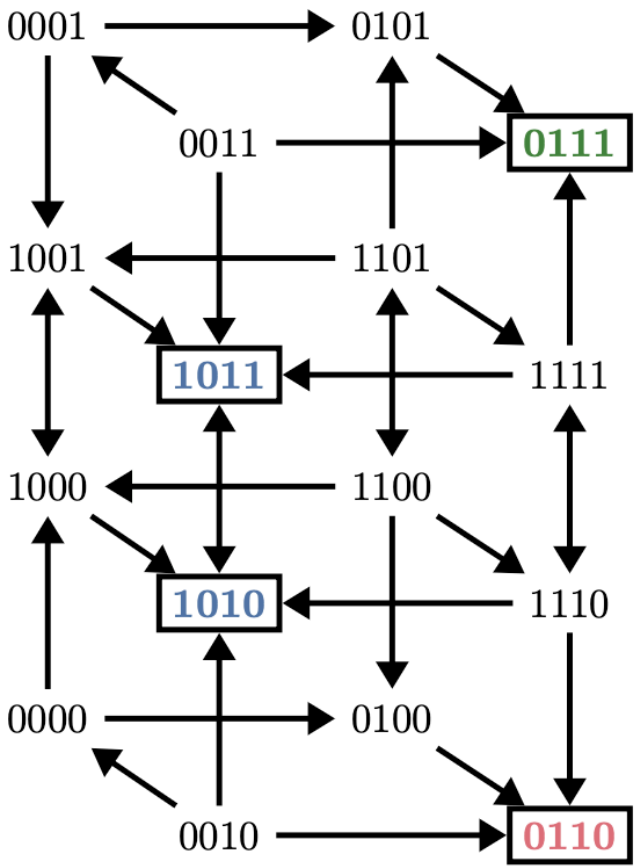
Mapping attractor landscapes in Boolean networks with Biobalm

Van-Giang Trinh, Kyu Hyong Park, Samuel Pastva, Jordan Rozum

Motivation: Boolean network attractors

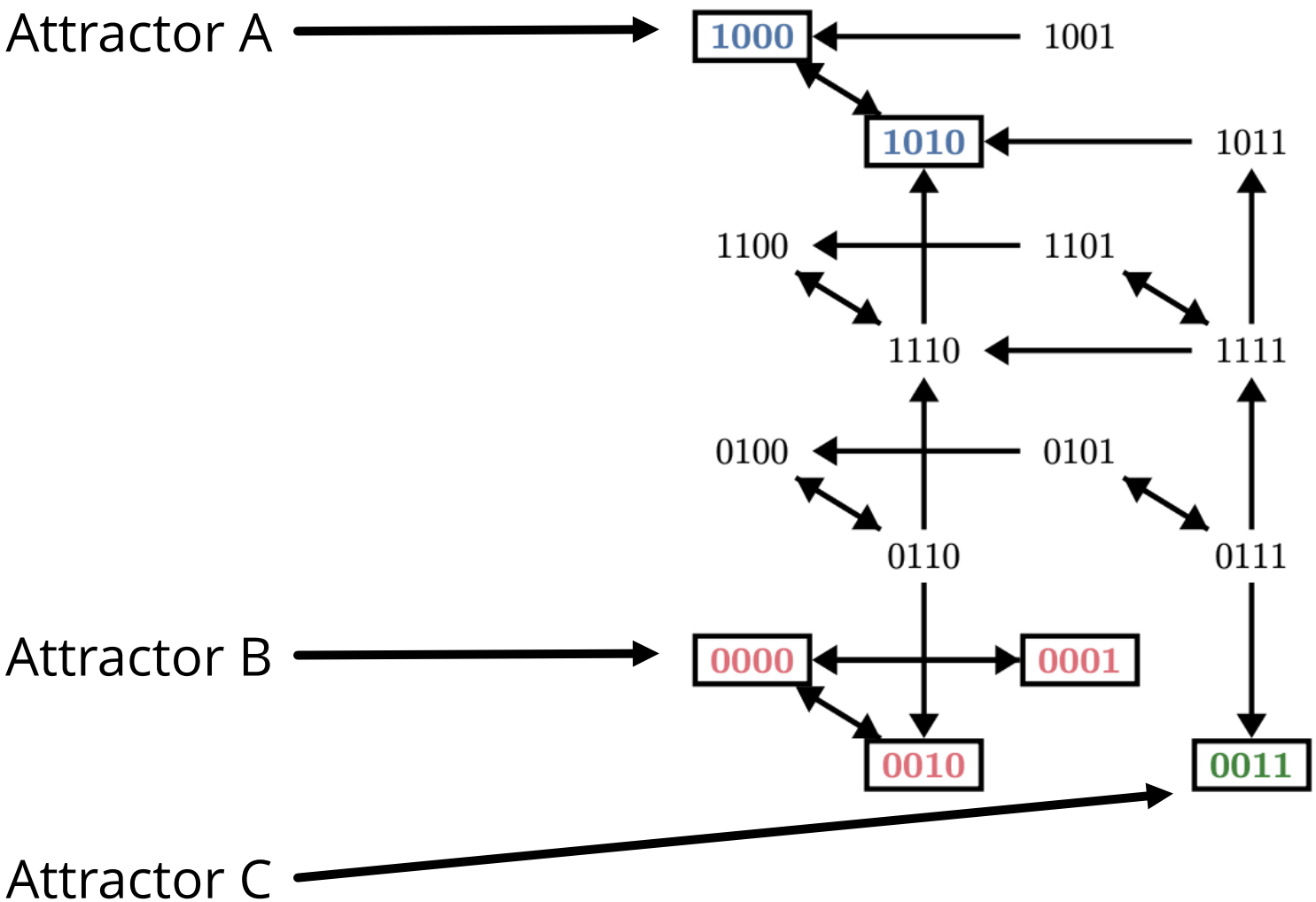
$$f_1(x) = \neg x_2$$
$$f_2(x) = \neg x_1$$
$$f_3(x) = x_1 \vee x_2$$
$$f_4(x) = x_1 \text{ xor } x_4$$

(a)

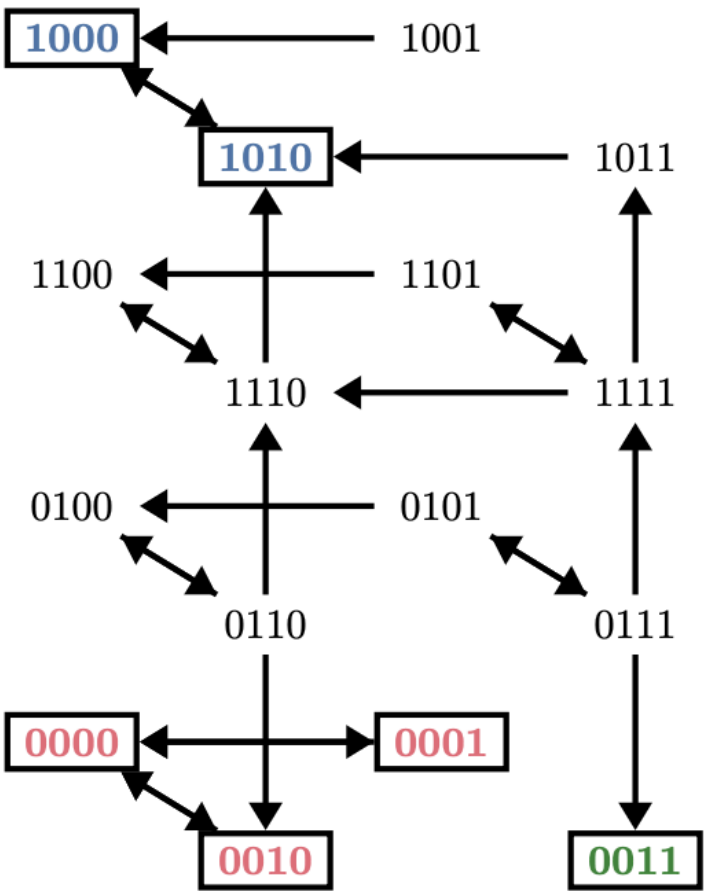


(b)

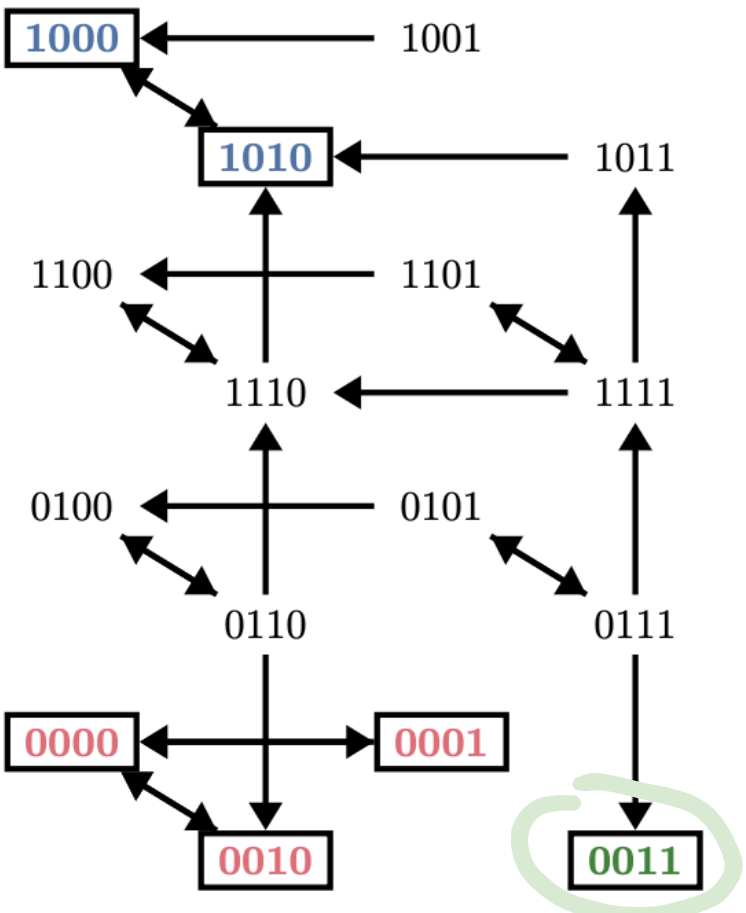
Trap spaces vs. attractors



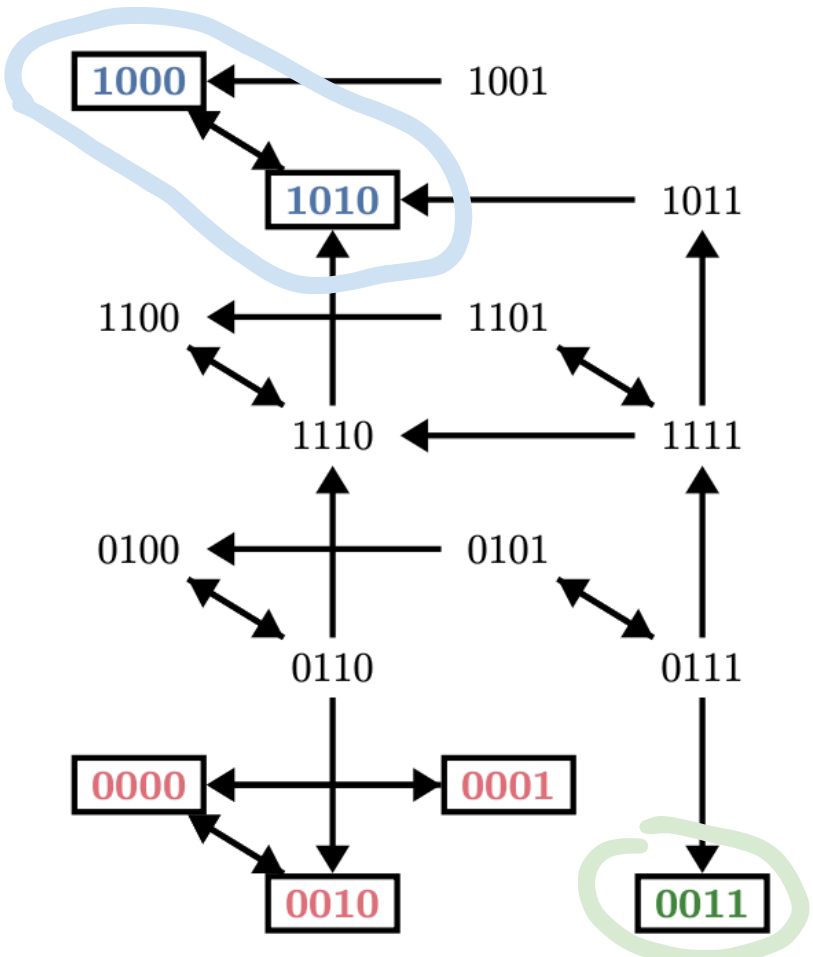
Trap spaces vs. attractors



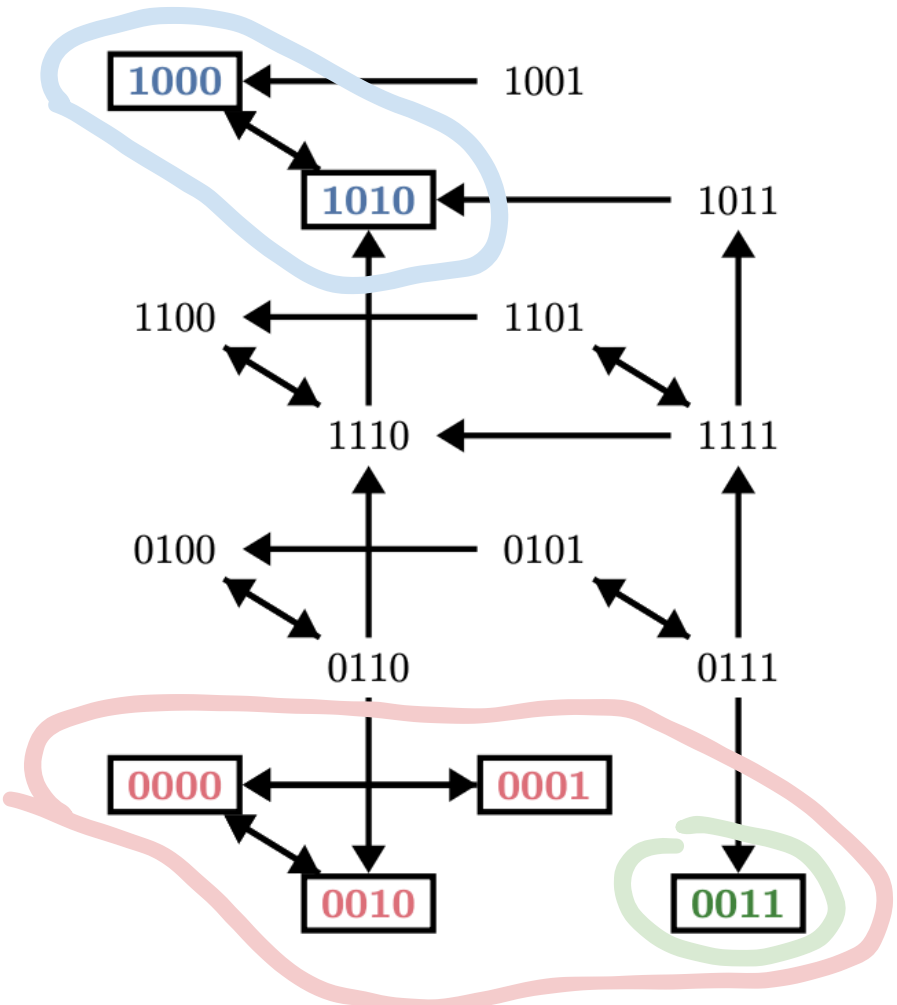
Trap spaces vs. attractors



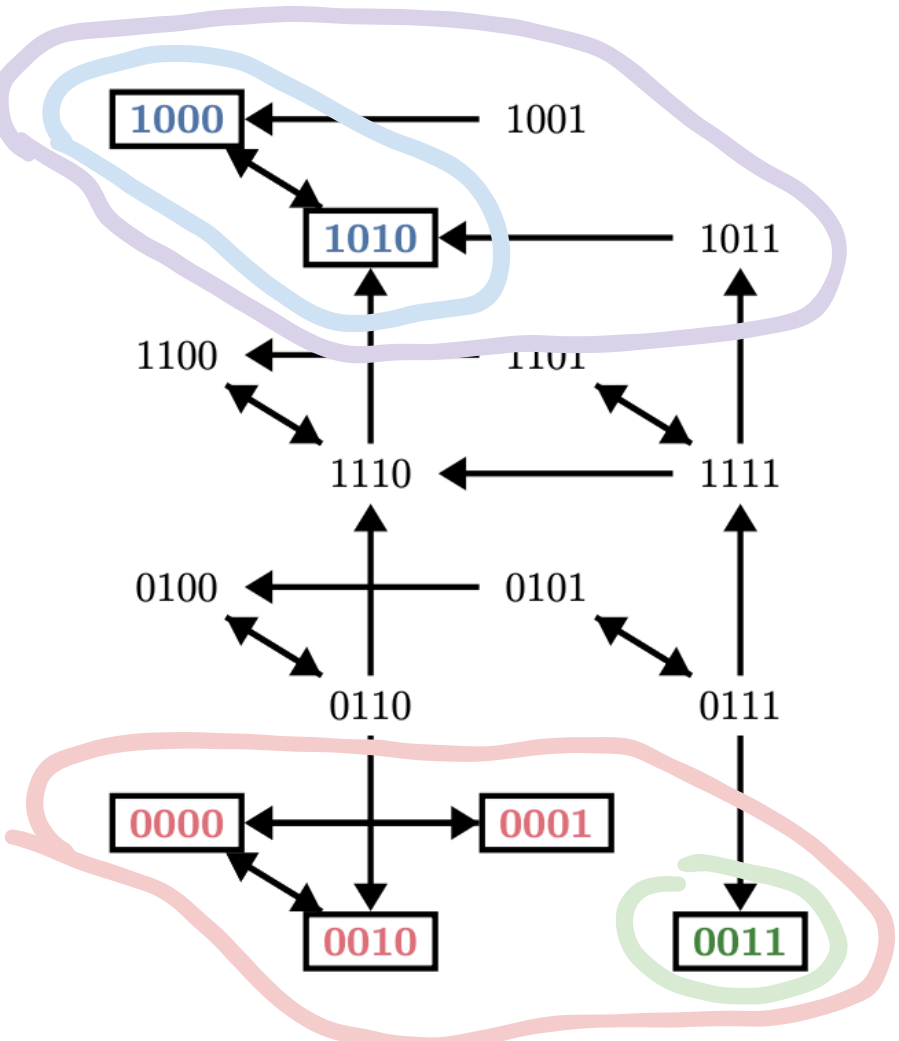
Trap spaces vs. attractors



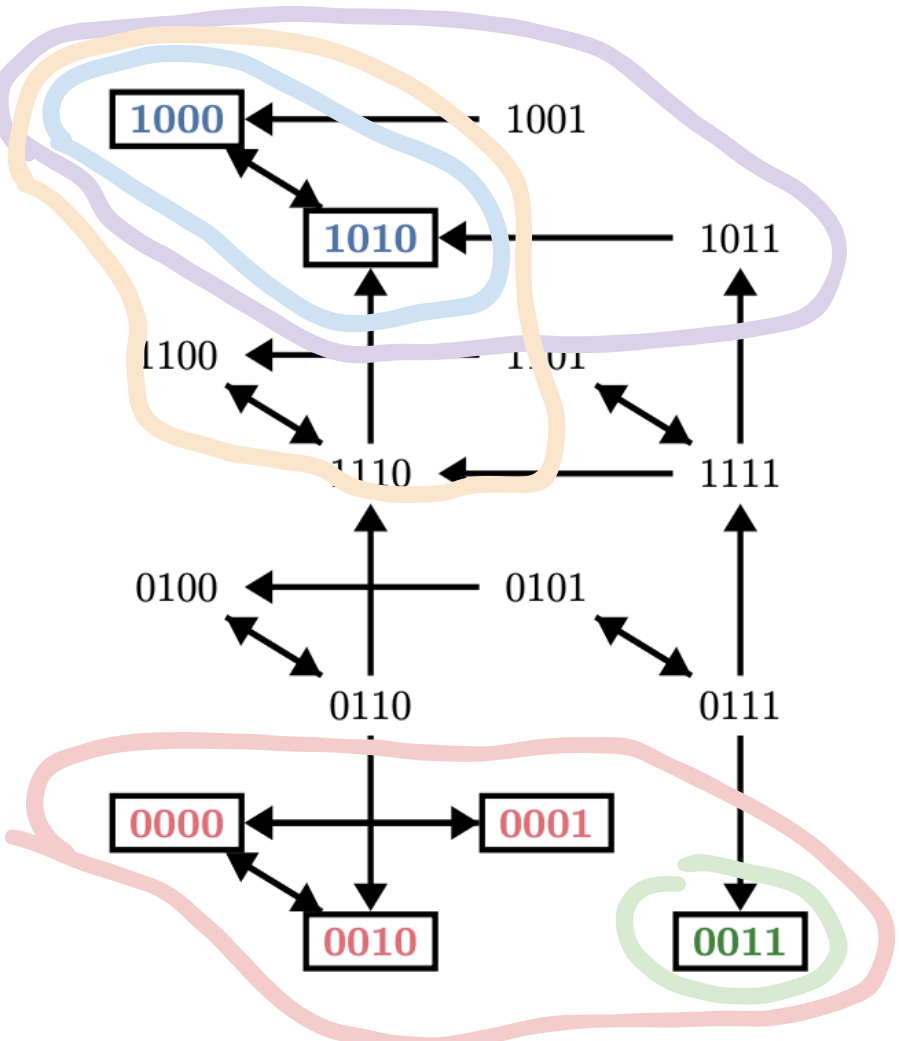
Trap spaces vs. attractors



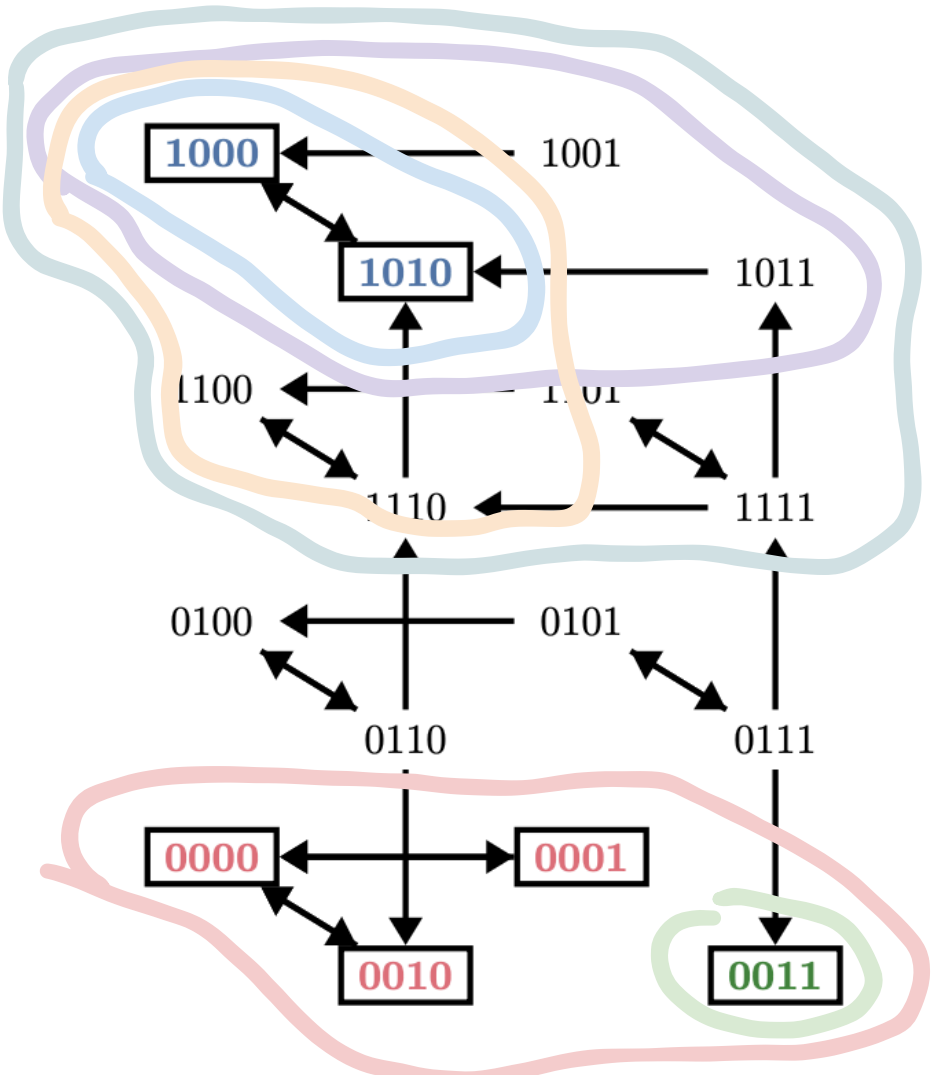
Trap spaces vs. attractors



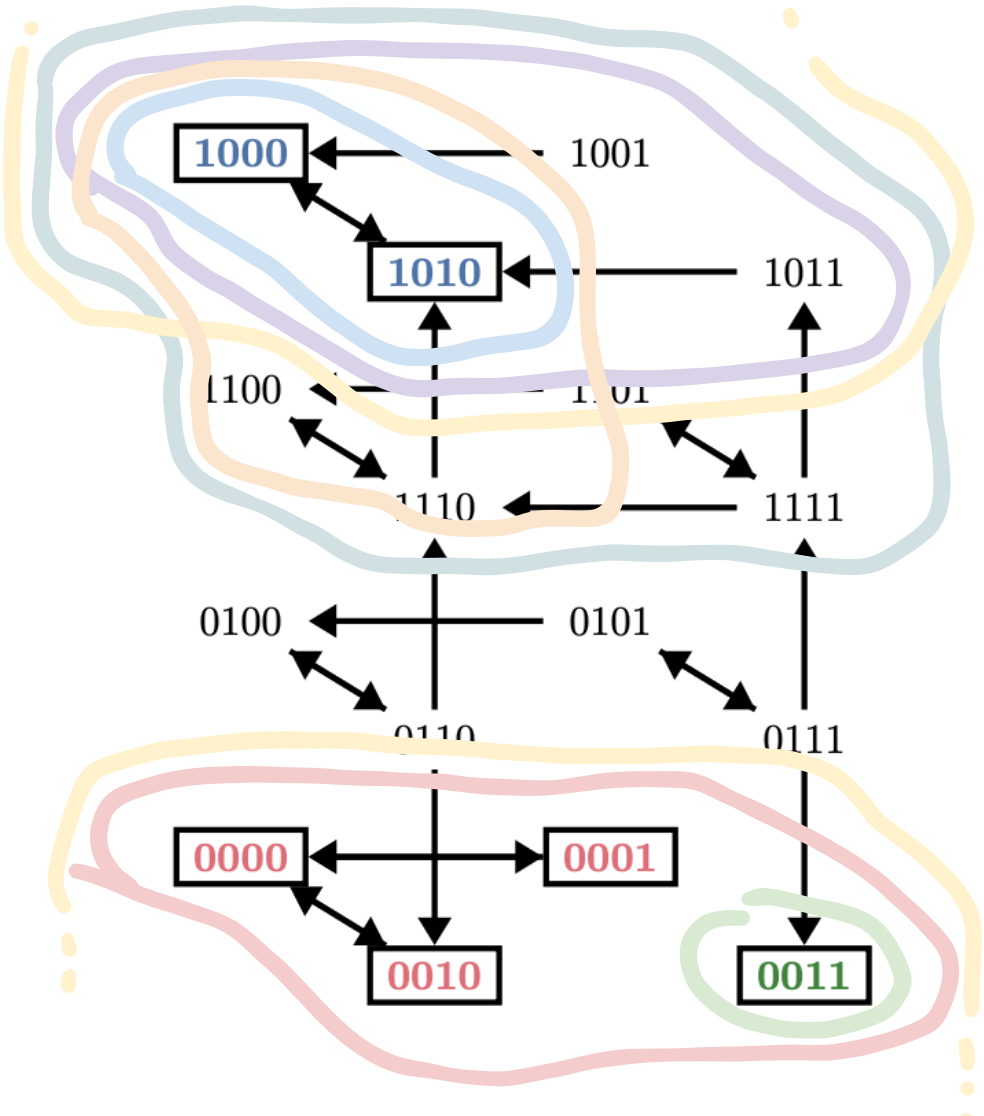
Trap spaces vs. attractors



Trap spaces vs. attractors



Trap spaces vs. attractors



Trap space inclusion hierarchy

- "Minimal" trap space *Min*: no smaller trap space exists inside *Min*.
- "Maximal" trap space *Max* inside *X*: no larger trap space exists inside *X*.
- Trap space percolation: Variable *A* can be percolated to Boolean value *B* in trap space *X* if *F_A(x) = B* for every state that belongs to *X*.
  - 1*** percolates to 1**0 in the last example.
  - 11** percolates to 10*0 in the last example.
- "Fully percolated" trap space: A trap space where no variable can be percolated.
- **Succession diagram**: Rooted oriented acyclic graph describing the hierarchy of percolated trap spaces.

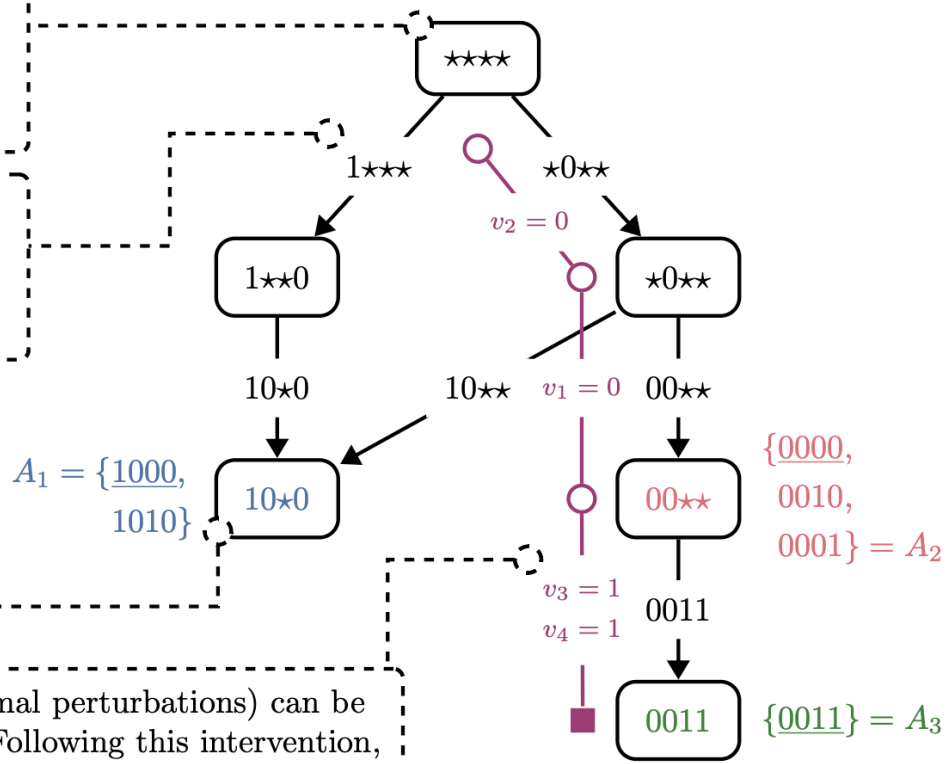
Biobalm: a tool for computing succession diagrams of large networks.

Biobalm features

Section 3.1: Succession diagram [21] is built from *percolated* trap spaces, ordered by set inclusion. These are computed by adapting the ASP method of [50]. Here, ⋆⋆⋆⋆ contains two maximal percolated trap spaces: 1⋆⋆0 and ⋆0⋆⋆.

Section 3.1: Each edge is labeled with the maximal trap spaces that percolate to the target node (*stable motifs*). For this purpose, `biobalm` provides an improved, symbolic percolation method. Here, trap space 1⋆⋆⋆ percolates to 1⋆⋆0.

Section 3.2: For every node, `biobalm` computes the *attractor seed states* (underlined) and optionally the full attractor set. This is achieved by improving the method of [30], and through a partial expansion strategy that targets motif-avoidant attractors (Section 3.1.1; Figure 3). Here, trap space 10⋆0 admits a cyclic attractor 1000 ↔ 1010.



Supplementary Text S4: A control intervention (series of minimal perturbations) can be computed using the method of [29] for any target (here 0011). Following this intervention, the network always reaches the target, regardless of the initial state or the update scheme.

Biobalm: computing trap spaces

Based on answer set programming [*]: The network functions are encoded as a set of logical rules that are then solved by an ASP solver. Depending on the solver configuration, we obtain max. or minimal trap spaces.

We improved [*] by using a more efficient method for generating the logical rules through binary decision diagrams.

[*] Trinh, Van-Giang, et al. "Minimal trap spaces of logical models are maximal siphons of their petri net encoding." *International Conference on Computational Methods in Systems Biology*. Cham: Springer International Publishing, 2022.

Biobalm: building the diagram

The successors of node (trap space) *X*: Compute maximal trap spaces *M_1, ... , M_k*. For each *M_i*, compute its fully percolated subspace *P(M_i)*, these are the successors.

We can now build/explore the succession diagram as any other graph. BFS, DFS, various "objective" guided expansion strategies, etc.

- "Unexpanded" node: A node for which we have not computed successors yet.
- "Skipped" node: A node for which we have not computed successors, but instead link it directly to the minimal trap spaces in that node.

Biobalm: building the diagram

The successors of node (trap space) *X*: Compute maximal trap spaces *M_1, ... , M_k*. For each *M_i*, compute its fully percolated subspace *P(M_i)*, these are the successors.

We can now build/explore the succession diagram as any other graph. BFS, DFS, various "objective" guided expansion strategies, etc.

- "Unexpanded" node: A node for which we have not computed successors yet.
- "Skipped" node: A node for which we have not computed successors, but instead link it directly to the minimal trap spaces in that node.

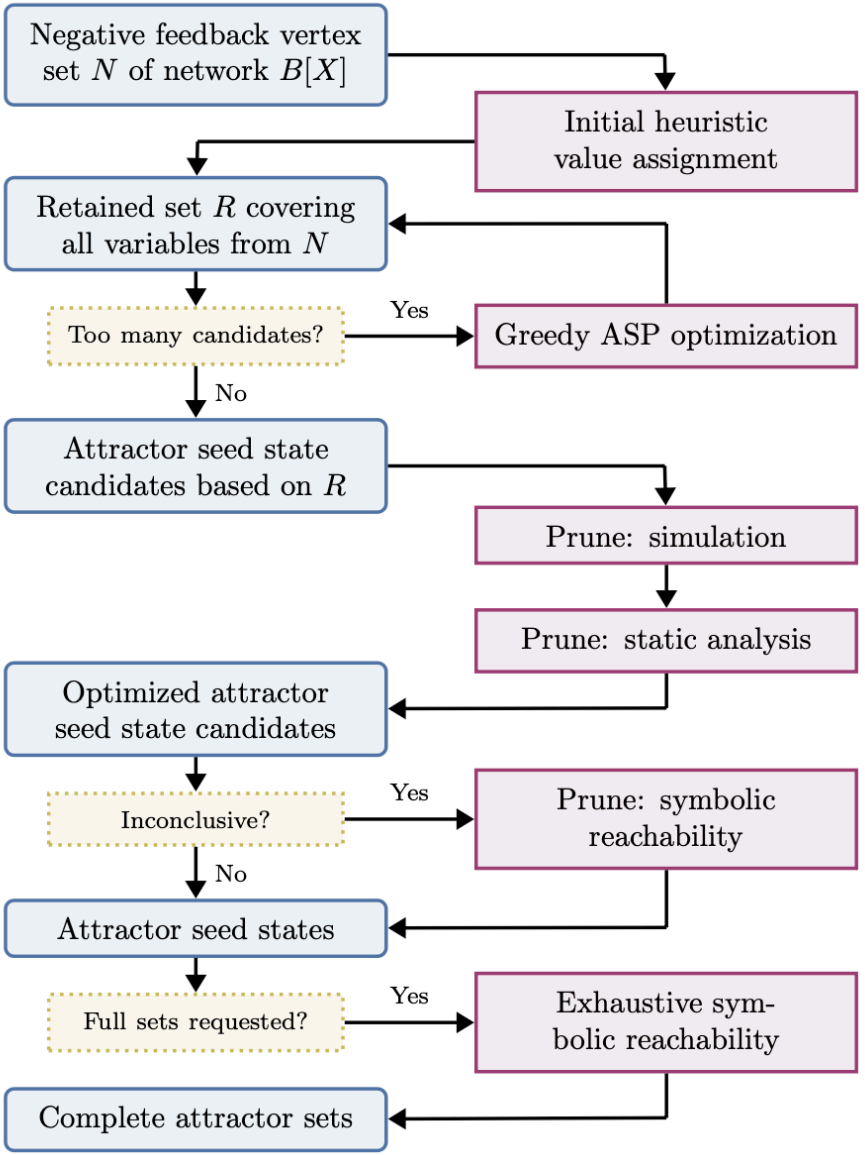
If the succession diagram is too large, we can skip some nodes to make it smaller.

Biobalm: Searching for attractors

Every attractor has a smallest enclosing trap space *X*. We only need to consider variables that are free in *X*.

That trap space is always fully percolated, so it must be in the succession diagram.

By fixing variables of its negative feedback set, every complex attractor turns into one or more fixed-points [*]. From these, we can select one *attractor seed*.



[*] Van Giang, Trinh, Tatsuya Akutsu, and Kunihiko Hiraishi. "An FVS-based approach to attractor detection in asynchronous random Boolean networks." *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19.2 (2020): 806-818.

What if we can't build the full diagram?

Source block expansion

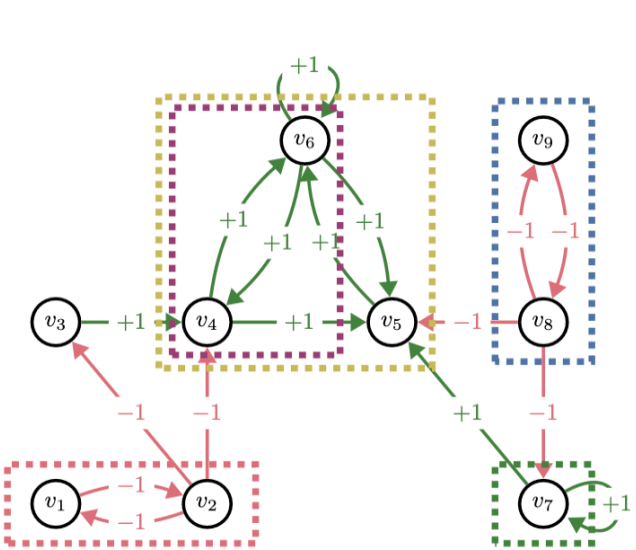
Idea: Some groups of maximal trap spaces are "independent" of the remaining trap spaces. If we only explore one "independent group" (*block*), we can still combine it with the remaining trap spaces in the successor nodes.

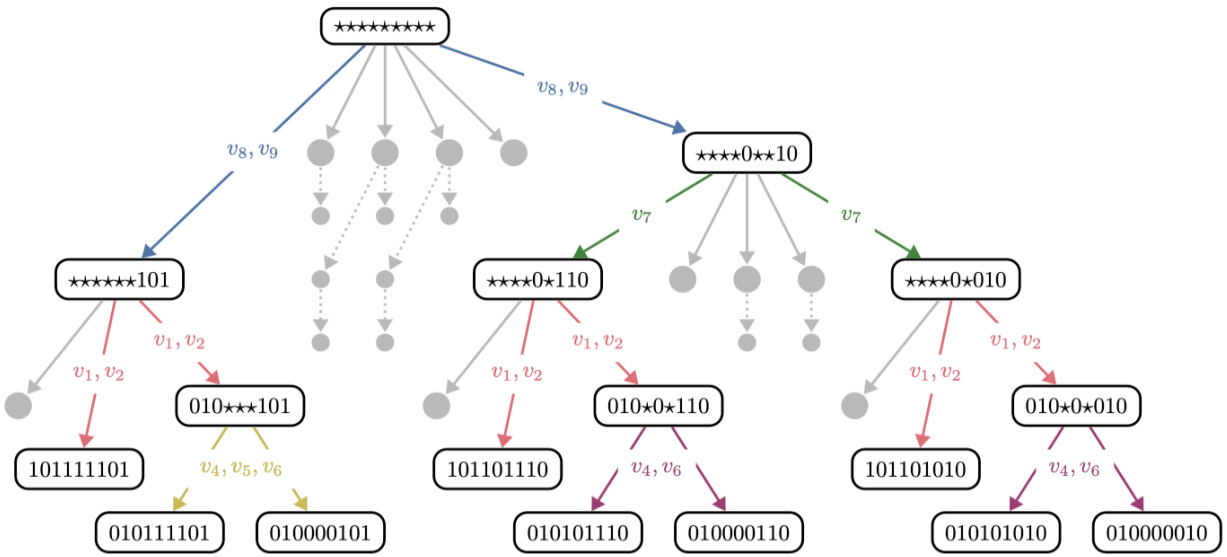
Extreme example:

$$f_1(x) = x_1$$
$$f_2(x) = x_2$$
$$f_3(x) = x_1 \wedge x_2$$

Maximal trap spaces in *** are 1**, 0**, *1*, and *0*. But in 1**, we can still fix x_2 to 0/1 (because x_1 and x_2 don't depend on each other), so we have not lost any choices, we only "saved them for later".

Real-world example

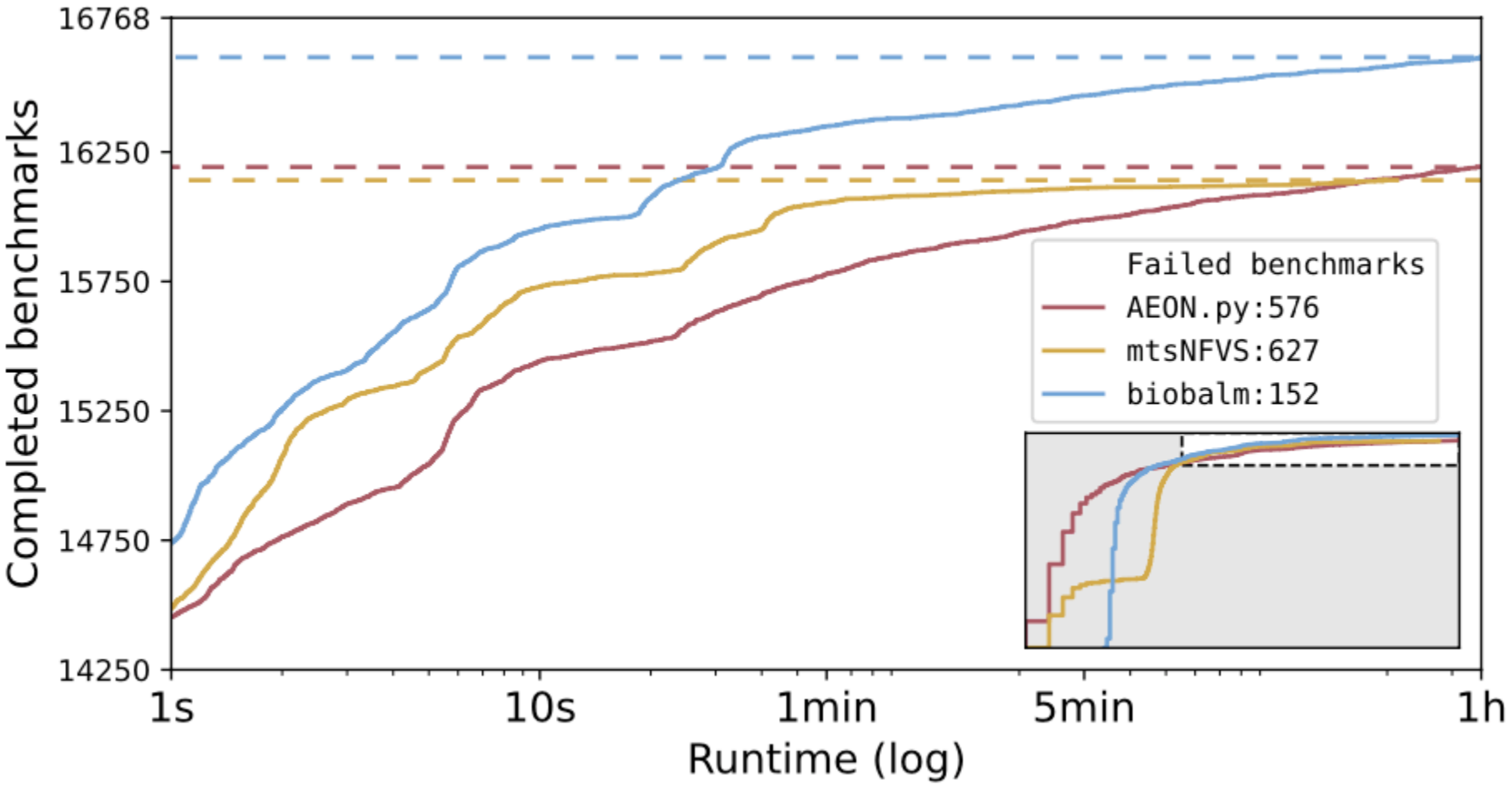




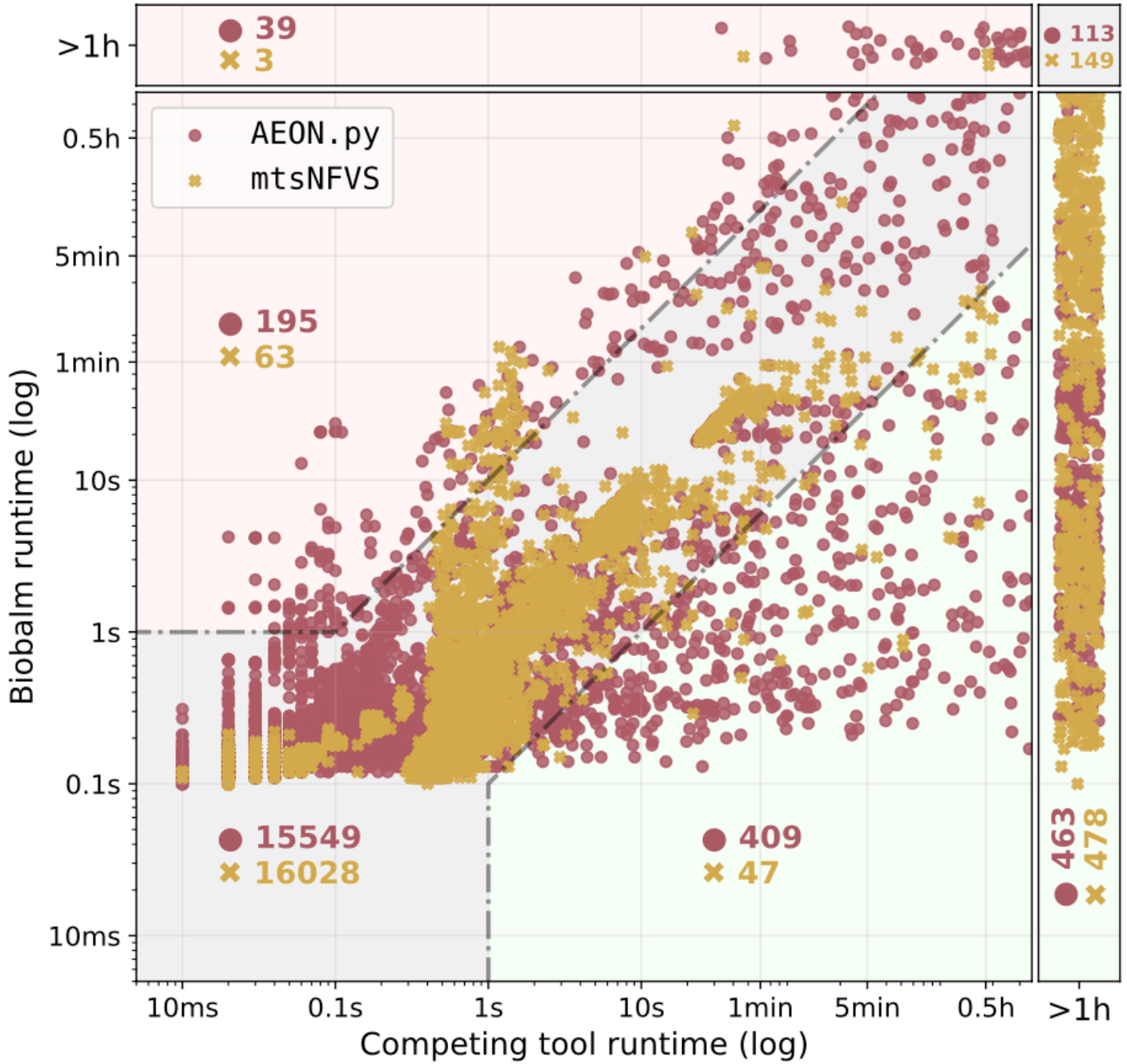
Why does this work?

- The trap spaces of one independent group always contain all minimal trap spaces.
  - ⇒ No "outcomes" are lost by only expanding one independent group.
  - ⇒ All attractors in minimal trap spaces are eventually discovered.
- Attractors of the sub-network defined by the group variables map to the attractors of the whole network.
  - ⇒ We can check the smaller network for attractors w.r.t. the trap space that is being expanded.
  - ⇒ If it has no such attractors, we can skip this trap space during attractor detection. Furthermore, no attractors can appear in the nodes we did not expand (if we find an attractor in the sub-network, we still have to expand everything).

Is it fast?



230 * (up to) 128 real-world networks + ~3000 random networks of various types (1.000+ variables)


AEON.py
mtsNFVS
39
3
113
149
195
63
15549
16028
409
47
463
478
>1h
0.5h
5min
1min
10s
1s
0.1s
10ms
Biobalm runtime (log)
10ms
0.1s
1s
10s
1min
5min
0.5h
>1h
Competing tool runtime (log)

A few observations...

- Only one model (BBM-004) has a diagram that is too large to enumerate with the block expansion.
  - In some of these cases, AEON.py still wins because it does not have to build the diagram.
- Random networks have smaller succession diagrams.
  - ⇒ Getting "interesting" behavior by chance is hard.
- Most often, two different maximal trap spaces also have different percolations. But in some random networks, we get 10.000+ maximal trap spaces that all percolate to one trap space.
- Even if you can't actually compute all attractors, having the succession diagram can be useful to narrow down what parts of the network to focus on.