An Introduction to Hybrid Automata

Jean-François Raskin
U.L.B.

Plan of the talk

- Introduction - Motivations
- Hybrid automata: syntax and semantics
- Properties of hybrid automata
- Rectangular hybrid automata
- Semi-algorithms for rectangular hybrid automata
- Approximations of affine hybrid automata
  by rectangular hybrid automata
- Conclusion

Introduction
Motivations

Reactive and embedded systems

- Reactive systems are systems that maintain a continuous interaction with their environment.
- Reactive systems :
  - are non-terminating systems
  - have to respect or enforce real-time properties
  - have to cope with concurrency
  - are often embedded into an complex, continuous and safety critical, environments.


777
BOEING 777-300

300 horses power

100 processors

Windows

An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) + 00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) + 00000000. It may be possible to continue normally.

* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue


French Guniea, june 4, 1996

Reactive and embedded systems

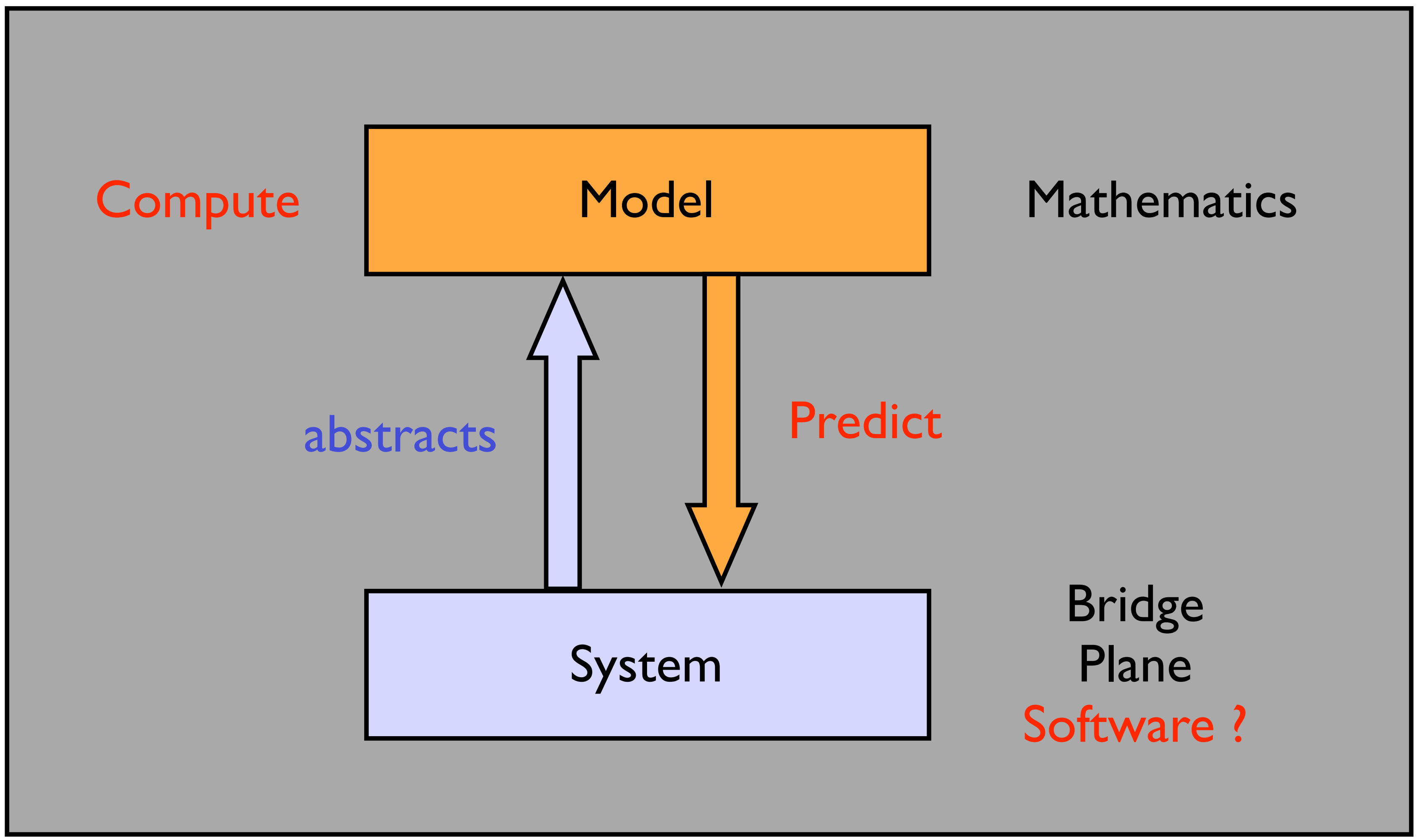
- The specification that have to meet ES are very complex (e.g. environment=continuous system, concurrency, real-time, ...)
- ES are difficult to test:
  - the environment in which they are embedded does not preexist/is difficult to simulate (e.g. rocket, medical equipment, ...);
  - even when errors are found, their diagnostic is diffult, we may not be able to "replay" the erroneous behavior.

Need for FM and verification

- ... they are difficult to develp correctly !
- ... they are often safety critical !

$\Rightarrow$ we should very them !

How to cope with complexity in sciences ?



Hybrid Automata

Mixing discrete-continuous evolutions

- Finite state automata have shown useful to model the control of reactive systems
- Reactive systems often have non-trivial interactions with continuous environment in which they are embedded
- We need a model which is able to model the discrete evolution of the controller and the continuous evolution of the environment
- Hybrid automata extend finite automata with continuous variables whose behaviors are described using differential constraints.

Our running example

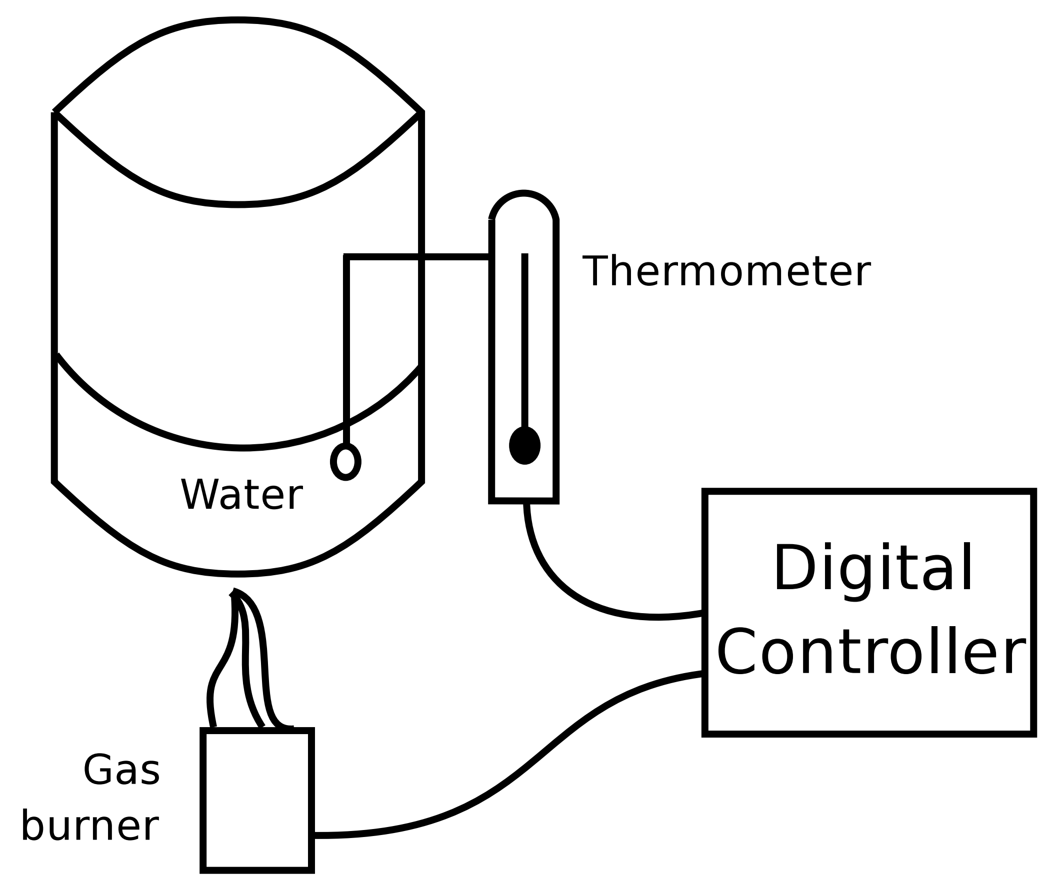


Fig. 1. Our running example

Our running example

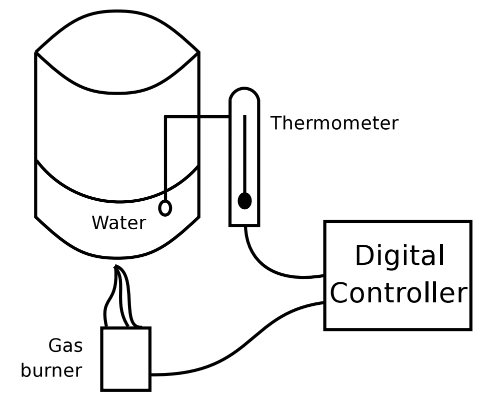


Fig. 1. Our running example

- Three environment components plus a controller:
  - A tank containing water;
  - A gas burner that can be turn on or off;
  - A digital thermometer that monitor the temperature within the tank.
- We want to design a controller that will maintain the temperature in the tank within an interval of safe temperatures.

Continuous part

- Behavior of the temperature in the tank
  - When the gas burner is OFF the temperature evolves according to
    $$\mathbf{x(t) = I\, e^{-Kt}}$$
    i.e. $\dot{x} = -Kx$
  - When the gas burner is ON the temperature evolves according to
    $$\mathbf{x(t) = I\, e^{-Kt} + h\,(1 - e^{-Kt})}$$
    i.e. $\dot{x} = K(h-x)$

    Where I is the initial temperature of the water, K is a constant that depends on the nature of the tank (how much it conducts heat for example), h is a constant that depends on the power of the gas burner, and t models time.

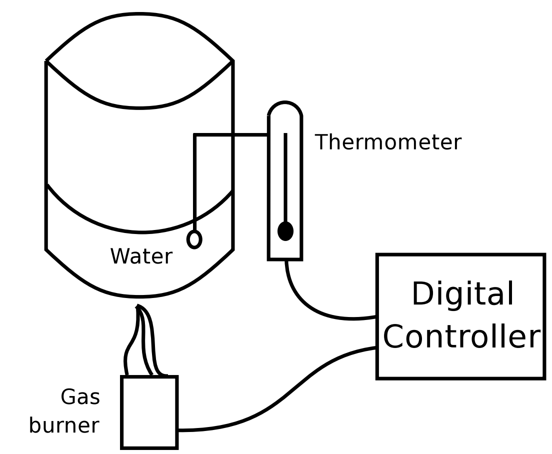


Fig. 1. Our running example

- We will refer to ON and OFF as modes of the tank evolution.
- Note that those rules are valid only when the temperature of the water is less than 100° celcius.

Fragment of the evolution of the temperature

Mode changes

Continuous Evolutions

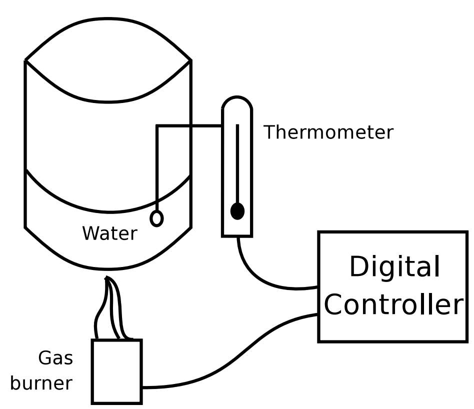


Fig. 1. Our running example

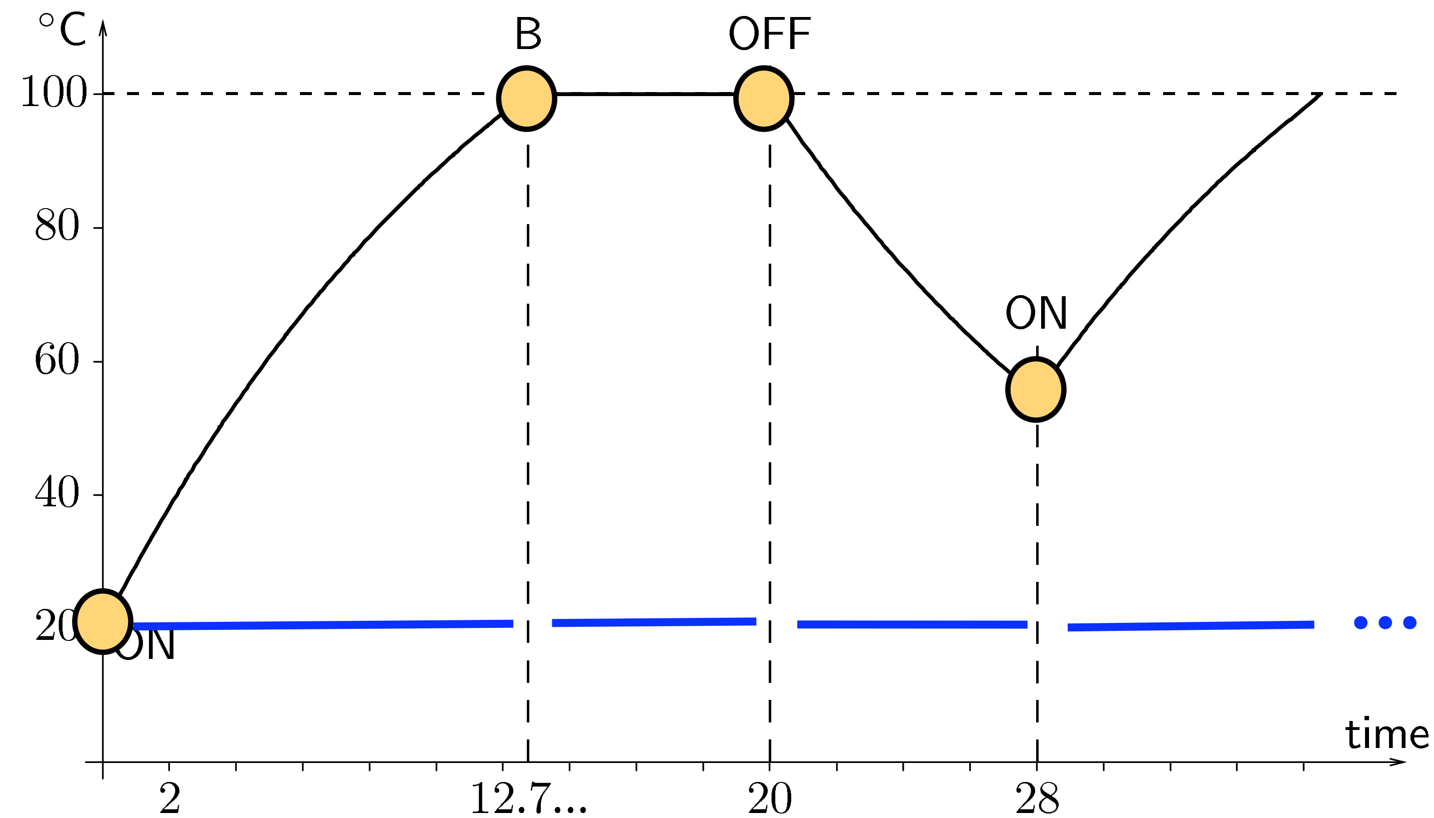


Fig. 2. One possible behavior of the tank

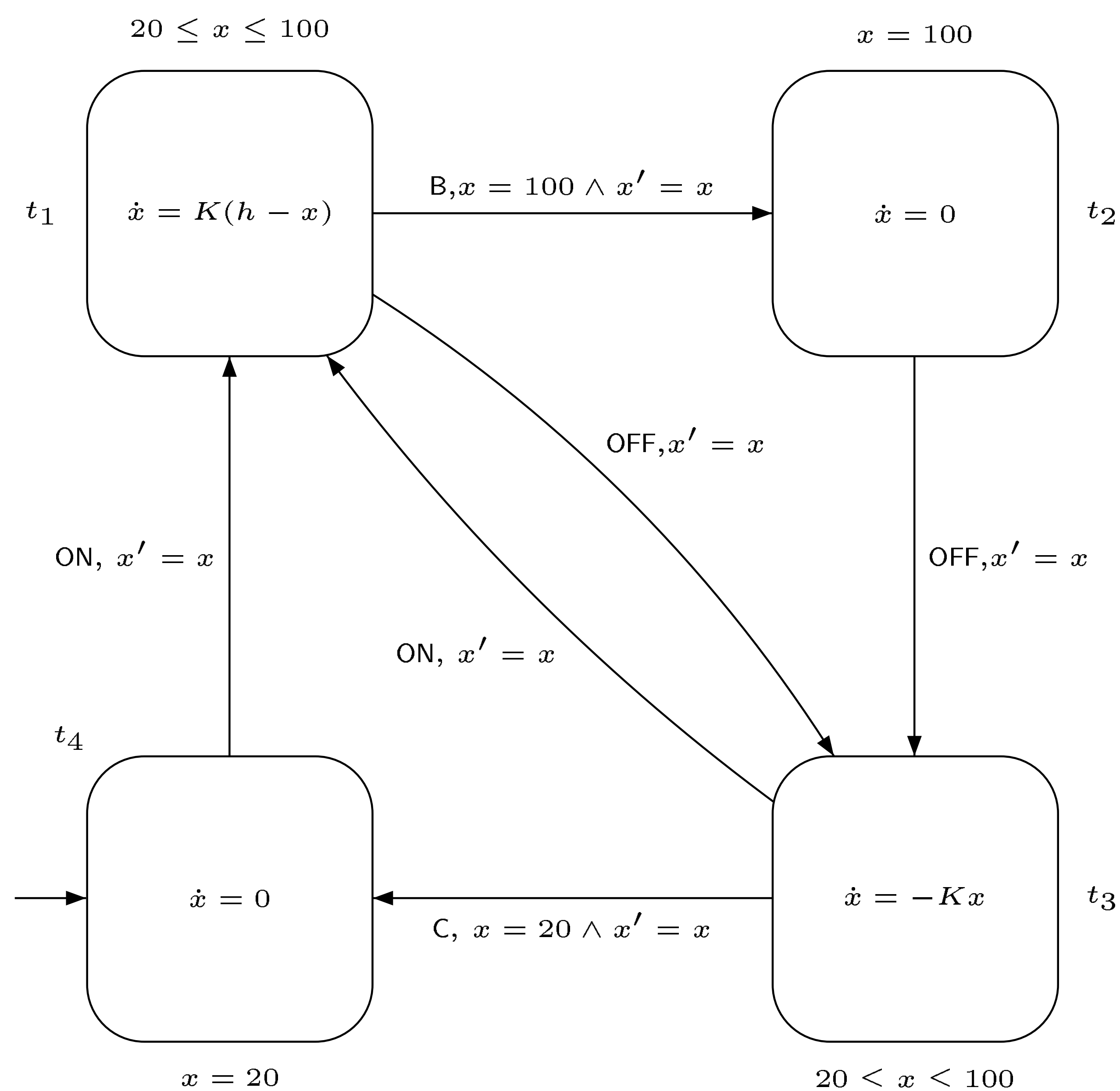
Clearly the evolution of the temperature is not purely continuous. It depends on the mode ON and OFF for example, and on the fact that the temperature is below 100° Celcius or not.

HA syntax and semantics

Hybrid automata - Syntax

- H=(Loc,Σ,Edge,X,Init,Inv,Flow,Jump), where:
  - Loc is a finite set $\{l_1,l_2,\ldots,l_n\}$ of (control locations) modeling control modes of the automaton;
  - Σ is a finite set of event names;
  - Edge ⊆ Loc × Σ × Loc is a finite set of labelled edges that represent discrete changes between control modes;
  - X is a finite set $\{x_1,x_2,\ldots,x_m\}$ of real-valued variables.
    We write $\dot{X}=\{\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_m\}$ for the associated dotted variables and $X'= \{x'_1,x'_2,\ldots,x'_m\}$ for the associated primed variables.
  - Init, Inv, and Flow are functions that assign three predicates to each location.
  - Jump is a function that assigns a predicate to each labelled edge.

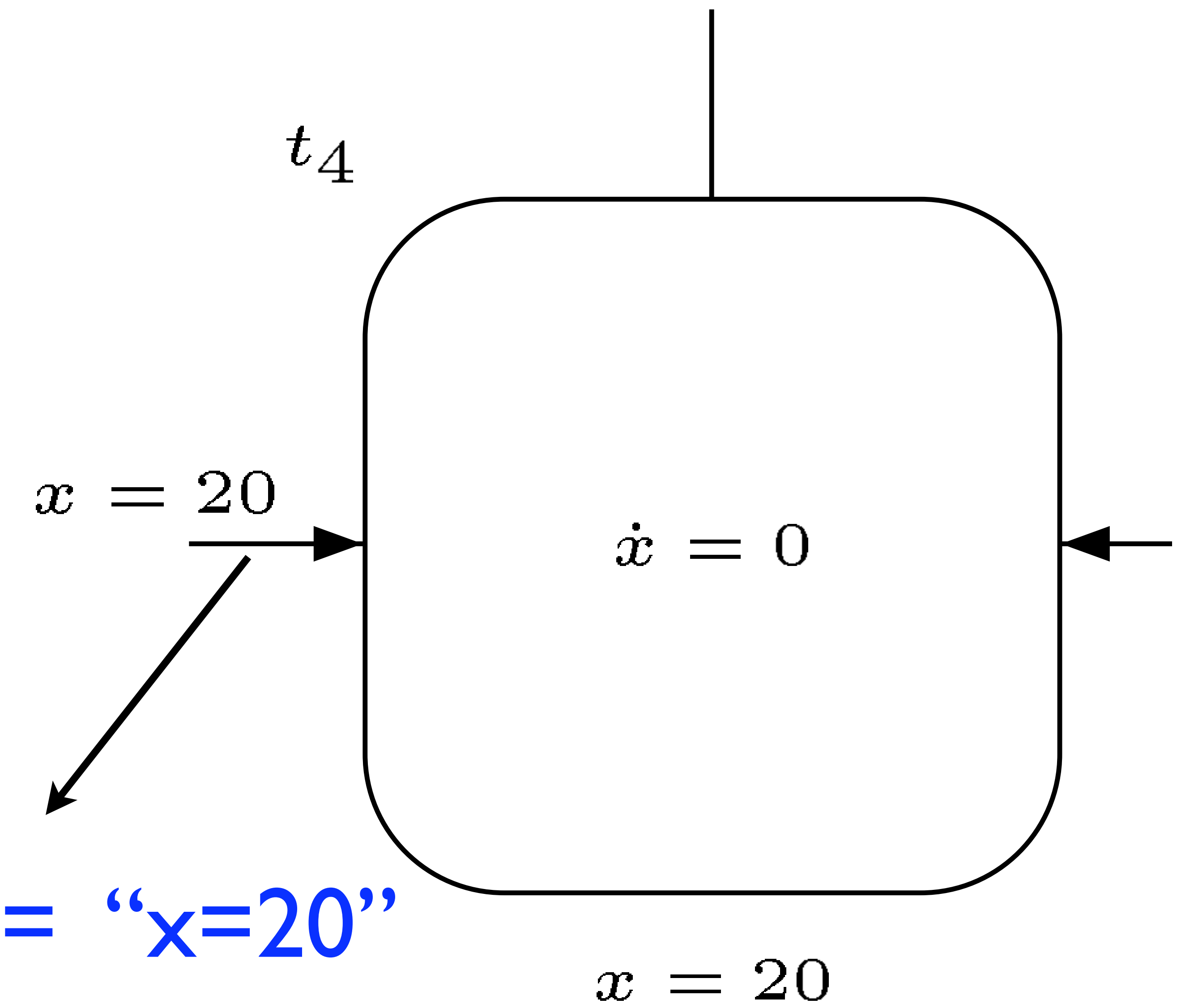
An hybrid automaton for the tank



Hybrid automata - Syntax

- Init(l) is a predicate whose free variables are in X and which states the possible initial valuations for those variables when the automaton starts its execution in l;

An hybrid automaton for the tank

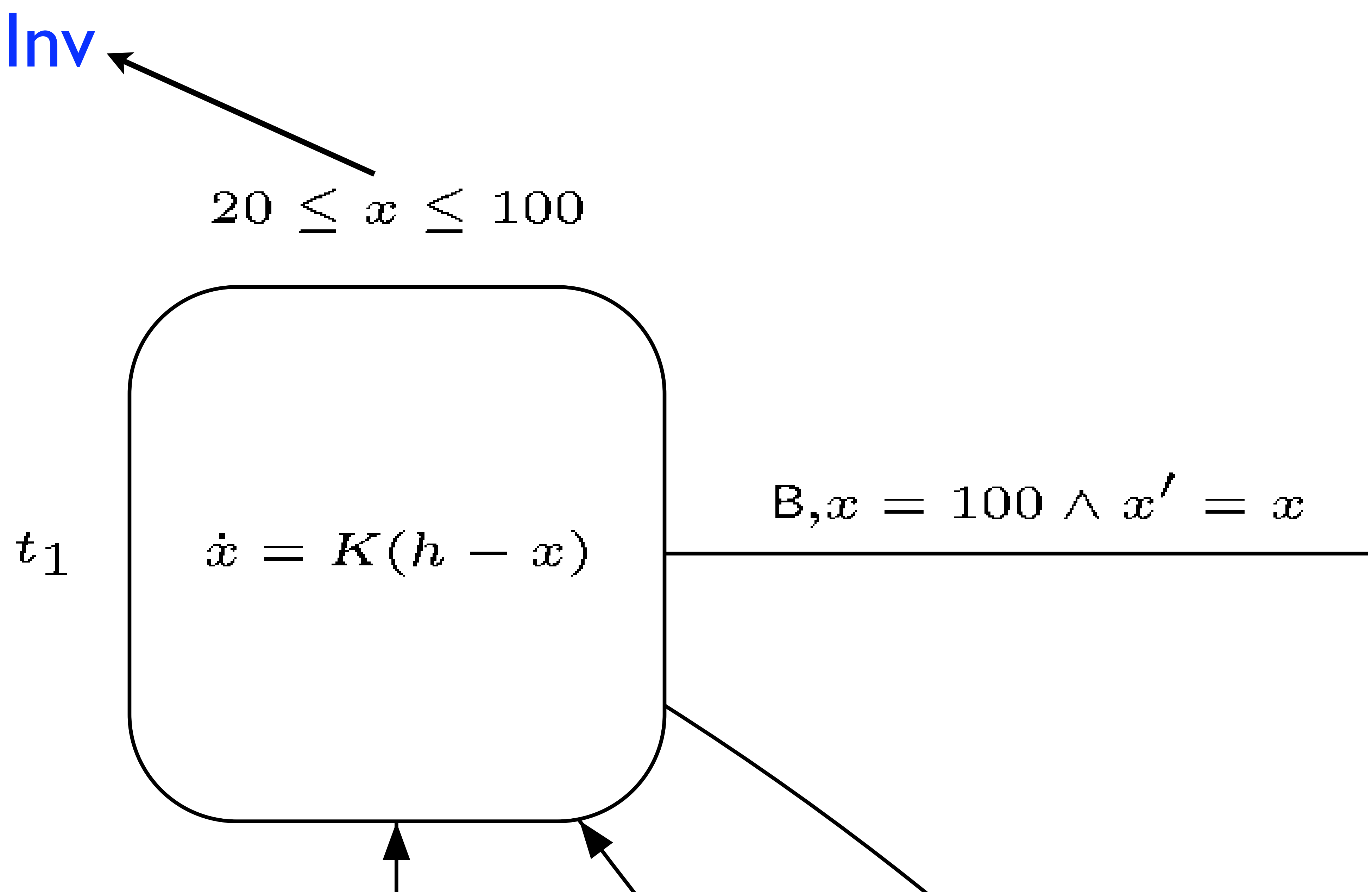


Init(t_4) = "x=20"

Hybrid automata - Syntax

- Init(l) is a predicate whose free variables are in X and which states the possible initial valuations for those variables when the automaton starts its execution in l;
- Inv(l) is a predicate whose free variables are in X and which states the possible valuations for those variables when the control of the automaton lies in l;

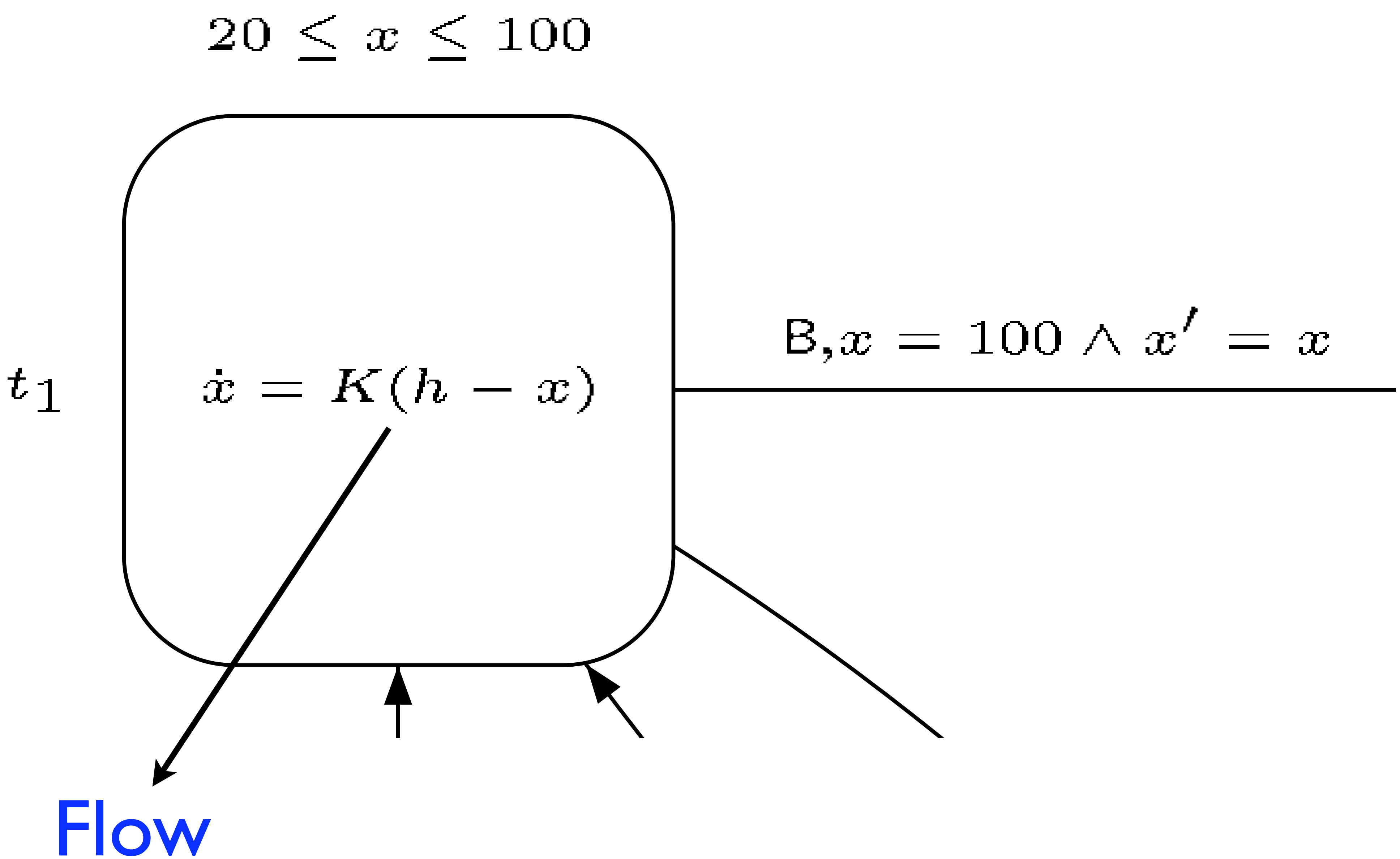
An hybrid automaton for the tank



Hybrid automata - Syntax

- Init(l) is a predicate whose free variables are in X and which states the possible initial valuations for those variables when the automaton starts its execution in l;
- Inv(l) is a predicate whose free variables are in X and which states the possible valuations for those variables when the control of the automaton lies in l;
- Flow(l) is a predicate whose free variables are in $X \cup \dot{X}$ and which states the possible continuous evolutions when the control of the hybrid automaton is in l;
- Jump(e) is a predicate whose free variables are in $X \cup X'$ and which states when the discrete jump is possible and what is its effect on the continuous variables.

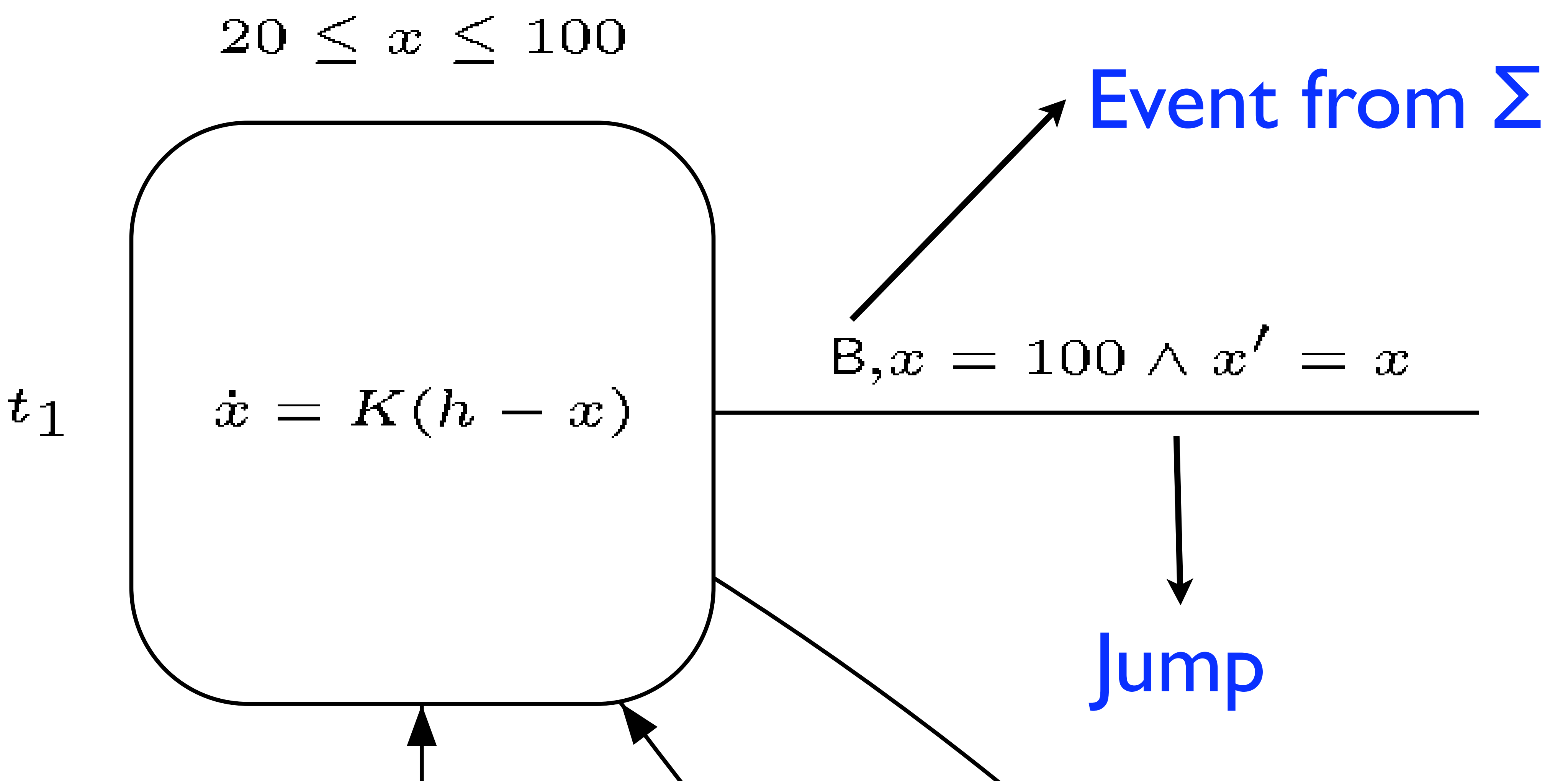
An hybrid automaton for the tank



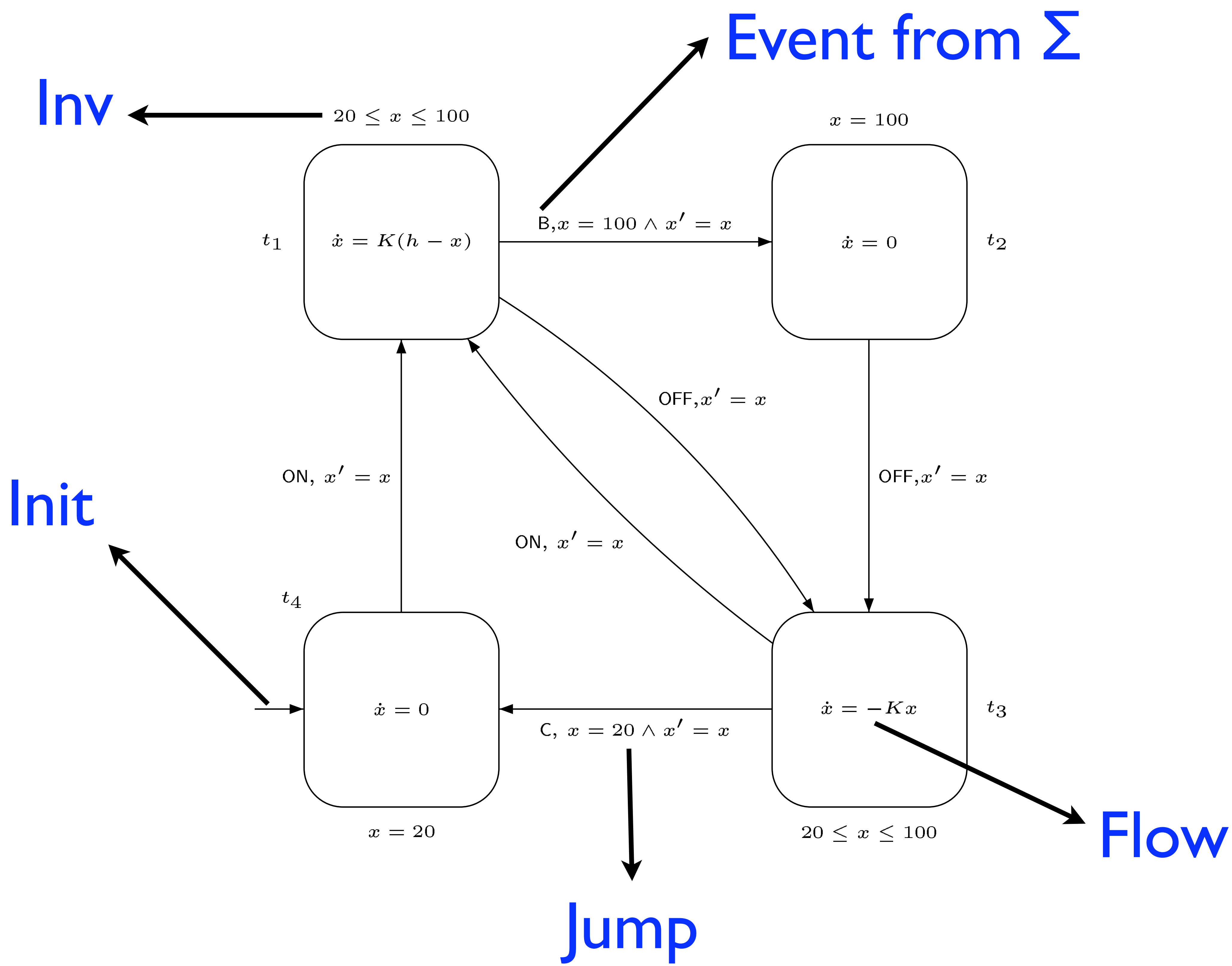
Hybrid automata - Syntax

- Init(l) is a predicate whose free variables are in X and which states the possible initial valuations for those variables when the automaton starts its execution in l;
- Inv(l) is a predicate whose free variables are in X and which states the possible valuations for those variables when the control of the automaton lies in l;
- Flow(l) is a predicate whose free variables are in $X \cup \dot{X}$ and which states the possible continuous evolutions when the control of the hybrid automaton is in l;
- Jump(e) is a predicate whose free variables are in $X \cup X'$ and which states when the discrete jump is possible and what is its effect on the continuous variables.

An hybrid automaton for the tank



An hybrid automaton for the tank



Semantics

- At any instant in time, the state of the hybrid automaton specifies the control location and values for all the real-valued variables.
- The state can change in two ways:
  - discrete: by an instantaneous jump that changes possibly both the control and the values of real-variables;
  - continuous: by a time delay that changes only the values of the real-valued variables in a smooth manner according to the flow and invariant of the current control location.
- To capture such behaviors in a formal way, we use timed transition systems.

Timed transition systems

- A timed transition system (TTS) is a tuple $(S,S_0,\Sigma,\rightarrow)$ where:
    - S is a (possibly infinite) set of states;
    - S_0 is the subset of initial states;
    - Σ is a finite set of labels;
    - $\rightarrow \subseteq S \times \Sigma \cup \mathbb{R}^{\geq 0} \times S$ is the transition relation.

Timed transition systems

- Notations:
  - $[X \rightarrow \mathbb{R}]$ denotes the set of functions from X to $\mathbb{R}$;
  - Let p be a predicate over the set of variables X,
    we note $[\![p]\!]$ the set of valuations $v \in [X \rightarrow \mathbb{R}]$ satisfying p;
  - Let q be a predicate over the set of variables $X \cup X'$,
    we note $[\![q]\!]$ the set of pairs of valuations
    $(v,v') \in [X \rightarrow \mathbb{R} \times X' \rightarrow \mathbb{R}]$ satisfying q;
  - Let r be a predicate over the set of variables $X \cup X^{\cdot}$,
    we note $[\![r]\!]$ the set of pairs of valuations
    $(v,v^{\cdot}) \in [X \rightarrow \mathbb{R} \times X^{\cdot} \rightarrow \mathbb{R}]$ satisfying r.

Timed transition system of a HA

- Let H=(Loc,Σ,Edge,X,Init,Inv,Flow,Jump) be a HA.
- Its associated TTS $[\![H]\!]=(S,S_0,\Sigma,\rightarrow)$ is defined as follows:
  - S is the set of pairs (l,v) where $l \in$ Loc and $v \in [\![Inv(l)]\!]$;
  - S_0 is the subset of pairs $(l,v) \in S$ such that $v \in [\![Init(l)]\!]$;

Timed transition system of a HA

Transition relation

- discrete steps:
  for each edge $e=(l,\sigma,l')\in E$, we have $(l,v)\rightarrow_{\sigma}(l',v')$
  if $(l,v)\in S$, $(l',v')\in S$ and $(v,v')\in[\![Jump(e)]\!]$;

- continuous steps: for each $\delta\in\mathbb{R}\geq 0$, we have $(l,v)\rightarrow_{\delta}(l',v')$
  if $(l,v)\in S$,$(l',v')\in S$, $l=l'$, and there exists a *differentiable*

  function $f:[0,\delta]\rightarrow\mathbb{R}^m$, with derivative $\dot{f}(0,\delta)\rightarrow\mathbb{R}^m$
  such that :

  1) $f(0)=v$,
  2) $f(\delta)=v'$ and
  3)for all $\varepsilon\in(0,\delta)$, both $f(\varepsilon)\in[\![Inv(l)]\!]$ and

  $(f(\varepsilon), \dot{f}(\varepsilon))\in[\![Flow(l)]\!]$.

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

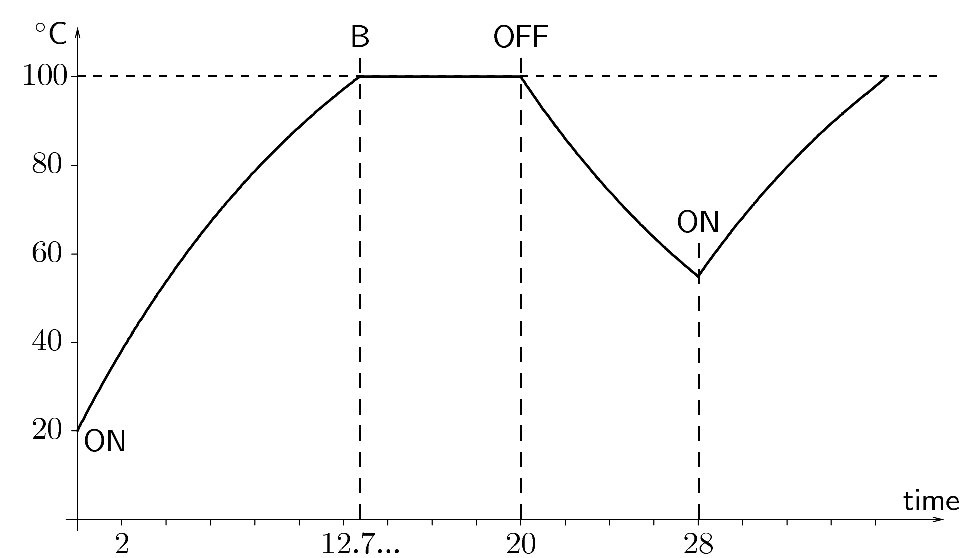
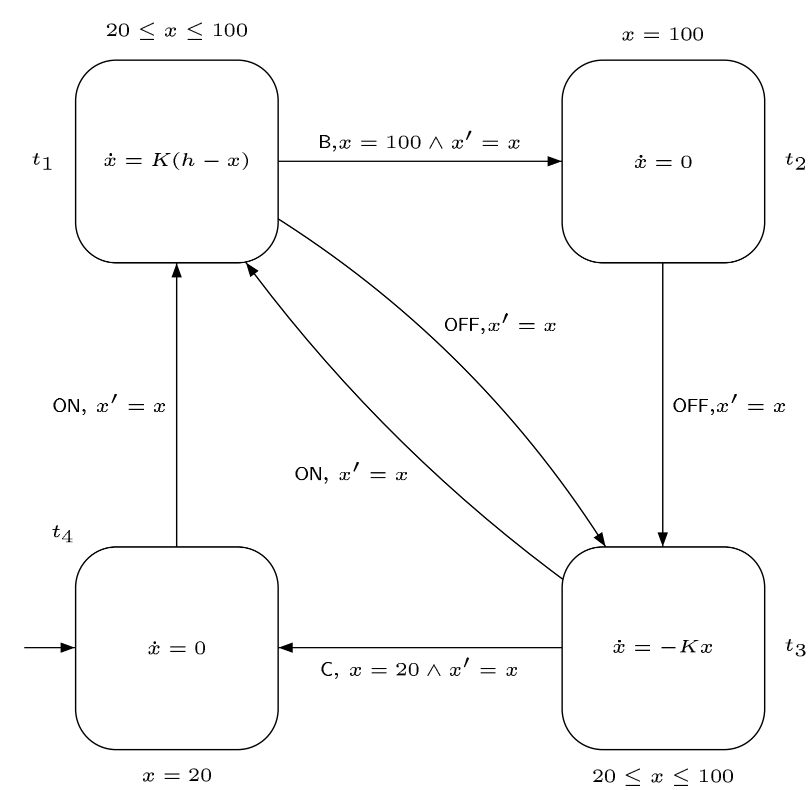


Fig. 2. One possible behavior of the tank



Abstracted by:
(t1,20)

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

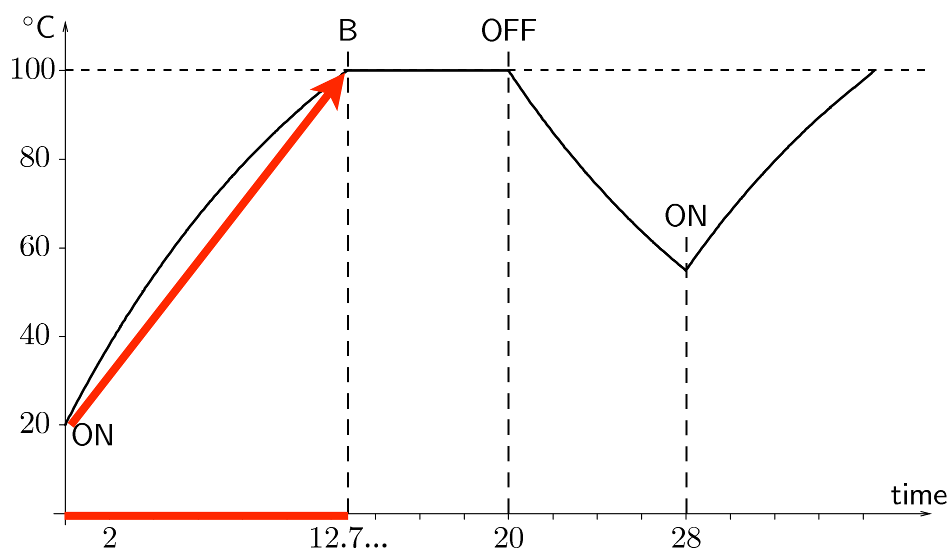
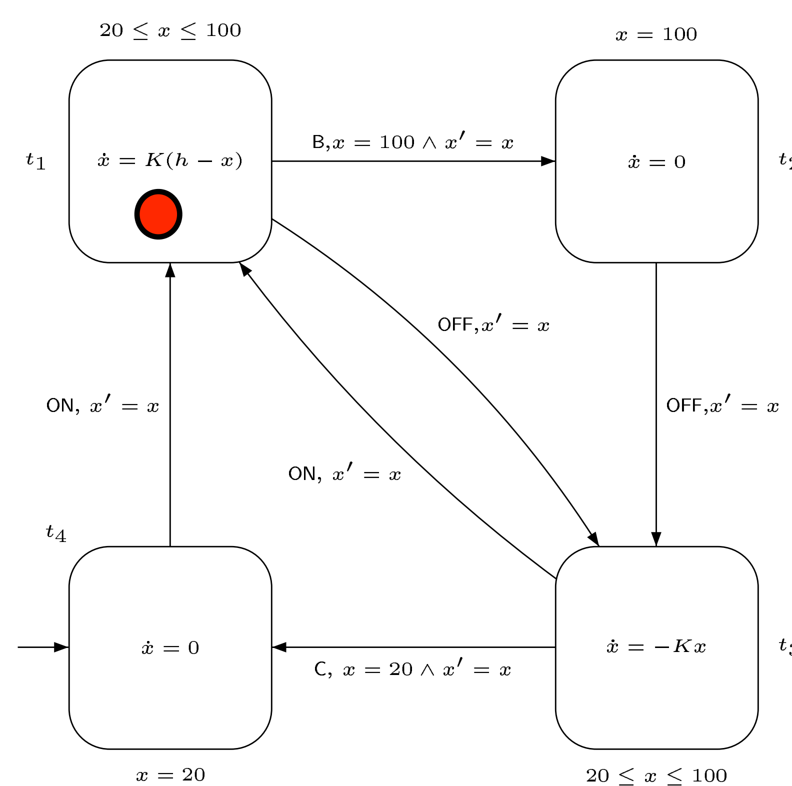


Fig. 2. One possible behavior of the tank



Abstracted by:
(t1,20) →12,7...→(t1,100)

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

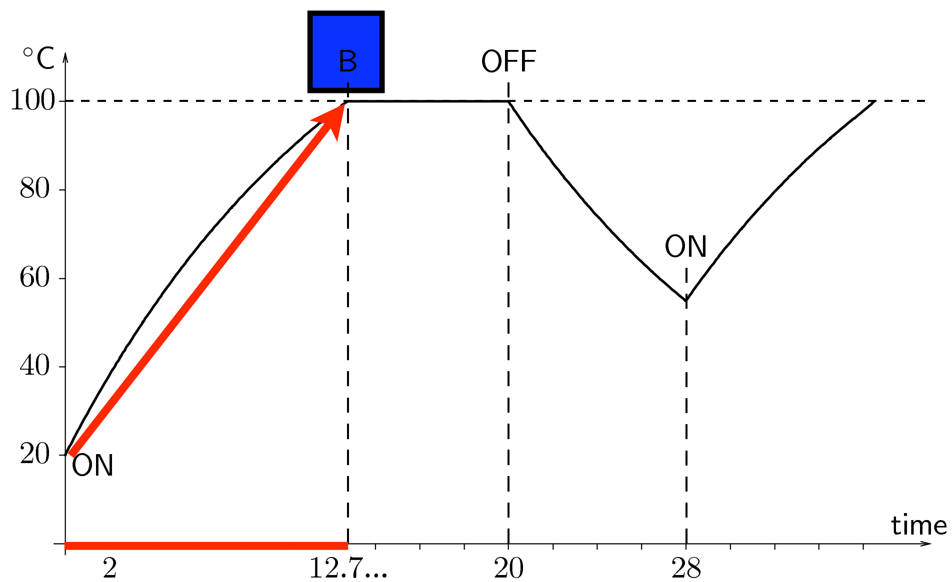
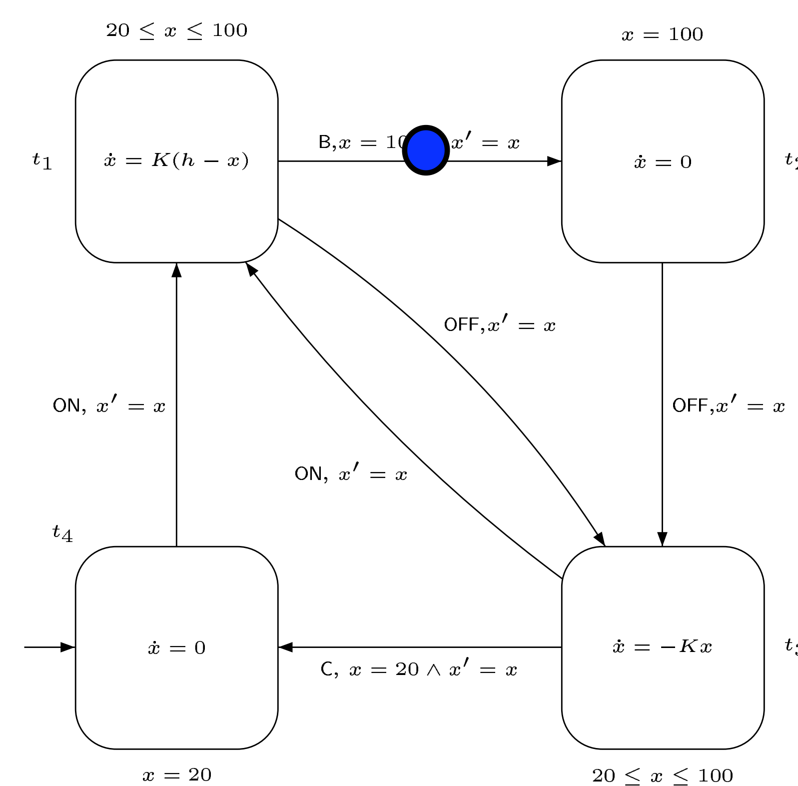


Fig. 2. One possible behavior of the tank



Abstracted by:

(t1,20) →12,7...→(t1,100)→B→ (t2,100)

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

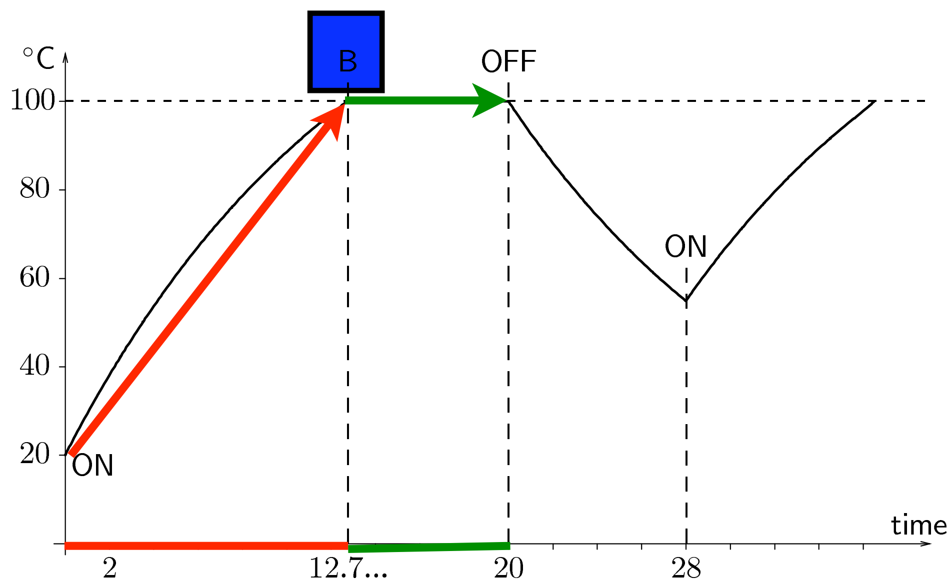
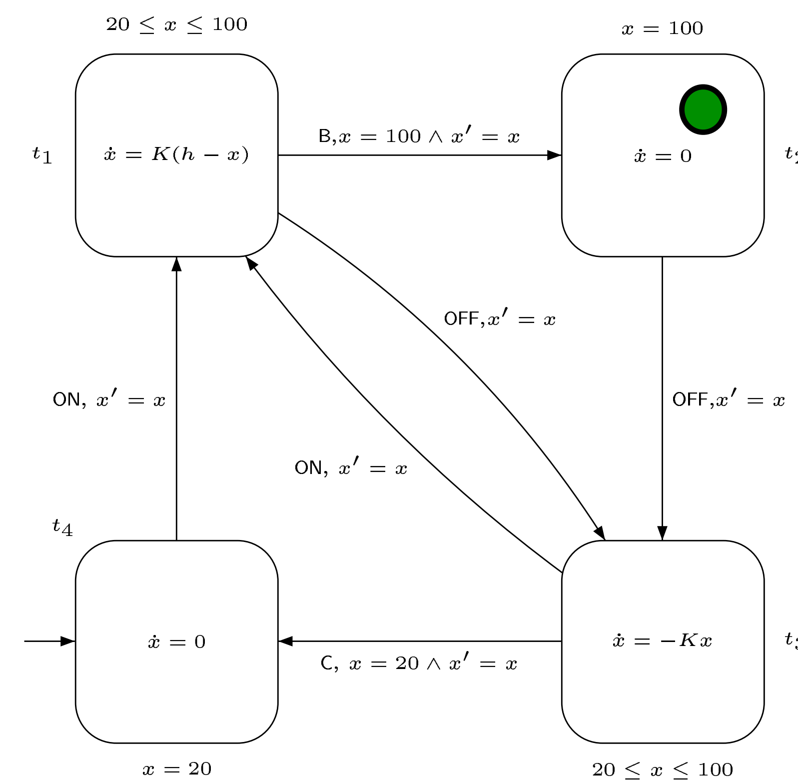


Fig. 2. One possible behavior of the tank



Abstracted by:

(t1,20) →12,7...→(t1,100)→B→ (t2,100)→7,2...→(t1,100)

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

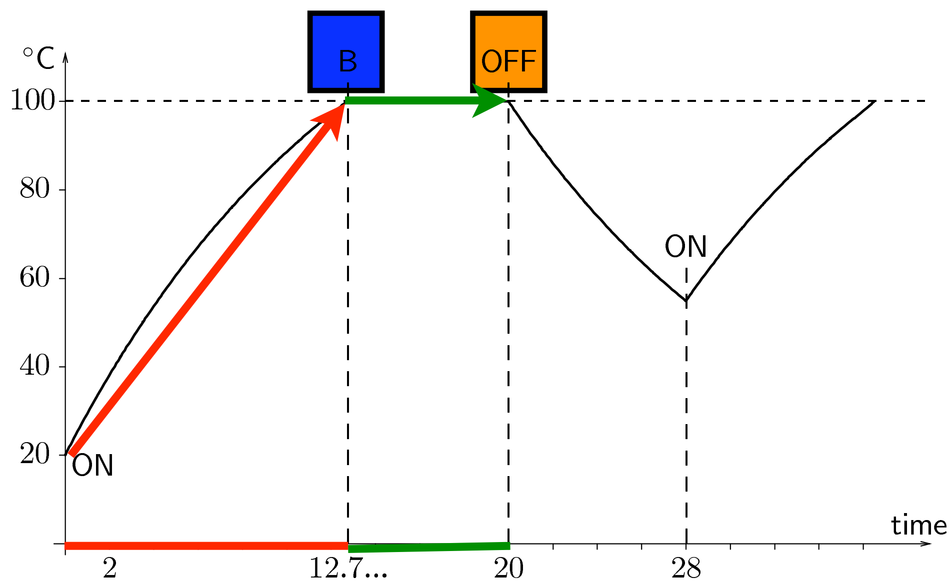
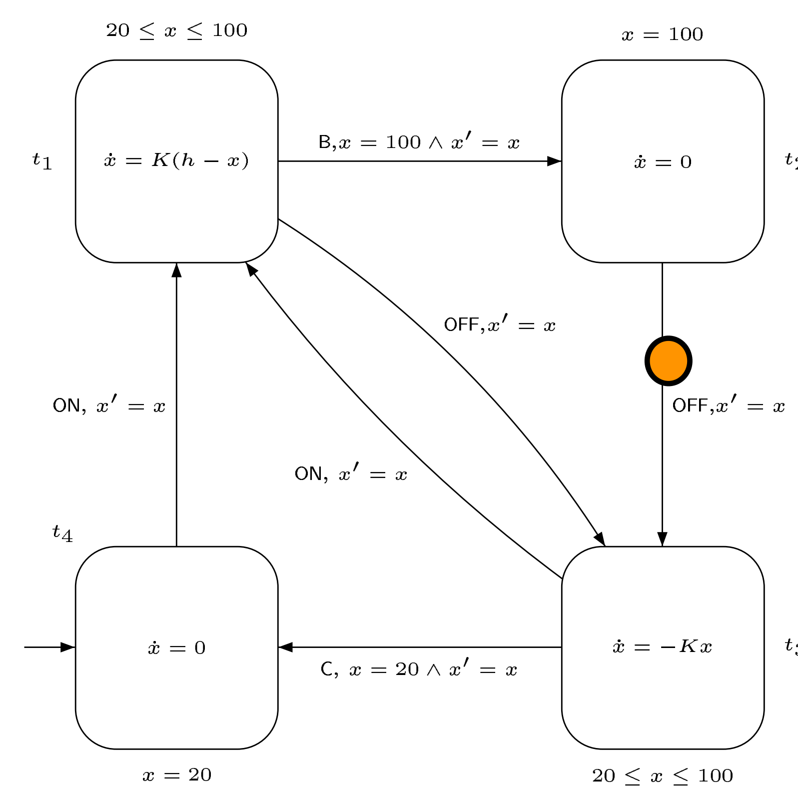


Fig. 2. One possible behavior of the tank



Abstracted by:

(t1,20) →12,7...→(t1,100)→B→ (t2,100)→7,2...→(t1,100)→OFF→(t3,100)

Timed transition system of a HA

- In the timed transition system giving the semantics of the HA, we abstract continuous flows by transitions retaining only the information about the source, target and duration of each flow.

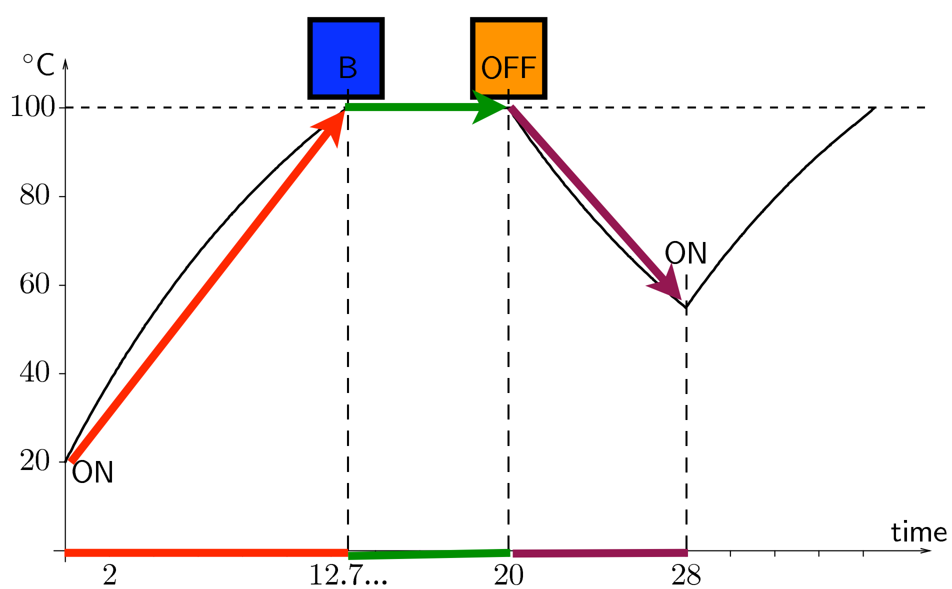
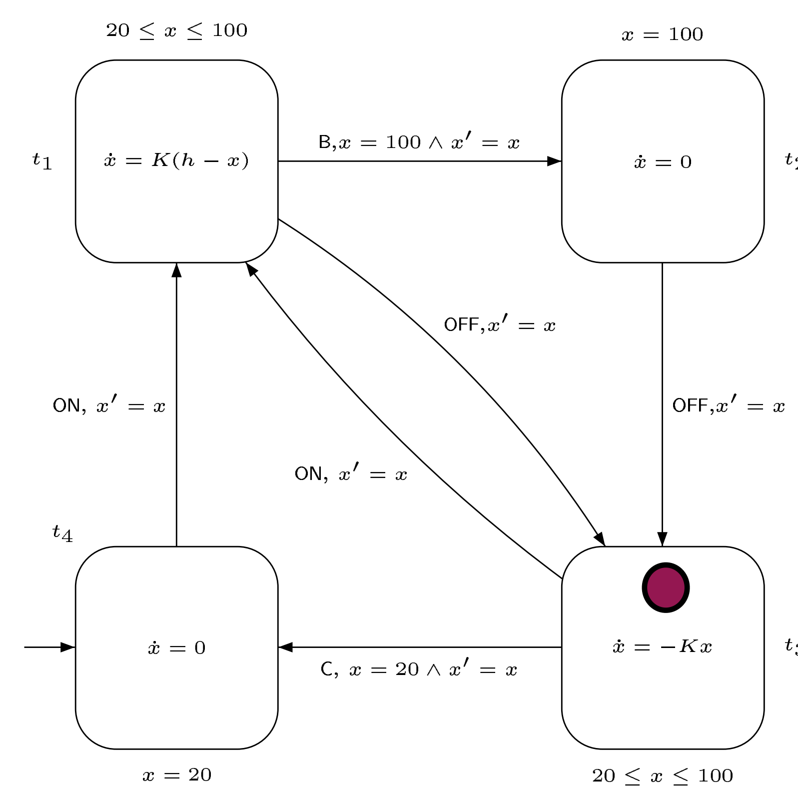


Fig. 2. One possible behavior of the tank



Abstracted by:

(t1,20) →12,7...→(t1,100)→B→ (t2,100)→7,2...→(t1,100)→OFF→(t3,100)→8→(t3,60)...

Behaviors of HA=paths in TTS

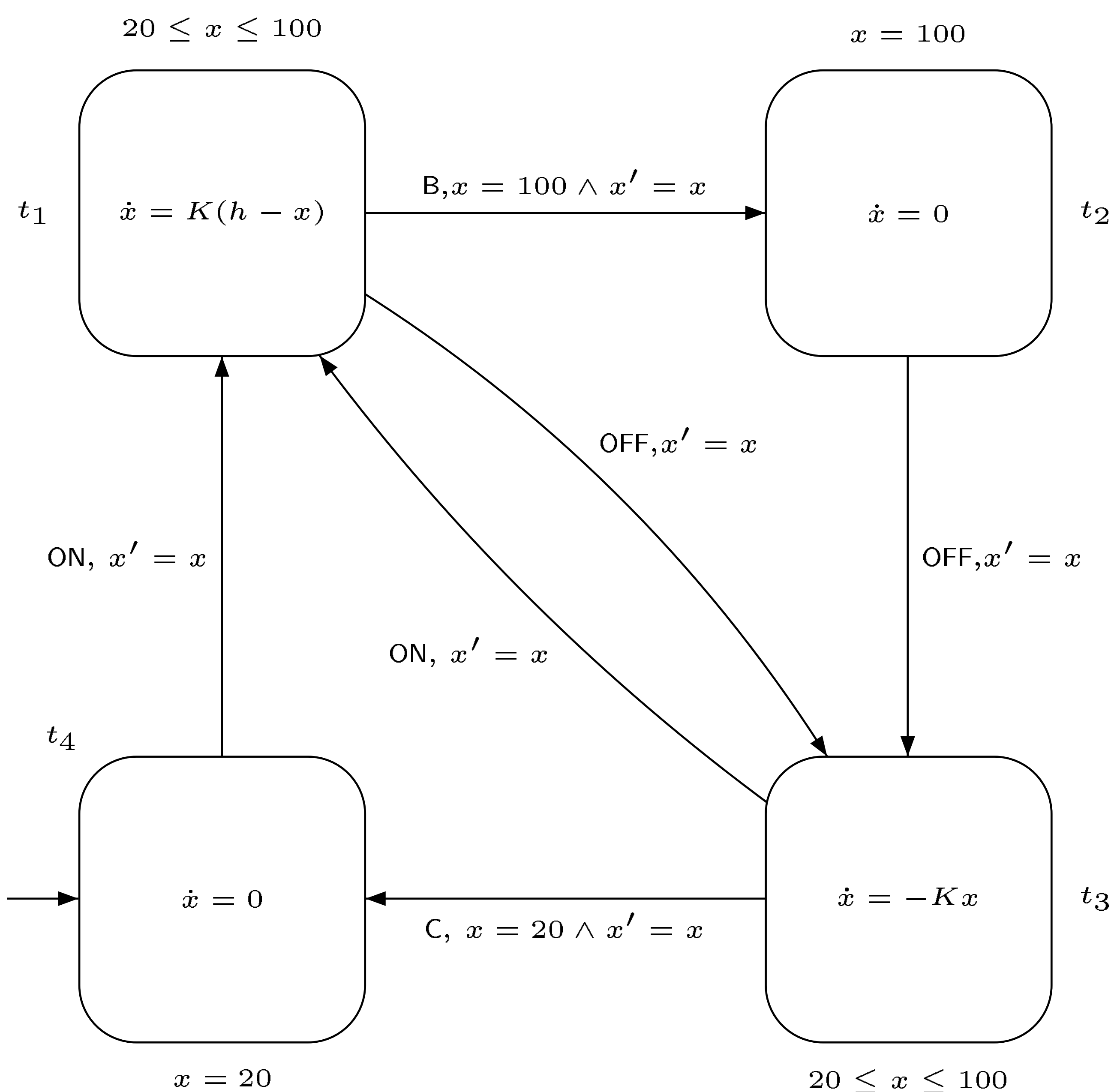
- The paths contained in the TTS formalize the behaviors of the HA;
- Formally, a finite path, noted λ, in the TSS $T=(S,S0,\Sigma,\rightarrow)$ is finite sequence $s_0\tau_0 s_1\tau_1\ldots\tau_{n-1}s_n$ such that for all i, $1\leq i\leq n$, $(s_i,\tau_i,s_{i+1})\in\rightarrow$.

  This definition extends to infinite paths.
- The duration of a path is the sum of the durations of its time elapsing steps.
- We write $Path_F(\llbracket H\rrbracket)$ for the set of finite paths in $\llbracket H\rrbracket$ and $Path_\infty(\llbracket H\rrbracket)$ for the set of infinite paths in $\llbracket H\rrbracket$.

Ex. of an element in PathF(⟦Tank⟧)

$$(t_4, x \mapsto 20) \overbrace{\rightarrow_{\mathsf{ON}}}^{(1)} (t_1, x \mapsto 20) \overbrace{\rightarrow_{10}}^{(2)} (t_1, x \mapsto 88.59\ldots) \overbrace{\rightarrow_{2.74\ldots}}^{(3)} (t_1, x \mapsto 100) \overbrace{\rightarrow_{\mathsf{B}}}^{(4)}$$

$$(t_2, x \mapsto 100) \overbrace{\rightarrow_{5}}^{(5)} (t_2, x \mapsto 100) \overbrace{\rightarrow_{\mathsf{OFF}}}^{(6)} (t_3, x \mapsto 100) \overbrace{\rightarrow_{8}}^{(7)} (t_3, x \mapsto 54.88\ldots)$$



(1) discrete step

(2) f(t)=20e$^{-0.075t}$+150(1-e$^{-0.075t}$)
on [0,10]
and f(0)=20, f(10)=88,59...

(3) f(t)=88,59...e$^{-0.075t}$+150(1-e$^{-0.075t}$)
on [0,2.75]

...

(5) f(t)=100

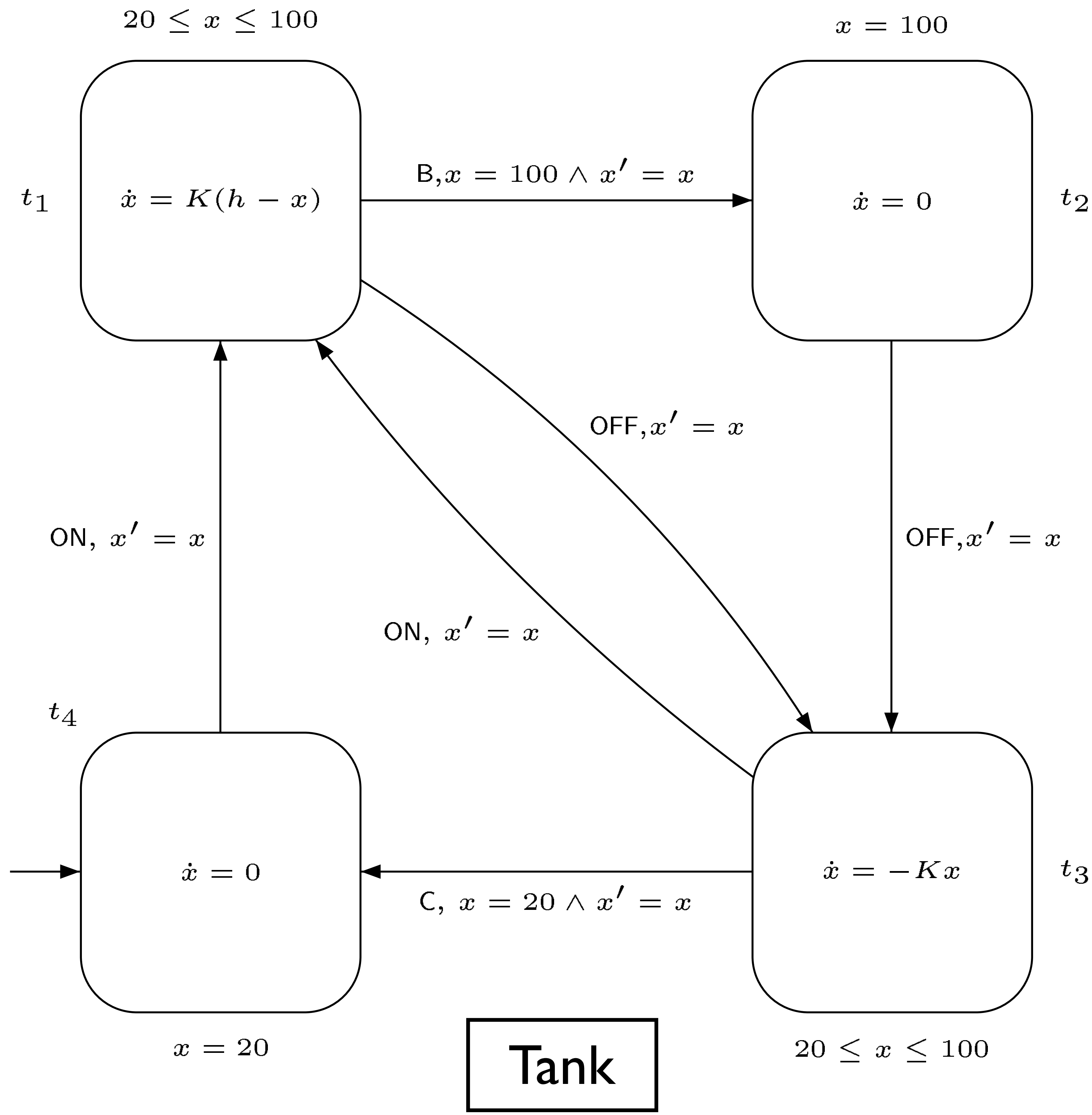
Remark on non Zenoness

- Often, when considering behaviors of systems along (real) time, we are interested by nonZeno behaviors, that is behaviors in which time is not blocked.
- In fact, a trajectory in which there are discrete jumps say at times 0.5, 0.75, 0.875, 0.9375, 0.96875, ... is not implementable by a discrete controller.
- We say that an infinite path λ is nonZeno if $\mathrm{Duration}(\lambda)=+\infty$.
- The divergence of time is a liveness assumption.

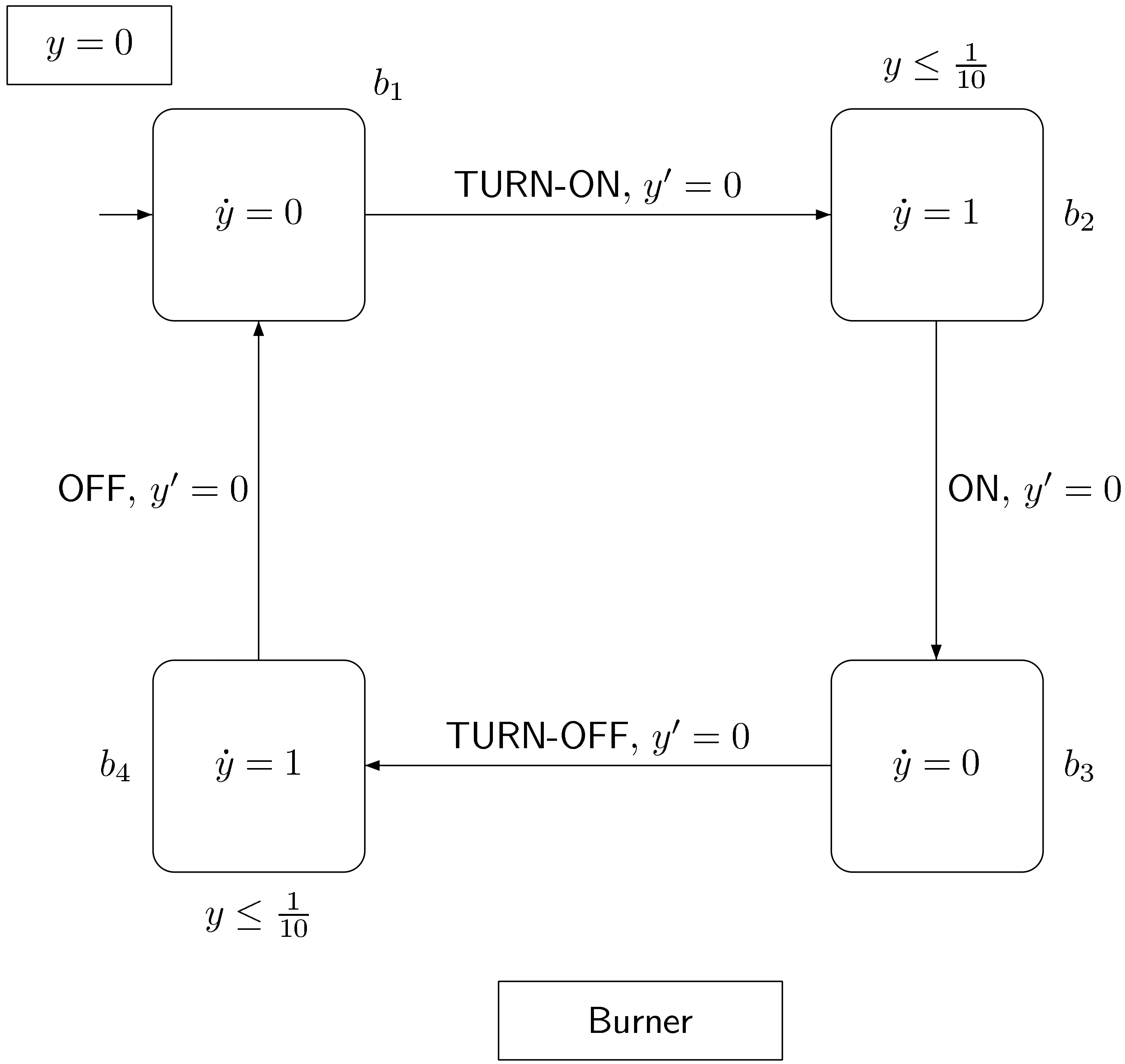
Composition of HA

- Nontrivial hybrid systems consist of several interacting components;
- We model each component as a hybrid automaton ...
- ... and the components coordinate with each other by shared variables and shared events;

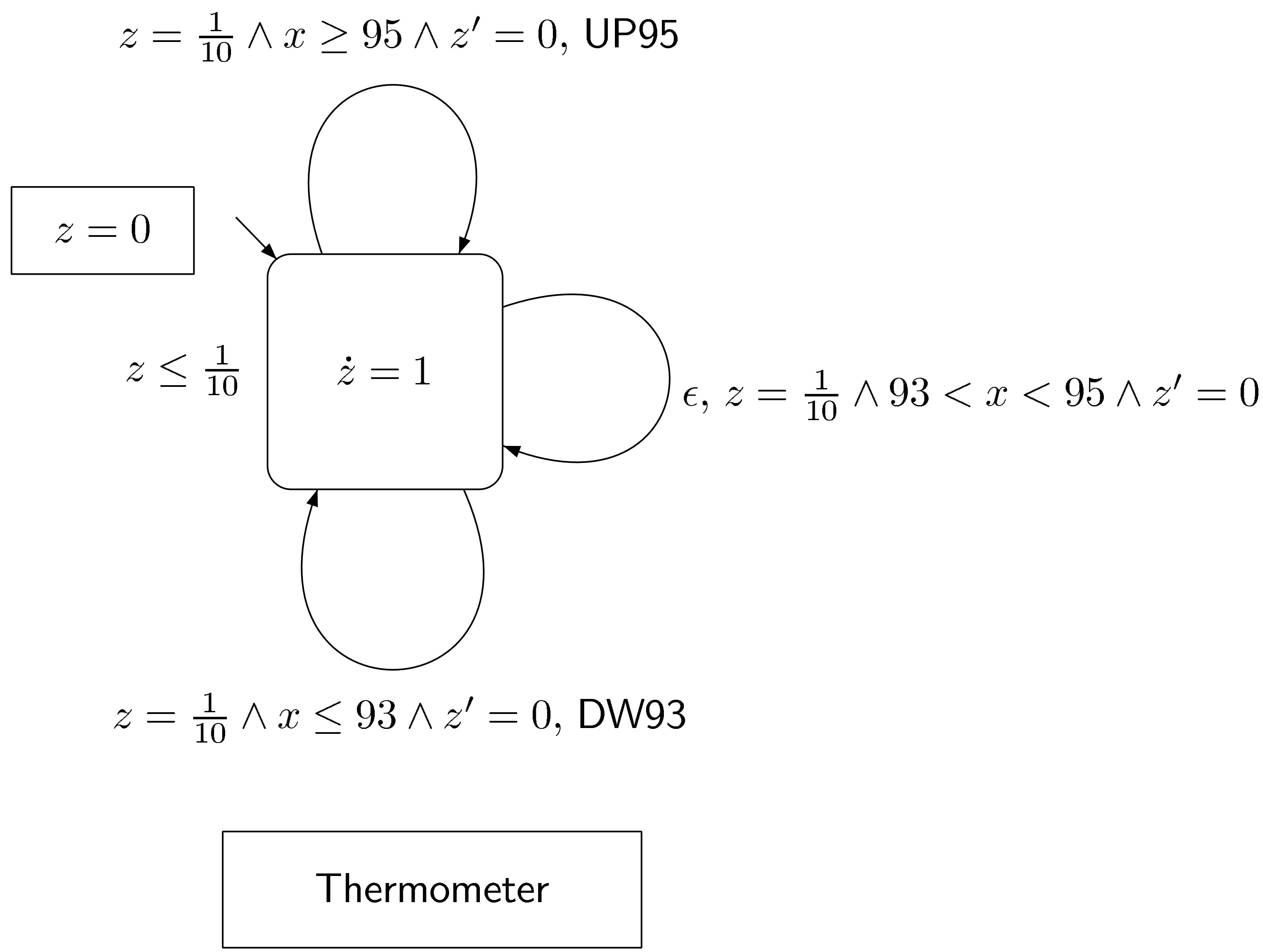
Our running example



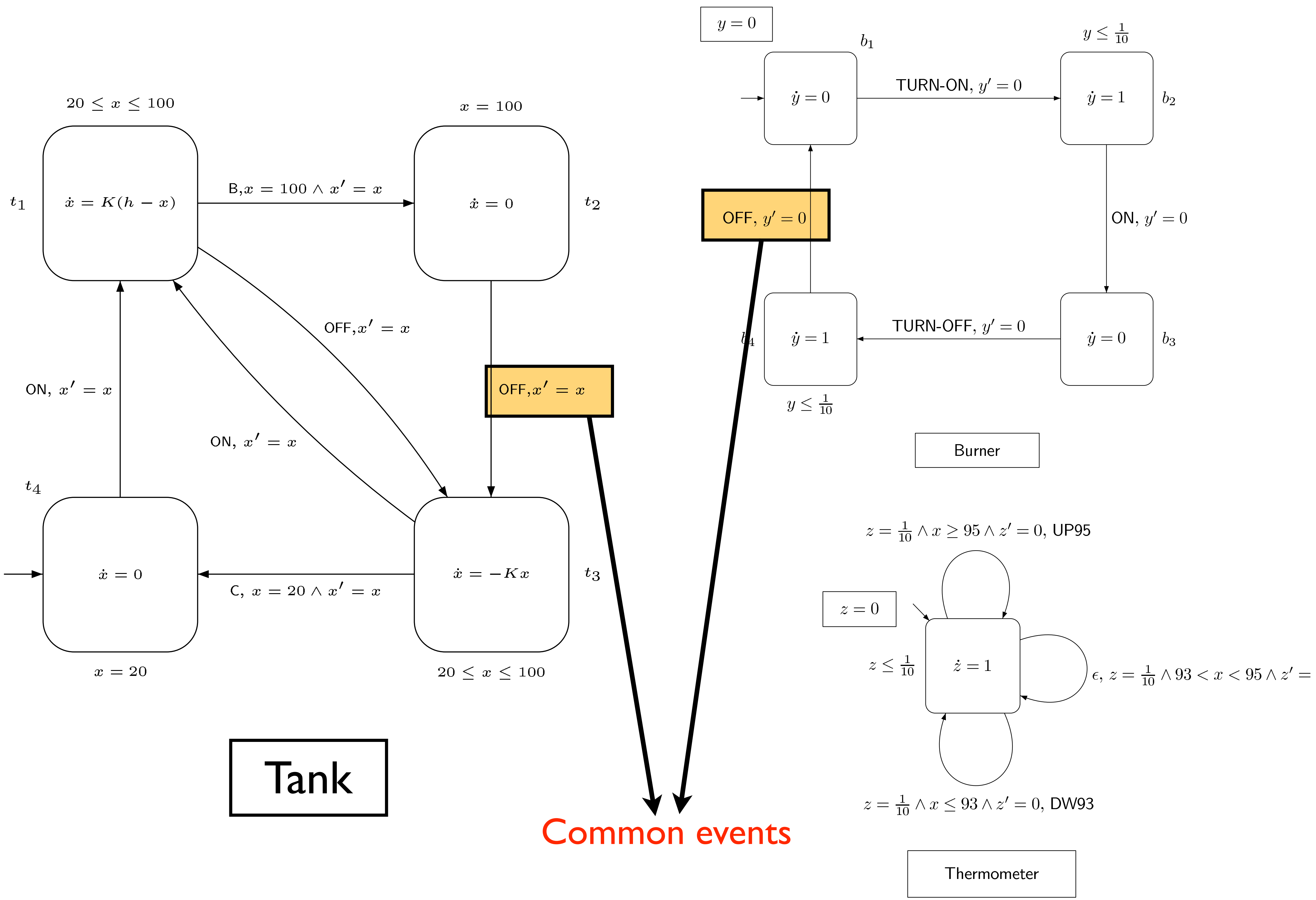
Our running example



Our running example



Our running example



HA product

- Let $H^1=(Loc^1,\Sigma^1,Edge^1,X^1,Init^1,Inv^1,Flow^1,Jump^1)$ and
  $H^2=(Loc^2,\Sigma^2,Edge^2,X^2,Init^2,Inv^2,Flow^2,Jump^2)$.
- Their synchronized product is the hybrid automaton $H_1 \otimes H_2=(Loc,\Sigma,Edge,X,Init,Inv,Flow,Jump)$ defined as follows:
  - $Loc=\{ \{l^1,l^2\} \mid l^1 \in Loc^1 \wedge l^2 \in Loc^2 \}$
  - $\Sigma=\Sigma^1 \cup \Sigma^2$; $X=X^1 \cup X^2$;
  - $Init(\{l^1,l^2\})=Init^1(l^1) \wedge Init^2(l^2)$;
    $Inv(\{l^1,l^2\})=Inv^1(l^1) \wedge Inv^2(l^2)$;
    $Flow(\{l^1,l^2\})=Flow^1(l^1) \wedge Flow^2(l^2)$;

HA product

- $(\{l^1,l^2\},\sigma,\{l^3,l^4\}\} \in$ Edge iff either
  (i) $\sigma\in\Sigma^1\backslash\Sigma^2$, $(l^1,\sigma,l^3)\in\text{Edge}^1$, and $l^2=l^4$;
  (ii) $\sigma\in\Sigma^2\backslash\Sigma^1$, $(l^2,\sigma,l^4)\in\text{Edge}^2$, and $l^1=l^3$;
  (iii) $\sigma\in\Sigma^1\cap\Sigma^2$, $(l^1,\sigma,l^3)\in\text{Edge}^1$, and $(l^2,\sigma,l^4)\in\text{Edge}^2$.

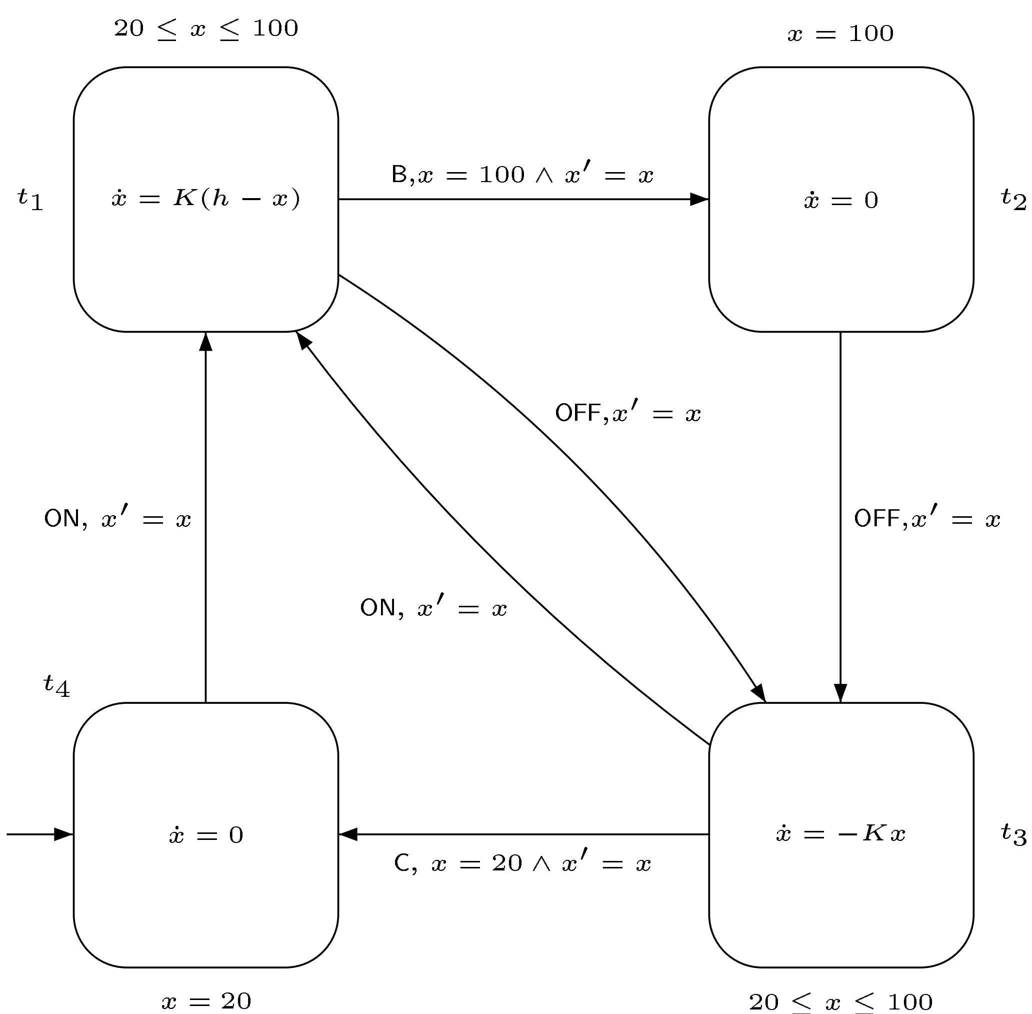
- for any edge $(\{l^1,l^2\},\sigma,\{l^3,l^4\})\in$Edge, we have that:

  (i) $\text{Jump}(\{l^1,l^2\},\sigma,\{l^3,l^4\})$
  $=\text{Jump}^1(l^1,\sigma,l^3)\wedge\bigwedge_{x\,\in X\backslash X^1} x'=x$ if $\sigma\in\Sigma^1\backslash\Sigma^2$

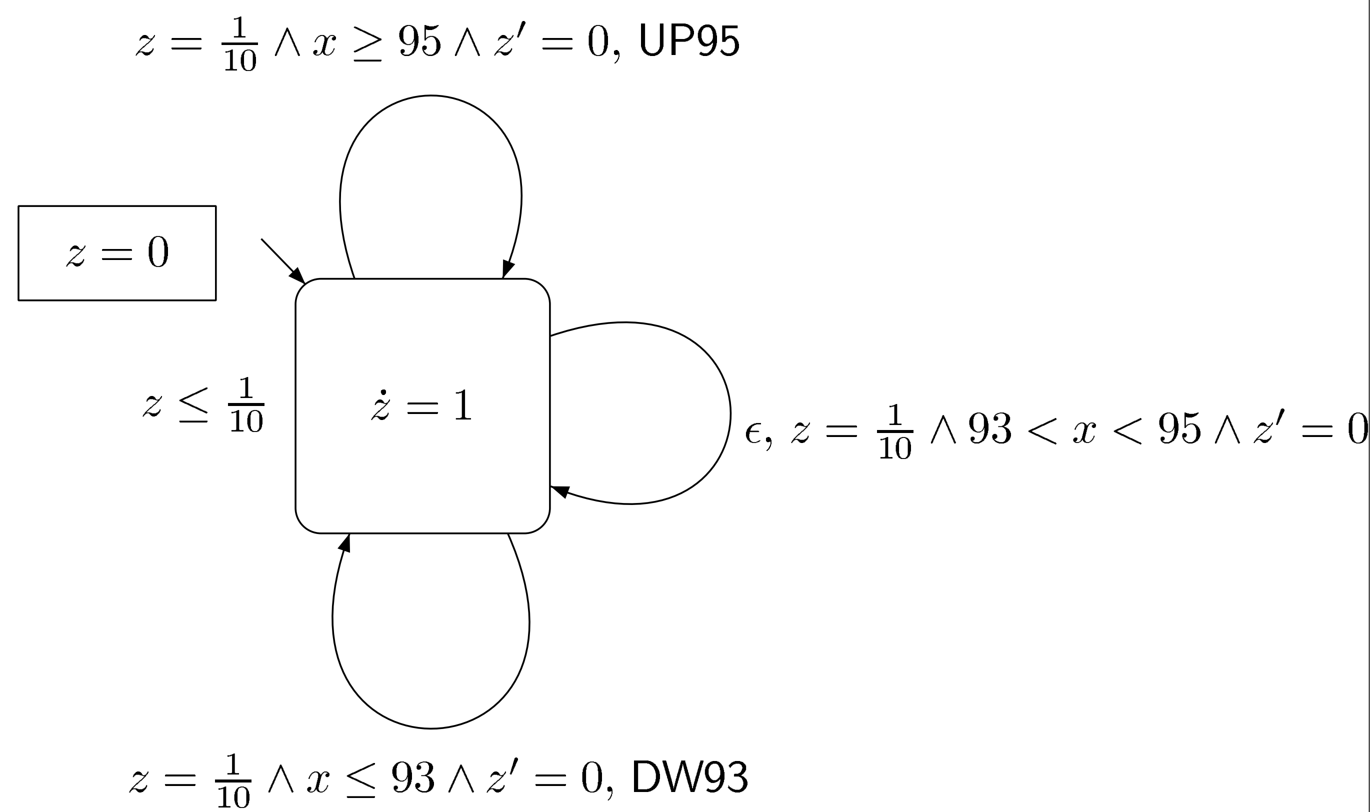
  (ii) Symmetrically for $\sigma\in\Sigma^2\backslash\Sigma^1$

  (ii) $\text{Jump}(\{l^1,l^2\},\sigma,\{l^3,l^4\})$
  $=\text{Jump}^1(l^1,\sigma,l^3)\wedge\text{Jump}^2(l^2,\sigma,l^4)$ if $\sigma\in\Sigma^1\cap\Sigma^2$

Ex. product of Burner and Thermometer

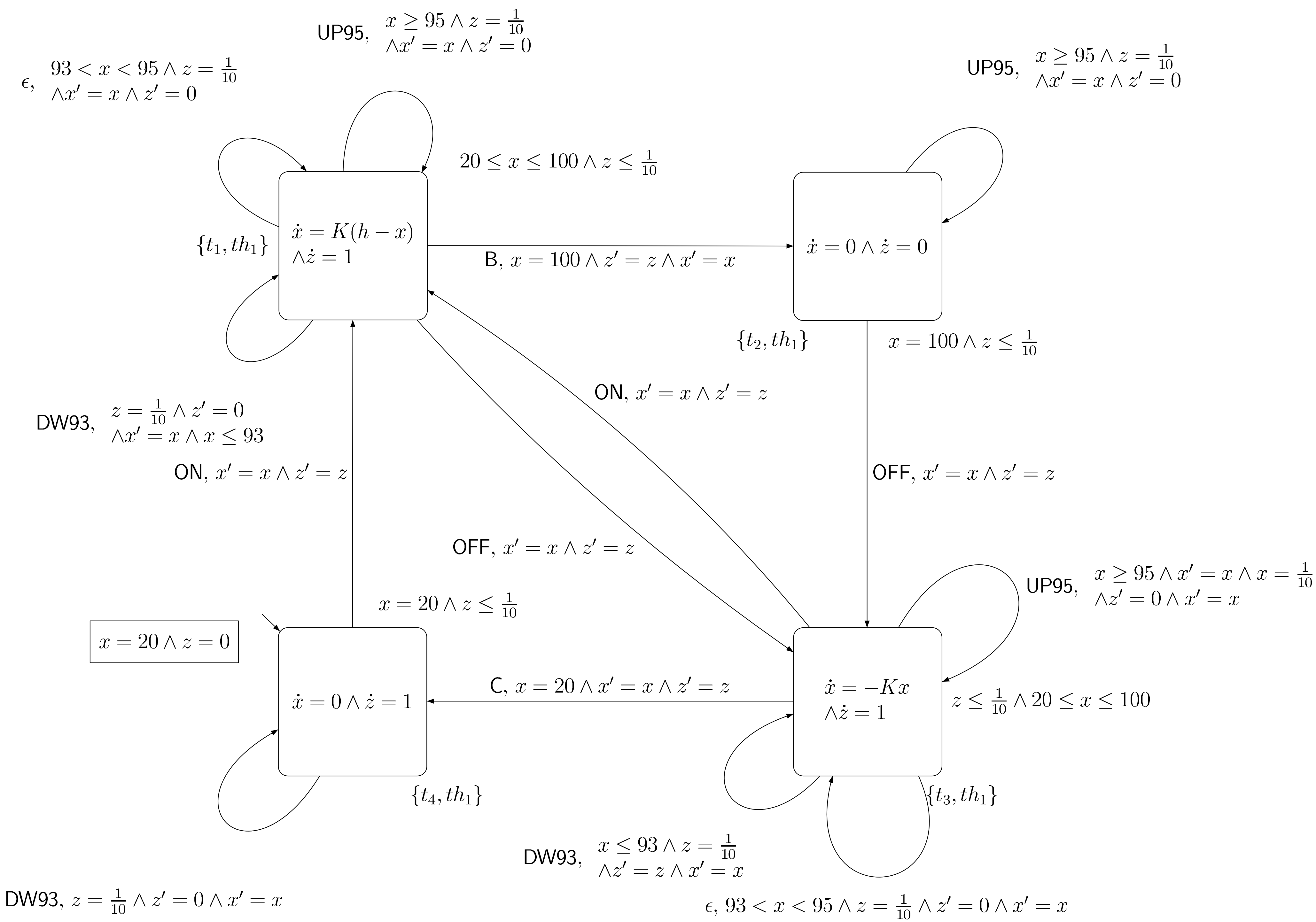


Tank

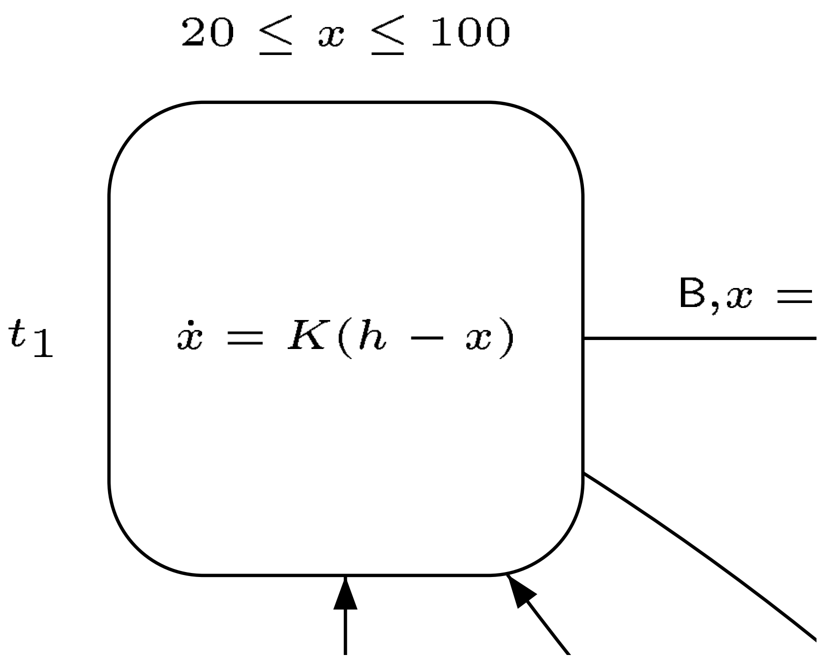


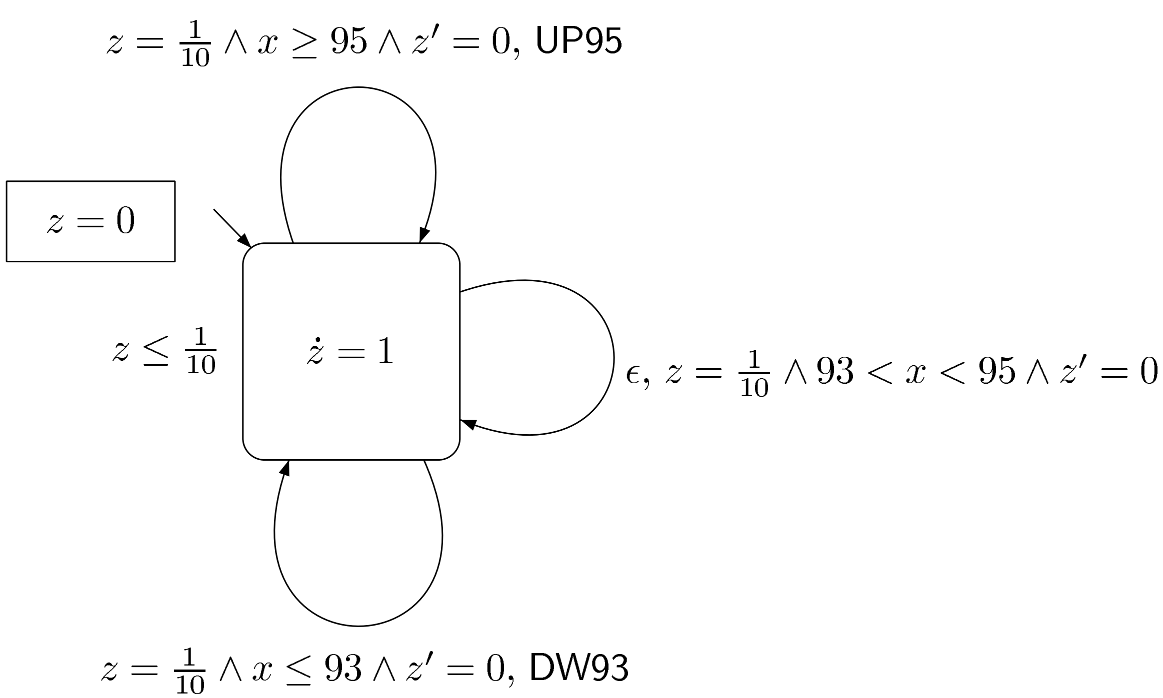
Thermometer

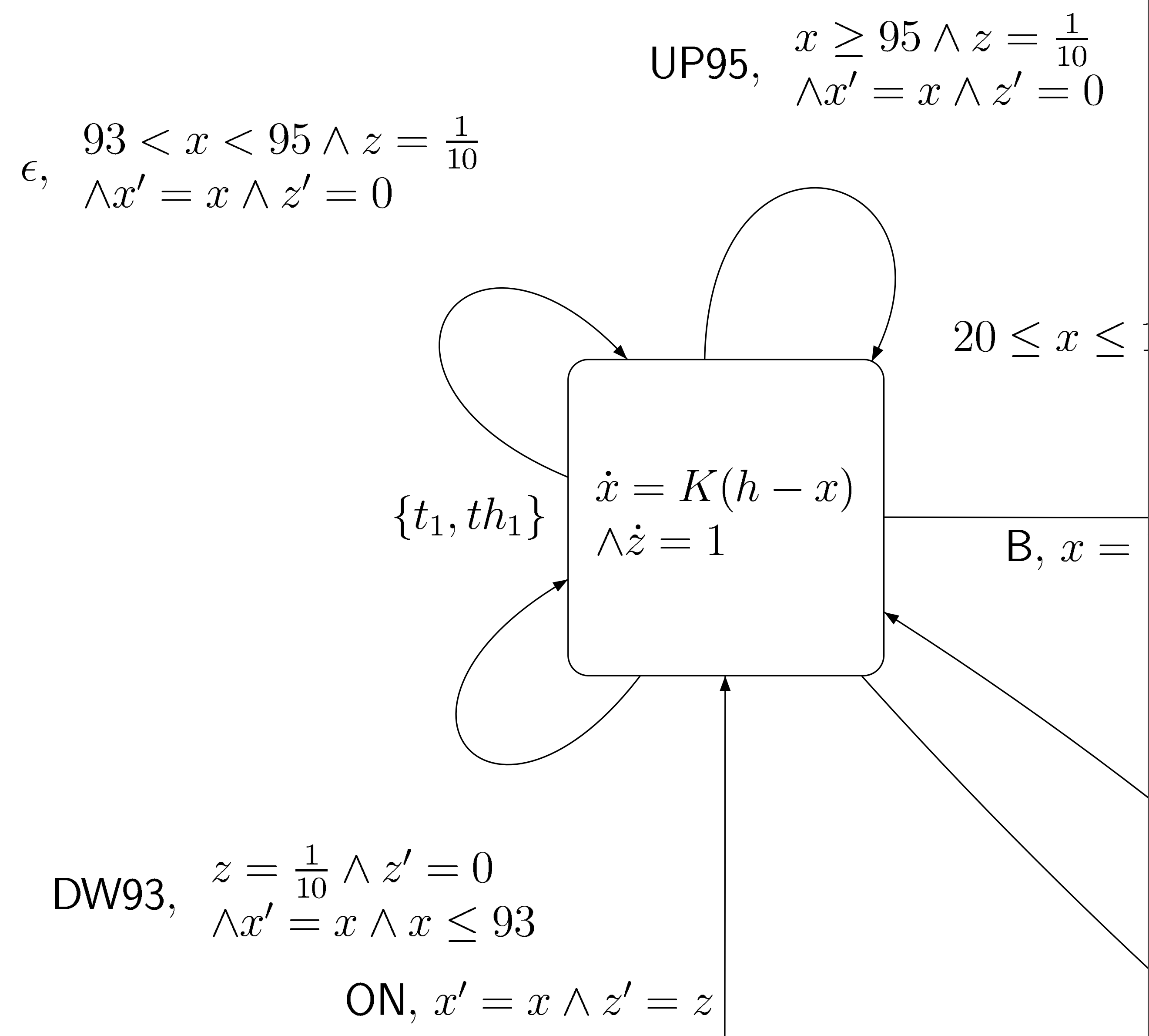
Ex. product of Burner and Thermometer



Ex. product of Burner and Thermometer







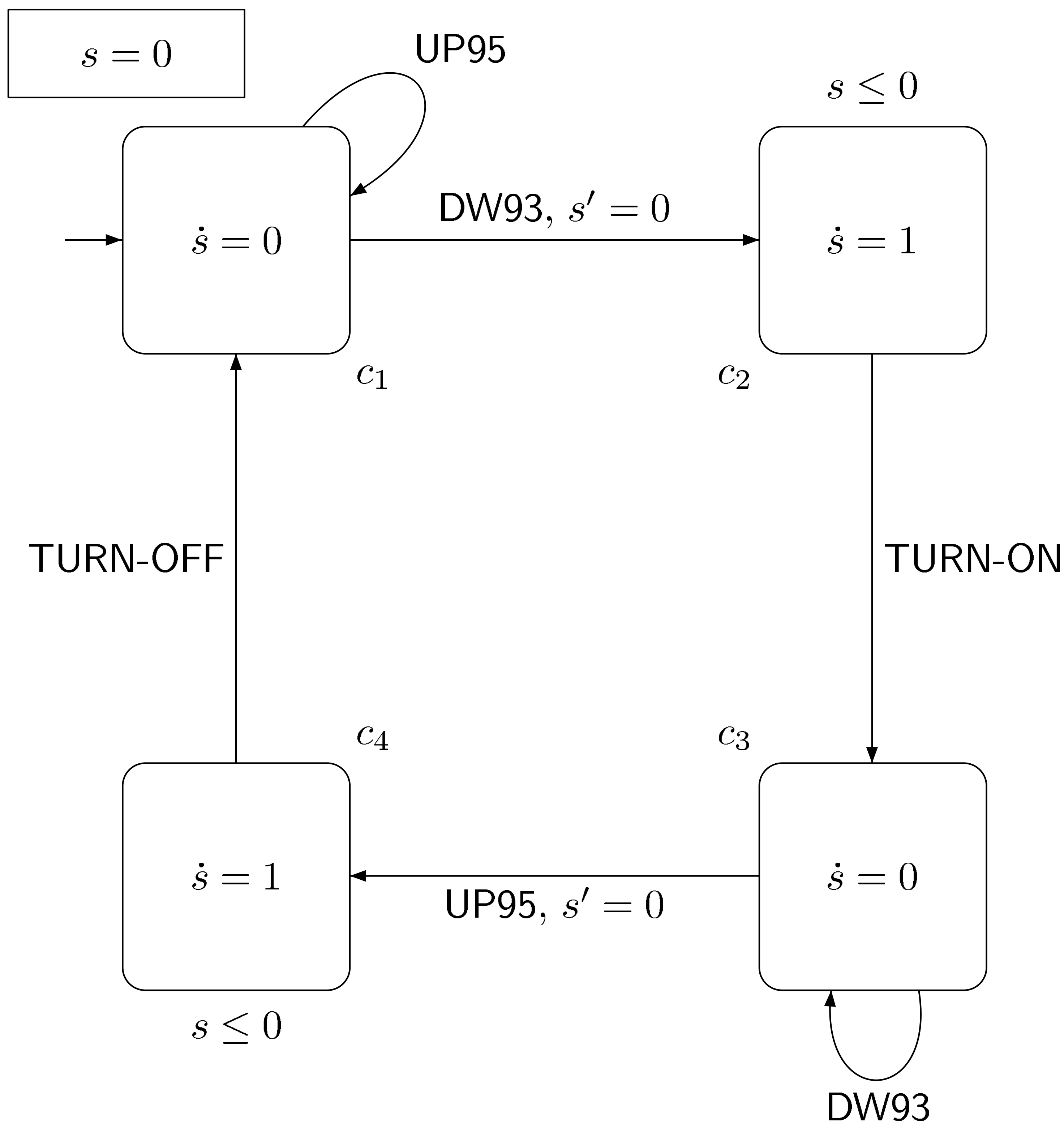
Properties of Hybrid Systems

Ex. of properties for our running example

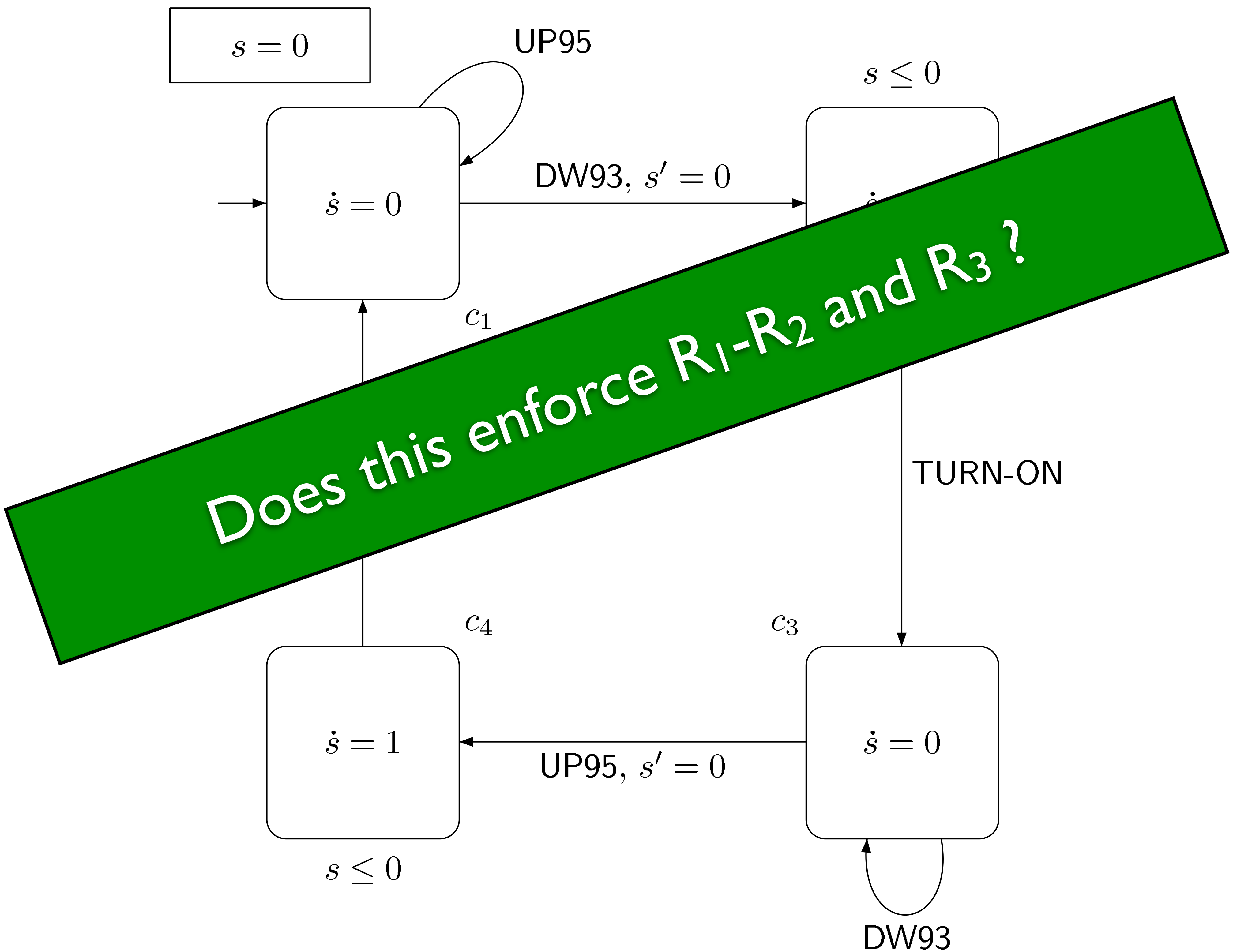
- (R_1) the temperature in the tank must never reach 100°;
- (R_2) after 15 seconds of operation, the system must be in stable regime (the temperature must stay in the interval 91°-97° Celcius);
- (R_3) during this stable regime, the burner is never continously ON for more than two seconds.

All those properties are safety properties.

Candidate controller for our system



Candidate controller for our system



Safety and reachability

- To formalize safety, we need some more notations.
- Let $T=(S,S_0,\Sigma,\rightarrow)$ be a TTS. Let $\lambda=s_0\tau_0s_1\tau_1\ldots s_n \in Path_F(T)$. State($\lambda$) denotes the set of states that appear along λ.
- We say that a path λ reaches a state s if $s \in State(\lambda)$.
- We say that s is reachable in T if $s \in \cup_{\lambda \in Path_F(T)} State(\lambda)$.
- Reach(T) denotes the set of states reachable in T.

Safety and reachability

- A set of state $R \subseteq S$ is called a region.
- A region R is reachable in T iff $R \cap Reach(T) \neq \varnothing$.
- The rechability problem associated to a TTS T and a region R asks if $R \cap Reach(T) \neq \varnothing$.
- The safety problem associated to a TTS T and a region R asks if $Reach(T) \subseteq R$.
- Those two problems are dual in the following formal sense:

  Let R be a region and $R' = S \setminus R$.

  $Reach(T) \subseteq R$ iff $R' \cap Reach(T) = \varnothing$.

Monitors

- Requirement R_1 can be formalized using a region of Bad states.

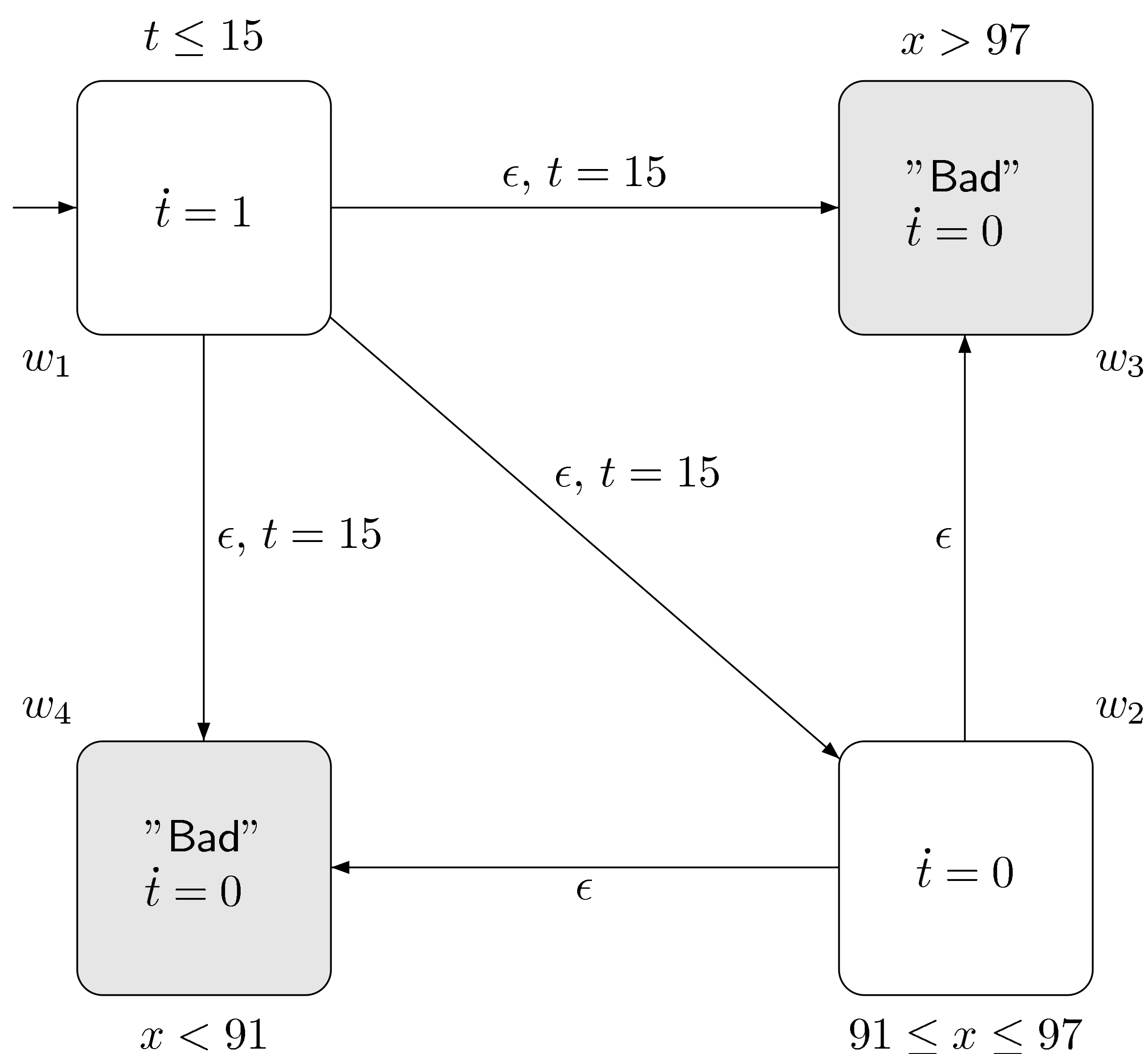
  The system is correct if we avoid the region Bad i.e., Bad is unreachable.

- Requirements R_2 and R_3 can not be formalized directly using regions.

  Instead, we will use monitors.

- A monitor (also called observer) is an HA that watches the trajectory of the system and enters "Bad" locations whenever the observed behavior violates the safety condition.

Monitor for requirement R₂

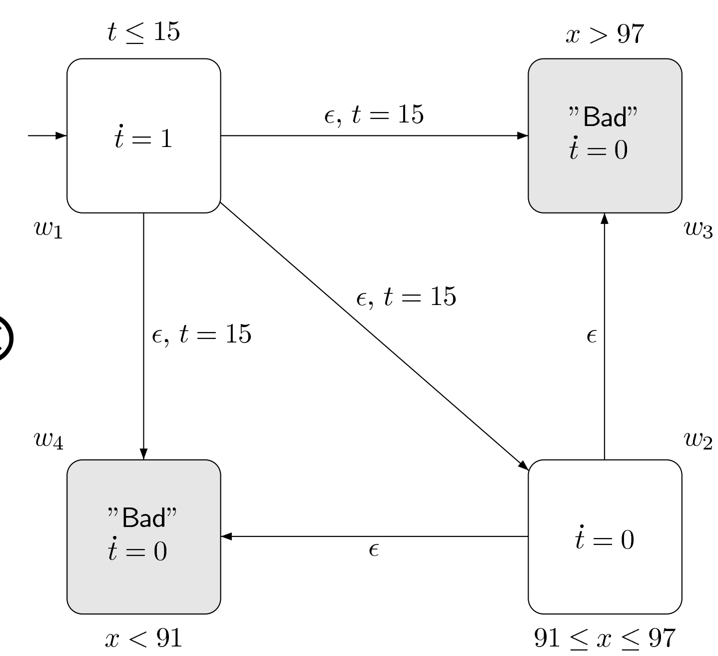


From safety to monitors and reachability

- To verify R_2 on our system, we consider the product of the monitor with the system i.e.,

$$\text{Tank} \otimes \text{Burner} \otimes \text{Thermo} \otimes \text{Controller} \otimes \text{Moni}_2$$

- Then we check for the reachability of the region that contains the states in which Moni_2 is in location w_3 or w_4 (the locations labelled with "Bad").

Tank⊗Burner⊗Thermo⊗Controller⊗ 

How do we solve reachability problems ?

- Direct successor operator $Post^T : 2^S \rightarrow 2^S$:

  $Post^T(S')$
  $= \{\ s \in S \mid \exists s' \in S' \bullet (\exists \sigma \in \Sigma : (s', \sigma, s) \in \rightarrow) \vee (\exists \delta \in \mathbb{R}^{\geq 0} : (s', \delta, s) \in \rightarrow)\ \}$

- Direct predecessor operator $Pre^T : 2^S \rightarrow 2^S$:

  $Pre^T(S')$
  $= \{\ s \in S \mid \exists s' \in S' \bullet (\exists \sigma \in \Sigma : (s, \sigma, s') \in \rightarrow) \vee (\exists \delta \in \mathbb{R}^{\geq 0} : (s, \delta, s') \in \rightarrow)\ \}$

How do we solve reachability problems ?

- The set of reachable states of a HA H with TTS ⟦H⟧ is defined by the least solution of the following equation:

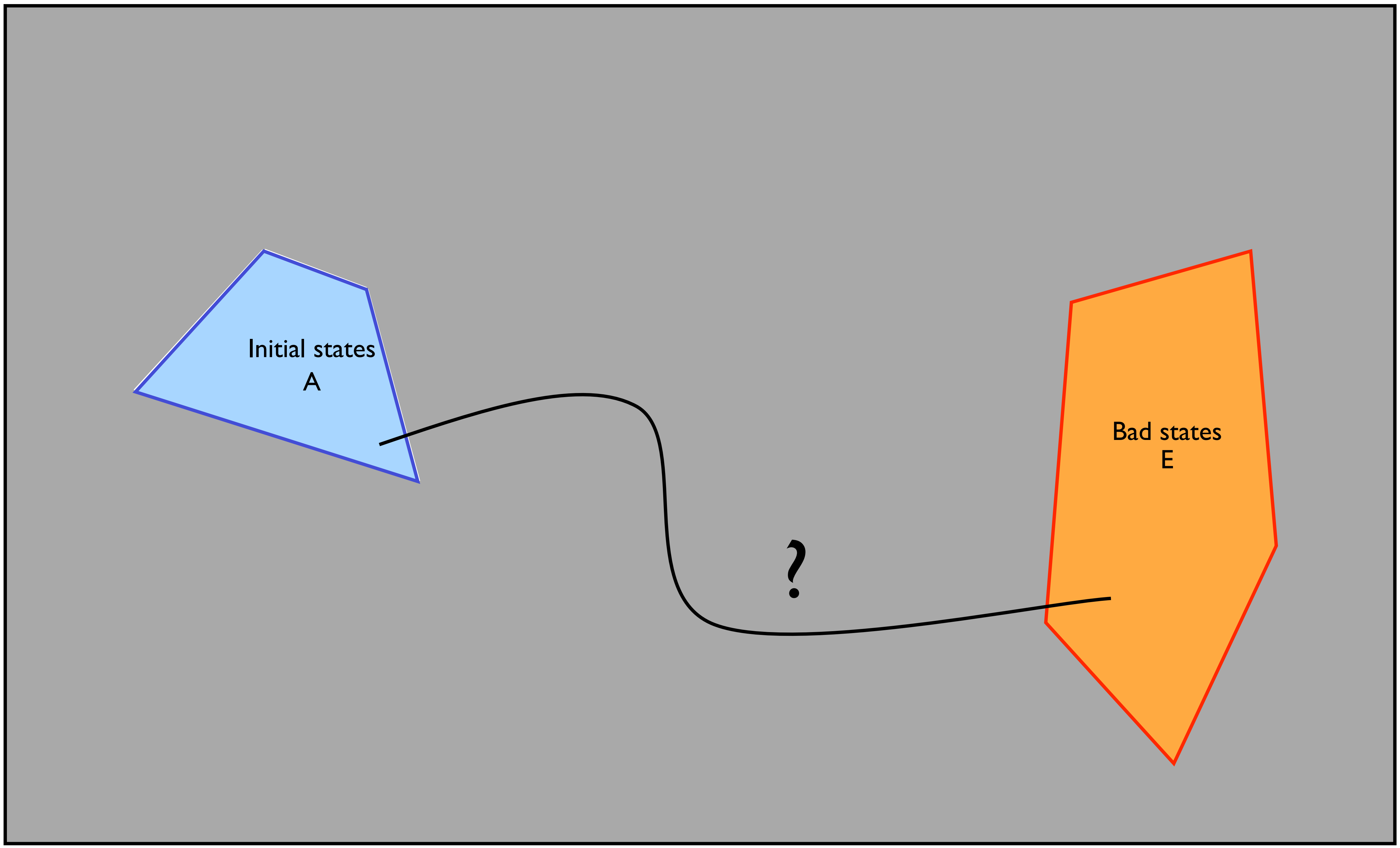
$$X=(S_0 \cup Post^{[\![H]\!]}(X))$$

where X ranges over sets of states.

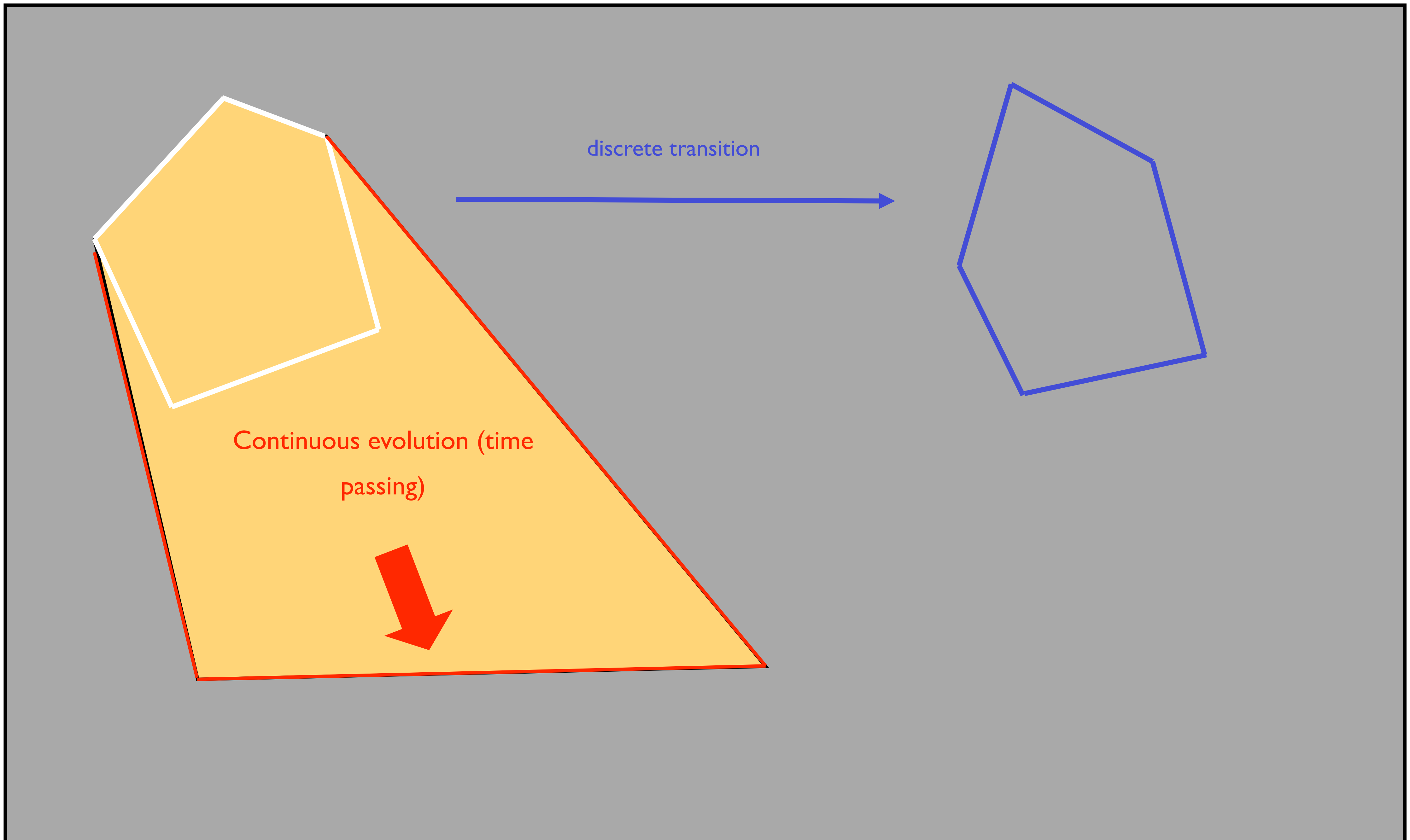
- Symmetrically, the set of states that can reach R is defined by the least solution of the following equation:

$$X=(R \cup Pre^{[\![H]\!]}(X))$$

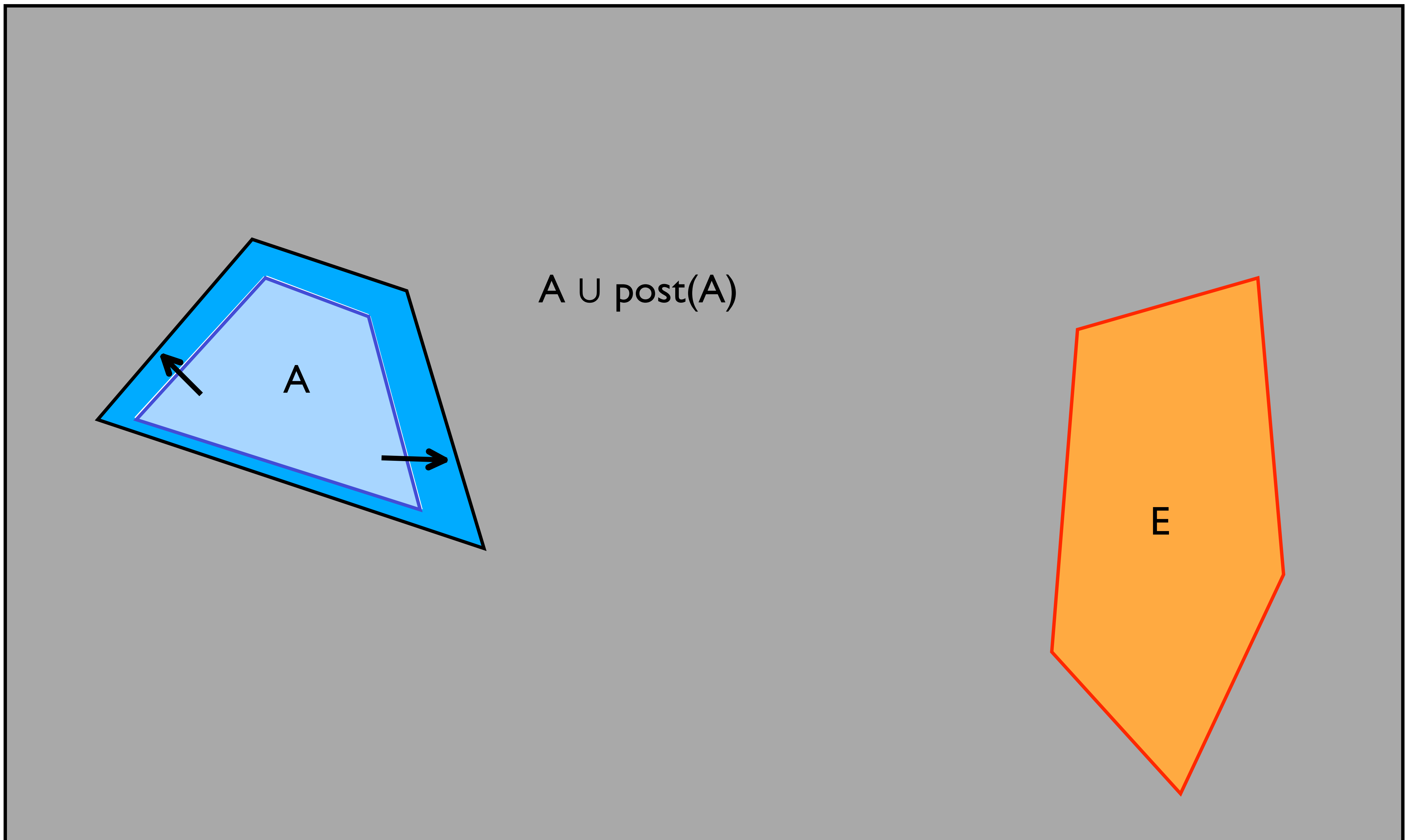
The reachability problem



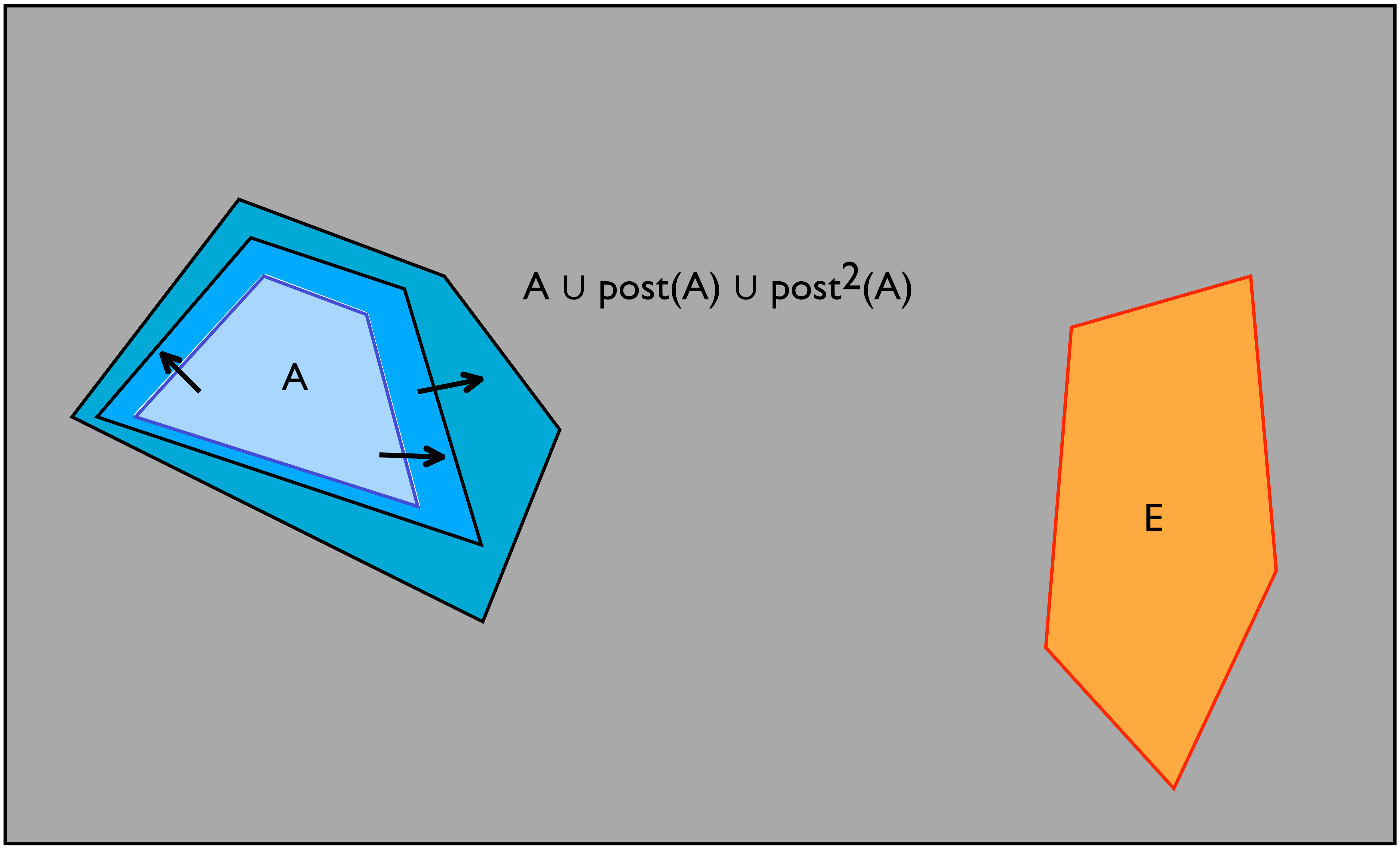
Post operator



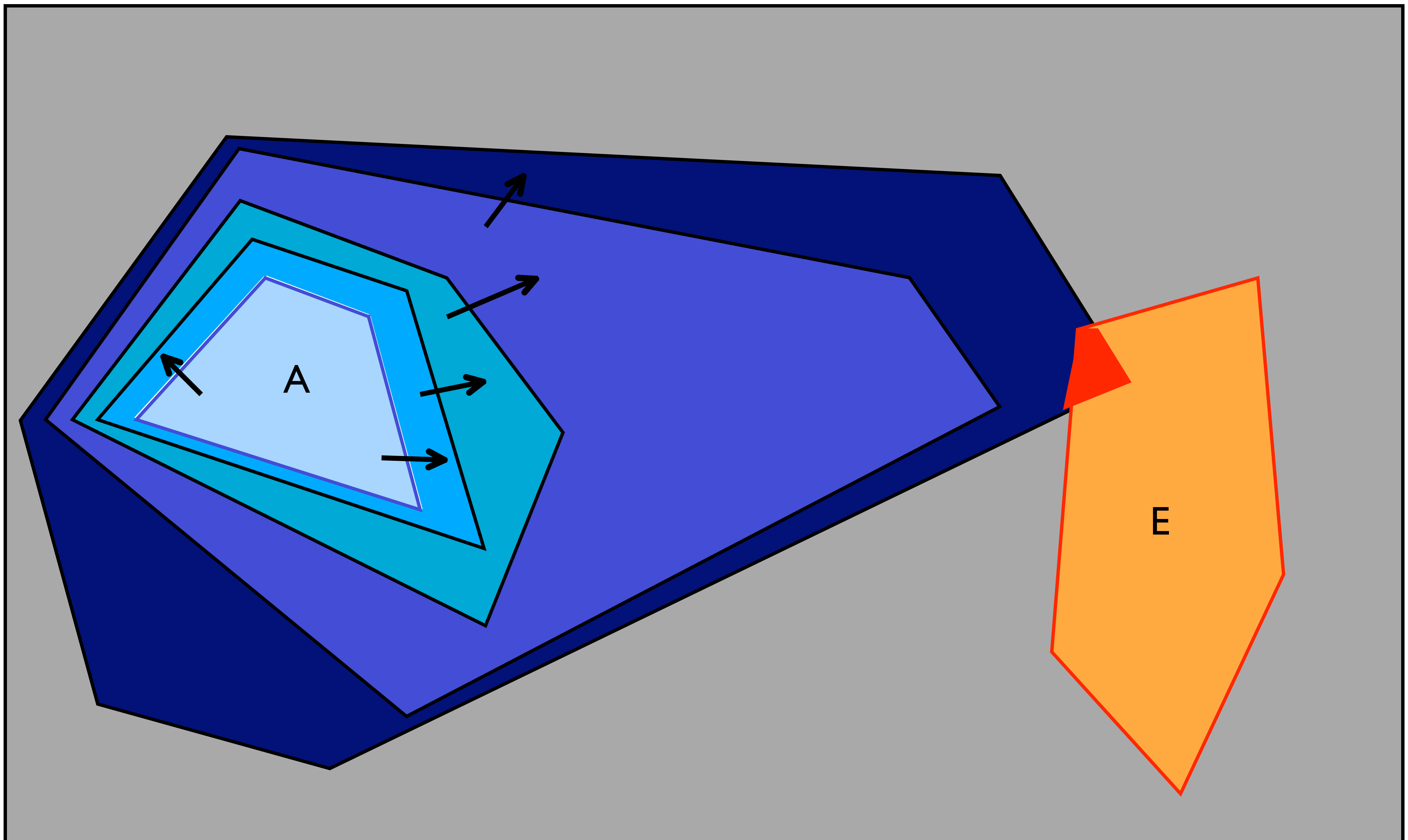
Iteration of the Post operator



Iteration of the Post operator



Iteration of the Post operator



Undecidability - Non representability

- Computing solutions of the fixed point equations (forward or the backward approach) is often difficult.

  Convergence in a finite number of approximation steps is not guaranteed ...

- ... furthermore, even one step of computation may not be feasible, as we can not solve general differential equations/inclusions ...

- ... there are also representability issues: how to represent the set of successors of a region ? This set may not have a symbolic representation.

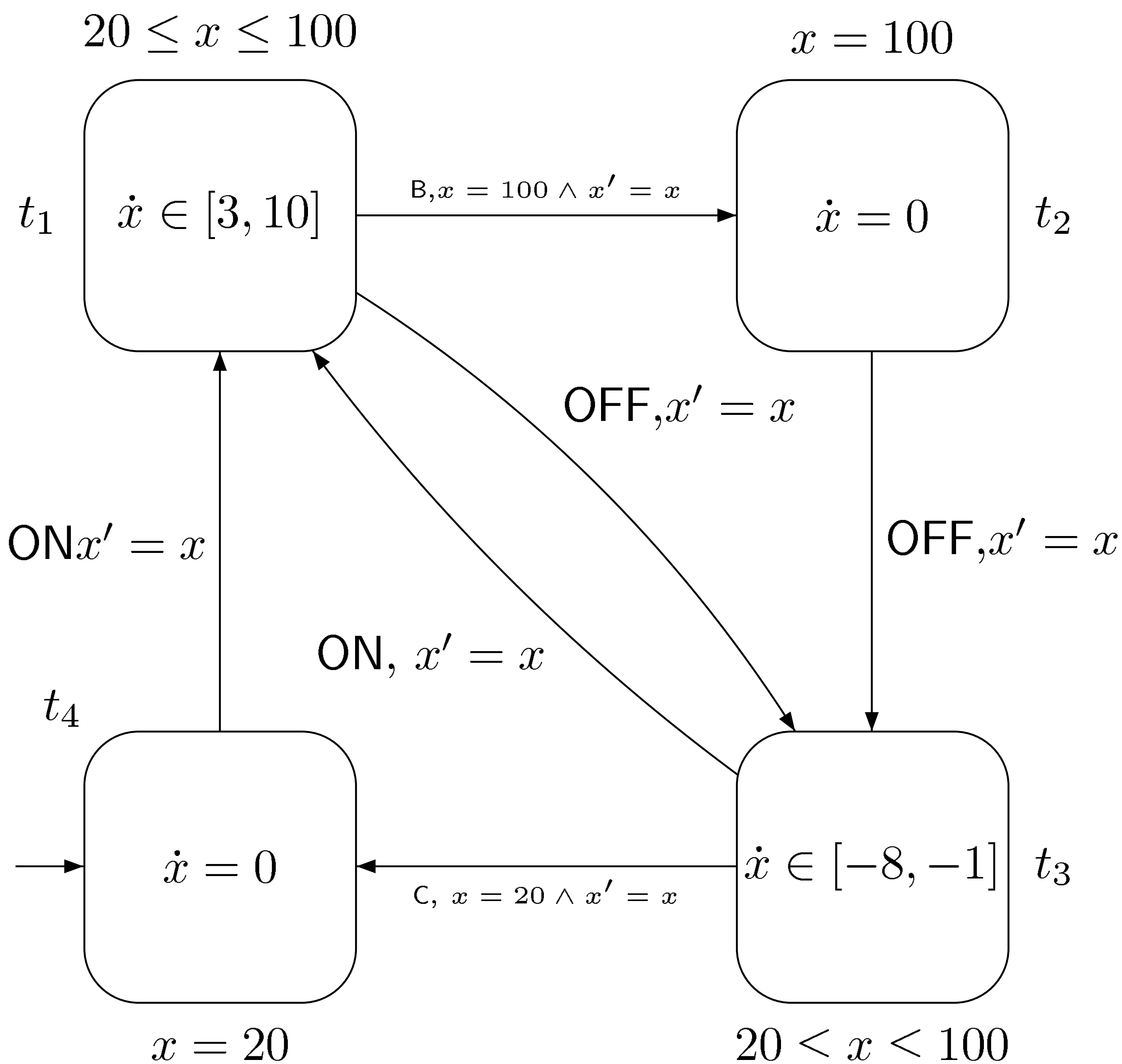
- Furthermore, even very restricted subclasses of hybrid automata have an undecidable reachability problem.

Rectangular hybrid automata

Rectangular Hybrid Automata

- Rectangular automata are a subclass of hybrid automata where dynamics are constraints by rectangular constraints and updates are restricted by rectangular updates;
- A interval is a convex non-empty subset of the positive real-numbers with rational bounds;
- $\mathrm{Rect}(X) \ni \Phi_1,\Phi_2 := \bot \mid \top \mid x \in I \mid \Phi_1 \wedge \Phi_2$
- $\mathrm{UpdateRect}(X) \ni \Phi_1,\Phi_2$
  $$:= \bot \mid \top \mid x \in I \mid x' \in I \mid x' = x \mid \Phi_1 \wedge \Phi_2$$
- A rectangular HA is an HA where Init(·) and Inv(·) are constraints in Rect(X), Jump(·) in UpdateRect(X), and Flow(·) in $\mathrm{Rect}(\dot{X})$.

Example of a rectangular automaton



Semi-algorithms for rectangular hybrid automata

Effective procedure for Post in RHA

- A linear term over X is a linear combination of the variables in X with integer coefficients.

  ex : $3x+2y-1$.

- A linear formula over X is a boolean combination of inequalities between linear terms over X.

  ex : $3x+2y-1 \geq 0 \wedge y \geq 5$.

- Given a linear formula ψ, we write $[\![\psi]\!]$ for the set of valuations v such that $v \models \psi$.

Effective procedure for Post in RHA

- If we allow quantifiers with linear formulas we obtain the theory of reals with addition $T(\mathbb{R},0,1,+,\leq)$.

  This theory allows for quantifier elimination.

  ex : "$\forall y \bullet y \geq 5 \rightarrow x+y \geq 7$" is equivalent to "$x \geq 2$".

- A symbolic region of H is a finite set

$$\{ (l,\psi_l) \mid l \in Loc \} \text{ where } [\![\psi_l]\!] \subseteq [\![Inv(l)]\!].$$

Effective procedure for Post in RHA

Given a location $l \in Loc$ and a set of valuations $V \subseteq [X \rightarrow \mathbb{R}]$ such that $V \subseteq Inv(l)$, the forward time closure, noted $\langle V \rangle_l^{\nearrow}$ is the set of valuations of variables in X that are reachable from some valuation $v \in V$ by letting time pass.

This set is defined as follows:

$\langle V \rangle_l^{\nearrow}$ is the set of valuation $v' \in [X \rightarrow \mathbb{R}]$ such that

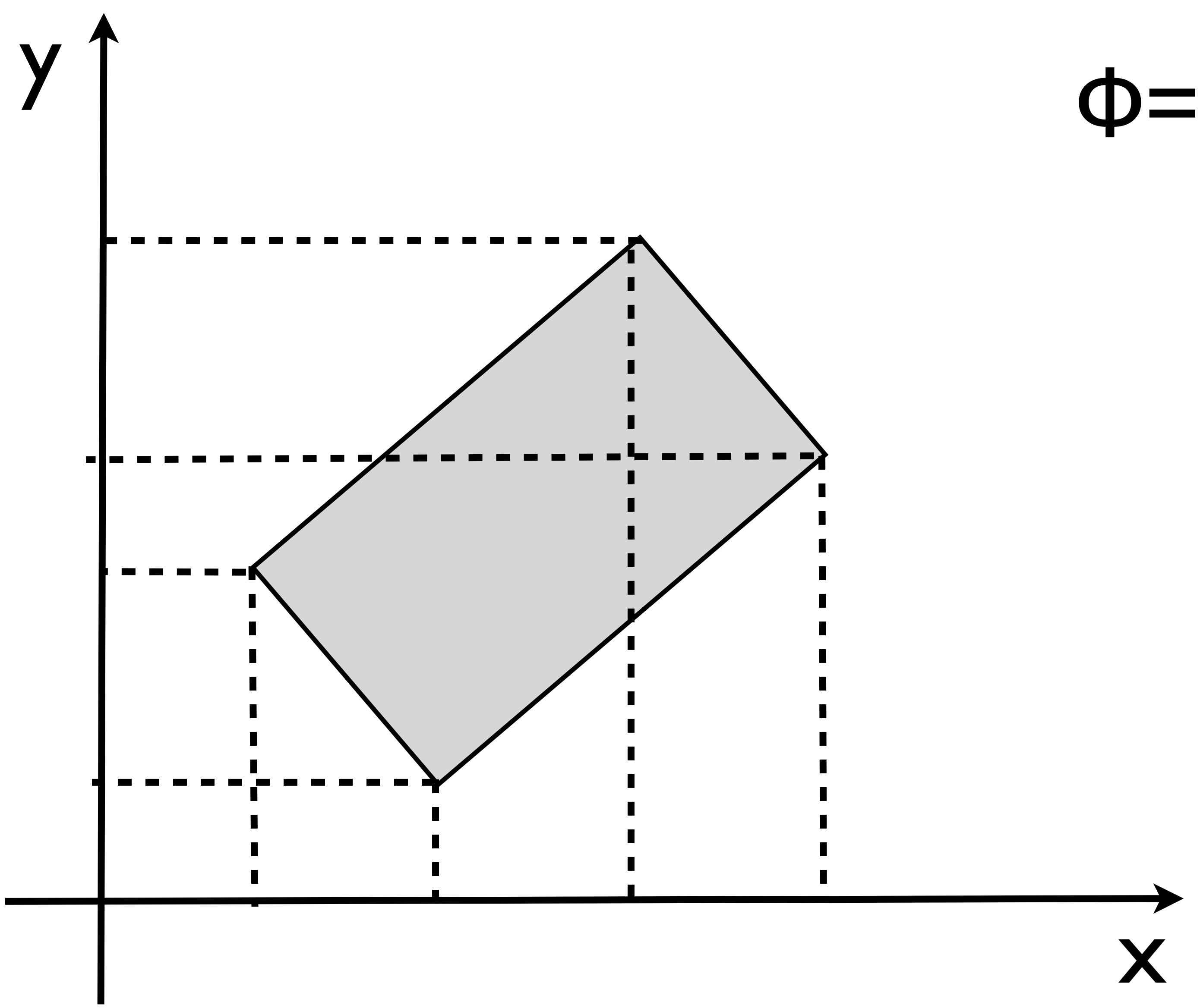
$\exists v \in V \bullet \exists t \in \mathbb{R}_{\geq 0} \bullet \forall x \in X \bullet$

$v(x)+t \times \mathbf{Inf}([\![Flow(l)]\!](x)) \leq v'(x) \leq v(x)+t \times \mathbf{Sup}([\![Flow(l)]\!](x))$
and $v'(x) \in [\![Inv(l)]\!]$.

As quantifiers can be eliminated, the resulting formula is a boolean combination of linear constraints.

An example of time elapsing

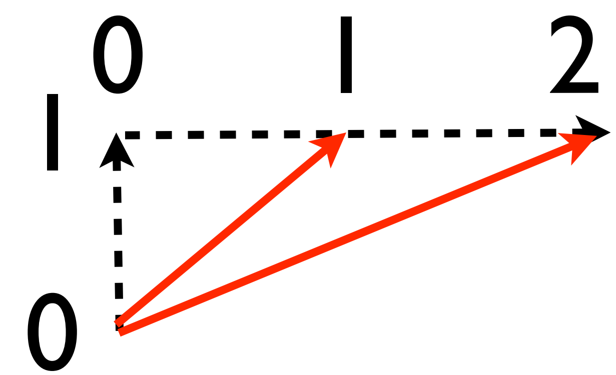
Assume $\dot{x}=[1,2]$ and $\dot{y}=1$

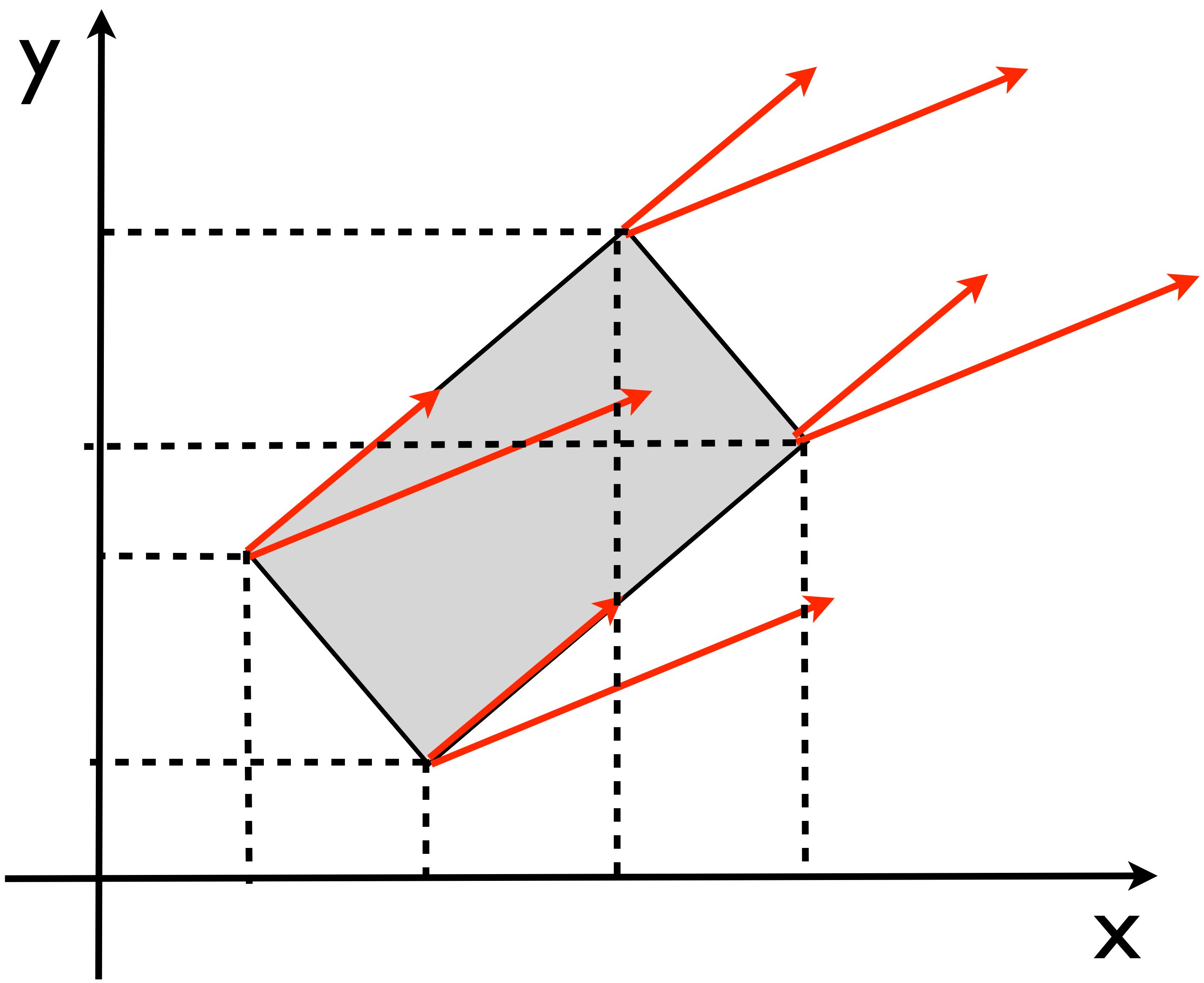


$$\Phi=\{\ (x,y)\ |\quad x\in[1,4] \\ \wedge\ y\in[1,6] \\ \wedge\ y\geq -2x+5\ \wedge\ \ldots\ \}$$

An example of time elapsing

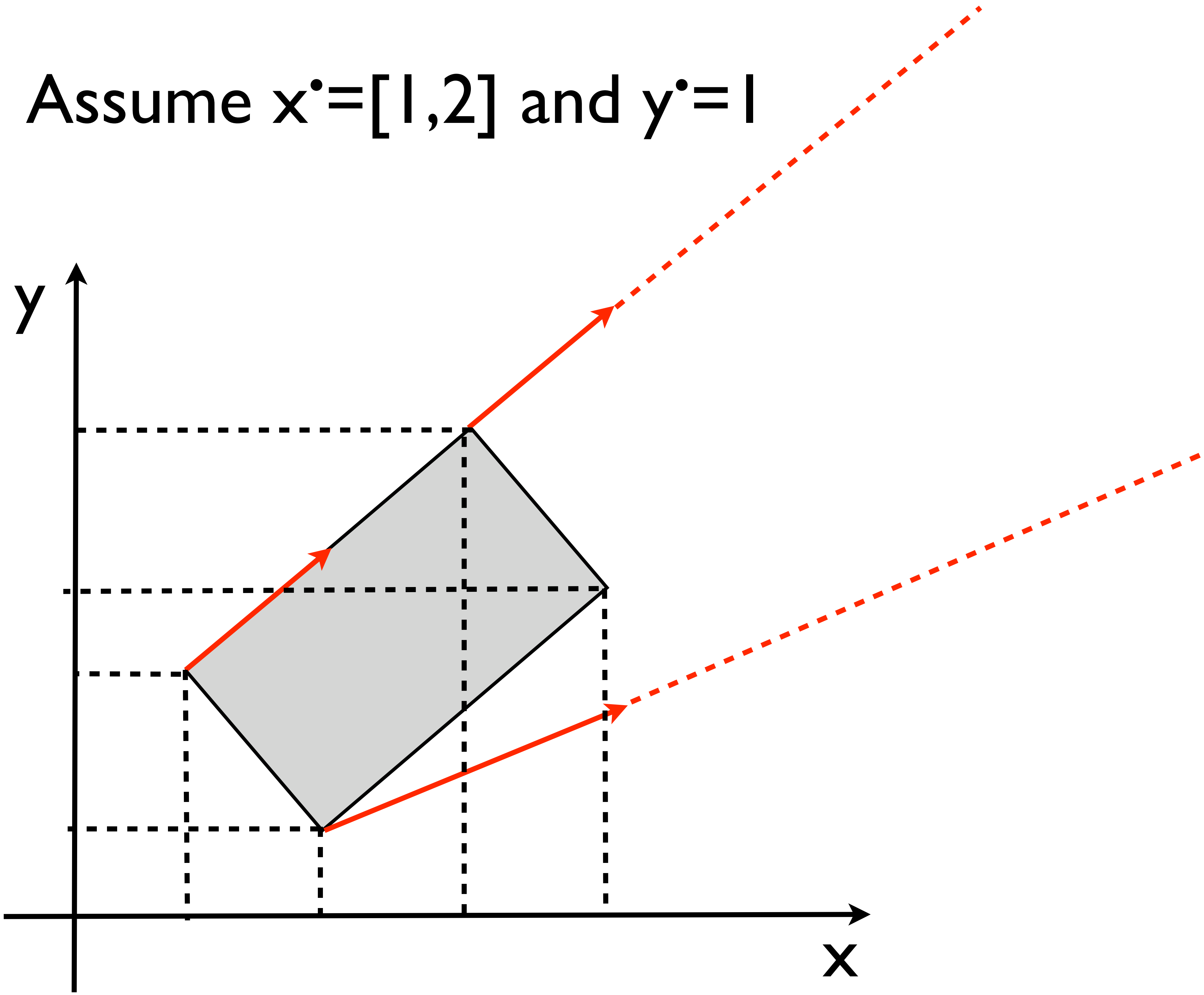
Assume $x^{\bullet}=[1,2]$ and $y^{\bullet}=1$





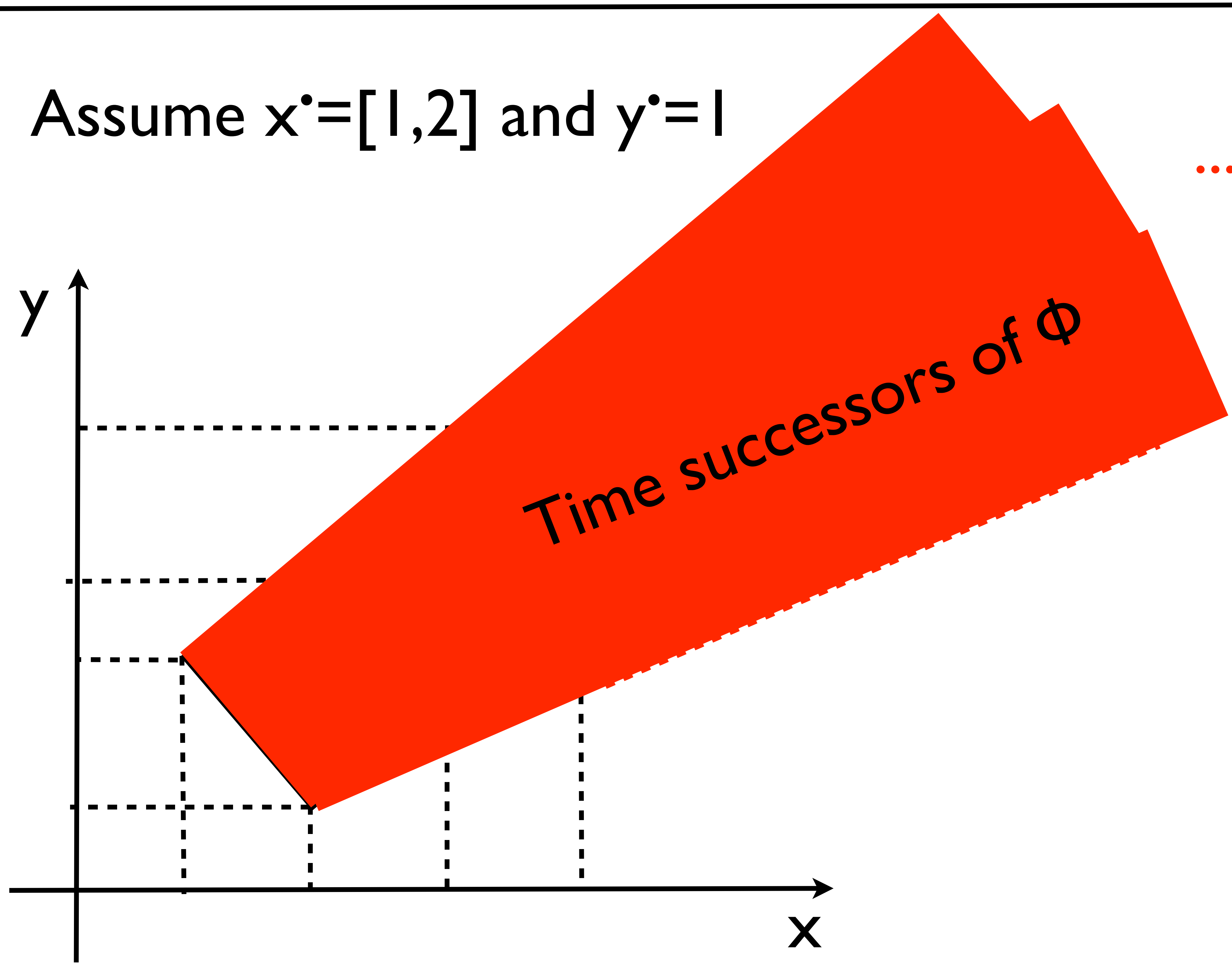
An example of time elapsing

Assume $x^{\bullet}=[1,2]$ and $y^{\bullet}=1$



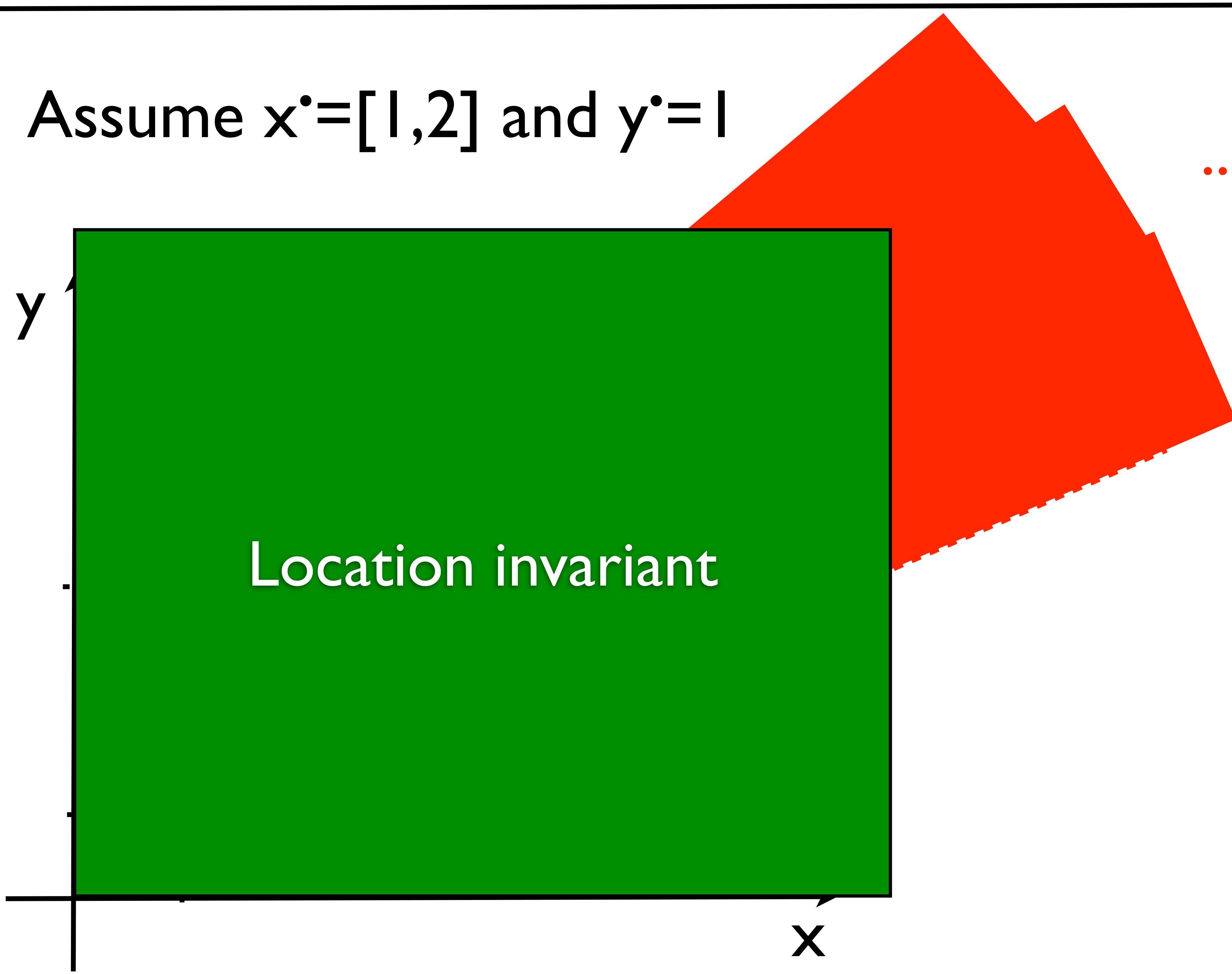
An example of time elapsing

Assume $x^{\cdot}=[1,2]$ and $y^{\cdot}=1$



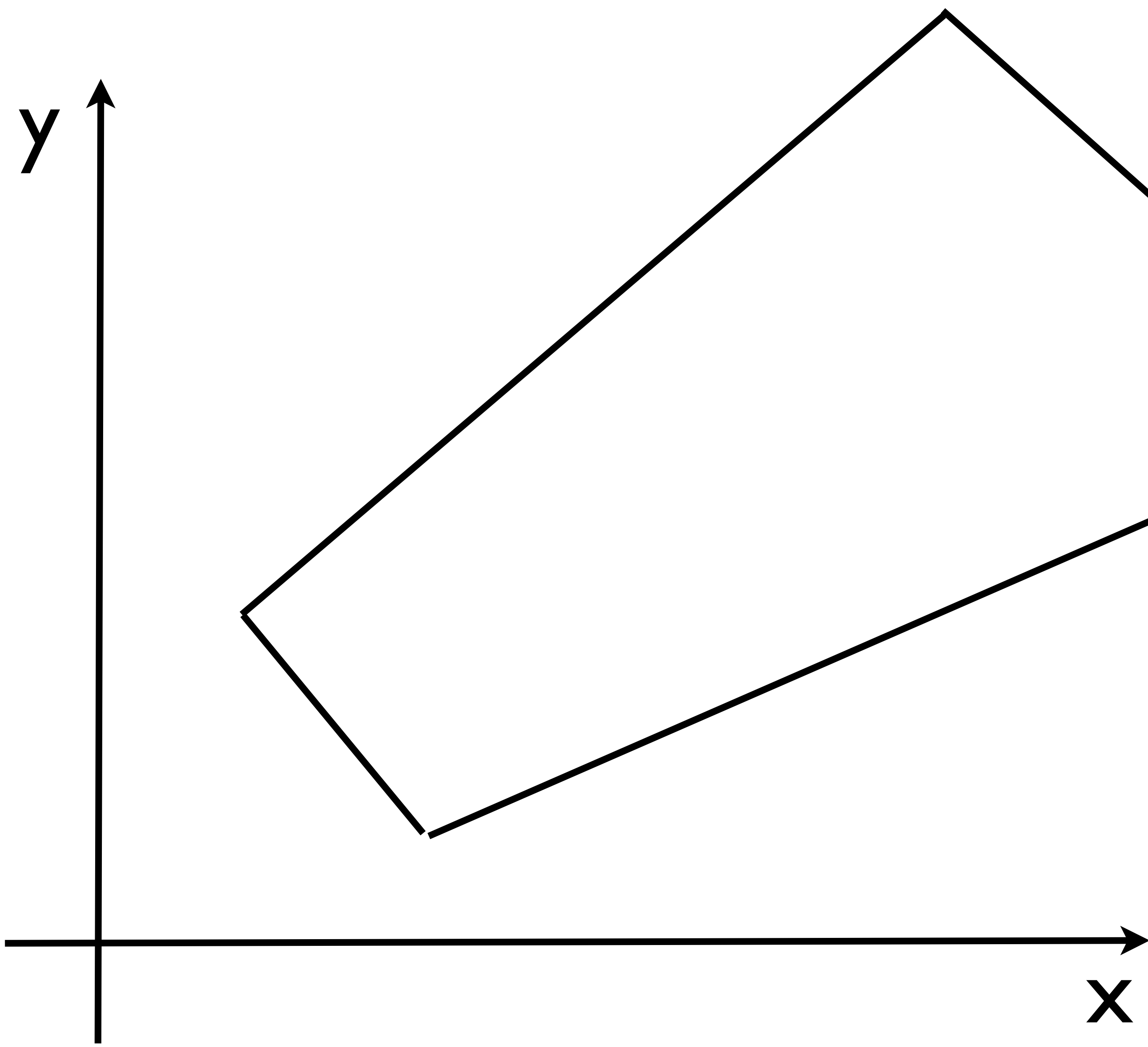
An example of time elapsing

Assume $\dot{x}=[1,2]$ and $\dot{y}=1$

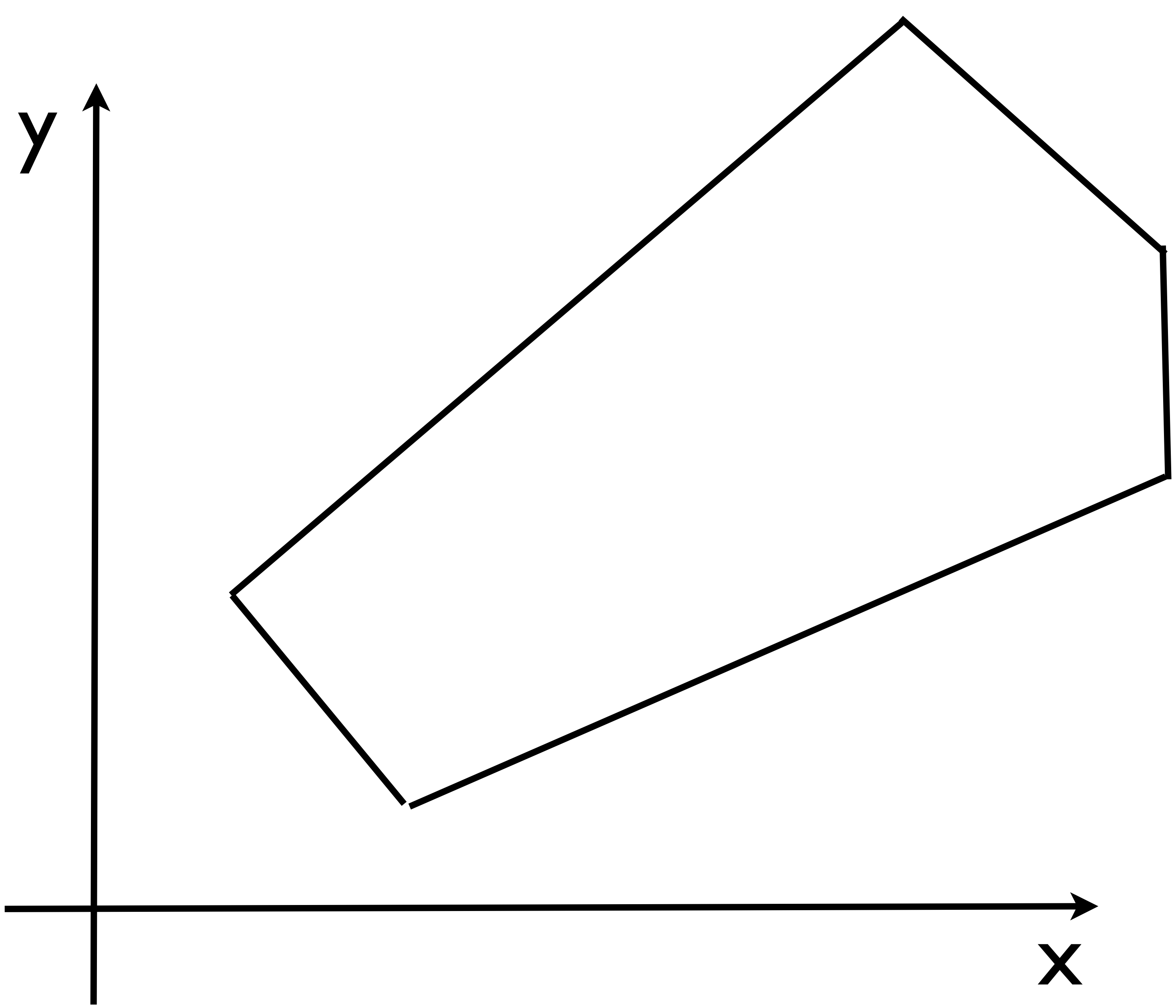


An example of time elapsing

Assume $x^{\bullet}=[1,2]$ and $y^{\bullet}=1$

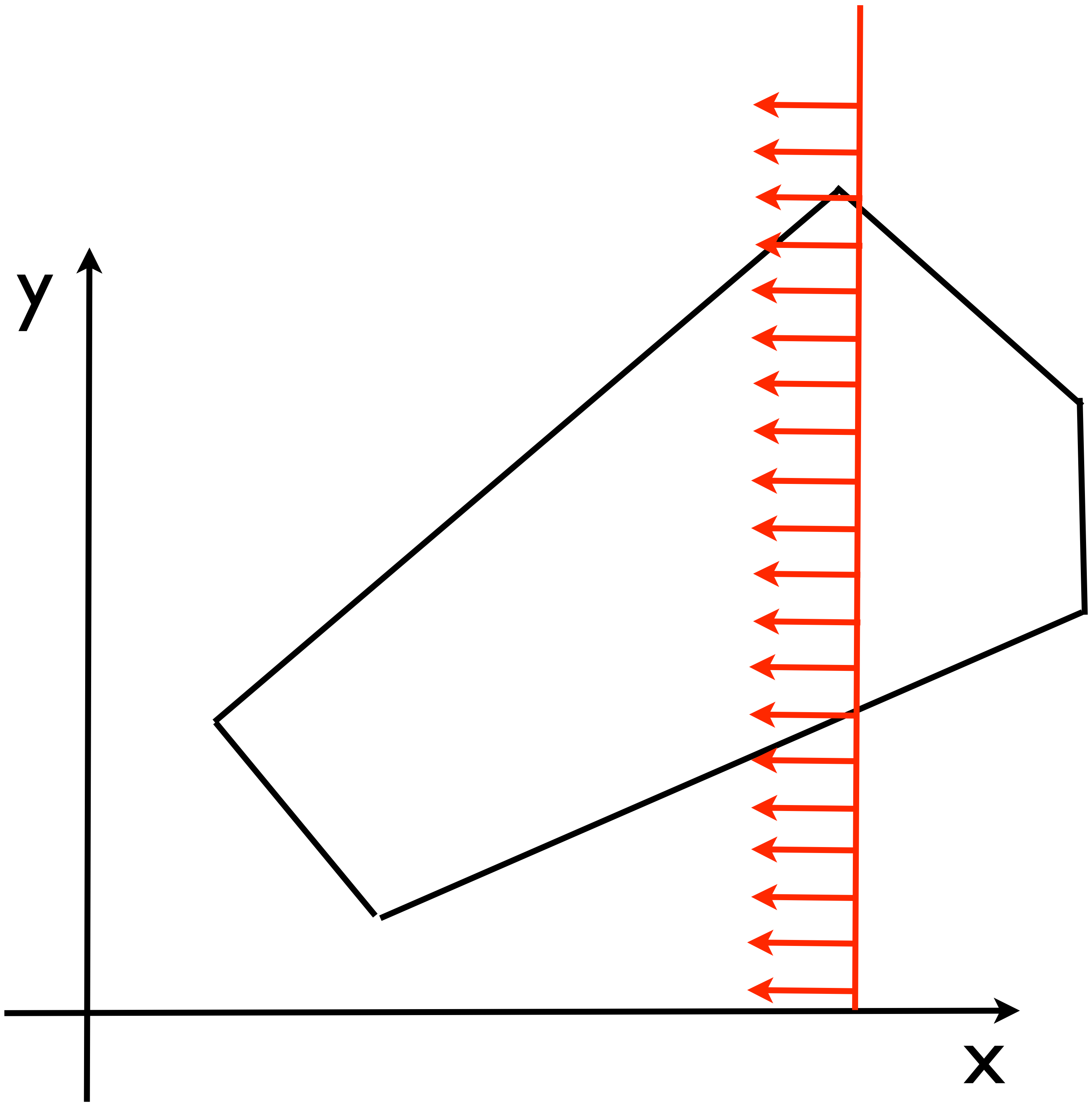


An example of discrete step



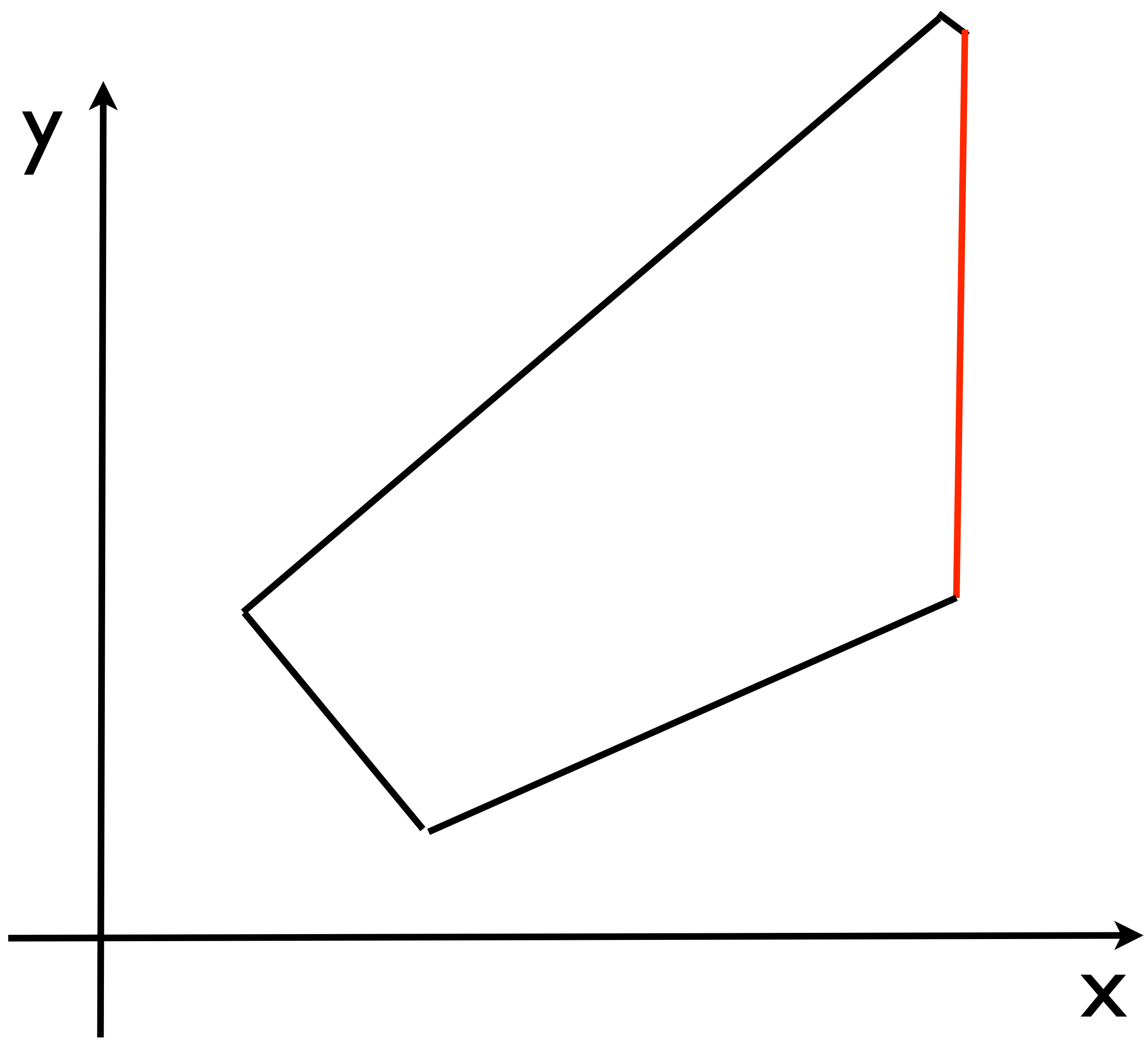
Assume a transition with guard $x \leq 5$ and reset of y to zero.

An example of discrete step



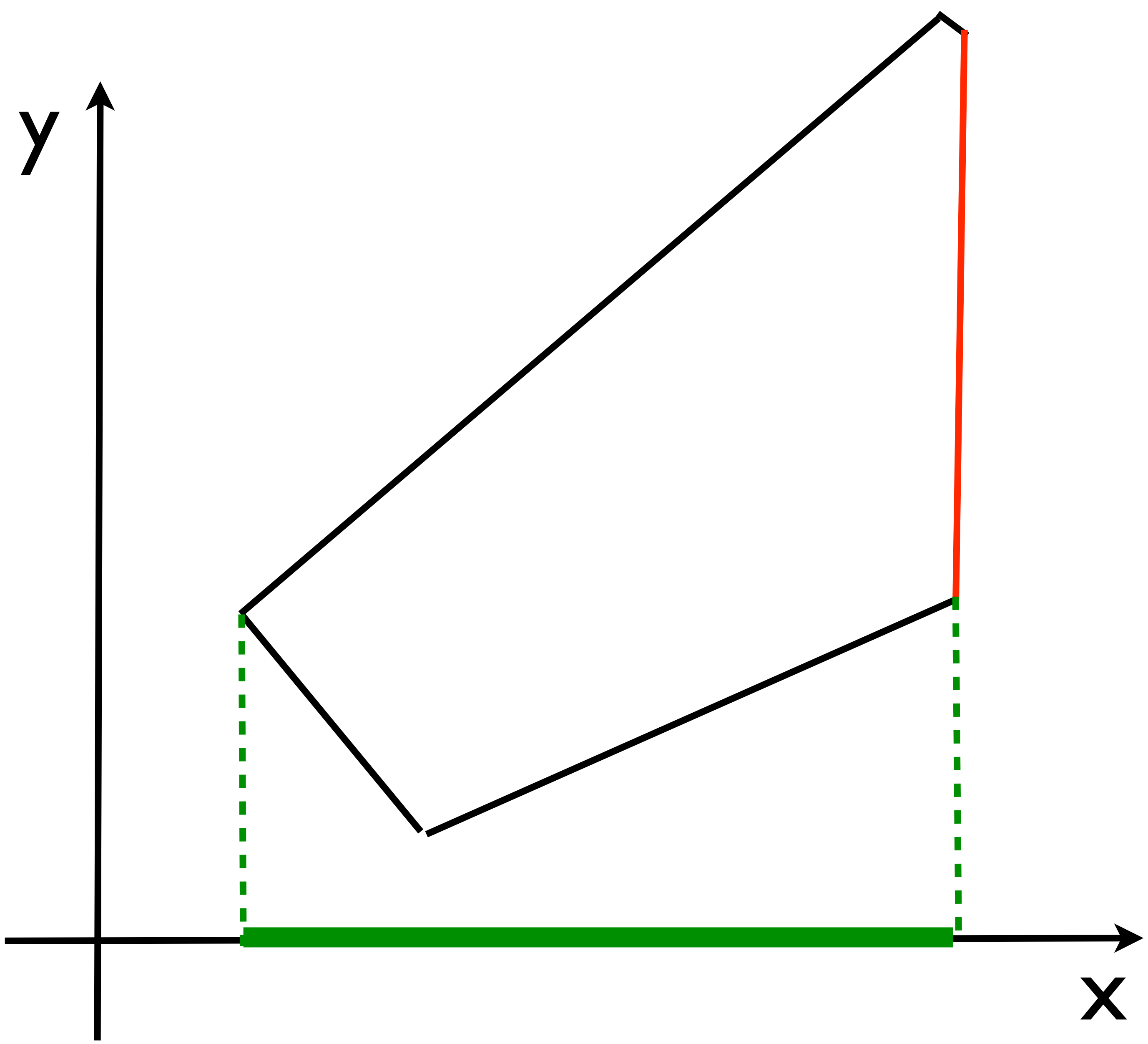
Assume a transition with guard $x \leq 5$ and reset of y to zero.

An example of discrete step



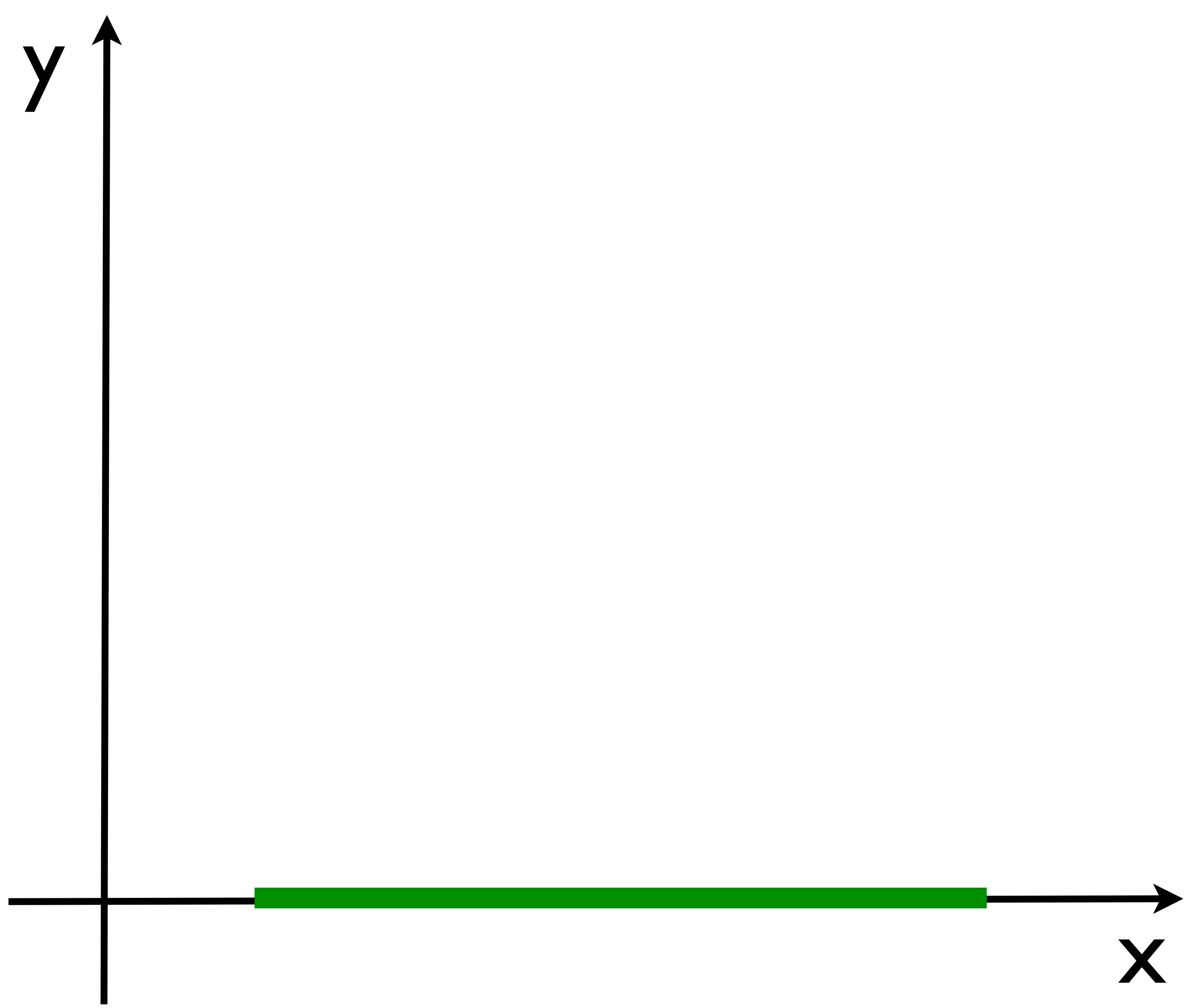
Assume a transition with guard $x \leq 5$ and reset of y to zero.

An example of discrete step



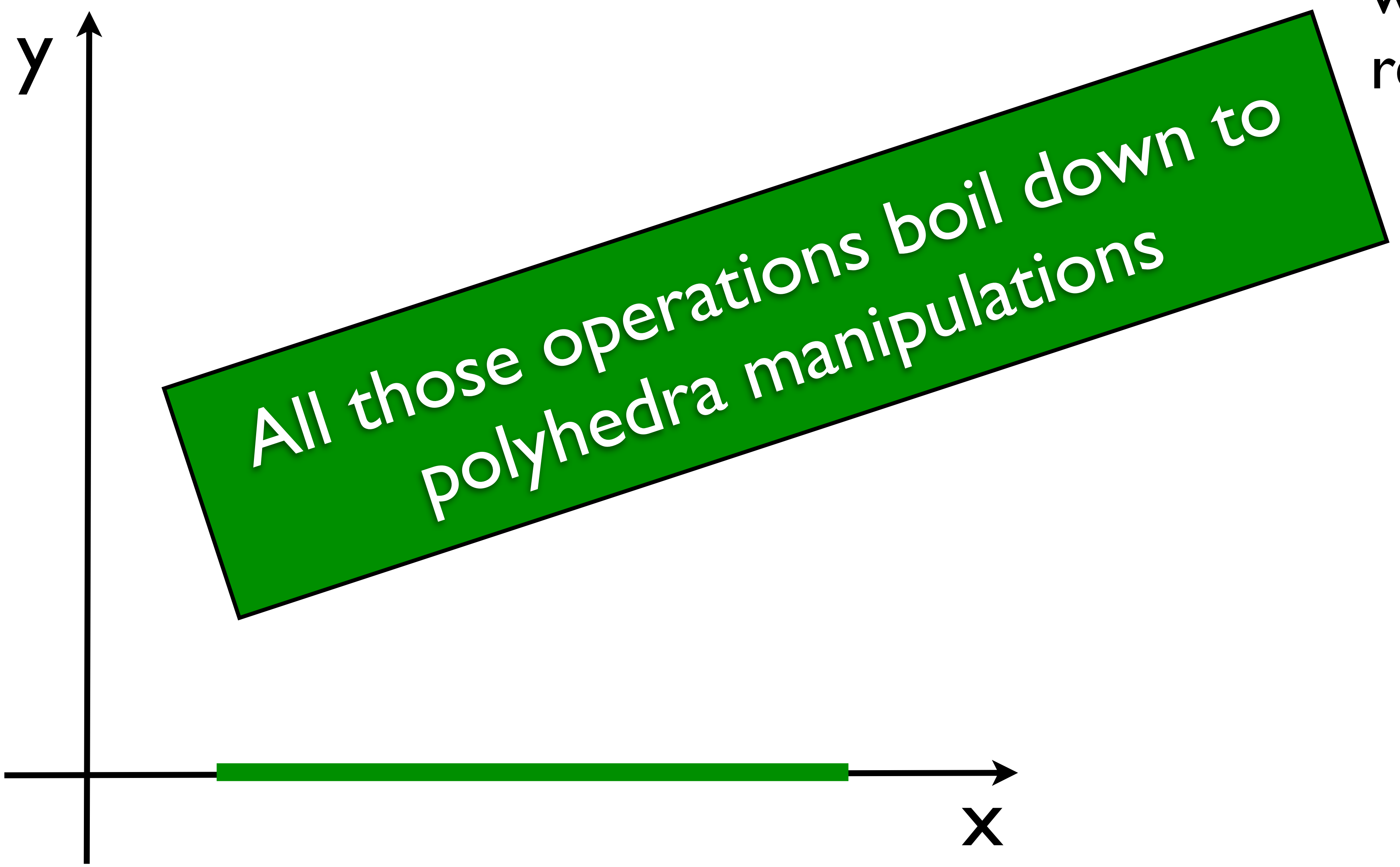
Assume a transition with guard $x \leq 5$ and reset of y to zero.

An example of discrete step



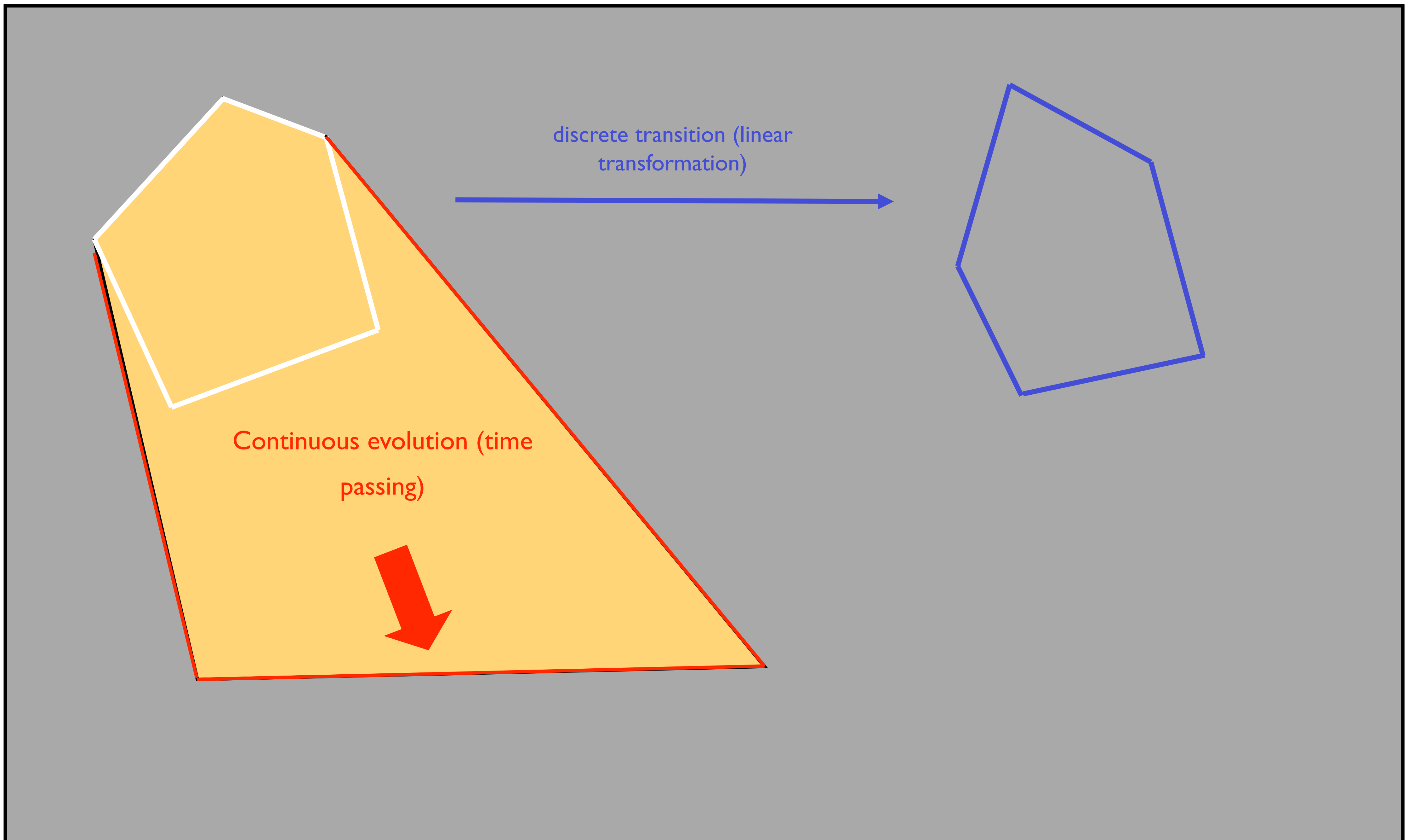
Assume a transition with guard $x \leq 5$ and reset of y to zero.

An example of discrete step

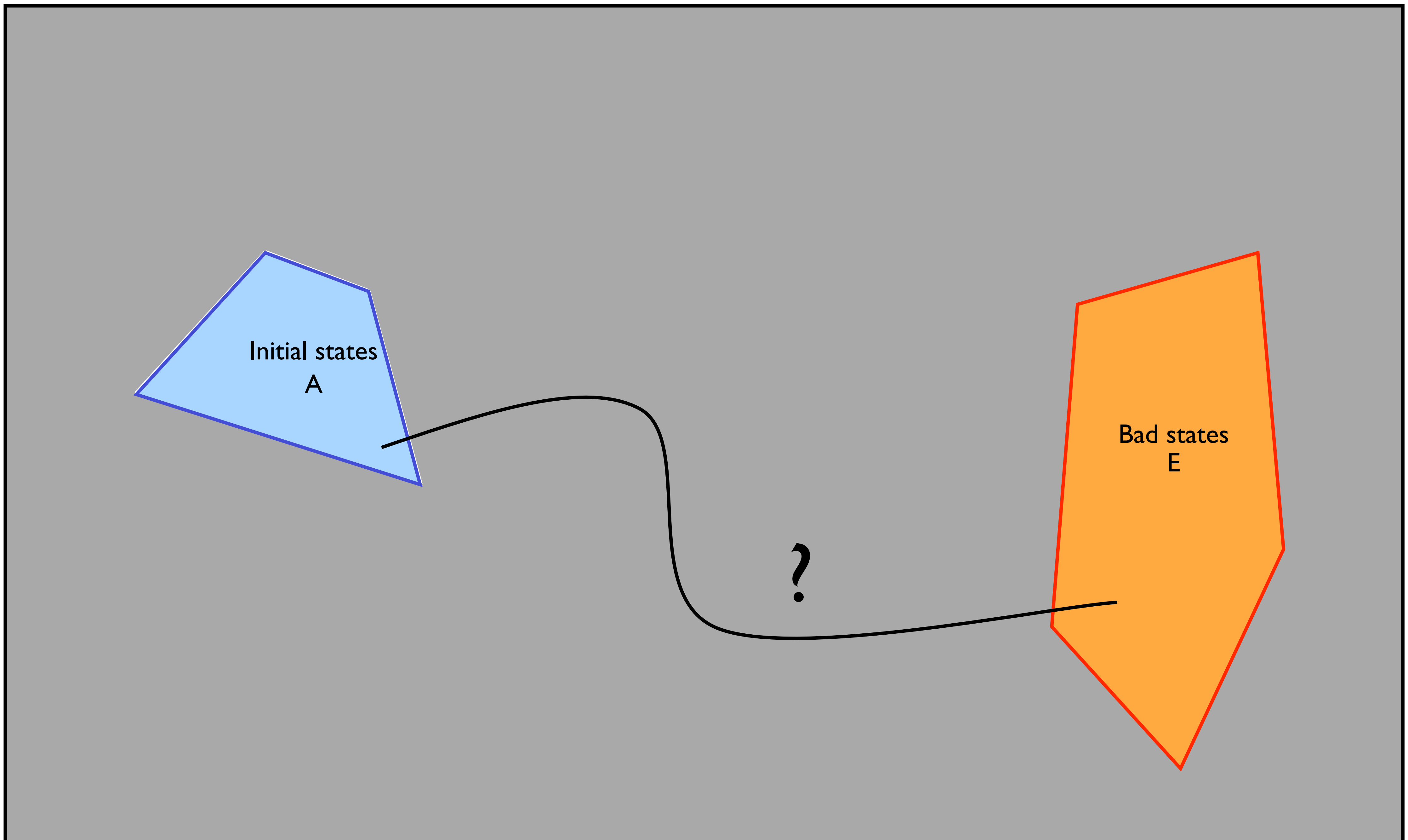


Assume a transition with guard $x \leq 5$ and reset of y to zero.

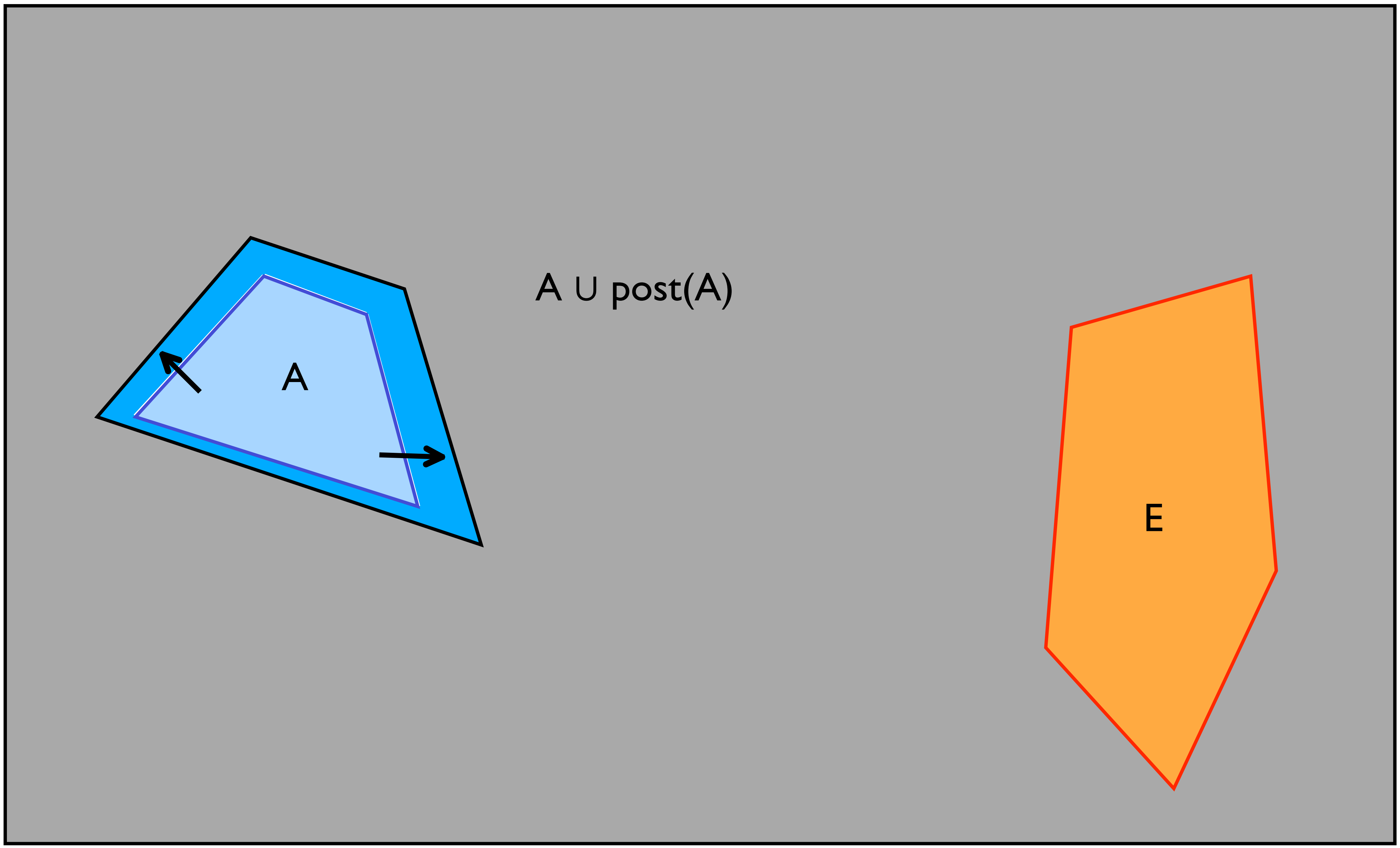
Forward reachability analysis



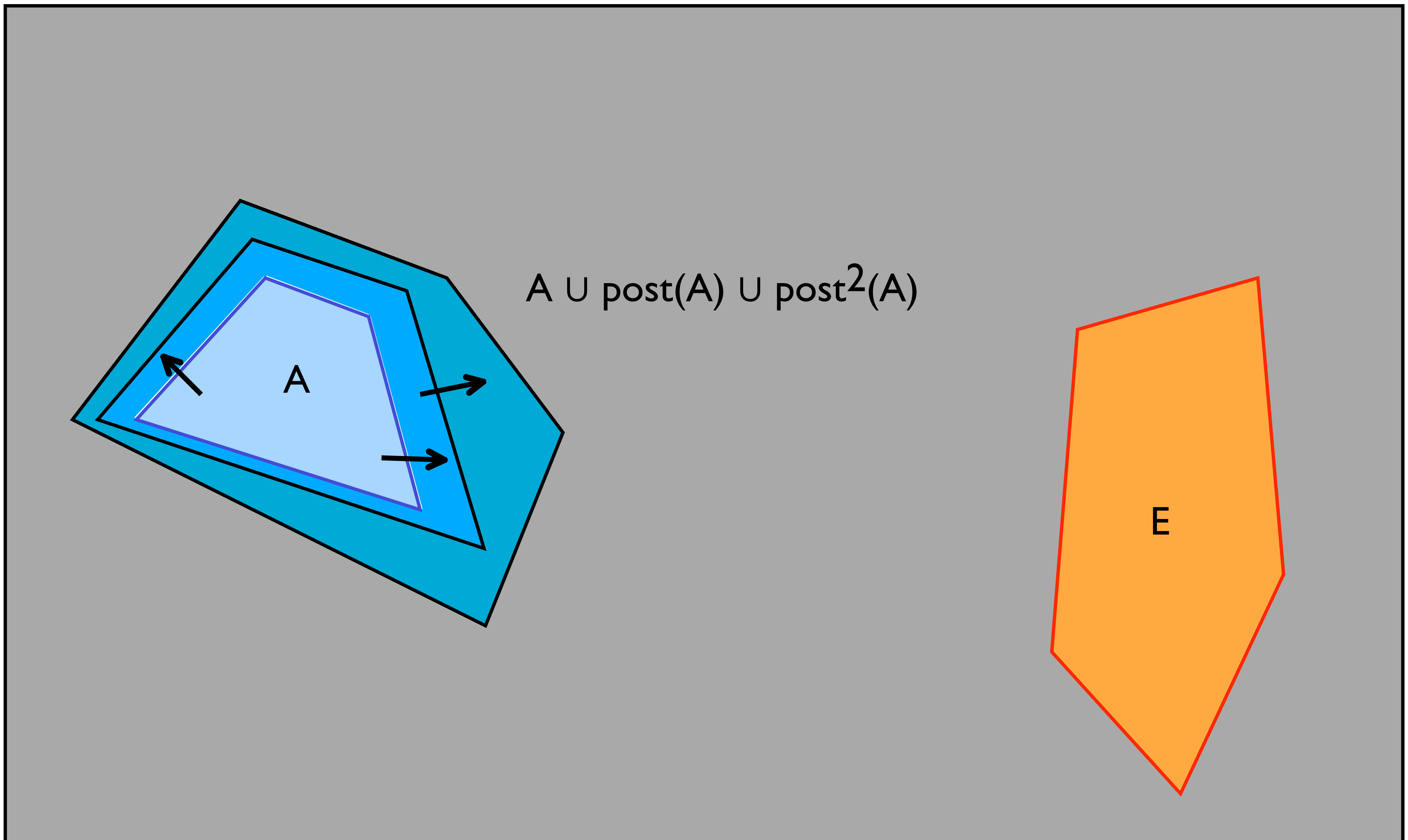
Forward reachability analysis



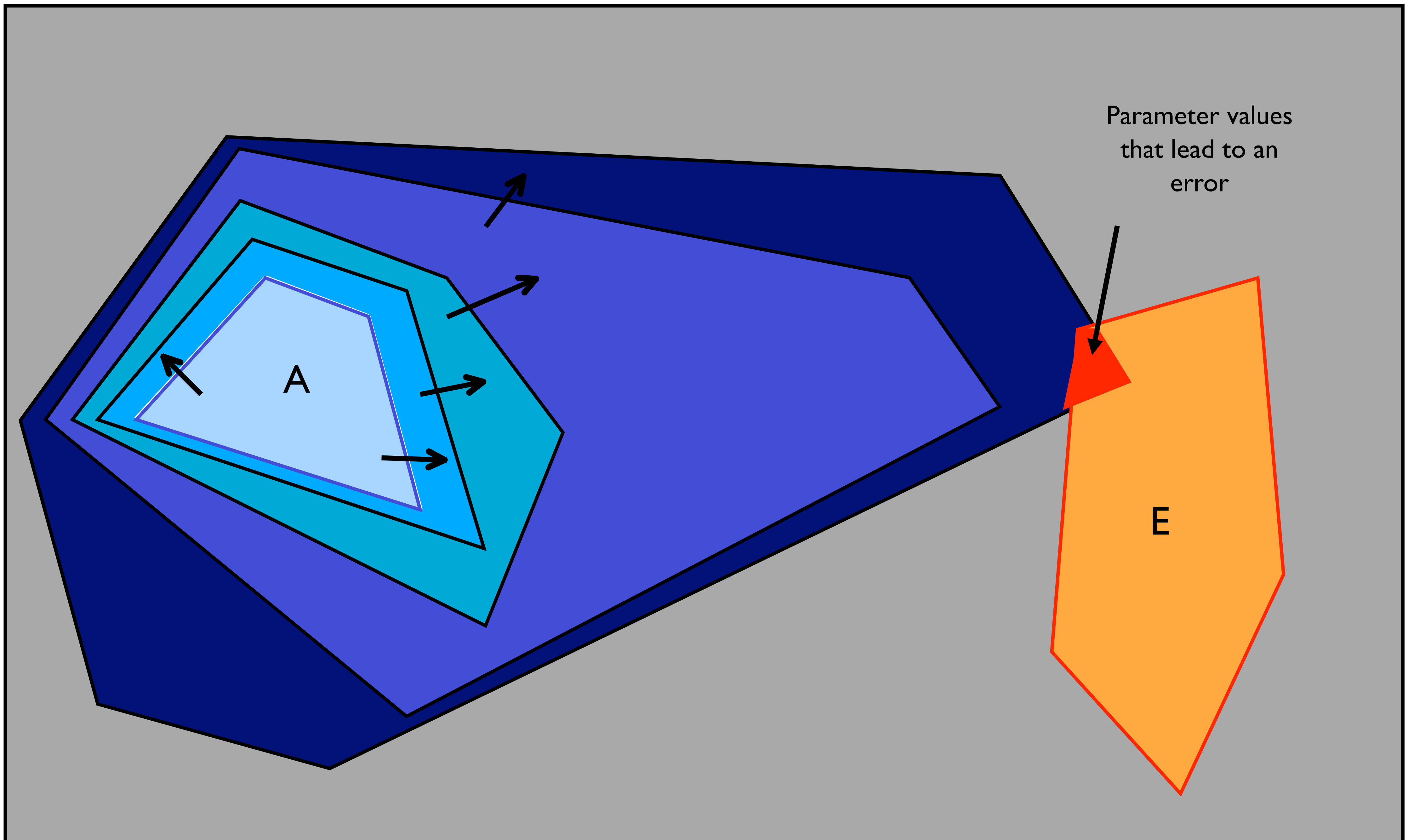
Forward reachability analysis



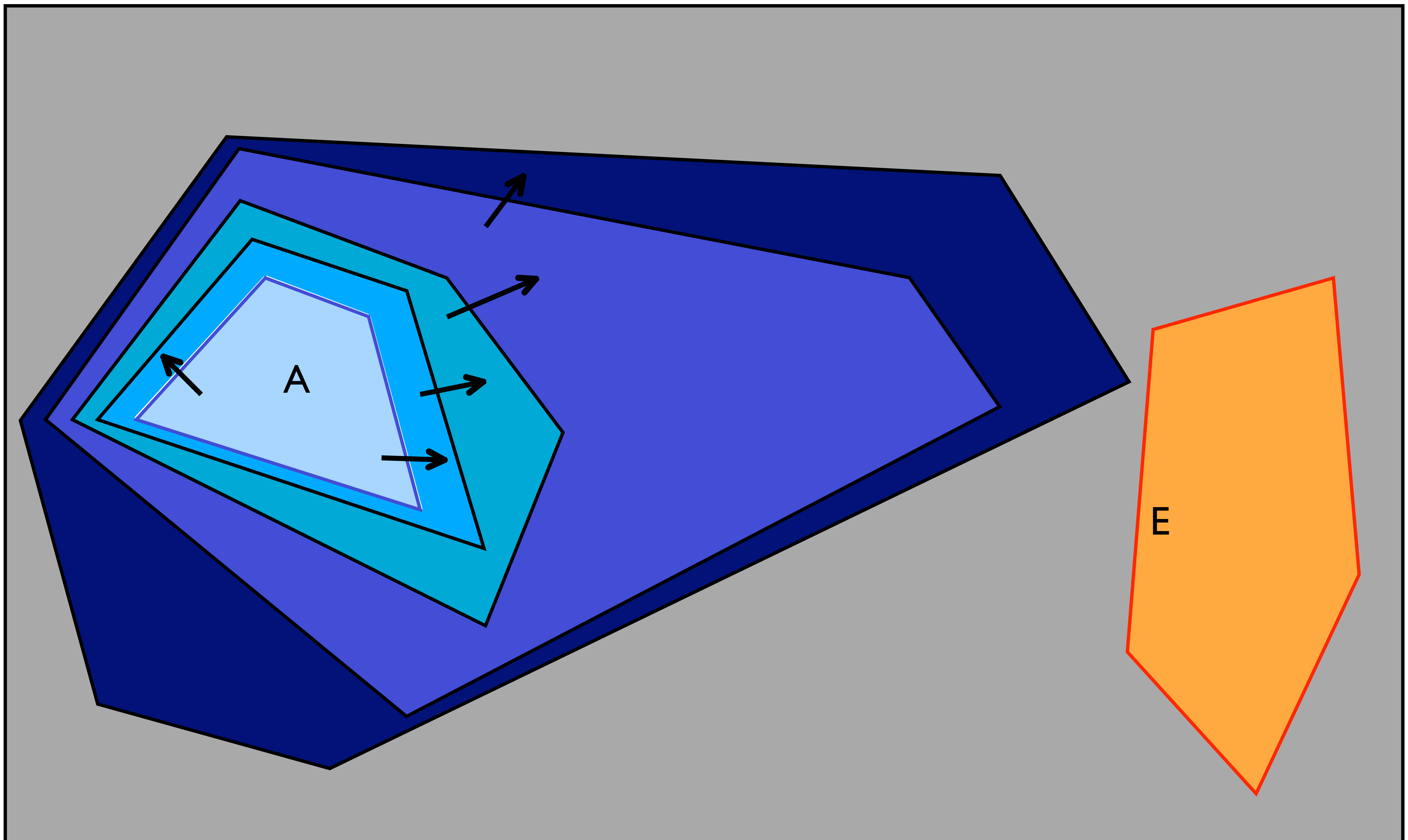
Forward reachability analysis



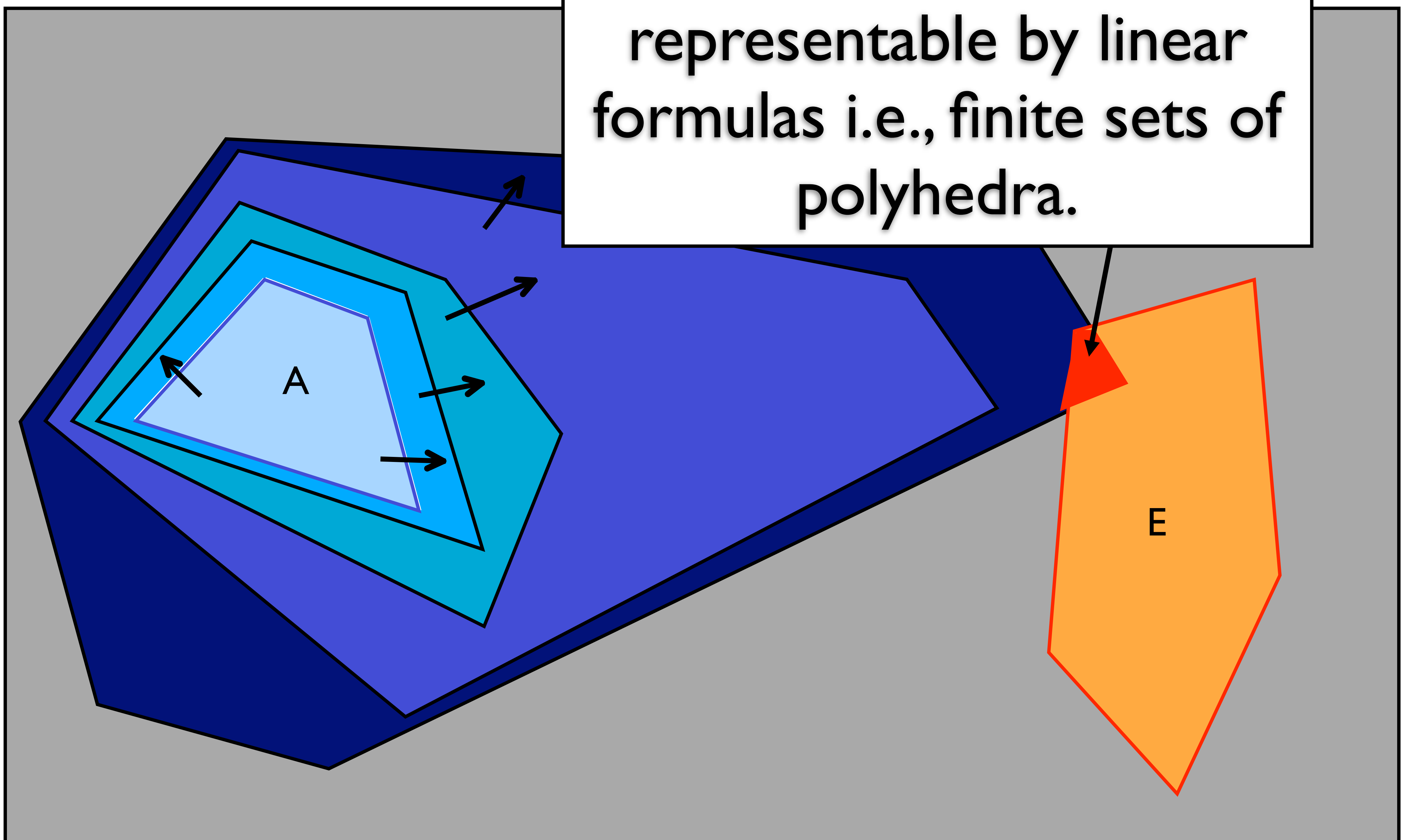
Forward reachability analysis



Forward reachability analysis



Forward reachability analysis



Here, all sets are representable by linear formulas i.e., finite sets of polyhedra.

Undecidability

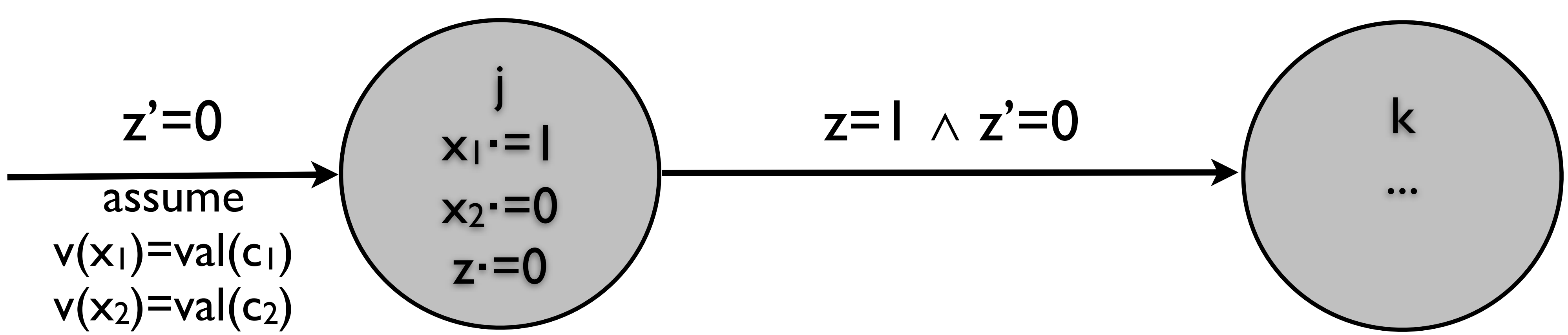
Nevetheless...

Theorem. *The reachability problem for rectangular hybrid automata is undecidable.*

Proof (sketch). By simulation of two counter machines for which the halting problem is undecidable.

To simulate a 2-CM M, we use a RHA with 3 continuous variables.

Let us consider the instruction **j: c_1:=c_1+1; goto k;**

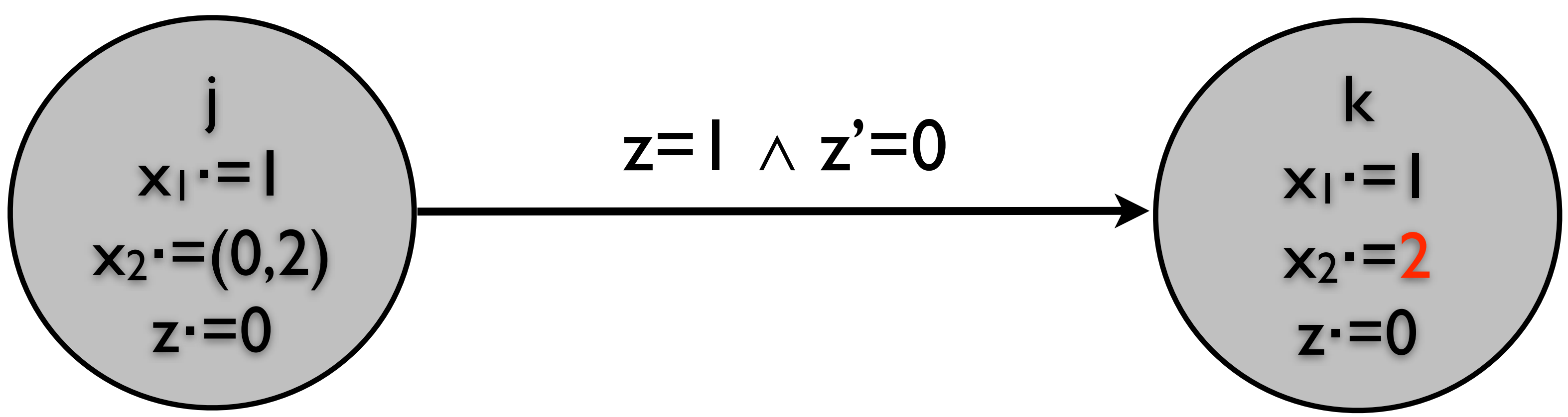


Initialized RHA

- A RHA is initialized if for all discrete jumps (l_1,σ,l_2), for all variables $x \in X$:
    - either the flow constraints on x in l_1 and l_2 are identical
    - or variable x is updated during the discrete jump from l_1 to l_2.

Initialized RHA

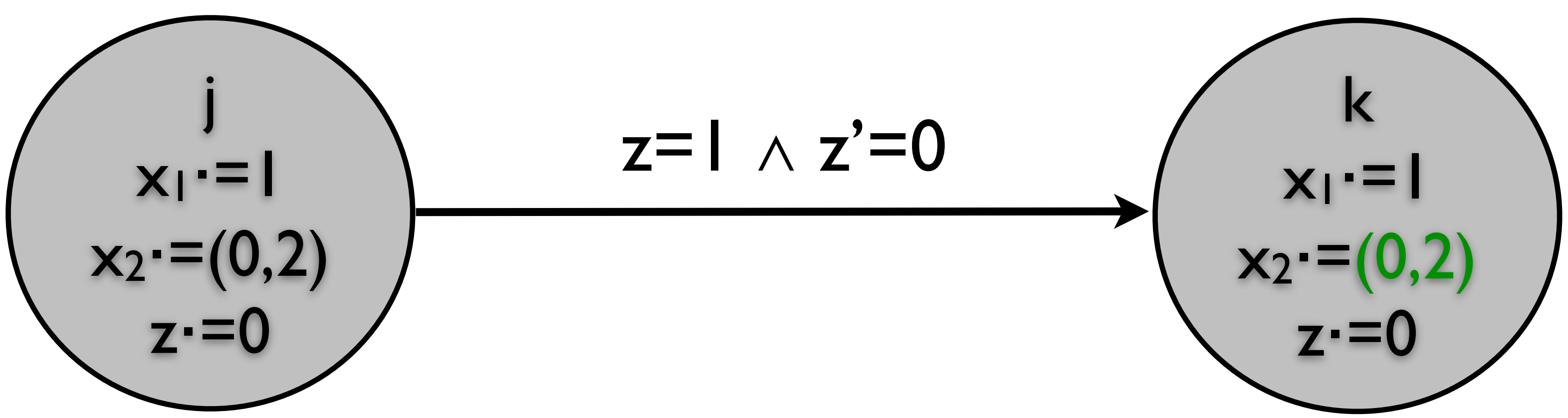
- A RHA is initialized if for all discrete jumps (l_1,σ,l_2), for all variables $x \in X$:
  - either the flow constraints on x in l_1 and l_2 are identical
  - or variable x is updated during the discrete jump from l_1 to l_2.



is not initialiazed

Initialized RHA

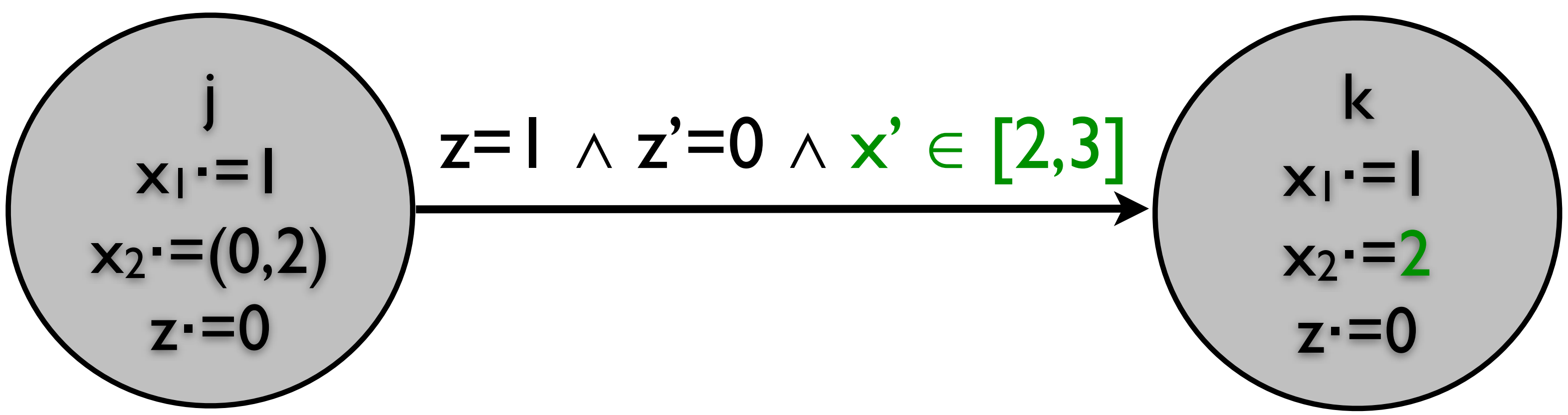
- A RHA is initialized if for all discrete jumps (l_1,σ,l_2), for all variables x∈X:
    - either the flow constraints on x in l_1 and l_2 are identical
    - or variable x is updated during the discrete jump from l_1 to l_2.



is initialiazed

Initialized RHA

- A RHA is initialized if for all discrete jumps (l_1,σ,l_2), for all variables $x \in X$:
    - either the flow constraints on x in l_1 and l_2 are identical
    - or variable x is updated during the discrete jump from l_1 to l_2.



is initialiazed

Initialized RHA

- A RHA is initialized if for all discrete jumps (l_1,σ,l_2), for all variables $x \in X$:
  - either the flow constraints on x in l_1 and l_2 are identical
  - or variable x is updated during the discrete jump from l_1 to l_2.

Theorem. The reachability problem (and LTL model-checking problem) is decidable for the class of initialized rectangular automata.

Note that Initialized RHA generalizes timed automata.

Approximating affine hybrid automata by rectangular hybrid automata

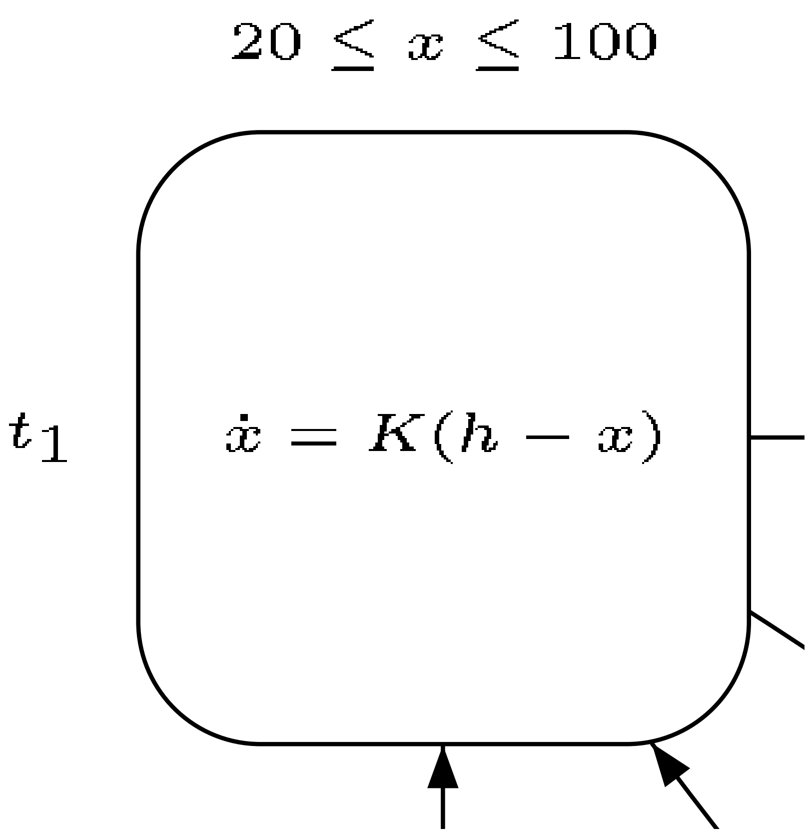
Rectangular approximations

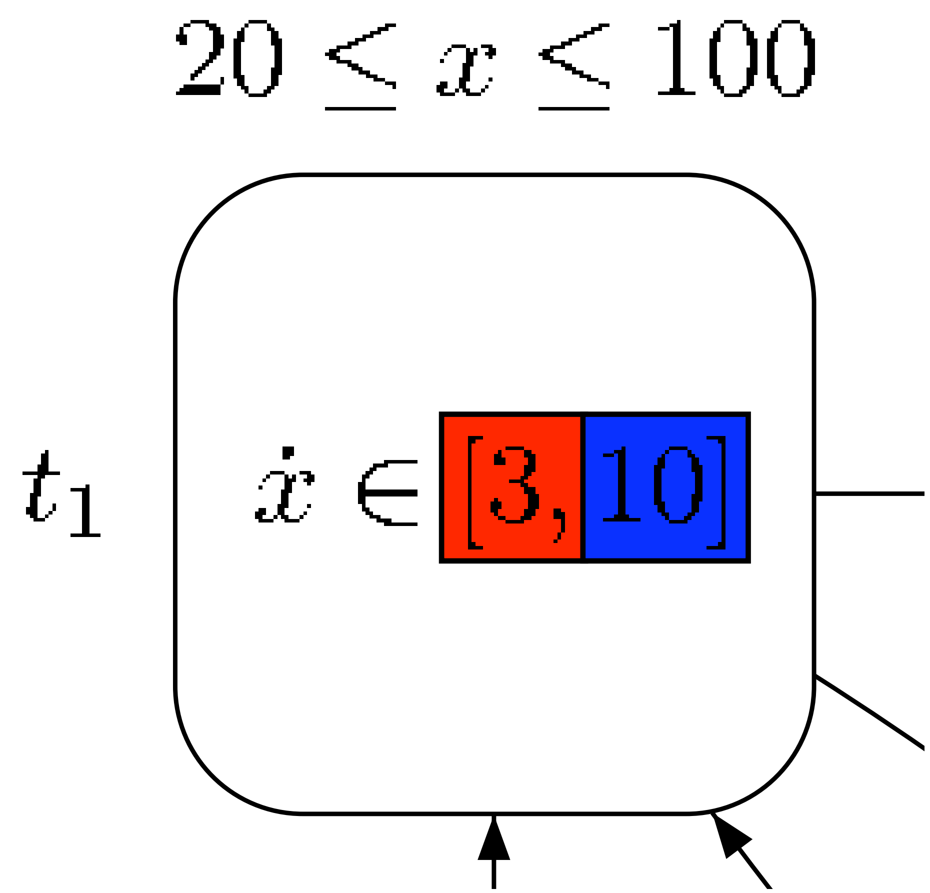
- Approximate complex dynamics with rectangular dynamics in a systematic way;
- ... this allow us to use **PhaVer** or **Hytech** for example.
- In practice, those approximations are often precise enough to infer important properties of the original HA.
- ... this is related to *abstract interpretation*.

Rectangular approximations

- For each control mode we partition the space into rectangular regions;
- Within each region, the flow field is over-approximated using rectangular flows.
- Those approximations can often be obtained automatically by computing lower and upper bounds on derivatives within rectangular regions.
- The approximations can be made arbitrarily precise by approximating over suitably small regions of the state space.

An example



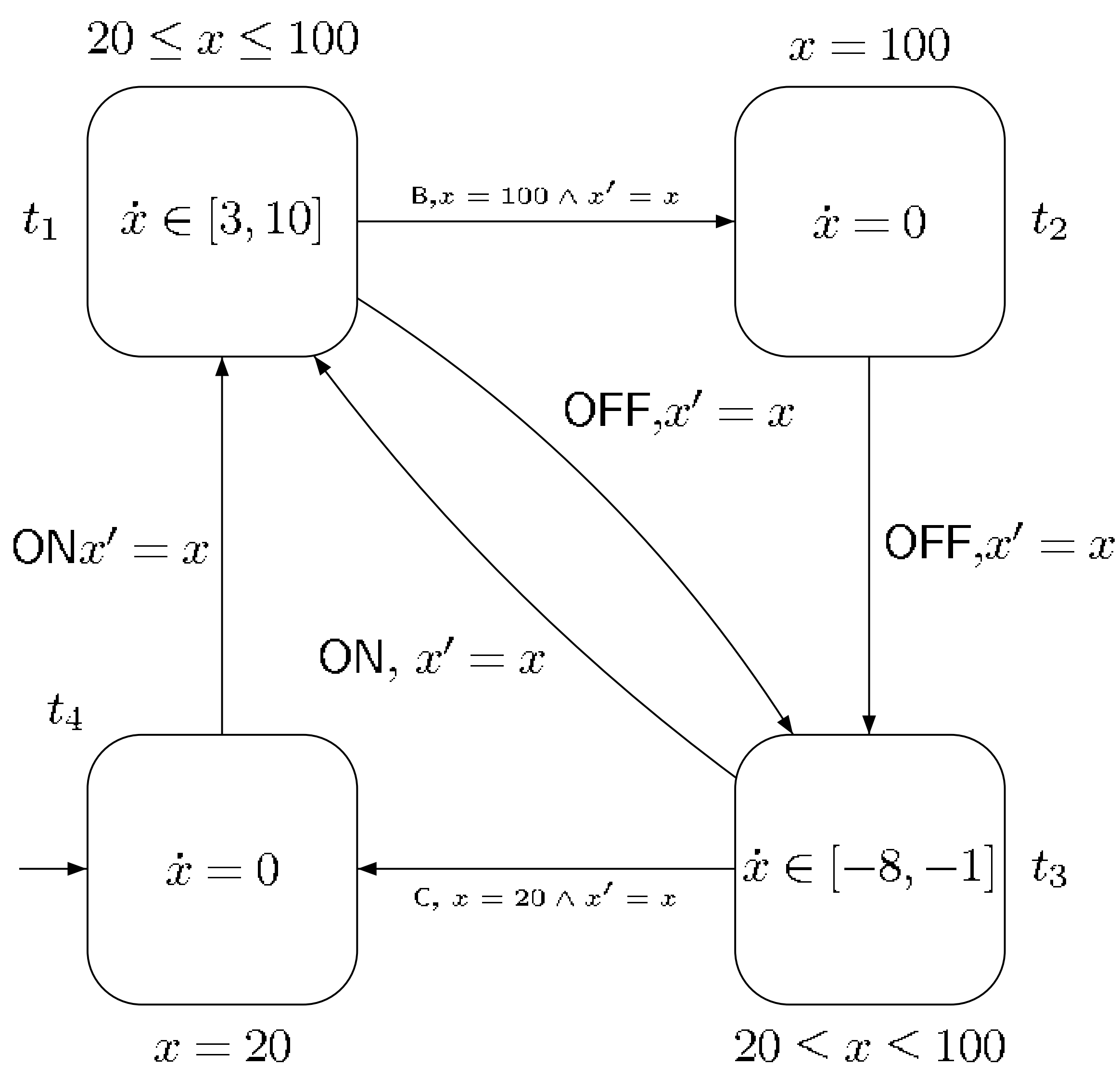


$\mathrm{Max}_{x \in [20,100]} K(h-x) = K(h-20) = 0.075(150-20) = 9.75 \le 10$

$\mathrm{Min}_{x \in [20,100]} K(h-x) = K(h-100) = 0.075(150-100) = 3.75 \ge 3$

An example

- Applying this computation for each location, we get the following rectangular approximation of the tank:



Over-approximations and correctness

- Let us note RectOver(H) the rectangular overapproximation obtained using the previous method;
- RectOver(H) is a over-approximaiton of the original system in the following formal sense:

$$\mathbf{Path_F}(\llbracket \mathbf{H} \rrbracket) \subseteq \mathbf{Path_F}(\llbracket \mathbf{RectOver(H)} \rrbracket)$$

- Transfert of correctness from overapproximations:

$$\text{if } \mathbf{Path_F}(\llbracket \mathbf{RectOver(H)} \rrbracket) \cap \mathbf{BadPaths} = \emptyset$$
$$\text{then } \mathbf{Path_F}(\llbracket \mathbf{H} \rrbracket) \cap \mathbf{BadPaths} = \emptyset$$

Over-approximations and false negatives

- When over-approximating the behavior of a system, we face the possibility to get false negatives during verification;
- Indeed, the set of behaviors of the over-approximation is a superset of the behaviors of the original system...
- ...so if we have that

$$\mathbf{Path_F}(\llbracket \mathbf{RectOver(H)} \rrbracket) \cap \mathbf{BadPaths} \neq \varnothing$$

it is not nessarily the case that

$$\mathbf{Path_F}(\llbracket \mathbf{H} \rrbracket) \cap \mathbf{BadPaths} \neq \varnothing$$

Candidate counter examples

- A path $\lambda = s_0 \tau_0 s_1 \tau_1 \ldots \tau_{n-1} s_n$ is an candidate counter example if
  - $\lambda \in [\![\mathrm{OverRect}(H)]\!] \cap \mathrm{BadPaths}$
- When facing a candidate counter example, we check if the counter example is realizable in the original model, so we ask:
  - $\lambda \in^{?} [\![H]\!]$

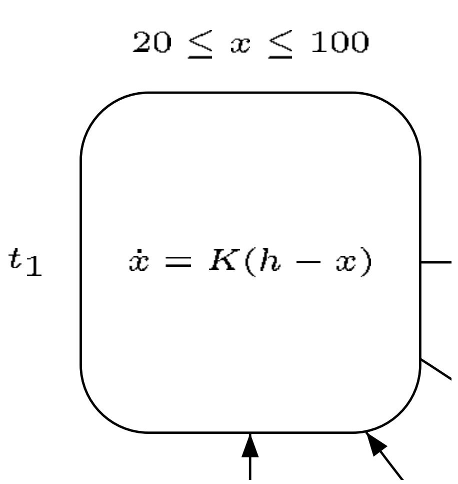
    This test is possible for larger class than rectangular automata, i.e. affine/polynomial hybrid automata.
- If $\lambda \in [\![H]\!]$, then we have found a real counter example i.e., the a Bad path in the original HA H.

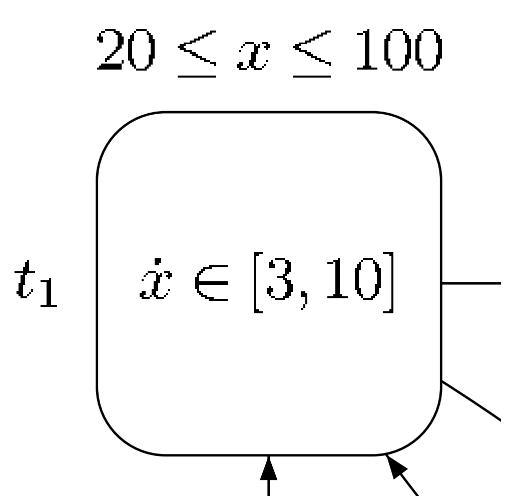
Spurious counter-examples

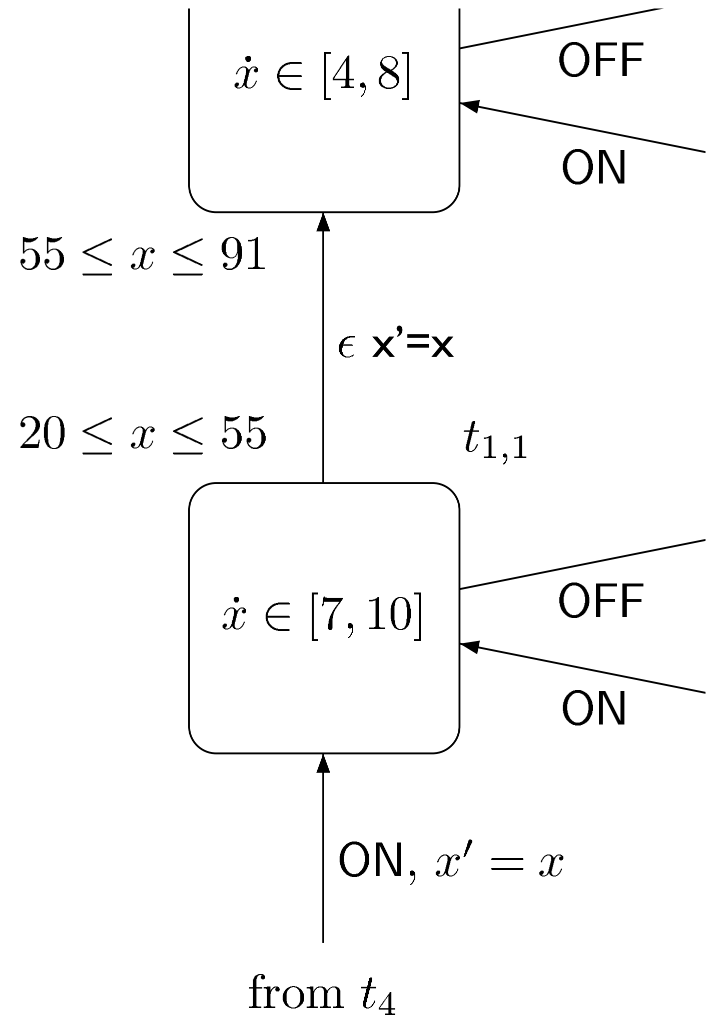
- If $\lambda \notin [\![H]\!]$, then λ is a spurious counter example i.e.:
  - $\lambda \in [\![\mathrm{OverRect}(H)]\!] \cap \mathrm{BadPaths}$
  - $\lambda \notin [\![H]\!]$
- In this case, we must refine OverRect(H) in order to eliminate the counter example.
- There is a large research effort in the CAV community on the so-called counter-example based abstraction refinement, and variants.

Abstraction refinement

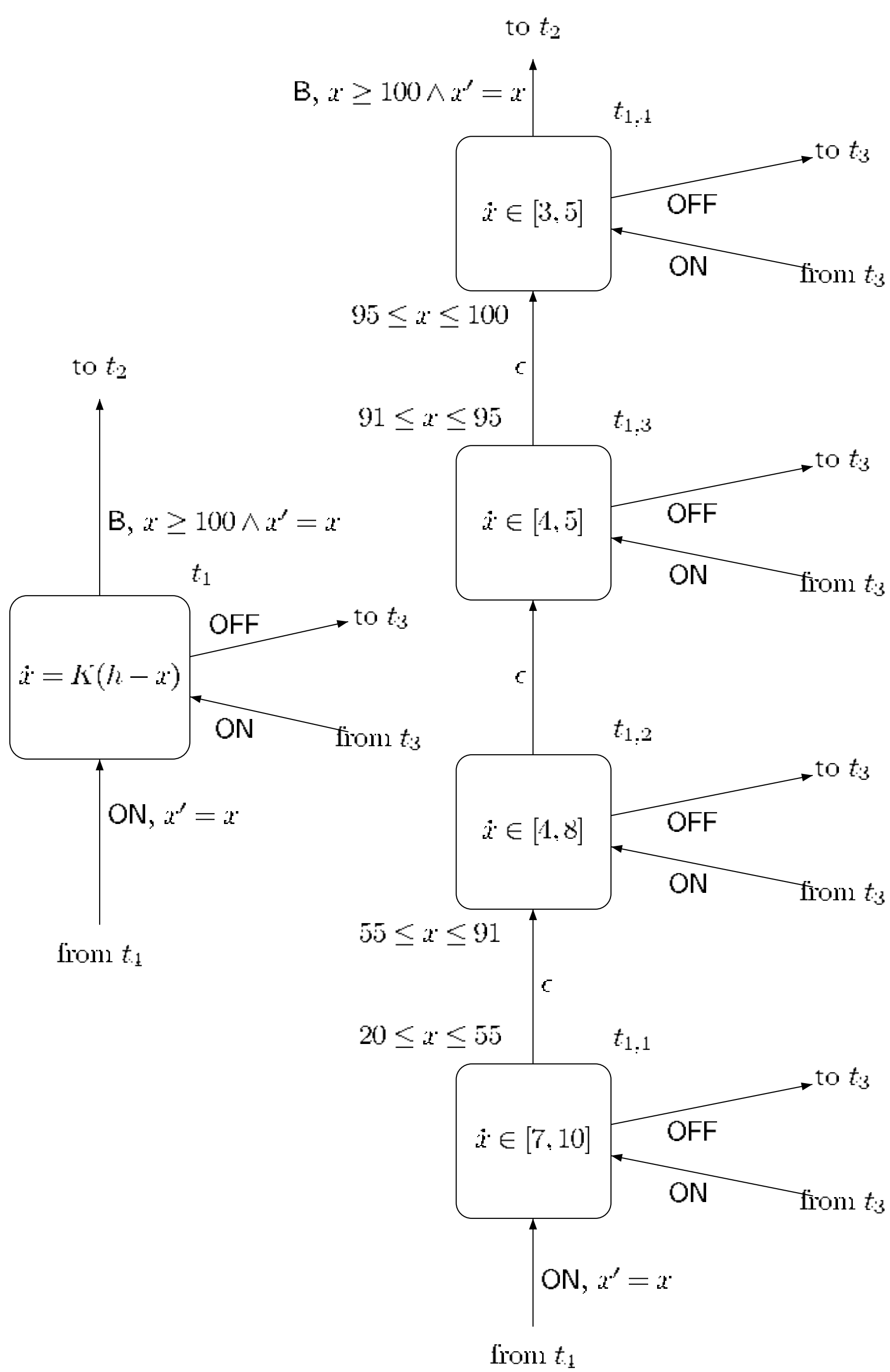
- In presence of spurious counter examples, we refine the rectangular approximation by splitting locations to decorate them with smaller rectangular regions.



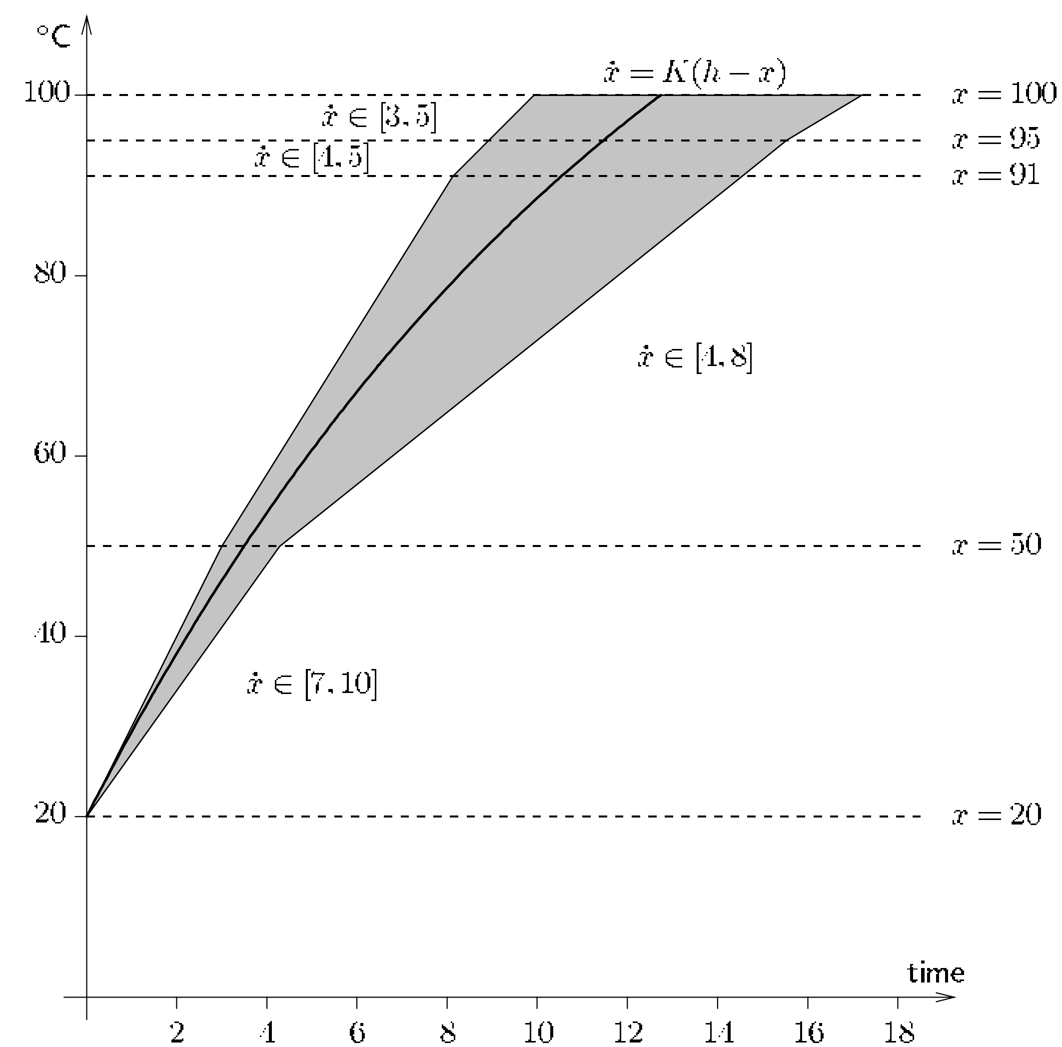




Example

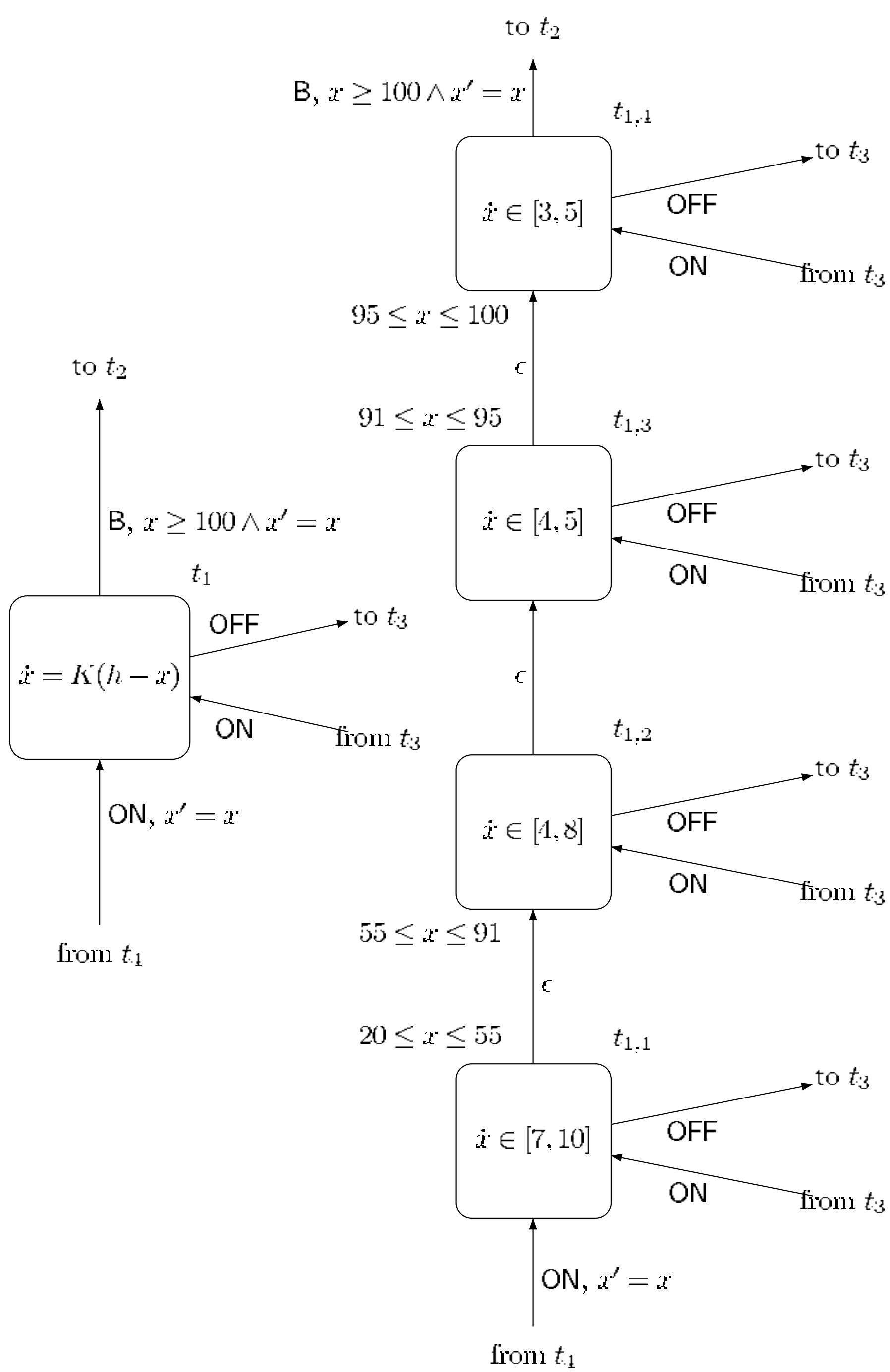


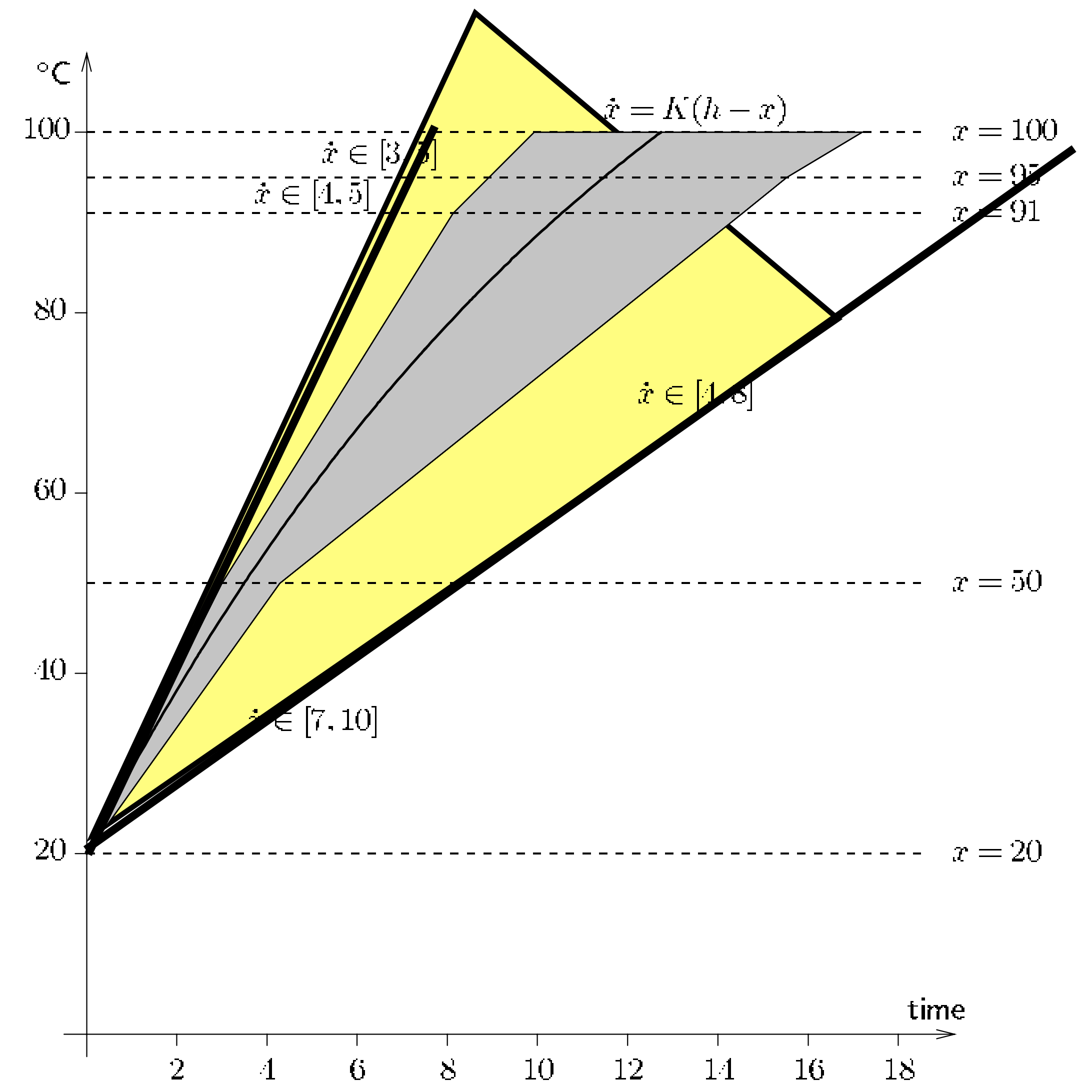
to t_2
B, $x \geq 100 \wedge x' = x$
$t_{1,4}$
$\dot{x} \in [3,5]$
to t_3
OFF
ON
from t_3
$95 \leq x \leq 100$
c
to t_2
$91 \leq x \leq 95$
$t_{1,3}$
$\dot{x} \in [4,5]$
B, $x \geq 100 \wedge x' = x$
t_1
OFF
to t_3
$\dot{x} = K(h - x)$
ON
from t_3
$t_{1,2}$
$\dot{x} \in [4,8]$
ON, $x' = x$
from t_1
$55 \leq x \leq 91$
$20 \leq x \leq 55$
$t_{1,1}$
$\dot{x} \in [7,10]$
ON, $x' = x$
from t_1



°C
$\dot{x} = K(h - x)$
$\dot{x} \in [3,5]$
$\dot{x} \in [4,5]$
$\dot{x} \in [4,8]$
$\dot{x} \in [7,10]$
$x = 100$
$x = 95$
$x = 91$
$x = 50$
$x = 20$
100
80
60
40
20
time
2
4
6
8
10
12
14
16
18

Example


to t_2
B, $x \geq 100 \wedge x' = x$
$t_{1,4}$
$\dot{x} \in [3,5]$
OFF
to t_3
ON
from t_3
$95 \leq x \leq 100$
c
$91 \leq x \leq 95$
$t_{1,3}$
$\dot{x} \in [4,5]$
c
$t_{1,2}$
$\dot{x} \in [4,8]$
$55 \leq x \leq 91$
c
$20 \leq x \leq 55$
$t_{1,1}$
$\dot{x} \in [7,10]$
ON, $x' = x$
from t_1
t_1
$\dot{x} = K(h-x)$


°C
100
80
60
40
20
$\dot{x} = K(h-x)$
$\dot{x} \in [3,5]$
$\dot{x} \in [4,5]$
$\dot{x} \in [4,8]$
$\dot{x} \in [7,10]$
$x = 100$
$x = 95$
$x = 91$
$x = 50$
$x = 20$
time
2
4
6
8
10
12
14
16
18

Conclusion

Conclusion

- We have defined the syntax and semantics of hybrid automata ...
- ... shown how to use them to model compositionally a hybrid system ...
- ... shown how to formalize safety requirements using monitors ...
- ... recalled the main ingredients for safety and reachability analysis ...
- ... introduced the subclasses of rectangular hybrid automata and initialized RHA ...
- ... shown how to over-approximate complex hybrid automata using RHA.

Bibliography

See written notes !