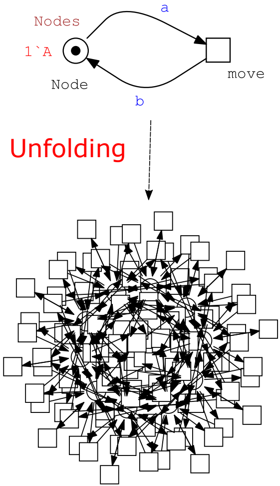
Efficient Unfolding of Coloured Petri Nets using Interval Decision Diagrams

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

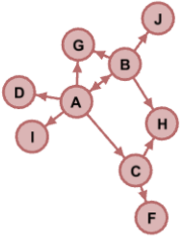
Brandenburg Technical University
Petri Nets 2020 - Paris

25 June 2020

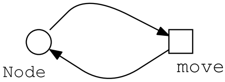
Outline



Powerful modelling - Coloured Petri nets



Coloured PNs - Powerful modelling

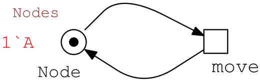


Color definitions:

\\\\ 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};



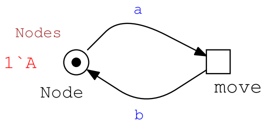
Color definitions:

\\ 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};

Coloured PNs - Powerful modelling



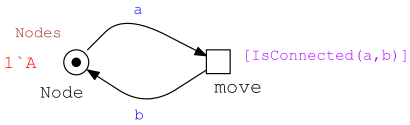
Color definitions:

\\\\ 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};

Coloured PNs - Powerful modelling



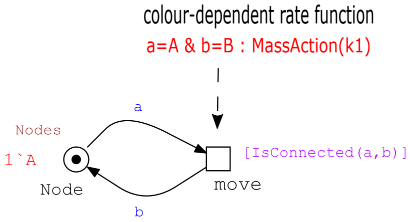
Color definitions:

\\ 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};

Coloured PNs - Powerful modelling



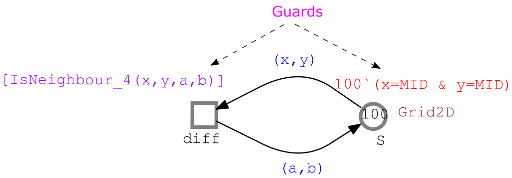
Color definitions:

\\ 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))];

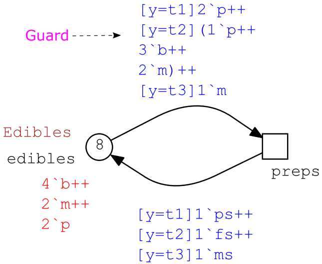
variables:
Nodes : a;
Nodes : b;
functions:
bool IsConnected(Node a1, Node b1)

{(a1,b1) elemOf Connections};

Powerful modelling - Coloured Petri nets



coloured continuous Petri net



coloured stochastic Petri net

State of the Art

- 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.

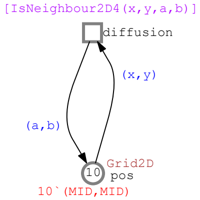
State of the Art

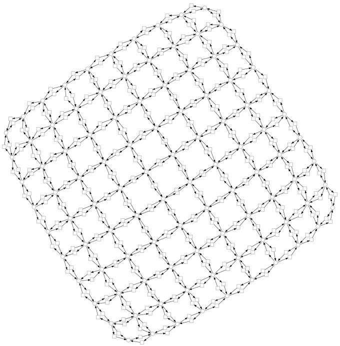
- Coloured Petri nets are in use for a wide range of applications, covering natural/engineering/life sciences.
- Currently, most analysis and simulation techniques require unfolding: coloured Petri net → plain Petri net.
- Unfolding tends to be time consuming.

State of the Art

- Example of a scaleable model to adjust grid size.

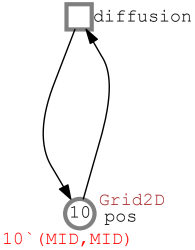
- Example of a scaleable model to adjust grid size.
- 2D Diffusion in space.





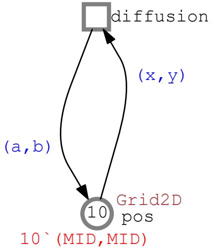
The Problem

- The core problem of efficient unfolding is to determine the transition instances, e.g, all bindings of the involved variables.

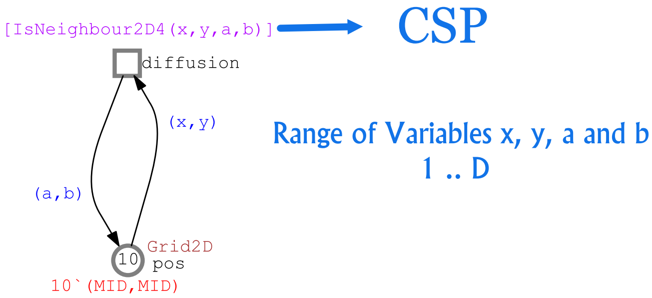


The Problem

- 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.

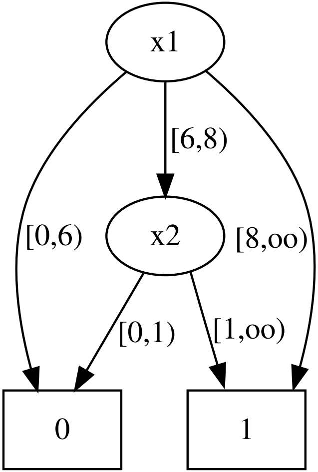


Solution

Interval decision diagrams $\rightarrow$ CSP.

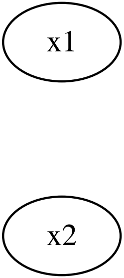
What are IDD?

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



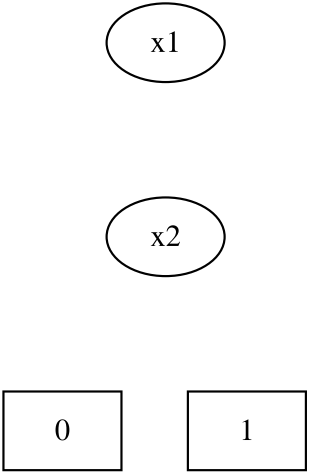
What are IDD?

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



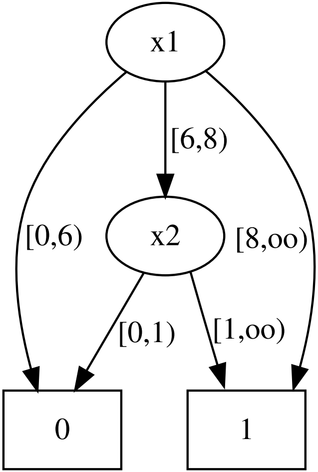
What are IDD?

- 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).



IDD basic algorithm

- The set of all paths going from: the root $\rightarrow$ the terminal node 1 describes all solutions of the given constraint problem.

IDD basic algorithm

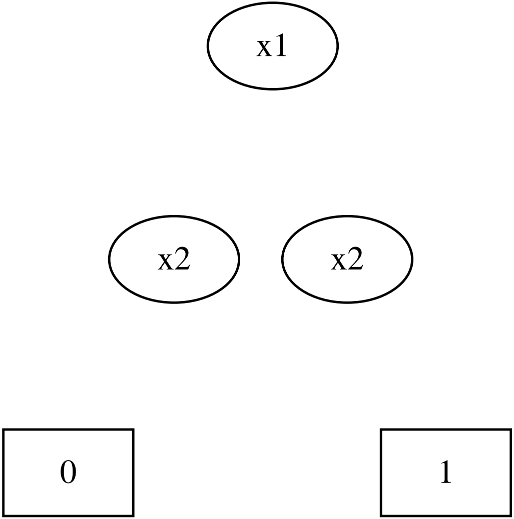
- The set of all paths going from: the root $\rightarrow$ the terminal node 1 describes all solutions of the given constraint problem.
- Typically, one path encodes more than one solution.

IDD basic algorithm

- The set of all paths going from: the root $\rightarrow$ 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.

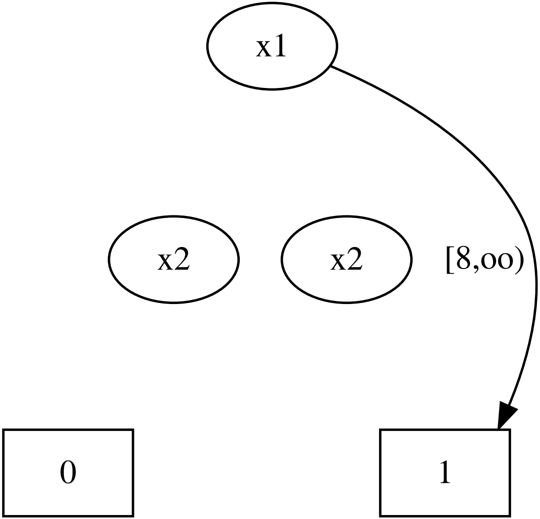
Example $(x1 \geqslant 8) \vee (x1 \in [6,8) \wedge x2>0)$

- Variable ordering



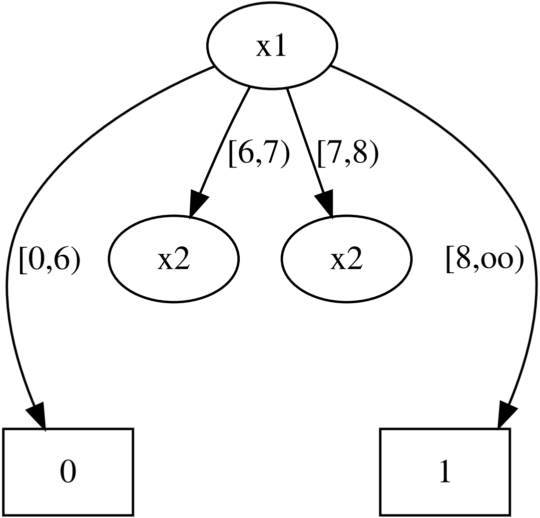
Example (x1 $\geqslant$ 8) $\vee$ (x1 $\in [6, 8)$ $\wedge$ x2>0)

- x1 $\geqslant$ 8.



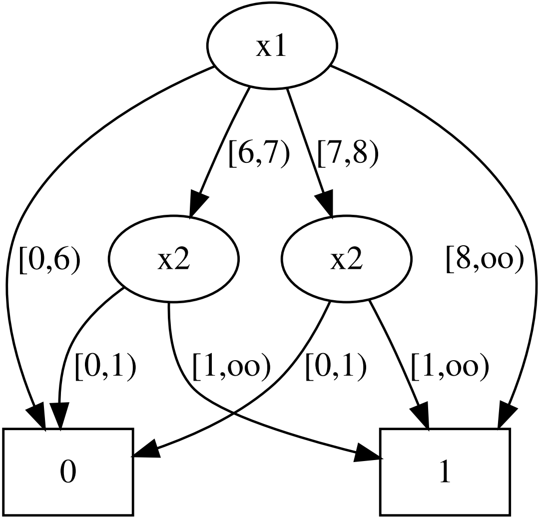
Example $(x1 \geqslant 8) \vee (x1 \in [6,8) \wedge x2>0)$

- some intermediate screenshots.



Example $(x1 \geqslant 8) \vee (x1 \in [6,8) \wedge x2>0)$

- One final solution.



Reducing IDD

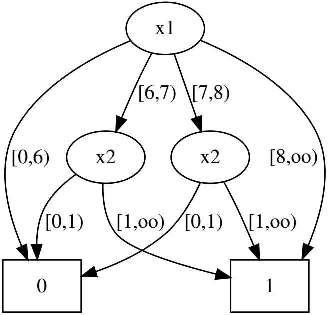
- Interval partitions labelling the outgoing arcs of each non-terminal node are reduced. For example, $[6, 7)$ and $[7, 8]$

Reducing IDD

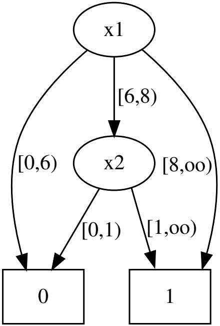
- Interval partitions labelling the outgoing arcs of each non-terminal node are reduced. For example, $[6,7)$ and $[7,8]$
- Each non-terminal node has at least two different children.

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

IDD-based unfolding algorithm

- 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.

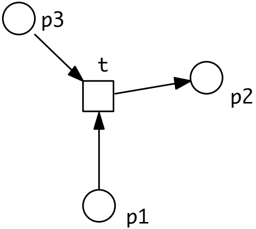
IDD-based unfolding algorithm

- 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.

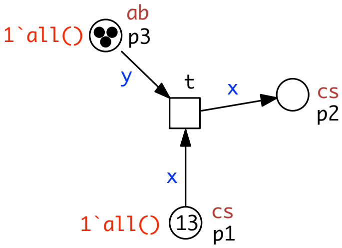
Example



Color definitions:

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

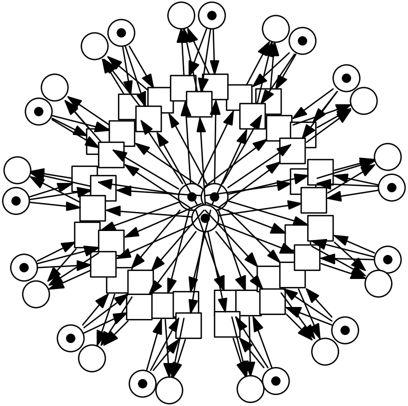
Example (no constraints)



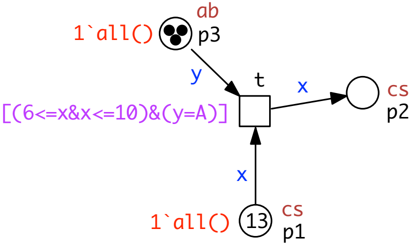
Color definitions:

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

Example (no constraints)



Example (with Guard)



Color definitions:

cs = {1,8,3..6,10,9,11,20..23};

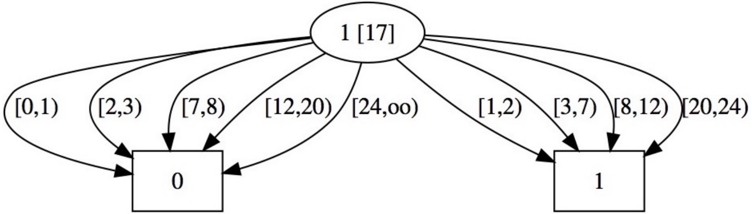
enum ab = {A,C,D};

variables:

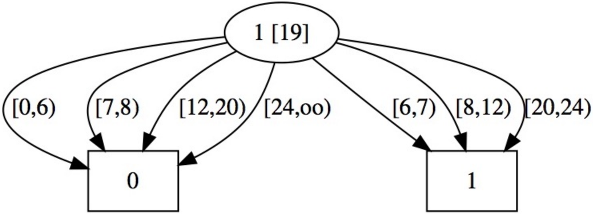
cs : x;

ab : y;

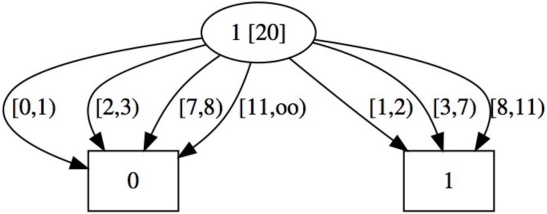
Encoding the entire cs = {1,8,3..6,10,9,11,20..23}



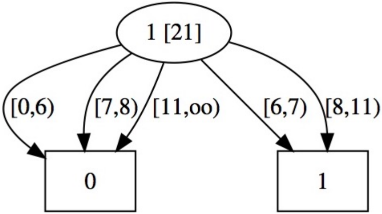
Constraining the color set cs to $6 \leqslant x$



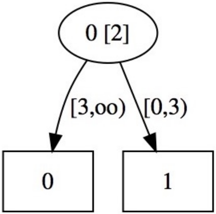
Constraining the color set cs to $x \leqslant 10$



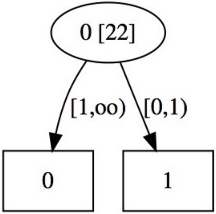
Combining $6 \leqslant x$ and $x \leqslant 10$ using & operator



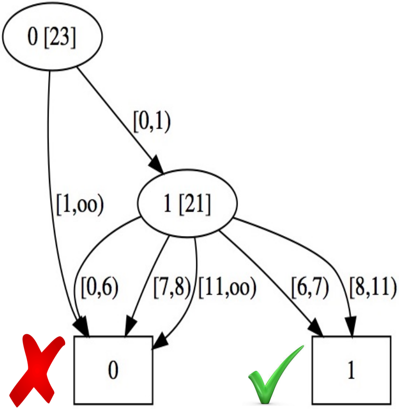
Encoding the entire color set ab ={A,C,D}



Constraining the color set ab to y = A



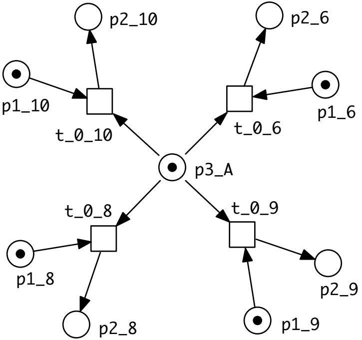
Merging the result of ($6 \leqslant x$ and $x \leqslant 10$) and ($y = A$) using & operator



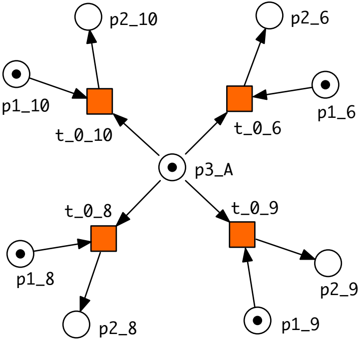
Two-path solution:

1st path : ($y = A, x = 6$)
2nd path : ($y = A$, $x = 8$ ..10)

Unfolded Net



Unfolded Net



Experiments

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

Experiments

- 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.

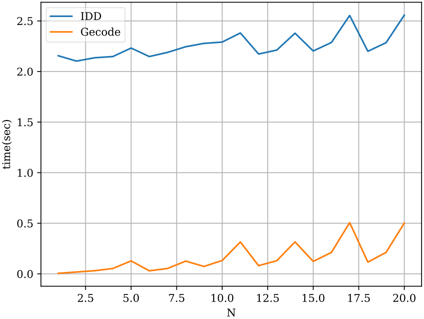
Experiments

- We compared our IDD unfolding with an unfolding employing the popular constraint solver library Gecode.
- 22 MCC models (PNML format) $\rightarrow$ https://mcc.lip6.fr/models.php
  - 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=IddUnfolding
  - 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.

Bridges and Vehicles (MCC)



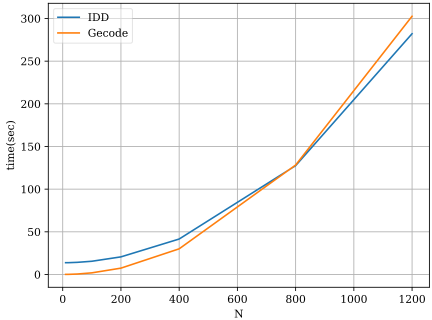
N	$\lvert P \rvert$	$\lvert T \rvert$	$\lvert A \rvert$
1	28	52	326
2	48	288	2090
4	78	968	7350
5	108	2228	17 190
9	128	1328	10 010
10	138	2348	18 090
11	168	5408	42 330
15	188	2108	15 950
16	198	3728	28 830
20	228	8588	67 470

Figure: Bridges and vehicles (MCC); requires no substantial unfolding time

the model

- reunification process.
- the coloured model has 104 places, 66 transition and198 arcs.
- it is scaled by the number of legal residents.

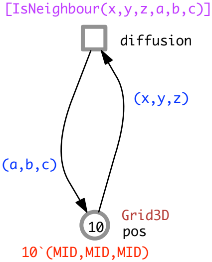
Family reunion (MCC)



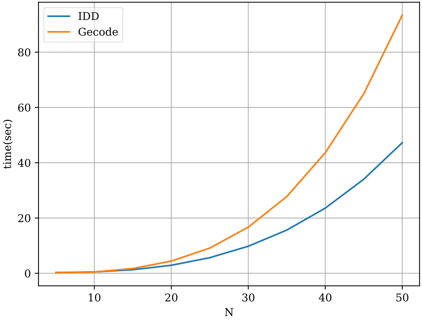
N	$\mid P \mid$	$\mid T \mid$	$\mid A \mid$
10	1475	1234	3799
20	3271	2753	8446
50	12 194	10 560	32 238
100	40 605	36 871	112 728
200	143 908	134 279	411 469
400	537 708	508 489	1 558 729
800	2 075 308	1 976 909	6 061 249
1200	4 612 908	4 405 329	13 507 769

Figure: Family Reunion (MCC); requires substantial unfolding time

Diffusion in space (3D)



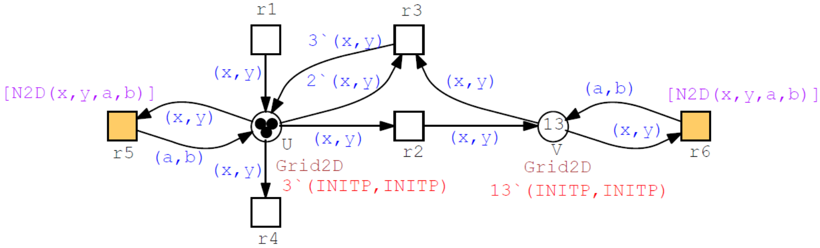
Diffusion in space (3D)



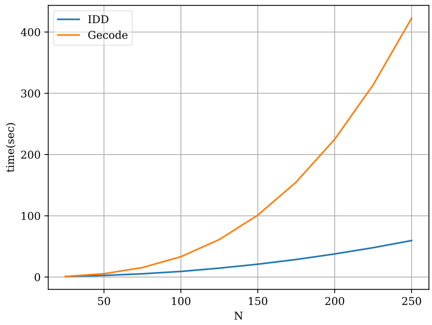
N	$\lvert P\rvert$	$\lvert T\rvert$	$\lvert A\rvert$
5	125	600	1200
10	1000	5400	10 800
15	3375	18 900	37 800
20	8000	72 998	145 996
25	15 625	90 000	180 000
30	27 000	156 600	313 200
35	42 875	249 900	499 800
40	64 000	374 400	748 800
45	91 125	534 600	1 069 200
50	125 000	735 000	1 470 000

Figure: Diffusion (3D); N – Grid size

Brusselator



Brusselator



N	$\|P\|$	$\|T\|$	$\|A\|$
25	1250	11 908	23 191
50	5000	48 808	95 116
75	11 250	110 708	215 791
100	20 000	197 608	385 216
125	31 250	309 508	603 391
150	45 000	446 408	870 316
175	61 250	608 308	1 185 991
200	80 000	795 208	1 550 416
225	101 250	1 007 108	1 963 591
250	125 000	1 244 008	2 425 516

Figure: Brusselator ; N – Grid size of a 2D square


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.

- 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.informatik.tu-cottbus.de/DSSZ/Software/Examples?dir=IddUnfolding

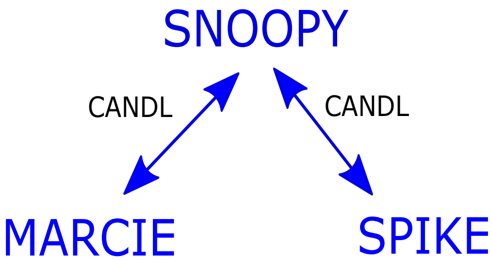
Our tools

SNOOPY

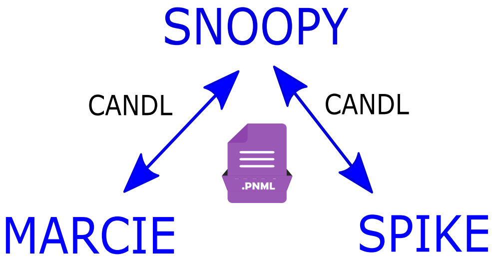
MARCIE

SPIKE

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



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



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