Discrete and Continuous Models for Concurrent Systems

Uli Fahrenberg
LMF, Université Paris-Saclay, France

ISIMM, April 2026

Exercise 1 For each of the following objects: (1) make a drawing to understand it; (2) decide where it is situated in the hierarchy po-space $\hookrightarrow$ lpo-space $\hookrightarrow$ d-space; (3) draw a few dipaths.

1. The directed square $\vec{I} \times \vec{I}$
2. The "half-directed" square $\vec{I} \times I$, where I is the po-space with order $x \leq y \iff x = y$
3. The "half-twiggly" square $\vec{I} \times \tilde{I}$, where $\tilde{I}$ is the d-space with $\vec{P}\tilde{I} = \mathsf{Top}(I, I)$ (all paths directed)
4. $I \times \tilde{I}$
5. $\vec{S}_1$, where $S_1 = \{e^{it} \mid 0 \leq t \leq 2\pi\} \subseteq \mathbb{C}$ is the unit circle and $e^{it_1} \leq e^{it_2} \iff 0 \leq t_2 - t_1 \mod 2\pi < \pi$
6. $\vec{O}_1$, where $O_1 = S_1$ and $e^{it_1} \leq e^{it_2} \iff 0 \leq t_1 \leq t_2 \leq \pi$ or $\pi \leq t_2 \leq t_1 \leq 2\pi$
7. a finite automaton with language given by the regular expression $a(b + c)a$, seen as a "geometric graph" where the vertices are points and the edges, unit intervals
8. a finite automaton with language given by the regular expression $a(b + c)^*$

Solution to exercise 1

1. $\vec{I} \times \vec{I}$: po-space
2. $\vec{I} \times I$: po-space
3. $\vec{I} \times \tilde{I}$: d-space
4. $I \times \tilde{I}$: d-space
5. $\vec{S}_1$: lpo-space
6. $\vec{O}_1$: po-space
7. $a(b + c)a$: po-space
8. $a(b + c)^*$: lpo-space

For the next two exercises you will need two tools:

- A graphical editor for Petri Nets that also lets us "execute" them on the fly. We will use `wolfgang` for this:
  https://github.com/iig-uni-freiburg/WOLFGANG
  We recommend using the jar. It can be run by simply using:
  `java -jar wolfgang-1.0.3.jar`
  When launched, chose place/transition net. Once your Petri Net constructed with all its places, transitions and initial marking, you can simulate an execution by clicking `edit`. Fireable transition are marked red and are executed by clicking them. This will consume and produce the corresponding tokens. Note that `wolfgang` uses interleaving semantics: you can only fire one transition at a time.

- `pn2HDA`: A tool for translating Petri Nets into HDAs and perform several actions on them.
  First you need to install `Symmetri`:
  Clone the repo
  `git clone https://github.com/thorstink/Symmetri.git`
  Setup and install

  ```
  git submodule update --init --recursive
  mkdir build
  cd build
  # Release build
  cmake -DCMAKE_BUILD_TYPE=Release ..
  # Building and installing:
  make && sudo make install
  ```

  Once this is done you can install `pn2HDA`:

  ```
  git clone https://gitlabev.imtbs-tsp.eu/philipp.schlehuber-caissier/pn2hda.git
  cd pn2hda
  cmake -DCMAKE_BUILD_TYPE=Release -S . -B build
  cmake --build build
  ```

  Now you should be all set up. To display the ST-automaton corresponding to an HDA as pdf you will also need `dot`, usually provided by the `graphviz` package.
  If you use a local installation, `pn2HDA` below must be replaced by `/your/path/to/build/pn2HDA`.

Using pn2HDA

Given that there exists a `test.pnml` petri in your current folder you can:

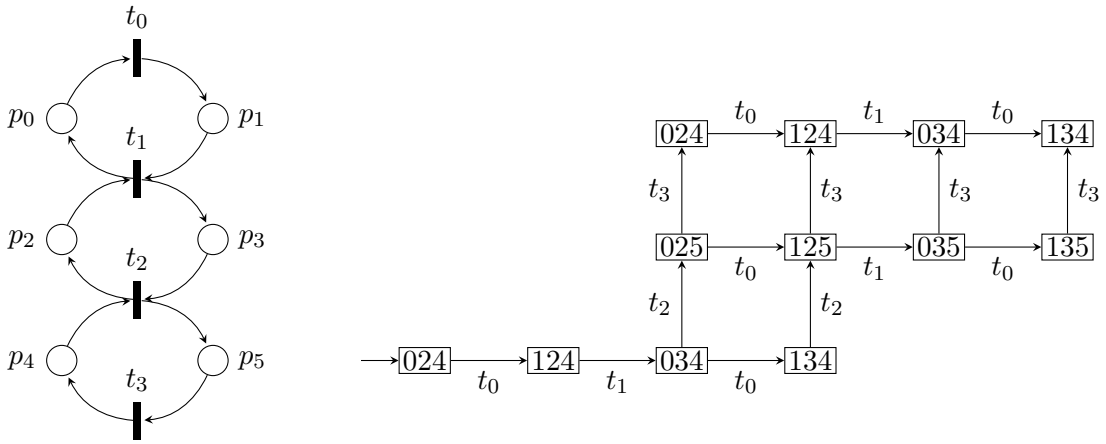
- create a pdf showing the corresponding ST-automaton like so:
  `pn2HDA standard test.pnml dot > test.dot && dot -Tpdf -o test.pdf test.dot`

- show a summary of statistics of the HDA like so:
  `pn2HDA standard test.pnml "csv_header"`

Exercise 2 Implement the "Bakery" Petri net from the lecture in `wolfgang` and simulate its steps. Save it as `bakery.pnml` and have a look at that file in a text editor.

Run the net until you have explored all reachable markings. Draw the reachability graph of the net and its HDA semantics. Use `pn2hda` to confirm your results.

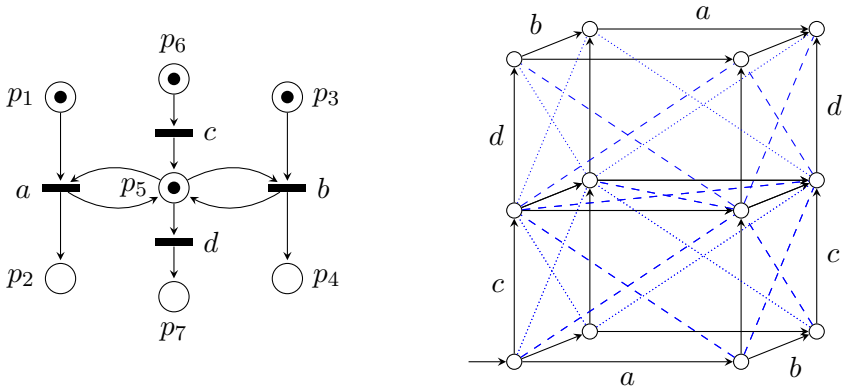
Solution to exercise 2 The Petri net, redrawn with new names, and its reachability graph (markings indicated by indices of places with token (all places are 1-bounded); states with the same name are identified: this is a 2D loop):



The concurrent step reachability graph is the same but with multi-step transitions in all squares, and the HDA has all squares filled in.

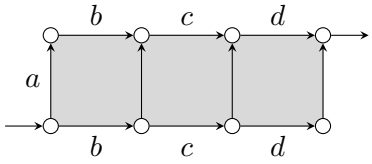
Exercise 3 Implement the last example of the lecture in `wolfgang` and translate it to an ST-automaton. Add a new place and a transition d which *disables* the concurrency of a and b. Repeat the translation to an ST-automaton. What does the corresponding HDA look like?

Solution to exercise 3 One possibility:



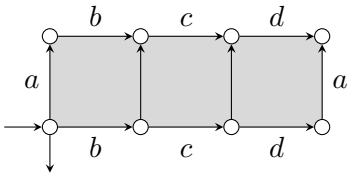
(This is not quite correct: in the Petri net, d can eat the token in p_5 before c can ever enable concurrency of a and b; in the HDA this is not possible. Amelioration: add a p_8 to the net with no tokens and arcs $c \to p_8 \to d$.)

Exercise 4 What is the language of the HDA below?



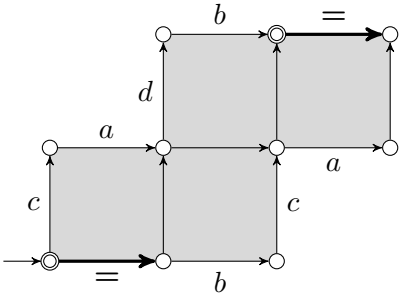
Solution to exercise 4 $\{[\,{}^{a}_{b\to c\to d}]\}{\downarrow}$

Exercise 5 What is the language of the HDA below? Horizontal cells with the same label are identified, as are the left and right vertical cells, and the initial cell is accepting. (Geometrically this is a torus.)



Solution to exercise 5 It's complicated...

Exercise 6 Give a few ipomsets in the language of the following HDA, with the cells marked "=" identified.



Solution to exercise 6

$$\left[\begin{array}{c} a \longrightarrow b \\ c \longrightarrow d \end{array}\right]$$

$$\left[\begin{array}{c} a \longrightarrow b \longrightarrow a \longrightarrow b \\ c \longrightarrow d \longrightarrow c \longrightarrow d \end{array}\right]$$

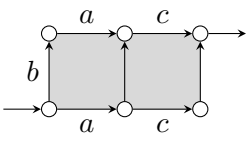
$$\left[\begin{array}{c} a \longrightarrow b \longrightarrow a \longrightarrow b \longrightarrow a \longrightarrow b \\ c \longrightarrow d \longrightarrow c \longrightarrow d \longrightarrow c \longrightarrow d \end{array}\right]$$

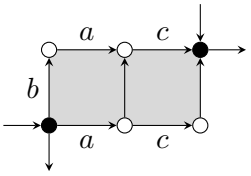
...

Exercise 7 Construct HDAs which correspond to the following rational expressions.

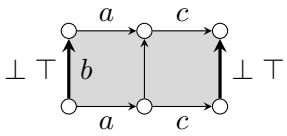
1. $\left[\begin{smallmatrix} b\bullet \\ a \end{smallmatrix}\right]\left[\begin{smallmatrix} \bullet b \\ c \end{smallmatrix}\right]$
2. $\left(\left[\begin{smallmatrix} b\bullet \\ a \end{smallmatrix}\right]\left[\begin{smallmatrix} \bullet b \\ c \end{smallmatrix}\right]\right)^+$
3. $\left(\left[\begin{smallmatrix} \bullet b\bullet \\ a \end{smallmatrix}\right]\left[\begin{smallmatrix} \bullet b\bullet \\ c \end{smallmatrix}\right]\right)^+$
4. $\left(a\left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right] + \left[\begin{smallmatrix} a \\ a \end{smallmatrix}\right]a\right)^+$

Solution to exercise 7

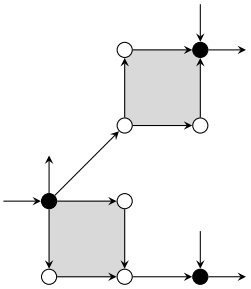
1. 

2. 

(black states identified)

3. 

(thick b-labeled edges identified and also both initial and accepting)

4. 

(black states identified, all edges labeled a)

Exercise 8 Let's look a bit closer at the HDA of the last part of the last exercise, for $(a\,[\,{}^{a}_{a}\,] + [\,{}^{a}_{a}\,]\,a)^{+}$. Call it X.

1. Let $n \geq 1$. How many paths in X recognize the word $(aaa)^n$?
2. Does there exist another HDA Y with $L(Y) = (a\,[\,{}^{a}_{a}\,] + [\,{}^{a}_{a}\,]\,a)^{+}$ and in which fewer paths recognize the words $(aaa)^n$?
3. Conclude that the language $(a\,[\,{}^{a}_{a}\,] + [\,{}^{a}_{a}\,]\,a)^{+}$ is inherently infinitely ambiguous.

Solution to exercise 8

1. There are 4^n: at every iteration of aaa, two paths in the upper part and two paths in the lower part.
2. It seems not:
   - At the initial state we need to have a branching between a and $[\,{}^{a}_{a}\,]$, otherwise X would recognize $[\,{}^{a}_{a}\,][\,{}^{a}_{a}\,]$.
   - After the a-transition in the upper part, we need a square for $[\,{}^{a}_{a}\,]$, and this needs to be separate from the lower part, otherwise we would again recognize $[\,{}^{a}_{a}\,][\,{}^{a}_{a}\,]$.
3. We have constructed an argument that any HDA recognizing $(a\,[\,{}^{a}_{a}\,] + [\,{}^{a}_{a}\,]\,a)^{+}$ must have an exponential number of accepting paths for the word $(aaa)^n$. That is, for every $N \geq 0$ there exists a pomset in $(a\,[\,{}^{a}_{a}\,] + [\,{}^{a}_{a}\,]\,a)^{+}$ which has at least N accepting paths: this is the definition of "inherently infinitely ambiguous."