MPRI 2.19 Biochemical Programming

Rule-based Modeling

Syntax, semantics, simulation

Jérôme Feret

DIENS (ÉNS, CNRS, INRIA, PSL)

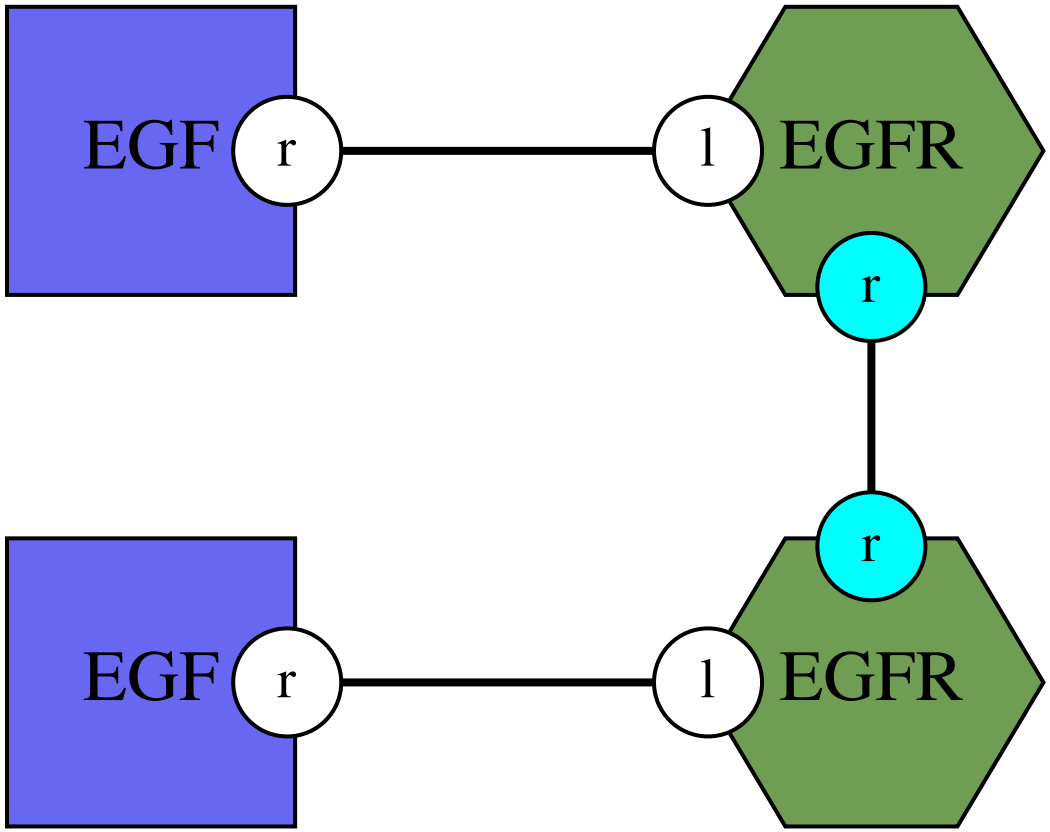
kappalanguage.org

Monday, the 4th of December, 2023

Overview

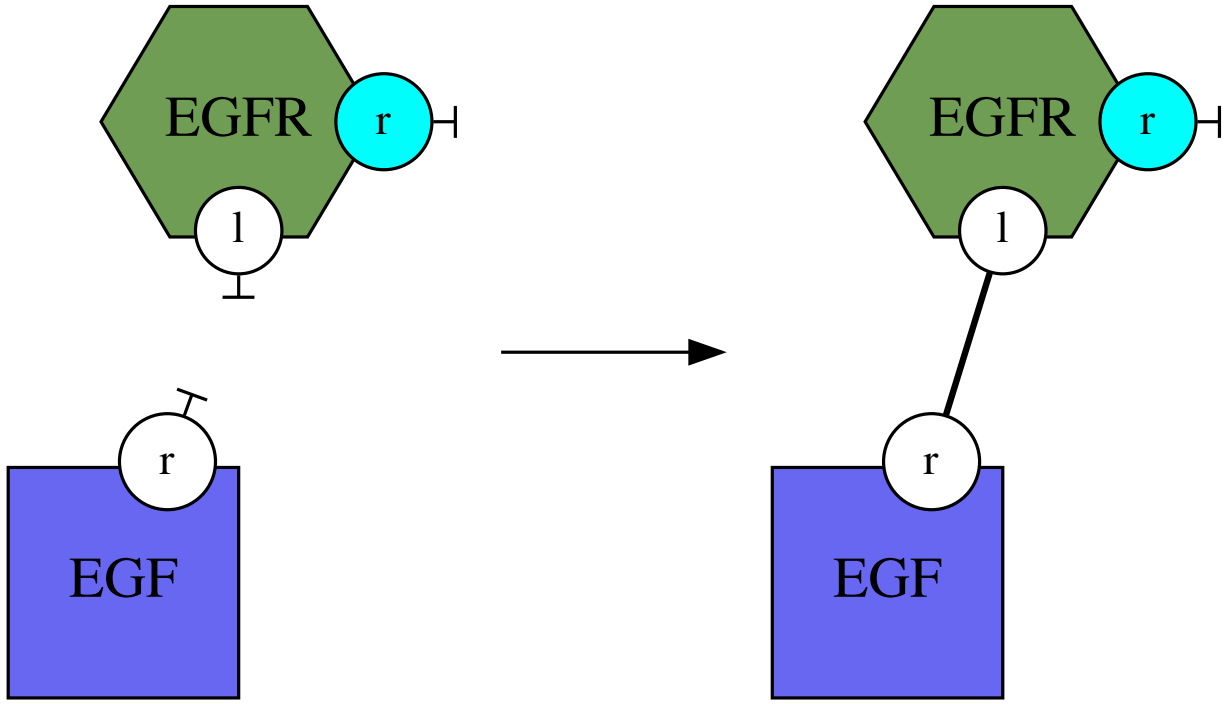
1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

A chemical species



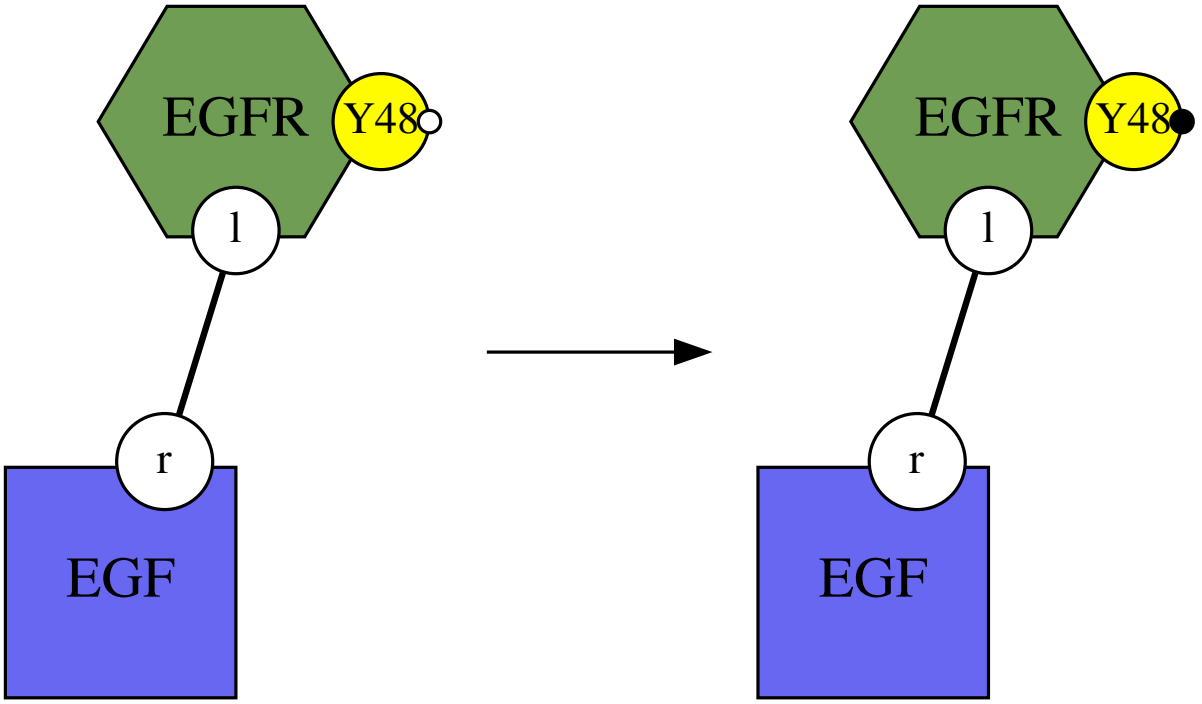
EGF(r[1]), EGFR(l[1],r[2]), EGFR(r[2],l[3]), EGF(r[3])

A Unbinding/Binding Rule



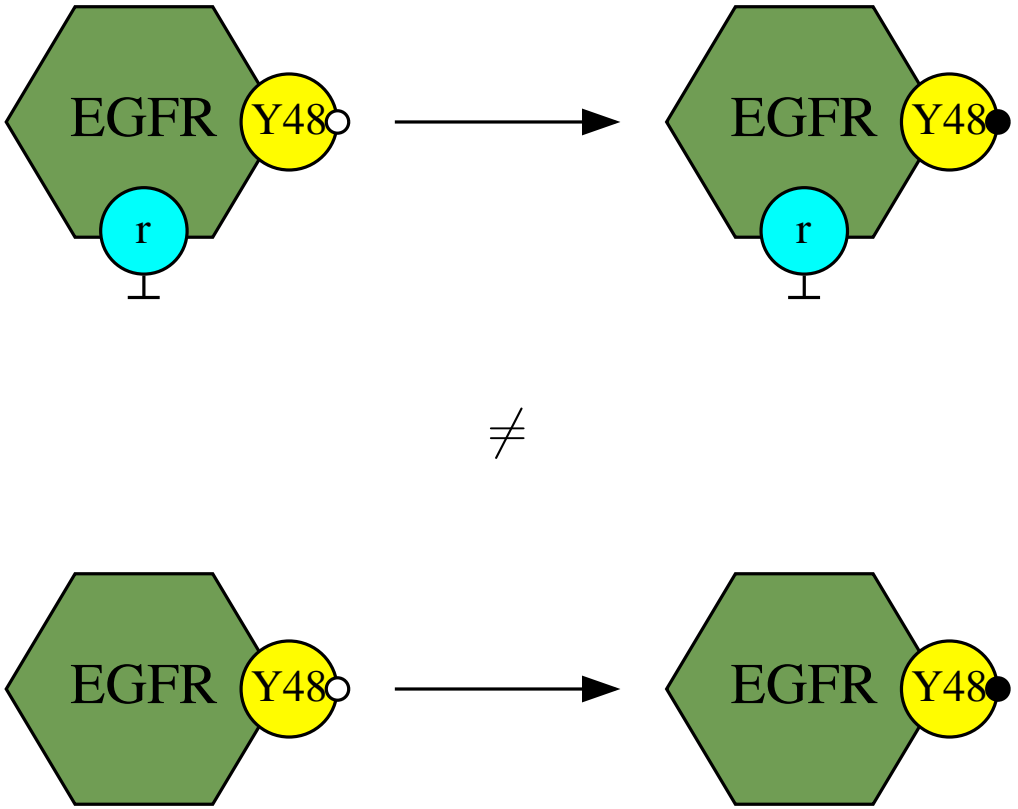
EGF(r[.]), EGFR(l[.],r[.]) -> EGF(r[1]), EGFR(l[1],r[.])

Site properties

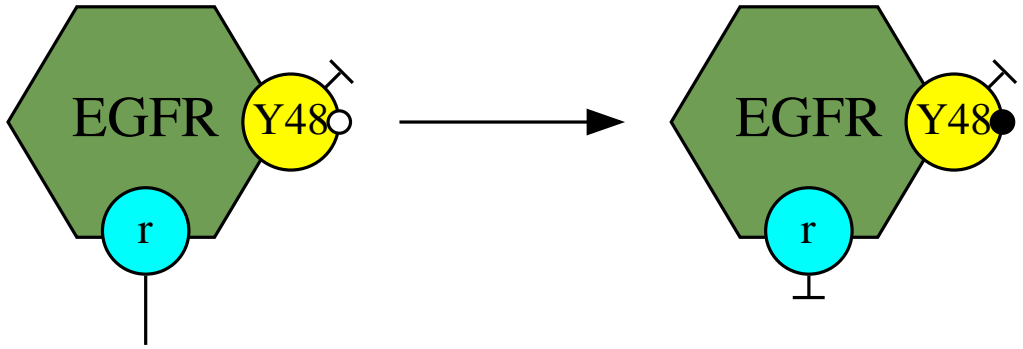


EGFR(Y48{u},l[1]), EGF(r[1]) -> EGFR(Y48{p},l[1]), EGF(r[1])

Don’t care, Don’t write

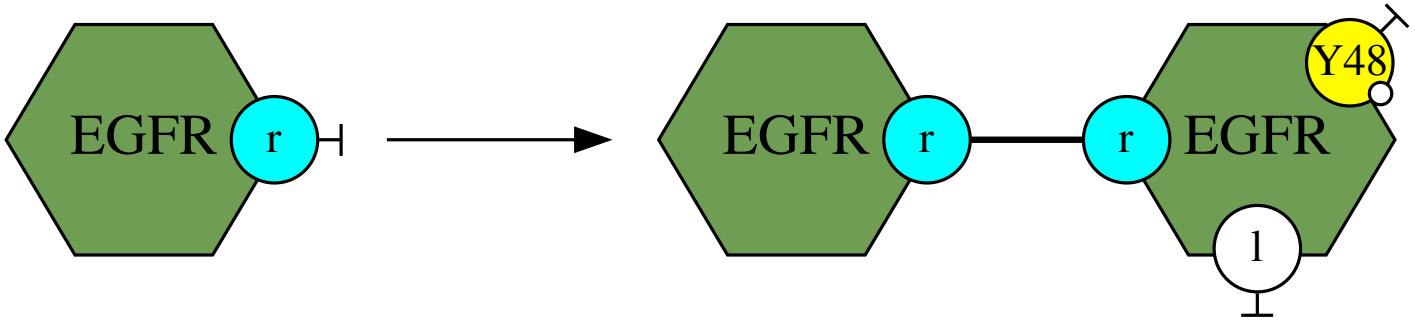


A contextual rule

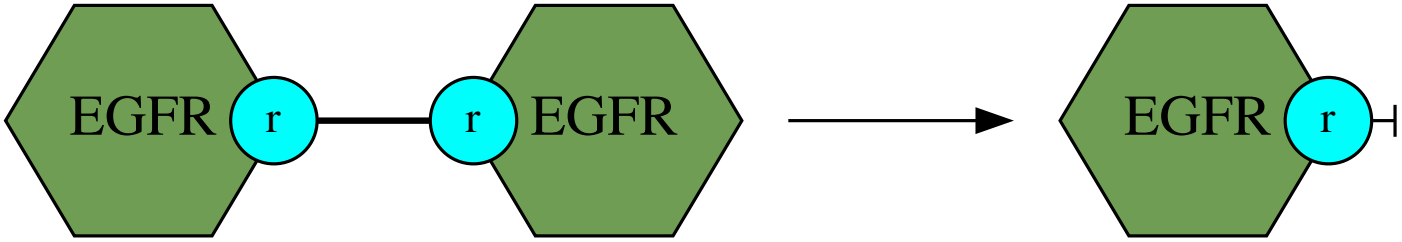


EGFR(Y48{u}[.],r[_]) -> EGFR(Y48{p}[.],r[.])

Creation/Suppression

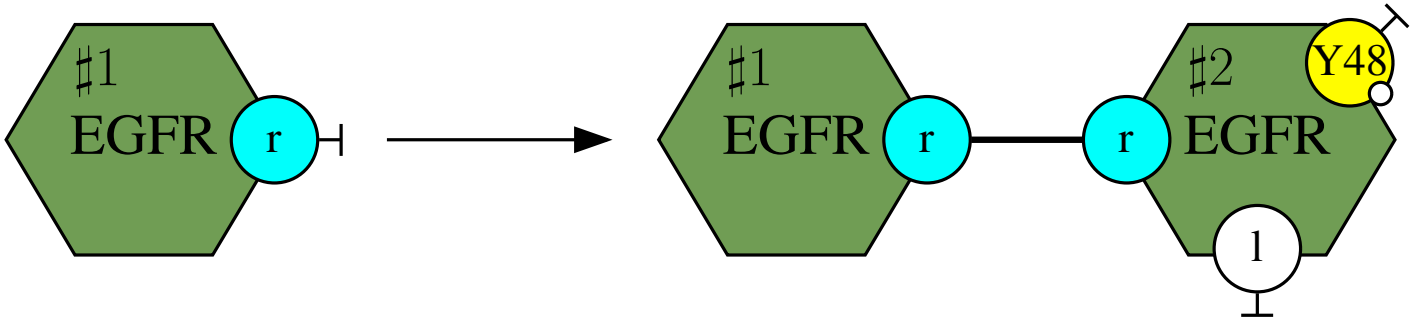


EGFR(r[.]),. -> EGFR(r[1]), EGFR(r[1],l[.],Y48{u}[.])

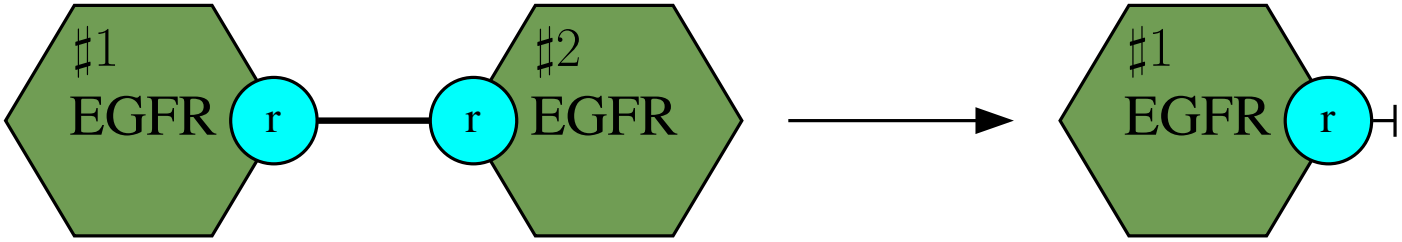


EGFR(r[1]), EGFR(r[1]) -> EGFR(r[.]),.

Creation/Suppression

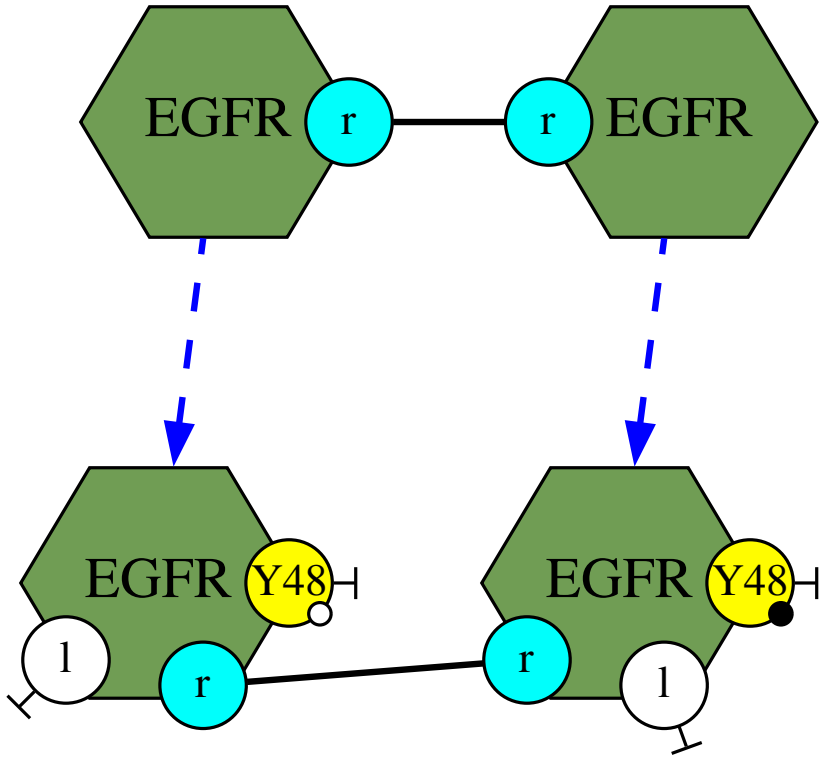


EGFR(r[.]),. -> EGFR(r[1]), EGFR(r[1],l[.],Y48{u}[.])

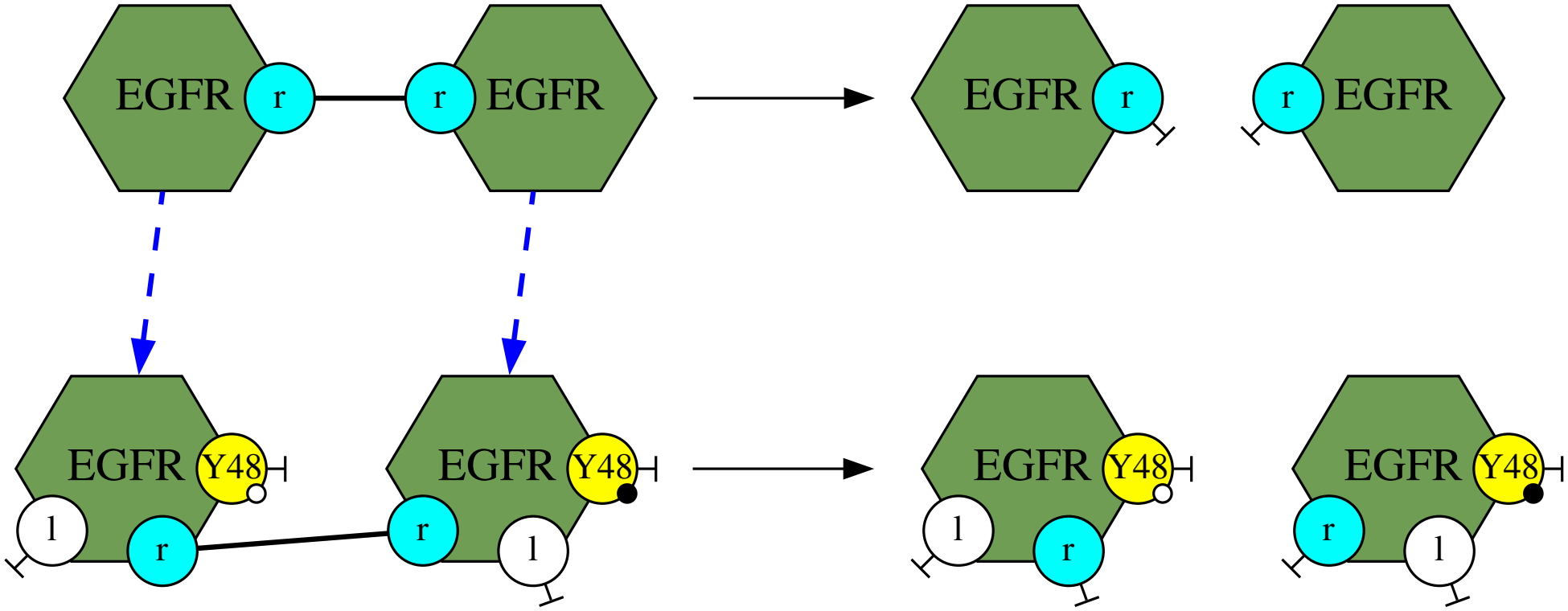


EGFR(r[1]), EGFR(r[1]) -> EGFR(r[.]),.

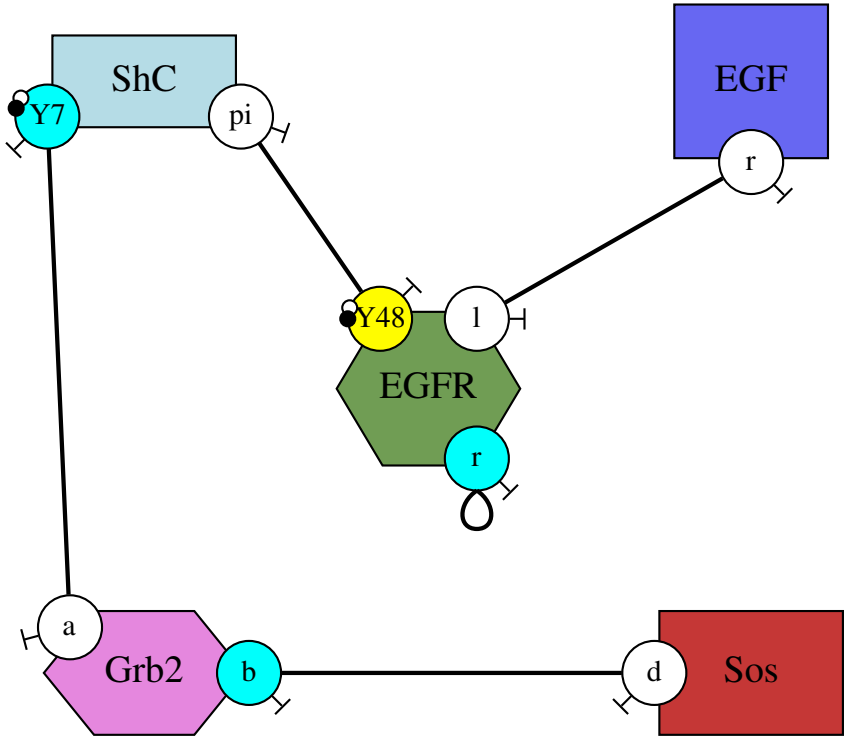
Embedding



Rule application

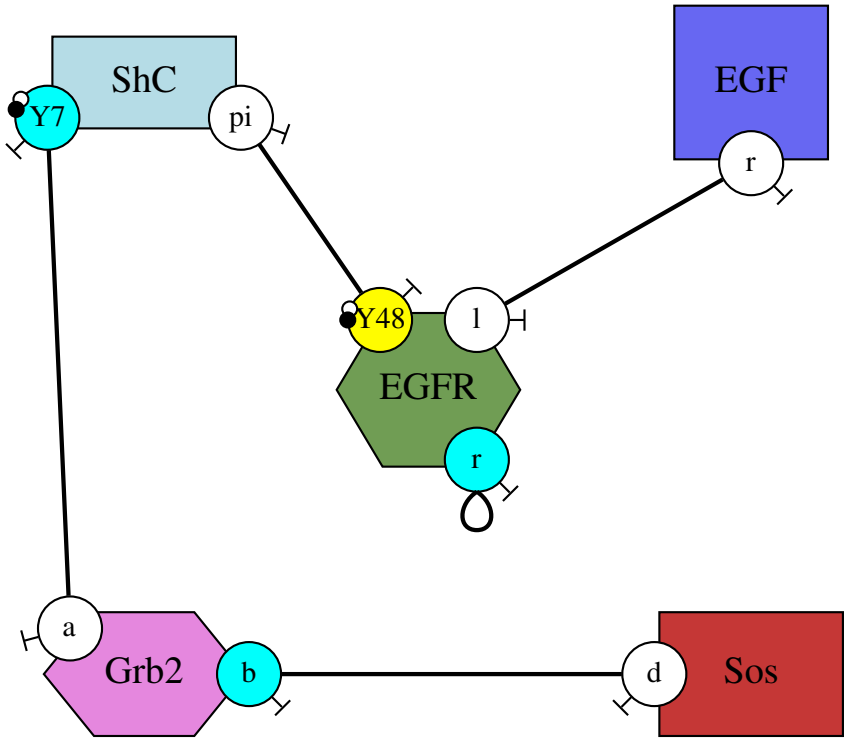


Contact map



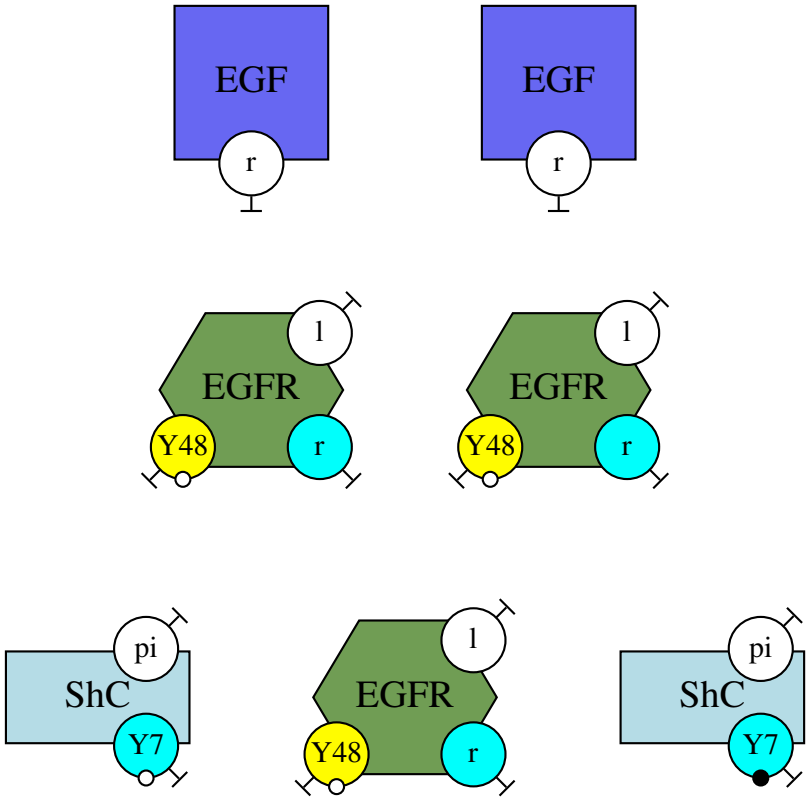
%agent: EGF(r)
%agent: EGFR(l r Y48{u p})
%agent: ShC(pi Y7{u p})
%agent: Grb2(a b)
%agent: Sos(d)

Extended contact map



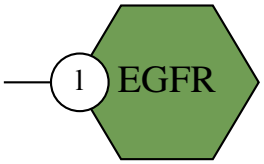

```
%agent: EGF(r[l.EGFR])
%agent: EGFR(l[r.EGF]
             r[r.EGFR]
             Y48{u p}[pi.ShC])
%agent: ShC(pi[Y48.EGFR]
            Y7{u p}[a.Grb2])
%agent: Grb2(a[Y7.ShC]
             b[d.Sos])
%agent: Sos(d[b.Grb2])
```

Initial state

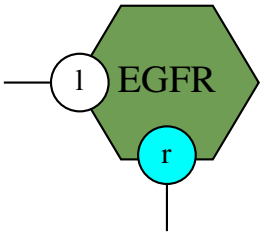



```
%init: 2 EGF()
%init: 3 EGFR()
%init: 1 ShC()
%init: 1 ShC(Y7{p})
```

Observation



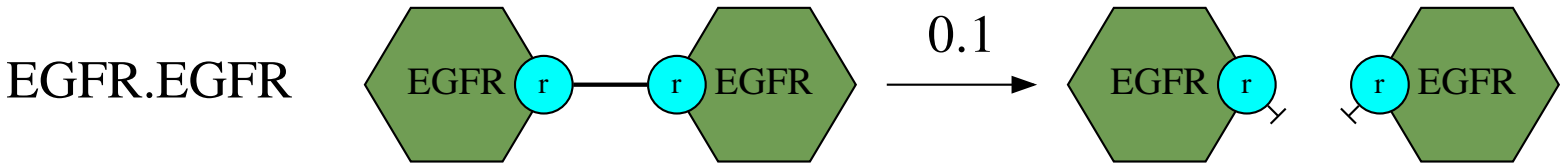
%obs: 'l' |EGFR(l[_])|



%obs: 'r' |EGFR(l[_] r[_])|

%obs: 'r' |EGFR(Y48{p})|

Labels and rates

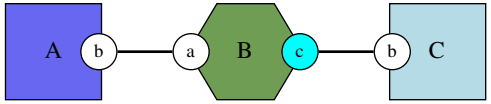


'EGFR..EGFR' EGFR(r[1]),EGFR(r[1])-> EGFR(r[.]),EGFR(r[.]) @ 0.1

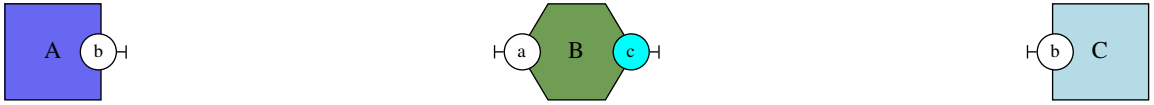
Practical activity

Consider the four following rules:

and study the abundance of the following pattern:



when the initial abundance of the following agents:



is changing.

Overview

1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

Exponential law: Definition

We denote as $P(e > T)$ the probability that a given event e does not occur before time T.
We assume $P(e > T + t \mid e > T) = P(e > t)$.

We have:

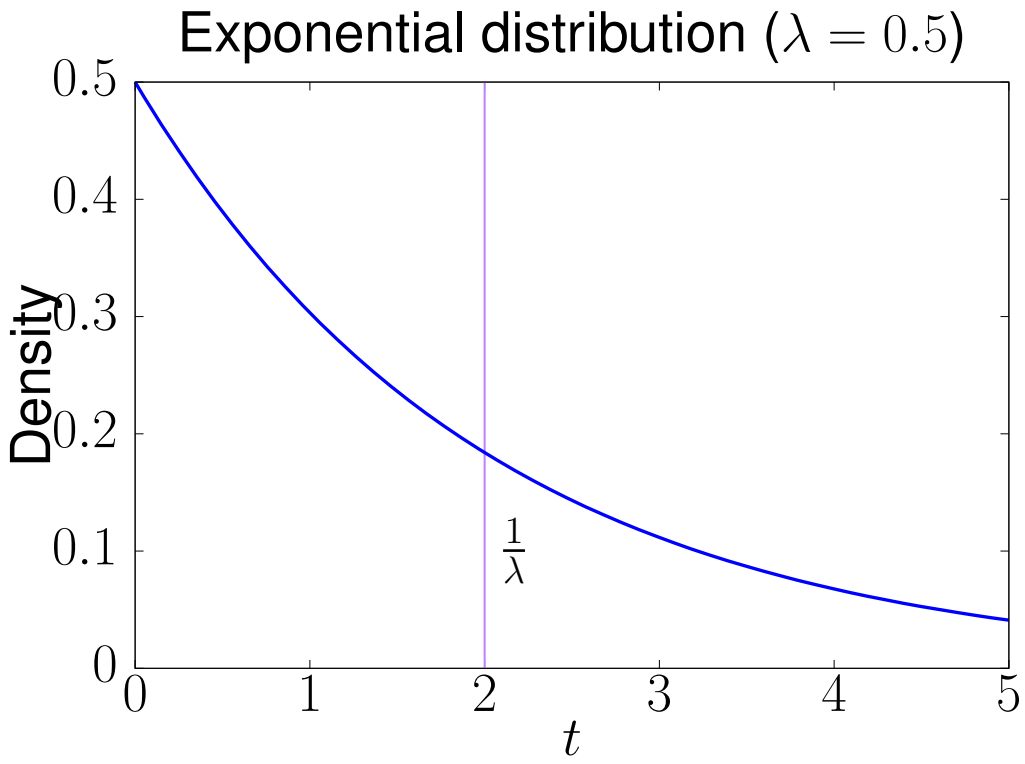
1. $P(e > t)$ is monotonic (decreasing);
2. $P(e > 0) = 1$;
3. $P(e > T + t) = P(e > T)P(e > T + t \mid e > T) = P(e > T)P(e > t)$.

So $P(e > t) = e^{-\lambda t}$ for a given $\lambda > 0$.

That is to say: $P(T + t > e > T) = e^{-\lambda T}(1 - e^{-\lambda t})$.

The time distribution of the event e is said to be defined by the exponential law with parameter λ.

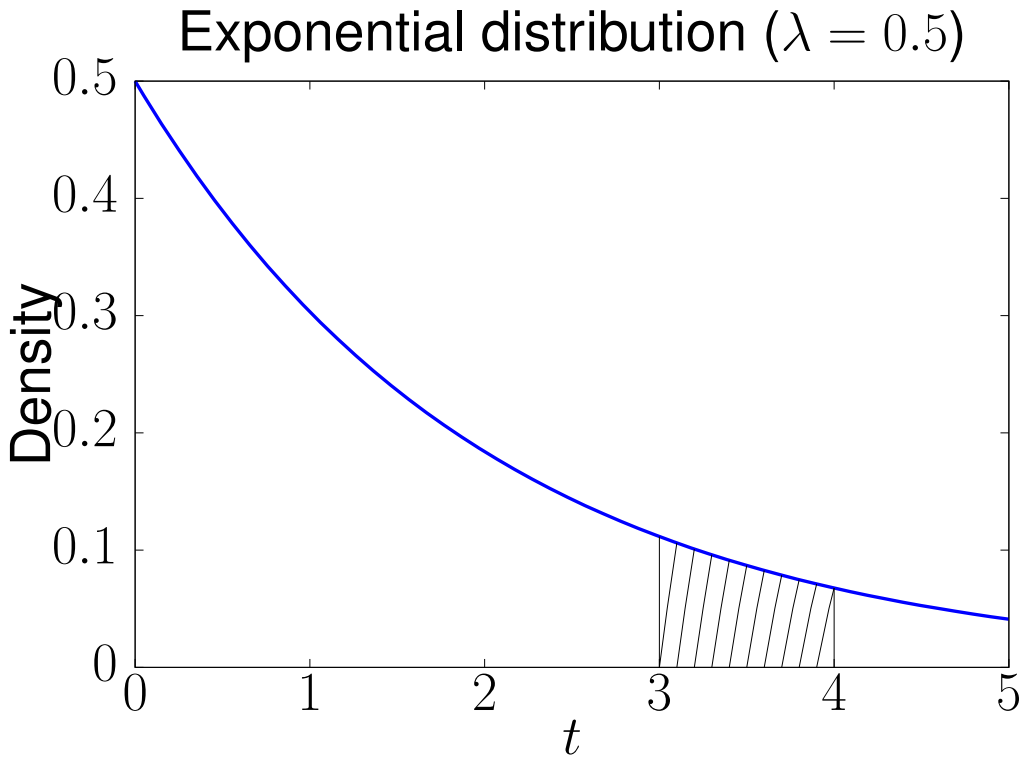
Exponential law: time distribution



$lim_0 \frac{P(t<e<t+dt)}{dt}$ ——

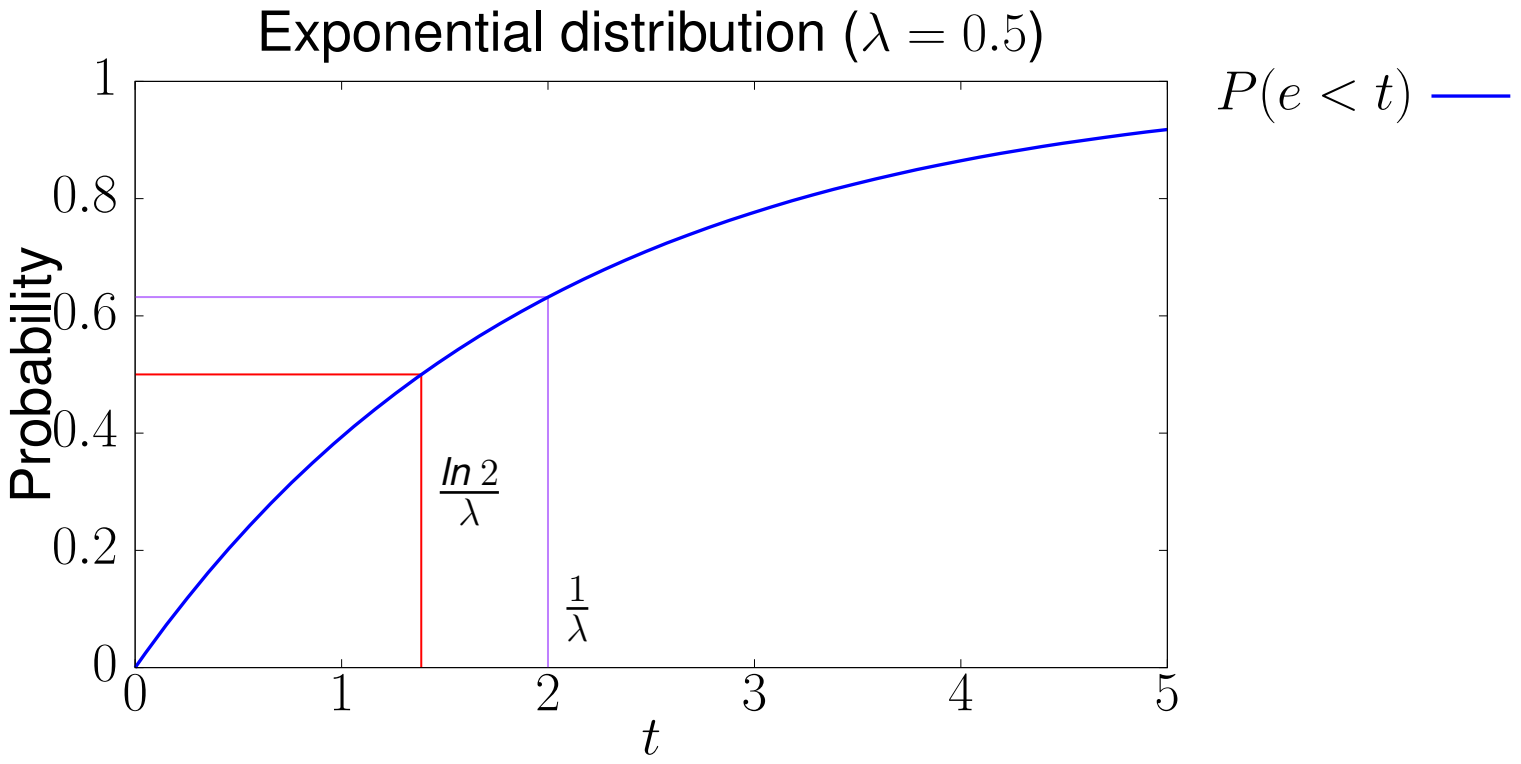
Average time: $\frac{1}{\lambda}$
Variance: $\frac{1}{\lambda^2}$.

Exponential law: probability to occur in a time interval



$lim_0 \frac{P(t<e<t+dt)}{dt}$ ——

Exponential law: cumulative distribution



Average time: $\frac{1}{\lambda}$ Median time: $\frac{\ln 2}{\lambda}$

One Markovian clock

We consider a random clock such that the time between two consecutive ticks is drawn according to an exponential law of parameter λ.

We denote as X_t^λ the random variables that counts the number of ticks.

X_t^λ is called a Markovian clock with time parameter λ.

Combining Markovian clocks

The sum of two Markovian clock of parameters λ and μ is a Markovian clock of parameters $\lambda + \mu$.

The probability that a clock ticks between time T and $T + t$ is equal to:

$$e^{-(\lambda+\mu)T}(1 - e^{-(\lambda+\mu)t})$$

The probability that the next tick is from the first clock is:

$$\frac{\lambda}{\lambda + \mu}.$$

Overview

1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

Set of events

We consider:

- $\mathcal{Q}$, a set of states.
- $\mathcal{E}$, a set of events
  each event $e \in \mathcal{E}$ is made of:
  - $\mathsf{ENABLED}_e$, a subset of the set $\mathcal{Q}$;
  - ACTION_e, a function from the set $\mathcal{Q}$ into the set $\mathcal{Q}$;
  - $\lambda(e)$, a time parameter.

Stochastic semantics

The stochastic semantics of a set of events $\mathcal{E}$ is defined as follows:

- Each event $e \in \mathcal{E}$ ticks according to a Markovian clock with parameter $\lambda(e)$.
- On a clock tick,
  - when the current state q belongs to the set $\mathsf{ENABLED}_e$, the state q is replaced with the state $\mathsf{ACTION}_e(q)$.
  - otherwise, the state q remains as it is.

Doob-Gillespie algorithm

Using linearity properties of exponential time distribution, the stochastic semantics of a set of events can be sampled by the following loop:

At state q:

1. Compute the set of enabled events: $\overline{\mathsf{ENABLED}}(q) = \{e \in \mathcal{E} \mid q \in \mathsf{ENABLED}_e\}$.
2. Compute the activity of the system: $\mathsf{ACT}(q) = \sum_{e \in \overline{\mathsf{ENABLED}}(q)} \lambda(e)$.
3. Draw the time period between the last and the next event: $\frac{-ln(U)}{\mathsf{ACT}(q)}$.
   (U is the uniform density distribution from 0 to 1)
4. Draw which event e is the next one
   (each event $e \in \overline{\mathsf{ENABLED}}(q)$ has probability $\frac{\lambda(e)}{\mathsf{ACT}(q)}$ to be the next one).
5. Replace the current state q with the state $\mathsf{ACTION}_e(q)$.
6. Repeat while $\overline{\mathsf{ENABLED}}(q) \neq \emptyset$.

Overview

1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

Potential optimizations

- Dynamic update of the set of enabled events.
- Over-approximating the set of enabled events.
- Rectangular approximation.
- Rigidity.
- Use sharing.

Wake up/Inhibition map

Positive (wake-up) and negative influences between rules can be computed statically.

1. $\textsc{wake-up}^\sharp \supseteq \left\{ (e, e') \in \mathcal{E} \;\middle|\; \begin{array}{l} \exists q_0 \in \mathcal{Q}_0, q \in \mathcal{Q} q_0 \rightarrow^* q, \\ q \in \textsc{enabled}_e \setminus \textsc{enabled}_{e'}, \\ \textsc{action}_e(q) \in \textsc{enabled}_{e'} \end{array} \right\}.$
2. $\textsc{inhibition}^\sharp \supseteq \left\{ (e, e') \in \mathcal{E} \;\middle|\; \begin{array}{l} \exists q_0 \in \mathcal{Q}_0, q \in \mathcal{Q}, q_0 \rightarrow^* q, \\ q \in \textsc{enabled}_e \cap \textsc{enabled}_{e'}, \\ \textsc{action}_e(q) \notin \textsc{enabled}_{e'} \end{array} \right\}.$

   where $\mathcal{Q}_0$ is the set of potential initial states.

Three levels of accuracy

- low resolution:

  Manhattan distance:

  The engine only checks that the event e change one bit of information that is required for the event e'.

- medium resolution:

  Takes into account local context:

  Wake-up: Additionally checks that the rhs of a refinement of the rule related to the event e and the lhs of a refinement of the rule related to the event e' may share a common connected component.

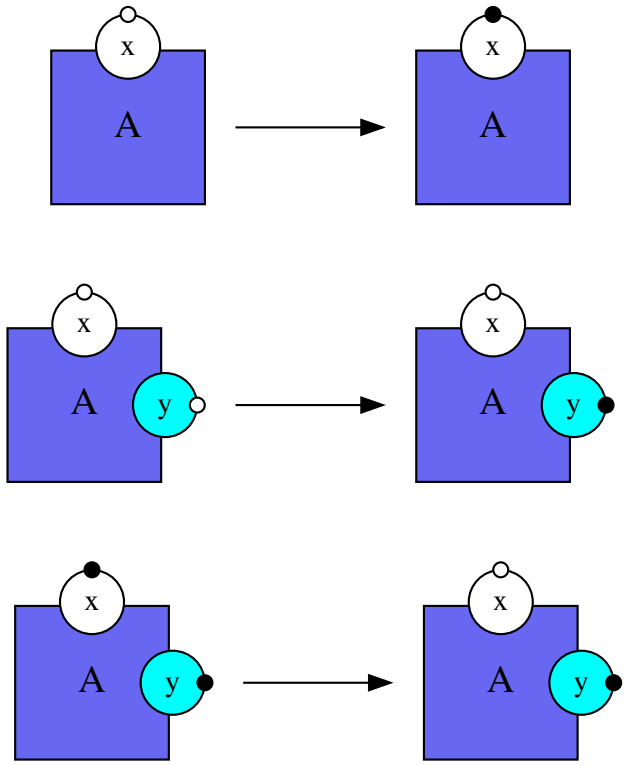
  Inhibition: Additionally checks that the lhs of a refinement of the rule related to the event e and the lhs of a refinement of the rule related to the event e' may share a common connected component.

- high resolution:

  The engine additionally checks that this pattern is reachable.

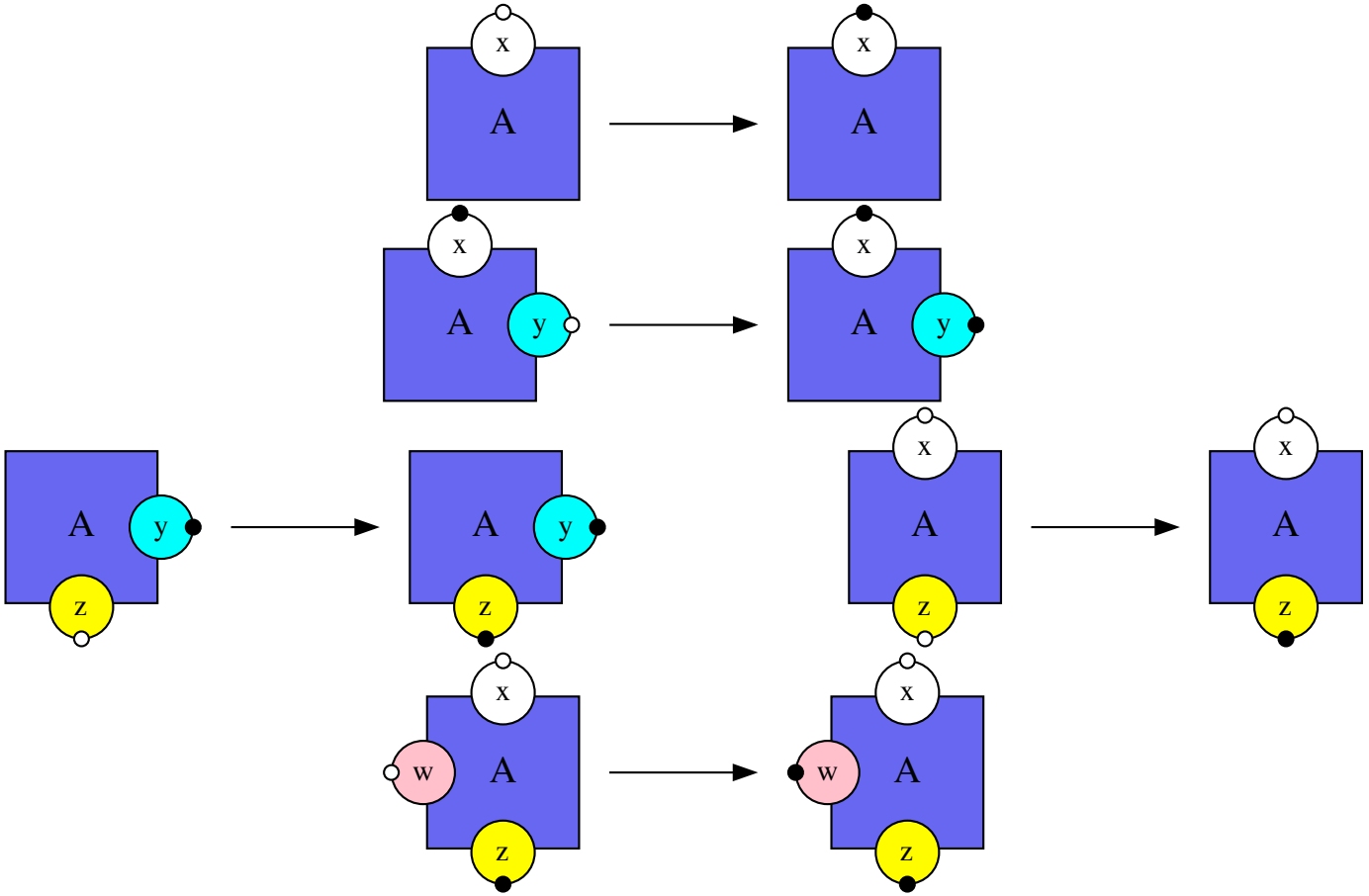
Practical activity

Compute the influence map, at each level of resolution, for the following set of rules:



Practical activity

Compute the influence map, at each level of resolution, for the following set of rules:



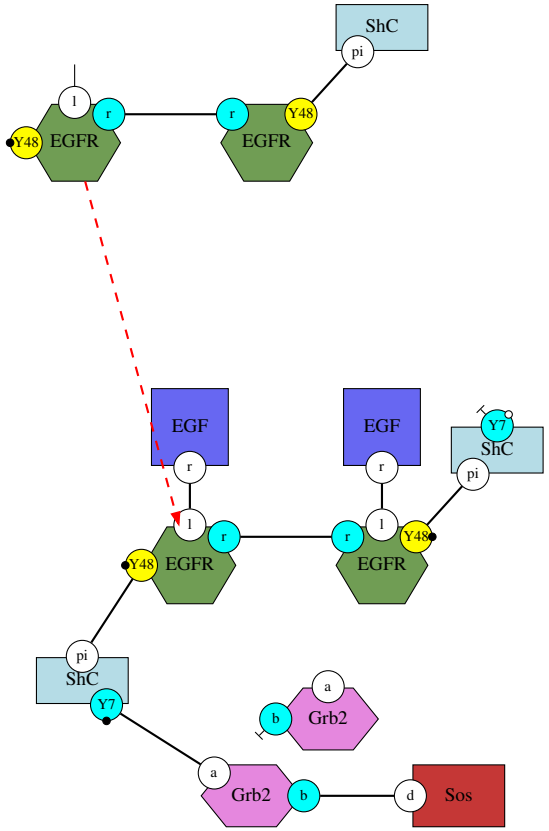
Over-approximating the set of enabled events

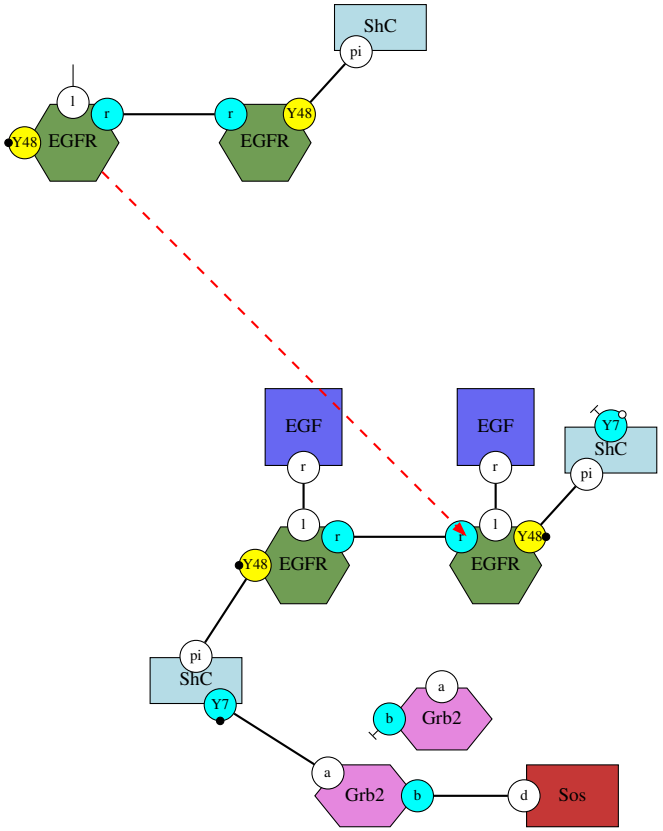
Using linearity properties of exponential time distribution, the stochastic semantics of a set of events can be sampled by the following loop:
At state q:

1. Compute a super-set $\overline{\mathsf{ENABLED}^\sharp}(q) \supseteq \{e \in \mathcal{E} \mid q \in \mathsf{ENABLED}_e\}$ of the set of enabled events.
2. Over-approximate the activity of the system: $\mathsf{ACT}^\sharp(q) = \sum_{e \in \overline{\mathsf{ENABLED}^\sharp}(q)} \lambda(e)$.
3. Increment time by $\frac{-ln(U)}{\mathsf{ACT}^\sharp(q)}$.
4. Draw which event is the next one
   (each event $e \in \overline{\mathsf{ENABLED}^\sharp}(q)$ has probability $\frac{\lambda(e)}{\mathsf{ACT}^\sharp(q)}$).
5. whenever $e \in \overline{\mathsf{ENABLED}}(q)$, replace the state with $\mathsf{ACTION}_e(q)$;
   otherwise ignore e.
6. Repeat while $\overline{\mathsf{ENABLED}^\sharp}(q) \neq \emptyset$.

Rigidity

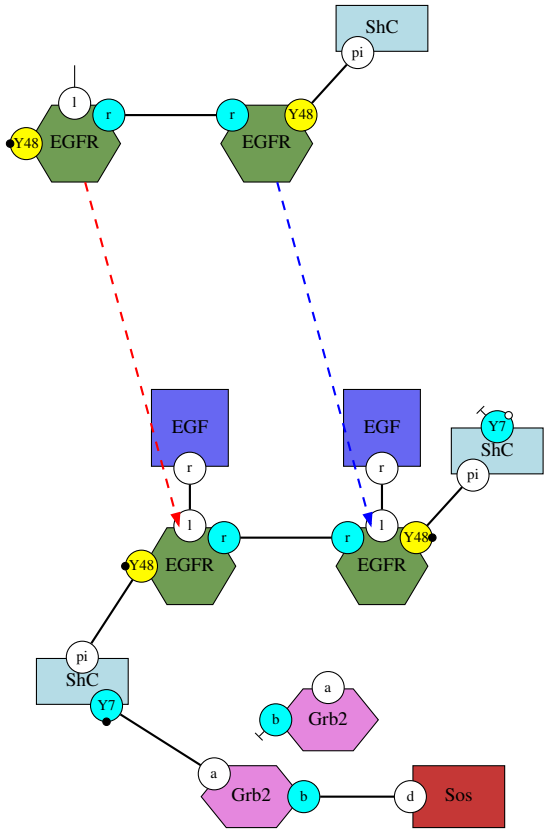
An embedding between a connected pattern and a graph is fully defined by the image of an agent.

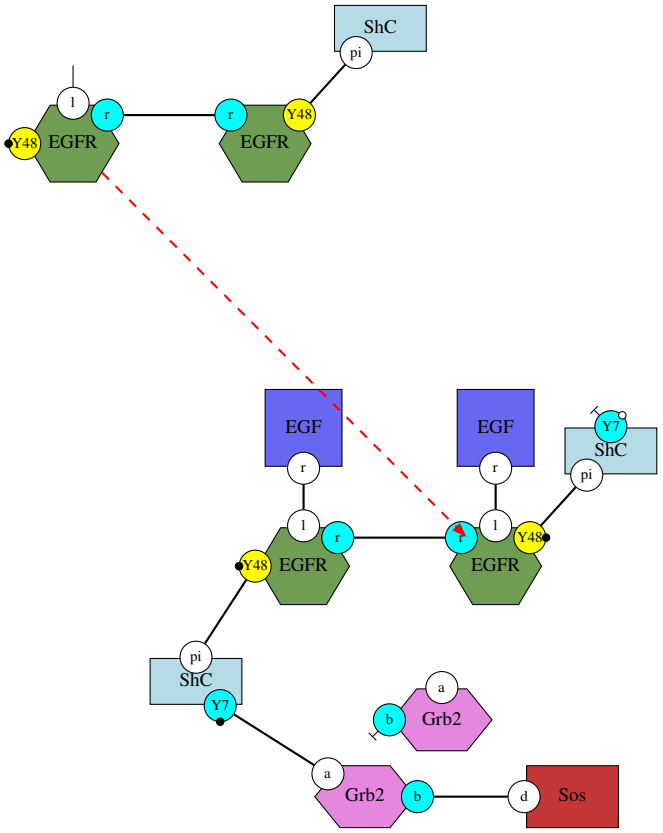




Rigidity

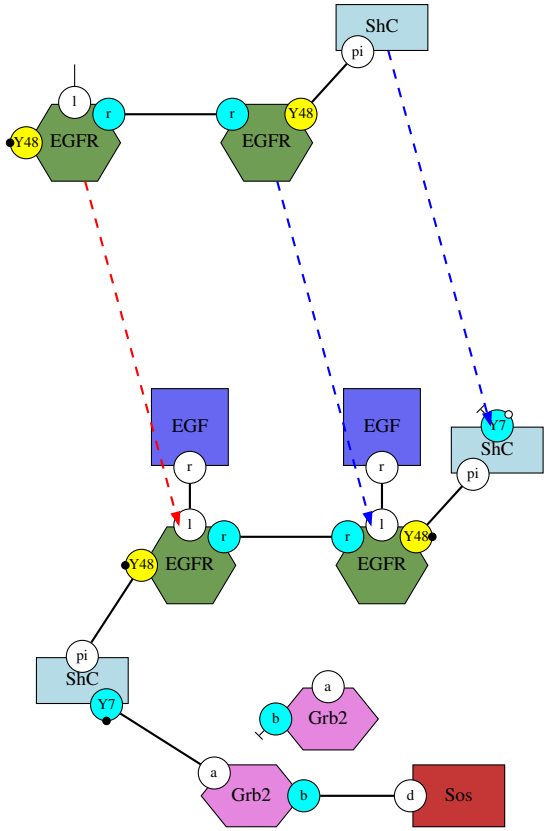
An embedding between a connected pattern and a graph is fully defined by the image of an agent.

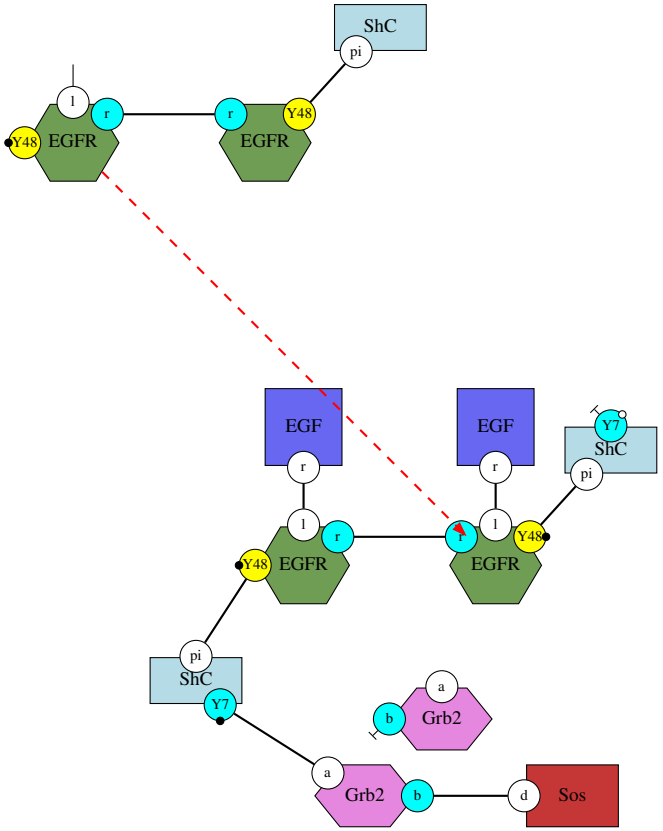




Rigidity

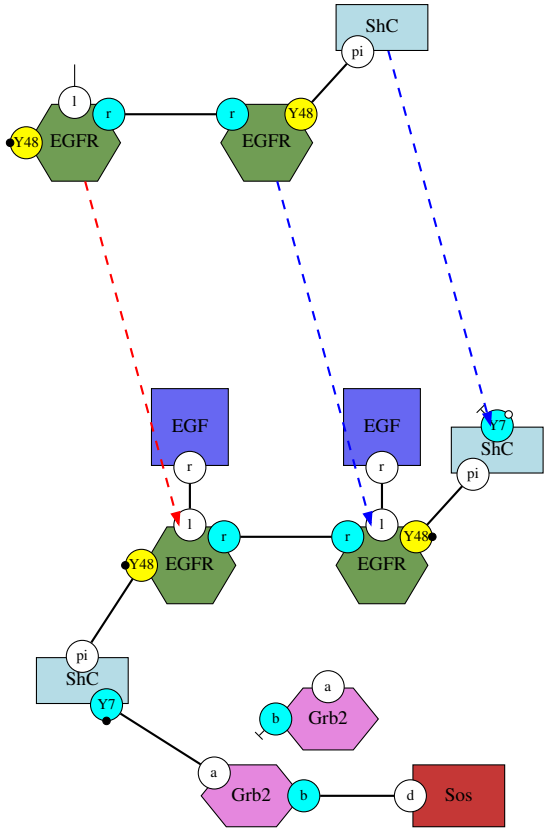
An embedding between a connected pattern and a graph is fully defined by the image of an agent.

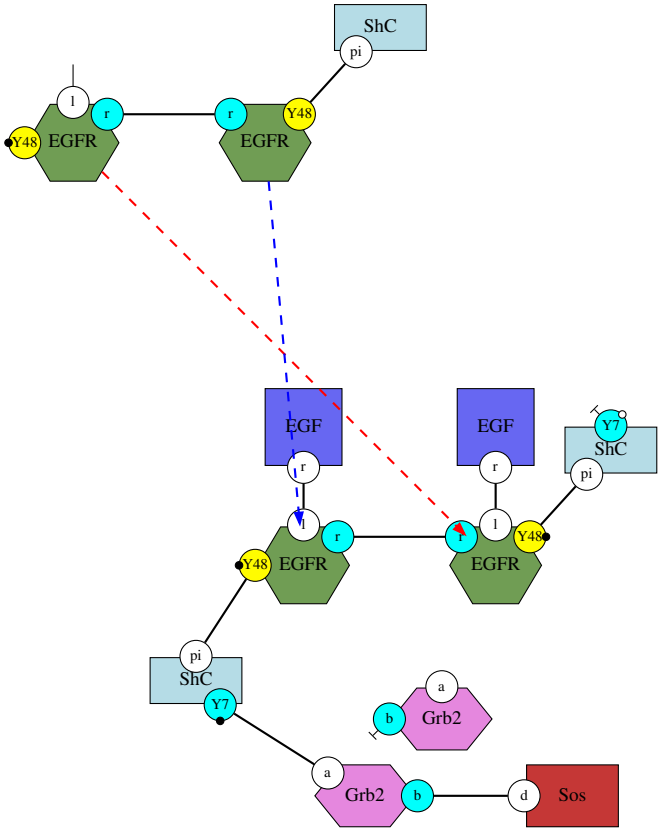




Rigidity

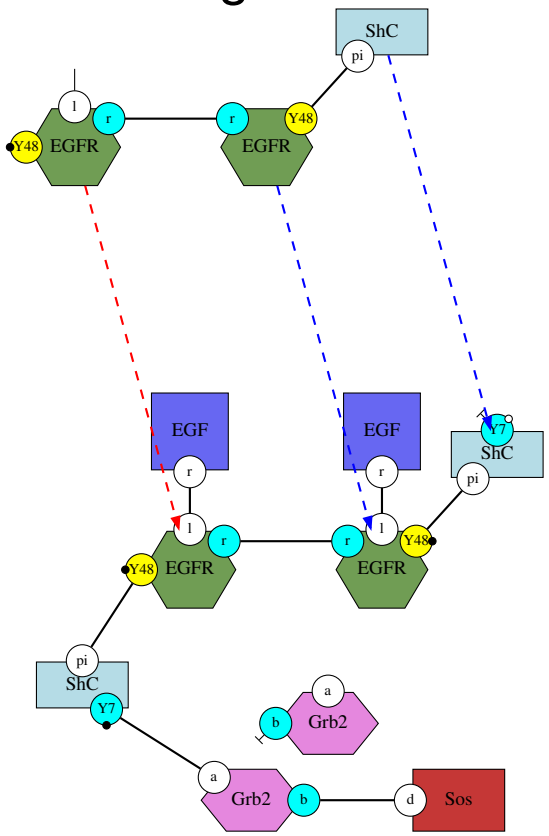
An embedding between a connected pattern and a graph is fully defined by the image of an agent.

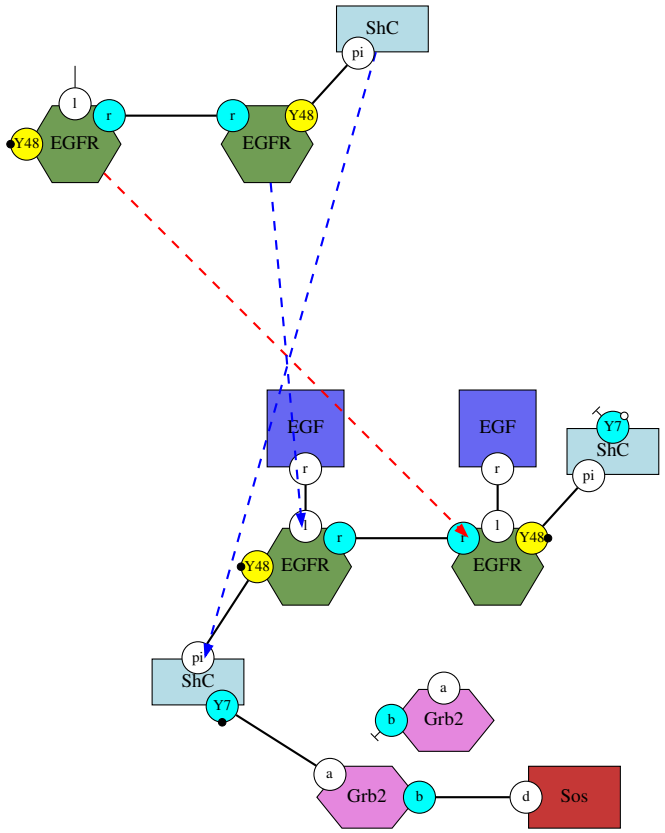




Rigidity

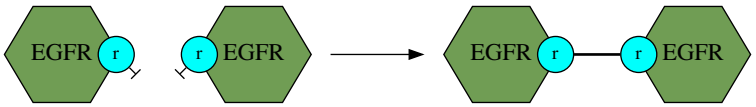
An embedding between a connected pattern and a graph is fully defined by the image of an agent.



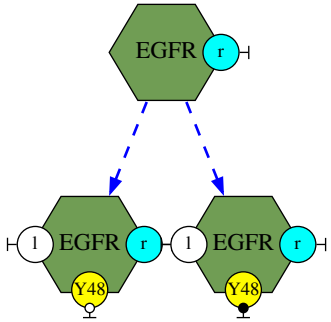


‘

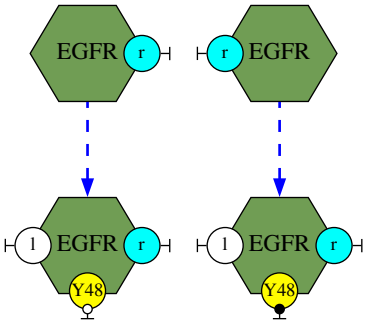
Rectangular approximation

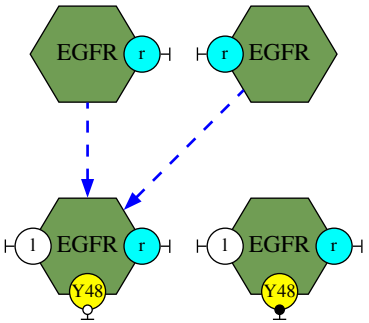


We collect potential embeddings for each connected component of the lhs.

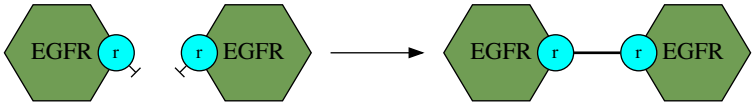


An event is restored by recombining the embeddings (one per cc).
In case of collision, it is dealt with as a null event.

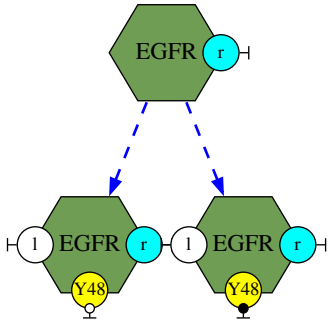




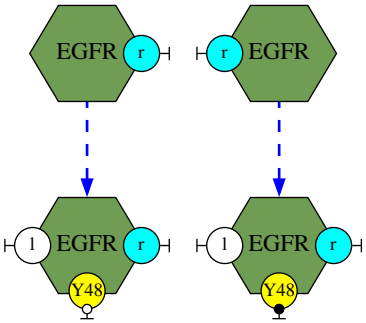
Rectangular approximation

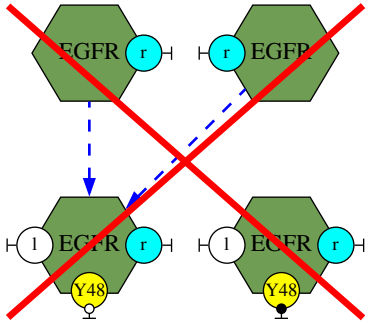


We collect potential embeddings for each connected component of the lhs.

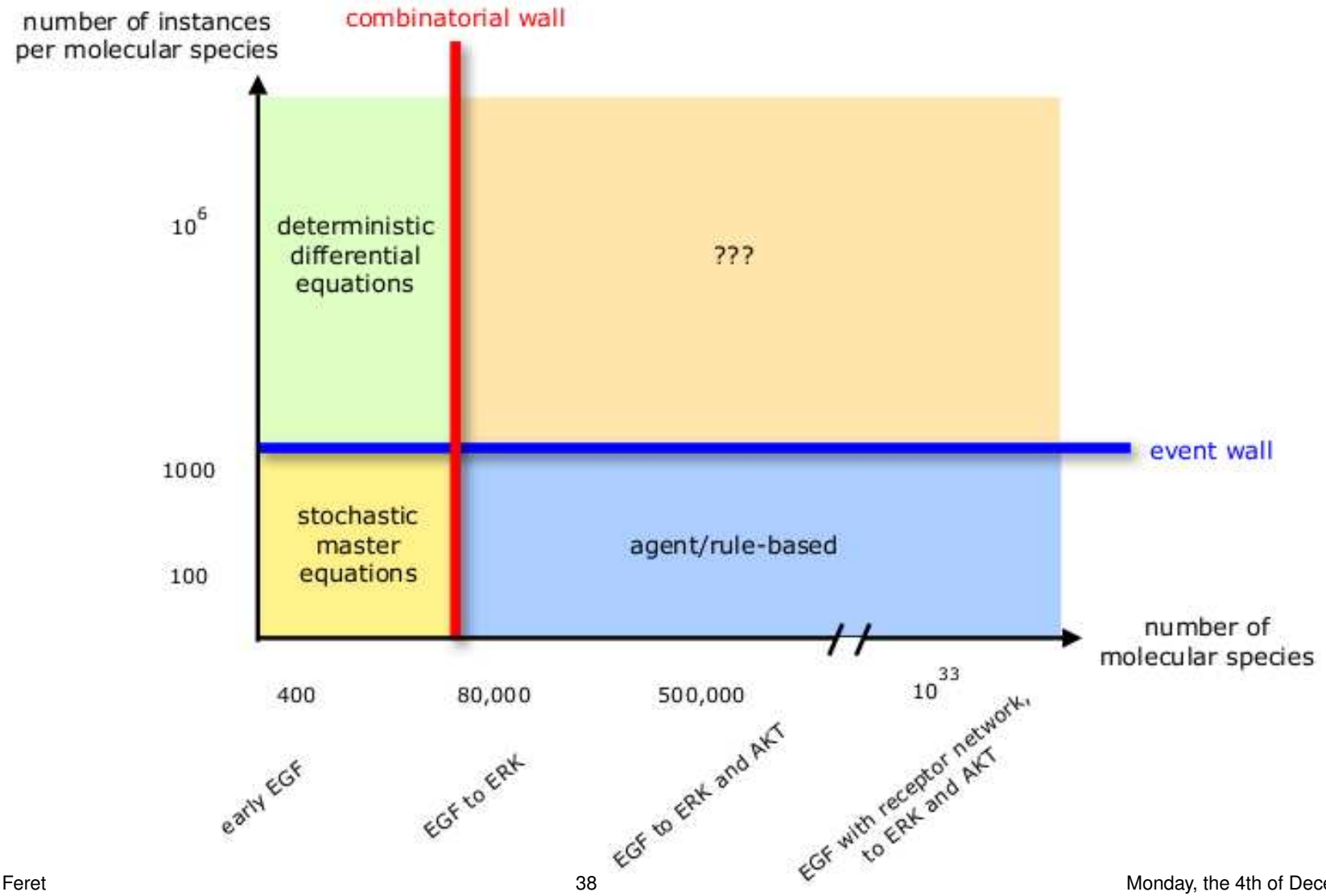


An event is restored by recombining the embeddings (one per cc).
In case of collision, it is dealt with as a null event.





Complexity walls

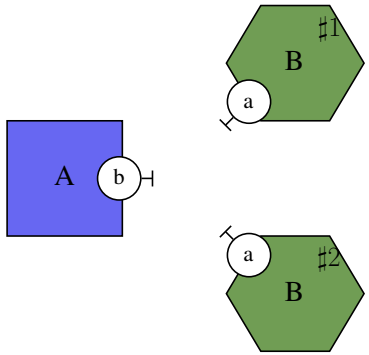


Overview

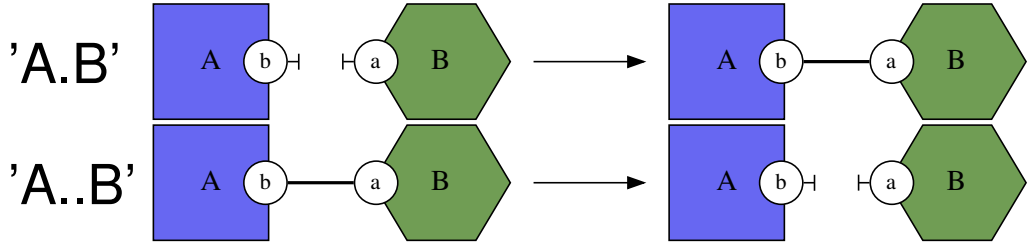
1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

Counter-factual execution

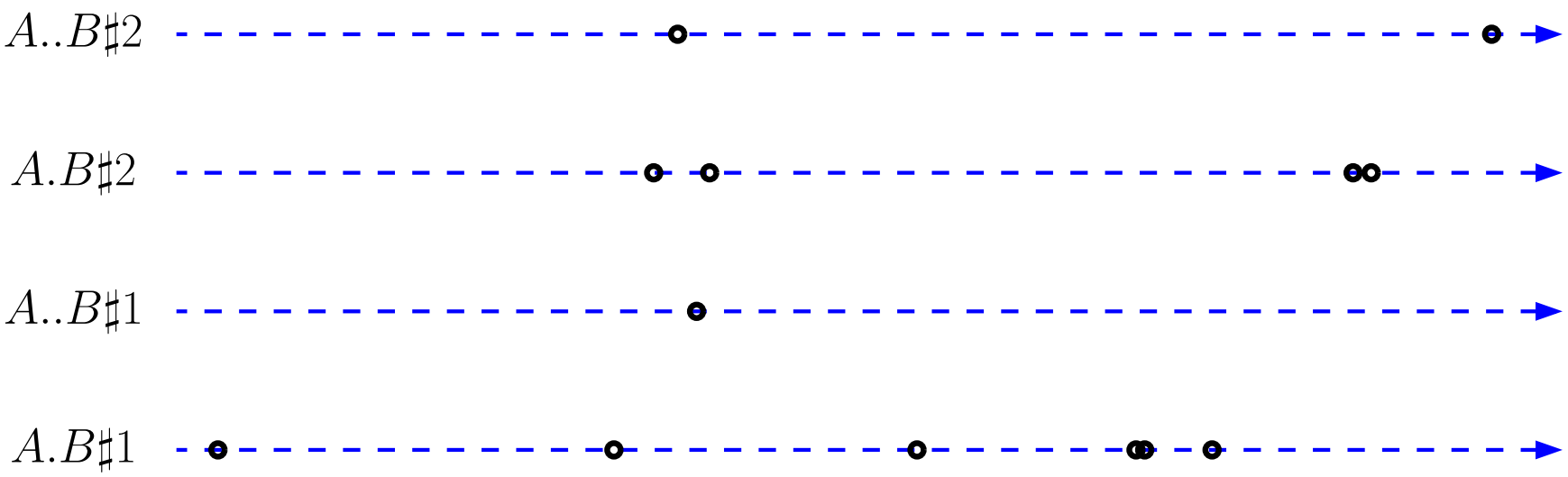
We consider the following initial state:

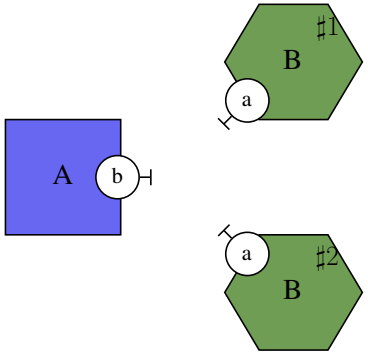


and the following rules:

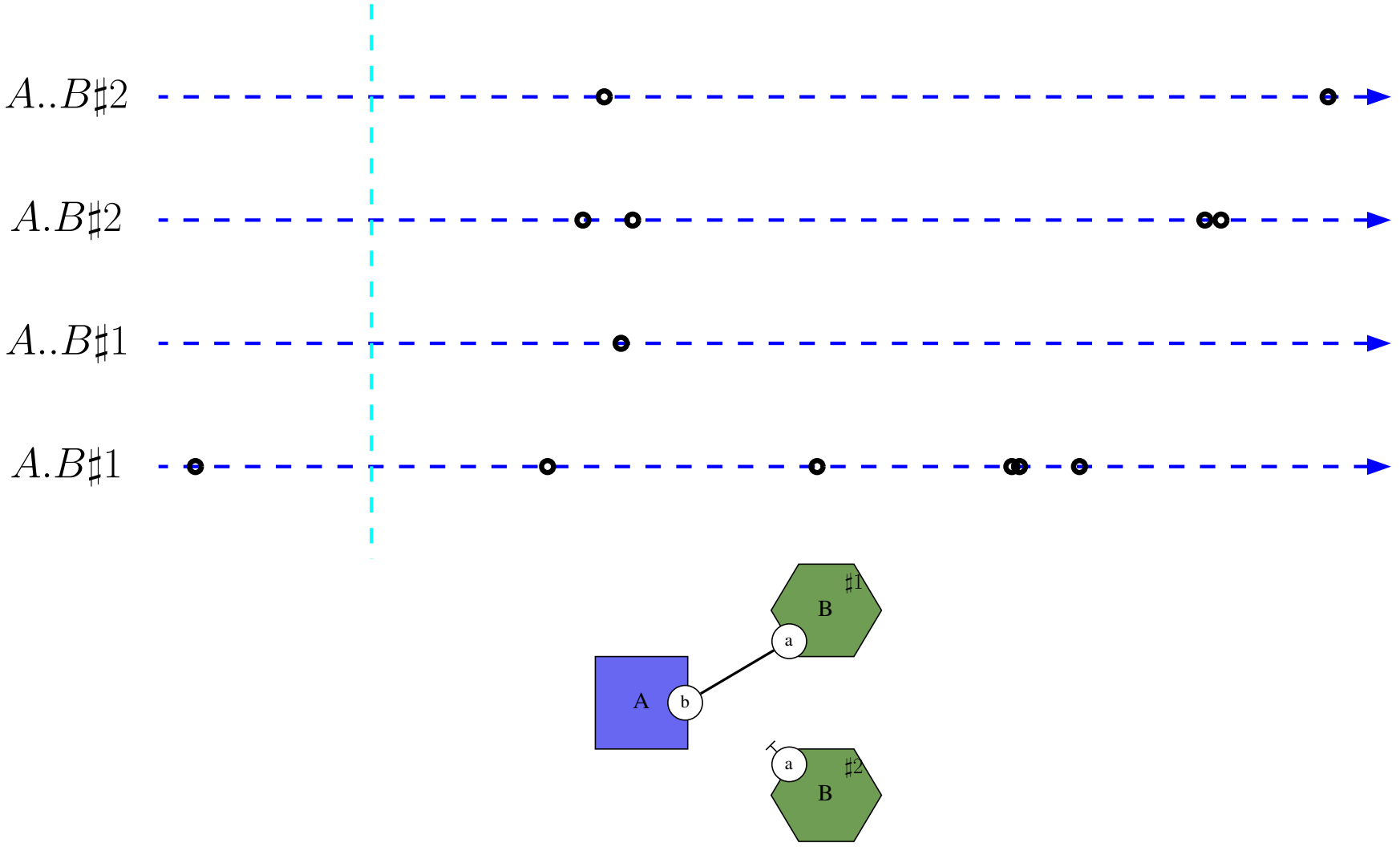


One execution

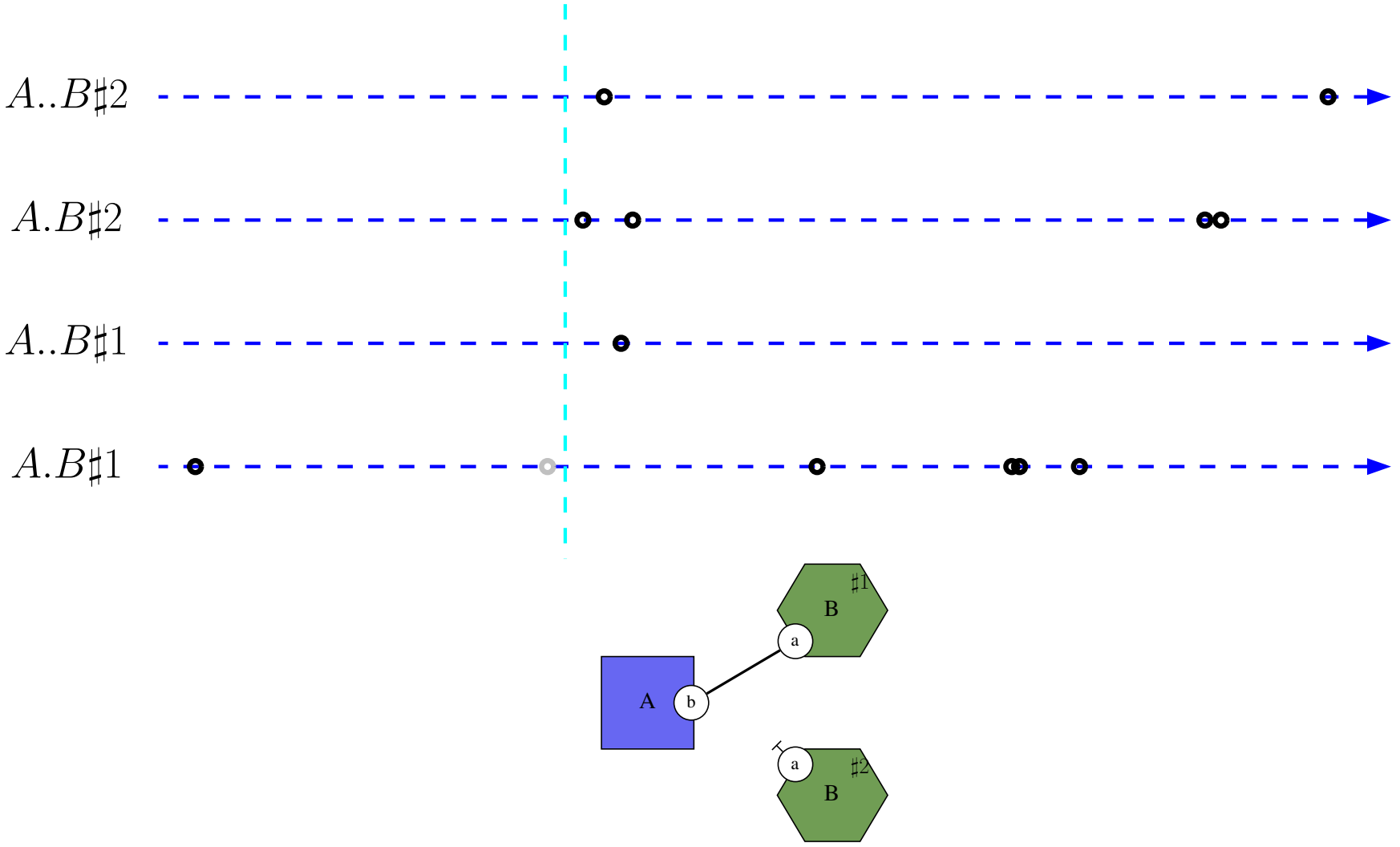




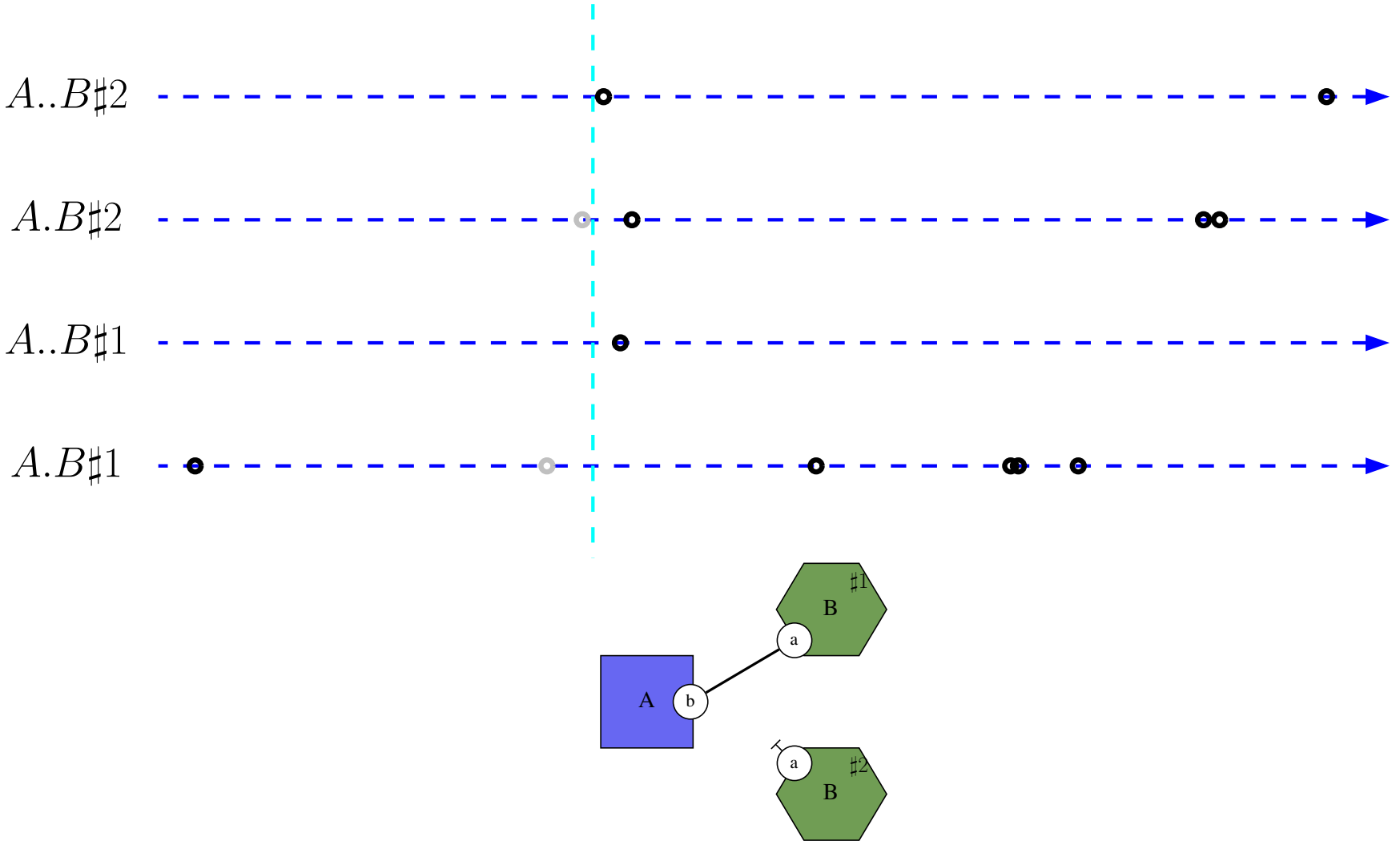
One execution



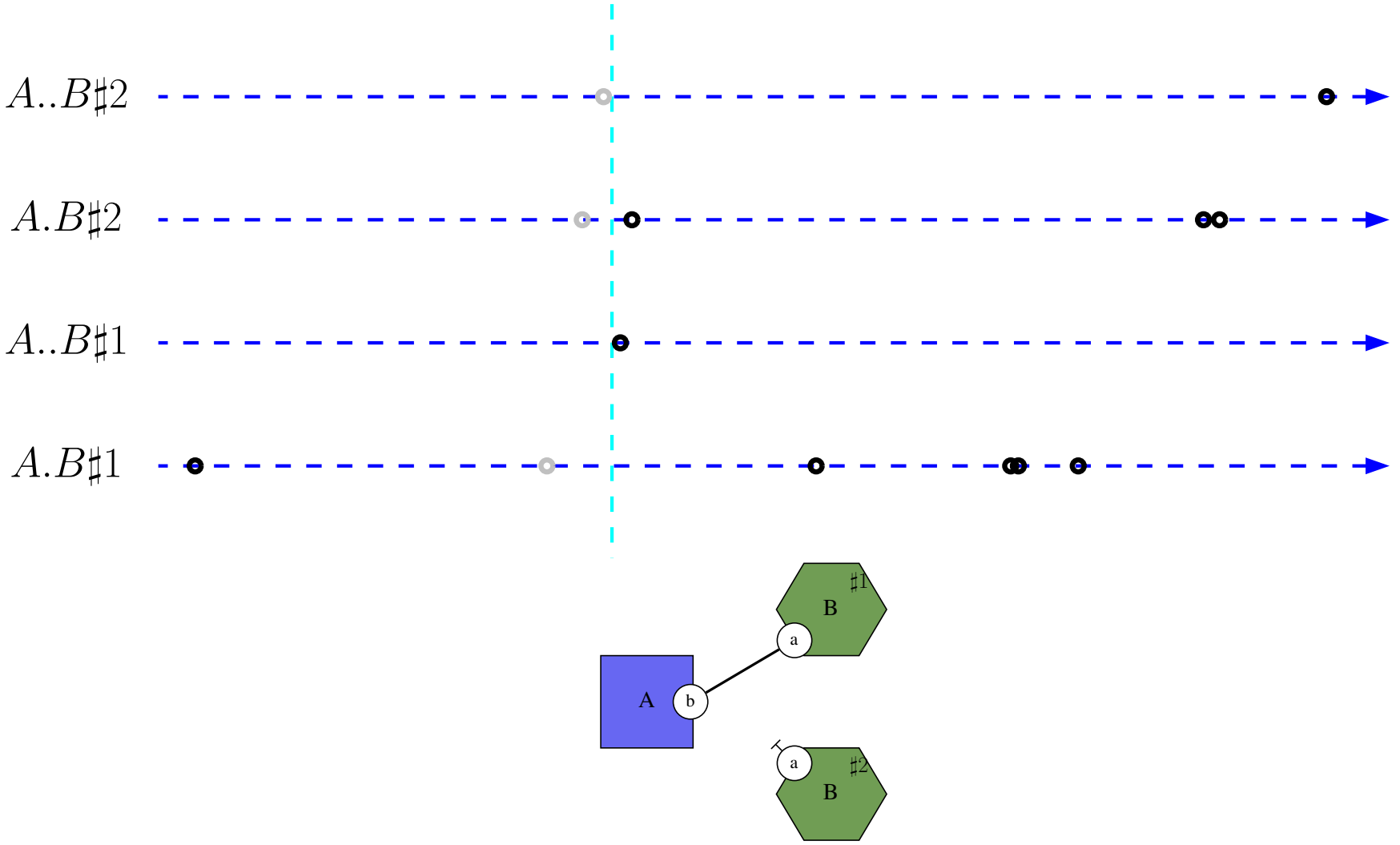
One execution



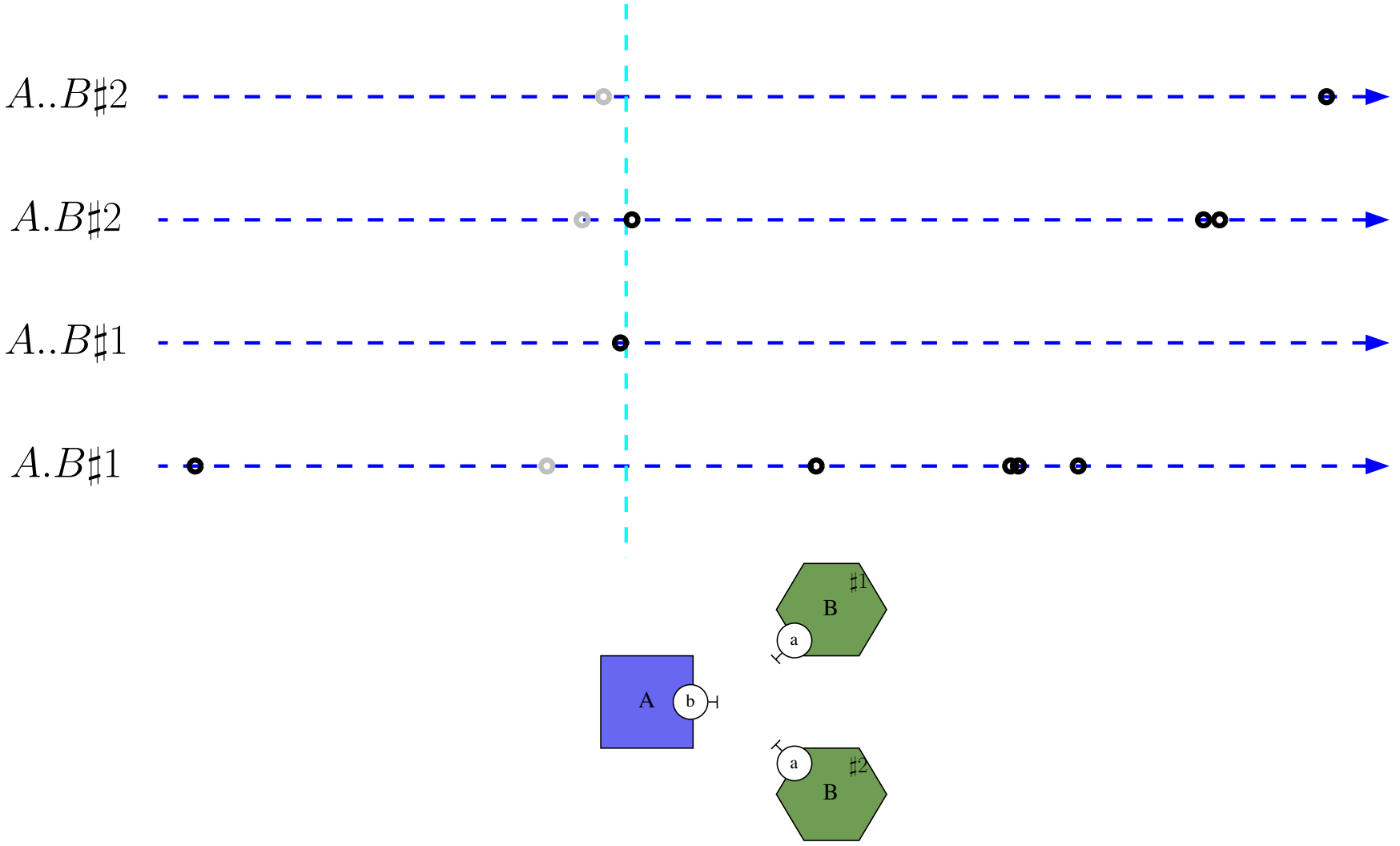
One execution



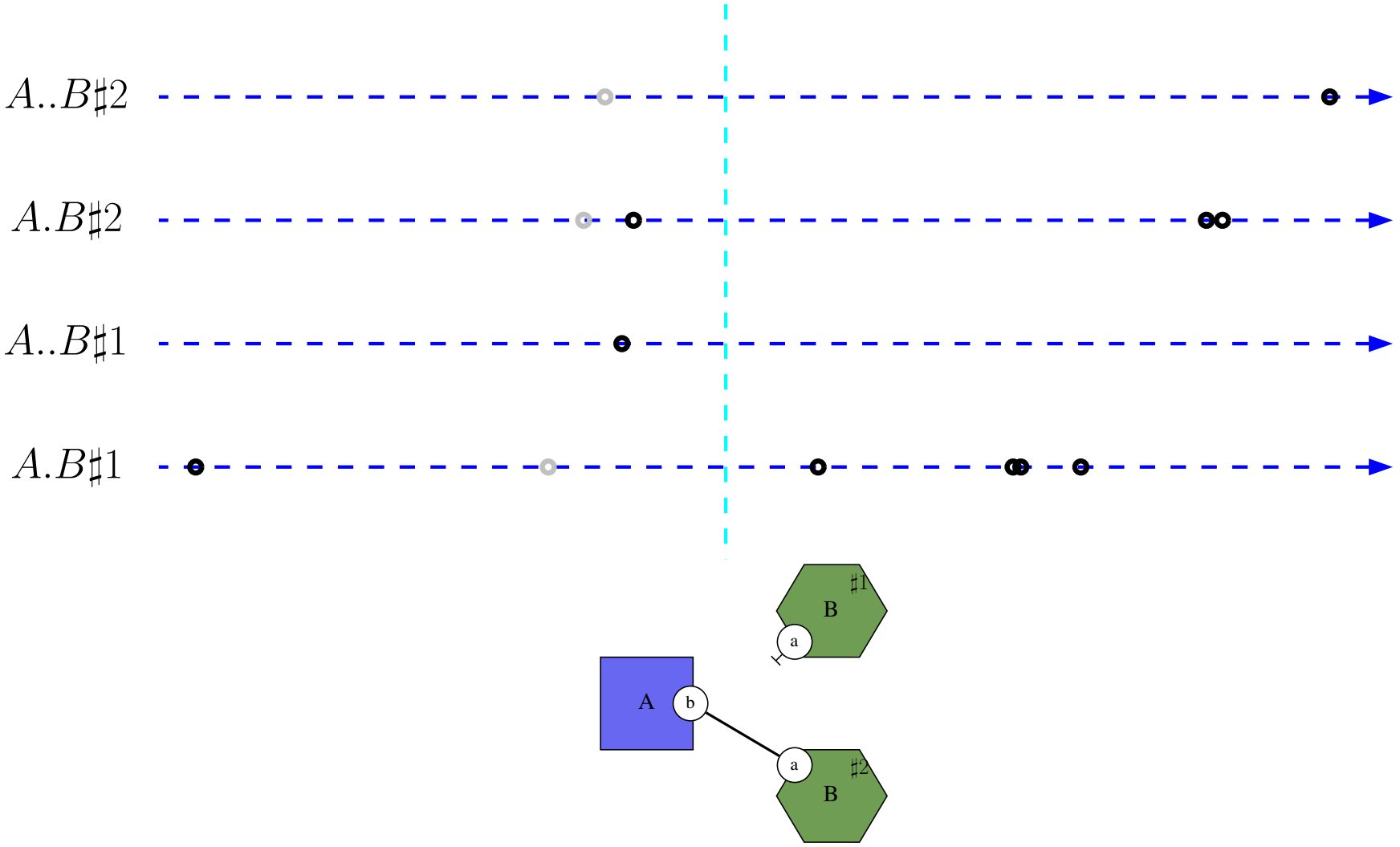
One execution



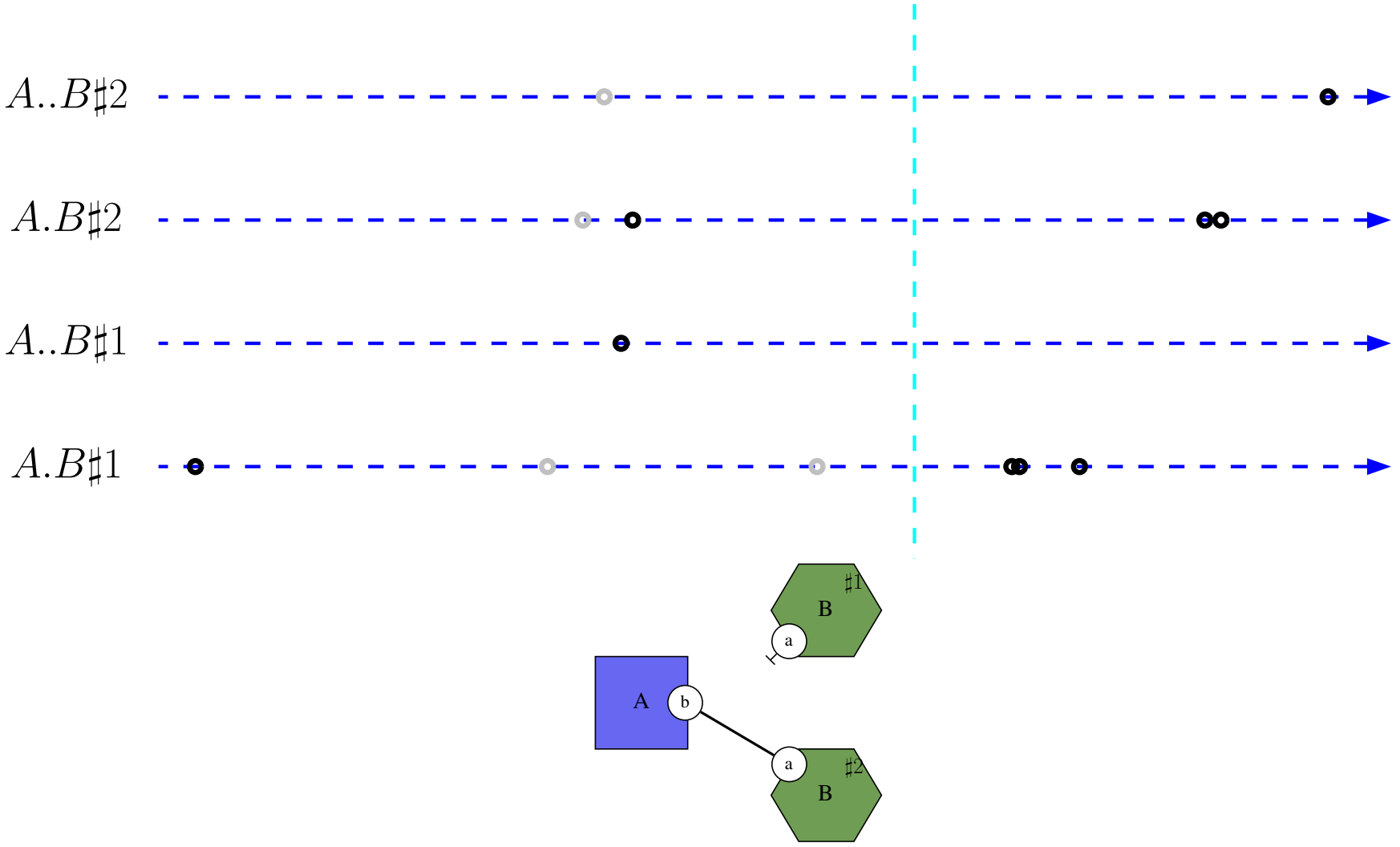
One execution



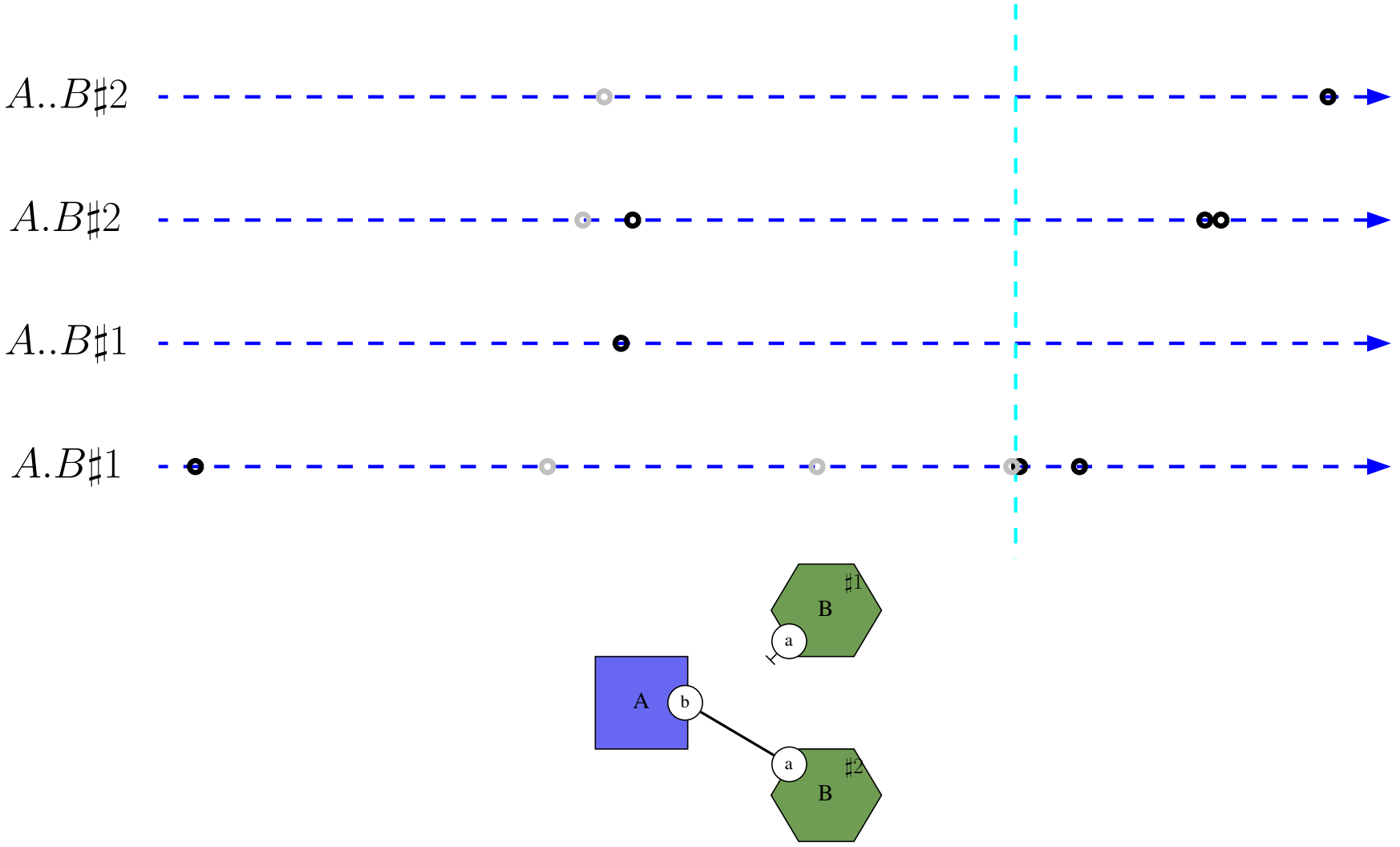
One execution



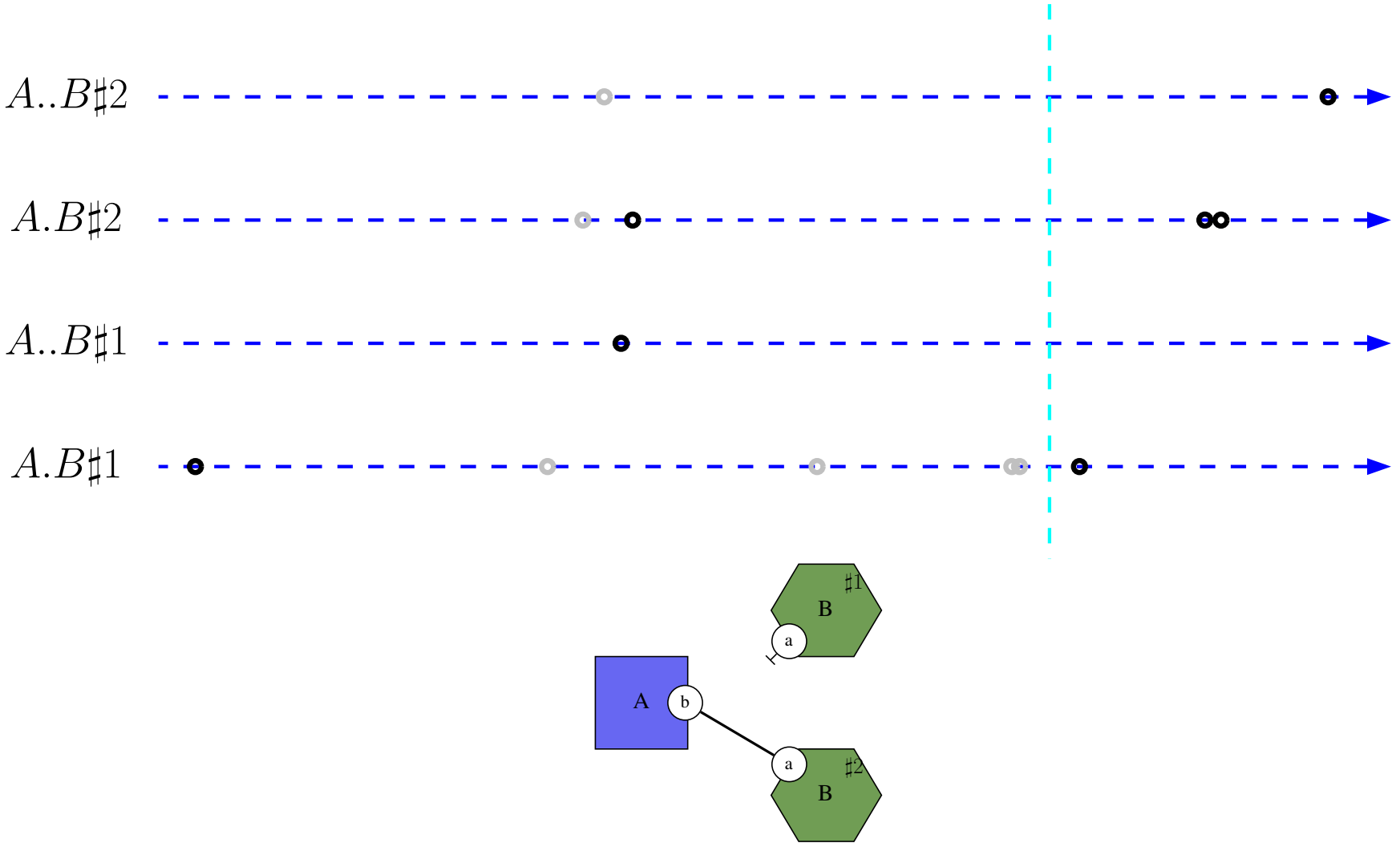
One execution



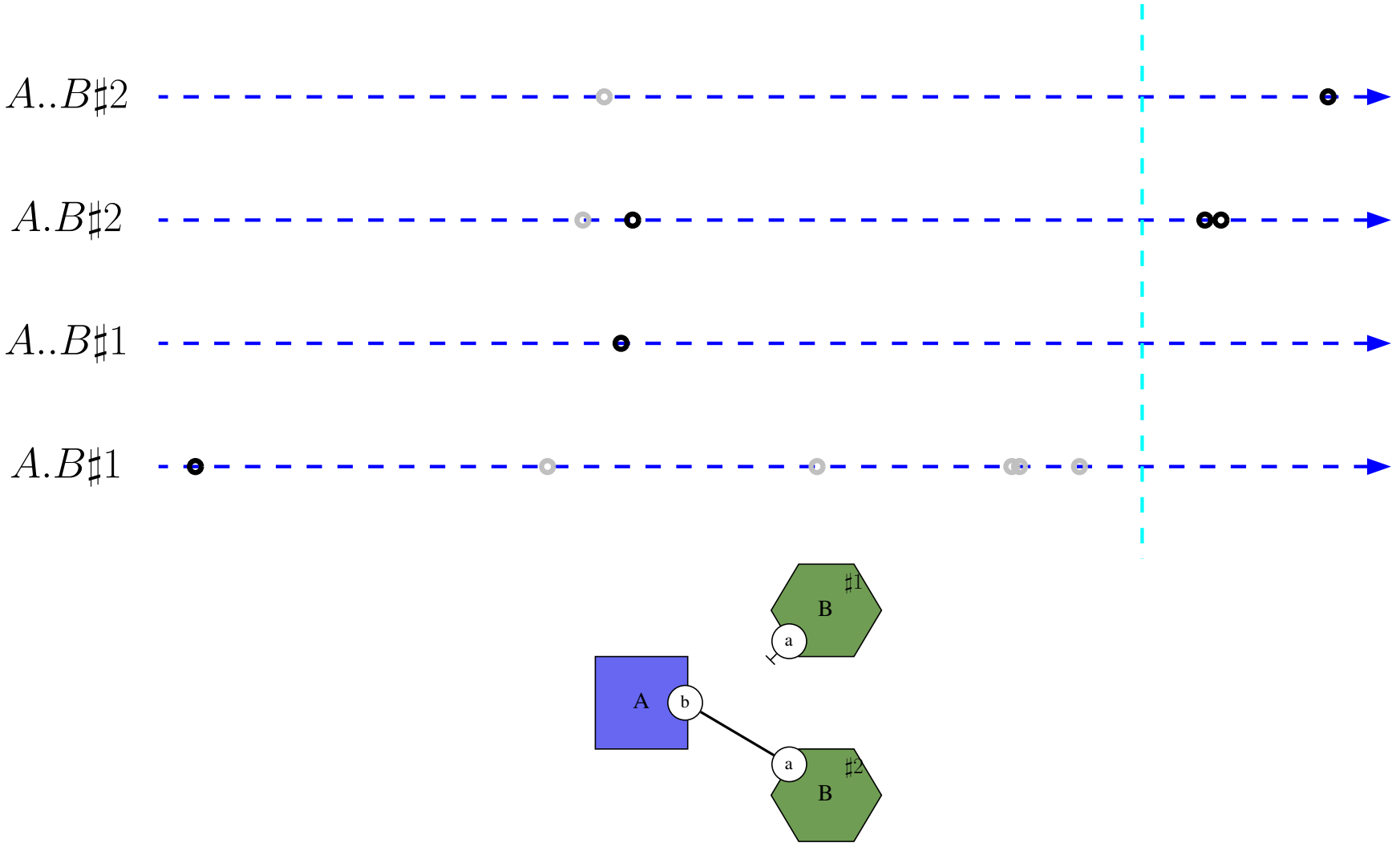
One execution



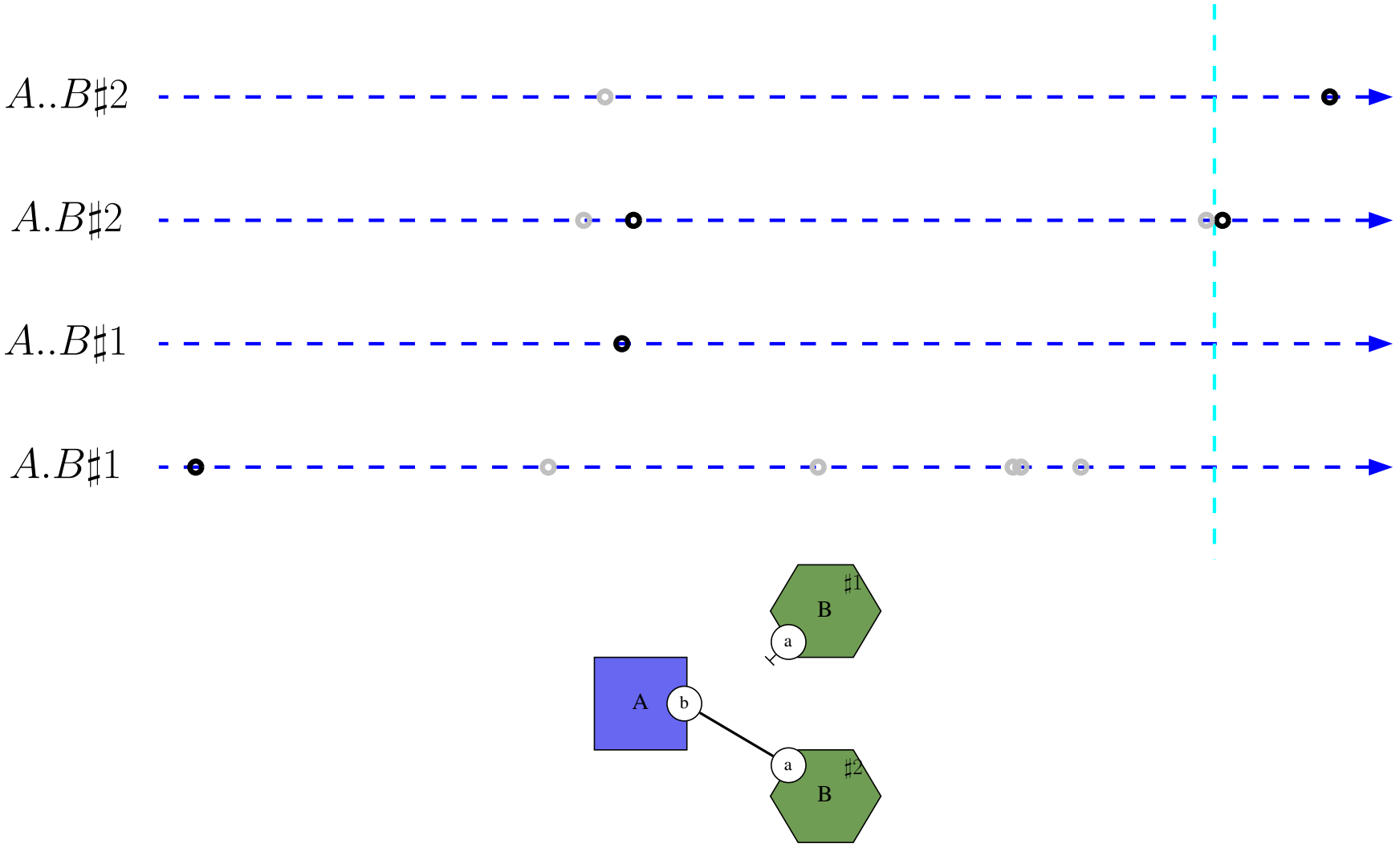
One execution



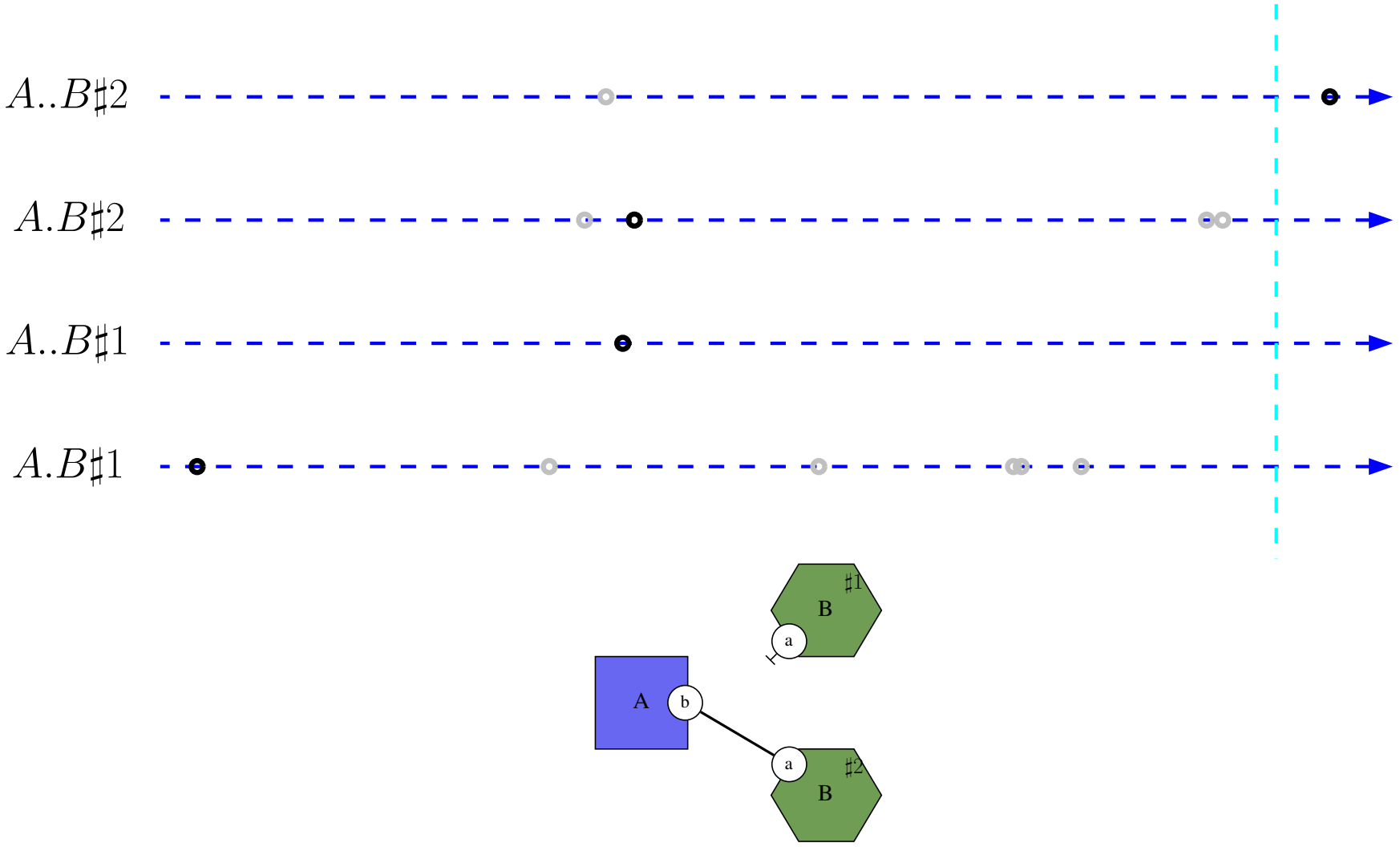
One execution



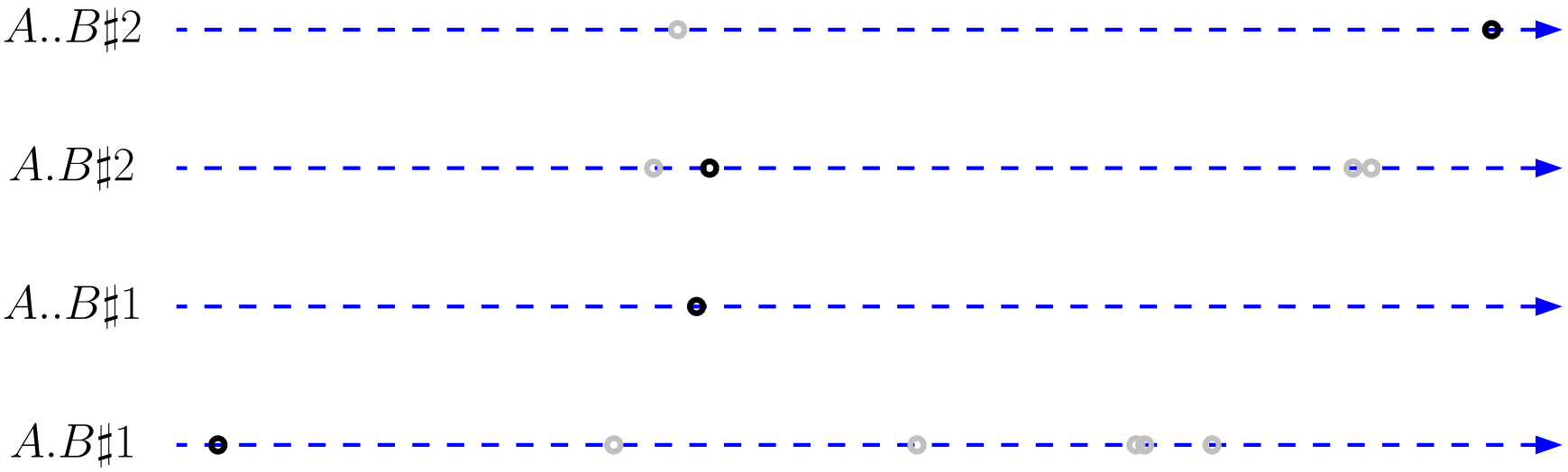
One execution

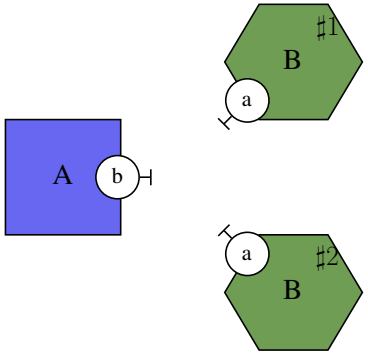


One execution

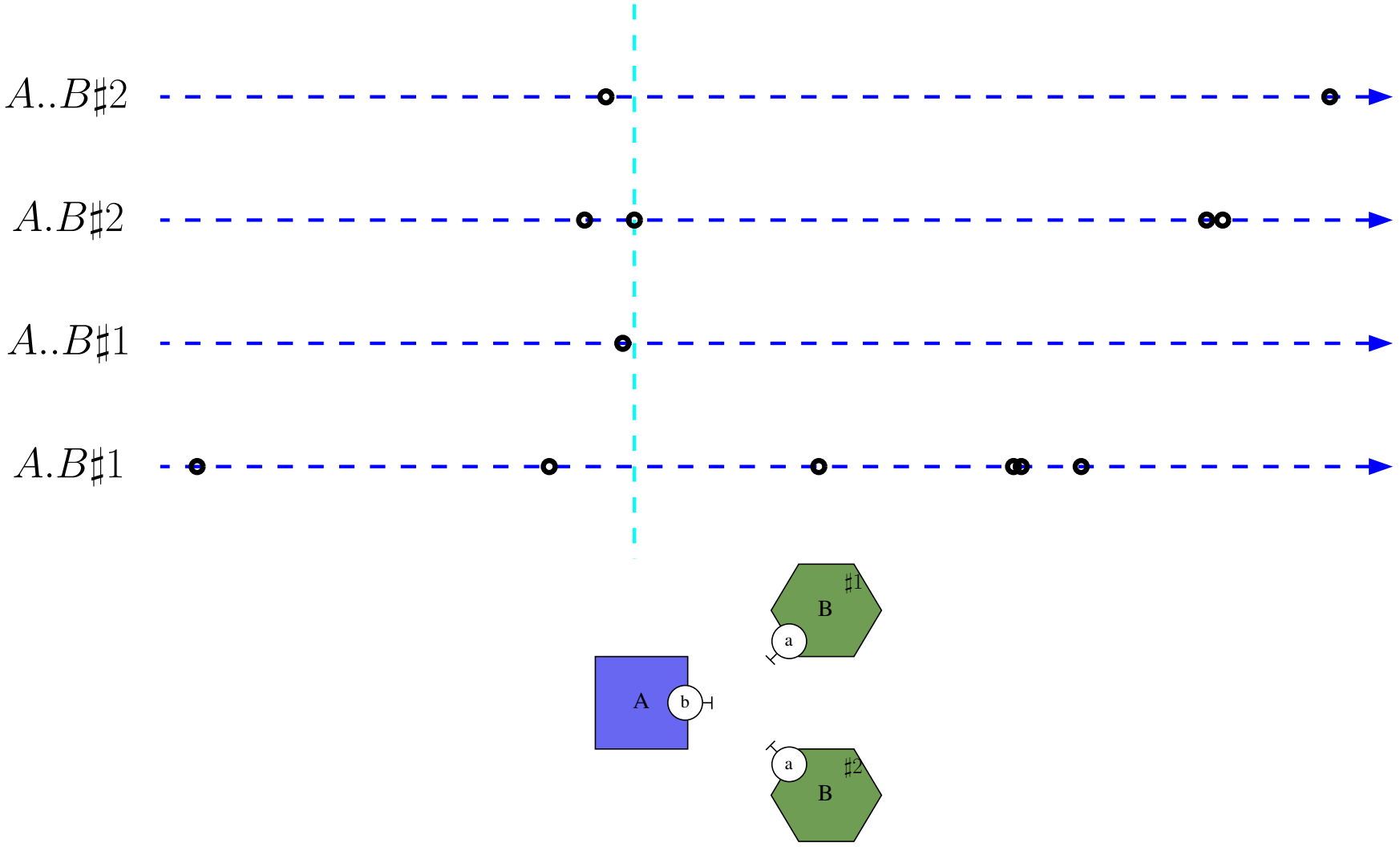


One execution

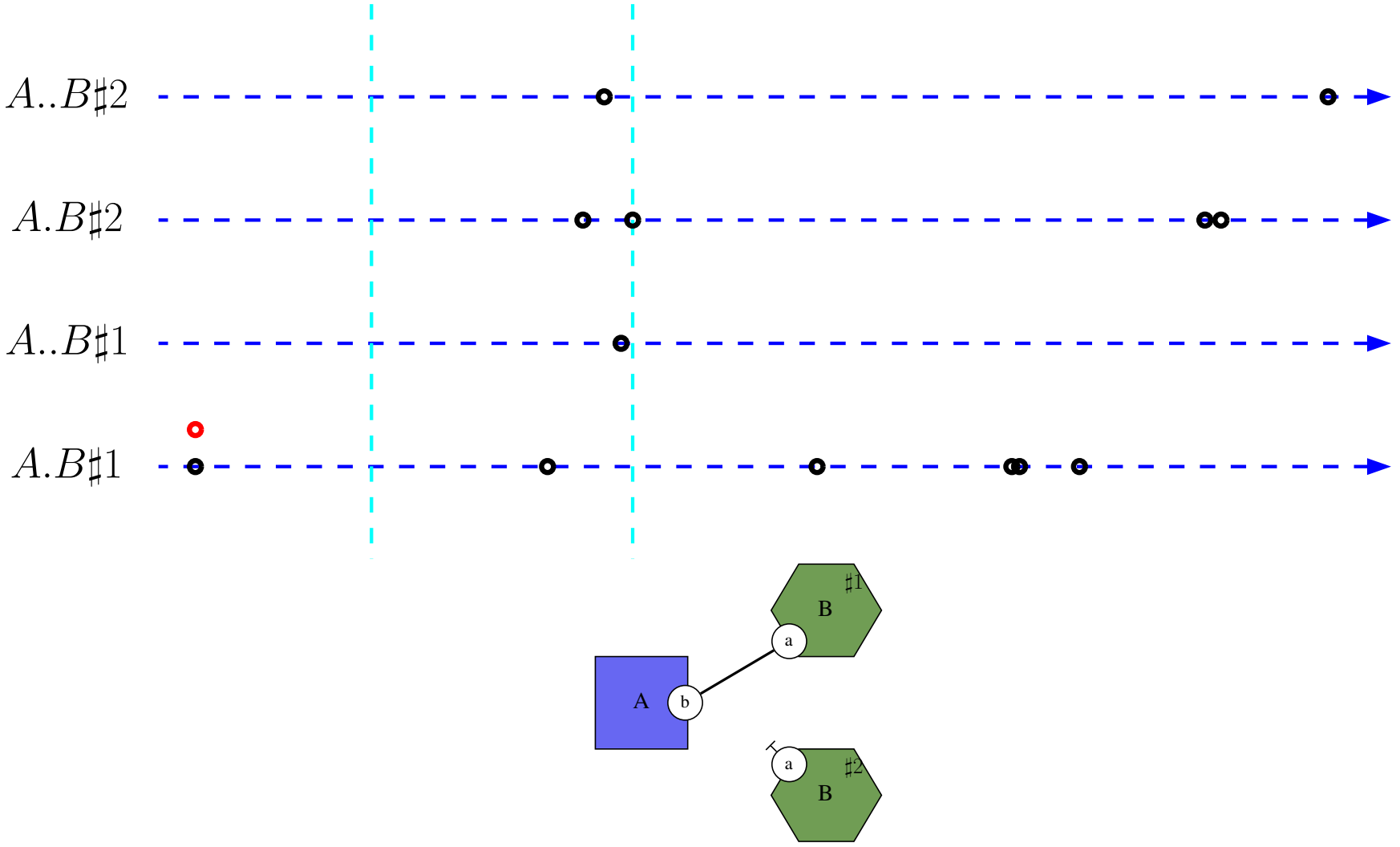




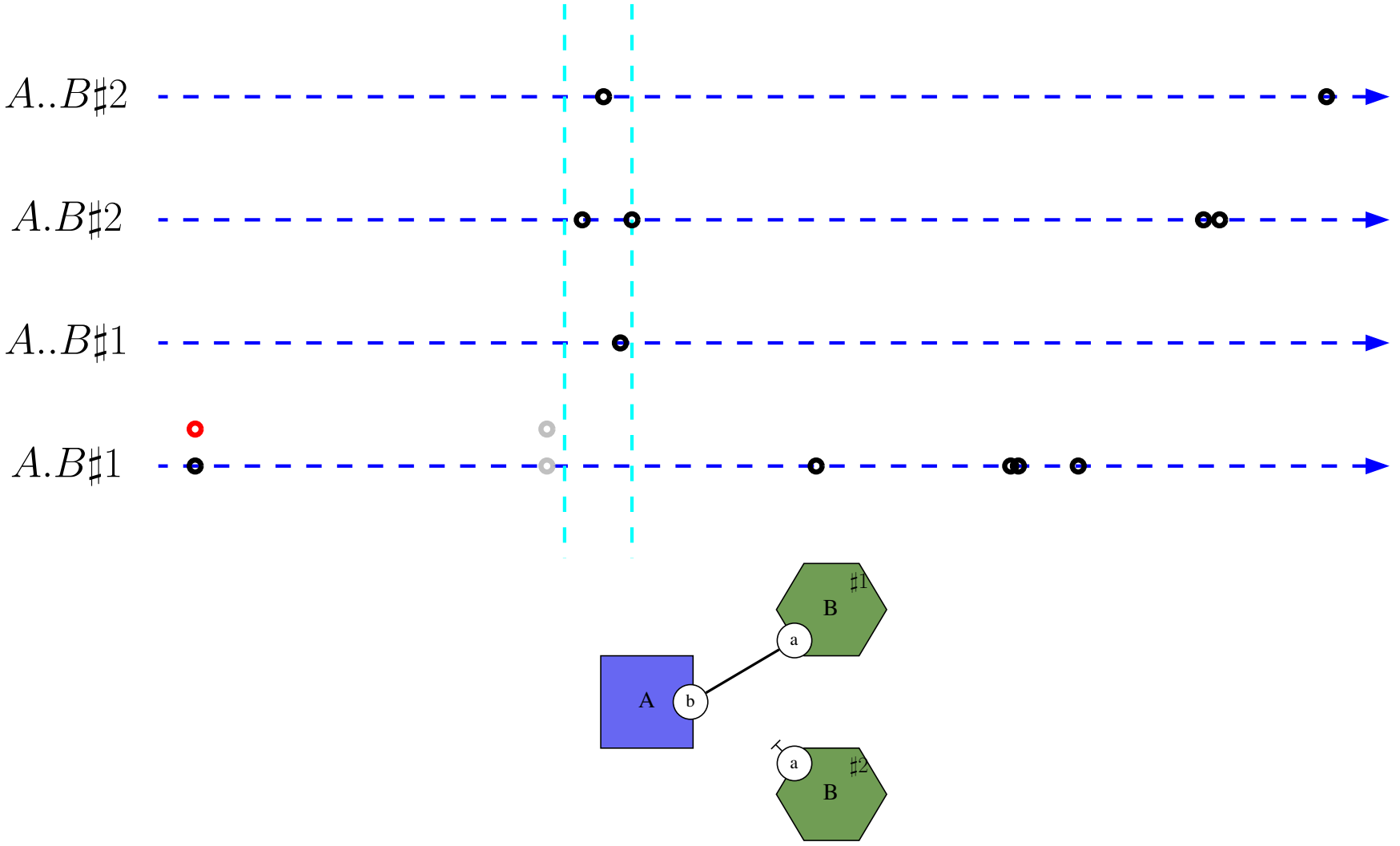
Counterfactual execution



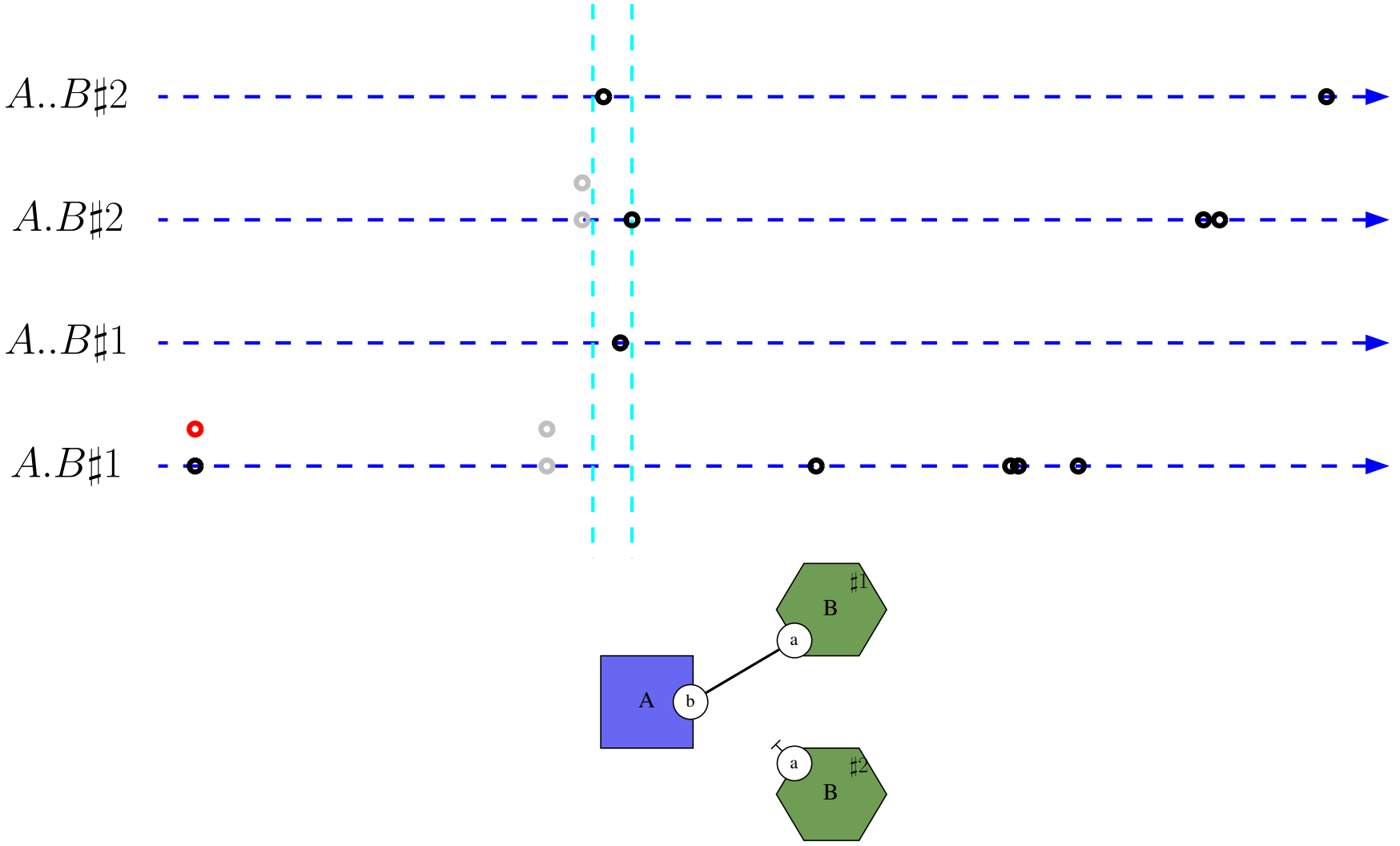
Counterfactual execution



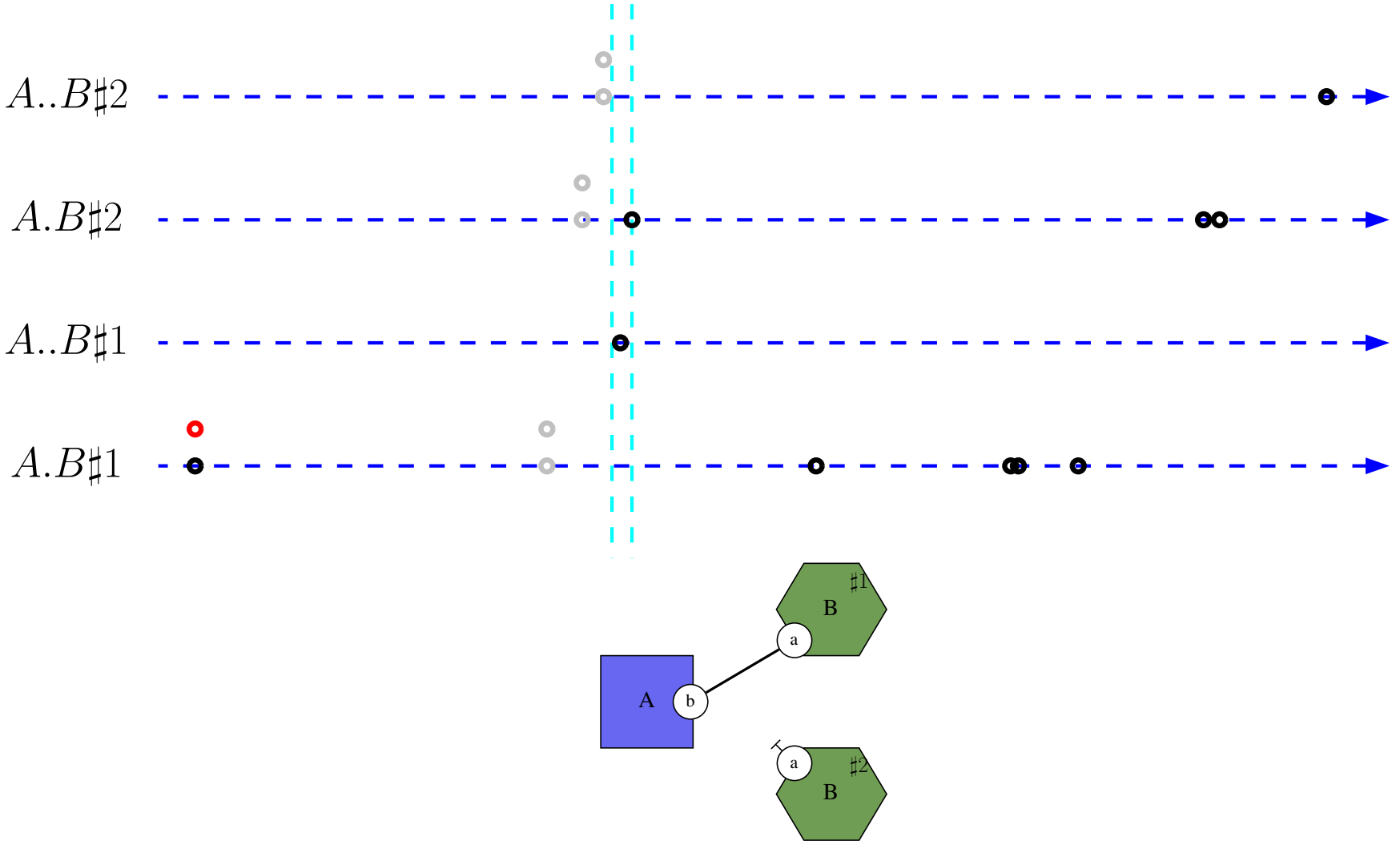
Counterfactual execution



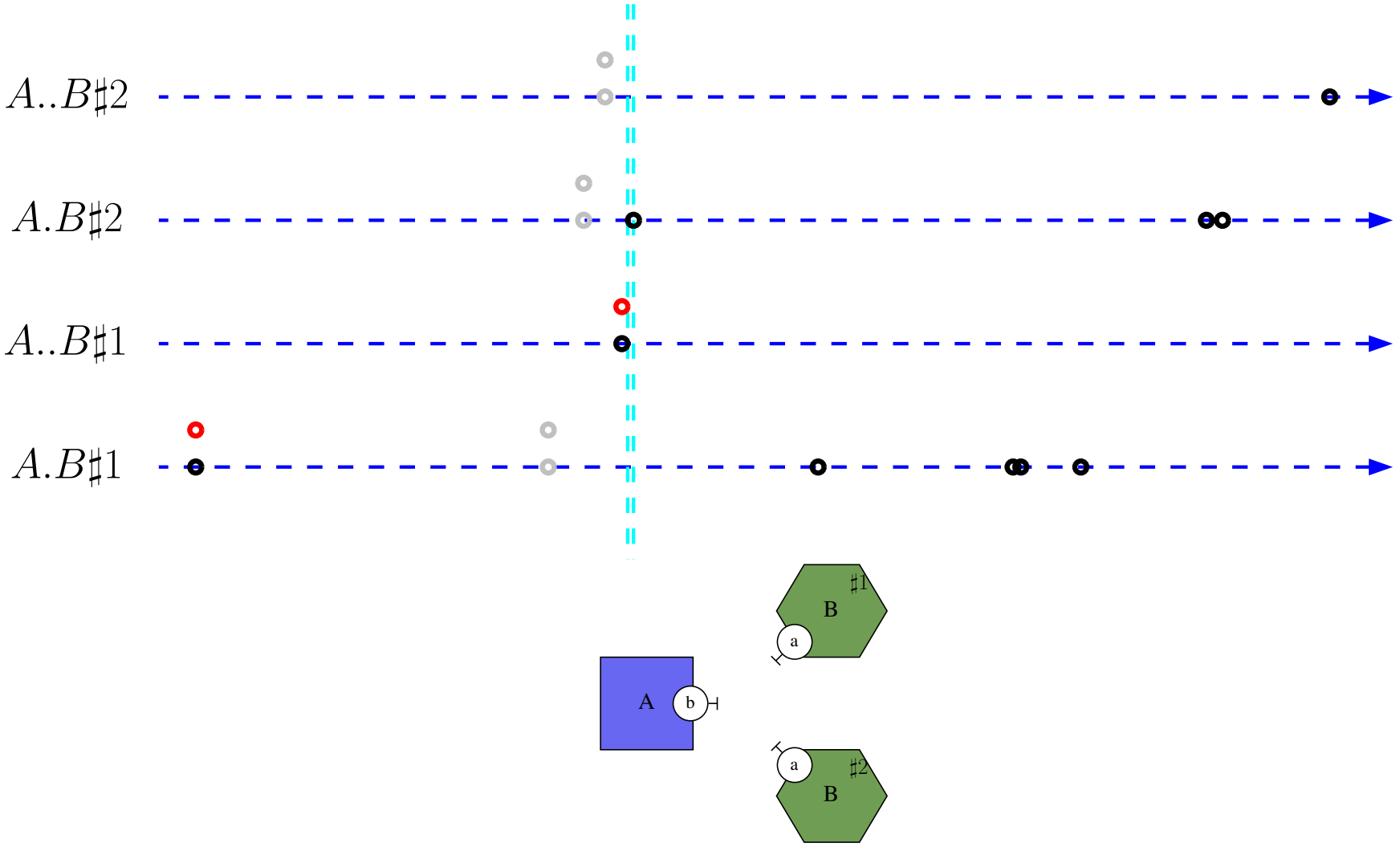
Counterfactual execution



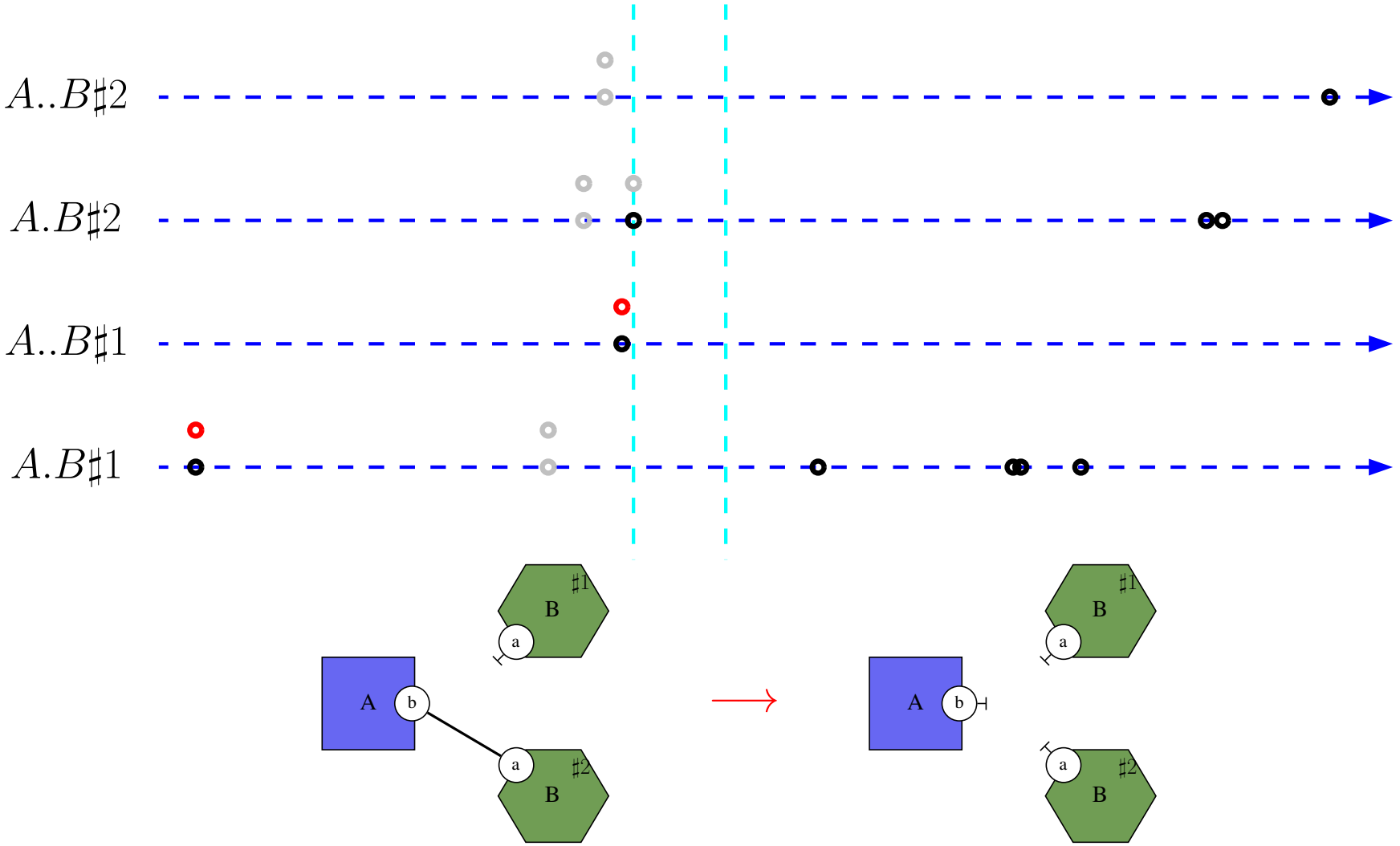
Counterfactual execution



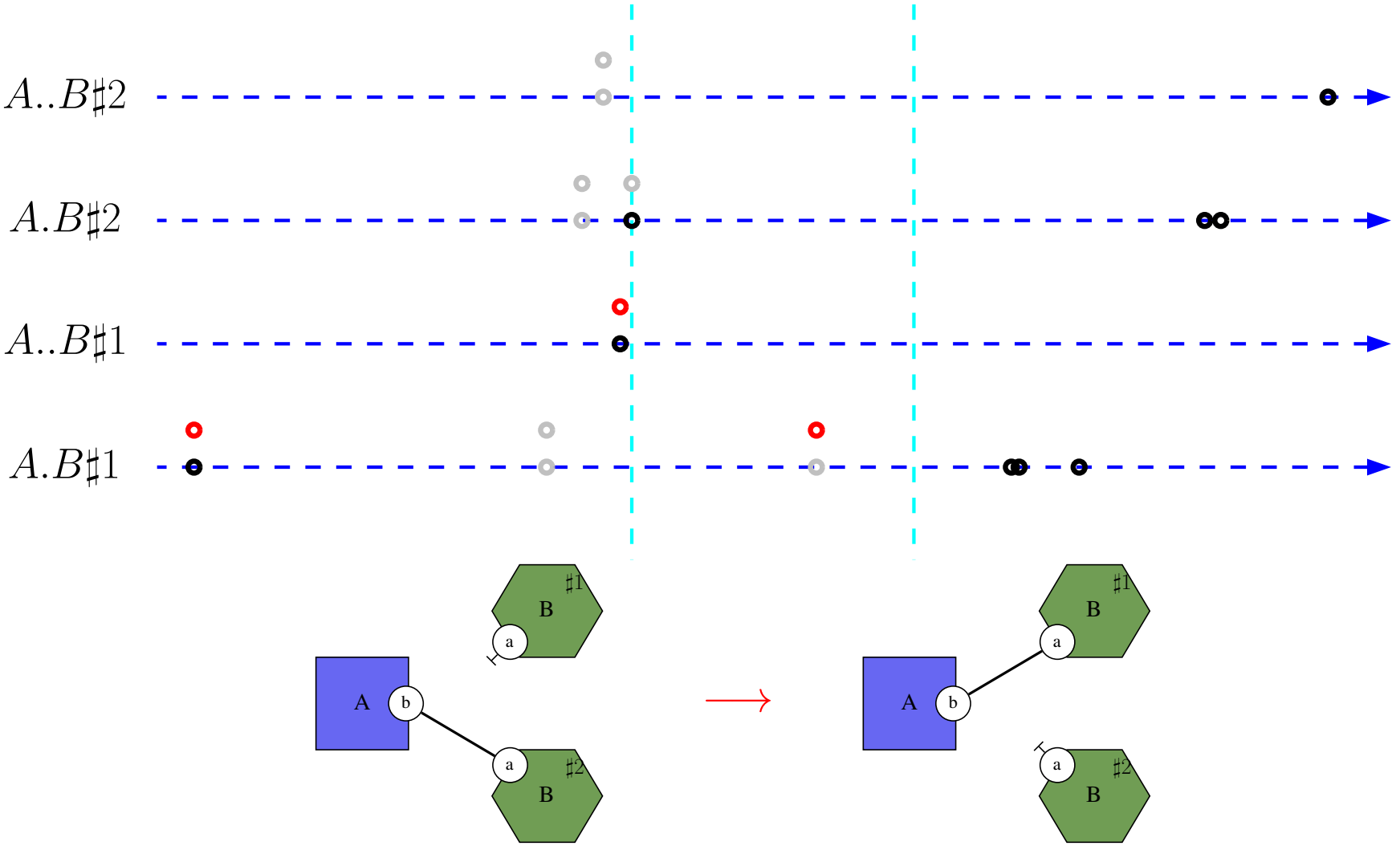
Counterfactual execution



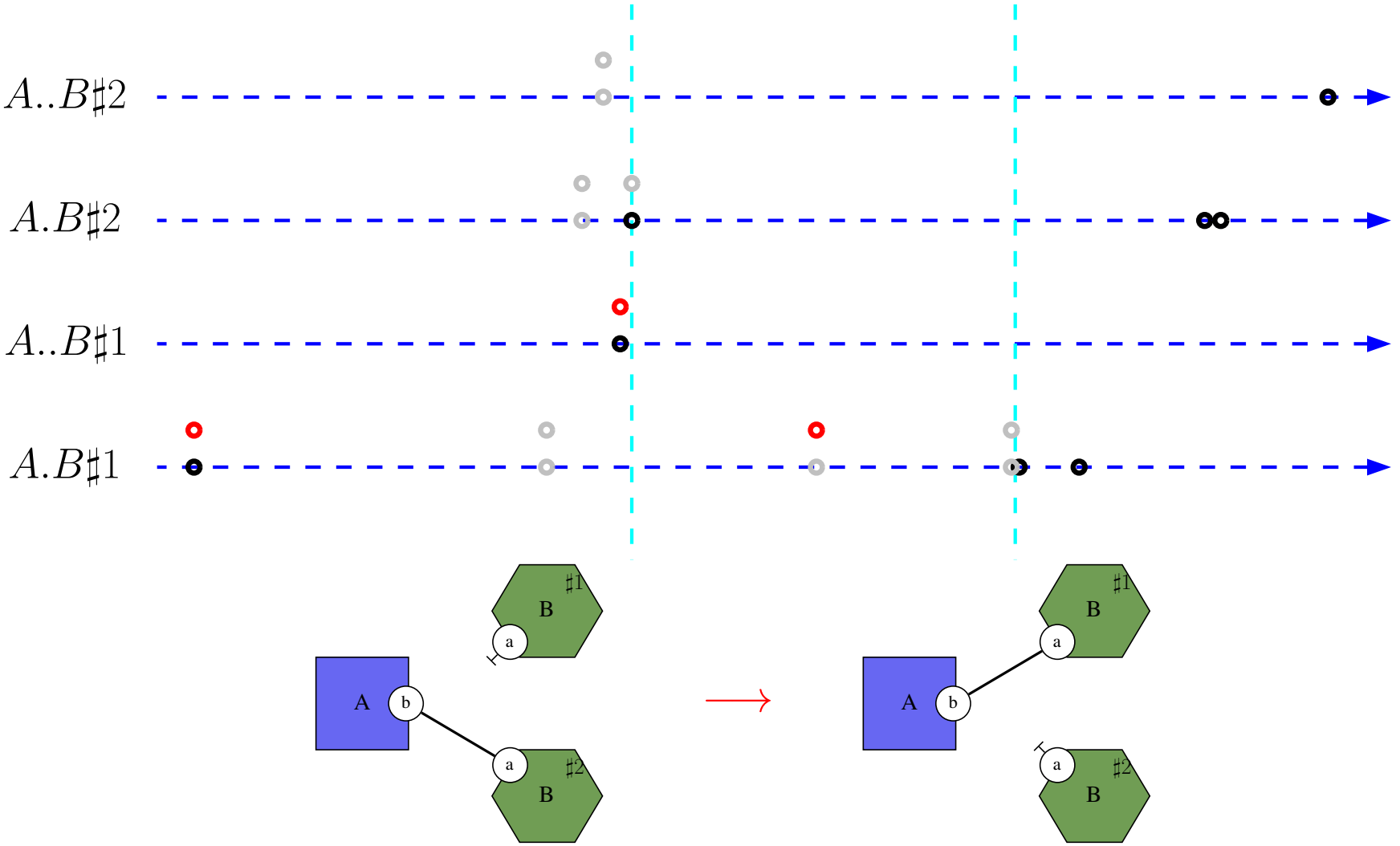
Counterfactual execution



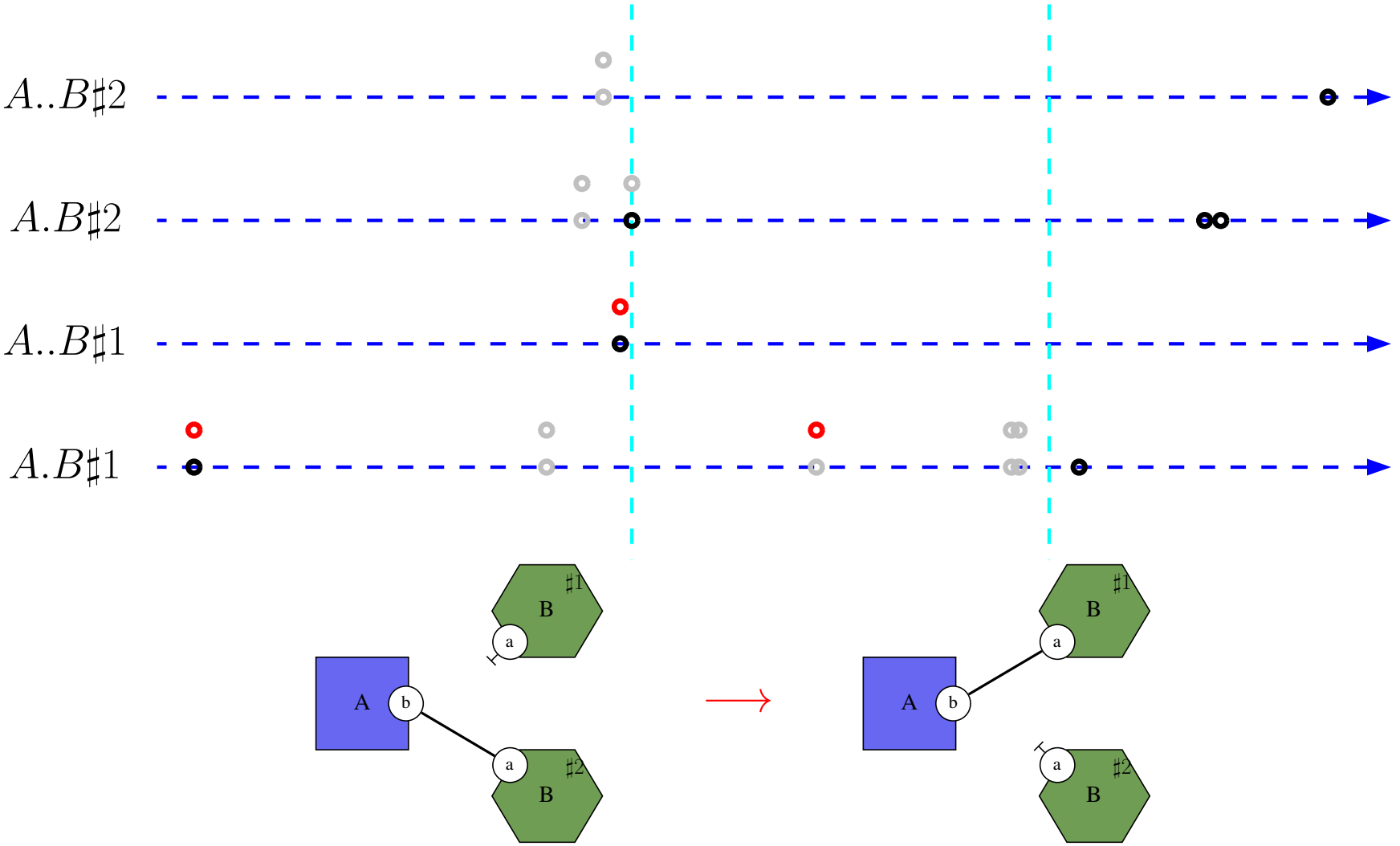
Counterfactual execution



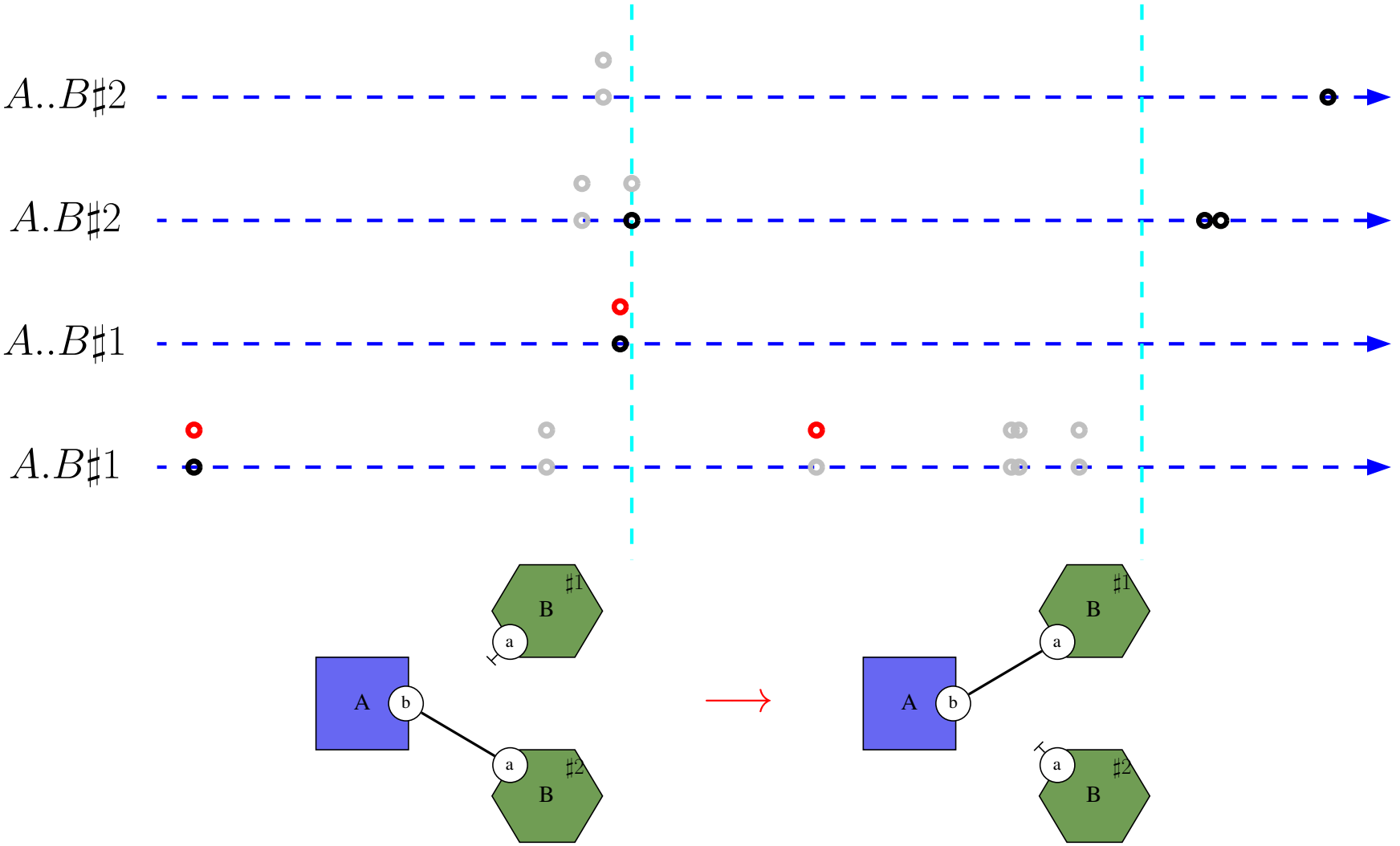
Counterfactual execution



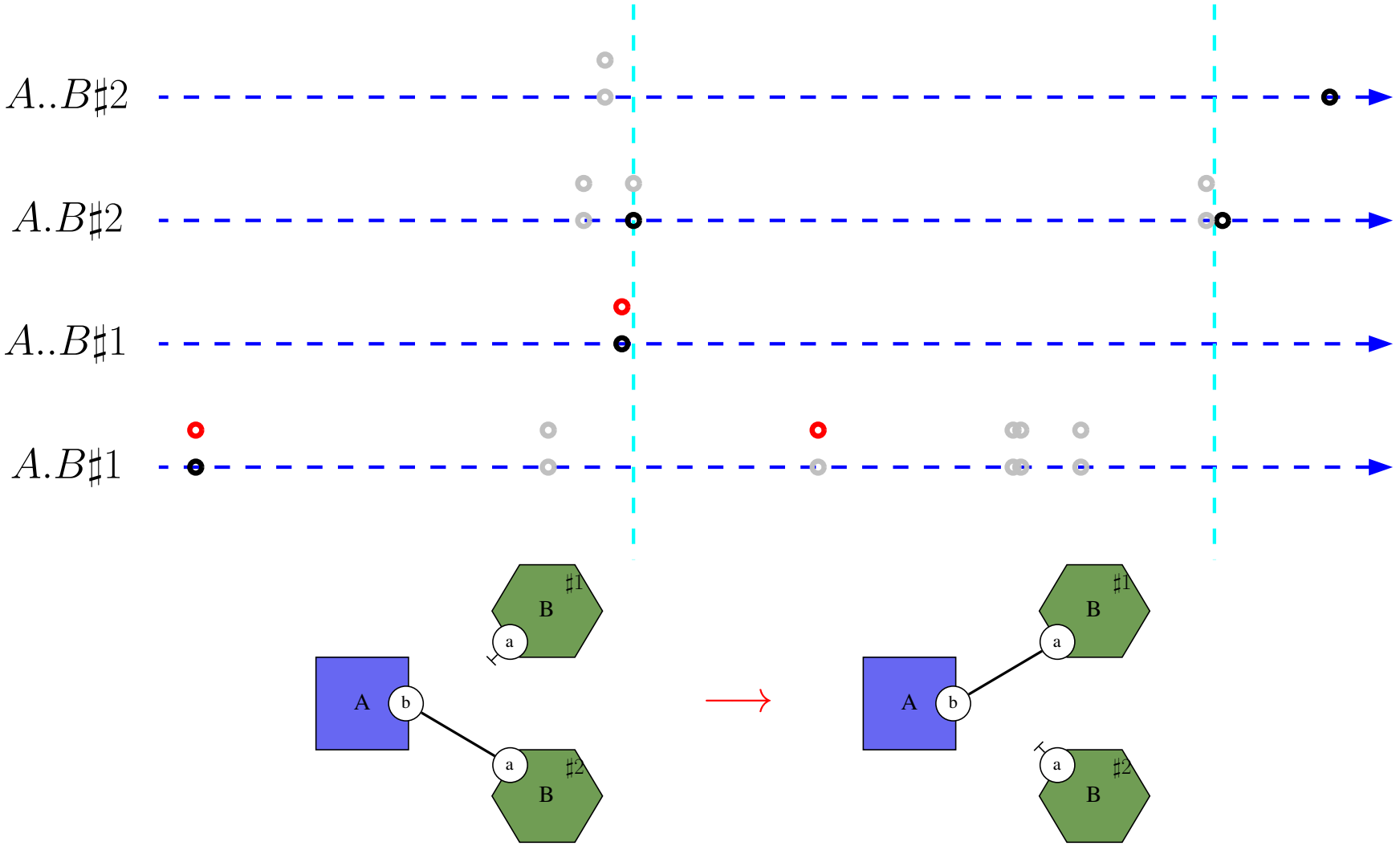
Counterfactual execution



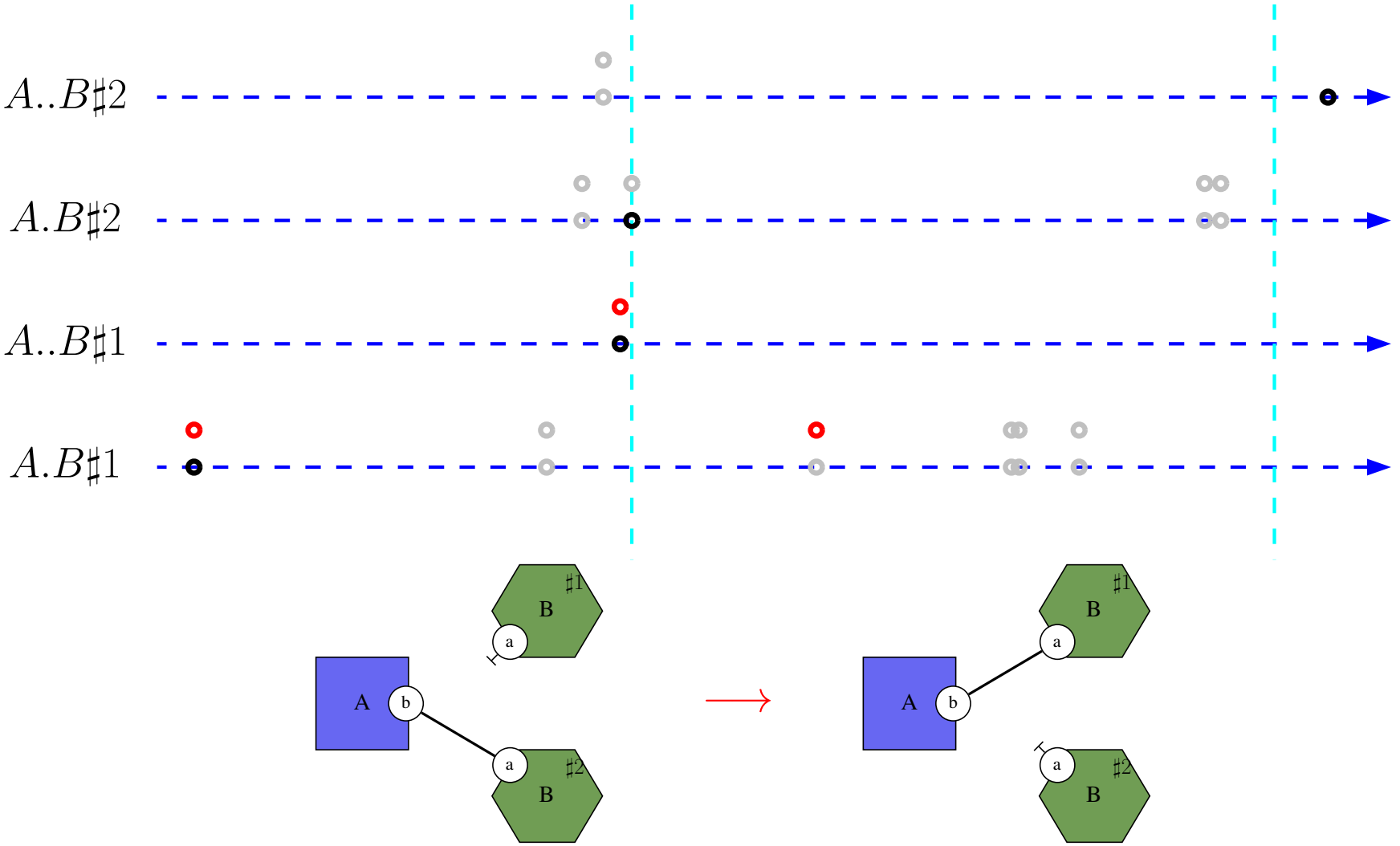
Counterfactual execution



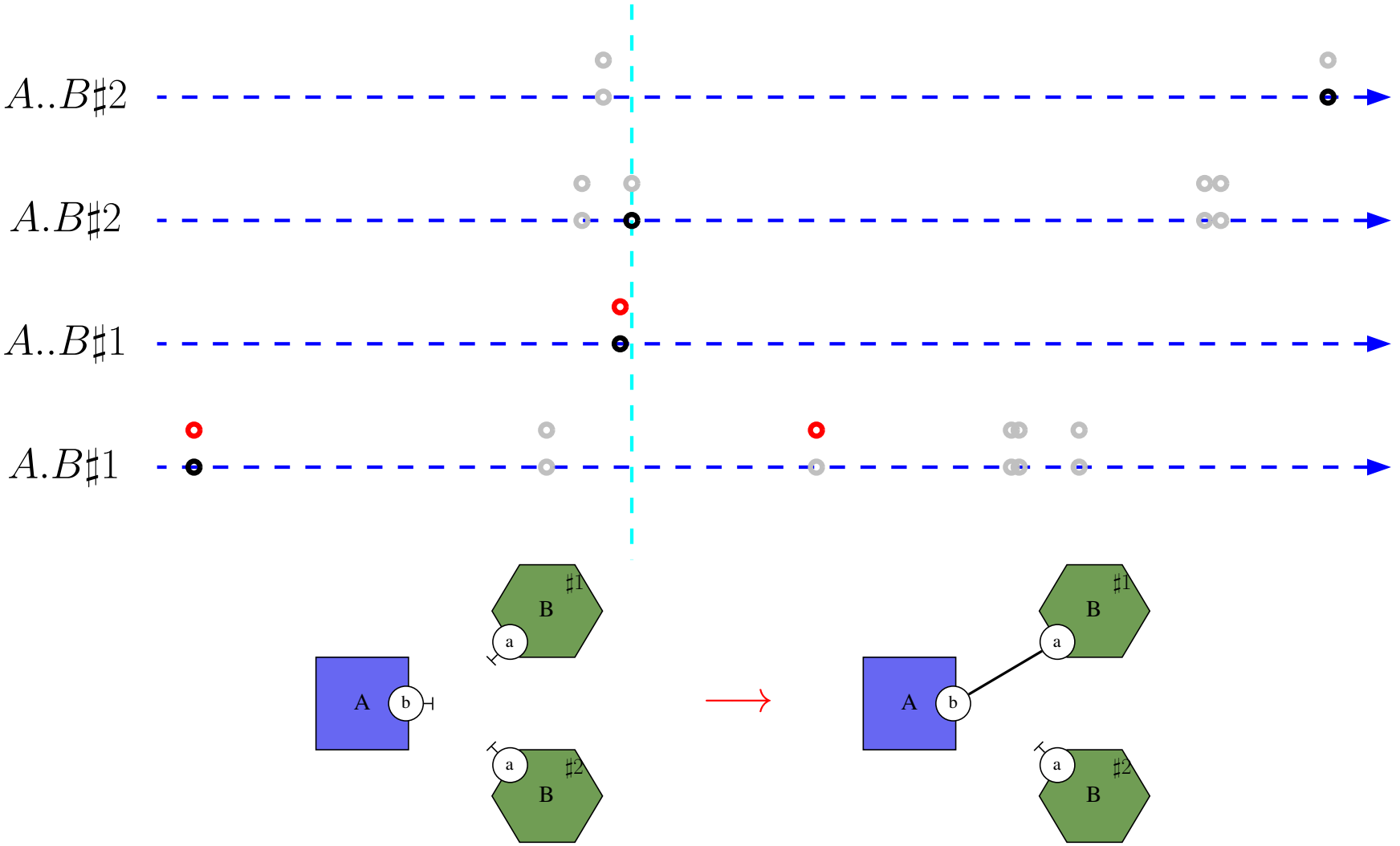
Counterfactual execution



Counterfactual execution



Counterfactual execution



Take home message about counterfactual execution

- Reasoning on what would have happened if a given event had not occurred in a specific execution trace.
- Given a prefix of trace, it defines a distribution of pairs of suffixes of traces

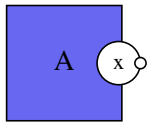
  (instead of a pair of distributions of suffixes of traces)

  according to whether or not the event under investigation has occurred
- It requires a semantics that describes the operational execution as a function of random draws.

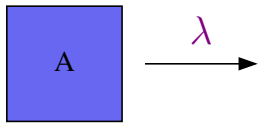
  The counterfactual execution is replayed with the same stream of random draws.
- This stronger semantics is arguable.

Practical activity

We consider 1000 occurrences of the following agent:



1. Use the simulator to check the time necessary to consume half of the agents with the following rule:



2. Same question with the combination of both following rules:

3. Empirically fit λ' so that both time are equal.
4. Compare both abundance curves and conclude.

Overview

1. Syntax
2. Markovian clocks
3. Stochastic semantics
4. Optimizations
5. Counter-factual execution
6. Bibliography

Bibliography

- Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jean Krivine: Rule-Based Modelling of Cellular Signalling. CONCUR 2007: 17-41
- Vincent Danos, Jérôme Feret, Walter Fontana, Jean Krivine: Scalable Simulation of Cellular Signaling Networks. APLAS 2007: 139-157
- Pierre Boutillier, Thomas Ehrhard, Jean Krivine: Incremental Update for Graph Rewriting. ESOP 2017: 201-228
- Jonathan Laurent, Jean Yang, Walter Fontana: Counterfactual Resimulation for Causal Analysis of Rule-Based Models. IJCAI 2018: 1882-1890
- Pierre Boutillier, Mutaamba Maasha, Xing Li, Héctor F. Medina-Abarca, Jean Krivine, Jérôme Feret, Ioana Cristescu, Angus G. Forbes, Walter Fontana: The Kappa platform for rule-based modeling. Bioinform. 34(13): i583-i592 (2018)