Symbolic Simulation of Clocks

Guillaume Baudart Timothy Bourke Marc Pouzet

Work In Progress

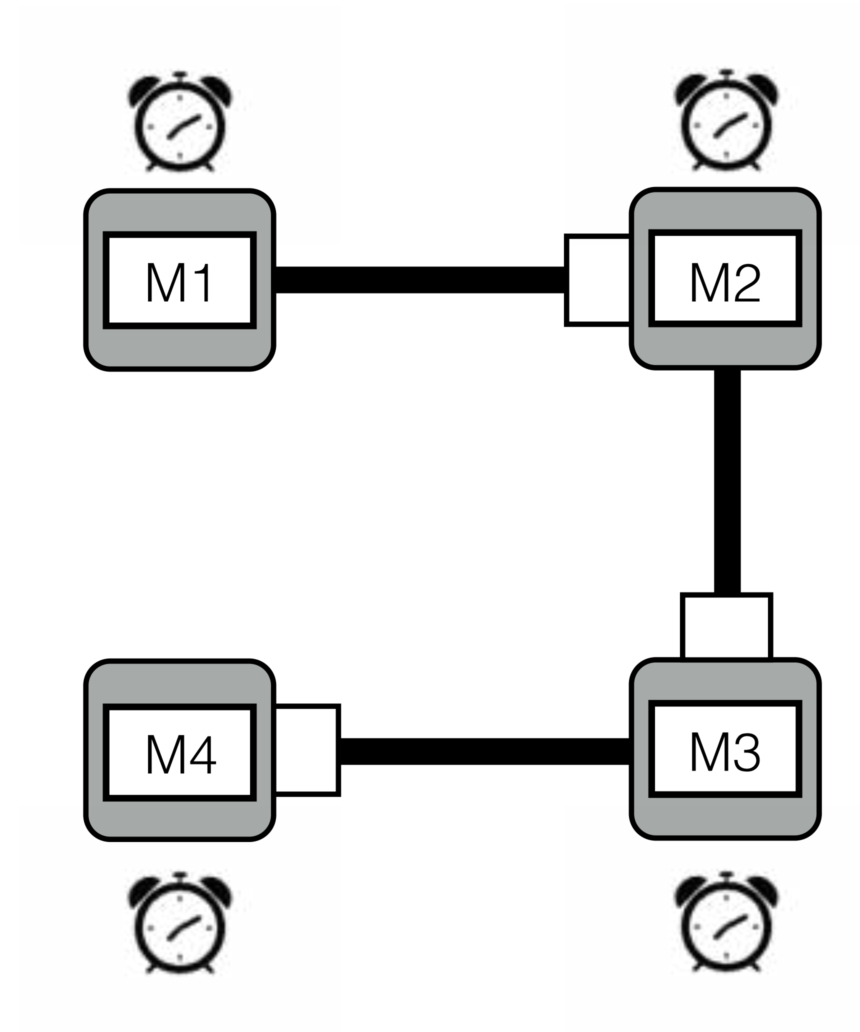
Example: Quasi-Periodic Architectures

- A set of “quasi-periodic” processes with local clocks and nominal period T^n (jitter ε)

$$0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or}$$
$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss



“Cooking Book” [Caspi 2000]

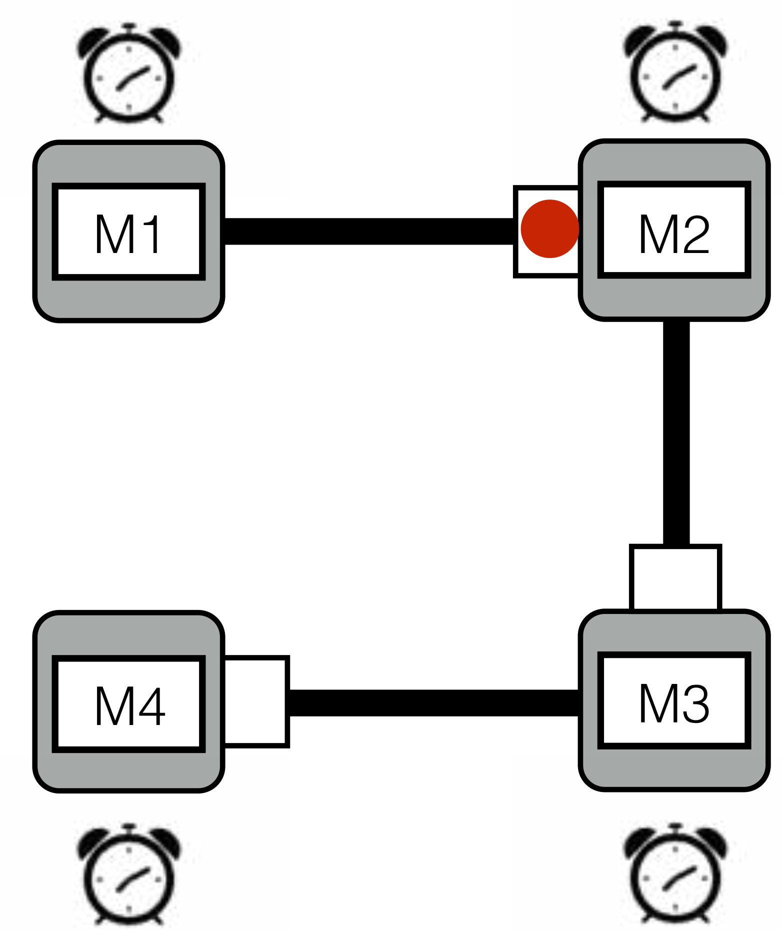
Example: Quasi-Periodic Architectures

- A set of "quasi-periodic" processes with local clocks and nominal period T^n (jitter ε)

$$0 < T_{\min} \le T^n \le T_{\max} \quad \text{or}$$
$$T^n - \varepsilon \le \kappa_i - \kappa_{i-1} \le T^n + \varepsilon$$

$(\kappa_i)_{i\in\mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss



"Cooking Book" [Caspi 2000]

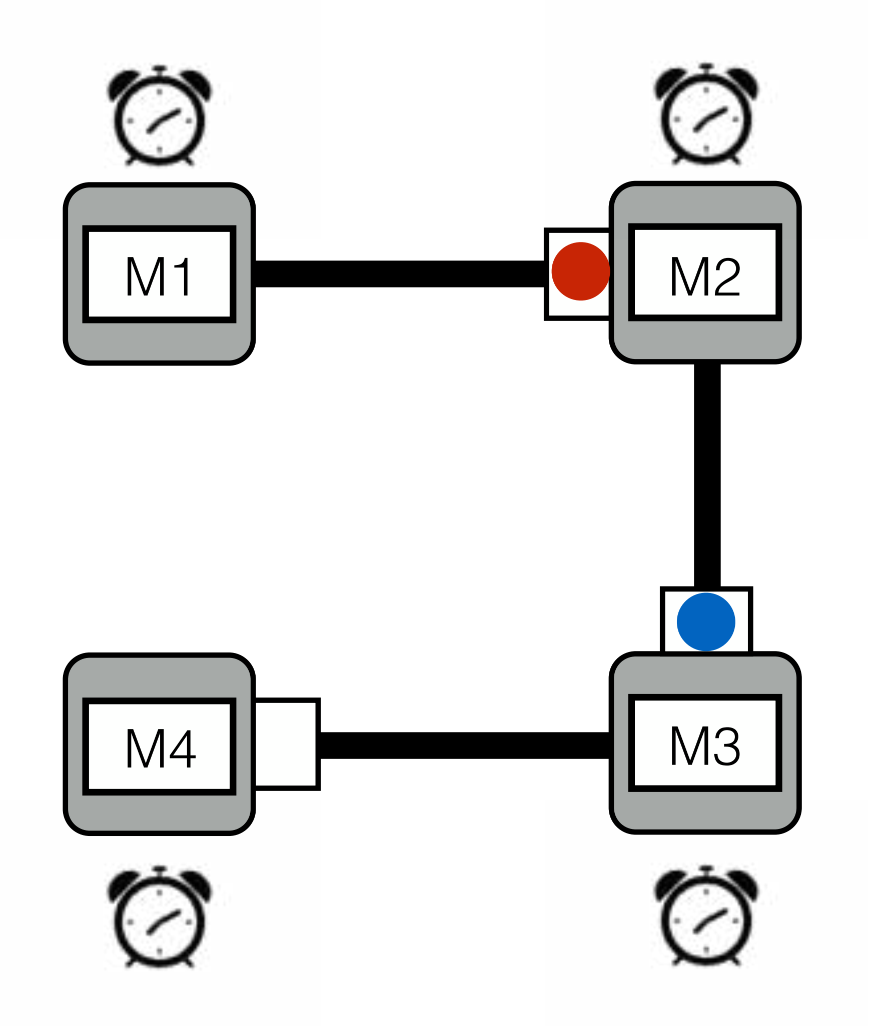
Example: Quasi-Periodic Architectures

- A set of “quasi-periodic” processes with local clocks and nominal period T^n (jitter ε)

$$0 < T_{\min} \leq T^n \leq T_{\max} \quad \text{or}$$
$$T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

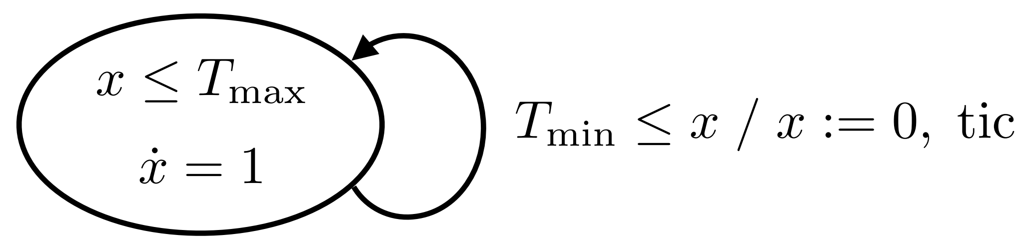
- Buffered communication without message inversion or loss



“Cooking Book” [Caspi 2000]

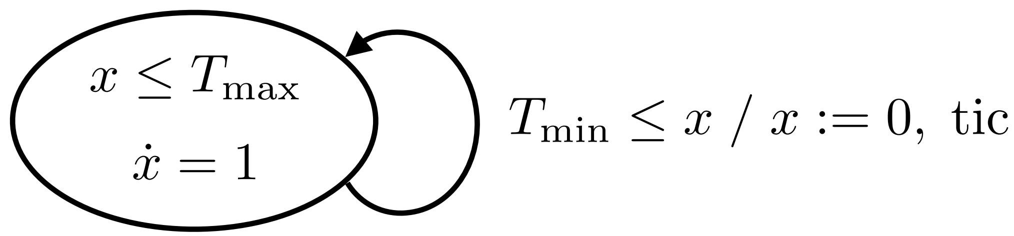
Example: Quasi-Periodic Architectures

Fuzzy Metronome



Example: Quasi-Periodic Architectures

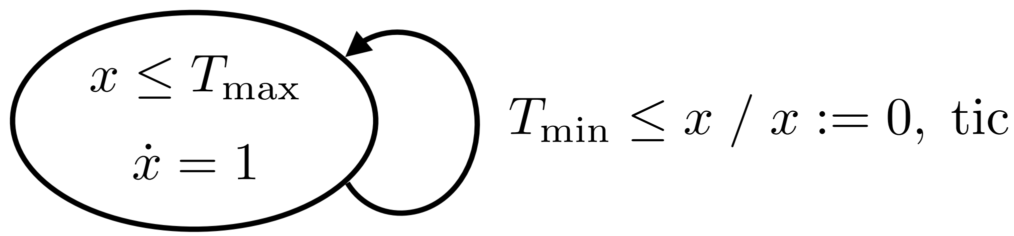
Fuzzy Metronome




```
let hybrid metro (      tmin, tmax) = tic where
  rec clock x reset tic()
  and tic = present          (tmin <= x <= tmax) -> ()
```

Example: Quasi-Periodic Architectures

Fuzzy Metronome

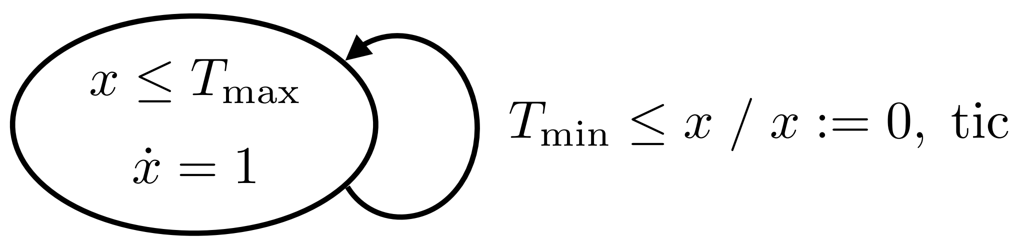



```
let hybrid metro (       tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present            (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

Example: Quasi-Periodic Architectures

Fuzzy Metronome



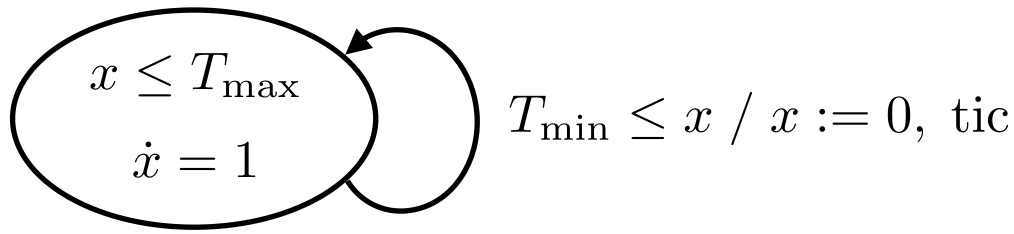

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

Example: Quasi-Periodic Architectures

⏰ Fuzzy Metronome



tmin = 3
tmax = 5

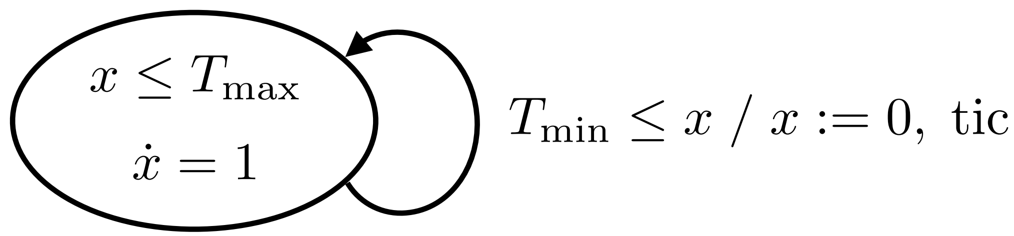
```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

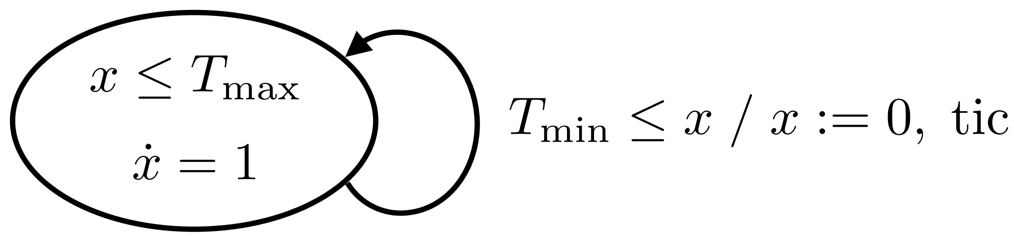
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

```
[x = 0]
```

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

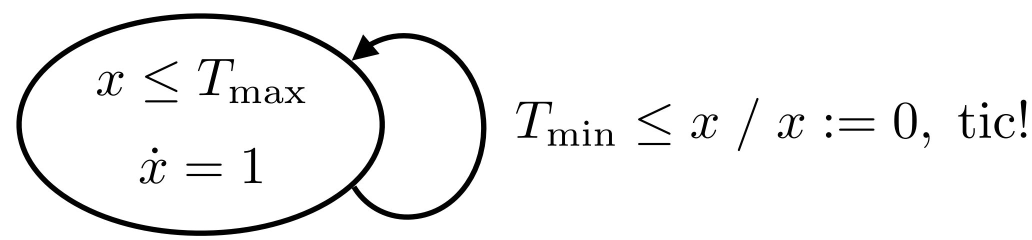
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

[x = 0] ⟶ [x = 2.4]

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

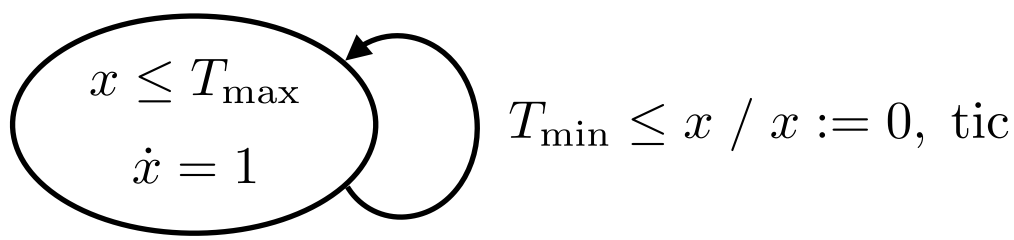
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$[x = 0] \longrightarrow [x = 2.4] \xrightarrow{\texttt{in\_x/}} [x = 2.4]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

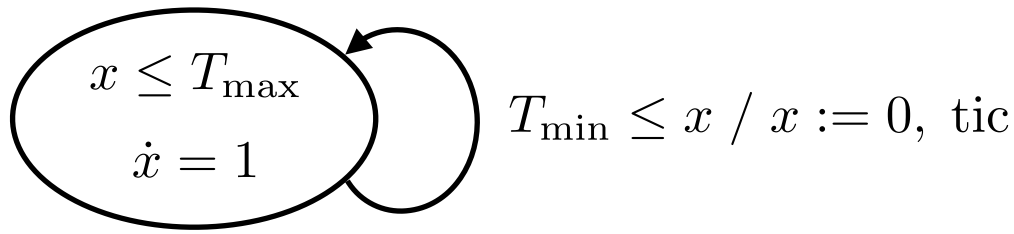
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$[x = 0] \longrightarrow [x = 2.4] \xrightarrow{\texttt{in\_x/}} [x = 2.4] \longrightarrow [x = 3.5]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

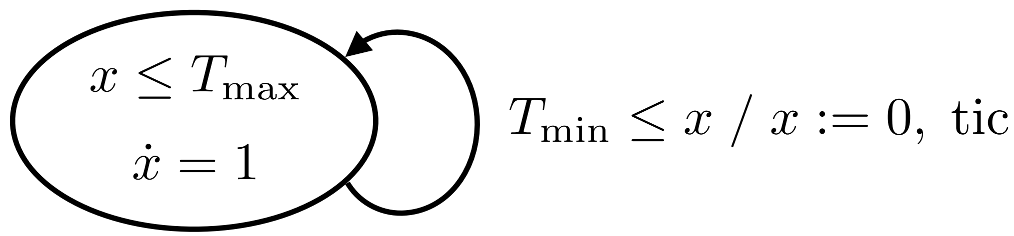
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$$[\mathtt{x = 0}] \longrightarrow [\mathtt{x = 2.4}] \xrightarrow{\mathtt{in\_x/}} [\mathtt{x = 2.4}] \longrightarrow [\mathtt{x = 3.5}] \xrightarrow{\mathtt{in\_x/tic}} [\mathtt{x = 0}]$$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

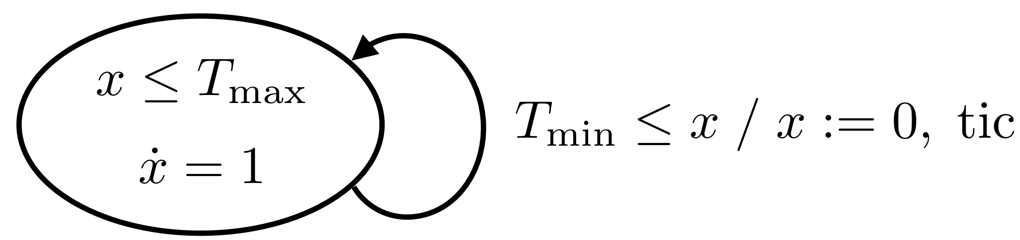
Non-determinism is handle via additional inputs

$[x = 0] \longrightarrow [x = 2.4] \xrightarrow{\text{in\_x/}} [x = 2.4] \longrightarrow [x = 3.5] \xrightarrow{\text{in\_x/tic}} [x = 0]$

$[x = 0]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

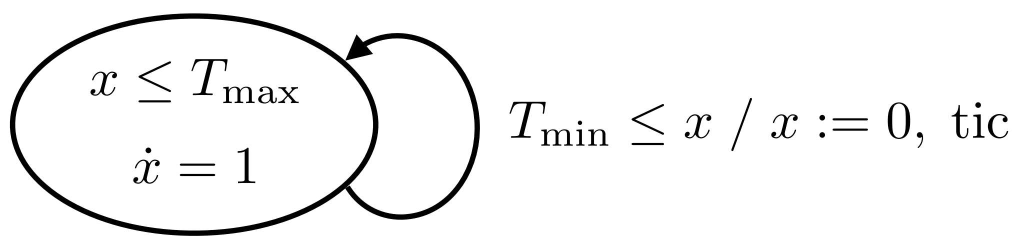
Non-determinism is handle via additional inputs

$\texttt{[x = 0]} \longrightarrow \texttt{[x = 2.4]} \xrightarrow{\texttt{in\_x/}} \texttt{[x = 2.4]} \longrightarrow \texttt{[x = 3.5]} \xrightarrow{\texttt{in\_x/tic}} \texttt{[x = 0]}$

$\texttt{[x = 0]} \longrightarrow \texttt{[x = 1.1]}$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

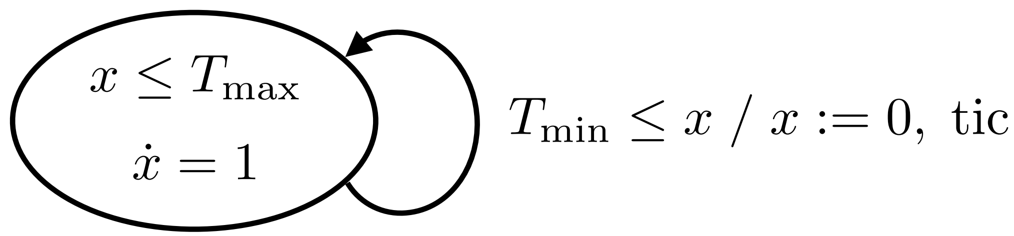
Non-determinism is handle via additional inputs

$$[\mathtt{x = 0}] \longrightarrow [\mathtt{x = 2.4}] \xrightarrow{\mathtt{in\_x/}} [\mathtt{x = 2.4}] \longrightarrow [\mathtt{x = 3.5}] \xrightarrow{\mathtt{in\_x/tic}} [\mathtt{x = 0}]$$

$$[\mathtt{x = 0}] \longrightarrow [\mathtt{x = 1.1}] \xrightarrow{\mathtt{in\_x/}} [\mathtt{x = 1.2}]$$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

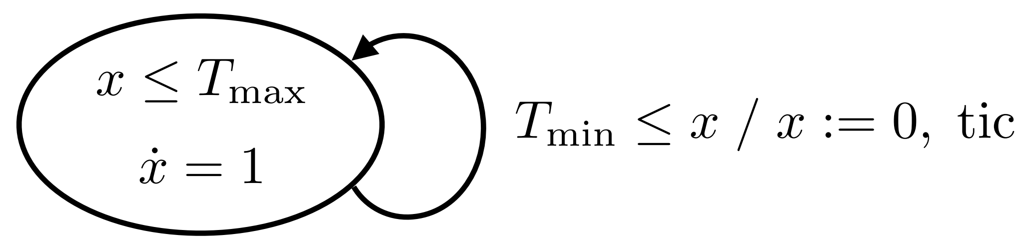
Non-determinism is handle via additional inputs

$[x = 0] \longrightarrow [x = 2.4] \xrightarrow{\texttt{in\_x/}} [x = 2.4] \longrightarrow [x = 3.5] \xrightarrow{\texttt{in\_x/tic}} [x = 0]$

$[x = 0] \longrightarrow [x = 1.1] \xrightarrow{\texttt{in\_x/}} [x = 1.2] \longrightarrow [x = 4.2]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

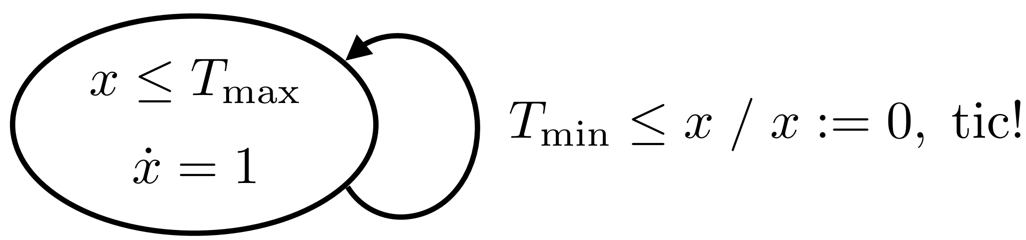
Non-determinism is handle via additional inputs

$$[x = 0] \longrightarrow [x = 2.4] \xrightarrow{\texttt{in\_x/}} [x = 2.4] \longrightarrow [x = 3.5] \xrightarrow{\texttt{in\_x/tic}} [x = 0]$$

$$[x = 0] \longrightarrow [x = 1.1] \xrightarrow{\texttt{in\_x/}} [x = 1.2] \longrightarrow [x = 4.2] \xrightarrow{\texttt{in\_x/tic}} [x = 0]$$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

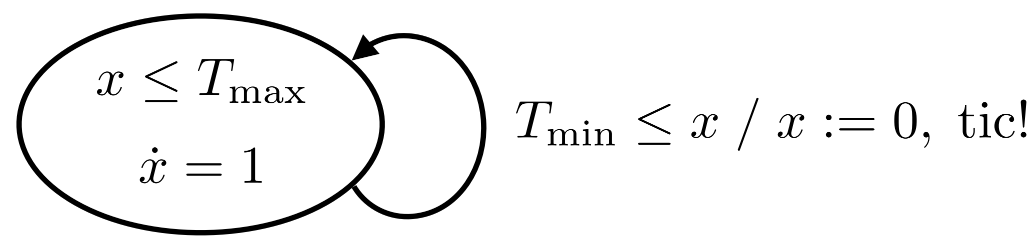
```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

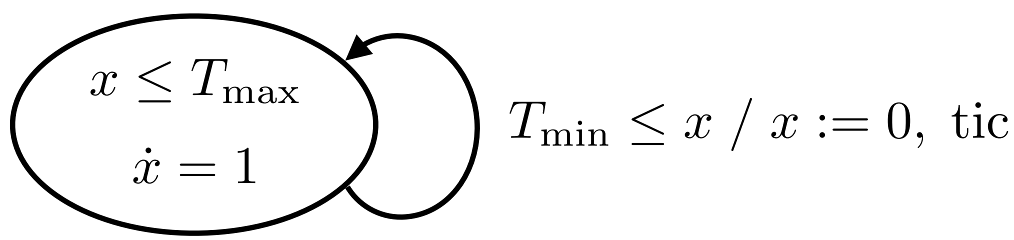
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

[0≤x≤3]

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

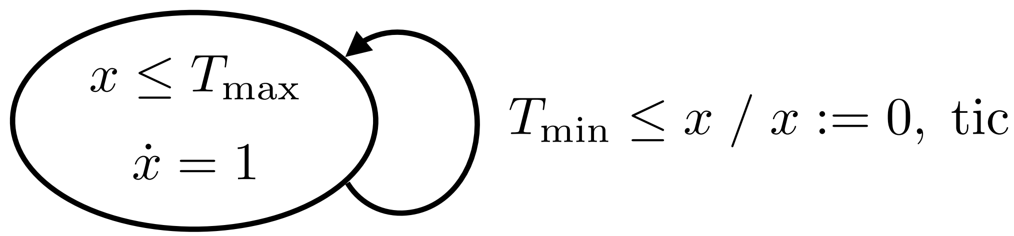
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$[0 \le x \le 3] \xrightarrow{\texttt{in\_x/}} [0 \le x \le 3]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

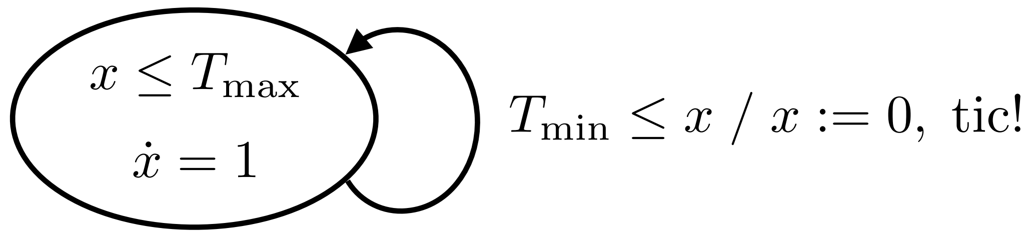
Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$[0 \leq x \leq 3] \xrightarrow{\texttt{in\_x/}} [0 \leq x \leq 3] \xrightarrow{\texttt{wait}} [3 \leq x \leq 5]$

Example: Quasi-Periodic Architectures

Fuzzy Metronome



tmin = 3
tmax = 5

```
let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()
```

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

$$[0 \leq x \leq 3] \xrightarrow{\texttt{in\_x/}} [0 \leq x \leq 3] \xrightarrow{\texttt{wait}} [3 \leq x \leq 5] \xrightarrow{\texttt{in\_x/tic}} [0 \leq x \leq 3]$$

Symbolic Simulation

Definition: Two simulation states (q,s) and (q’,s’) are equivalent if q = q’ and actions(s) = actions(s’)

Symbolic Simulation

Definition: Two simulation states (q,s) and (q’,s’) are equivalent if q = q’ and actions(s) = actions(s’)

For each state, there is a set of enabled actions:

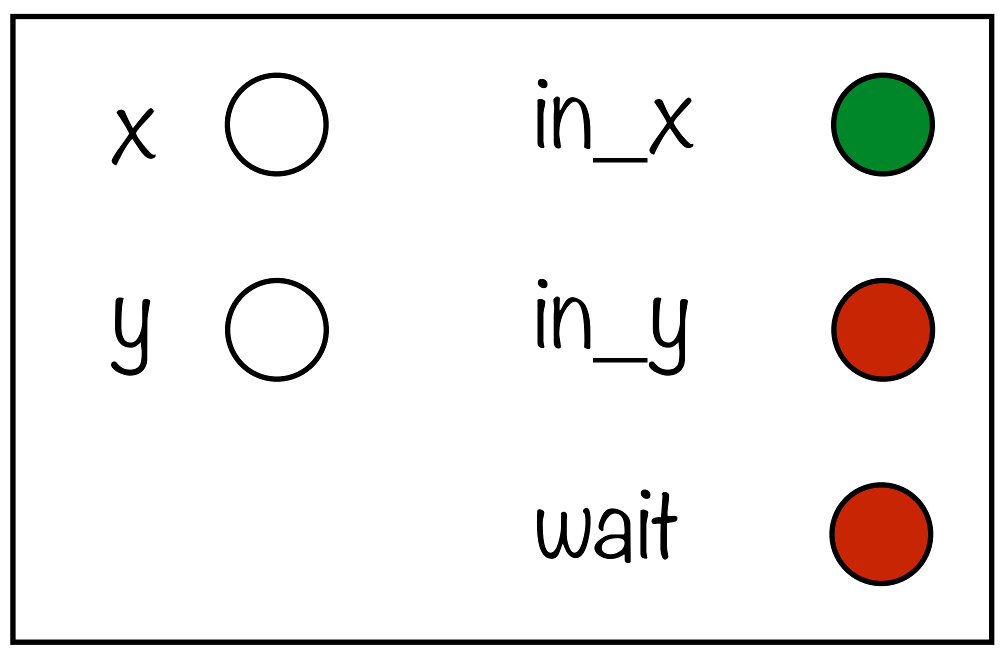
- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse

Symbolic Simulation

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse



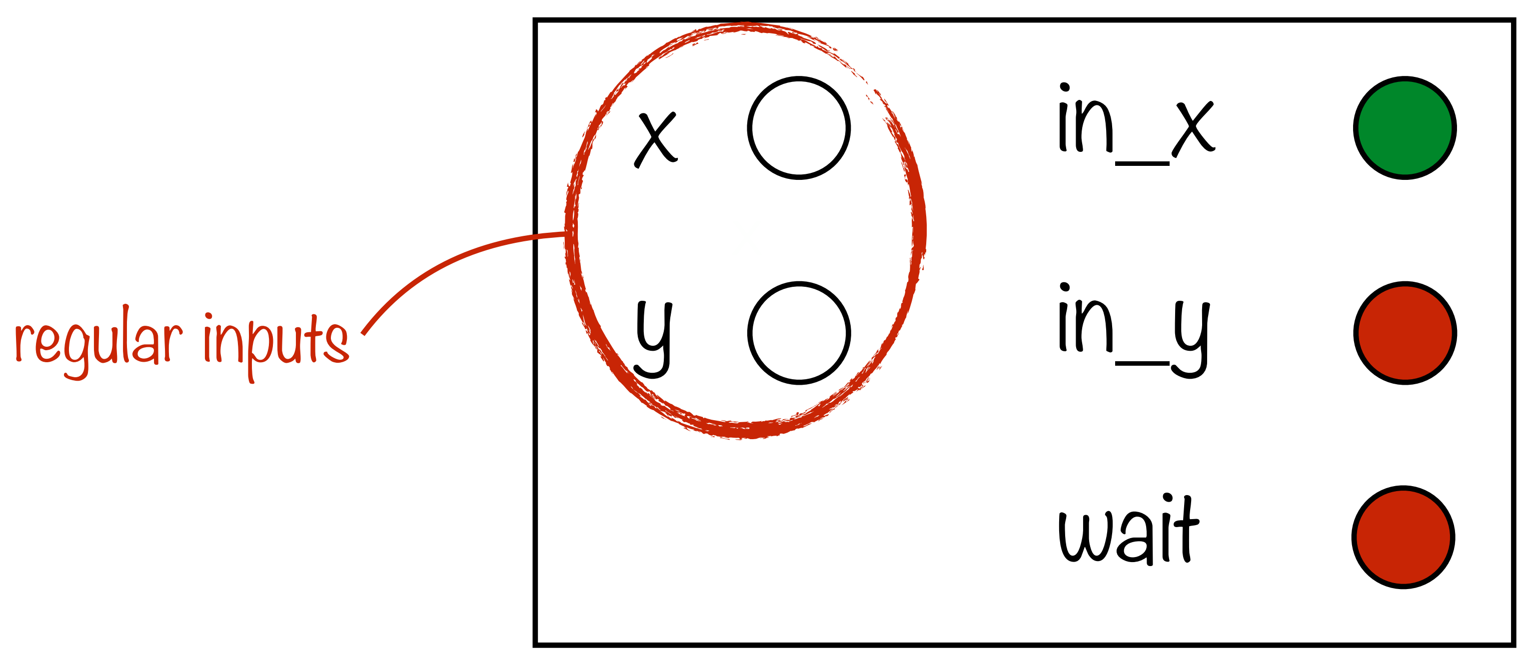
Simulation Box

Symbolic Simulation

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse



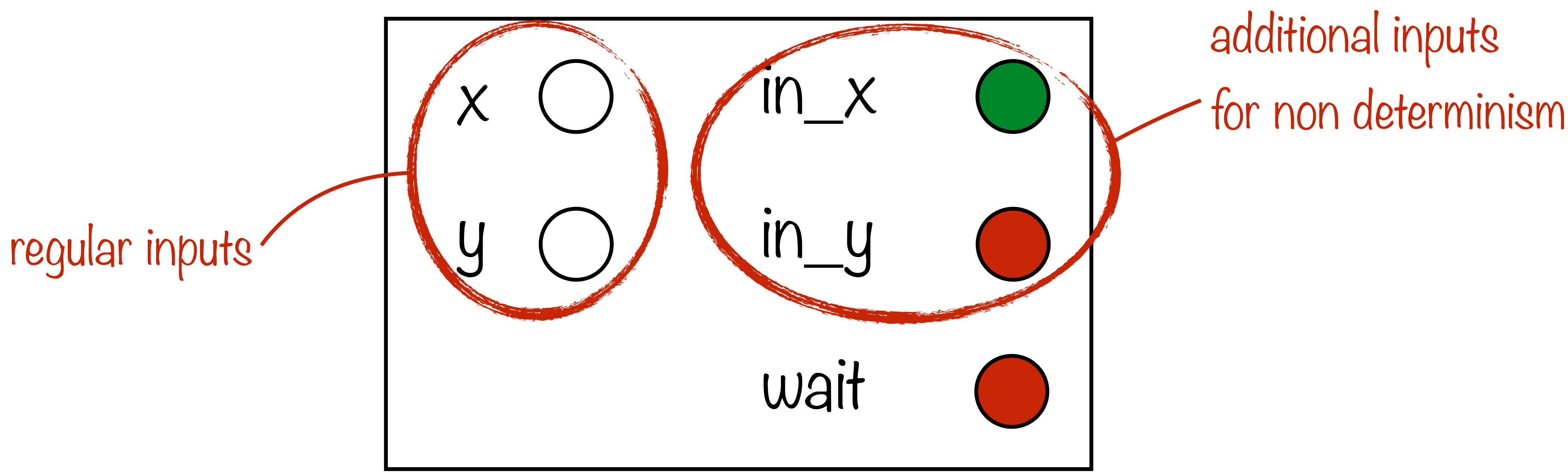
Simulation Box

Symbolic Simulation

Definition: Two simulation states (q,s) and (q’,s’) are equivalent if q = q’ and actions(s) = actions(s’)

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse



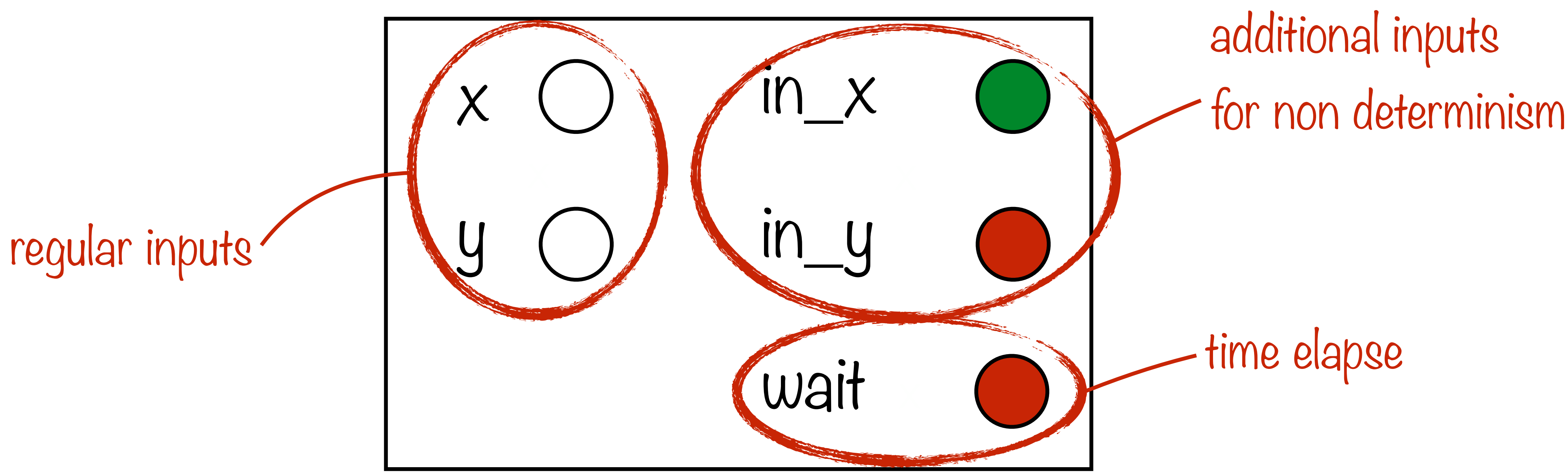
Simulation Box

Symbolic Simulation

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse



Simulation Box

Symbolic Simulation

Comparison with existing tool

Australian Walk

Symbolic Simulation

Comparison with existing tool

Australian Walk

Symbolic Simulation

Comparison with existing tool

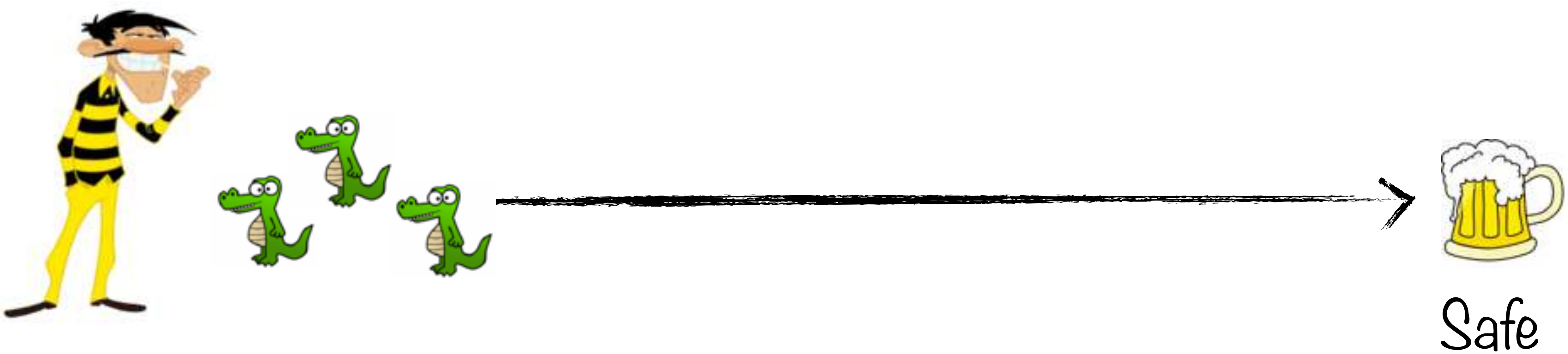
Australian Walk

Safe

Symbolic Simulation

Comparison with existing tool

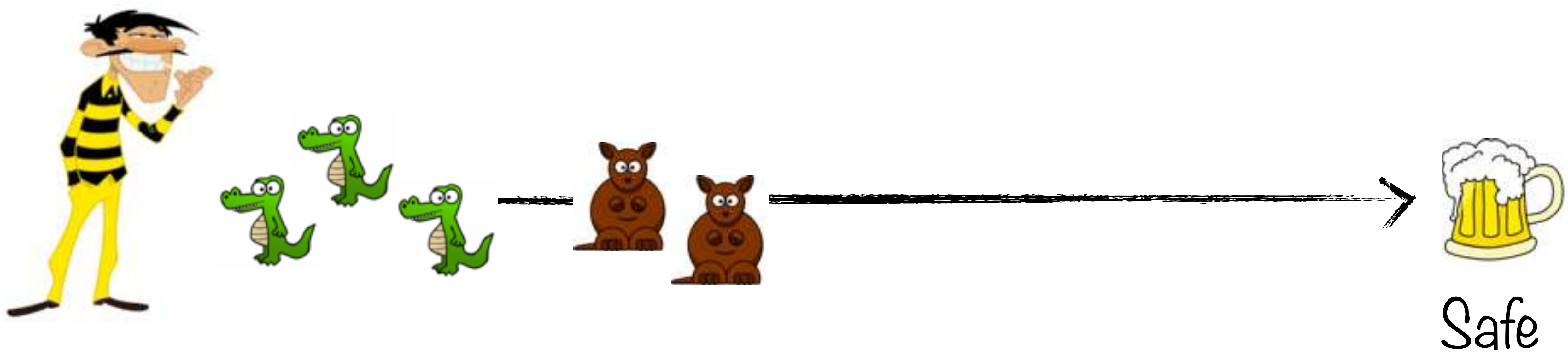
Australian Walk



Symbolic Simulation

Comparison with existing tool

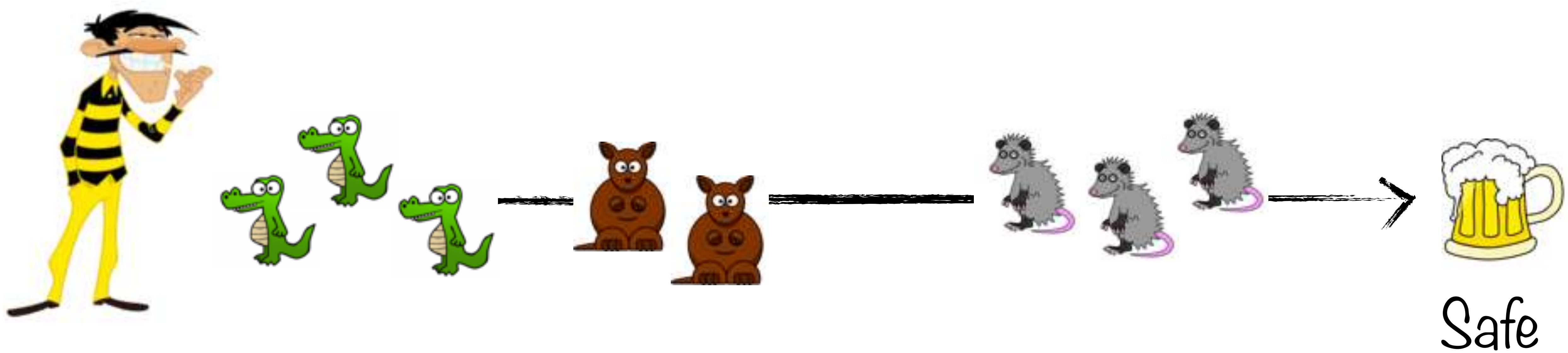
Australian Walk



Symbolic Simulation

Comparison with existing tool

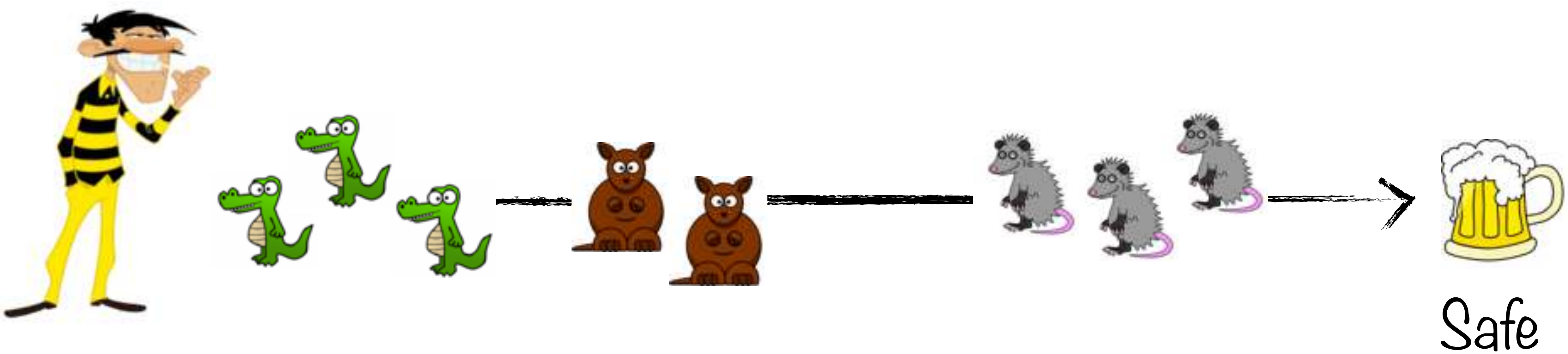
Australian Walk



Symbolic Simulation

Comparison with existing tool

Australian Walk



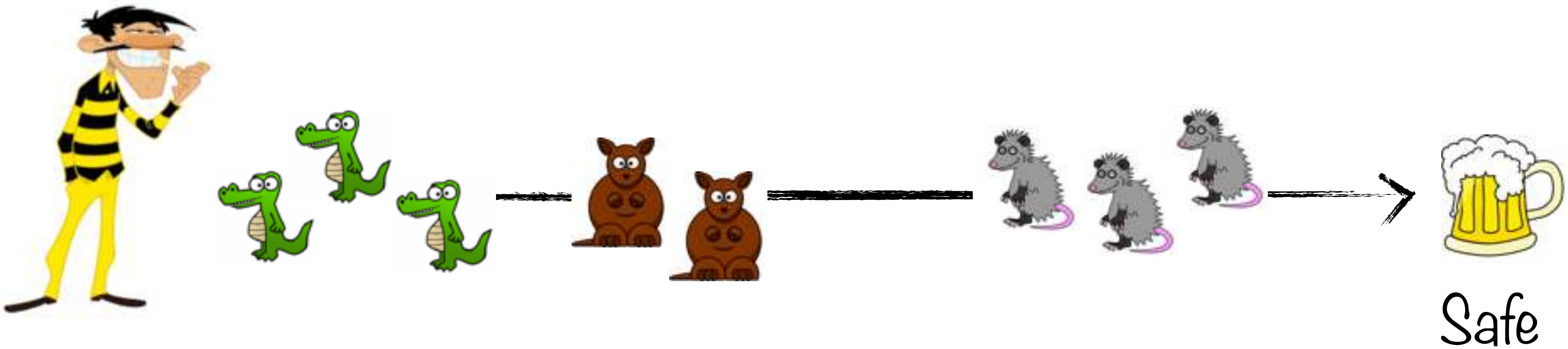
Demo: Uppaal

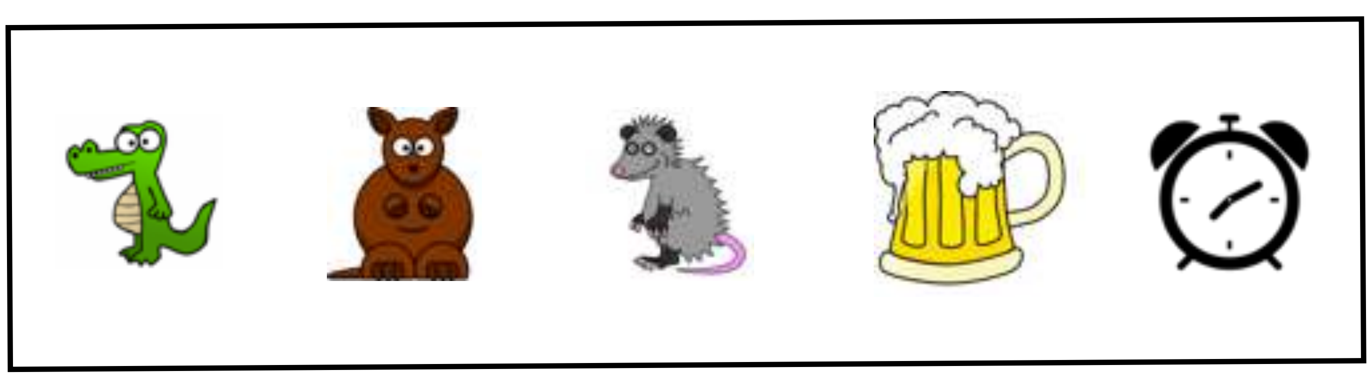
[Larsen et al. 1997]

Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

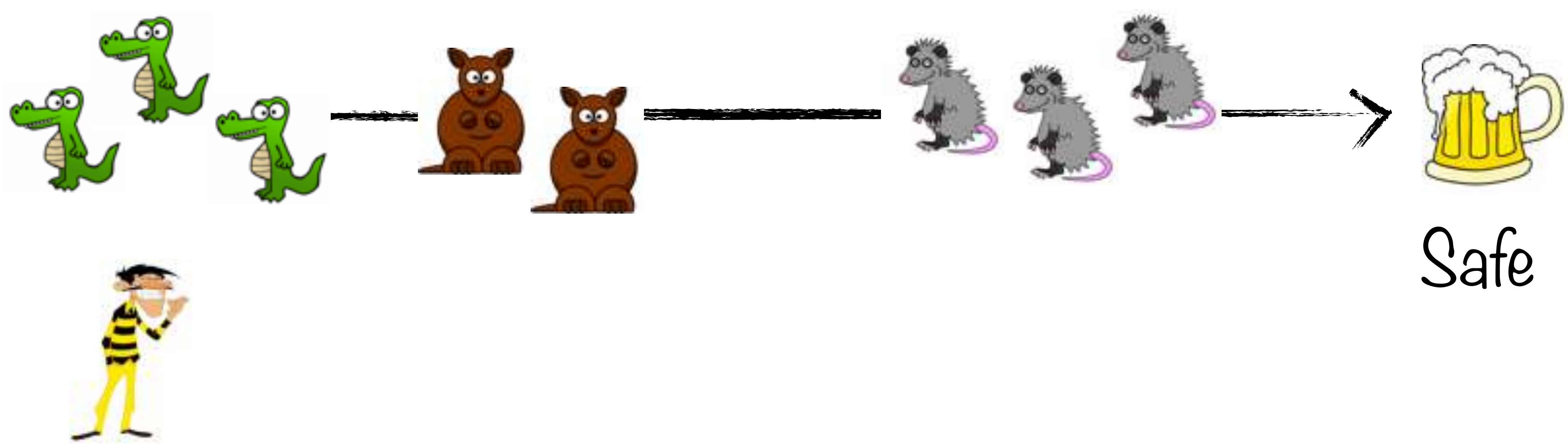


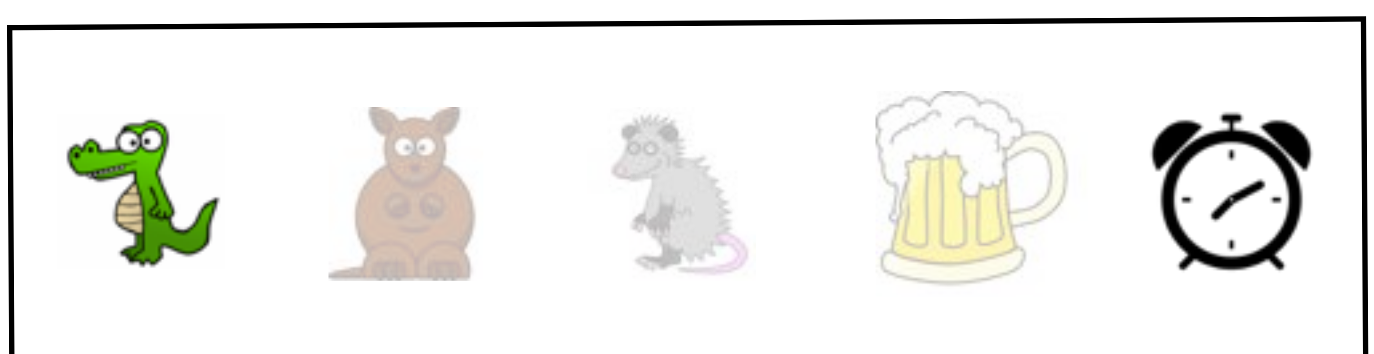


Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

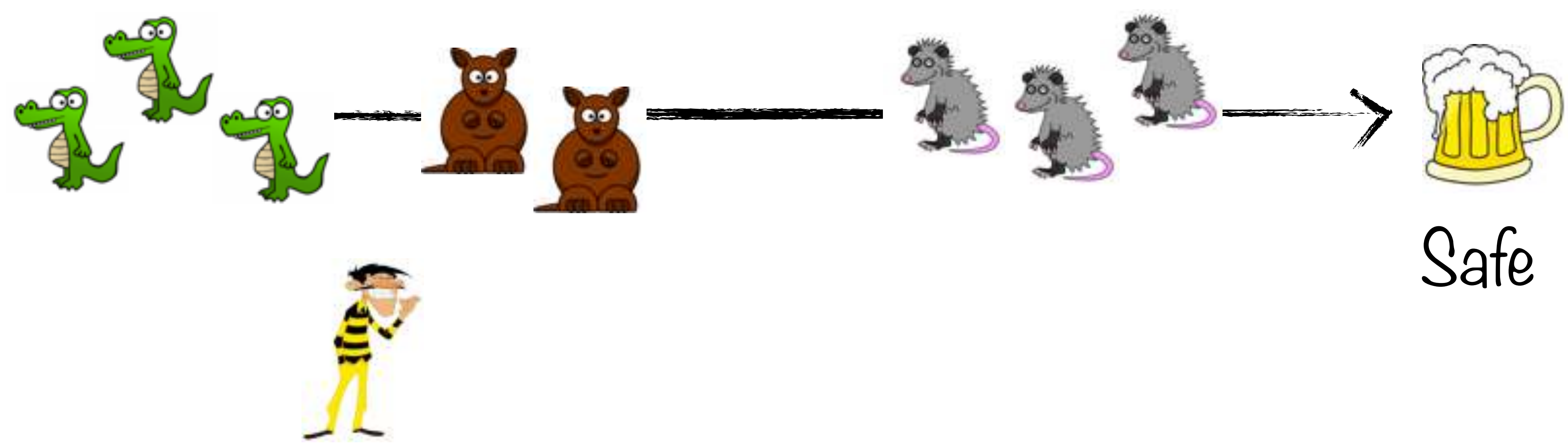


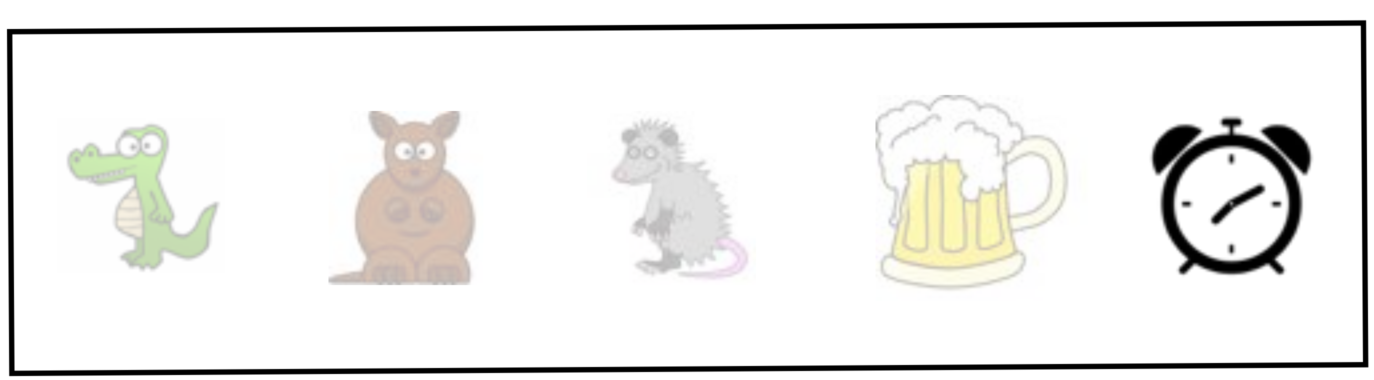


Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

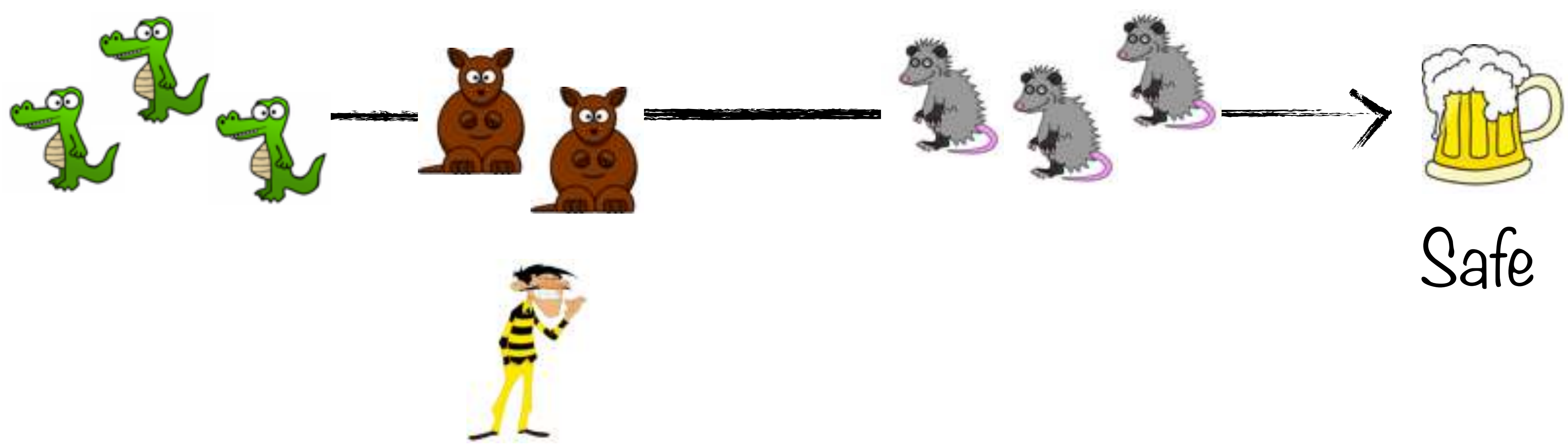


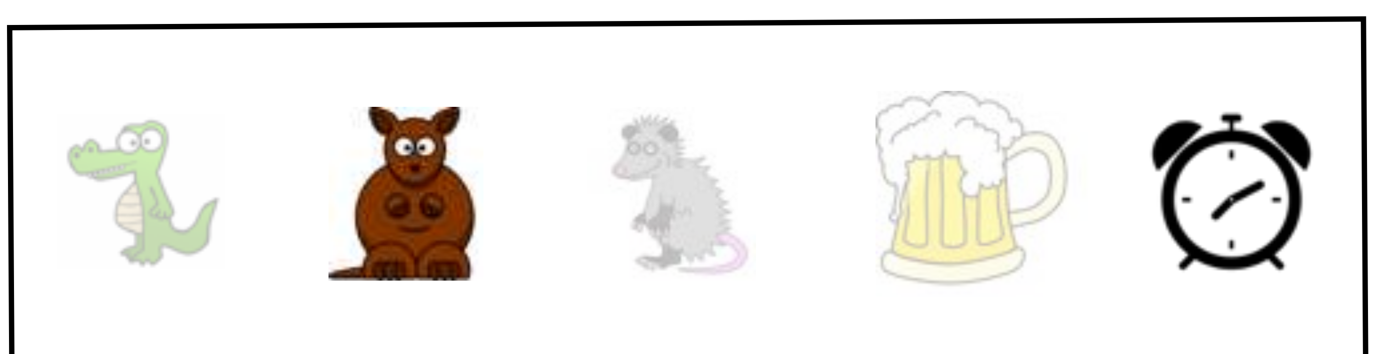


Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

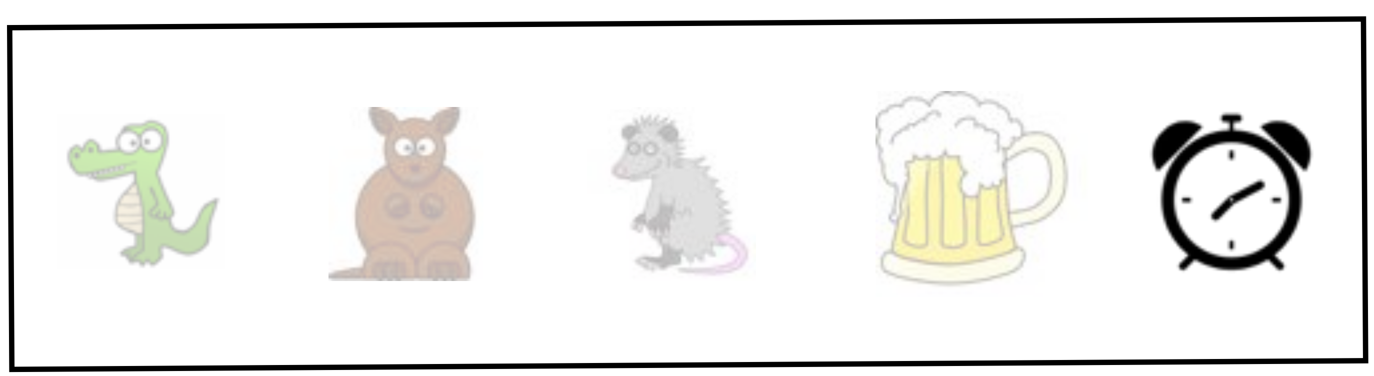




Symbolic Simulation

Our proposal: keep a notion of time elapsing

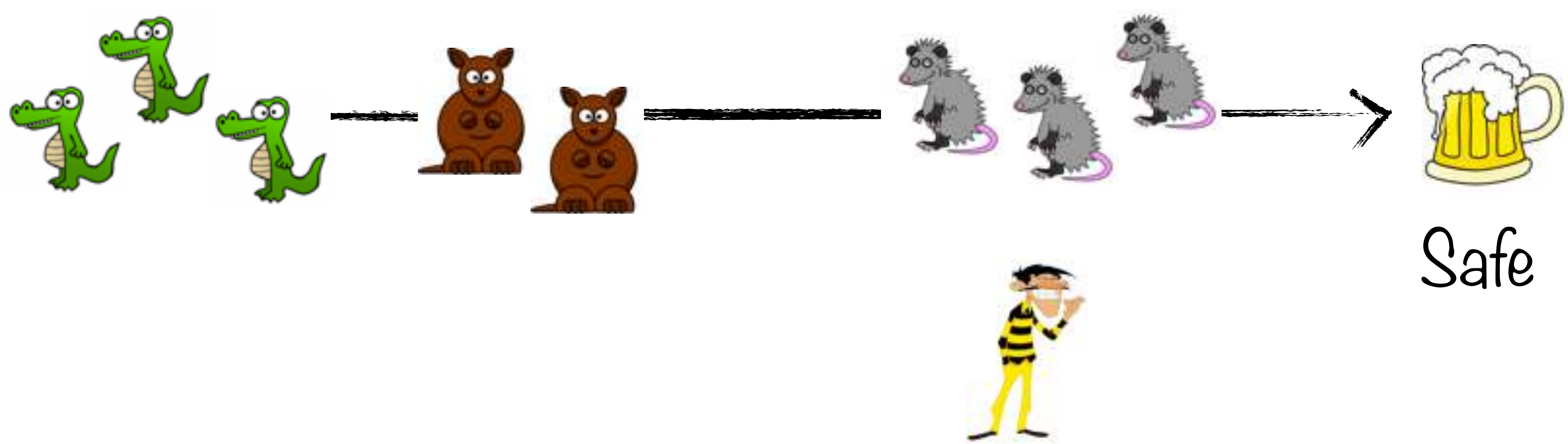
Australian Walk



Symbolic Simulation

Our proposal: keep a notion of time elapsing

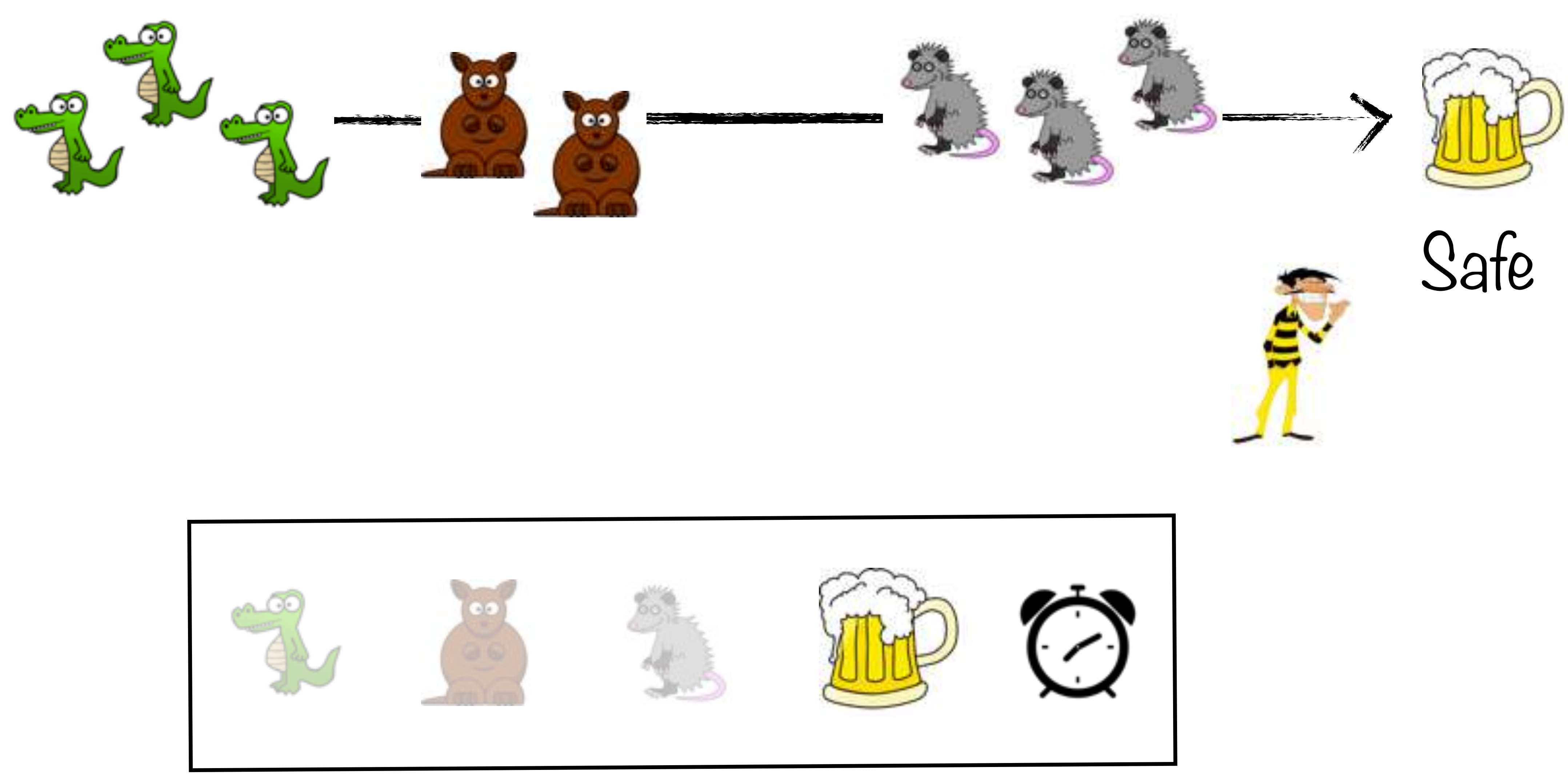
Australian Walk



Symbolic Simulation

Our proposal: keep a notion of time elapsing

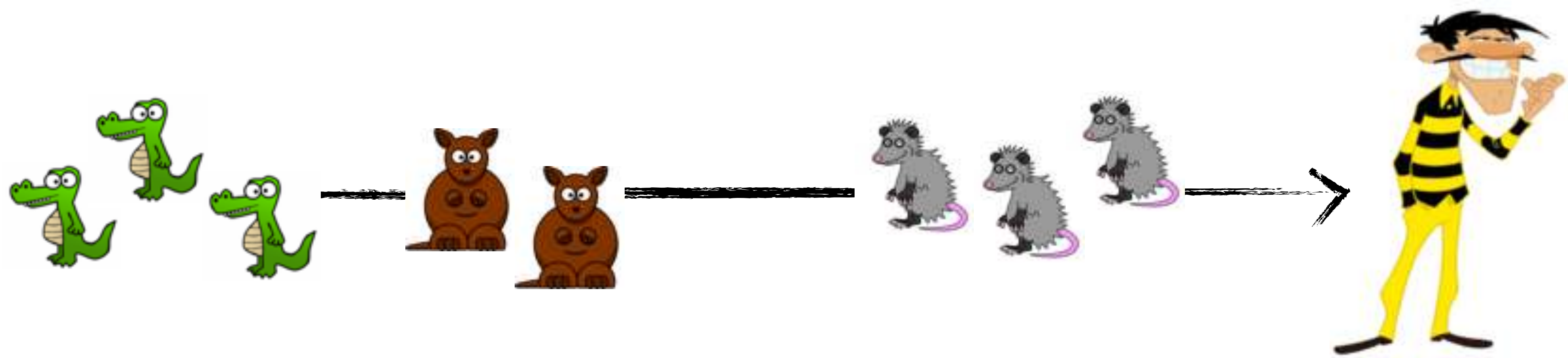
Australian Walk

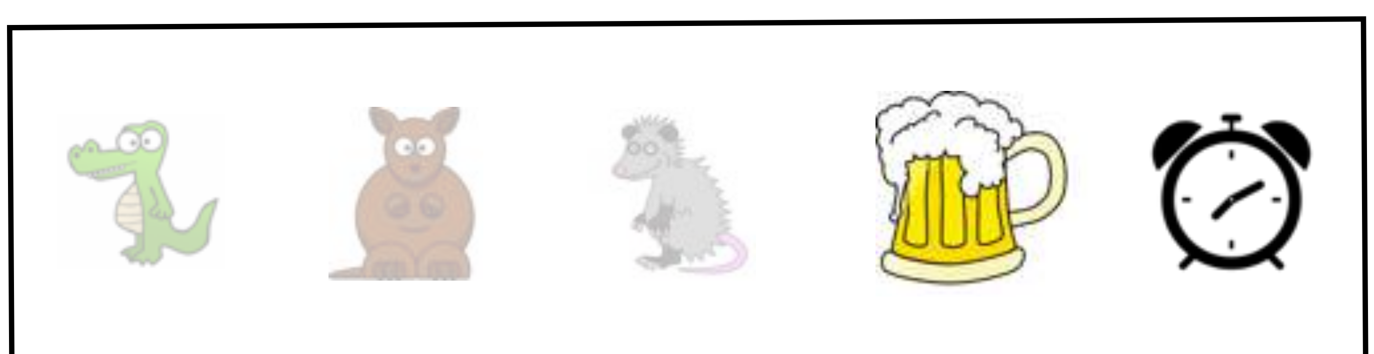


Symbolic Simulation

Our proposal: keep a notion of time elapsing

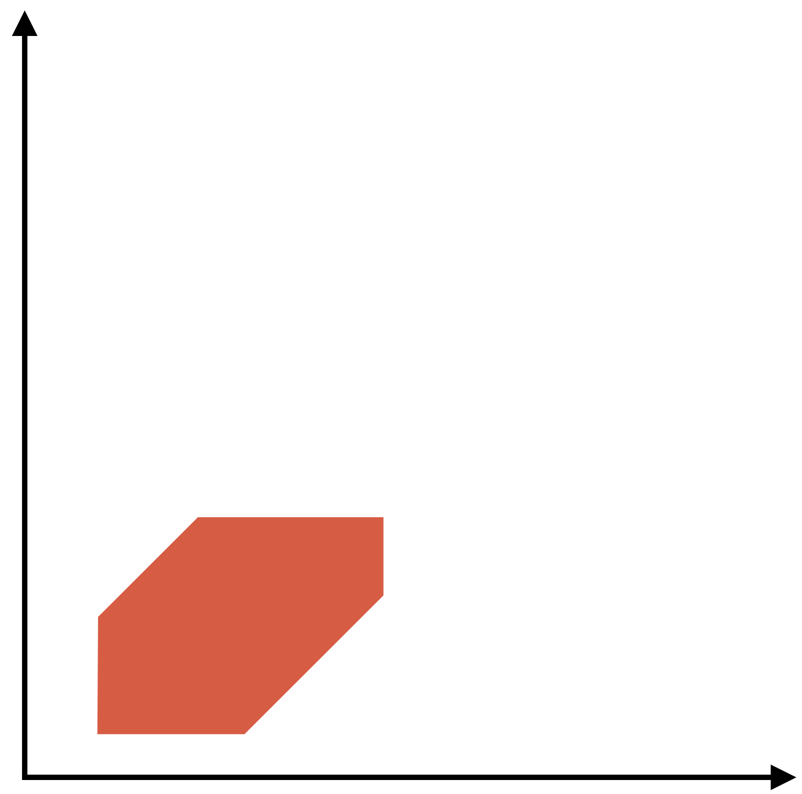
Australian Walk





Time Elapse… A New Horizon

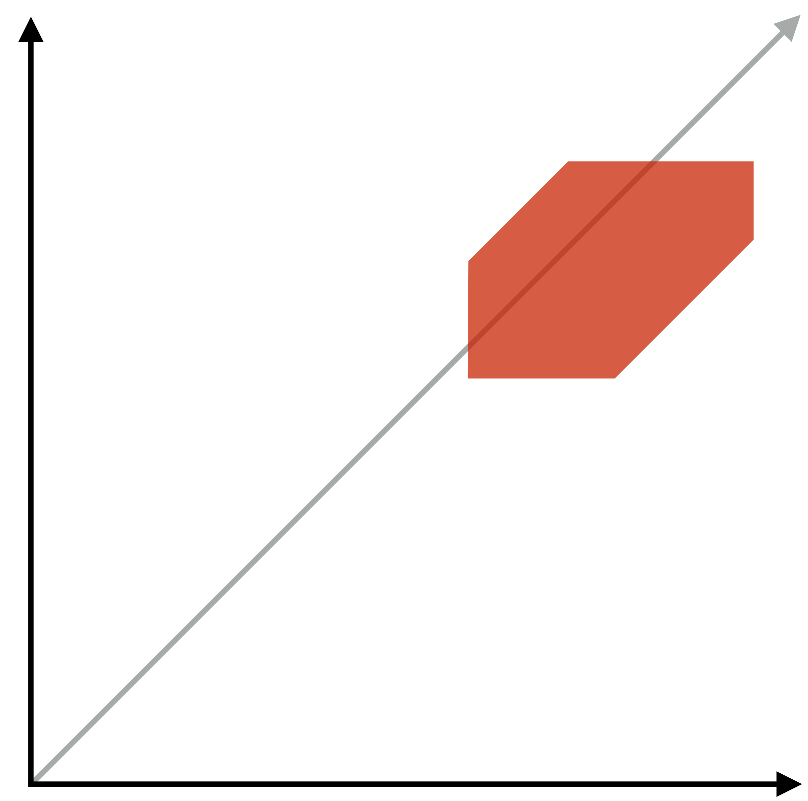
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



[Halbwachs et al. 1994]

Time Elapse… A New Horizon

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

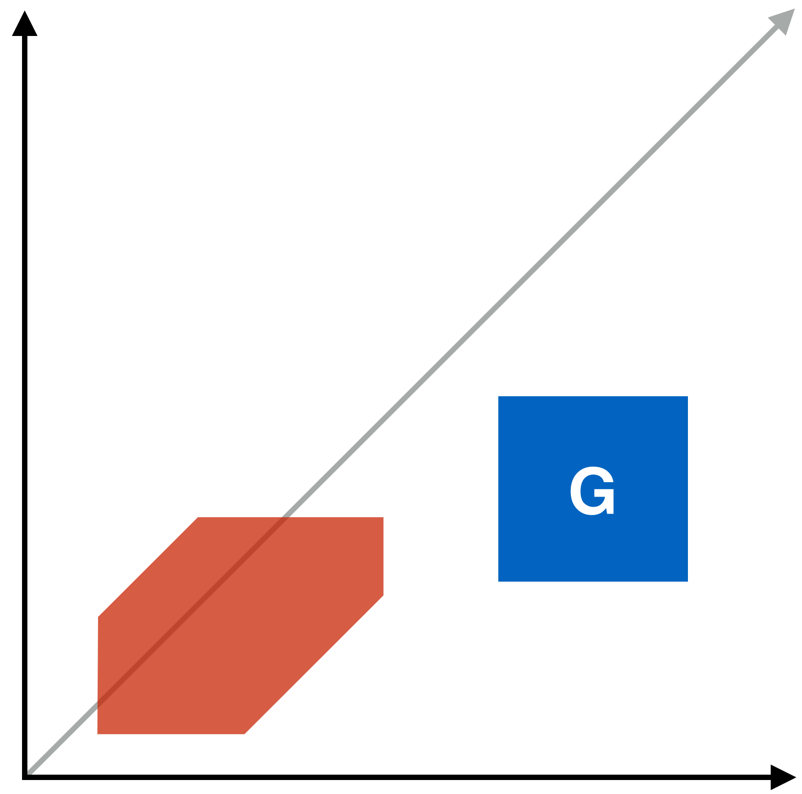


Increment all clocks at the same time: slide along direction (1, 1, …, 1)

[Halbwachs et al. 1994]

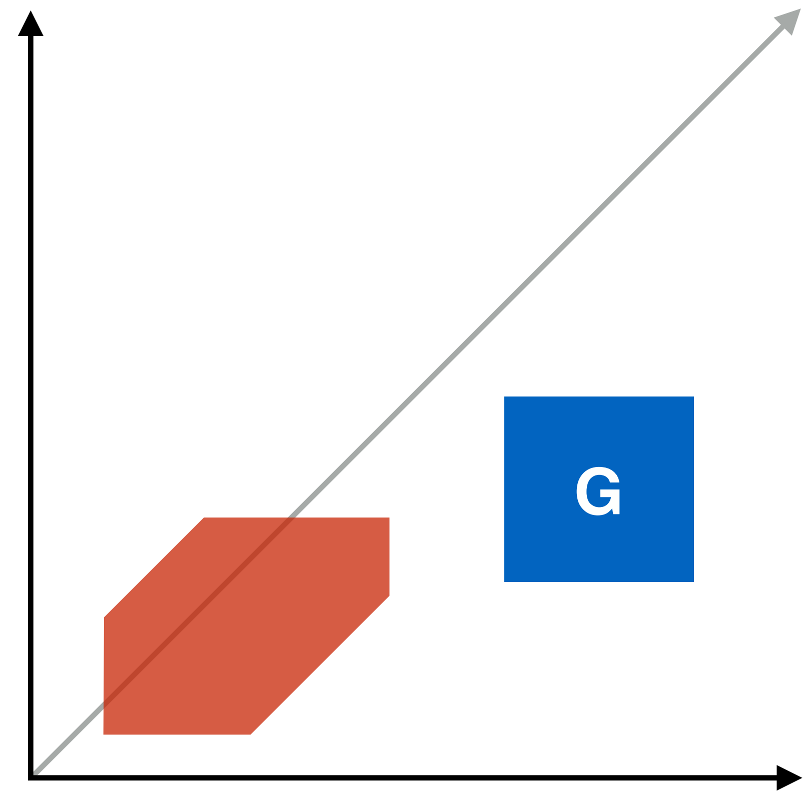
Time Elapse… A New Horizon

Definition: Two simulation states (q,s) and (q’,s’) are equivalent if q = q’ and actions(s) = actions(s’)



Time Elapse… A New Horizon

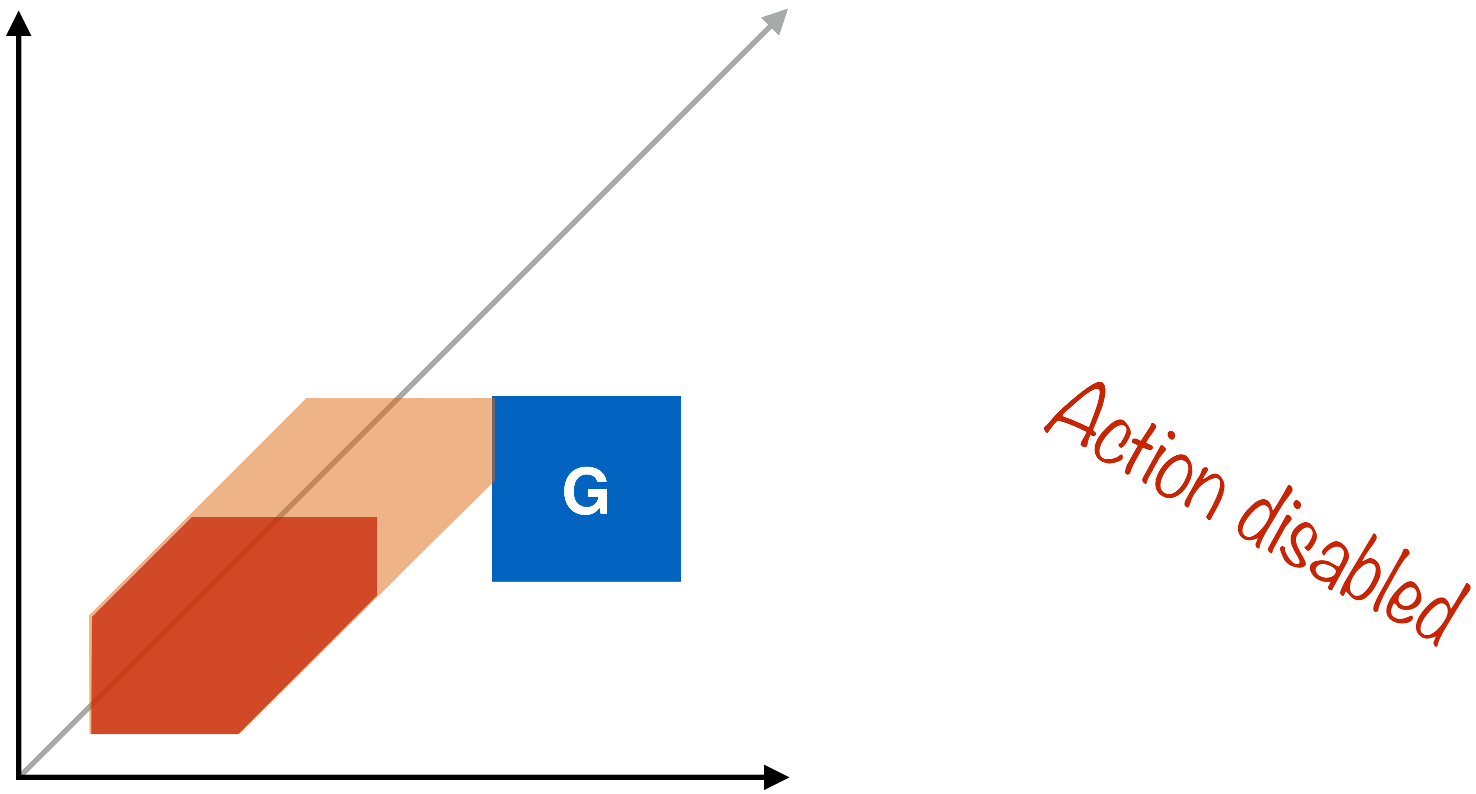
Definition: Two simulation states (q,s) and (q’,s’) are equivalent if q = q’ and actions(s) = actions(s’)



For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

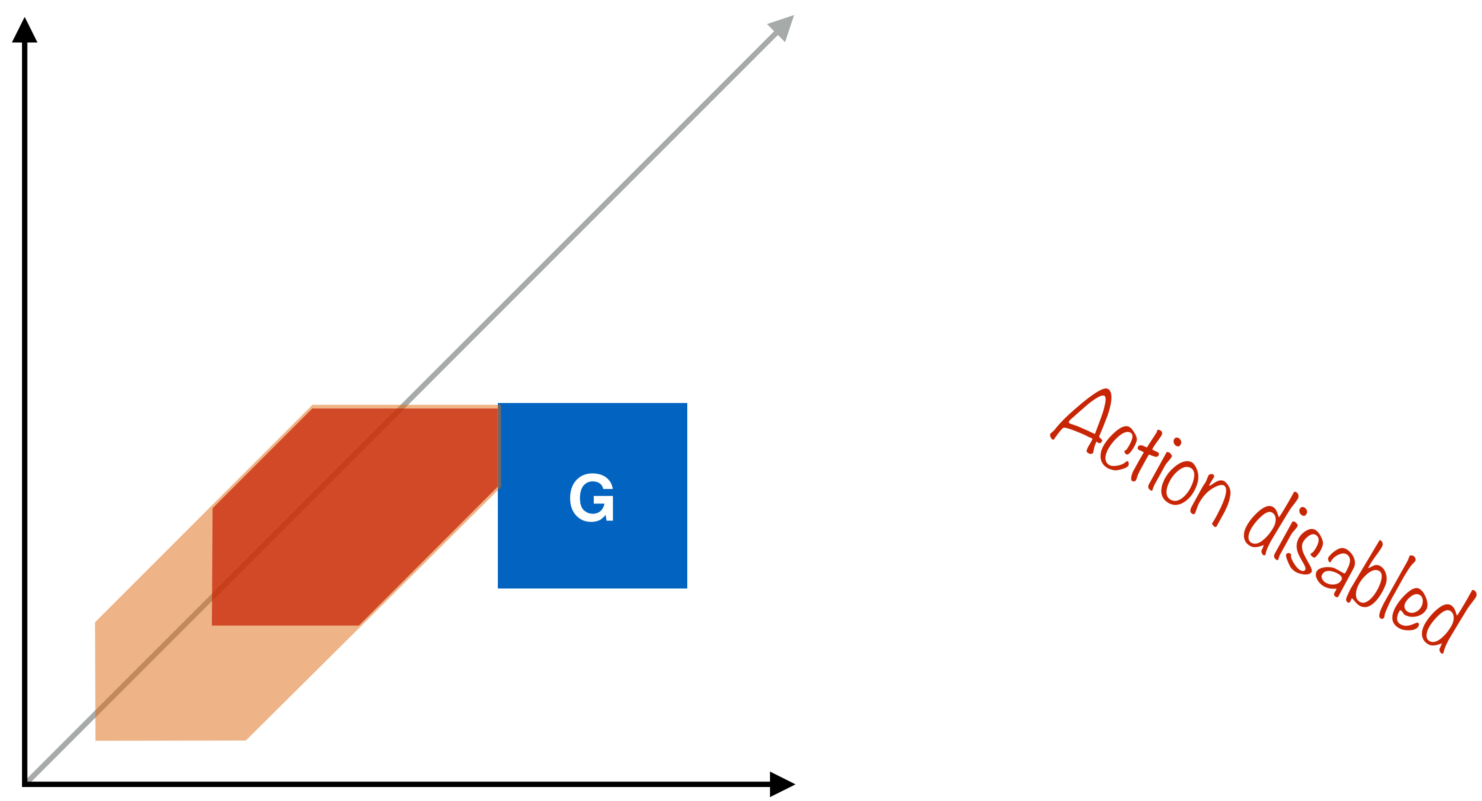
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Time Elapse… A New Horizon

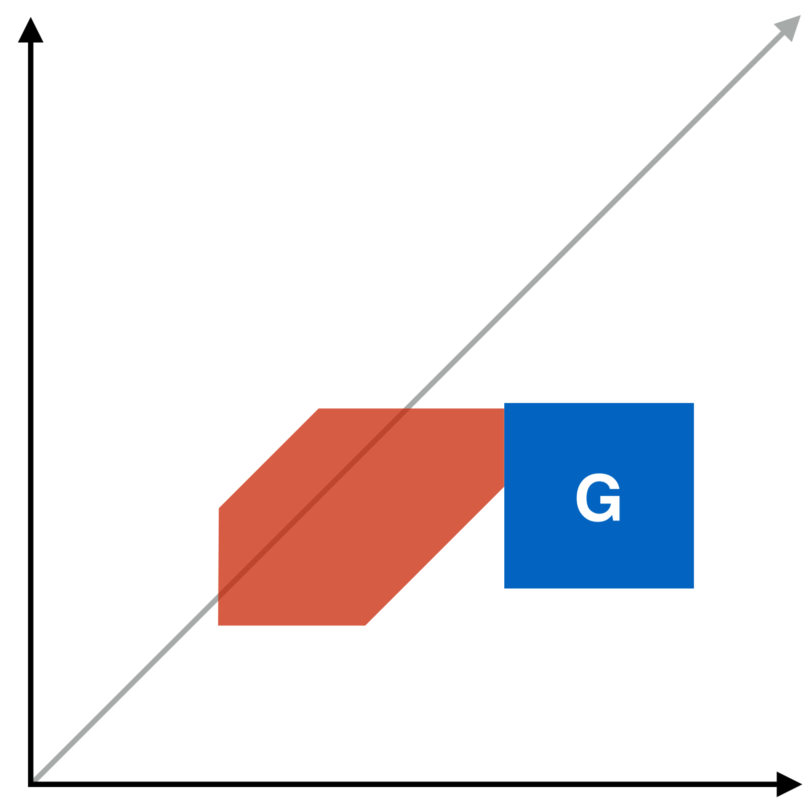
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

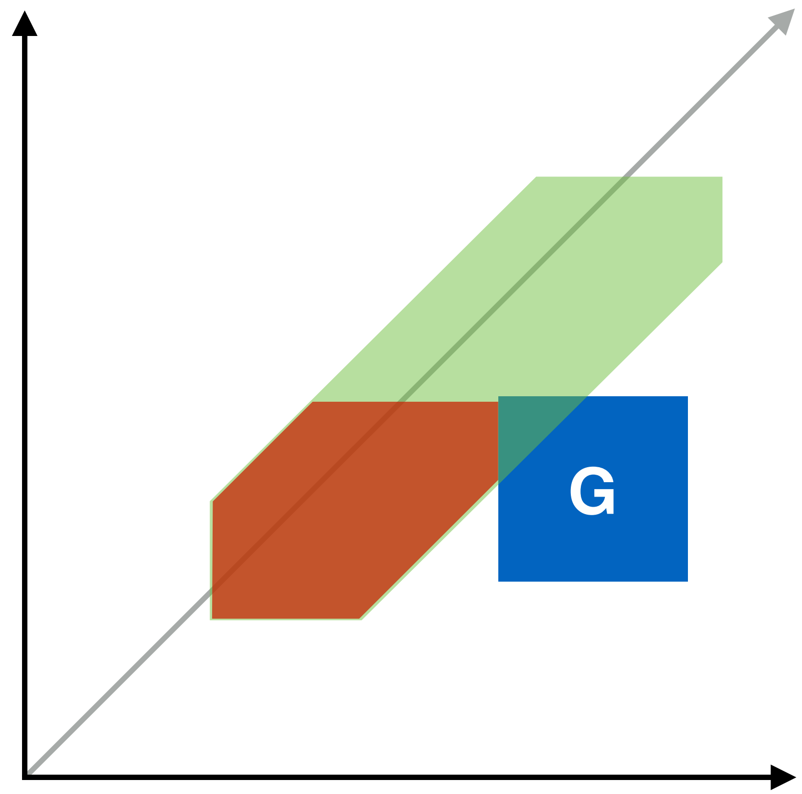


Action disabled

For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')

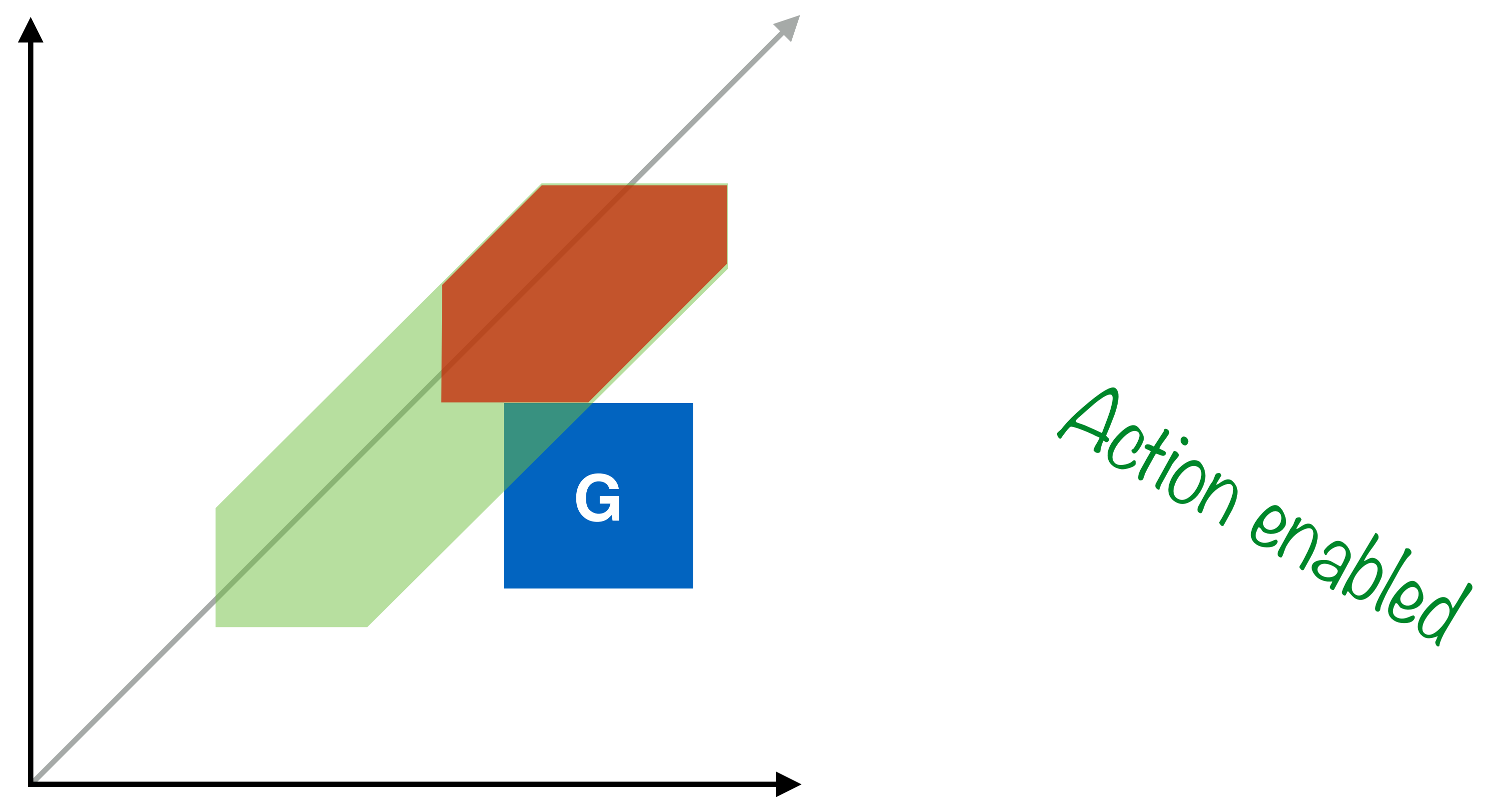


Action enabled

For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

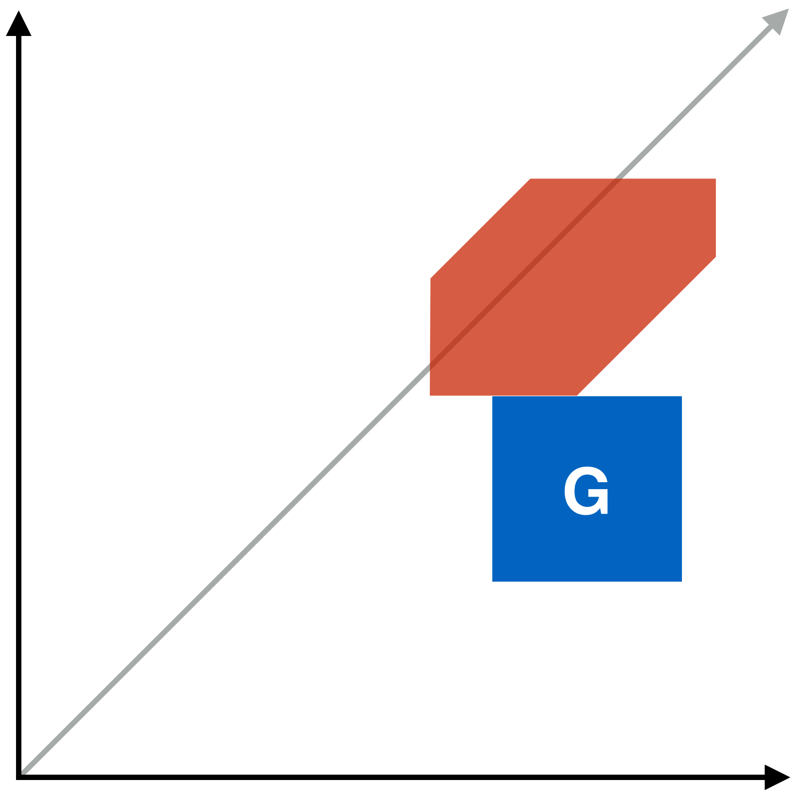
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

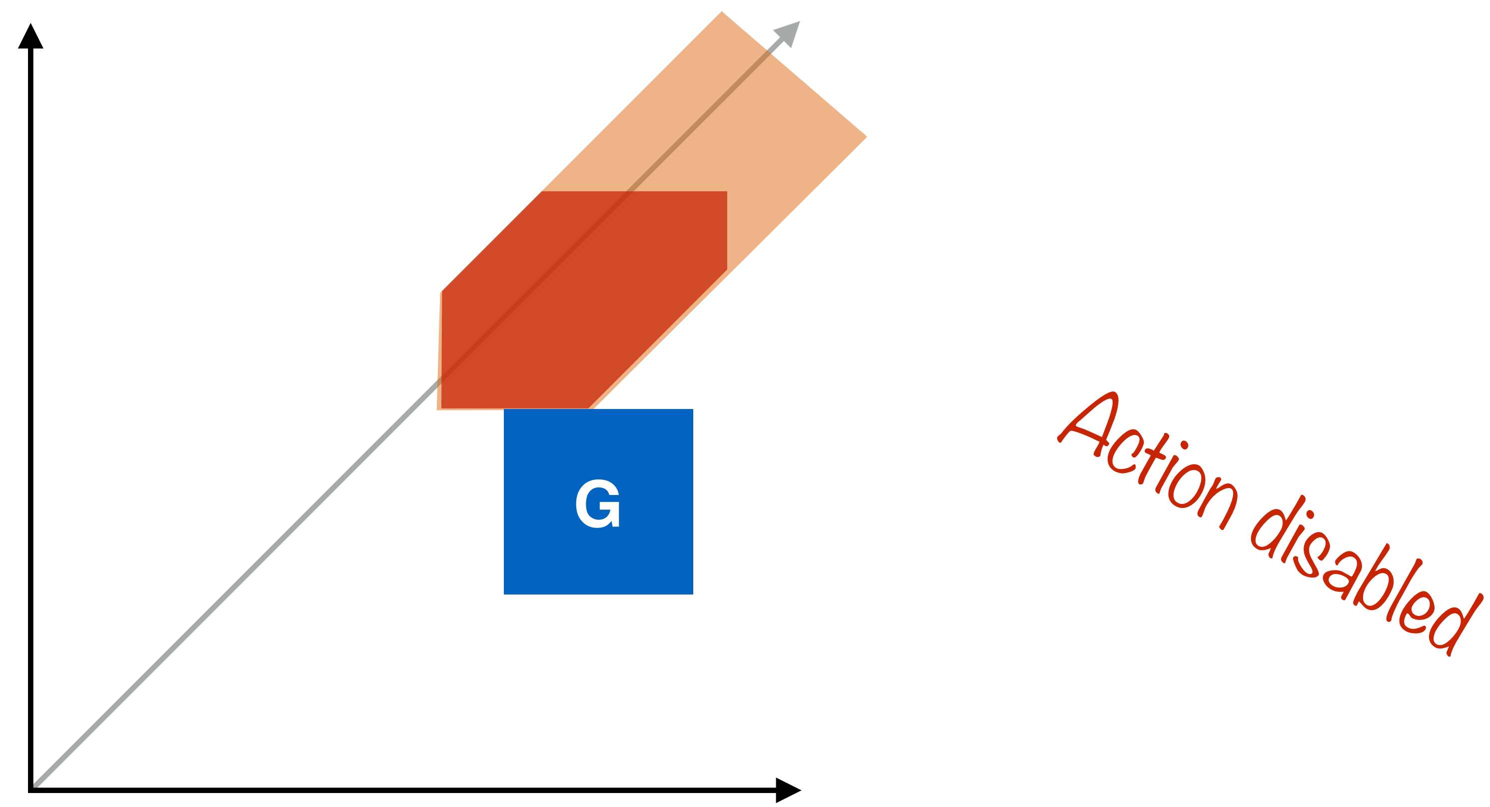
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Time Elapse... A New Horizon

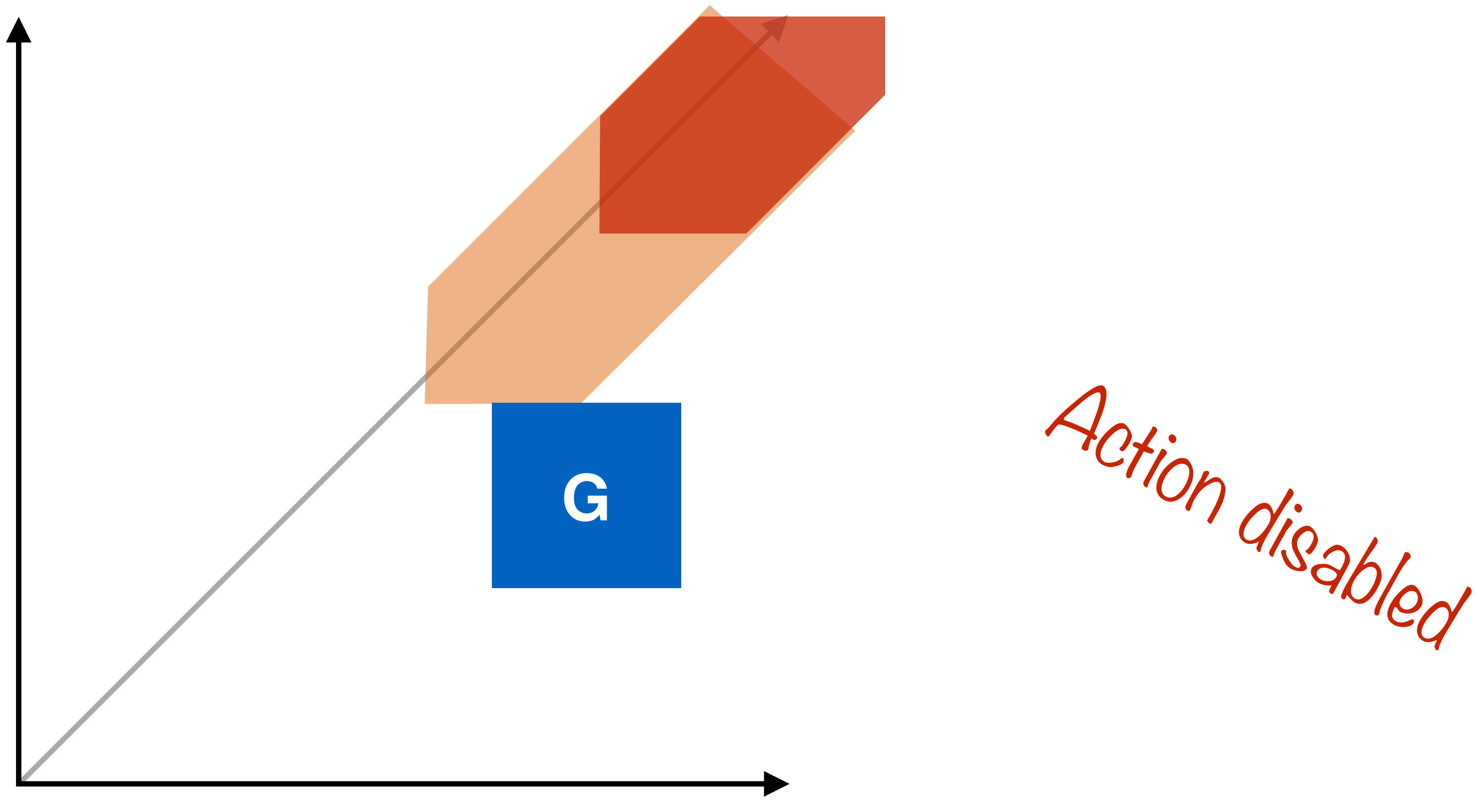
Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Time Elapse… A New Horizon

Definition: Two simulation states (q,s) and (q',s') are equivalent if q = q' and actions(s) = actions(s')



For each action, two horizons: entering and leaving

Language Restrictions

A subset of Zélus

Clocks: `der x = 1.0`

Clock constraints: $(x - y) \bowtie e$

with $\bowtie \in \{\leq, <, >, \geq\}$ and `e: float`

Operations on clocks:

- Reset: `x = v`
- Translation: `x = x + v`
- Synchronization: `x = y`

Language Restrictions

A subset of Zélus

Difference Bound Matrices (DBM)

Clocks: `der x = 1.0`

Clock constraints: `(x - y)` $\bowtie$ `e`

with $\bowtie\ \in \{\leq, <, >, \geq\}$ and `e: float`

Operations on clocks:

- Reset: `x = v`
- Translation: `x = x + v`
- Synchronization: `x = y`

[Dill 1990]

Compilation

Modularity: Each block returns the guard of the enabled transitions.

Global State: A DBM represents clock constraints of the entire system (current clock domain).

Simulation: At each step, we return the next horizon and compute the next clock domain.

Compilation

Modularity: Each block returns the guard of the enabled transitions.

Global State: A DBM represents clock constraints of the entire system (current clock domain).

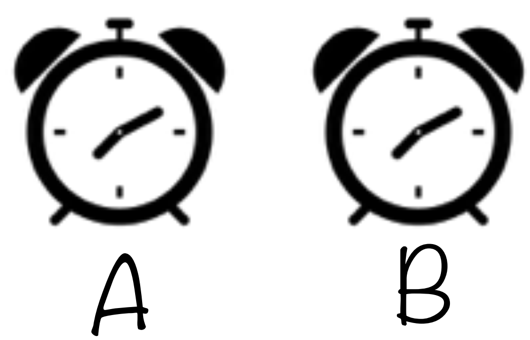
Simulation: At each step, we return the next horizon and compute the next clock domain.

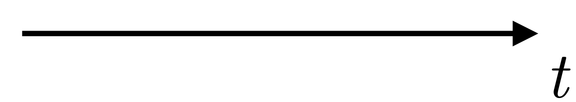
A way to discretize continuous systems

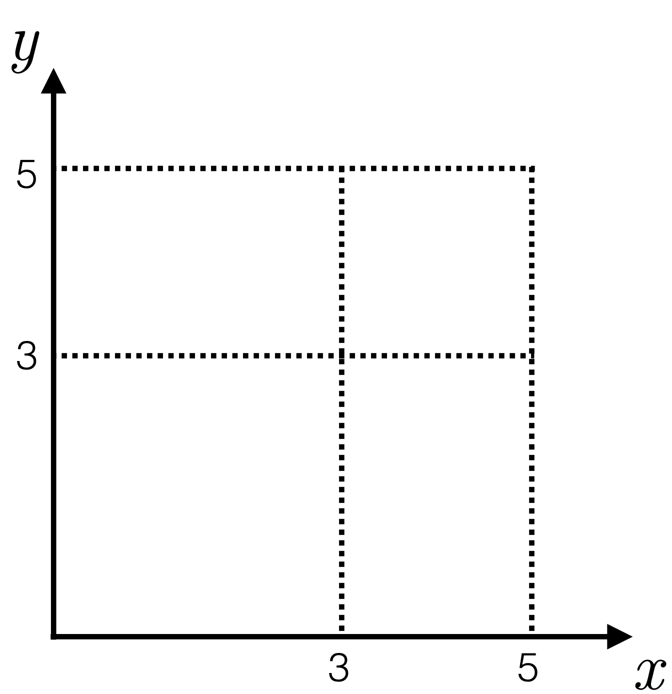
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







in_x

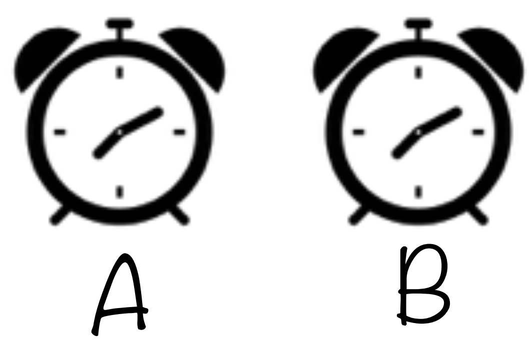
in_y

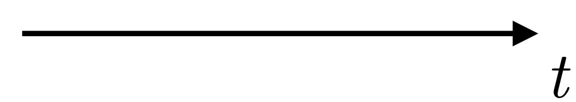
wait

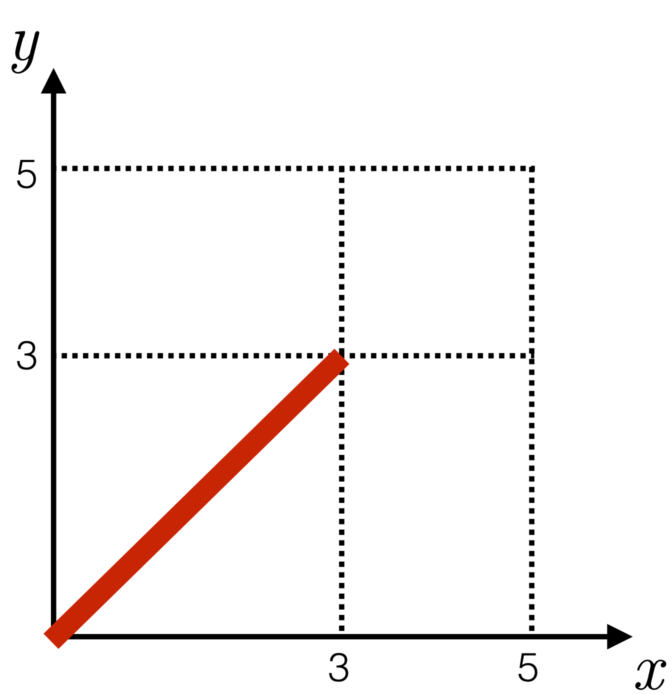
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

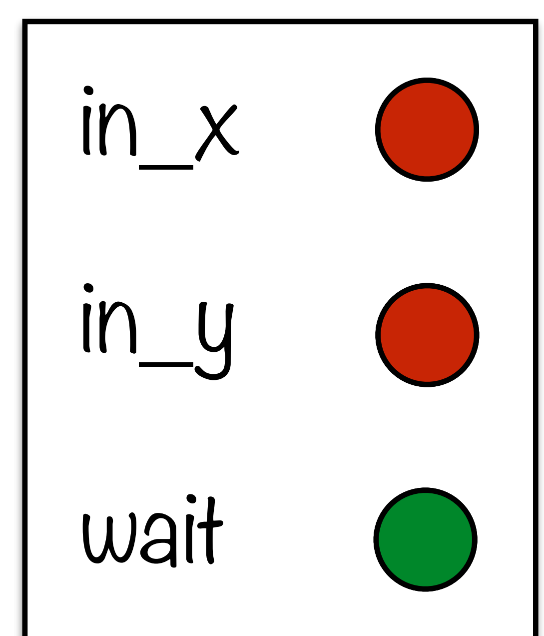
```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







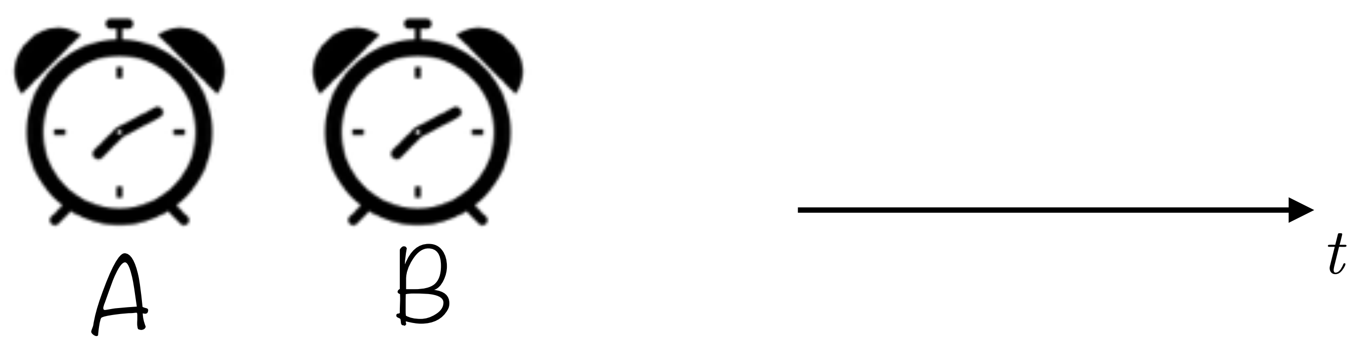
$$\left\{\begin{array}{l} 0 \leq x < 3 \\ 0 \leq y < 3 \\ x - y = 0 \end{array}\right\}$$

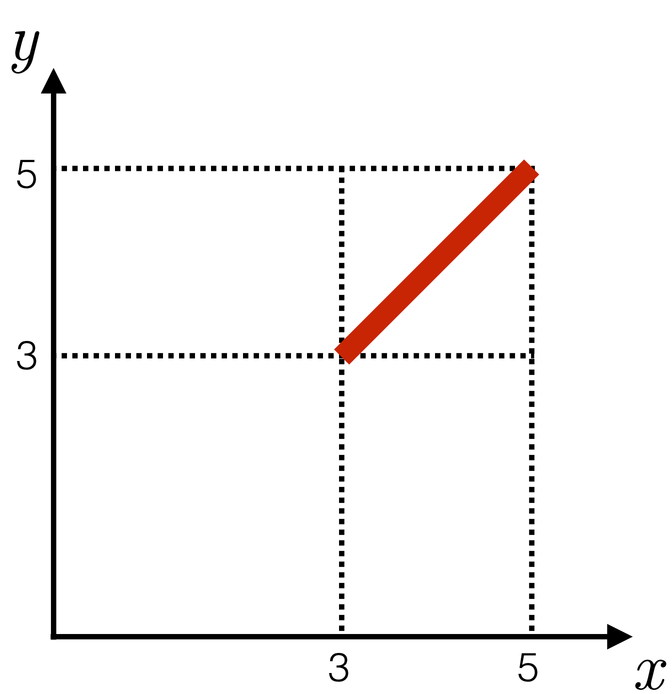


Example: Quasi-Periodic Architectures

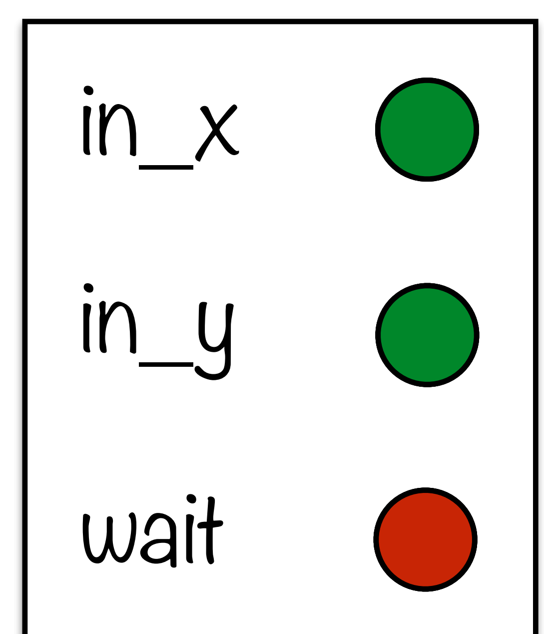
Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```





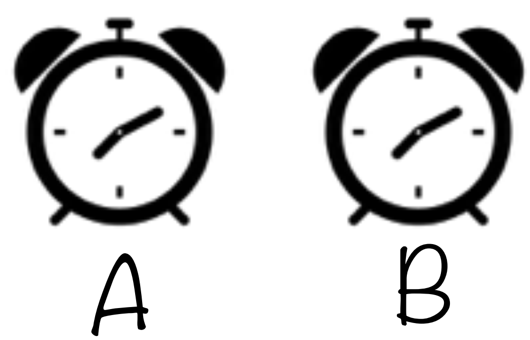
$$\begin{Bmatrix} 3 \leq x \leq 5 \\ 3 \leq y \leq 5 \\ x - y = 0 \end{Bmatrix}$$

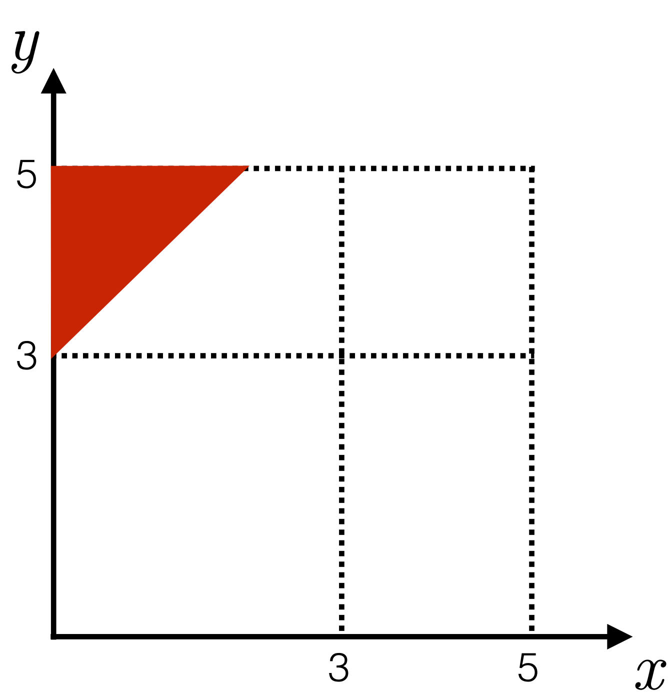


Example: Quasi-Periodic Architectures

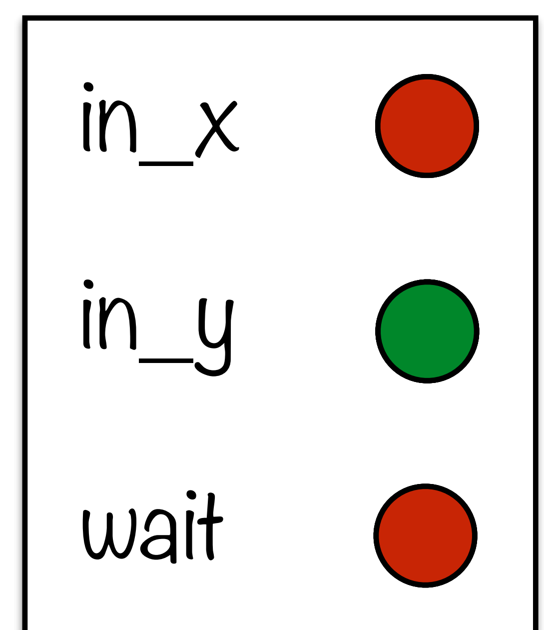
Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```





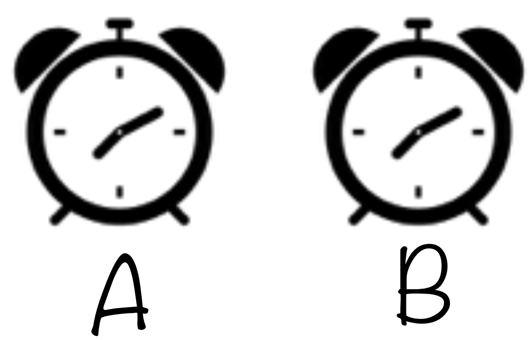
$$\left\{ \begin{array}{c} 0 \leq x < 2 \\ 3 \leq y \leq 5 \\ 3 \leq y - x \leq 5 \end{array} \right\}$$

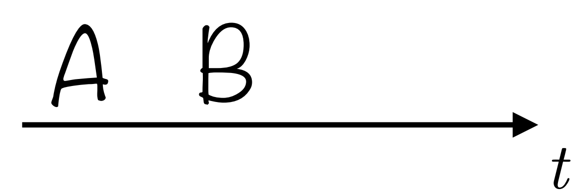


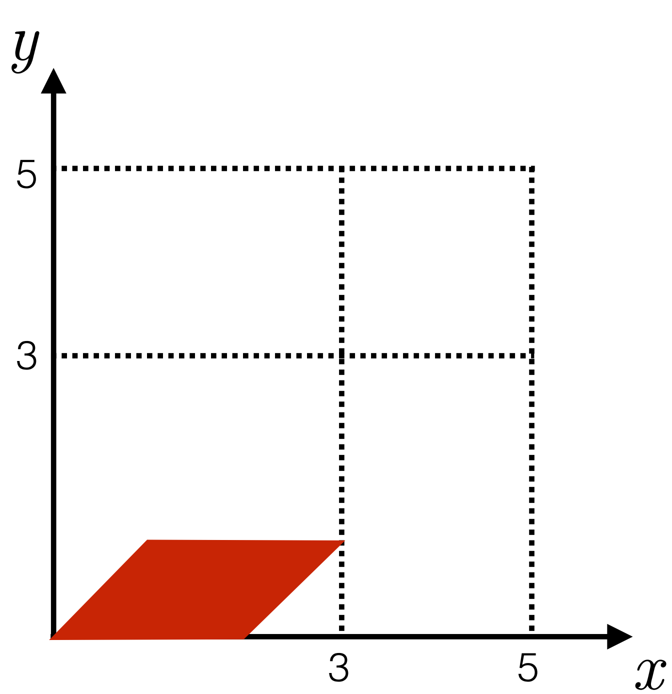
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

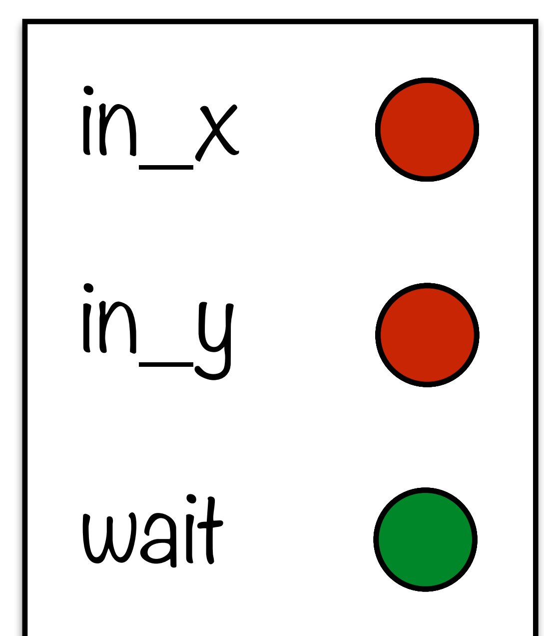
```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







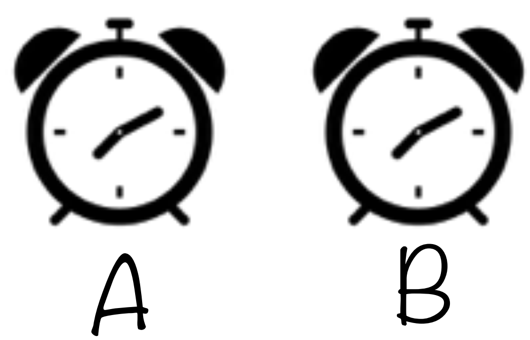
$$\left\{ \begin{array}{c} 0 \leq x < 3 \\ 0 \leq y \leq 1 \\ 0 \leq x - y \leq 2 \end{array} \right\}$$

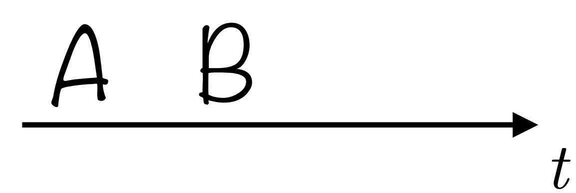


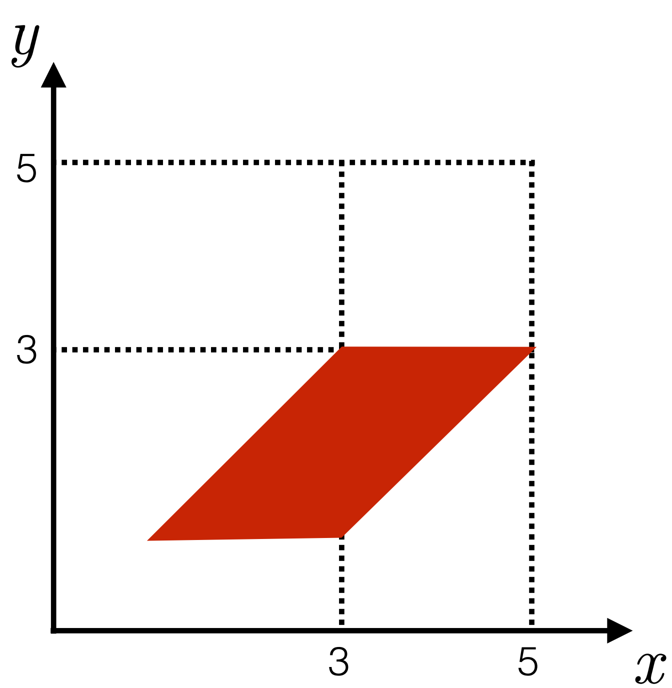
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

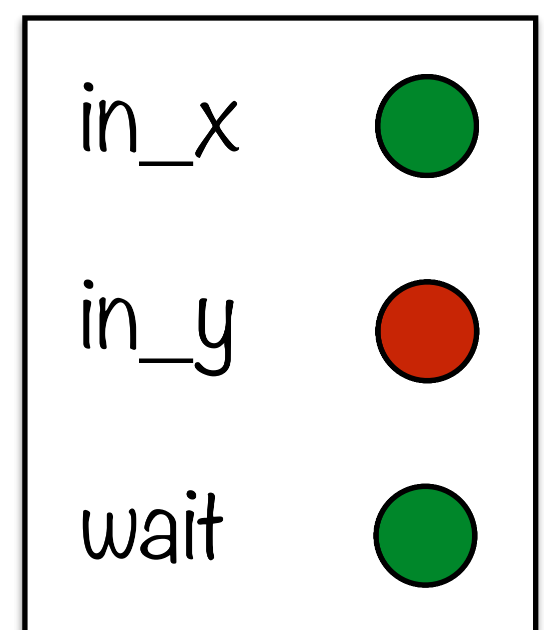
```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







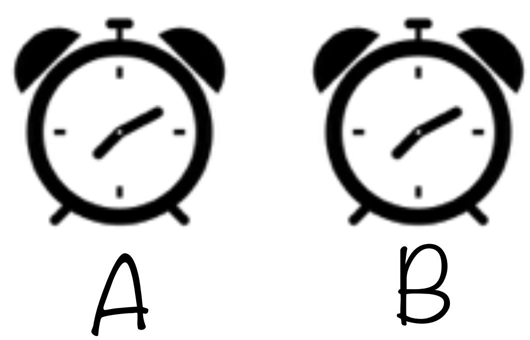
$$\left\{\begin{array}{c} 1 \leq x \leq 5 \\ 1 \leq y < 3 \\ 0 \leq x - y \leq 2 \end{array}\right\}$$

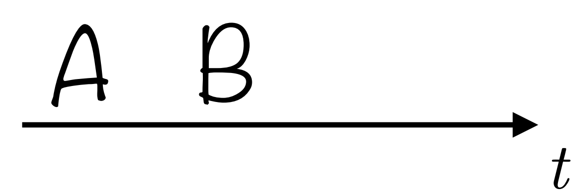


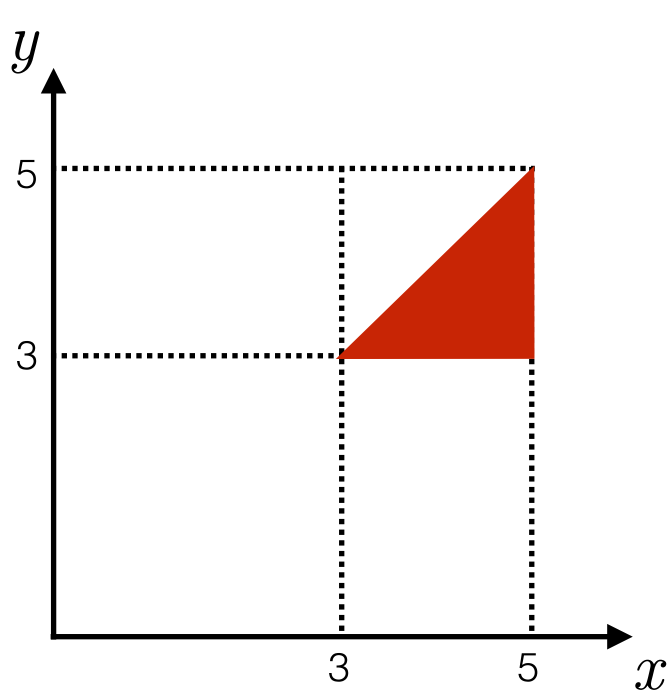
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

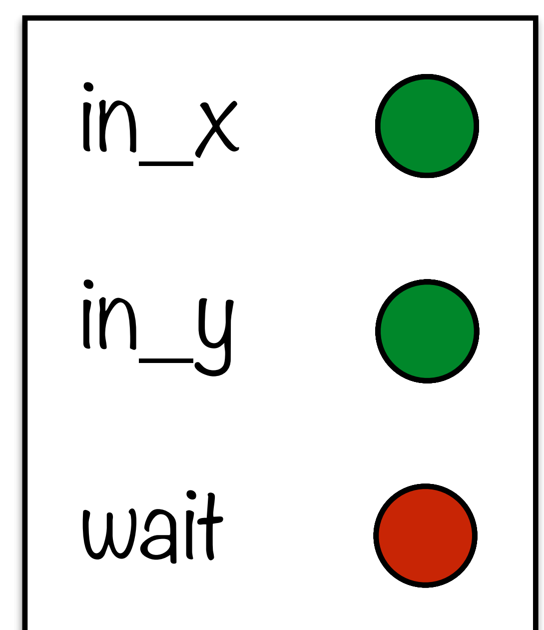
```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







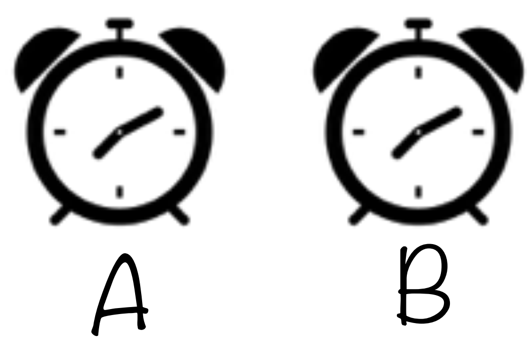
$$\left\{\begin{array}{c} 3 \leq x \leq 5 \\ 3 \leq y \leq 5 \\ 0 \leq x - y \leq 2 \end{array}\right\}$$

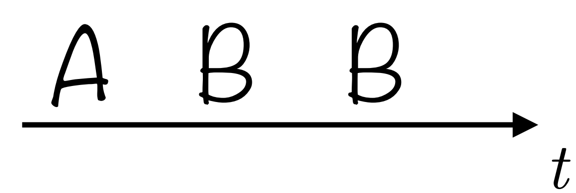


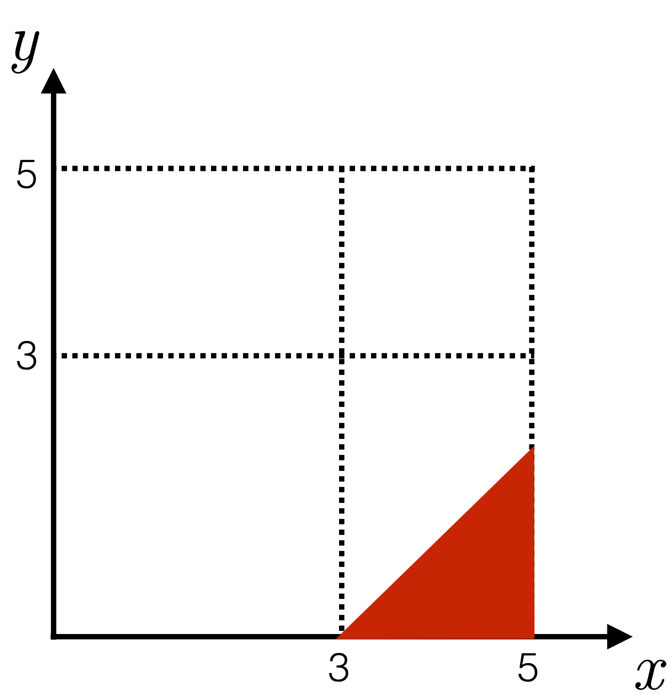
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

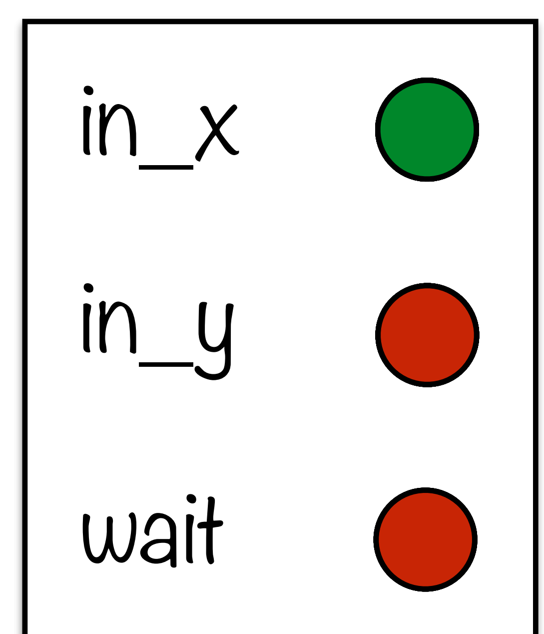
```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```







$$\left\{ \begin{array}{c} 3 \leq x \leq 5 \\ 0 \leq y \leq 2 \\ 3 \leq x - y \leq 5 \end{array} \right\}$$

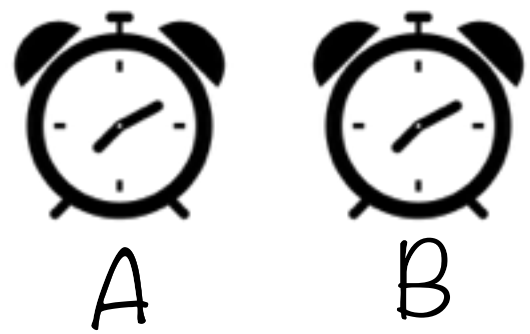


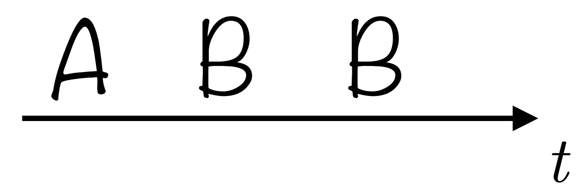
Example: Quasi-Periodic Architectures

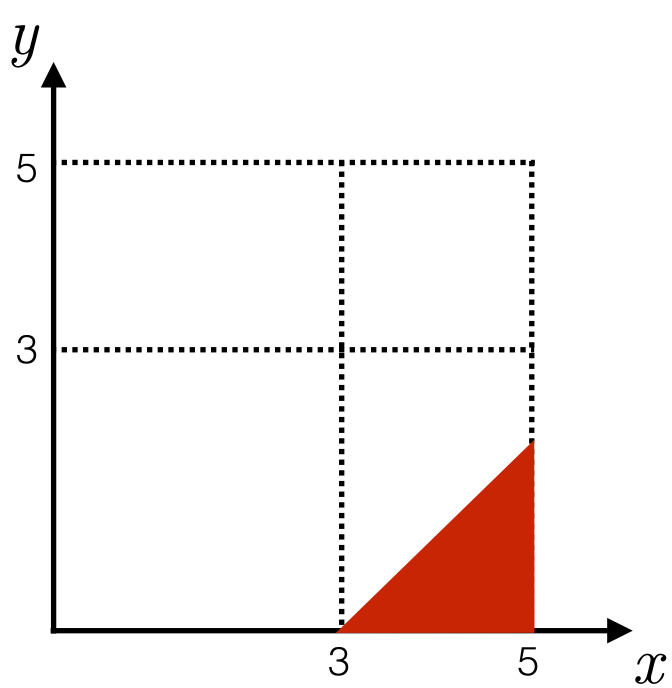
Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```

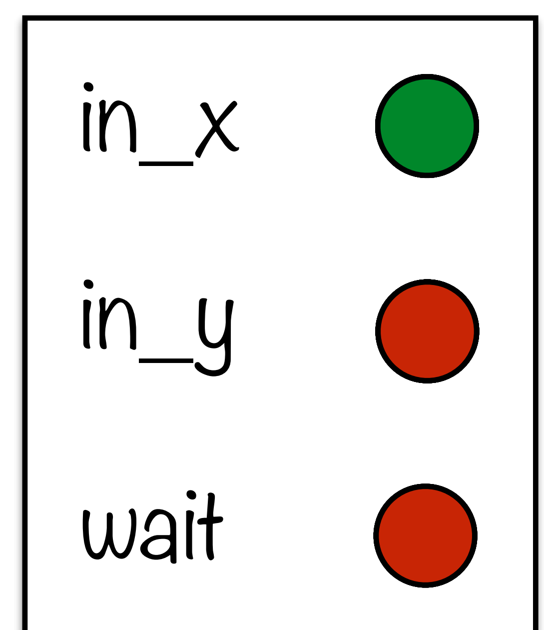
No more than two ticks of one clock between two ticks of the other







$$\left\{ \begin{array}{c} 3 \leq x \leq 5 \\ 0 \leq y \leq 2 \\ 3 \leq x - y \leq 5 \end{array} \right\}$$



Future Work

Prototype implementation in zélus
Source to source transformation and runtime

More complex clock domains
octagon, polyhedron, …

Under-approximation / Over-approximation
safety vs. precision

Generate discrete controllers
for instance quasi-synchronous controllers

Improve test coverage
see [Alur et al 2008]