Controller Synthesis for Linear Hybrid Systems

Part 3: Controller Synthesis

Formal Methods for Cyber-Physical Systems
Verona, September 12-16, 2017

Marco Faella
Università di Napoli “Federico II”

Controllers

- Automatic mechanisms that guide a physical system by:
  - receiving signals from ***sensors***, and
  - operating ***actuators***

- Examples:
  - control the temperature of a room
  - control the braking of a vehicle (ABS, ESP)
  - control the flight of an airplane
  - control an industrial process
  - ...

Classical Model

- All vars are continuous
- Differential equations of the type

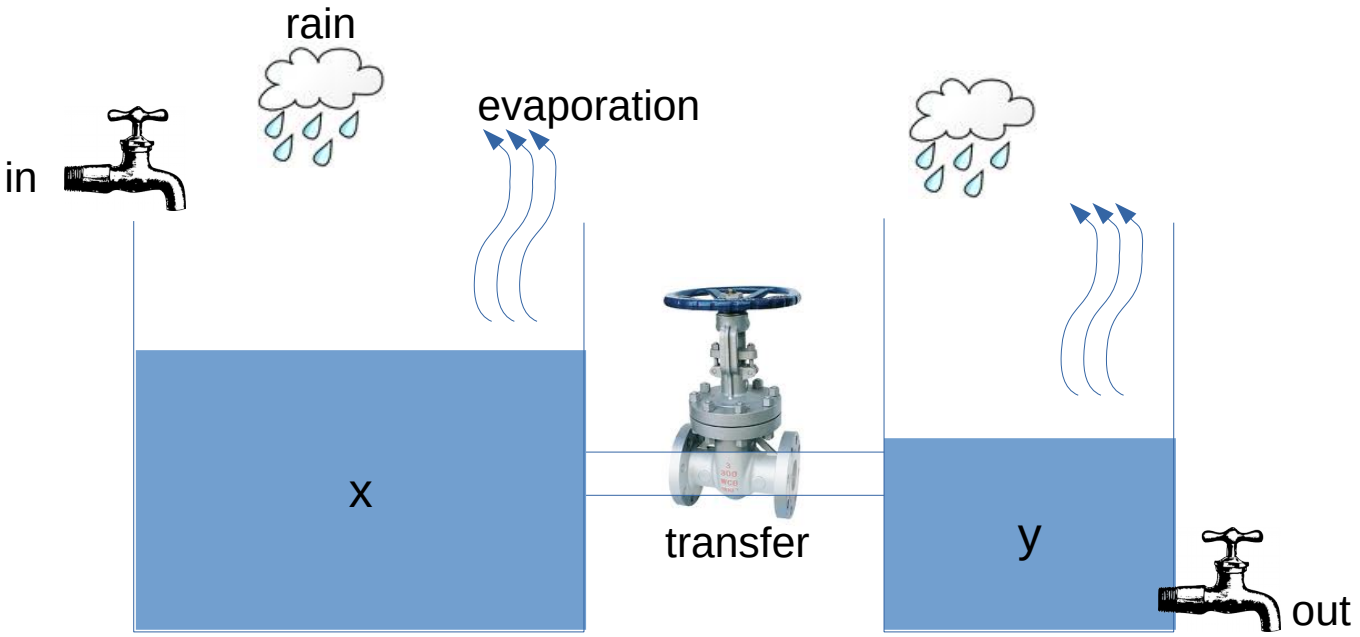
$$\dot{x} = f(x, u, d)$$

  where:
  - x: state vars
  - u: **controlled** vars (input signals from the controller)
  - d: disturbances (input signals from the environment)
- Commonly, **linear** equations
- *Dynamic systems* (math), (Optimal) *Control Theory* (eng), *differential games* (game t.)
- **Problem:** identify control function u(t) (*open loop*) or u(t, x) (*closed loop*) that guides the system to fulfill a given goal

Hybrid Games

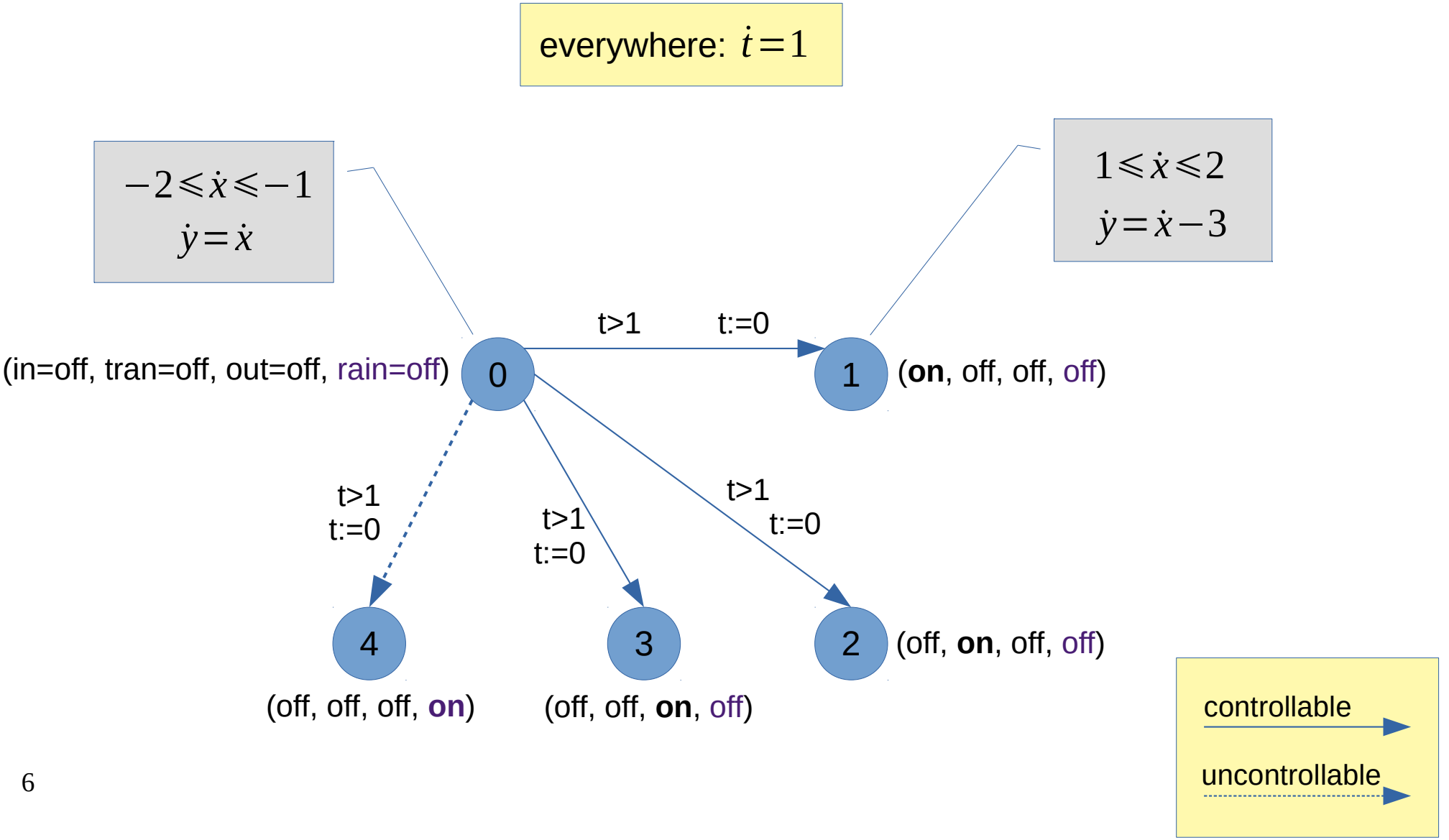
- A CS-oriented model of a non-linear control system
- A hybrid automaton whose transitions are divided between *controllable* and *uncontrollable*
- i.e., the controller can only take certain transitions
  - it does not directly influence the continuous behavior
  - a.k.a. *switching controller*
- Typical goals:
  - safety
  - reachability

Example: Two Open-air Tanks



- We control 3 discrete valves (on/off): *in*, *transfer*, and *out*
- Rain is an uncontrollable discrete event (on/off)
- Evaporation rate varies within given bounds
- Control goal: keep the level in both tanks within bounds, despite rain and evaporation (safety goal)
- At least one time unit between any two transitions (prevents Zenoness)

HG Fragment for Two Open-air Tanks



The Safety Control Problem

- Control **goal**: keep system within a given set R of *safe* states
  - *safety game*
- A (control) **strategy** is a function from states to moves of the controller
  - possible moves: take an enabled controllable transition or do nothing
  - a form of *closed-loop* control

- A **strategy** is **winning** if it constrains the system within the safe set
- **Problem:** Given a HG and a safe set, compute the set of states from which the controller has a winning strategy (**winning states**)

The General Algorithm for Safety Games

- Inspired by finite-state games
- Based on the “controllable predecessors” operator

$$CPre : 2^S \rightarrow 2^S$$

Definition. Given a set of states Z, **CPre(Z)** contains the states from which the controller can ensure that the system **remains in Z until the next discrete transition (included).**

The General Algorithm for Safety Games

Given the set of safe states R, the set of winning states is the **fixed point** of the sequence:

$$Z_0 = R$$
$$Z_1 = R \cap CPre(Z_0)$$
$$\ldots$$
$$Z_{n+1} = Z_n \longleftarrow \nu Z \,.\, R \cap CPre(Z)$$

Properties of CPre

Clearly, CPre(Z) $\subseteq$ Z.

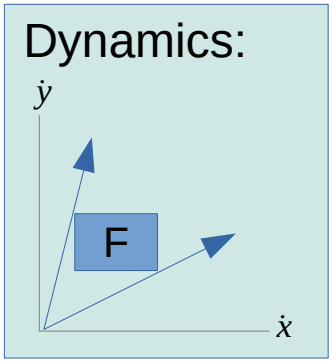
The state (q,x) $\in$ Z does **not** belong to CPre(Z) if:

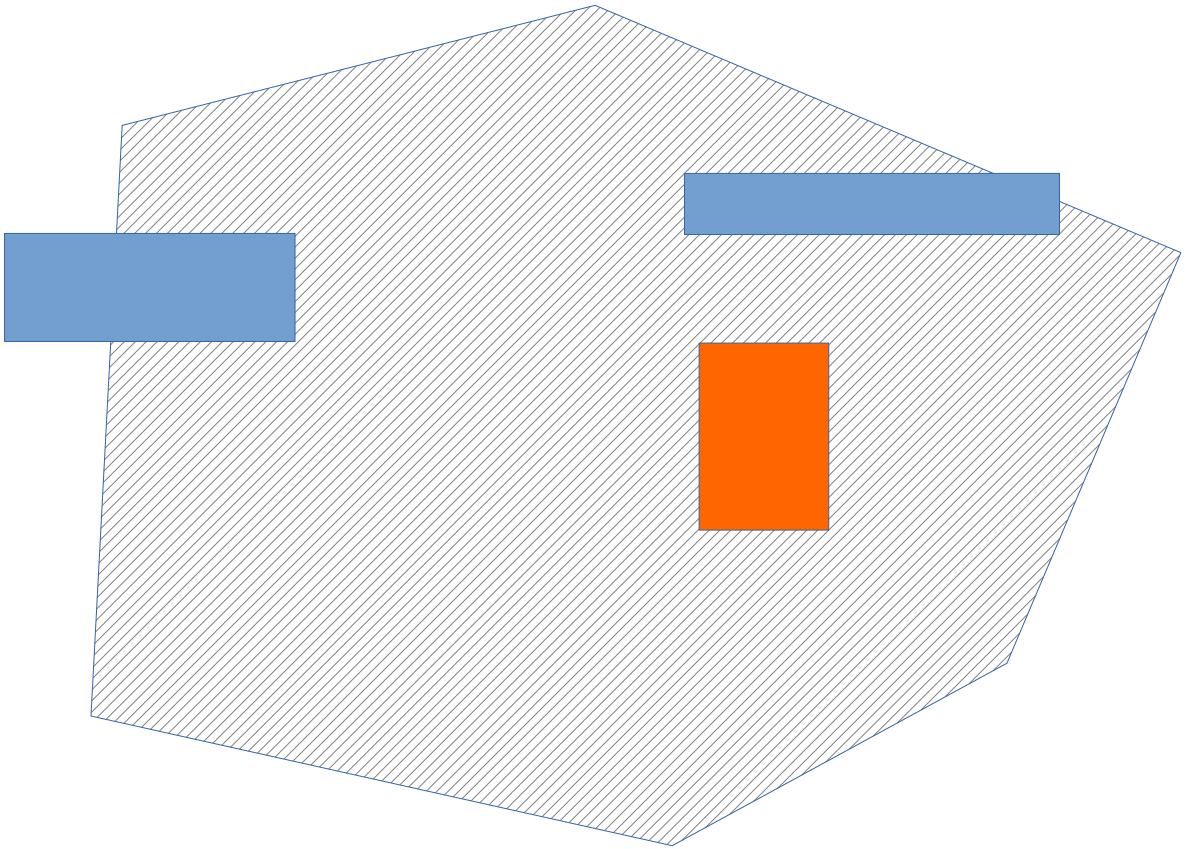
- **there exists** a trajectory from (q,x) such that:
  - either the trajectory exits from Z,
  - or the trajectory reaches the guard of an *uncontrollable* transition that leads *outside* Z;
  - in the meanwhile, the trajectory avoids the guards of the *controllable* transitions that lead *into* Z.

Linear Hybrid Games

- LHAs with two types of transitions (controllable and uncontrollable)
- **Fact:** CPre is exactly computable on polyhedra (see next)
- The fixed point may **not** be reached in a finite number of steps
- Finite-horizon controller synthesis decidable

Computing CPre(Z) on LHGs



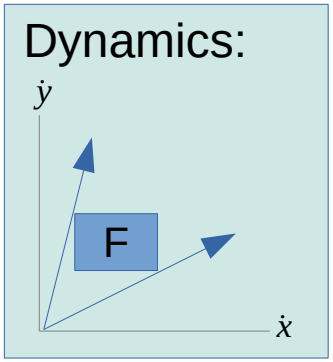


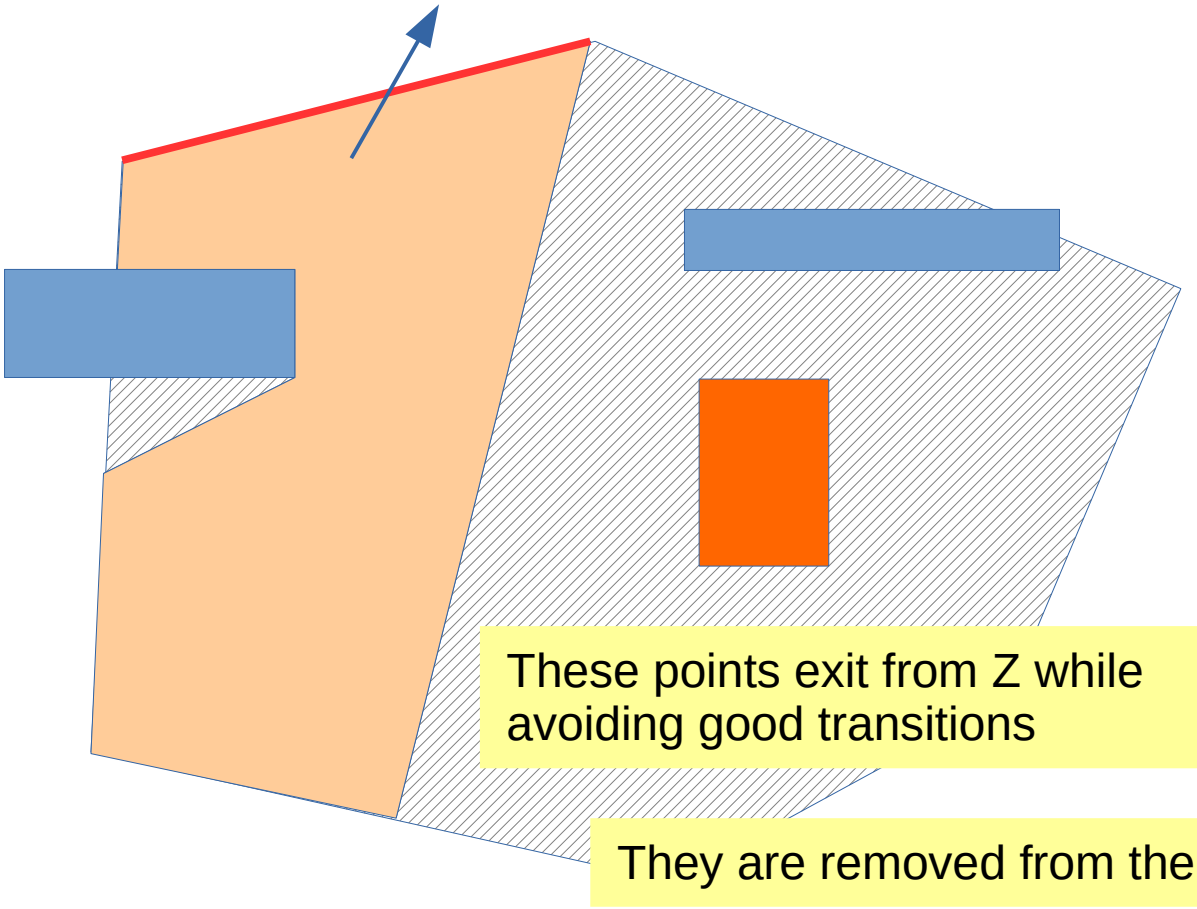
Z(q)

controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs





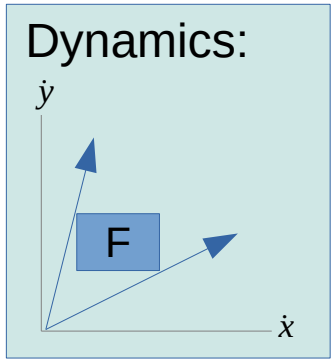
These points exit from Z while avoiding good transitions

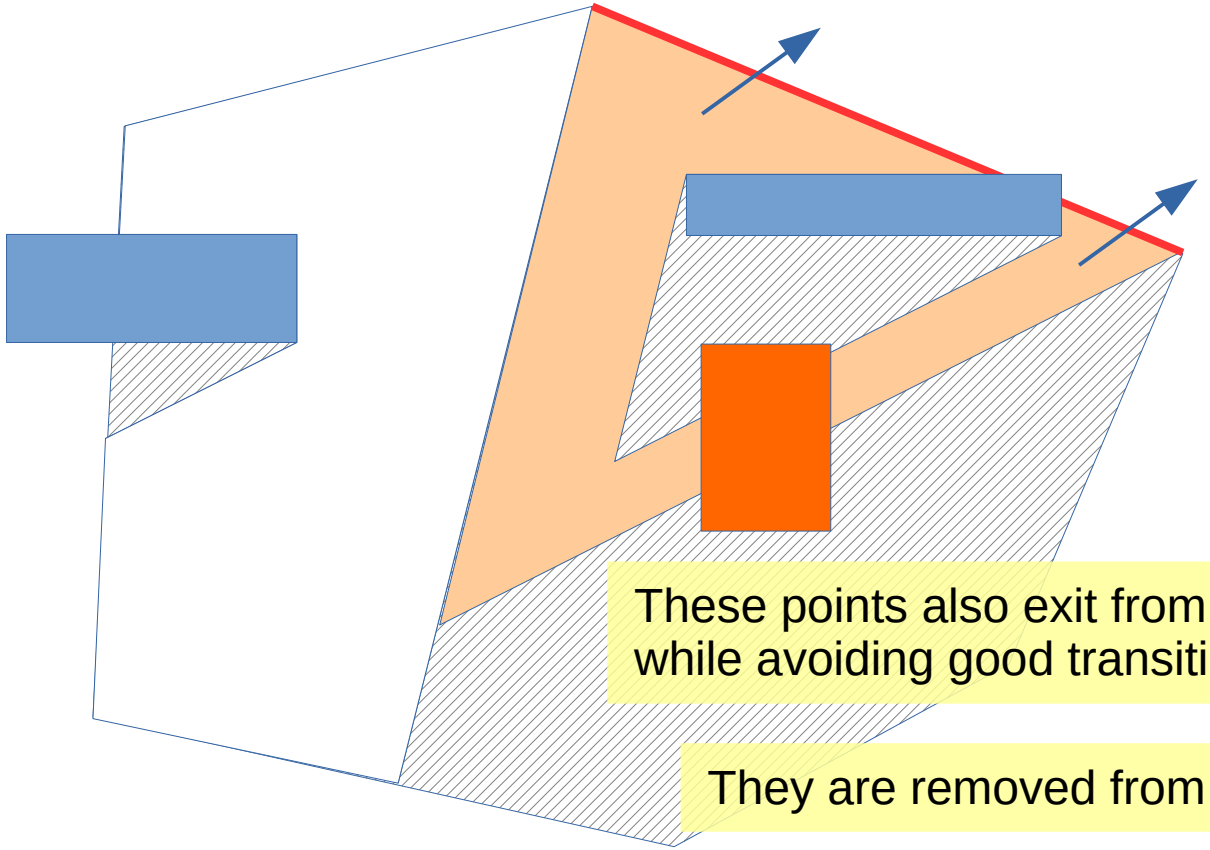
They are removed from the result

controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs





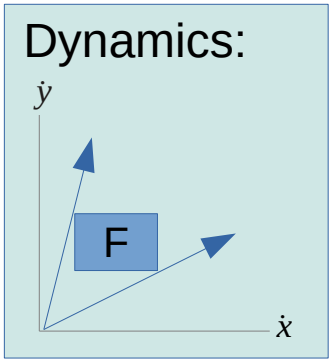
These points also exit from Z
while avoiding good transitions

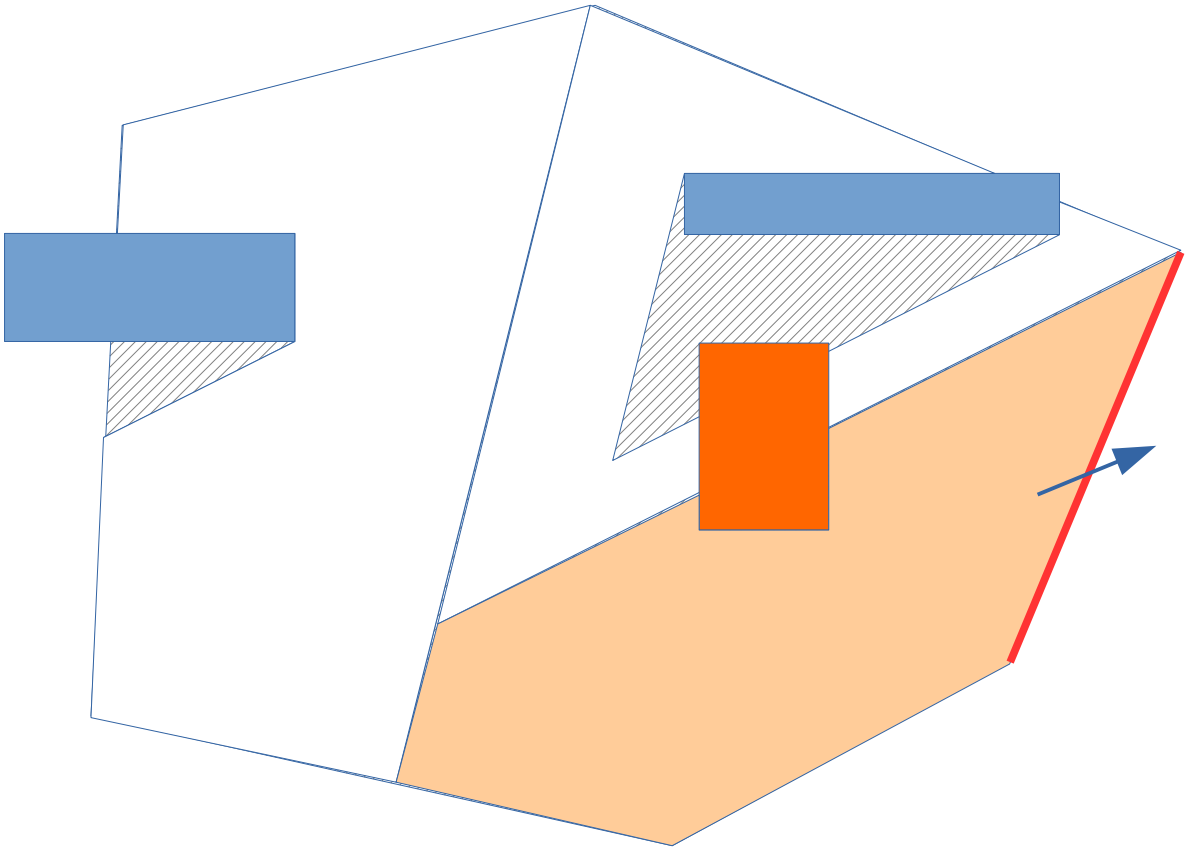
They are removed from the result

controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs

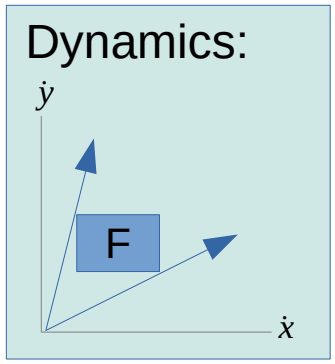


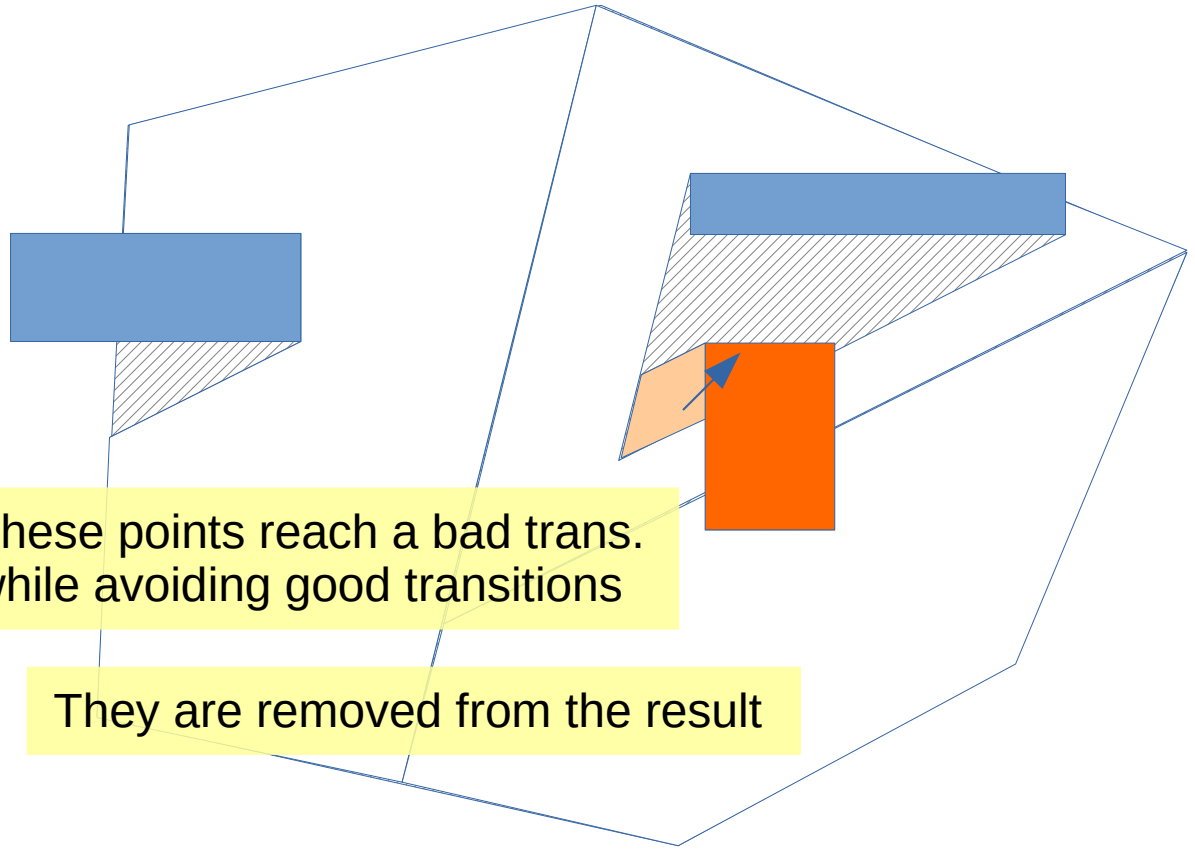


controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs





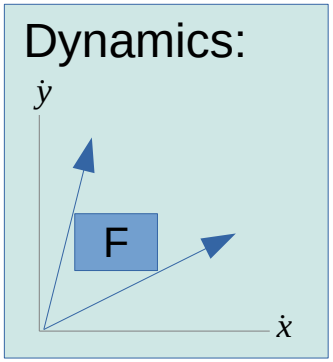
These points reach a bad trans. while avoiding good transitions

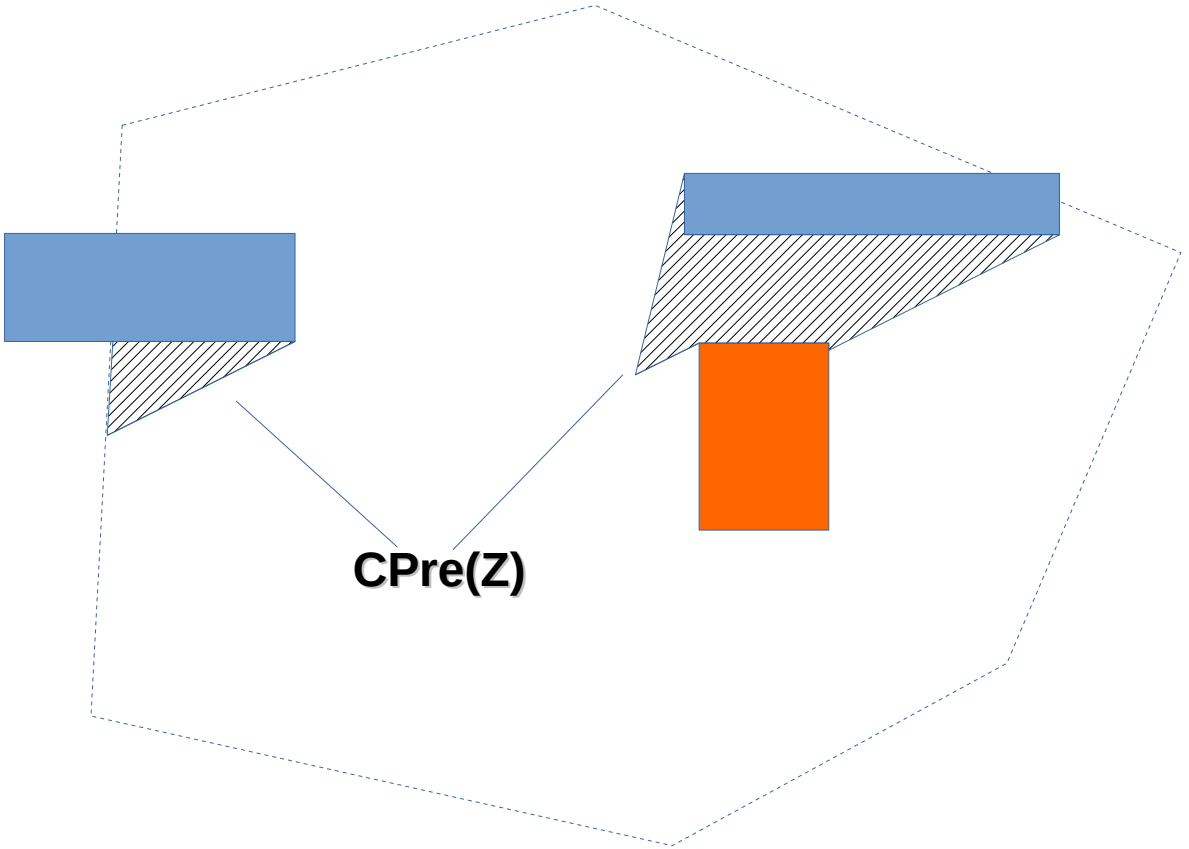
They are removed from the result

controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs

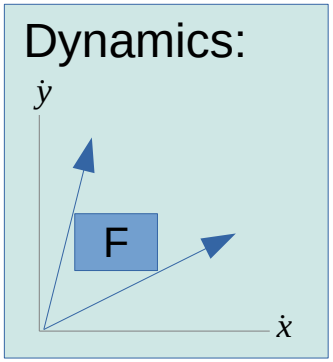


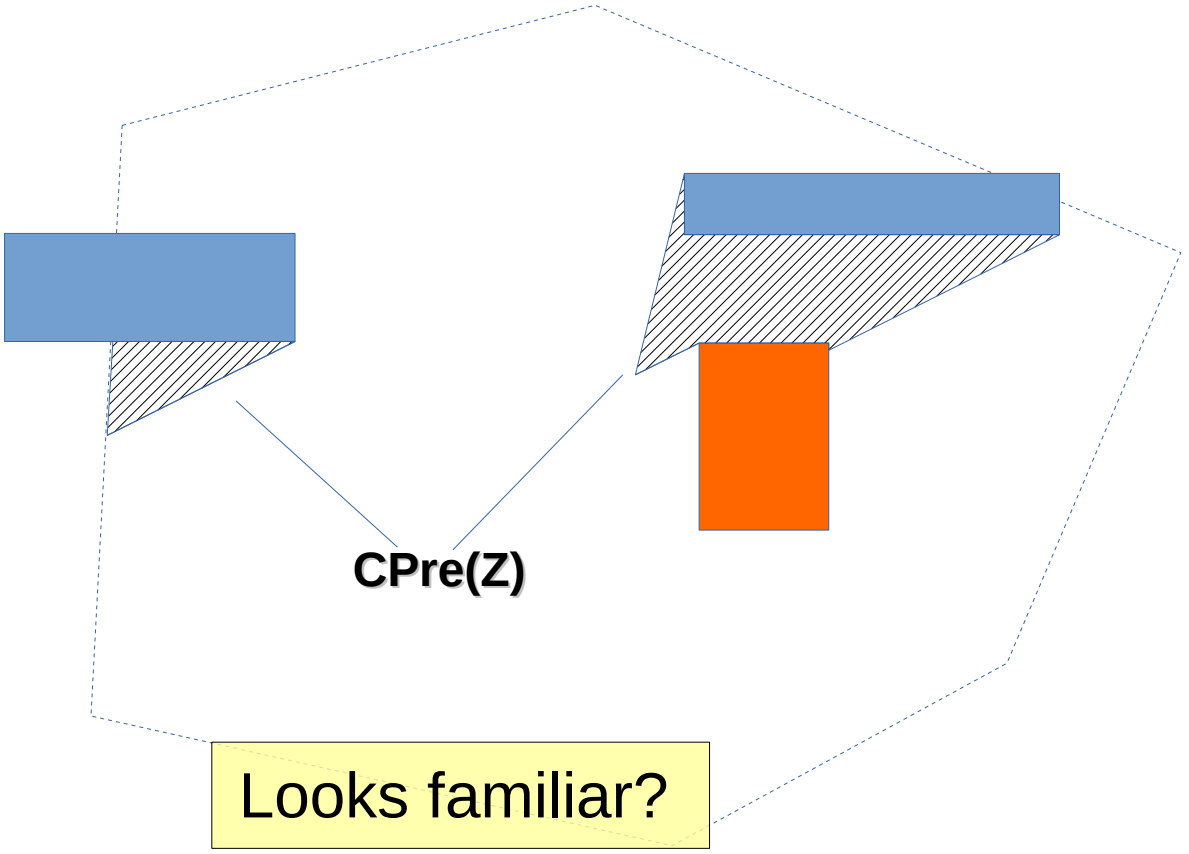


controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs





controllable trans. leading into Z (good)

uncontrollable trans. leading outside Z (bad)

Computing CPre(Z) on LHGs

We can compute CPre using RWA!

$$\mathrm{CPre}(Z) = Z \setminus \mathbf{RWA}(\overline{Z} \cup \mathrm{Bad}, \mathrm{Good})$$

where:

Bad = $\mathrm{PreJmp}_{\mathbf{u}}(\overline{Z})$ predecessors under **u**ncontrollable trans.

Good = $\mathrm{PreJmp}_{\mathbf{c}}(Z)$ predecessors under **c**ontrollable trans.

Warning: there are special effects when a trajectory exits from Z and from the invariant at the same time.

In Temporal Logic

$$RWA(U,V) = \exists\, \overline{V}\ \textit{Until}\ U$$

$$CPre(Z) = Z \cap \forall\, (Z \cap \overline{Bad})\ \textit{WUntil}\ Good$$

weak until

Recall the classical equivalence:

$$\forall\, A\ \textit{WUntil}\ B \ \equiv\ \neg\, \exists\, \overline{B}\ \textit{Until}\ \overline{A}$$

Hence,

$$CPre(Z) = \ Z \setminus \mathbf{RWA}(\overline{Z} \cup Bad, Good)$$

The Reachability Control Problem

The Reachability Control Problem

- Control **goal**: bring system into a desired set of target states
  - *reachability game*

- **Problem.** Given a HG and a target set, compute the set of states from which the controller has a winning strategy (**winning states**)

(non-) Duality

- In most types of finite-state games, a reachability game can be reduced to its *dual safety game*
- I.e., given a target region R, to solve the game with goal “F R” we exchange players and solve the game with goal “G $\overline{R}$”

- This does not work for LHGs
- LHGs are **asymmetric**: continuous flow (timed steps) is always adversarial
- We need a **direct algorithm** for reachability games

The General Algorithm for Reachability Games

Based on a variant to the controllable predecessors operator

$$CPreReach : 2^S \rightarrow 2^S$$

Definition. Given a set of states Z, **CPreReach(Z)** contains the states from which the controller can ensure that **the system reaches Z within the next discrete transition (included).**

The General Algorithm for Reachability Games

- The set of winning states is the fixed point of the sequence:

$$Z_0 = \emptyset$$
$$Z_1 = R \cup CPreReach(Z_0)$$
$$\ldots$$
$$Z_{n+1} = Z_n \longleftarrow \mu Z . R \cup CPreReach(Z)$$

Properties of CPreReach

Clearly, $Z \subseteq \text{CPreReach}(Z)$.

A state $(q,x) \notin Z$ belongs to CPreReach(Z) when:

- **for all trajectories** from (q,x):
  - either the trajectory reaches Z,
  - or the trajectory reaches the guard of a controllable transition that leads into Z;
  - in the meanwhile, the trajectory must avoid the guards of the uncontrollable transitions that lead outside Z.

once again, we are ignoring issues related to exiting the invariant (*well-formedness*)

Computing CPreReach

In other words, **all** trajectories should reach Z ∪ Good while avoiding Bad.

$$\text{CPreReach}(Z) = \forall\ \overline{\text{Bad}}\ \textit{Until}\ (Z \cup \text{Good})$$

where as before:

Bad = $\text{PreJmp}_{\mathbf{u}}(\overline{Z})$

Good = $\text{PreJmp}_{\mathbf{c}}(Z)$

Another Type of RWA!

Fix a location q and the corresponding flow constraint Flow(q).

Definition. Given two (possibly non-convex) polyhedra **U** and **V**, **must-RWA(U,V)** is the set of points from which **all trajectories reach U while avoiding V.**

in temporal logic (CTL): $\forall\, \overline{\forall}\, \mathit{Until}\ \mathrm{U}$

CPreReach(Z) = **must-RWA**(Z ∪ Good, Bad)

Computing must-RWA

Unfortunately, forall-until (must-RWA) cannot be reduced to exists-until (RWA).

However, we can use exists-until to do the *heavy lifting*.

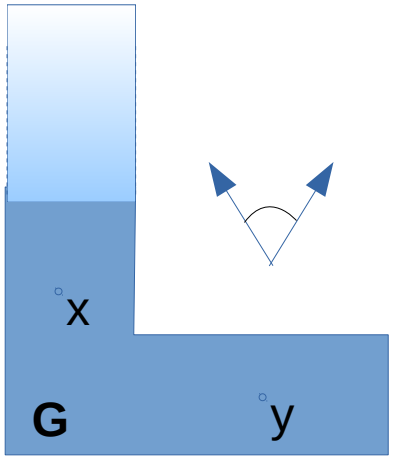
Idea:

- First, we compute an **over-approximation** of must-RWA
- Then, we **use RWA to refine it** (remove unwanted points) and obtain must-RWA

Boundedness

Given a location *q* and a (possibly non-convex) polyhedron G:

- **A point p** in G is **q-*bounded*** if all trajectories starting from p eventually exit from G
- **G is q-bounded** if all of its points are



- y is q-bounded
- x is not q-bounded
- G is not q-bounded

Computing must-RWA: the Idea in Detail

Assume w.l.o.g. that U and V are **disjoint**.

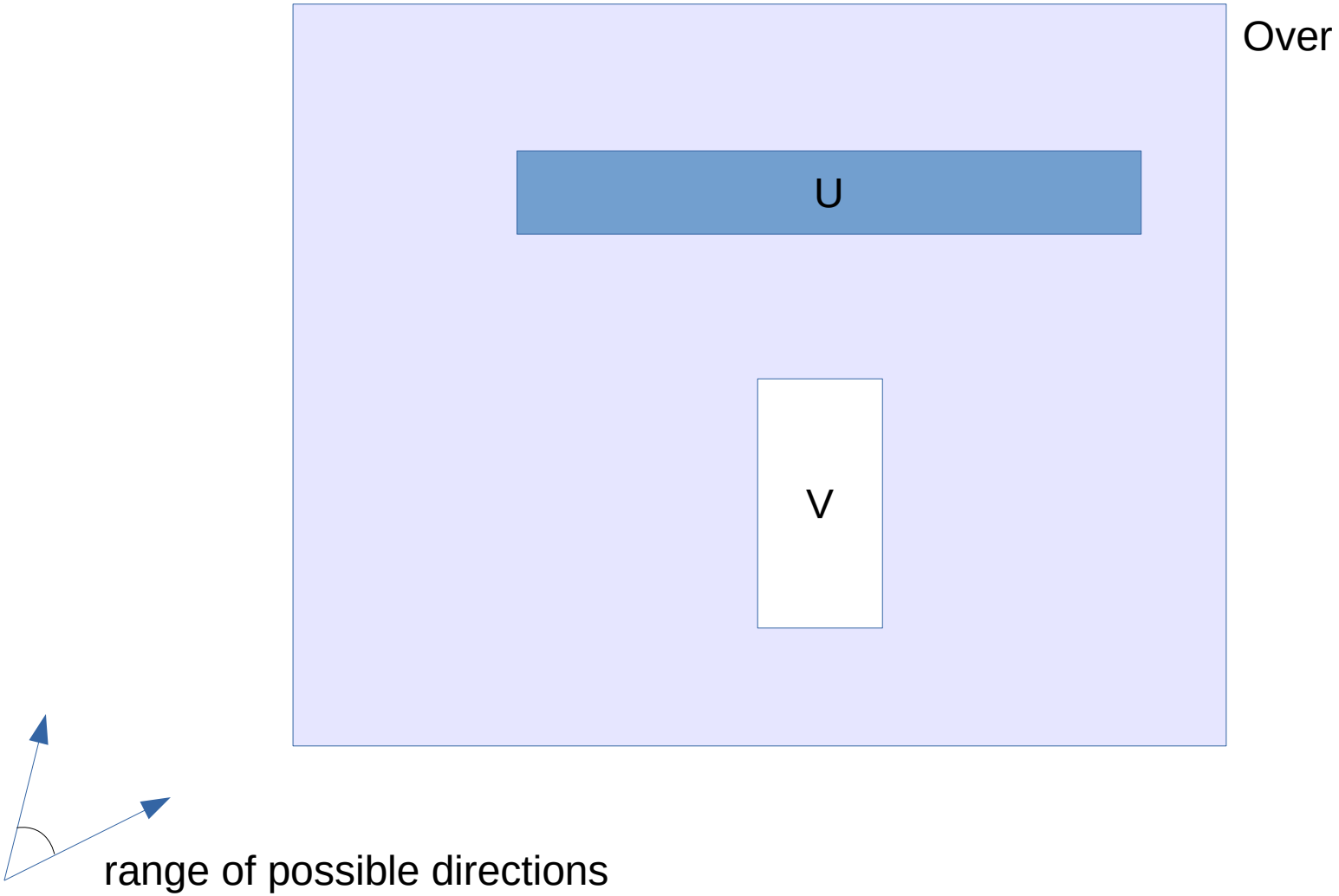
Clearly, U $\subseteq$ must-RWA(U,V).

Let **Over** be a set such that:

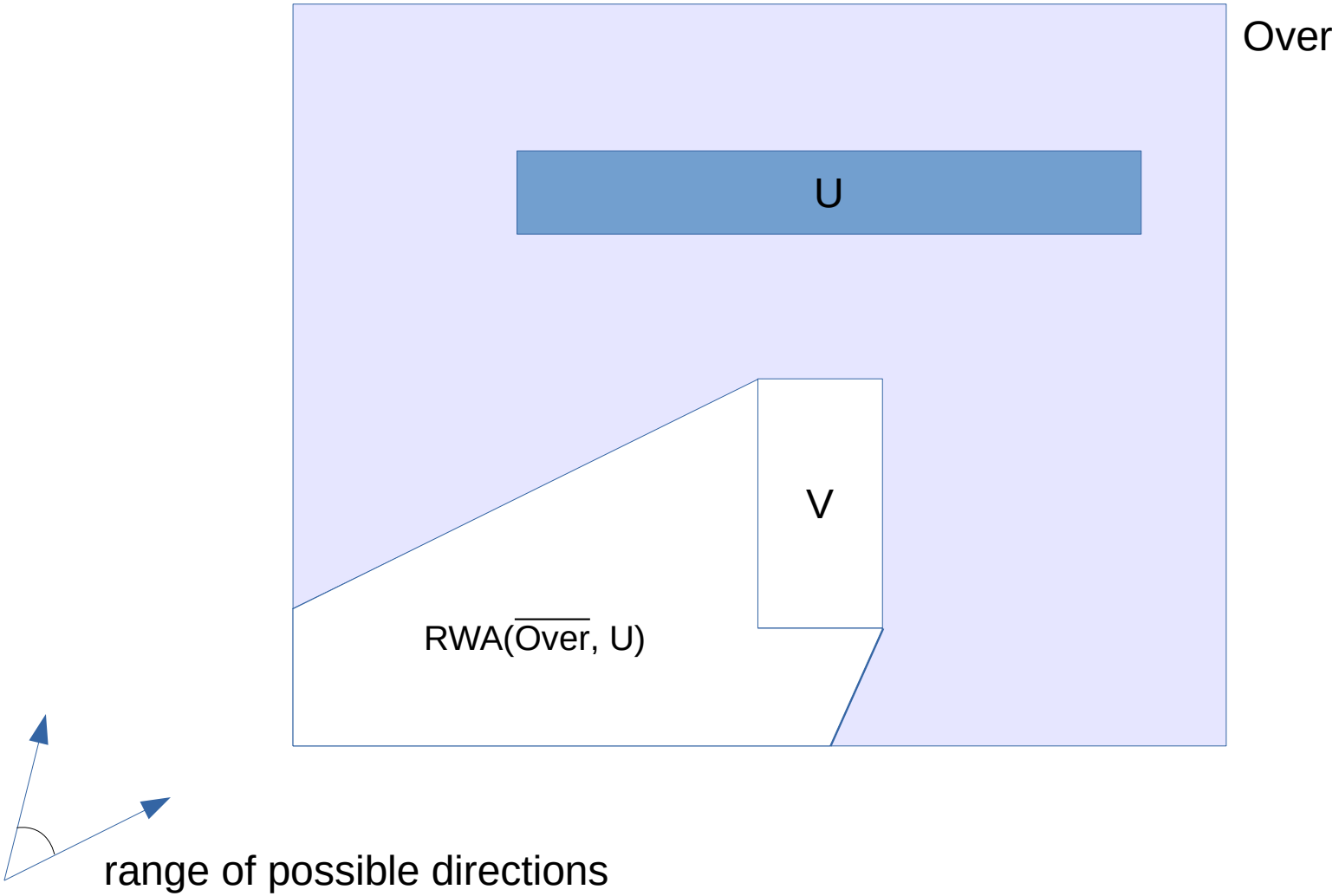
1) must-RWA(U,V) $\subseteq$ Over (it is an over-approximation)
2) Over $\cap$ V = $\varnothing$ (it does not contain trivially "losing" points)
3) Over \ U is q-*bounded* (the "unsafe" part of Over is q-bounded)

Theorem: must-RWA(U,V) = Over \ RWA**($\overline{\text{Over}}$, U)**

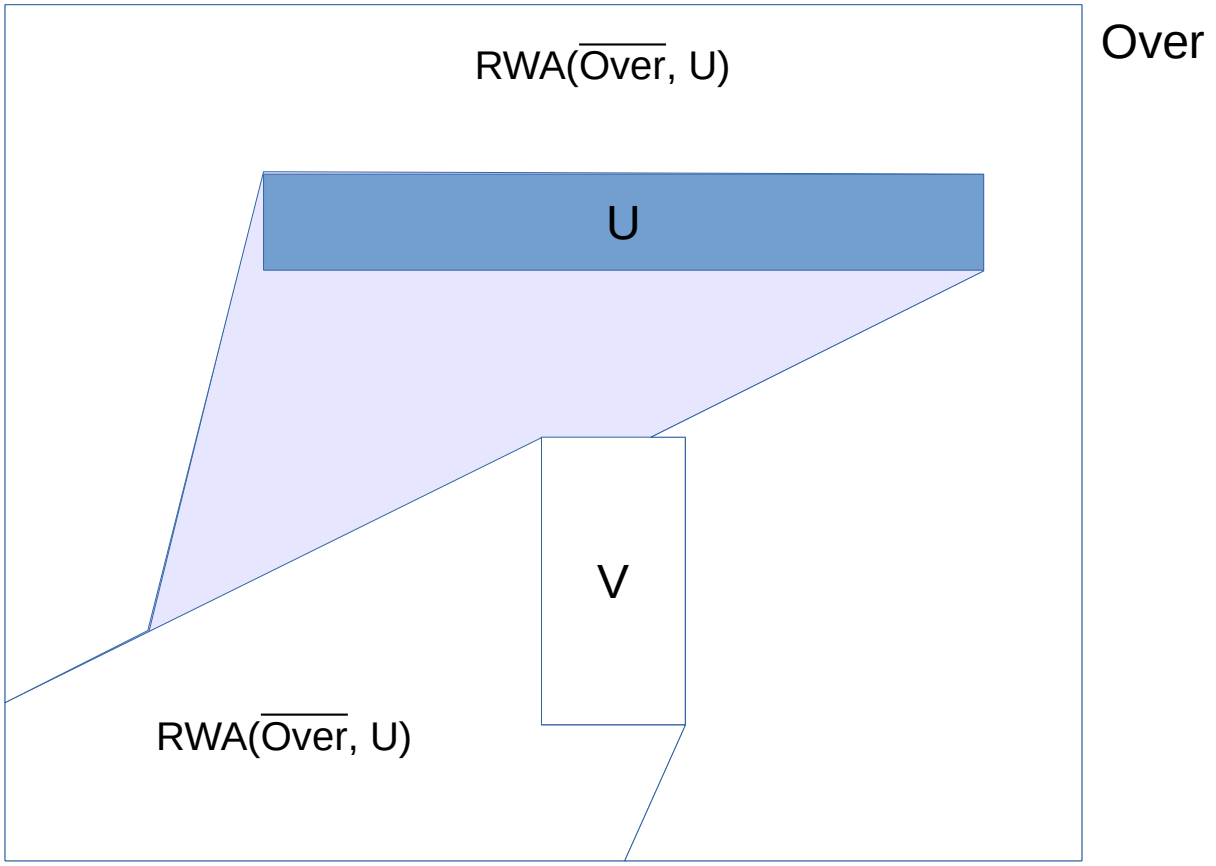
Computing must-RWA Using RWA

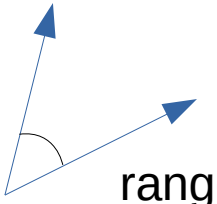


Computing must-RWA Using RWA



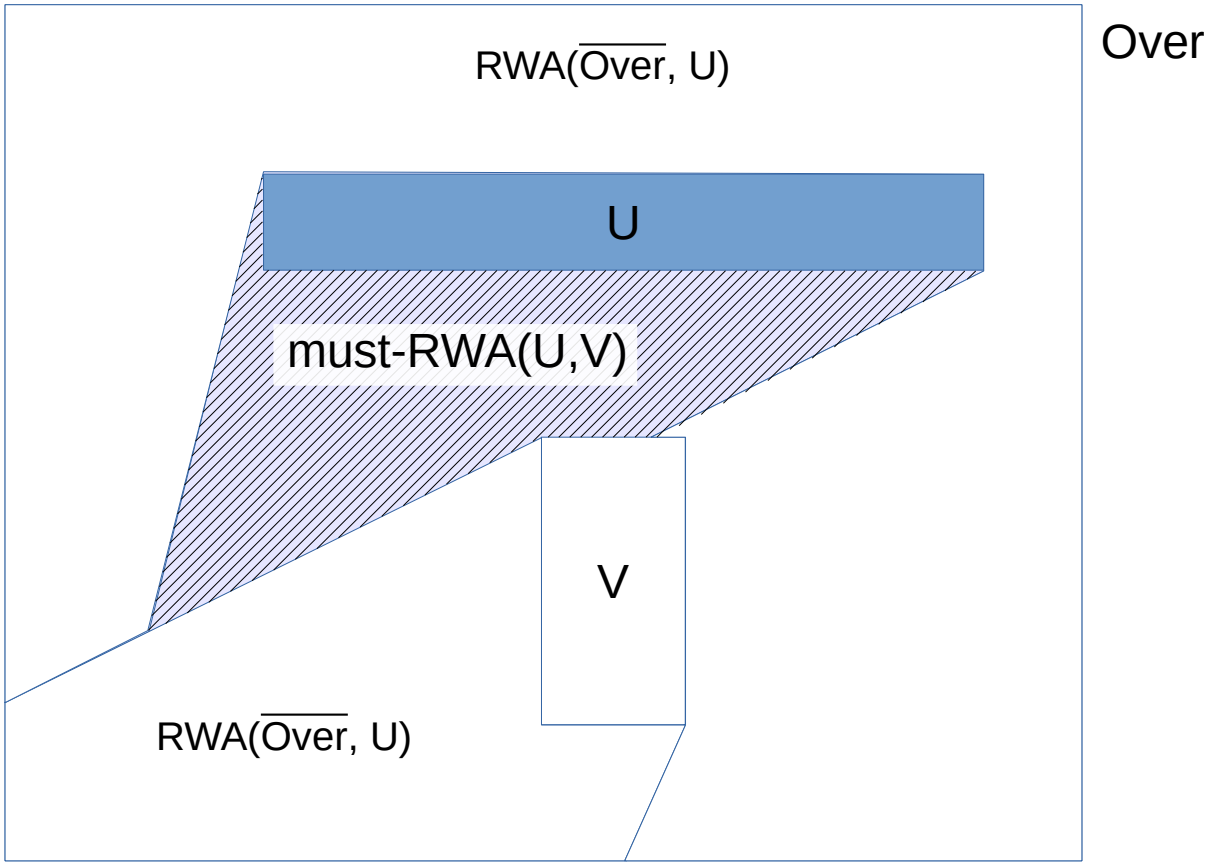
Computing must-RWA Using RWA

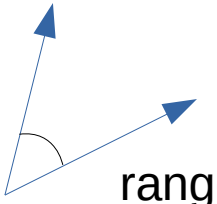




range of possible directions

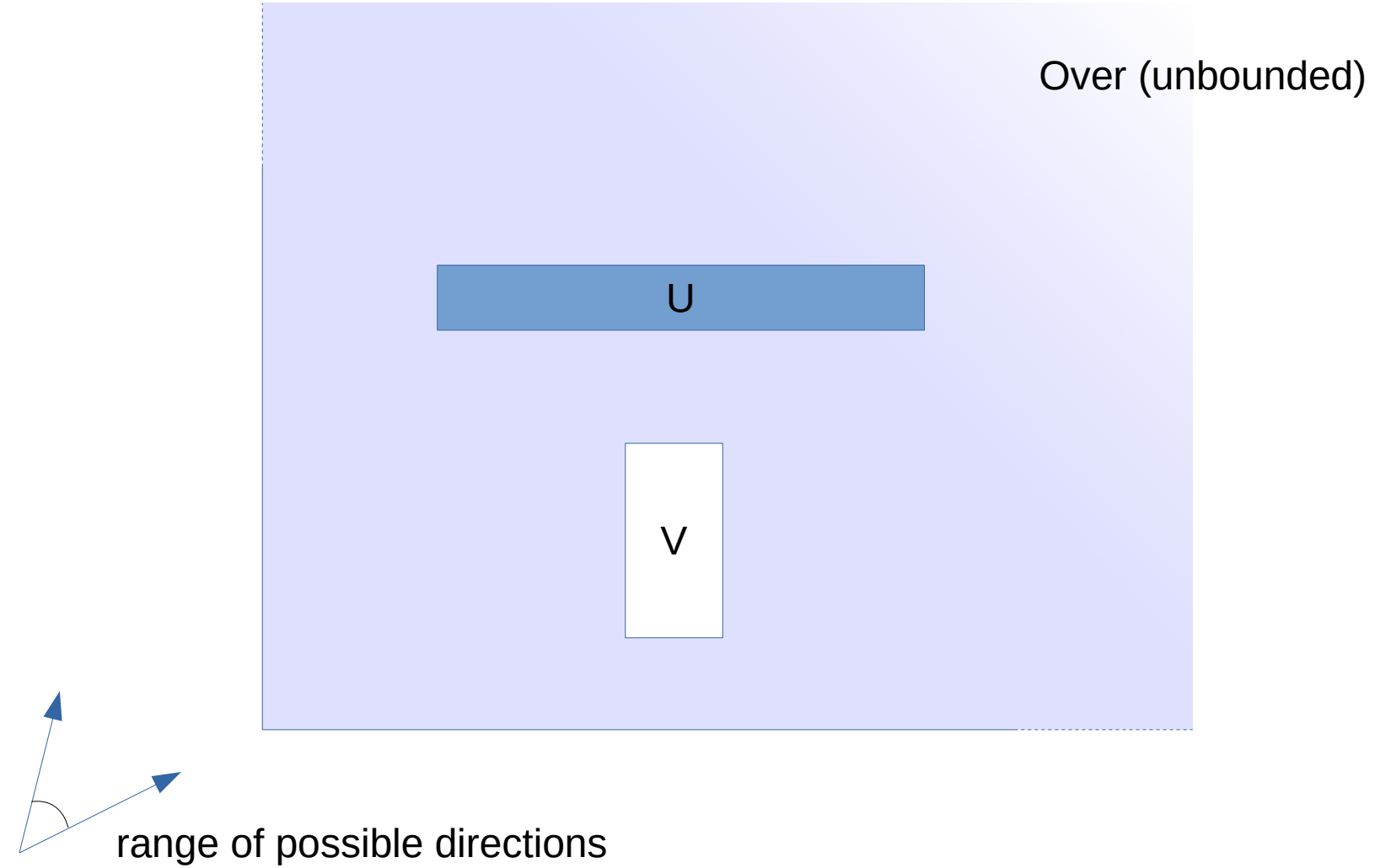
Computing must-RWA Using RWA



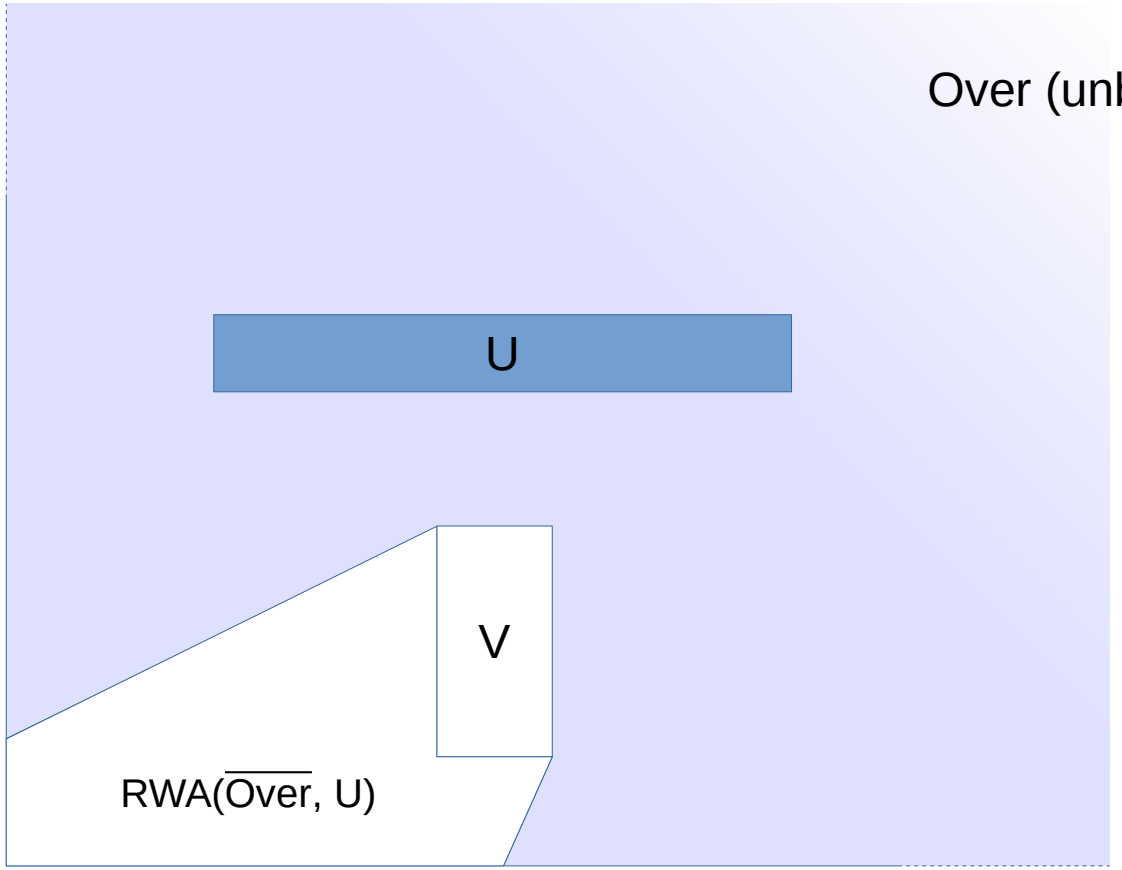


range of possible directions

Why the Over-approximation Must Be Bounded

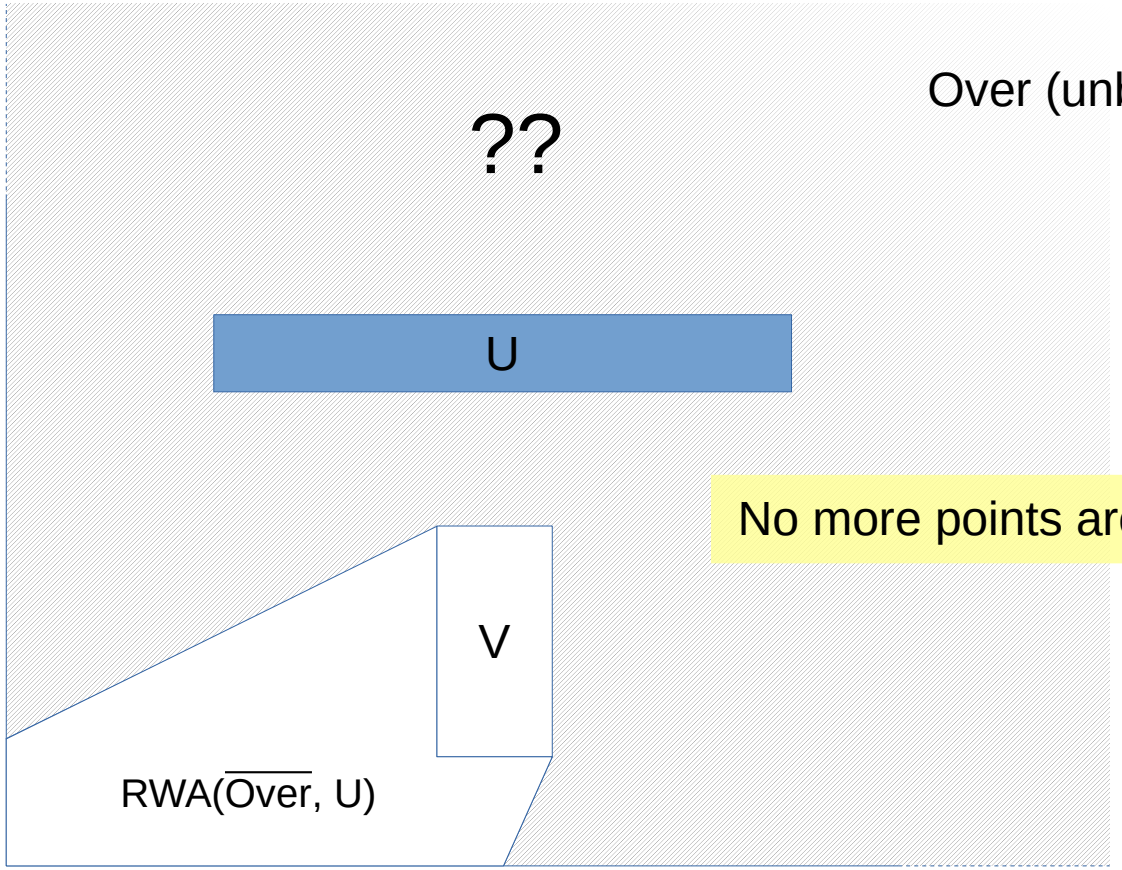


Why the Over-approximation Must Be Bounded

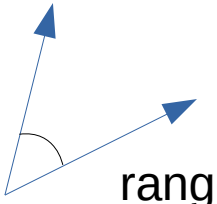


range of possible directions

Why the Over-approximation Must Be Bounded



No more points are removed!



range of possible directions

Computing the Over-approximation

- “not V” is an obvious over-approximation
  - however, its unsafe part Over\U is not necessarily q-**bounded**
- How do we make it q-bounded while preserving all good points?
  - *How do we distinguish good points?*

- **Idea:**

  - The **good points** in Over\U are all q**-bounded**
    - because they have to reach U
  - Hence, **preserve the q-bounded points!**

Computing the Over-approximation

- Introducing the “Remove-Unbounded” operator **RU(G)**
  - removes some convex polyhedra from G
  - such that all q-bounded points are kept
  - and the result is q-bounded

- RU(G) can be computed using basic operations on polyhedra

more info in [Benerecetti & F., ACM TECS, 2017]

References on Controller Synthesis

- T.A. Henzinger. **The theory of hybrid automata**. *LICS 96*, IEEE Computer Society Press.
- C.J. Tomlin, J. Lygeros, S. Shankar Sastry. **A game theoretic approach to controller design for hybrid systems**. *Proceedings of the IEEE*, 88(7), 2000.
- E. Asarin et al. **Effective synthesis of switching controllers for linear systems.** *Proceedings of the IEEE, 88(7), 2000.*
- O. Maler. **Control from Computer Science**. *IFAC Annual Reviews in Control*, 2003.

References on LHGs

- H. Wong-Toi. **The synthesis of Controllers for Linear Hybrid Automata**. *CDC 97*. IEEE.
- T.A. Henzinger, B. Horowitz, R. Majumdar. **Rectangular hybrid games**. *CONCUR 99*, LNCS 1664, Springer 1999.

Recent on LHGs

- M. Benerecetti, M. Faella. **Automatic Synthesis of Switching Controllers for Linear Hybrid Systems: Reachability Control**. *ACM Trans. on Embedded Computing Systems*, 16(4), 2017.
- M. Benerecetti, M. Faella. **Automatic Synthesis of Switching Controllers for Linear Hybrid Systems: Safety Control**. *Theoretical Computer Science*, 493. Elsevier, 2013.
- M. Benerecetti, M. Faella. **Tracking Differentiable Trajectories across Polyhedra Boundaries**. *HSCC 2013*.
- M. Benerecetti, M. Faella, S. Minopoli. **Reachability Games for Linear Hybrid Systems**. *HSCC 2012*.
- M. Benerecetti, M. Faella, S. Minopoli. **Revisiting Synthesis of Switching Controllers for Linear Hybrid Systems**. IEEE CDC 2011.

Research in Hybrid Systems

- Application domains:
  - cyber-physical systems
  - modeling of biological systems
- Conferences:
  - specific: HSCC (part of the week on Cyber-Physical Systems), IEEE CDC, FORMATS, Workshop on Hybrid Systems and Biology
  - generic: CAV, CONCUR, TACAS
- Journals: TCS, ACM Trans Embedded Comp Sys (TECS), IEEE Trans Aut Control
- Tools: SpaceEx, Hsolver, Ariadne, KeYmaera

Directions of Further Research

Controllers can be evaluated according to:

1) **Performance** (how well they perform their duties)
   - idea: synthesize an **“optimal”** controller
   - problem: what are good notions of “optimal”?
     - also see: optimal control theory
   - for example, a controller that keeps the system **as far away as possible** from the boundaries of the safe region
2) **Cost**
   - idea: synthesize an **efficient** controller
   - problem: what is the cost of a controller?
   - for example, its **memory** footprint