Exploitation de la Hiérarchie et des Symétries dans les systèmes répartis

Fabrice Kordon
Distributed Systems...

Very «hot topic»

- Everywhere + increasing needs
- How to develop reliable ones
- Behavioral aspects

Characteristics
Distributed Systems...

Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

Characteristics
Distributed Systems...

Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

Characteristics
- Interoperability
Distributed Systems...

Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

Characteristics
- Interoperability
- Symmetries
Distributed Systems...

Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

 Characteristics
- Interoperability
- Symmetries
- Dynamicity
Distributed Systems...

Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

Characteristics
- Interoperability
- Symmetries
- Dynamicity
- Structuration
- Hierarchy (systems of systems)
Very «hot topic»
- Everywhere + increasing needs
- How to develop reliable ones
  - Behavioral aspects

Characteristics
- Interoperability
- Symmetries
- Dynamicity
- Structuration
- Hierarchy (systems of systems)

How to map such characteristics to improve formal verification!

Context: Model Checking
Interoperability
- Asynchronous communications => Complexity

Dynamicy
- Bound problem => Even more complexity
- Need to compute an upper bound

Symmetries
- Possibility to «repeat» parts of the system
- Permutability of some system elements

Structuration
- Locality in the system’s evolution
- Share common parts of states

Hierarchy
- Better organization
- Identification of state space patterns
Interoperability
- Asynchronous communications => Complexity

Dynamicty
- Bound problem => Even more complexity
  Need to compute an upper bound

Symmetries
- Possibility to «repeat» parts of the system
- Permutability

Structuration
- Locality in the system's evolution
- Share common parts of states

Hierarchy
- Better organization
- Identification of state space patterns

Enemies of model checking:
1) memory
2) CPU
Idea 1

You Need Hierarchy

[Hong 2012]
Based on the notion of locality [Bryant 1986] + [Clarke 1992]

- State = vector of integers
- Share of common parts in a set of states

Sequence of variable affectations

- Accepting sequence: \( x \leftarrow a, y \leftarrow 1 \) and \( x \leftarrow a, y \leftarrow 2 \)

Exponential gain in favorable cases

- Order of the variable encoding is crucial
Based on the notion of locality [Bryant 1986] + [Clarke 1992]

- State = vector of integers
- Share of common parts in a set of states

Sequence of variable affectations
- accepting sequence: \( x \leftarrow a, y \leftarrow 1 \) and \( x \leftarrow a, y \leftarrow 2 \)

Exponential gain in favorable cases
- Order of the variable encoding is crucial
Symbolic Encoding of the State Space with decision Diagrams

Based on the notion of locality [Bryant 1986] + [Clarke 1992]
- State = vector of integers
- Share of common parts in a set of states

Sequence of variable affectations
- Accepting sequence: \( x \leftarrow a, y \leftarrow 1 \) and \( x \leftarrow a, y \leftarrow 2 \)

\[
\left\{ \begin{align*}
 x \leftarrow a, y \leftarrow 1 \\
 x \leftarrow a, y \leftarrow 2
\end{align*} \right. 
\]

Exponential gain in favorable cases
- Order of the variable encoding is crucial
Symbolic Encoding of the State Space with decision Diagrams

Based on the notion of locality [Bryant 1986] + [Clarke 1992]
- State = vector of integers
- Share of common parts in a set of states

Sequence of variable affectations
- Accepting sequence: \( x \leftarrow a, y \leftarrow 1 \) and \( x \leftarrow a, y \leftarrow 2 \)

\[
\begin{align*}
\{ x \leftarrow a, y \leftarrow 1 \} \\
\{ x \leftarrow a, y \leftarrow 2 \}
\end{align*}
\]

Exponential gain in favorable cases
- Order of the variable encoding is crucial
Hierarchical Decision Diagrams [Couvreur 2005]

- Arcs labeled by sets
- A decision diagram represents a set
- Recursivity

Example

Structured data: \(<p1,\{1,3\}> + <p2,\{1,3\}>\)
Hierarchical Decision Diagrams [Couvreur 2005]

- Arcs labeled by sets
- A decision diagram represents a set
- Recursivity

Example

Structured data: \(<p1,\{1,3\}> + <p2,\{1,3\}>

\[
\text{inCS} \rightarrow \{p1, p2\} \rightarrow 1
\]
Hierarchical Decision Diagrams [Couvreur 2005]

- Arcs labeled by sets
- A decision diagram represents a set
- Recursivity

Example

Structured data: \(<p_1, \{1,3\}> + <p_2, \{1,3\}>\)
Hierarchical Decision Diagrams [Couvreur 2005]

- Arcs labeled by sets
- A decision diagram represents a set
- Recursivity

Example

Structured data: \(<p1,\{1,3\}> + <p2,\{1,3\}>

\[\text{inCS} \quad \{p1, p2\} \rightarrow 1\]

Can be reused to encode another part of the system
More required to have hierarchy

Hierarchical Decision Diagrams [Couvreur 2005]

Arcs labeled by sets

A decision diagram represents a set

Recursivity

Example

Structured data: \( \{p1, \{1, 3\}\} + \{p2, \{1, 3\}\} \)

Can be reused to encode another part of the system

Representation of a freeway [Bérard 2008]

\[
\begin{align*}
\text{CONTROL} & \quad \text{LANE} & \quad \text{POS} & \quad \text{SPEED} \\
0 & \quad 0..1 & \quad 0..10 & \quad 0..4 \\
\text{CONTROL} & \quad \text{del0} & \quad \text{veh1} & \quad \text{veh2} & \quad \text{veh3} \\
0..2 & \quad 1 & \quad 0..2 & \quad 0..1 \\
\text{delays} & \quad \text{del0} & \quad \text{veh1} & \quad \text{veh2} & \quad \text{veh3} \\
\end{align*}
\]
More required to have hierarchy

Hierarchical Decision Diagrams [Couvreur 2005]

Arcs labeled by sets

A decision diagram represents a set

Recursivity

Example

Structured data: \(<\text{p1,}{\{1,3\}}>+\text{<p2,}{\{1,3\}}>\)

Can be reused to encode another part of the system

Representation of a freeway [Bérard 2008]

Perfomances (regular system)

Performances (regular system)
Objective: exploit the structure of large Petri Nets
- Obtained from «unfolding of Colored Nets»
- Characteristics: large models with repeated patterns

Involved techniques
- Compute a variable order
  - FORCE [Haloul 2003] or NOA99 [Heiner 2009]
- Hierarchically cluster this order

Anonymize the clusters as much as possible
- Anonymization = Ccontextual interpretation
- Reused patterns contextually
Anonymization...
Anonymization...

Applies on places $P_1, P_2$

Applies on places $P_3, P_4, P_5$

Level 1

Level 2
Comparing to random order

CPU Time

Time (s)
PNXDD, performances

Comparing to random order

Memory

Memory (MB)

Nodes

CPU Time

Nodes

Time (s)

Memory (MB)
### PolyORB case Study (1 state = up to 2KB)

<table>
<thead>
<tr>
<th>number of Instances</th>
<th>State Space Size</th>
<th>Flat</th>
<th>Flat order performance</th>
<th>Hierarchical order performance</th>
<th>Gain (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Final</td>
<td>Peak</td>
<td>Time</td>
</tr>
<tr>
<td>2</td>
<td>1.6×10^6</td>
<td>F</td>
<td>223,243</td>
<td>3.1×10^6</td>
<td>580.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>78,785</td>
<td>451,494</td>
<td>98.7</td>
</tr>
<tr>
<td>3</td>
<td>2.8×10^7</td>
<td>F</td>
<td>593,363</td>
<td>1.2×10^7</td>
<td>4,708</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>280,068</td>
<td>2.5×10^6</td>
<td>948.8</td>
</tr>
<tr>
<td>4</td>
<td>2.1×10^8</td>
<td>F</td>
<td>1.2×10^8</td>
<td>3.1×10^7</td>
<td>8,310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>666,886</td>
<td>8.4×10^6</td>
<td>217,173</td>
</tr>
<tr>
<td>5</td>
<td>1.4×10^9</td>
<td>F</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
<tr>
<td>6</td>
<td>9.2×10^9</td>
<td>F</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>F</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
<td>TOF</td>
<td>TOF</td>
</tr>
</tbody>
</table>
PolyORB case Study (1 state = up to 2KB)

<table>
<thead>
<tr>
<th>number of Instances</th>
<th>State Space Size</th>
<th>Flat</th>
<th>Hierarchical order performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Final</td>
<td>Peak</td>
</tr>
<tr>
<td>2</td>
<td>1.6×10^6</td>
<td>F</td>
<td>223,243</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>78,785</td>
</tr>
<tr>
<td>3</td>
<td>2.8×10^7</td>
<td>F</td>
<td>593,363</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>280,068</td>
</tr>
<tr>
<td>4</td>
<td>2.1×10^8</td>
<td>F</td>
<td>1.2×10^6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>666,886</td>
</tr>
<tr>
<td>5</td>
<td>1.4×10^9</td>
<td>F</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
</tr>
<tr>
<td>6</td>
<td>3.2×10^9</td>
<td>F</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>F</td>
<td>TOF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>TOF</td>
</tr>
</tbody>
</table>
### NEOPPOD case Study (1 state = up to 2.4KB)

<table>
<thead>
<tr>
<th>number of Instances</th>
<th>State Space Size</th>
<th>Flat</th>
<th>Flat order performance</th>
<th>Hierarchical order performance</th>
<th>Gain (in %)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Final</td>
<td>Peak</td>
<td>Time</td>
</tr>
<tr>
<td>2</td>
<td>194</td>
<td>F</td>
<td>202</td>
<td>679</td>
<td>0.024</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>463</td>
<td>1,688</td>
<td>0.024</td>
</tr>
<tr>
<td>3</td>
<td>90,861</td>
<td>F</td>
<td>5,956</td>
<td>48,269</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>3,820</td>
<td>23,974</td>
<td>0.76</td>
</tr>
<tr>
<td>4</td>
<td>9.7×10^8</td>
<td>F</td>
<td>84,398</td>
<td>1.0×10^6</td>
<td>62.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>155,759</td>
<td>1.3×10^6</td>
<td>186.1</td>
</tr>
</tbody>
</table>
Idea 2

From Model Hierarchy to Hierarchical Representation with Symmetric Nets with Bags

[Colange 2011, Colange 2012]
Capturing Symmetries using Symmetric Nets with Bags

Symmetric Nets with Bags [Haddad 2009]
- Extension of Symmetric Nets

Symmetric nets = Colored Petri Nets + Constraints
- Discrete types only (enumerations, integer range)
  - Limited operators: ++, --, =, ≠, <, >, ≤, ≥
- Possible definition of equivalence classes
  - Automatically computed [Thierry-Mieg et. al. 2003]
- Cartesian product (record-like structures)
- Equivalence classes

Symmetric Nets with Bags
- Bag (multiset) of tokens based on discrete types
  - New operators for bag variables
- Cartesian product can embed bag-types
Global allocation of resources (deadlock avoidance)
Global allocation of resources (deadlock avoidance)

Class
   Proc is [p1, p2, p3];
   Res is 1..6;

Domain
   BagR is Rag(Res);
   P_BarG is <Poc, BagR>;

Var
   p in Proc;
   R, R2 in BagR;
Symmetric Nets with Bags, an Example

Global allocation of resources (deadlock avoidance)

Class
  Proc is [p1, p2, p3];
  Res is 1..6;

Domain
  BagR is Rag(Res);
  P_BarG is <Poc, BagR>;

Var
  p in Proc;
  R, R2 in BagR;
Symmetric Nets with Bags: An Example

Global allocation of resources (deadlock avoidance)

Class
Proc is [p1, p2, p3];
Res is 1..6;

Domain
BagR is Rag(Res);
P_BarG is <Poc, BagR>;

Var
p in Proc;
R, R2 in BagR;

InCS Proc P_BagR
OutCS
Symmetric Nets with Bags, an Example

Global allocation of resources (deadlock avoidance)

Class
  Proc is \([p_1, p_2, p_3]\);
  Res is 1..6;

Domain
  BagR is Rag(Res);
  P_BarG is <Poc, BagR>;

Var
  p in Proc;
  R, R2 in BagR;

Resources

\(Res\)

\(InCS\) \(P_{\text{BagR}}\)

\(OutCS\) \(Proc\)
Global allocation of resources (deadlock avoidance)

Class
Proc is [p1, p2, p3];
Res is 1..6;

Domain
BagR is Rag(Res);
P_BagR is <Poc, BagR>;

Var
p in Proc;
R, R2 in BagR;
Global allocation of resources (deadlock avoidance)

Class
Proc is \([\text{p1, p2, p3}]\);
Res is 1..6;

Domain
BagR is \(\text{Rag(Res)}\);
P_BagR is \(\langle\text{Poc, BagR}\rangle\);

Var
\(p\) in Proc;
\(R, R2\) in BagR;

Resources \(\text{Res}\)

\(\langle R \rangle\)
\(\langle R2 \rangle\)
\(\langle p, R \rangle\)
\(\langle p, R \setminus R2 \rangle\)
\(\langle p \rangle\)

InCS \(P_{\text{BagR}}\)

OutCS \(\text{Proc}\)

enter \([\text{card}(R) > 0]\)

release \([\text{card}(R2) > 0 \text{ and } R2 \subset R]\)
Global allocation of resources (deadlock avoidance)

Class
Proc is \([p1, p2, p3]\);
Res is 1..6;

Domain
BagR is \(\text{Rag}(\text{Res})\);
P_BagR is \(<\text{Proc}, \text{BagR}>\);

Var
p in Proc;
R, R2 in BagR;

Resources

\(\text{Res} \quad \text{Proc} \quad \text{InCS} \quad \text{OutCS} \quad \text{release} \quad \text{exit}

\text{enter} \quad \text{[card(R)>0]} \quad \text{[card(R)>0]}

\text{R} \quad \text{<p,R>} \quad \text{<p,R \setminus R2>} \quad \text{<p,R>} \quad \text{<p>} \quad \text{<p,R>} \quad \text{<p,R>}

\text{[card(R)>0]} \quad \text{[card(R)>0]}

[card(R2)>0 and \(R2 \subseteq R\)]
Symmetric Nets with Bags, an Example

Global allocation of resources (deadlock avoidance)

Class
Proc is \([p_1, p_2, p_3]\);
Res is 1..6;

Domain
BagR is Rag(Res);
P_BagR is \(<\text{Proc}, \text{BagR}>\);

Var
\(p \in \text{Proc};
\)
\(R, R_2 \in \text{BagR};\)

Resources
\(\text{Res}\)

Enter
\([\text{card}(R)>0]\)

Release
\([\text{card}(R_2)>0 \text{ and } R_2 \subset R]\)

InCS
\(\text{Proc} = P_{\text{BagR}}\)

Exit
\([\text{card}(R)>0]\)

OutCS
\(\text{Proc} = \langle\text{Proc.all}\rangle\)
Global allocation of resources (deadlock avoidance)

Class
Proc is \([p_1, p_2, p_3]\);
Res is \([1..6]\);

Domain
BagR is \(\text{Rag}(\text{Res})\);
P_BarG is \(<\text{Poc}, \text{BagR}>\);

Var
\(p \in \text{Proc};\)
\(R, R2 \in \text{BagR};\)

Resources
\(\text{Res} \succ \text{OutCS} \succ \text{Proc} \succ \text{InCS} \succ \text{Res} \)

Enter
\([\text{card}(R) > 0]\)

InCS
\(\text{P}\_\text{BagR}\)

Release
\([\text{card}(R2) > 0 \text{ and } R2 \subset R]\)

Exit
\([\text{card}(R) > 0]\)

much easier
for modeling
Symmetric Nets with Bags, an Example

Global allocation of resources (deadlock avoidance)

Class
Proc is [p1, p2, p3];
Res is 1..6;

Domain
BagR is Rag(Res);
P_BarG is <Poc, BagR>;

Var
p in Proc;
R, R2 in BagR;

Resources Res

Enter [card(R)>0]

InCS P_BagR

Release [card(R2)>0 and R2 ⊂ R]

extreme symmetry

OutCS Proc

Exit [card(R)>0]
Could we stack both techniques?

Mitigated results [Clarke 1996]

Why?

- The techniques seem orthogonal
- Good encoding for all possible «stats space patterns»?
  - Good encoding may only be «local»

New exploration of feasibility under new conditions

- First interesting experiments [Thierry-Mieg 2004]
- First application to SNB [Colange 2011]

Key feature: hierarchical structure (system + encoding)
Three parts

- Declaration (essentially bag constants)
- Set of Bag heaps
- Structure of the marking
Three parts
- Declaration (essentially bag constants)
- Set of Bag heaps
- Structure of the marking

**Symmetries are captured in equivalence classes**
Idea 1, reuse cardinalities

- Bound is useful when evaluating transition firing

\[
Bag_{m,n}(C) = \{ B \in Bag(C) \mid m \leq \text{card}(B) \leq n \}
\]

\[
Bag_n(C) \uplus Bag_m(C) = Bag_{n+m}(C)
\]

- Possible recursive approach (complexity in \( \log_2(|C|) \))
Representation of bag values

Idea 1, reuse cardinalities

Bound is useful when evaluating transition firing

\[ Bag_{m,n}(C) = \{ B \in Bag(C) \mid m \leq \text{card}(B) \leq n \} \]

\[ Bag_n(C) \uplus Bag_m(C) = Bag_{n+m}(C) \]

Possible recursive approach (complexity in \( \log_2(|C|) \))

Idea 2, composition of bags

Elementary marking = static equivalence classes

\[ Bag_n(C_1 \cup \ldots \cup C_k) = \bigcup_{i=1}^{k} Bag_n(C_i) \cup Bag^*(C_1, \ldots, C_k) \]

Once again recursive approach is possible
Representation of bag values

Idea 1, reuse cardinalities
- Bound is useful when evaluating transition firing

\[ Bag_{m,n}(C) = \{ B \in Bag(C) \mid m \leq \text{card}(B) \leq n \} \]

\[ Bag_n(C) \uplus Bag_m(C) = Bag_{n+m}(C) \]

Possible recursive approach (complexity in \( \log_2(|C|) \))

Idea 2, composition of bags
- Elementary marking = static equivalence classes

\[ Bag_n(C_1 \cup \ldots \cup C_k) = (\bigcup_{i=1}^{k} Bag_n(C_i)) \cup Bag^*(C_1, \ldots, C_k) \]

Once again recursive approach is possible

Idea 3, also use anonymization
Set of bags «heap»

«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

+ Anonymization

No explicit value
What to represent?
the «biggest» $C_i$
Small when static subclasses exist...

$Bag_n(C_1)$
«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

+ Anonymization
  
  No explicit value
  
  What to represent?
    
    the «biggest» $C_i$

  Small when static subclasses exist…

$\text{Bag}_n(C_1)$

$\text{Bag}_{n/2}(C_1)$
Set of bags «heap»

«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

\[ Bag_n(C_1 \cup C_2) \]

+ Anonymization

No explicit value

What to represent?

the «biggest» \( C_i \)

Small when static subclasses exist…
«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

+ Anonymization
  - No explicit value
  - What to represent?
    - the «biggest» $C_i$
  - Small when static subclasses exist…

$$\text{Bag}_n(C_1 \cup C_2)$$
$$\text{Bag}_{n/2}(C_1 \cup C_2)$$
$$\text{Bag}_n(C_1)$$
$$\text{Bag}_n(C_2)$$
$$\text{Bag}_{n/2}(C_1)$$
$$\text{Bag}_{n/2}(C_2)$$
$$\text{Bag}^*_{n/2}(C_1, C_2)$$
$$\text{Bag}_n(C_1, C_2)$$
$$\text{Bag}^*_n(C_1, C_2)$$
«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

\[ Bag_n(C_1 \cup C_2) \]

\[ Bag_{n/2}(C_1 \cup C_2) \]

Bag

n/2

(C

1

\cup C

2

)

Bag

n

(C

1

)

Bag

n/2

(C

1

)

Bag

n

(C

2

)

Bag

n/2

(C

2

)

Bag

n

(C

1

, C

2

)

Bag

n/2

(C

1

, C

2

)

+ Anonymization

No explicit value

What to represent?

the «biggest» \( C_i \)

Small when static subclasses exist…
Set of bags «heap»

«Pivot-based» representation, a recursive approach

Similar to the «recursive folding» approach [Thierry-Mieg 2009]

No explicit value

What to represent?

the «biggest» $C_i$

Small when static subclasses exist...

+ Anonymization

preserve memory and CPU

$Bag_n(C_1 \cup C_2)$

$Bag_{n/2}(C_1 \cup C_2)$

$Bag_{n/2}(C_1)$

$Bag_{n/2}(C_2)$

$Bag_n(C_1)$

$Bag_n(C_2)$

$Bag_{n/2}(C_1, C_2)$

$Bag_n^*(C_1, C_2)$
Some performances with Crocodile

On a prototype version of our tool Crocodile

Salestore example [Colange 2011]
Some performances with Crocodile

On a prototype version of our tool Crocodile

Salestore example [Colange 2011]
Some performances with Crocodile

On a prototype version of our tool Crocodile

Salestore example [Colange 2011]

Could easily be improved!
Conclusion...

... and advertising
Conclusions & Perspectives

Intensive exploitation of symmetries and hierarchy
  - Implementation in tools
    - PNXDD, Crocodile
  - But other experimentations and libraries: LibDDD, ITS-Tools, etc

Good performances at the MCC 2011 @ Petri Nets
  - And possibly at the MCC 2012 @ Petri Nets

Generalization of the «Symbolic Symbolic» approach [Colange 2012]
  - Problem: require computed permutation groups
  - Petri Nets: nice playground due to their structural analysis
  - Potential use with discrete time... [Thierry-Mieg 2011]

PNXDD & Crocodile (yet β version) available in CosyVerif
  - Open verification platform
  - Promoted by the MeFoSyLoMa group
Conclusions & Perspectives

Intensive exploitation of symmetries and hierarchy

Implementation in tools

- PNXDD, Crocodile
- But other experimentations and libraries: LibDDD, ITS-Tools, etc

Good performances at the MCC 2011 @ Petri Nets
And possibly at the MCC 2012 @ Petri Nets

Generalization of the «Symbolic Symbolic» approach

[Colange 2012]

Problem: require computed permutation groups

Petri Nets: nice playground due to their structural analysis

Potential use with discrete time... [Thierry-Mieg 2011]

PNXDD & Crocodile (yet ß version) available in CosyVerif

Open verification platform
Promoted by the MeFoSyLoMa group

http://www.cosyverif.org


