e.g., Diffusion in space.
  - *models with sophisticated guards*.
  - e.g., most models in our own collection.

- **The complete performance report is available:**
  
Our tools

SNOOPY

MARCIE       SPIKE

https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/
Our tools

SNOOPY

CANDL \rightarrow \text{MARCE}

CANDL

\text{PNML}

\text{SPIKE}

https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/
Future work

- Performance:
  - Considering memory.
  - Power consumption.
Future work

- **Performance:**
  - Considering memory.
  - Power consumption.

- **Implementation efficiency:**
  - Multi-threading: unfolding the coloured places and transition is currently done sequentially.
  - Reuse of already computed solutions.
  - Choosing among several variable order strategies.
Thank You For Your Attention