A tool to remove colors from your High-Level Petri nets!

Silvano DAL ZILIO
LAAS-CNRS, Vertics team

MCC is not an acronym!

**mcc** is a tool designed for a very specific task:

transform models of High-Level Petri nets (symmetric nets in **PNML**), into equivalent P/T nets

**mcc** has been developed and made available for the last 3 years and designed with the goal to be **open**, easily **extensible**, and **good enough** for the Model-Checking Contest.
Why mcc?

• short answer ≡ solve a problem we faced when entering the Model-Checking Contest three years ago.

• the initial goal ≡ develop a collection of helper apps to deal with colored models in PNML
  • unfolding
  • computing invariants on colored models
  • computing symmetries

"There are only two hard things in Computer Science: cache invalidation and naming things."

-- Phil Karlton

why not compute directly on colored models?
A tool to remove colors from your High-Level Petri nets!

quick tool demo: \textit{mcc in 50”}

see also our 10' tool demonstration vidéo for the Petri Nets Conference—June 2020
Installing MCC or building it from source

• MCC is a classic CLI tool
  Just install the right binary file in your PATH.
  Binary files for Windows, Linux and MacOS.
  See the latest releases on GitHub

• You have the option to install the tool from source, using a recent Go distribution. Just:

  > go get github.com/dalzilio/mcc
Architecture of MCC

```
$> mcc tina -i lamport.pnml --name
```
Architecture of MCC

1. UTF-8 compatible parser, with error handling

2. Unfolding engine

3. "Code" generation

- PNML file (XML)
  - pnml Decoder
  - High-Level net (AST)
  - hlnet Build
  - Core Petri net (AST)
  - corenet Print

- TINA
- LOLA
- PNML (TEXT)
Parsing PNML files

1. UTF-8 compatible parser, with error handling
2. Unfolding engine
3. “Code” generation

- PNML file (XML) to pnml Decoder
- High-Level net (AST) to hlnet Build
- Core Petri net (AST) to corenet Print

- TINA
- LOLA
- PNML (TEXT)
What is in a PNML model?

- **types** ≡ what is the color of a place
- **constants** ≡ token (from a given color)
- **expressions** ≡ operations over multiset of constants
- **conditions** ≡ used in guards
Supported PNML elements

**types** ::= dot
| cyclicenumeration
| finiteenumeration
| finiteintrange
| productsort
| partition
| partitionelement

**constants** ::= dotconstant (●)
| feconstant
| finiteintrangeconstant

**expressions** ::= variable (x)
| successor (x++) | predecessor (x--)
| tuple
| all | add | subtract

**conditions** ::= or | and | equality
| inequality
| lessthan | greaterthan
| greaterthanorequal
| lessthanorequal
PNML 101: symmetric net

XML language for describing a subset of colored nets (with values of finite types)

Output format

1. UTF-8 compatible parser, with error handling
2. Unfolding engine
3. "Code" generation
PNML 101: Place/Transition nets

XML language for describing “core” P/T nets.

URI: http://www.pnml.org/version-2009/grammar/ptnet
The .net format from Tina

Same information than in the PNML model for P/T nets
⇒ this is just an oriented graph

Most tools in Tina actually supports the PNML ptnet format
Unfolding engine

UTF-8 compatible parser, with error handling

“code” generation
Unfolding: an example

(a = x ∨ a = x++ ∨ a = x--)
∧ (b = y ∨ b = y++ ∨ b = y--)
∧ (a ≠ x ∨ b ≠ y)

1'(x, y)

1'(a, b)

10'(0, 0)

Grid : CD × CD
CD : 0..1
Unfolding: types and environment

Guard $g(w, x, y, z)$

Patterns

$\pi_1(x, y)$

$\pi_2(x, w, z)$

Environment $\equiv x : T_x, \ldots, z : T_z$
Unfolding: types and environment

\[
guard \quad g(w, x, y, z)
\]

\[
\pi_1(x, y)
\]

\[
\pi_2(x, w, z)
\]

Environment \equiv x : T_x, ..., z : T_z
Unfolding: a constraint solving approach

\[ g(w, x, y, z) \]

patterns
\[ \pi_1(x, y) \]
\[ \pi_2(x, w, z) \]

guard \( g(w, x, y, z) \)

Environment \( \equiv x: T_x, \ldots, z: T_z \)
What else can you do with mcc?
- structured naming of places
- debugging + prettifying of colored models

How good is it?
- mcc finishes on all the feasible models in the MCC

How does it look like under the hood?
- we follow a constraint-solving approach
- use of colored invariant
- use of a Petri scripting language
Prettifying PNML models

```bash
$ mcc hlnet -i table.pnml --debug
```
Comparison with other Tools

Colored net (PNML) unfolding in the literature

Unfolding tools in the MCC
• verifypn (part of Tapaal)
• andl_converter (part of Marcie)
• GreatSPN editor
<table>
<thead>
<tr>
<th>Model</th>
<th>Places</th>
<th>Trans.</th>
<th>MCC</th>
<th>TAPAA</th>
<th>MARCIE</th>
<th>GSPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalResAllocation-07</td>
<td>133</td>
<td>291067</td>
<td>1.7</td>
<td>3</td>
<td>14.4</td>
<td>22.3</td>
</tr>
<tr>
<td>GlobalResAllocation-11</td>
<td>297</td>
<td>2.10^6</td>
<td>15.1</td>
<td>29.3</td>
<td>144.6</td>
<td>—</td>
</tr>
<tr>
<td>DrinkVendingMachine-16</td>
<td>192</td>
<td>10^6</td>
<td>15.5</td>
<td>10.7</td>
<td>52.8</td>
<td>108.1</td>
</tr>
<tr>
<td>DrinkVendingMachine-24</td>
<td>288</td>
<td>8.10^6</td>
<td>97.1</td>
<td>95.9</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>PhilosophersDyn-50</td>
<td>2850</td>
<td>255150</td>
<td>1</td>
<td>2.1</td>
<td>11.1</td>
<td>15.7</td>
</tr>
<tr>
<td>PhilosophersDyn-80</td>
<td>6960</td>
<td>10^6</td>
<td>4.1</td>
<td>9.9</td>
<td>55.9</td>
<td>61.0</td>
</tr>
<tr>
<td>Diffusion-D050</td>
<td>2500</td>
<td>8109</td>
<td>14.5</td>
<td>0.6</td>
<td>4.1</td>
<td>—</td>
</tr>
<tr>
<td>Diffusion-D100</td>
<td>10000</td>
<td>31209</td>
<td>243.3</td>
<td>8.6</td>
<td>31.3</td>
<td>—</td>
</tr>
<tr>
<td>TokenRing-100</td>
<td>10201</td>
<td>10^6</td>
<td>4</td>
<td>8.2</td>
<td>33.5</td>
<td>49.3</td>
</tr>
<tr>
<td>TokenRing-200</td>
<td>40401</td>
<td>8.10^6</td>
<td>67.4</td>
<td>166.1</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SafeBus-50</td>
<td>5606</td>
<td>140251</td>
<td>14.2</td>
<td>1.4</td>
<td>6.2</td>
<td>25.1</td>
</tr>
<tr>
<td>SafeBus-80</td>
<td>13766</td>
<td>550801</td>
<td>89.5</td>
<td>7</td>
<td>20.6</td>
<td>133.1</td>
</tr>
<tr>
<td>TrainTable-Dist</td>
<td>722</td>
<td>602</td>
<td>1.4</td>
<td>12.6</td>
<td>59.5</td>
<td>69.4</td>
</tr>
<tr>
<td>TrainTable-Stop+Dist</td>
<td>728</td>
<td>602</td>
<td>2.1</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>BART-002</td>
<td>764</td>
<td>646</td>
<td>3.1</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>BART-060</td>
<td>15032</td>
<td>19380</td>
<td>3.2</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SharedMemory-000200</td>
<td>40801</td>
<td>80400</td>
<td>0.3</td>
<td>1.7</td>
<td>2.6</td>
<td>5.1</td>
</tr>
<tr>
<td>SharedMemory-001000</td>
<td>10^6</td>
<td>2.10^6</td>
<td>8.9</td>
<td>—</td>
<td>60.3</td>
<td>160.2</td>
</tr>
<tr>
<td>SharedMemory-002000</td>
<td>4.10^6</td>
<td>8.10^6</td>
<td>55.3</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FamilyReunion-L800</td>
<td>2.10^6</td>
<td>2.10^6</td>
<td>5.5</td>
<td>—</td>
<td>84.8</td>
<td>143.0</td>
</tr>
<tr>
<td>FamilyReunion-L3000</td>
<td>28.10^6</td>
<td>27.10^6</td>
<td>89.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Implementation: colored invariants

Place $StopTable$ is stable.

Its type has $46 \times 6 = 276$ possible values, its initial marking only $6$

Therefore there are $276^2 = 76,176$ potential combinations to test for transitions $Dec$ and $Acc$; instead of $36$

$DisTable: DIST \times SPEED \times DIST$
$TrainState: ID \times SPEED \times DIST$
$DIST: 0..45$ $SPEED: 0..5$
Implementation: Petri scripting language

Here

4'(all : Resource)

1'(x--)

1'(x)

1'(x)

1'(x)

1'(x--)

1'(x)

1'(x)

1'(x)

Here, There : RESOURCE

RESOURCE : 1..10
Implementation: Petri scripting language

net \{Swap-COL-P10-N4\}

pl p0 : \{Here\} (4)
pl p1 : \{There\}
tr t0 : \{l-t1\} p1 \rightarrow p0
tr t1 : \{r-t1\} p0 \rightarrow p1
tr t2 : \{h-t2\} p0 \rightarrow p1

RING sync 10 /\{l-t1\},\{r-t1\} \{h-t2\}
thank you to:

- paxtonhare/demo-magic: for repeatable shell script demos
- Audacity: for the audio editing
- Captura: for the screen capture
- Shotcut: for my first experience editing videos

https://github.com/dalzilio/mcc

tina-users @ laas . fr