Higher-Dimensional Automata for Petri Net Analysis

Uli Fahrenberg   Philipp Schlehuber-Caissier

LMF, Université Paris-Saclay, France
SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France

AWPN 2025

Laboratoire Méthodes Formelles

TELECOM SudParis

IP PARIS

Higher-Dimensional Automata?

PN'23 UF, K. Ziemiański: *A Myhill-Nerode theorem for higher-dimensional automata*

PN'24 A. Amrane, H. Bazille, E. Clement, UF: *Languages of higher-dimensional timed automata*

PN'25 A. Amrane, H. Bazille, UF, L. Hélouët, P. Schlehuber-Caissier: *Petri Nets and higher-dimensional automata*

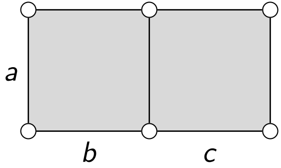
Higher-Dimensional Automata?

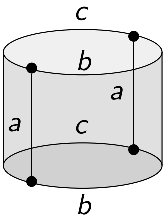
PN'23 UF, K. Ziemiański: *A Myhill-Nerode theorem for higher-dimensional automata*

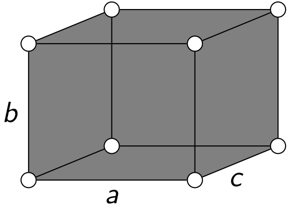
PN'24 A. Amrane, H. Bazille, E. Clement, UF: *Languages of higher-dimensional timed automata*

PN'25 A. Amrane, H. Bazille, UF, L. Hélouët, P. Schlehuber-Caissier: *Petri Nets and higher-dimensional automata*

- Rob van Glabbeek 2006: *On the expressiveness of higher-dimensional automata*

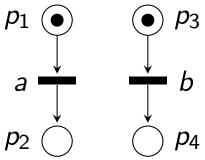




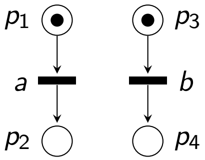


Semantics of Petri Nets

Petri net (S, T, F): places S; transitions T; weighted flows $F : S \times T \cup T \times S \to \mathbb{N}$



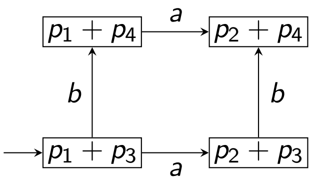
Semantics of Petri Nets



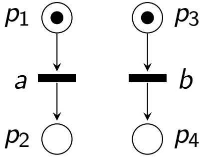
Petri net (S, T, F): places S; transitions T; weighted flows $F : S \times T \cup T \times S \to \mathbb{N}$

Interleaved semantics (reachability graph) (V, E):

- $V = \mathbb{N}^S$: all markings
- $E \subseteq V \times T \times V$: one transition at a time
- $E = \{(m, t, m') \mid {}^\bullet t \leq m, m' = m - {}^\bullet t + t^\bullet\}$
- initial marking $\implies$ initial state; take reachable part



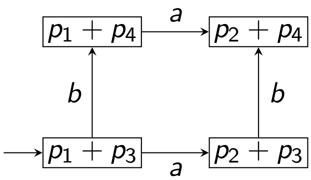
Semantics of Petri Nets



Petri net (S, T, F): places S; transitions T; weighted flows $F : S \times T \cup T \times S \to \mathbb{N}$

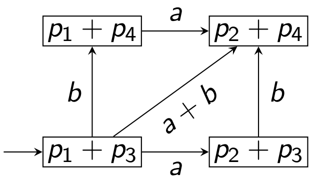
Interleaved semantics (reachability graph) (V, E):

- $V = \mathbb{N}^S$: all markings
- $E \subseteq V \times T \times V$: one transition at a time
- $E = \{(m, t, m') \mid {}^\bullet t \le m, m' = m - {}^\bullet t + t^\bullet\}$
- initial marking $\implies$ initial state; take reachable part

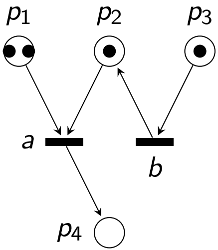


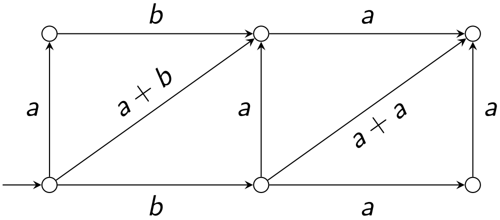
Concurrent step reachability graph (V, E'):

- $V = \mathbb{N}^S$
- $E' \subseteq V \times \mathbb{N}^T \times V$: multisets of transitions
- $E' = \{(m, U, m') \mid {}^\bullet U \le m, m' = m - {}^\bullet U + U^\bullet\}$



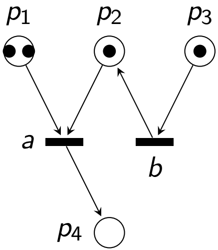
Another Example

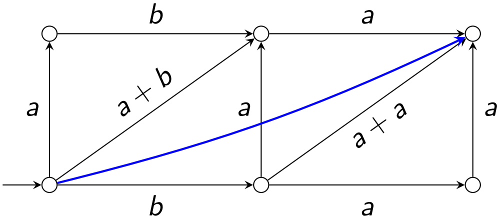




- after firing b, a is auto-concurrent

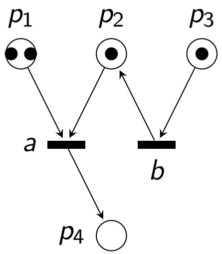
Another Example

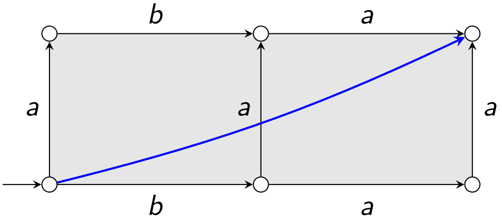




- after firing b, a is auto-concurrent
- (Concurrent step) semantics misses some behavior?
  - start a – start b – finish b – start another a – etc.

Another Example





- after firing b, a is auto-concurrent
- (Concurrent step) semantics misses some behavior?
  - start a – start b – finish b – start another a – etc.
- enter higher-dimensional automata
  - replace multi-transitions by squares / cubes / etc.

Higher-Dimensional Automata

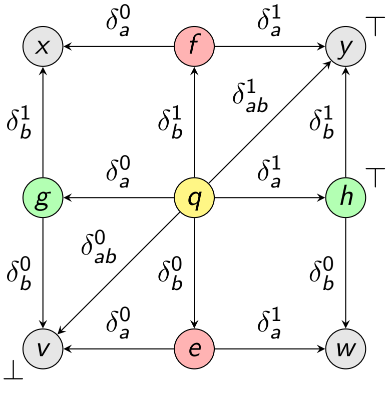
A conclist is a finite, totally ordered, Σ-labeled set. (a list of labeled events)

A precubical set X consists of:

- A set of cells X (cubes)
- Every cell $x \in X$ has a conclist $\mathrm{ev}(x)$ (list of events active in x)
- We write $X[U] = \{x \in X \mid \mathrm{ev}(x) = U\}$ for a conclist U (cells of type U)
- For every conclist U and $A \subseteq U$ there are:
  upper face map $\delta^1_A : X[U] \to X[U \setminus A]$ (terminating events A)
  lower face map $\delta^0_A : X[U] \to X[U \setminus A]$ ("unstarting" events A)
- Precube identities: $\delta^\mu_A \delta^\nu_B = \delta^\nu_B \delta^\mu_A$ for $A \cap B = \emptyset$ and $\mu, \nu \in \{0, 1\}$

A higher dimensional automaton (HDA) is a precubical set X with initial cells $\bot \subseteq X$ and accepting cells $\top \subseteq X$ (not necessarily vertices)

Example



$$X[\emptyset] = \{v, w, x, y\}$$

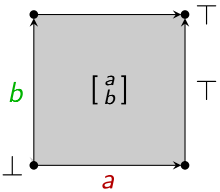
$$X[a] = \{e, f\}$$

$$X[b] = \{g, h\}$$

$$X[[\,{}^{a}_{b}\,]] = \{q\}$$

$$\perp_X = \{v\}$$

$$\top_X = \{h, y\}$$



Higher-Dimensional Automata & Concurrency Theory

HDAs as a model for concurrency:

- points: states
- edges: transitions
- squares, cubes etc.: independency relations / concurrently executing events
- two-dimensional automata $\cong$ asynchronous transition systems [Shields'85] [Bednarczyk'88]

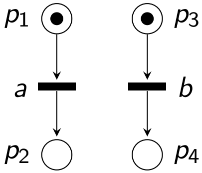
- Introduced in [van Glabbeek'89]
- Generalize all main models of concurrency proposed in the literature
- (event structures; Petri nets; communicating automata; etc.)
- [van Glabbeek'06]: translations from Petri nets
    - individual vs. collective tokens; autoconcurrency or not

Our Contributions

- Update van Glabbeek's translation to our new event-based HDA formalism
- Implement translation in new tool `pn2HDA`
- Extend to inhibitor arcs
- Extend to generalized self-modifying nets
- Continuous effort to make `pn2HDA` less stupid and more efficient ⟵ Philipp

Concurrent Semantics of Petri Nets

Petri net (S, T, F): places S; transitions T;
weighted flows $F : S \times T \cup T \times S \to \mathbb{N}$



Interleaved semantics (V, E): $V = \mathbb{N}^S$; $E \subseteq V \times T \times V$

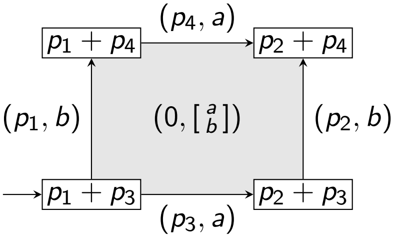
- $E = \{(m, t, m') \mid {}^\bullet t \le m, m' = m - {}^\bullet t + t^\bullet\}$

Concurrent semantics as HDA:
$\square = \square(T)$, $X = \mathbb{N}^S \times \square$, $\mathrm{ev}(m, \tau) = \tau$

- for $x = (m, \tau) \in X[\tau]$ with $\tau = (t_1, \dots, t_n)$:

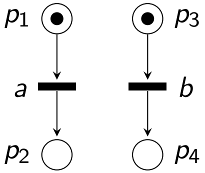
$$\delta^0_{t_i}(x) = (m + {}^\bullet t_i, (t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n))$$
$$\delta^1_{t_i}(x) = (m + t_i^\bullet, (t_1, \dots, t_{i-1}, t_{i+1}, \dots, t_n))$$

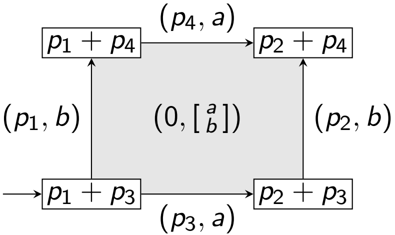
- initial marking $\implies$ initial cell; take reachable part
- (no accepting cells)



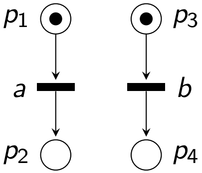
Event Order

- trouble with symmetry:
  have a cell $(0, \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right])$, but also $(0, \left[\begin{smallmatrix} b \\ a \end{smallmatrix}\right])$ (not shown)
- solution: fix an arbitrary order $\preccurlyeq$ on T
- and use $\square = \left\{ \left[\begin{smallmatrix} t_1 \\ \vdots \\ t_n \end{smallmatrix}\right] \;\middle|\; \forall i = 1, \dots, n-1 : t_i \preccurlyeq t_{i+1} \right\}$
  instead of $\square(T)$
- order $\preccurlyeq$ may be chosen (and re-chosen) at will
- here: lexicographic $a \prec b \prec \dots$





Example, Complete

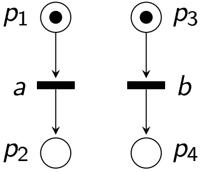


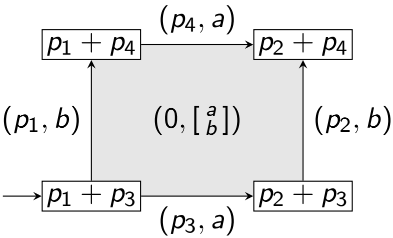
$$
\begin{aligned}
X[\emptyset] &= \{p_1 + p_3, p_2 + p_3, p_1 + p_4, p_2 + p_4\}\\
X[a] &= \{(p_3, a), (p_4, a)\}\\
X[b] &= \{(p_1, b), (p_2, b)\}\\
X[[\begin{smallmatrix}a\\b\end{smallmatrix}]] &= \{(0, [\begin{smallmatrix}a\\b\end{smallmatrix}])\}
\end{aligned}
$$

$$
\begin{tikzcd}
 & p_1+p_4 \arrow[r, "(p_4,a)"] & p_2+p_4 \\
 \arrow[r] & p_1+p_3 \arrow[u, "(p_1,b)"] \arrow[r, "(p_3,a)"'] \arrow[ur, phantom, "(0,[\begin{smallmatrix}a\\b\end{smallmatrix}])"] & p_2+p_3 \arrow[u, "(p_2,b)"']
\end{tikzcd}
$$

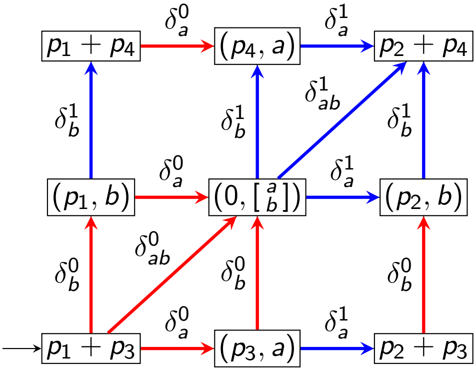
$$
\begin{tikzcd}
 & p_1+p_4 & (p_4,a) \arrow[l, "\delta_a^0"'] \arrow[r, "\delta_a^1"] & p_2+p_4 \\
 & (p_1,b) \arrow[u, "\delta_b^1"] \arrow[d, "\delta_b^0"'] & (0,[\begin{smallmatrix}a\\b\end{smallmatrix}]) \arrow[l, "\delta_a^0"'] \arrow[r, "\delta_a^1"] \arrow[u, "\delta_b^1"] \arrow[d, "\delta_b^0"'] \arrow[ur, "\delta_{ab}^1"] \arrow[dl, "\delta_{ab}^0"'] & (p_2,b) \arrow[u, "\delta_b^1"'] \arrow[d, "\delta_b^0"] \\
 \arrow[r] & p_1+p_3 & (p_3,a) \arrow[l, "\delta_a^0"] \arrow[r, "\delta_a^1"'] & p_2+p_3
\end{tikzcd}
$$

Example, Complete

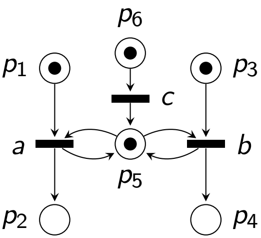


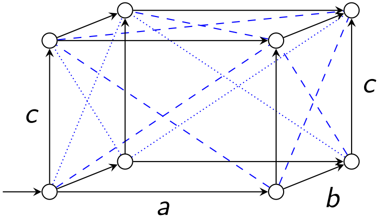


– for computations, *invert* lower face maps



Another Example





- initially, p_5 is a mutex place: it disables concurrency of a and b
- after c fires, p_5 holds two tokens, so a and b become concurrent
- semantically, a hollow cube without bottom face
- the five faces: front: $(p_3, \left[\begin{smallmatrix} a \\ c \end{smallmatrix}\right])$, back: $(p_4, \left[\begin{smallmatrix} a \\ c \end{smallmatrix}\right])$
  left: $(p_1, \left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right])$, right: $(p_2, \left[\begin{smallmatrix} b \\ c \end{smallmatrix}\right])$
  top: $(0, \left[\begin{smallmatrix} a \\ b \end{smallmatrix}\right])$

Working with Petri nets

We wish to build a tool for analyzing and verifying Petri nets based on HDAs.

Working with Petri nets

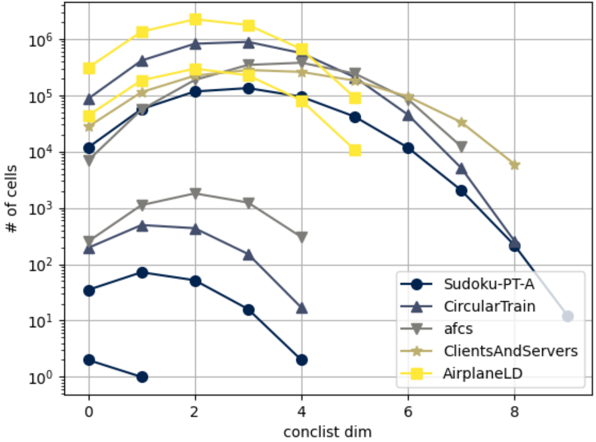
We wish to build a tool for analyzing and verifying Petri nets based on HDAs.

As a first step we seek to construct the reachable part of the state-space: `pn2HDA`

Working with Petri nets

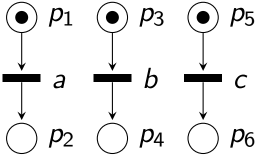
We wish to build a tool for analyzing and verifying Petri nets based on HDAs.

As a first step we seek to construct the reachable part of the state-space: `pn2HDA`



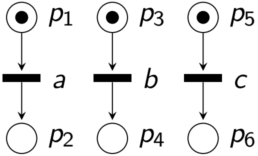
- ⊕ Implementation very close to mathematical definition
- ⊖ Very memory and cpu intensive
- ⊖ Only able to treat small instances from the MCC

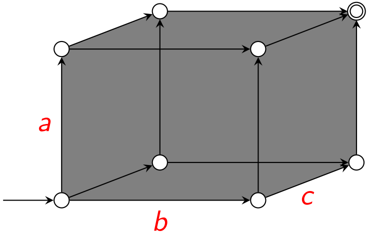
Taking a closer look at HDA cells



Taking a closer look at HDA cells

Three independent transitions form a full, three-dimensional cell: $(0, \begin{bmatrix} a \\ b \\ c \end{bmatrix})$.

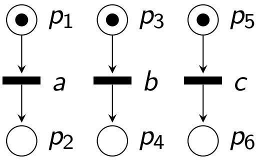


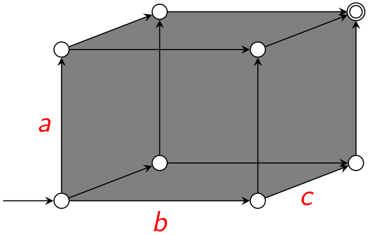


Taking a closer look at HDA cells

Three independent transitions form a full, three-dimensional cell: $(0, \begin{bmatrix} a \\ b \\ c \end{bmatrix})$.

But, in total we have
1 3-cell, 6 2-cells, 12 1-cells, and 8 0-cells.
Plus all of the face-map entries!
($2n$ for n-dim cells)





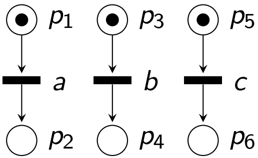
Taking a closer look at HDA cells

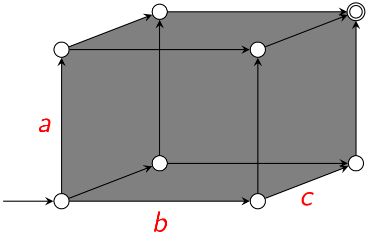
Three independent transitions form a full, three-dimensional cell: $(0, \begin{bmatrix} a \\ b \\ c \end{bmatrix})$.

But, in total we have
1 3-cell, 6 2-cells, 12 1-cells, and 8 0-cells.
Plus all of the face-map entries!
($2n$ for n-dim cells)

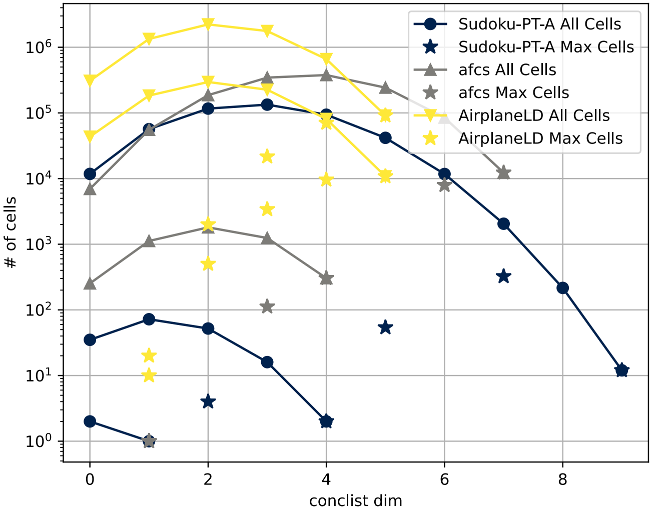
In fact, for a full HDA, any n-dimensional cell has $3^n - 1$ faces, which are guaranteed to exist.

An “exponential” waste!

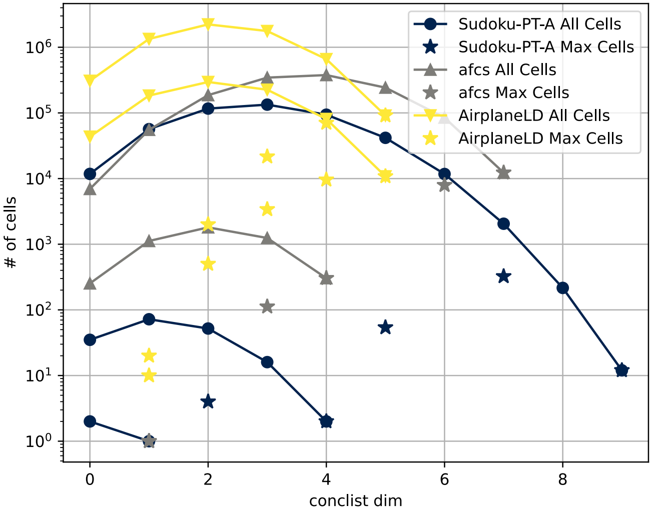




Maybe it is not that bad in practice?

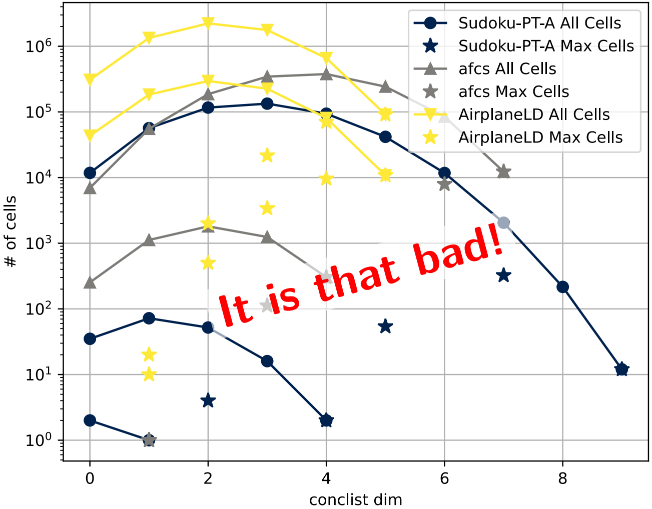


Maybe it is not that bad in practice?



instance	compr.
AirplaneLD-pt-0010	97.1%
AirplaneLD-pt-0020	97.1%
afcs_01_a	91.2%
afcs_02_a	98.5%
Sudoku-PT-A-N01	66.7%
Sudoku-PT-A-N02	96.6%
Sudoku-PT-A-N03	99.2%

Maybe it is not that bad in practice?



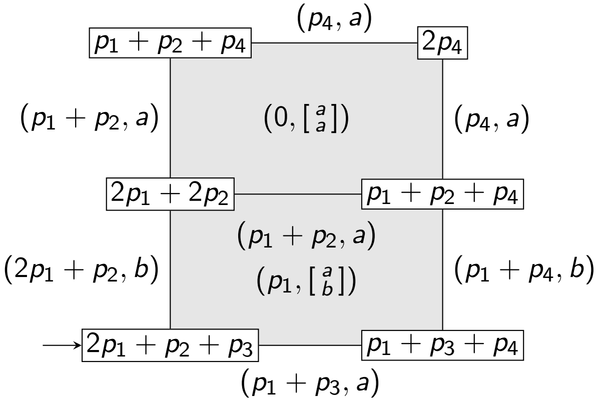
instance	compr.
AirplaneLD-pt-0010	97.1%
AirplaneLD-pt-0020	97.1%
afcs_01_a	91.2%
afcs_02_a	98.5%
Sudoku-PT-A-N01	66.7%
Sudoku-PT-A-N02	96.6%
Sudoku-PT-A-N03	99.2%

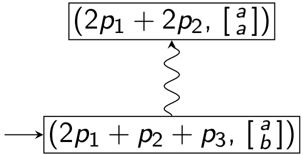
Extending pn2HDA

We are currently working on extending pn2HDA to support max-cells.

Extending pn2HDA

We are currently working on extending pn2HDA to support max-cells.
We strive for a model with minimal memory consumption, possibly at the cost of speed.



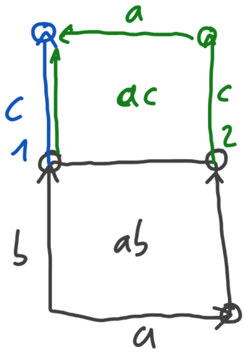


- Keep "lower-left"-marking and maximal conclist
- Keep which cells are connected (but not how?)

Direct Translation of Petri Nets to MHDA - 1

Given a partially constructed MHDA A and a max cell (m, c) to explore

- Compute the marking for each 0-cell
- Compute the set of maximal concurrent steps
- For each of them form the candidate cell (m', c')
- Check inclusion with all existing cells (m_e, c_e)
  - $(m', c') \sqsubseteq (m_e, c_e)$
  - $(m_e, c_e) \sqsubseteq (m', c')$



Direct Translation of Petri Nets to MHDA - 2

Inclusion checking is translated to an integer program, solved via z3.
$(m, c) \sqsubseteq (m', c')$ iff we can reach m from m' with steps in $c' \setminus c$.

Direct Translation of Petri Nets to MHDA - 2

Inclusion checking is translated to an integer program, solved via z3.
$(m, c) \sqsubseteq (m', c')$ iff we can reach m from m' with steps in $c' \setminus c$.

	HDA			MHDA			time (s)	
Name	cells	conclists	markings	cells	conclists	markings	PN $\to$ HDA	PN $\to$ MHDA
abx_3	272	50	146	10	10	4	0.003	0.7
abx_4	846	105	371	15	15	5	0.005	4
abx_5	2232	196	812	21	21	6	0.008	13
abx_6	5214	336	1596	28	28	7	0.01	44
Sudoku-PT-A-N02	177	35	176	6	23	5	0.4	19

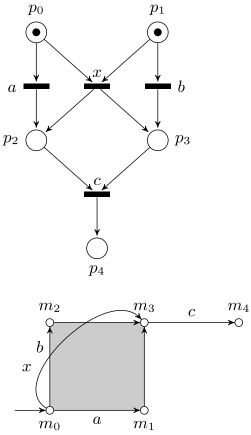
Problem Analysis

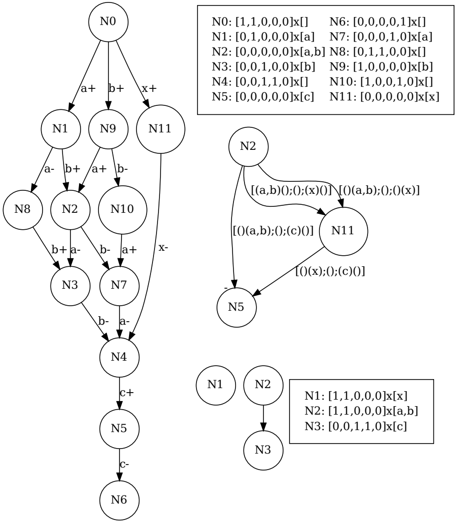
Why is it sooo slow??

What are the expensive parts?

- Investigate from all 0-cells ($2^{|C|}$)
- Finding (maximal) concurrent steps from a 0-cell (NP)
- Inclusion checks (linear in current size, but costly)

Intermezzo: Presenting abx





Simplifying Inclusion Checks

Integer programming is generally expensive

Simplifying Inclusion Checks

Integer programming is generally expensive

Luckily our problem for $(m, c) \sqsubseteq (m', c')$ looks like this

$$m_i = m'_i - \sum_t \mathtt{pre}(i, t) n_t + \sum_t \mathtt{post}(i, t) n_t$$
$$0 \leq n_t \leq |c' \setminus c|$$

with $\mathtt{pre}(i, t)/\mathtt{post}(i, t)$ being 1 if $i \in {}^\bullet t / t^\bullet$ else 0

Simplifying Inclusion Checks

Integer programming is generally expensive

Luckily our problem for $(m, c) \sqsubseteq (m', c')$ looks like this

$$m_i = m'_i - \sum_t \mathtt{pre}(i, t) n_t + \sum_t \mathtt{post}(i, t) n_t$$
$$0 \leq n_t \leq |c' \setminus c|$$

with $\mathtt{pre}(i, t)/\mathtt{post}(i, t)$ being 1 if $i \in {}^\bullet t/t^\bullet$ else 0

This looks much like a linear program!

Simplifying Inclusion Checks

Integer programming is generally expensive

Luckily our problem for $(m, c) \sqsubseteq (m', c')$ looks like this

$$m_i = m'_i - \sum_t \mathtt{pre}(i, t) n_t + \sum_t \mathtt{post}(i, t) n_t$$
$$0 \leq n_t \leq |c' \setminus c|$$

with $\mathtt{pre}(i, t)/\mathtt{post}(i, t)$ being 1 if $i \in {}^\bullet t / t^\bullet$ else 0

This looks much like a linear program!

What about non-integer solutions?
Claim:
They only arise for parallel transitions and imply the existence of an integer solution.

Computing successors more efficiently

Finding the set of maximal concurrent steps from a given marking is a hard problem in itself...

Computing successors more efficiently

Finding the set of maximal concurrent steps from a given marking is a hard problem in itself...

Can we find a more efficient way to enumerate them?

$$m_i \geq \sum_t \mathtt{pre}(i, t) n_t$$

$$0 \leq n_t$$

Computing successors more efficiently

Finding the set of maximal concurrent steps from a given marking is a hard problem in itself...

Can we find a more efficient way to enumerate them?

$$m_i \geq \sum_t \texttt{pre}(i,t) n_t$$
$$0 \leq n_t$$

This looks like an H-representation of a polytope!

Computing successors more efficiently

Finding the set of maximal concurrent steps from a given marking is a hard problem in itself...

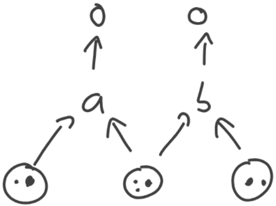
Can we find a more efficient way to enumerate them?

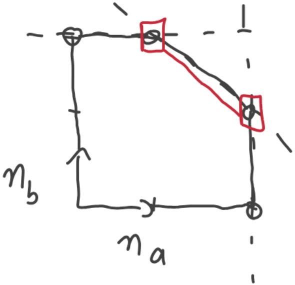
$$m_i \geq \sum_t \mathtt{pre}(i,t) n_t$$
$$0 \leq n_t$$

This looks like an H-representation of a polytope!

Even better, the polytope is bounded and all the coefficients are integer (even even better they are 0 or 1)!

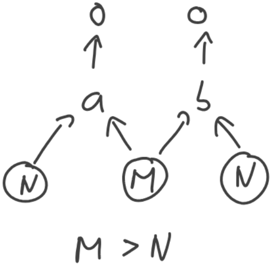
Why does it matter? # 2

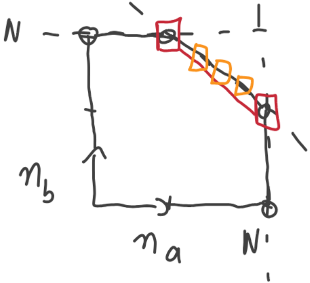




We can find the V-representation!
The pareto optimal vertices are those of interest for max-cell successors!

Why does it matter? # 2

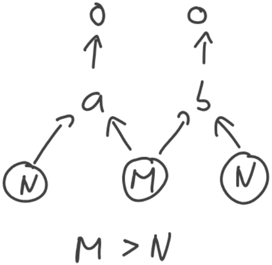


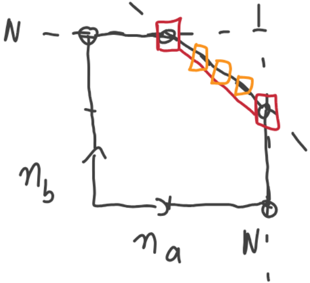


We can find the V-representation!
The pareto optimal vertices are those of interest for max-cell successors!

Going from H- to V-representation is independent of N and M.

Why does it matter? # 2





We can find the V-representation!
The pareto optimal vertices are those of interest for max-cell successors!

Going from H- to V-representation is independent of N and M.
Unfortunately. Problems enumerating the integer solutions in higher dimensions.

Computing all the successors of a max cell

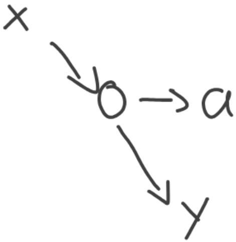
Even if the above would have worked, we still have several issues!

Computing all the successors of a max cell

Even if the above would have worked, we still have several issues!
Notably the iteration over all over the 0-cells is still necessary.

Computing all the successors of a max cell

Even if the above would have worked, we still have several issues!
Notably the iteration over all over the 0-cells is still necessary.

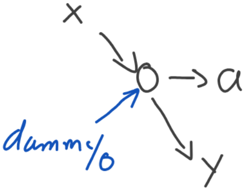


- $x^{-}a^{+}$
- $y^{-1}a^{+}$

Directions only *demand* at most one token per place.
We only construct maximal directions.

Computing all the successors of a max cell

Even if the above would have worked, we still have several issues!



- $x^- a^+$
- $y^{-1} a^+$
- $dummy_0^- a^+$

Directions only *demand* at most one token per place.
We only construct maximal directions.

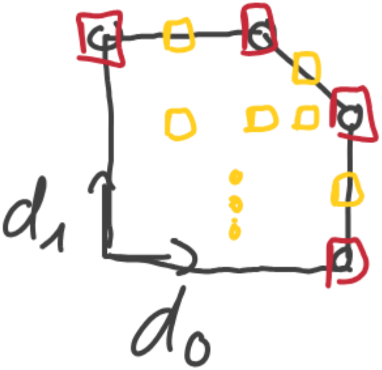
Why does it matter? # 3

This is once again a (bounded,...) polytope.

$$|c|_t \geq \sum_d \texttt{req}(d,t)n_d$$

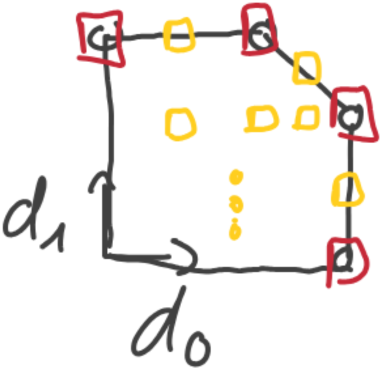
with $\texttt{req}(d,t)$ being 1 if the direction d unstarts or terminates transition t, 2 if it unstarts AND terminates t and is 0 otherwise.

Why does it matter? # 3



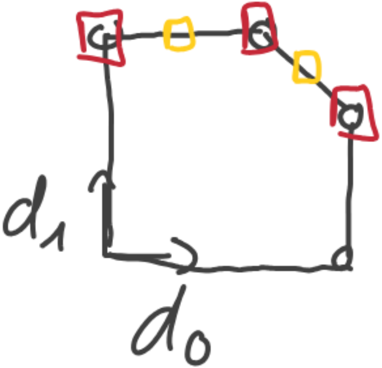
Here, all integer solutions are of interest.

Why does it matter? # 3



Here, all integer solutions are of interest. - Not only the pareto optimal ones

Why does it matter? # 3



Well except those who only terminate dummy transitions. $dummy_0^- \, dummy_1^- \, a^+$

Conclusion and Future Work

- HDAs provide detailed semantics for concurrent systems and Petri Nets.
- Fixing the event-order only solves half of the problem
  What is actually needed is to switch to anonymous HDAs.
  As for Petri Nets, the different events have no identity and can be interchanged.

- Improve data structures used for max-cell HDAs
- Improve algorithms
- Find a suitable logic for model checking working with max-cells

Conclusion and Future Work

- HDAs provide detailed semantics for concurrent systems and Petri Nets.
- Fixing the event-order only solves half of the problem
  What is actually needed is to switch to anonymous HDAs.
  As for Petri Nets, the different events have no identity and can be interchanged.

- Improve data structures used for max-cell HDAs
- Improve algorithms
- Find a suitable logic for model checking working with max-cells

Thank you for your attention!