Controller Synthesis for Linear Hybrid Systems

Part 2: Verification of LHAs

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

Marco Faella
Università di Napoli “Federico II”

Lesson Summary

- Symbolic operations on polyhedra
- Linear Hybrid Automata
- An exact semi-algorithm for the BRP

Convex Sets

- A **convex combination** of two points *p* and *q* is a point of the type

$$a \cdot p + (1-a) \cdot q, \qquad \text{for all a in } [0, 1]$$

- A set is **convex** if it is closed under convex combinations

- Examples:
  - a rectangle is convex
  - a disk is convex, whereas a circle is *not*
  - an L-shaped polygon is *not* convex
  - the empty set and the universe set are convex

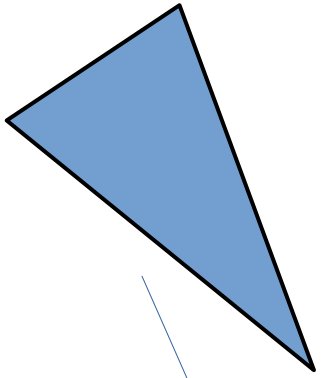
Convex Polyhedra

- A conjunction of (finitely many) linear inequalities
- Inequalities may be strict (<) and non-strict ($\leq$)
- Examples:
  - a rectangle is a convex polyhedron
  - a cube is a convex polyhedron
  - a cone is a convex set but *not* a convex polyhedron

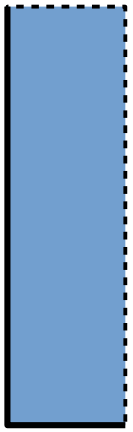
Types of Convex Polyhedra

- (topologically) **closed** if all inequalities are non-strict
- " **open** if all inequalities are strict

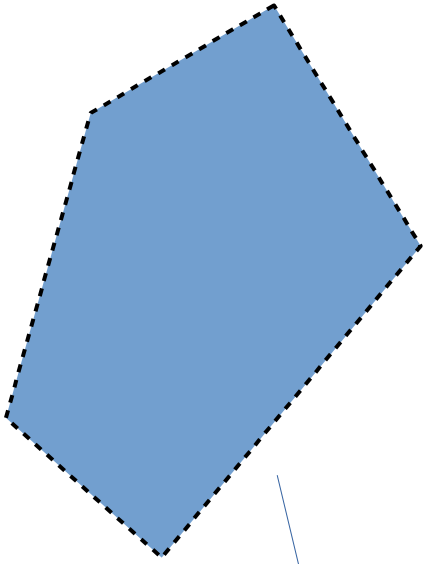
- **Bounded** if it is included in a hyper-sphere
- **Unbounded** otherwise



a *closed* convex polyhedron



a convex polyhedron



an *open* convex polyhedron

Basic Operations on Polyhedra (1)

- **Intersection**: P1 ∩ P2
- Implementation: merge the sets of constraints

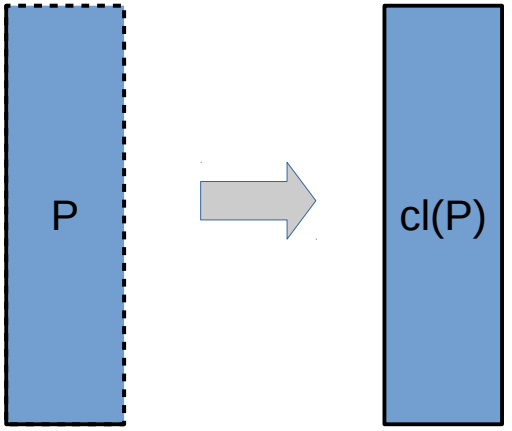
- Notice: convex polyhedra are not closed under union or complement

Basic Operations on Polyhedra (2)

- Topological **closure**:

cl(P)

- Implementation: convert all constraints into non-strict



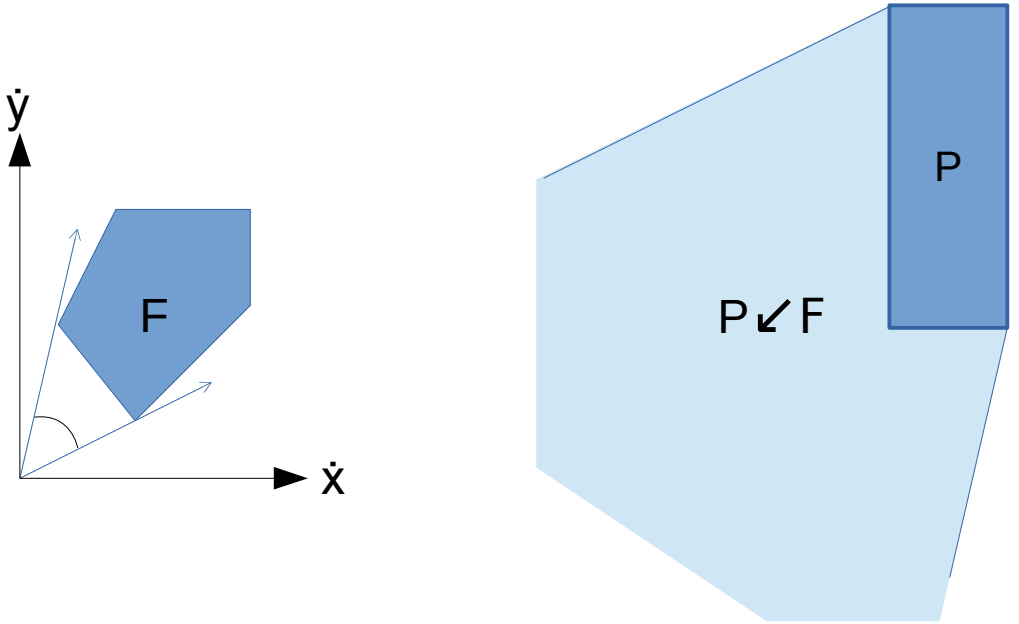
Basic Operations on Polyhedra (3)

- **Pre-flow** of P w.r.t. F:

$$P \swarrow F \quad \text{(or simply } P \swarrow \text{)}$$

The set of points that can reach P using a direction in F

Formally, $P \swarrow F = \{ p - t \cdot c \mid p \text{ in } P, c \text{ in } F, t \geq 0 \}$



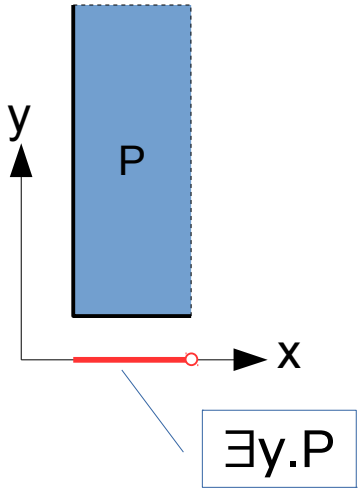
Exercise

Find two convex polyhedra P and F such that P↙F is *not* a convex polyhedron.

Basic Operations on Polyhedra (4)

- **Projection** (removal of variables):

$$\exists y.P$$



All provided by libraries like Parma Polyhedra Library (PPL)

Basic Operations on Polyhedra (5)

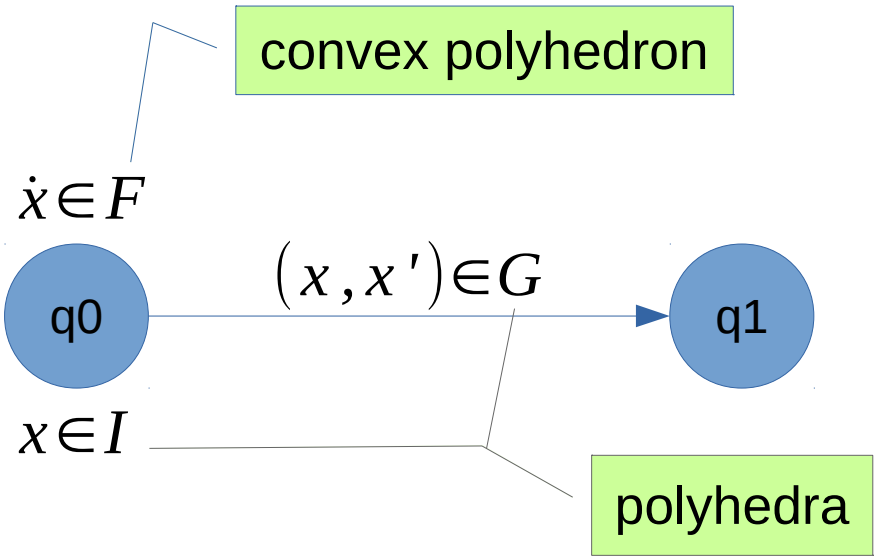
- **Variable renaming**

- Jump relations are defined over two sets of variables X and X’
- We need to convert polyhedra to the primed variables

- So:

prime(P) rename variables of P from X to X'

Linear Hybrid Automata



Polyhedral Sets of States

- For a set of states Z and a location q, let **Z(q)** be the set

$$\{ x \mid (q,x) \text{ in } Z \}$$

- We say that Z is **polyhedral** if Z(q) is a (possibly non-convex) polyhedron for all locations q

An exact semi-algorithm for BRP on LHAs

- Given an LHA and a polyhedral set of states E (error), compute the set of states that can reach E

- We build (sets of) runs backwards
- Given a polyhedral set of states Z, we simulate discrete steps and timed steps backwards (*symbolic execution*)

- For discrete steps:
  - **PreJmp**(Z) = states that can reach Z via a discrete step
- For timed steps:
  - **PreTime**(Z) = states that can reach Z via a timed step

An exact semi-algorithm for BRP on LHAs

- The solution to BRP is:

$$Z^* = \mu Z \,.\, E \cup \mathrm{PreJmp}(Z) \cup \mathrm{PreTime}(Z)$$

- By fixpoint theorem, Z^* is the limit of the sequence

$Z0 = \emptyset$

$Z1 = E \cup \mathrm{PreJmp}(Z0) \cup \mathrm{PreTime}(Z0)$

...

- **Fact**: When Z is polyhedral, PreJmp and PreTime can be effectively computed and their result is polyhedral
- However, reachability is undecidable, so the above sequence may not converge in a finite number of steps!

Defining PreJmp

Let Z be a polyhedral set of states

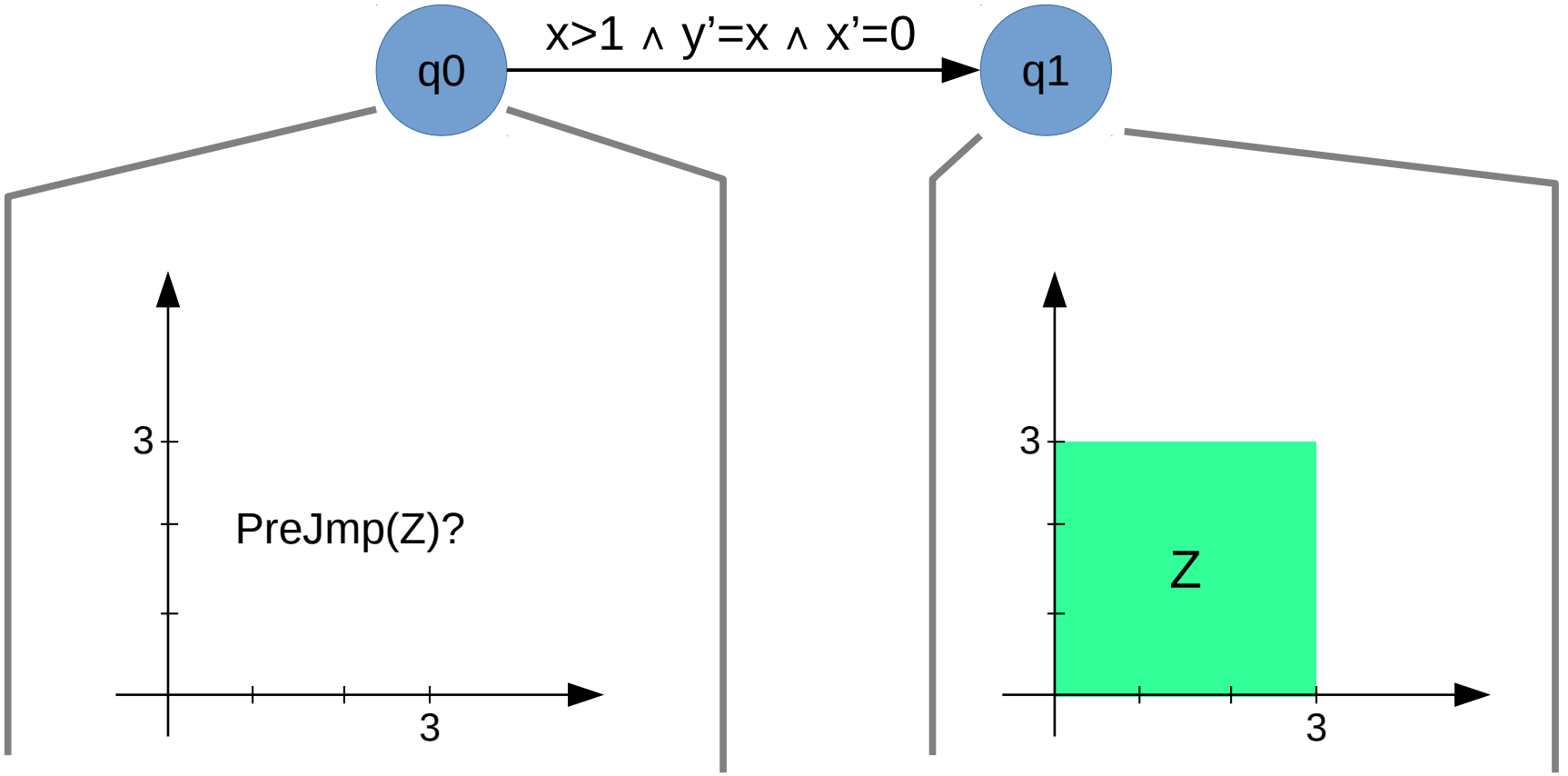
Let $j = (q, G, q') \in Jmp$

recall that G is a polyhedron on $\mathbb{R}^{2n}$

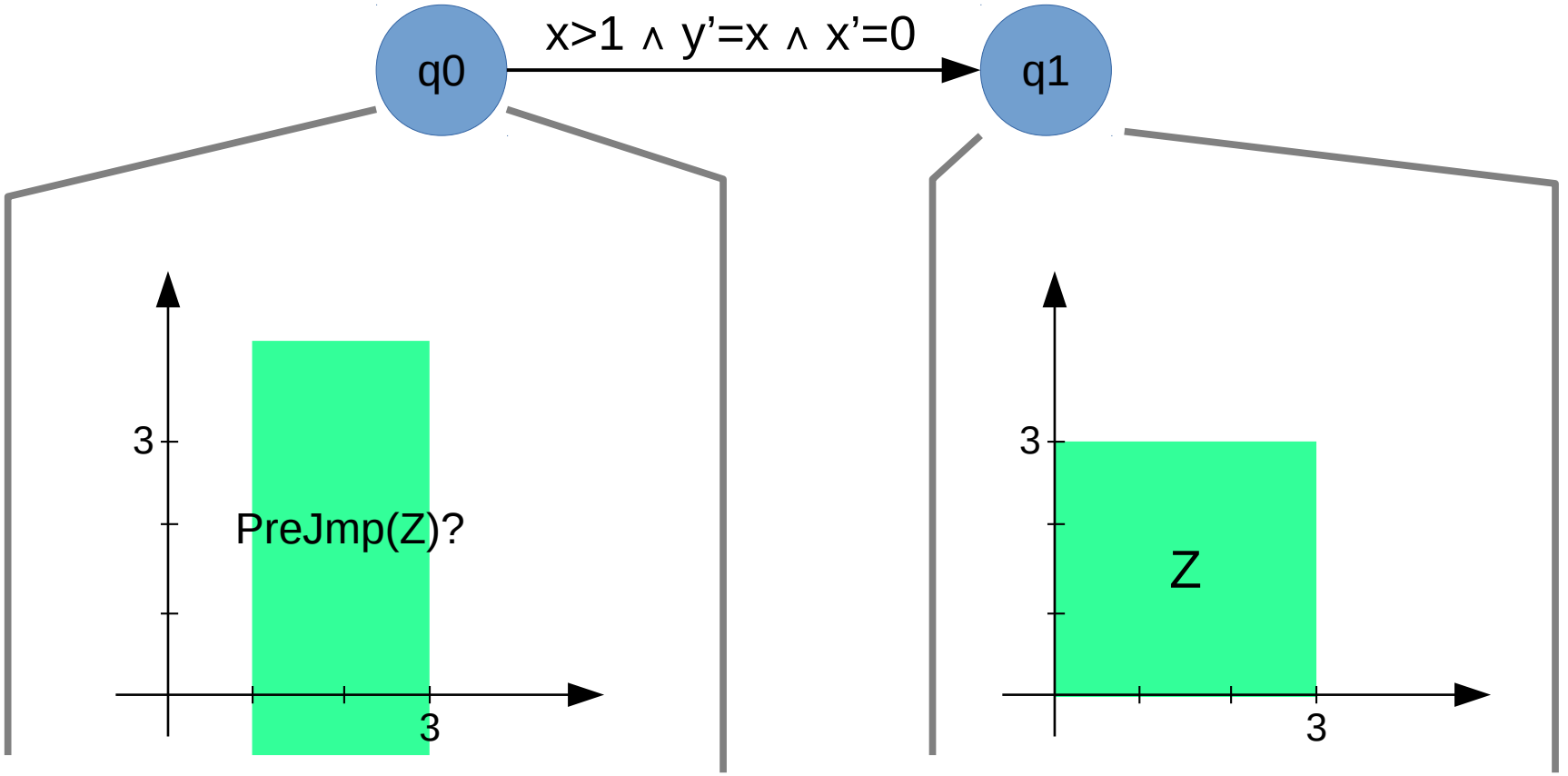
Define:

- **PreJmp(Z, j)** = $\{ (q,x) \mid (x,x') \in G \text{ for some } (q',x') \in Z \}$
- $PreJmp(Z) = \bigcup_{j \in Jmp} PreJmp(Z, j)$

PreJmp Example



PreJmp Example



Computing PreJmp

Let Z be a polyhedral set of states

Let $j = (q, G, q') \in Jmp$

We compute PreJmp(Z, j) as follows:

$$PreJmp(Z, j) = \{q\} \times \exists X'.\left(G \cap prime(\ Z(q')\ )\right)$$

where X' is the set of all primed variables

Computing PreTime

- PreTime(Z) is the set of states (q,x) from which there is a trajectory that reaches Z while staying in Inv(q) and satisfying Flow(q)

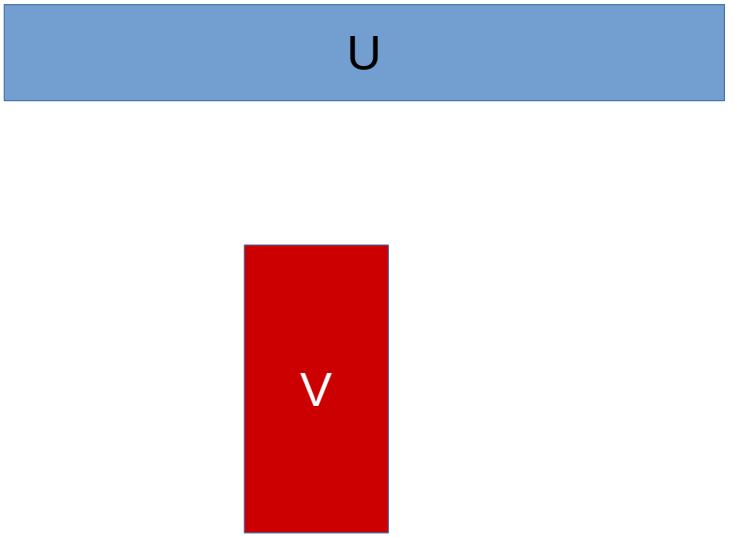
- First, assume that: $Z = \{q\} \times U$ and **Inv(q) is convex**
- Then,

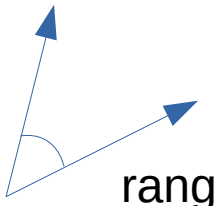
$$\text{PreTime}(Z) = (U \swarrow \text{Flow}(q)) \cap \text{Inv}(q)$$

- What if Inv(q) is not convex?
- The above formula does not work

PreTime and non-Convex Invariants

Inv(q) = $\overline{V}$ (i.e., V is a “hole” in the invariant)

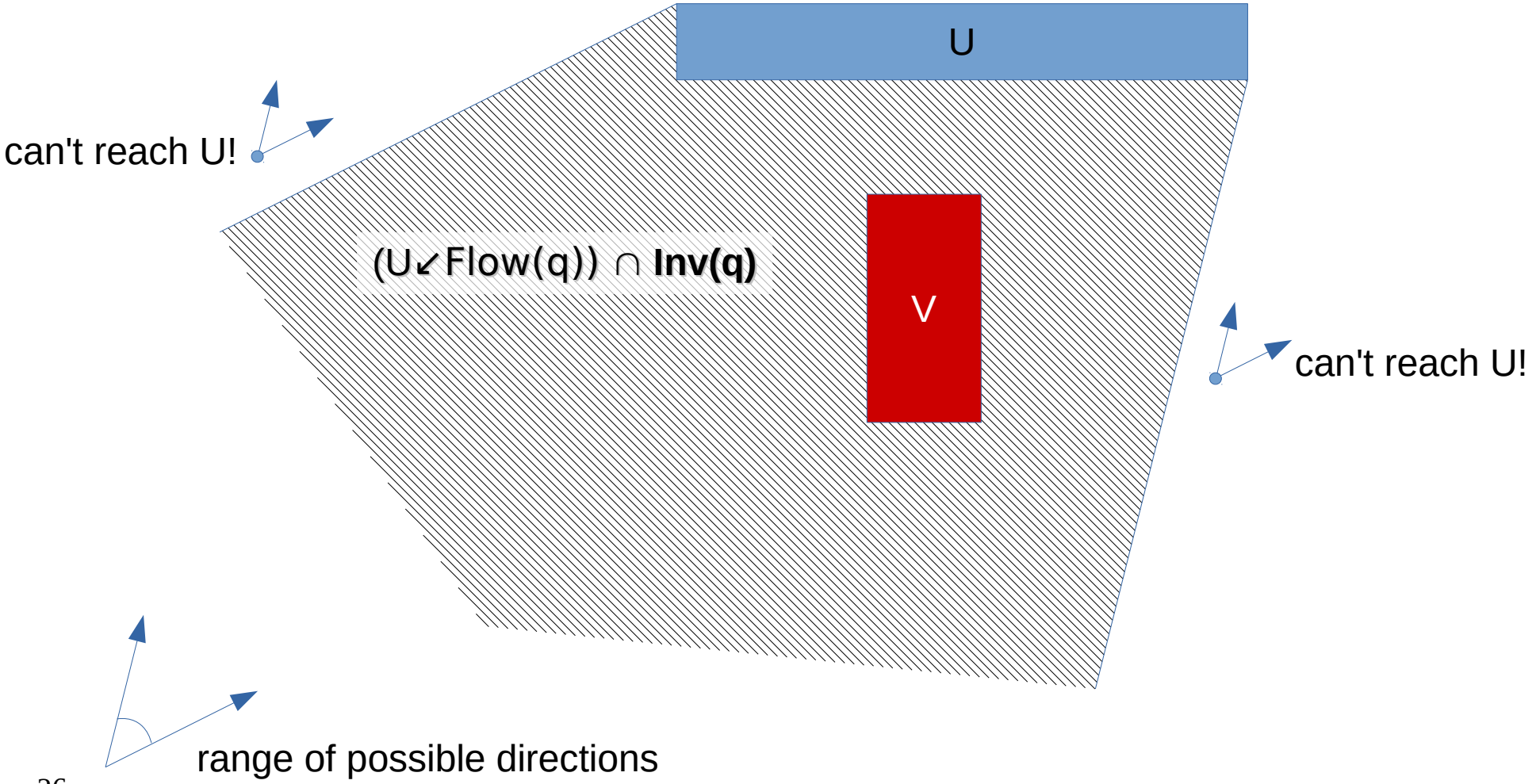




range of possible directions

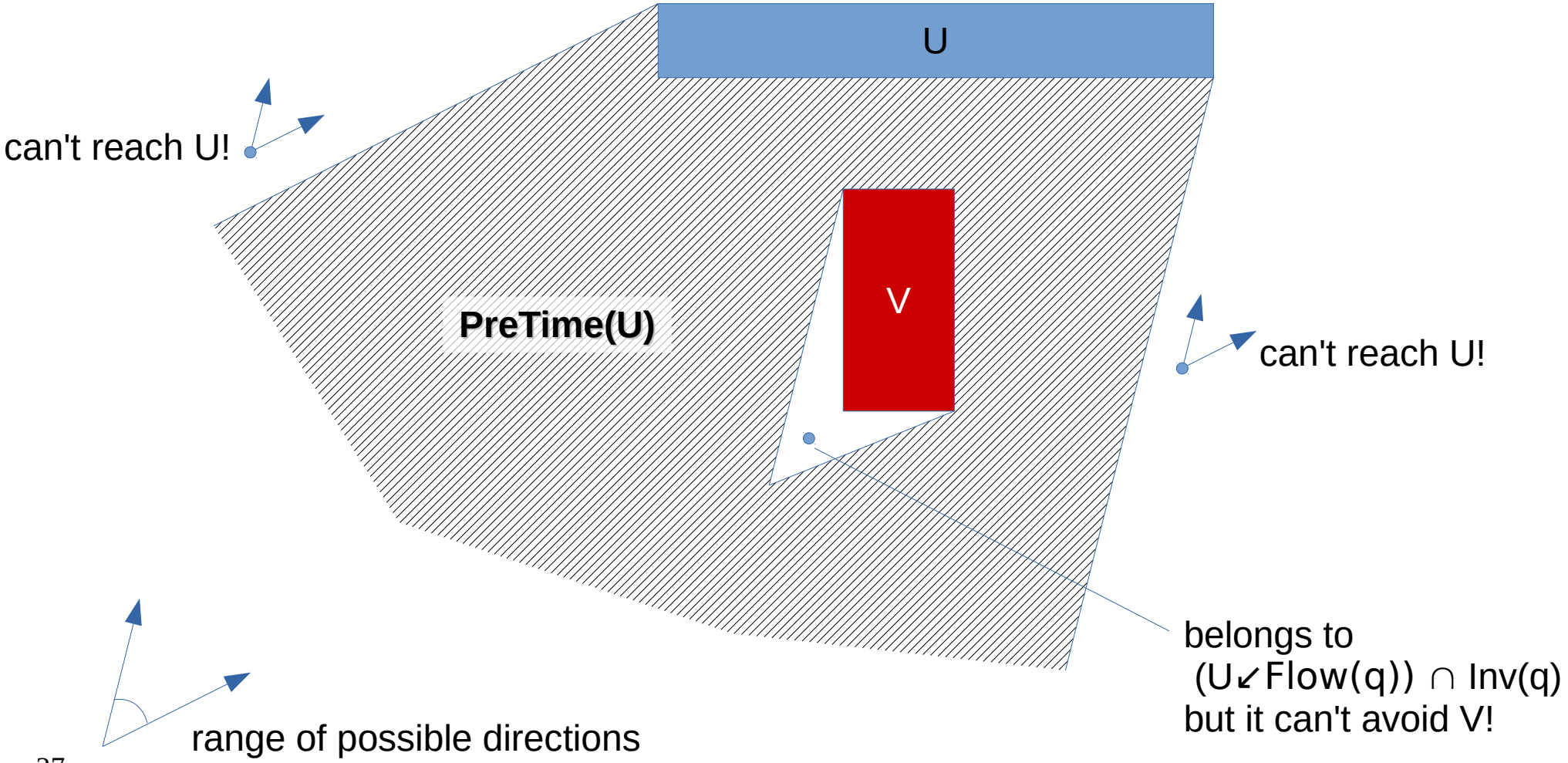
PreTime and non-Convex Invariants

Inv(q) = $\overline{V}$ (i.e., V is a “hole” in the invariant)



PreTime and non-Convex Invariants

Inv(q) = $\overline{V}$ (i.e., V is a “hole” in the invariant)



Reach-While-Avoiding Operator

Fix a location q and the corresponding dynamics Flow(q).

Definition. Given two polyhedra U and V,

RWA(U, V) is the set of points from which there is a trajectory that:

- reaches U
- while avoiding V at all times.

- a.k.a.: *flow_avoid* in [Wong-Toi,97], *Reach* in [Tomlin et al.,00]
- in temporal logic (CTL): **∃ $\overline{\mathbf{V}}$ *Until* U**

PreTime and RWA

PreTime is a special case of RWA, i.e.:

$$\text{PreTime}(\{q\} \times U) = \{q\} \times \text{RWA}(U, \overline{\text{inv}(q)})$$

Let's analyze the basic properties of RWA...

Properties of RWA

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

[*bounds 1*] $U \subseteq RWA(U, V) \subseteq \overline{V}$

[*bounds 2*] $RWA(U, V) \subseteq U{\swarrow}$

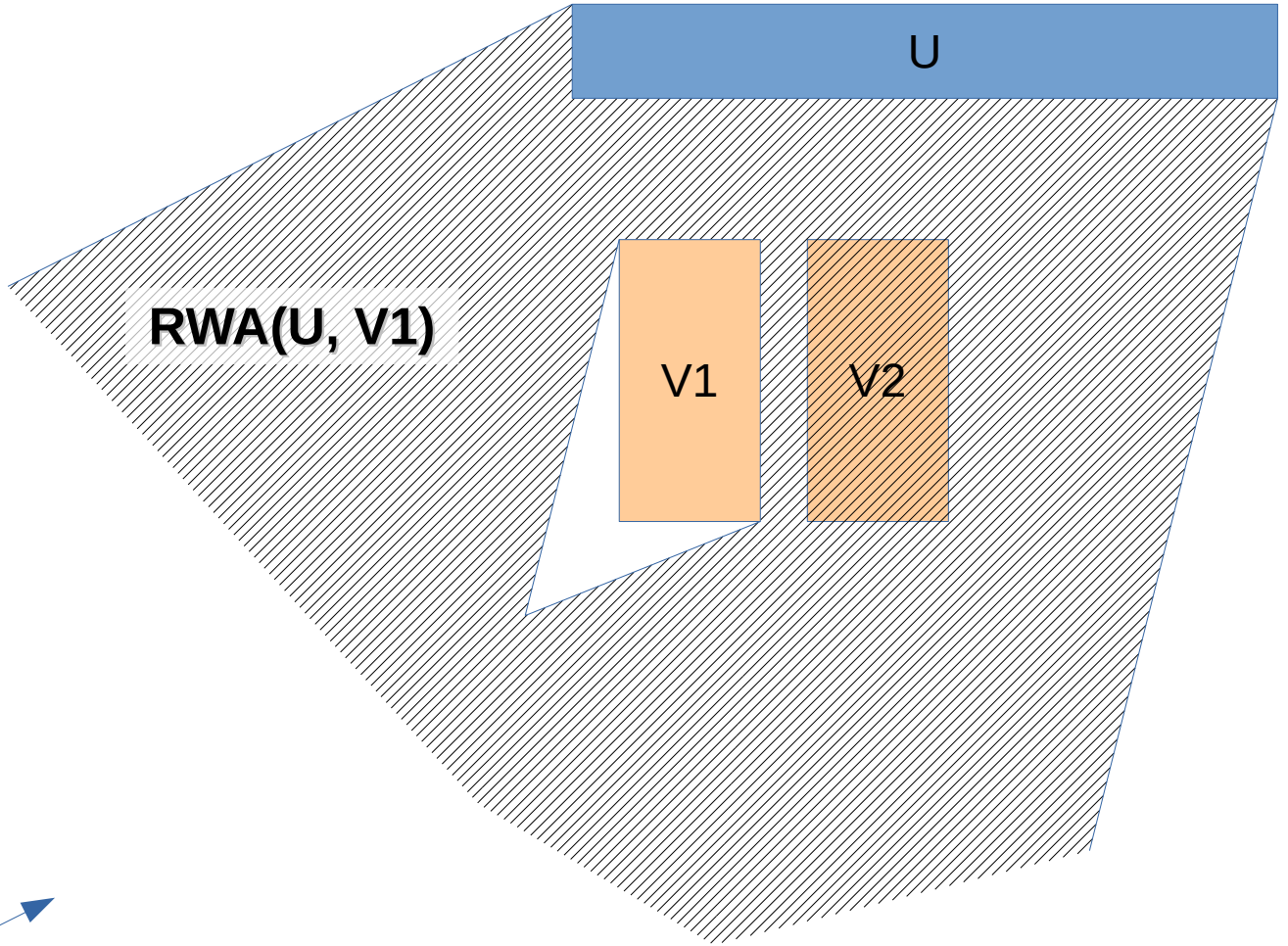
[*distributivity 1st arg*]

$$RWA(U1 \cup U2, V) = RWA(U1, V) \cup RWA(U2, V)$$

[*non-distrib. 2nd arg*]

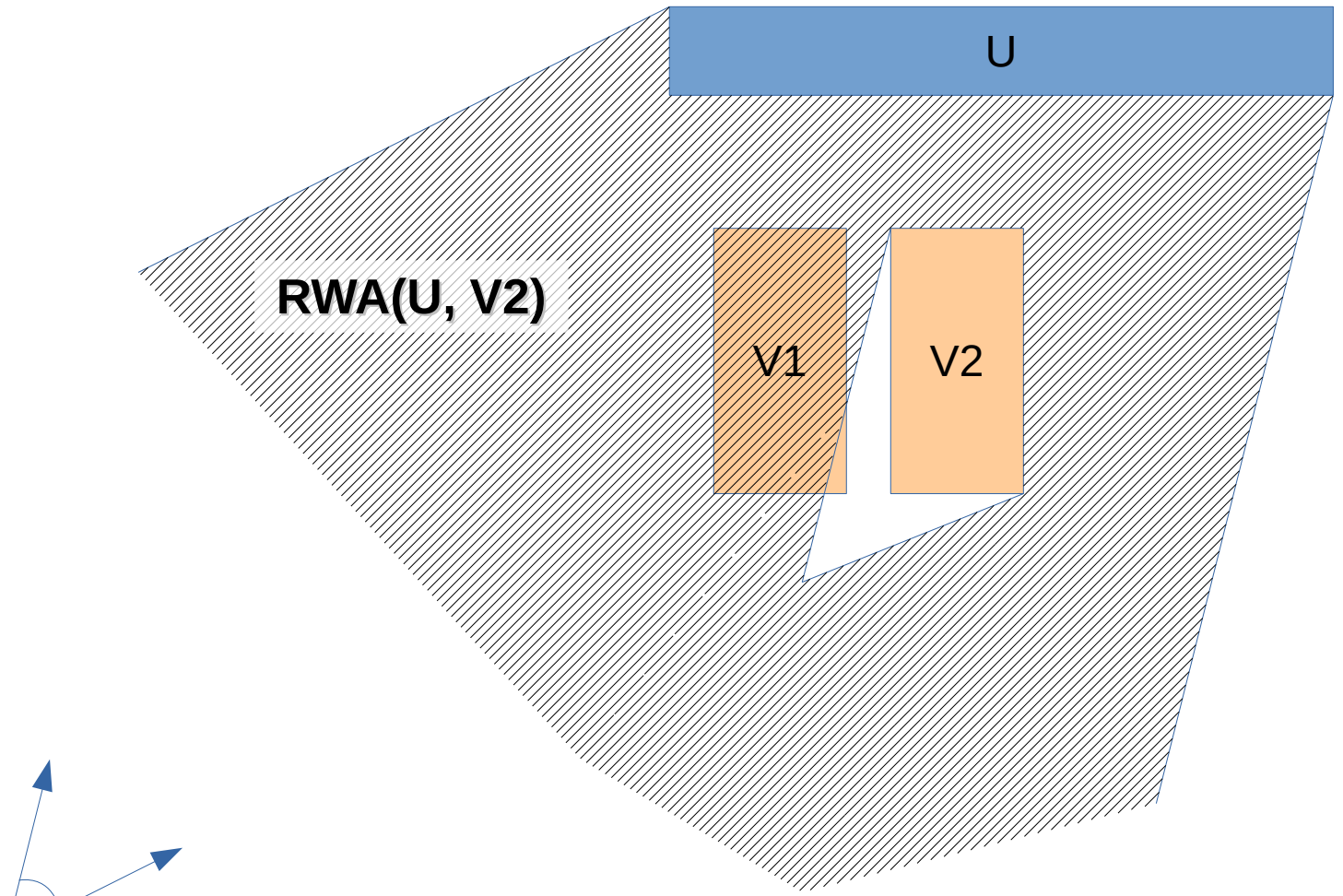
$$RWA(U, V1 \cup V2) \neq RWA(U, V1) \cap RWA(U, V2)$$

Non-distributivity



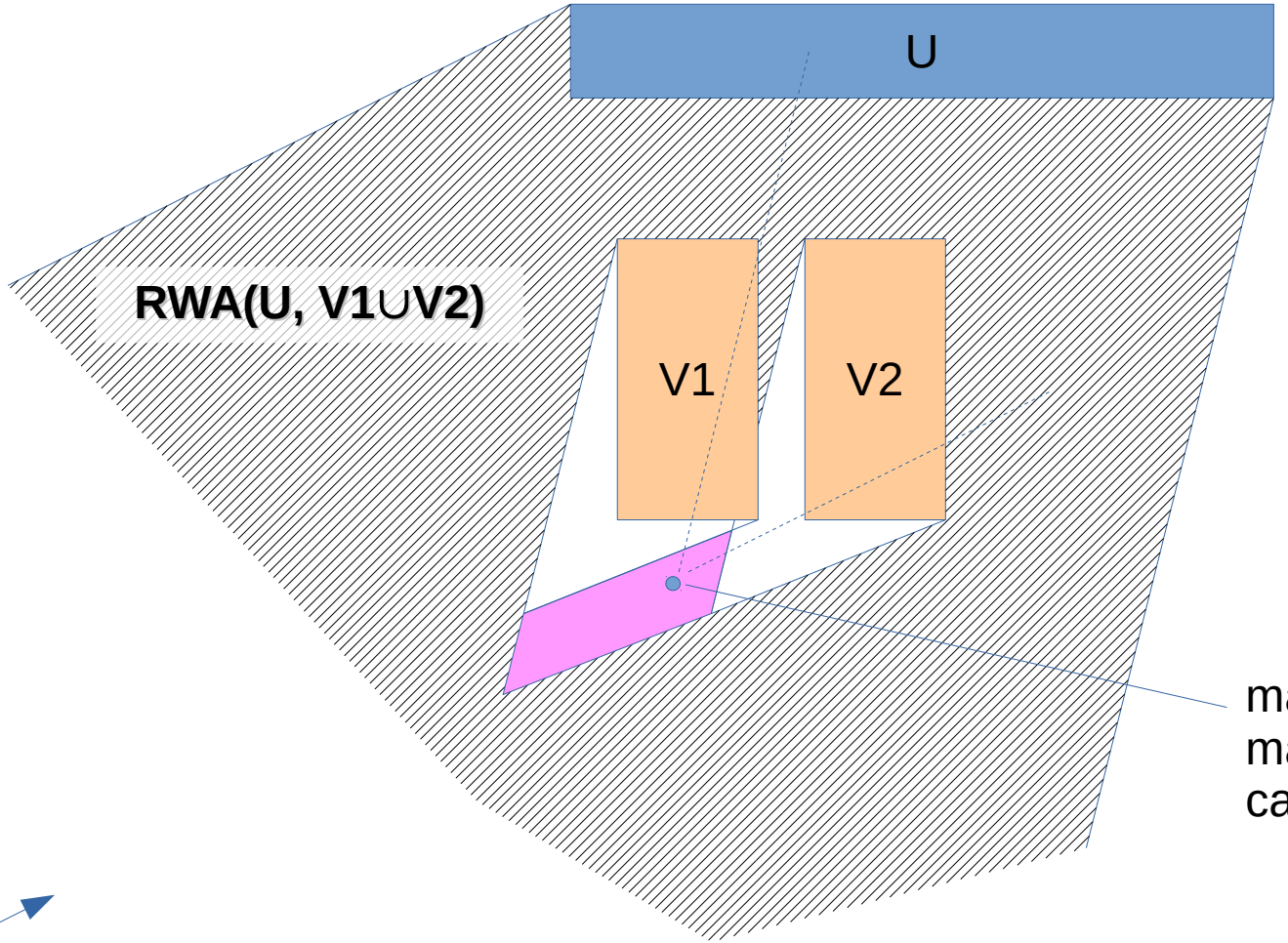
range of possible directions

Non-distributivity

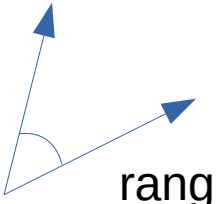


range of possible directions

Non-distributivity



may avoid V1
may avoid V2
can't avoid V1 ∪ V2



range of possible directions

Computing RWA

[*bounds 1*] $U \subseteq RWA(U, V) \subseteq \overline{V}$

[*bounds 2*] $RWA(U, V) \subseteq U\swarrow$

[*distrib*] $RWA(U1 \cup U2, V) = RWA(U1, V) \cup RWA(U2, V)$

We compute RWA incrementally:

$$R_0 \subseteq R_1 \subseteq \ldots \subseteq R_n = RWA(U, V)$$

By [distrib], we can assume that U is convex.

By [bounds 1], we can start with $R_0 = U$.

By [bounds 1 & 2], we only need to add points from $\overline{V} \cap U\swarrow$.

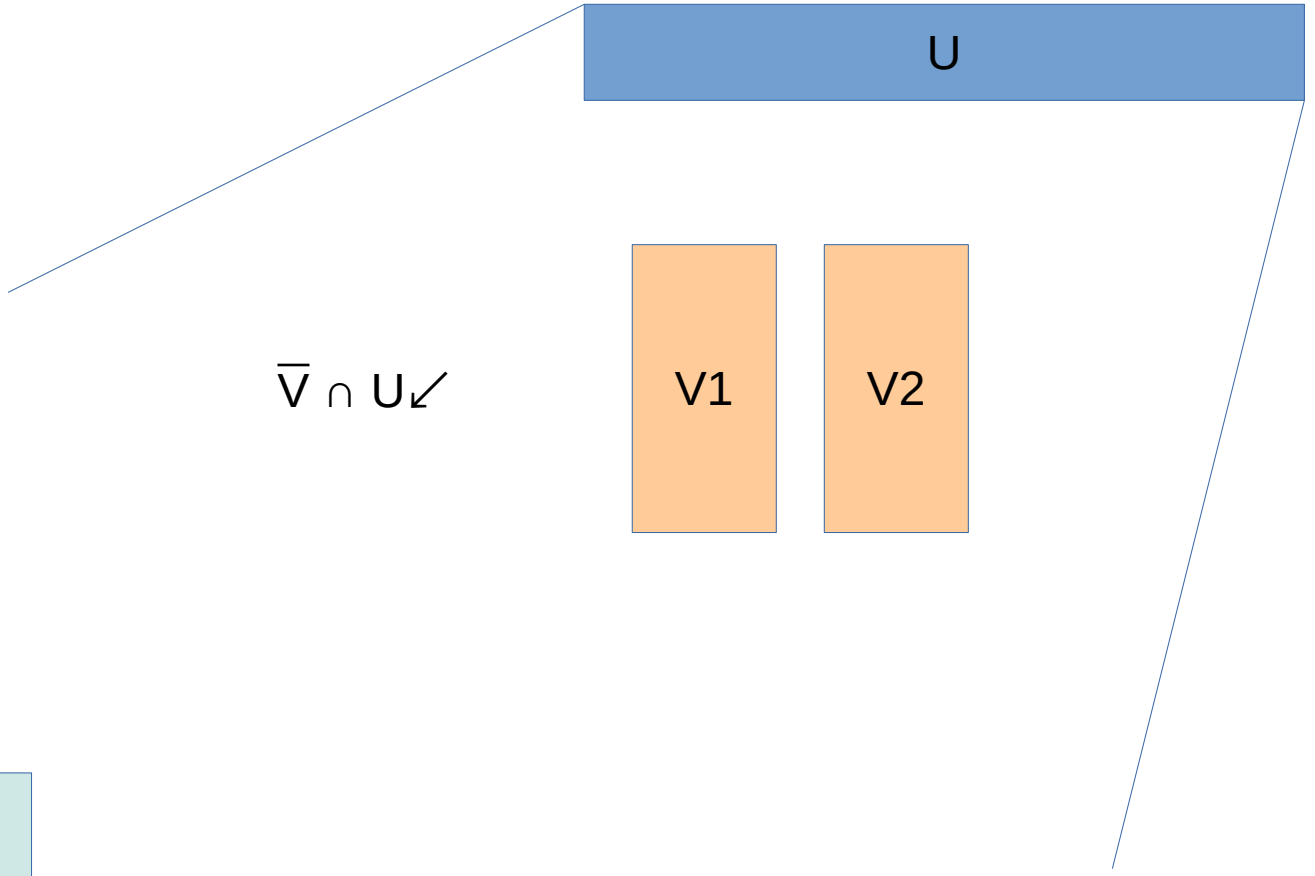
Computing RWA

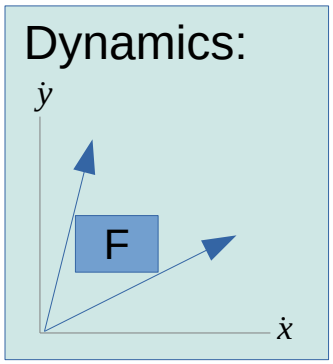
1) Start from $R_0 = U$
2) Let R_i be the current set
3) Pick a convex polyhedron P in $\overline{V} \cap U\swarrow$
4) Pick a convex polyhedron P' in R_i which is **adjacent to P** (if any)
5) Compute the set Q of points of P that can go **directly** into P'
6) $R_{i+1} = R_i \cup Q$
7) Repeat from Step 2, until a fixed point is reached

(This procedure terminates)

Computing RWA(U,V)

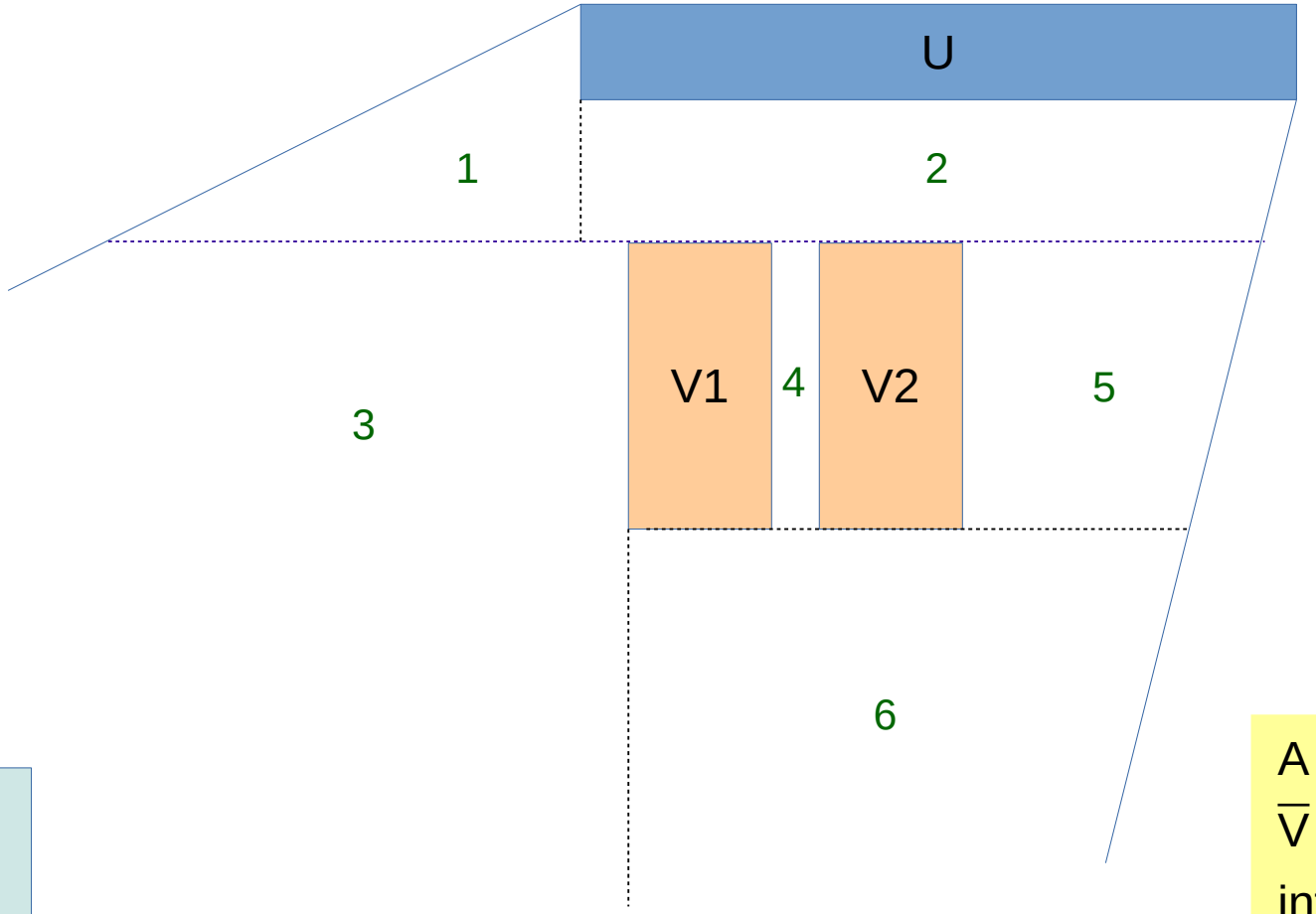
$V = V1 \cup V2$





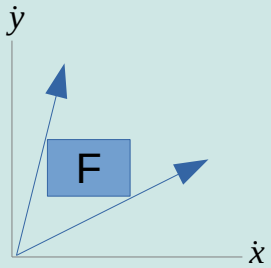
Computing RWA(U,V)

$V = V1 \cup V2$

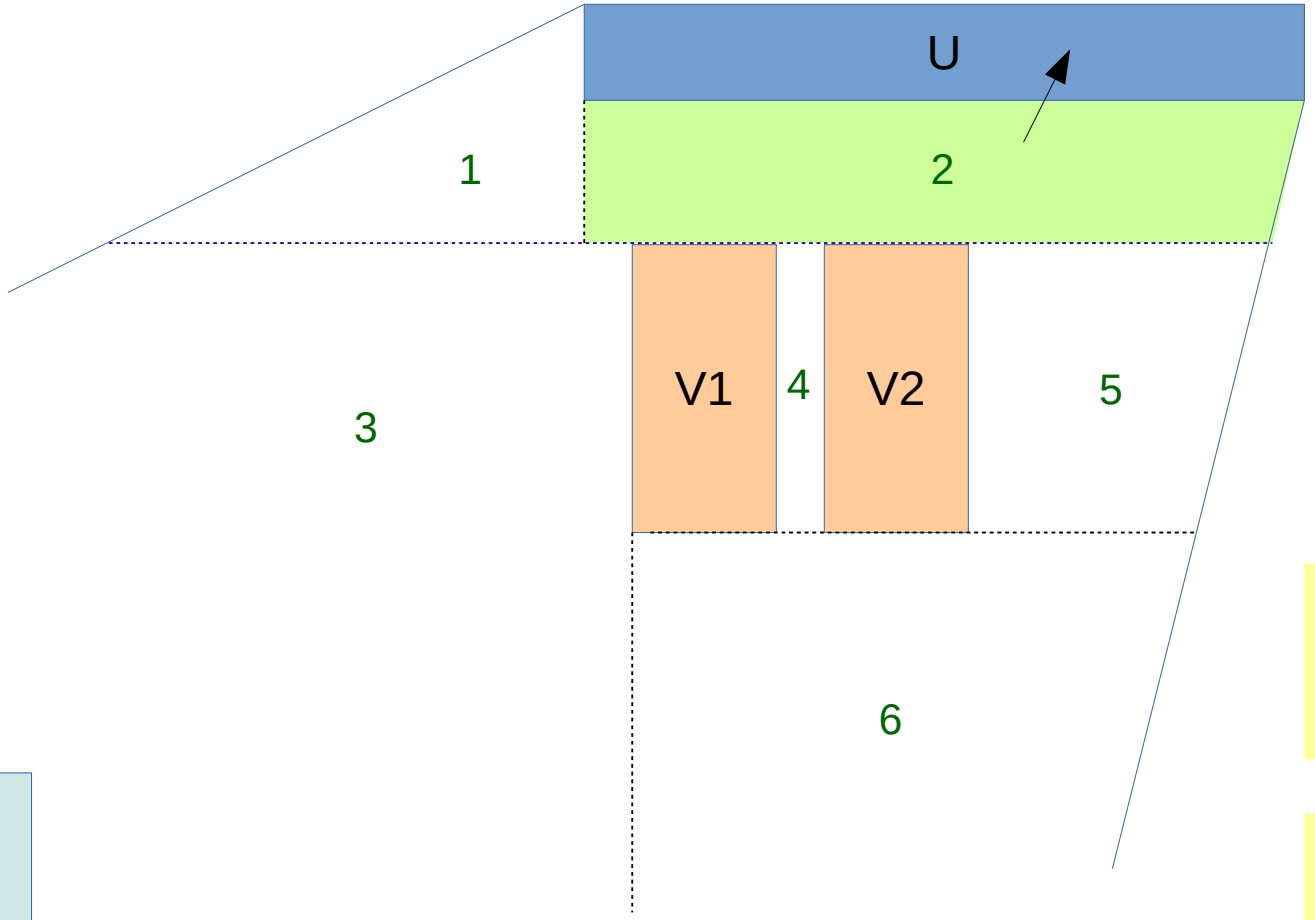


A partition of $\overline{V} \cap U\swarrow$ into 6 convex polyhedra

Dynamics:

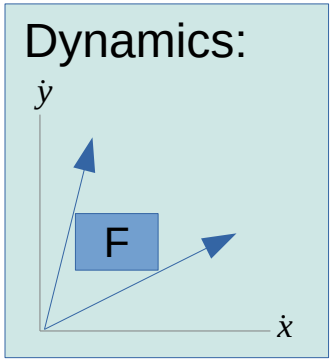


Computing RWA(U,V)

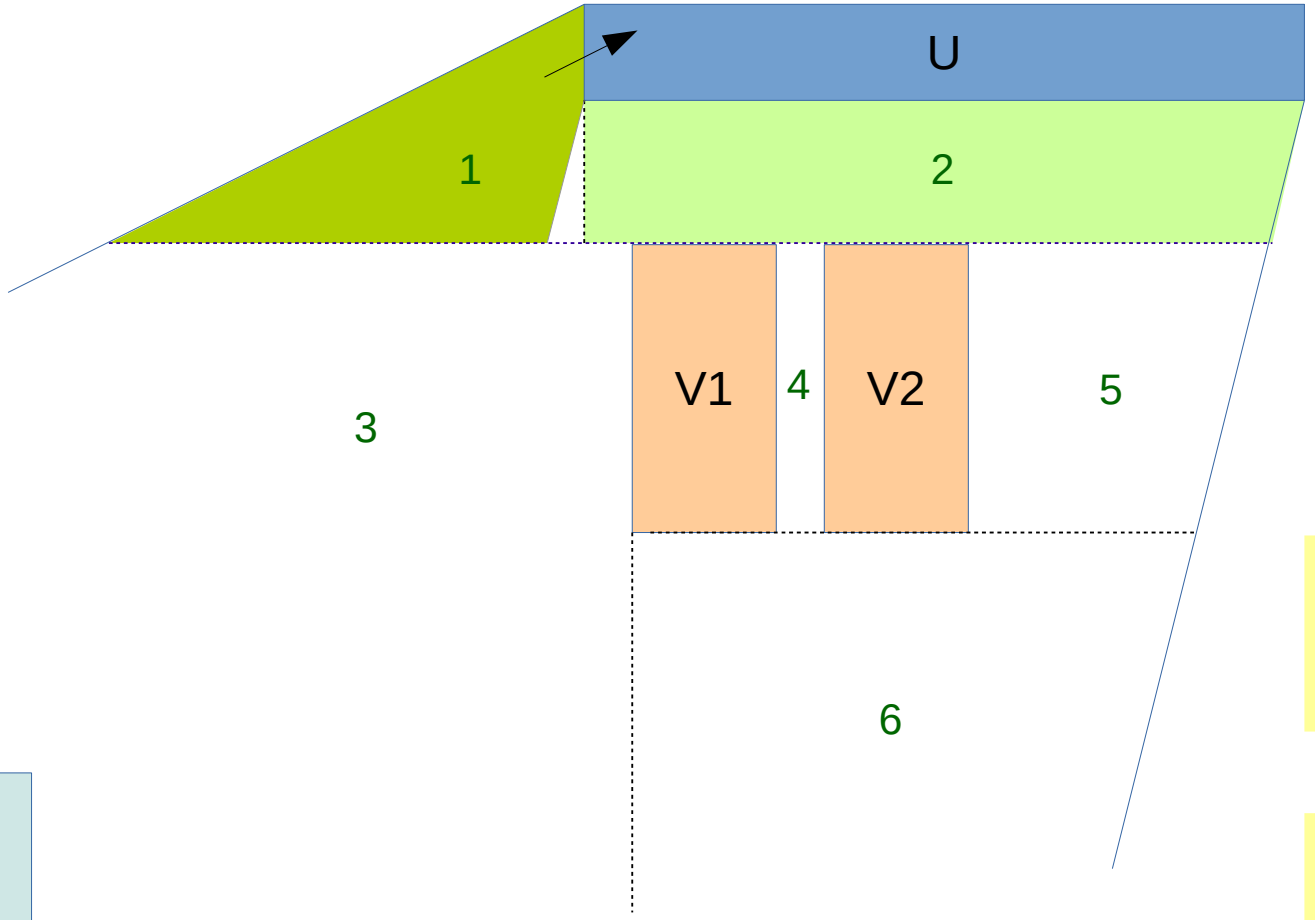


All points of P2 go directly into U

They are added to the result



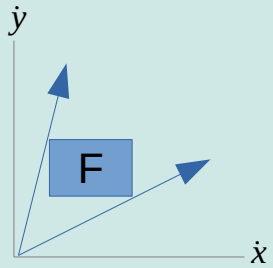
Computing RWA(U,V)



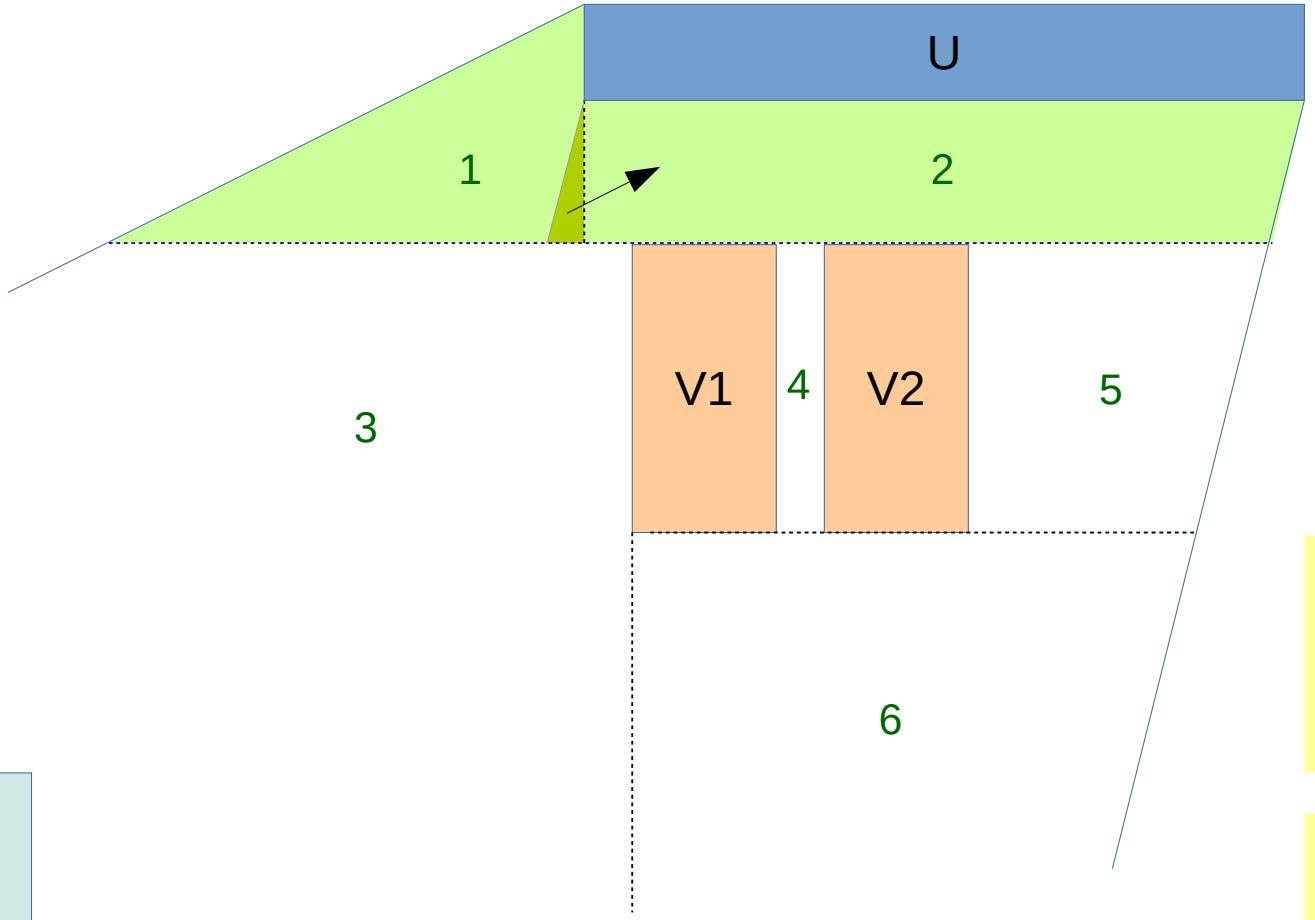
Some points of P1 go directly into U

They are added to the result

Dynamics:

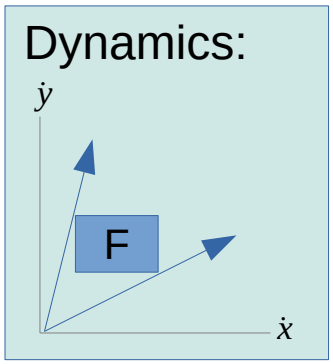


Computing RWA(U,V)

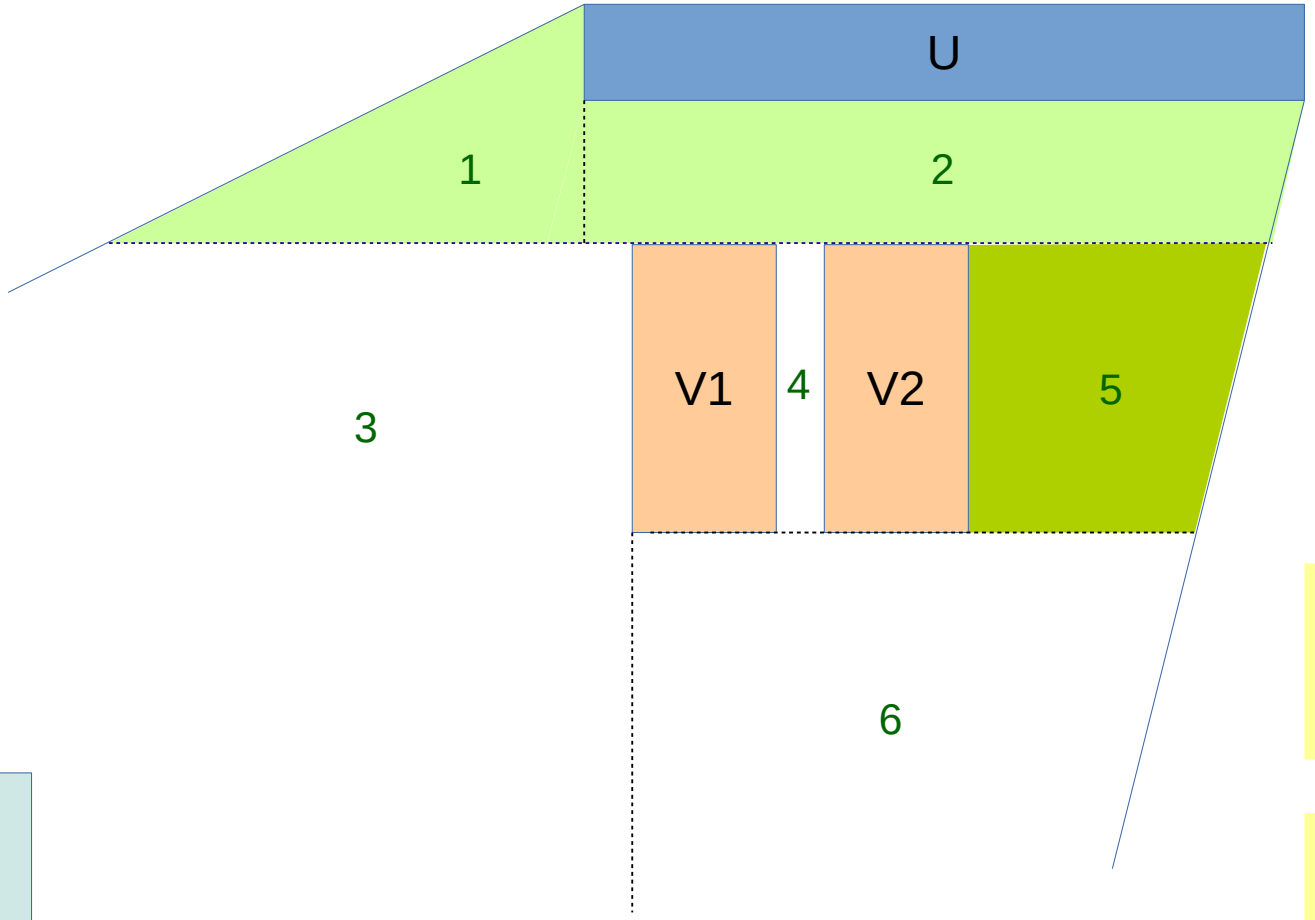


The other points of P1 go directly into P2

They are added to the result



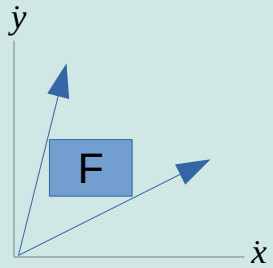
Computing RWA(U,V)



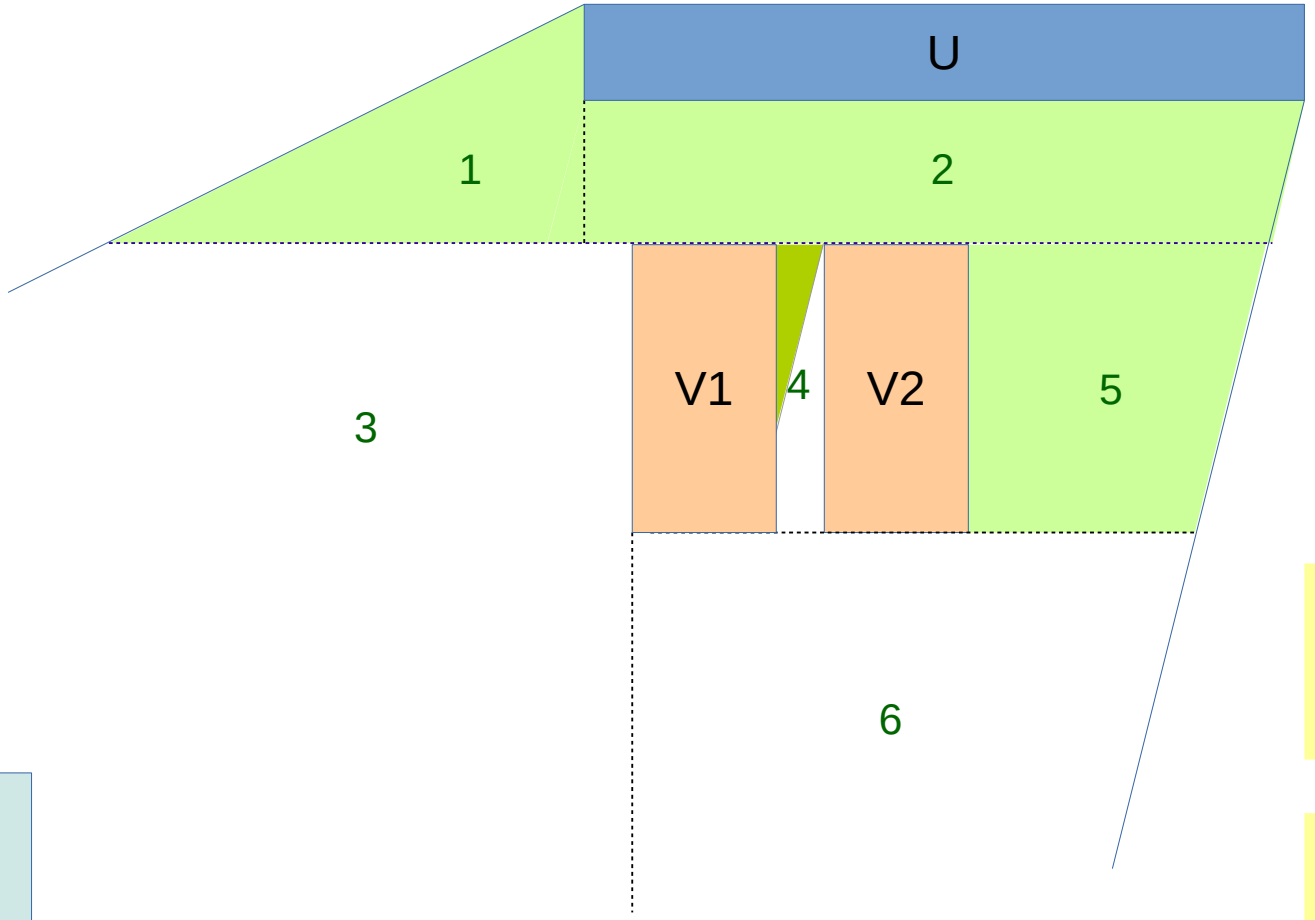
All points of P5 go directly into P2

They are added to the result

Dynamics:



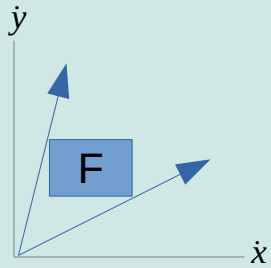
Computing RWA(U,V)



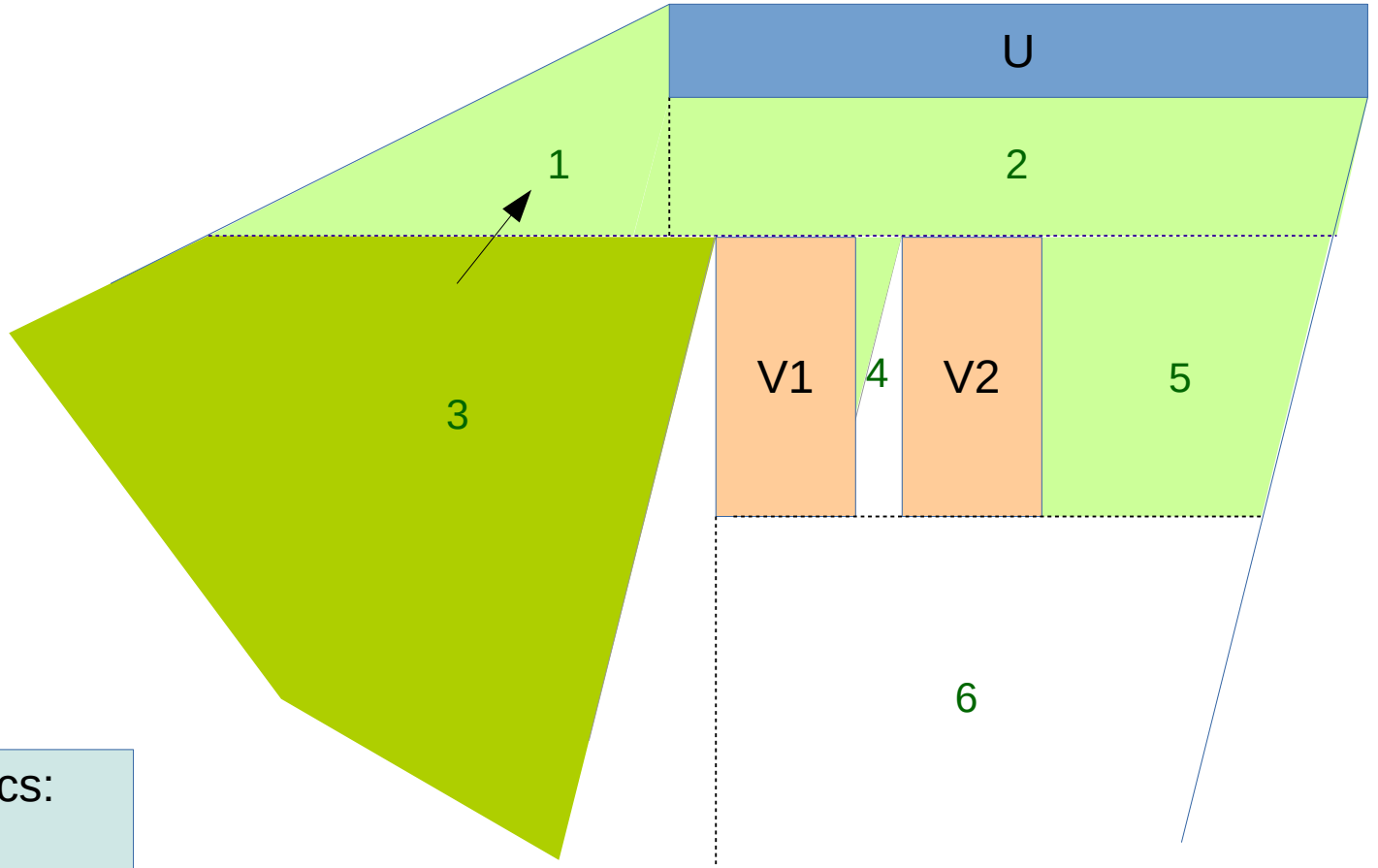
Some points of P4 go directly into P2

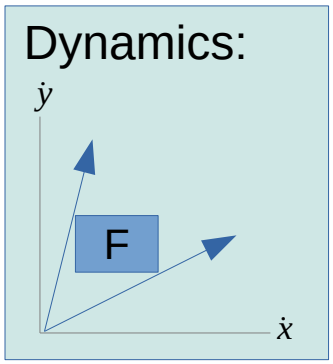
They are added to the result

Dynamics:

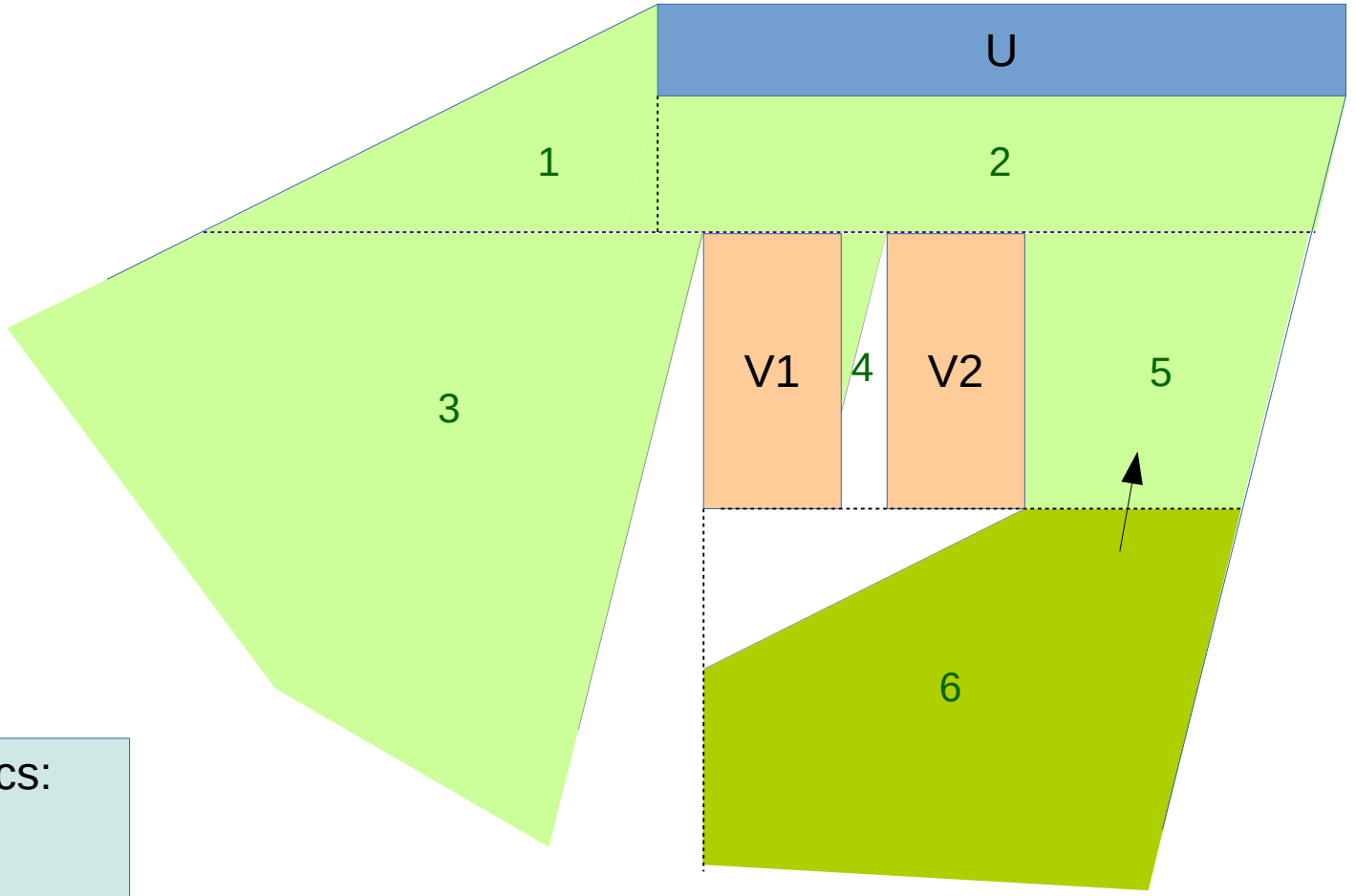


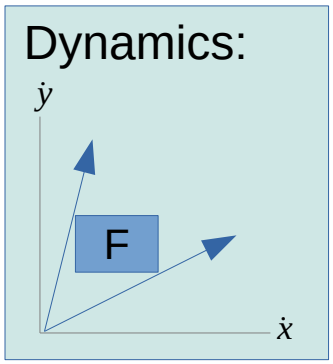
Computing RWA(U,V)



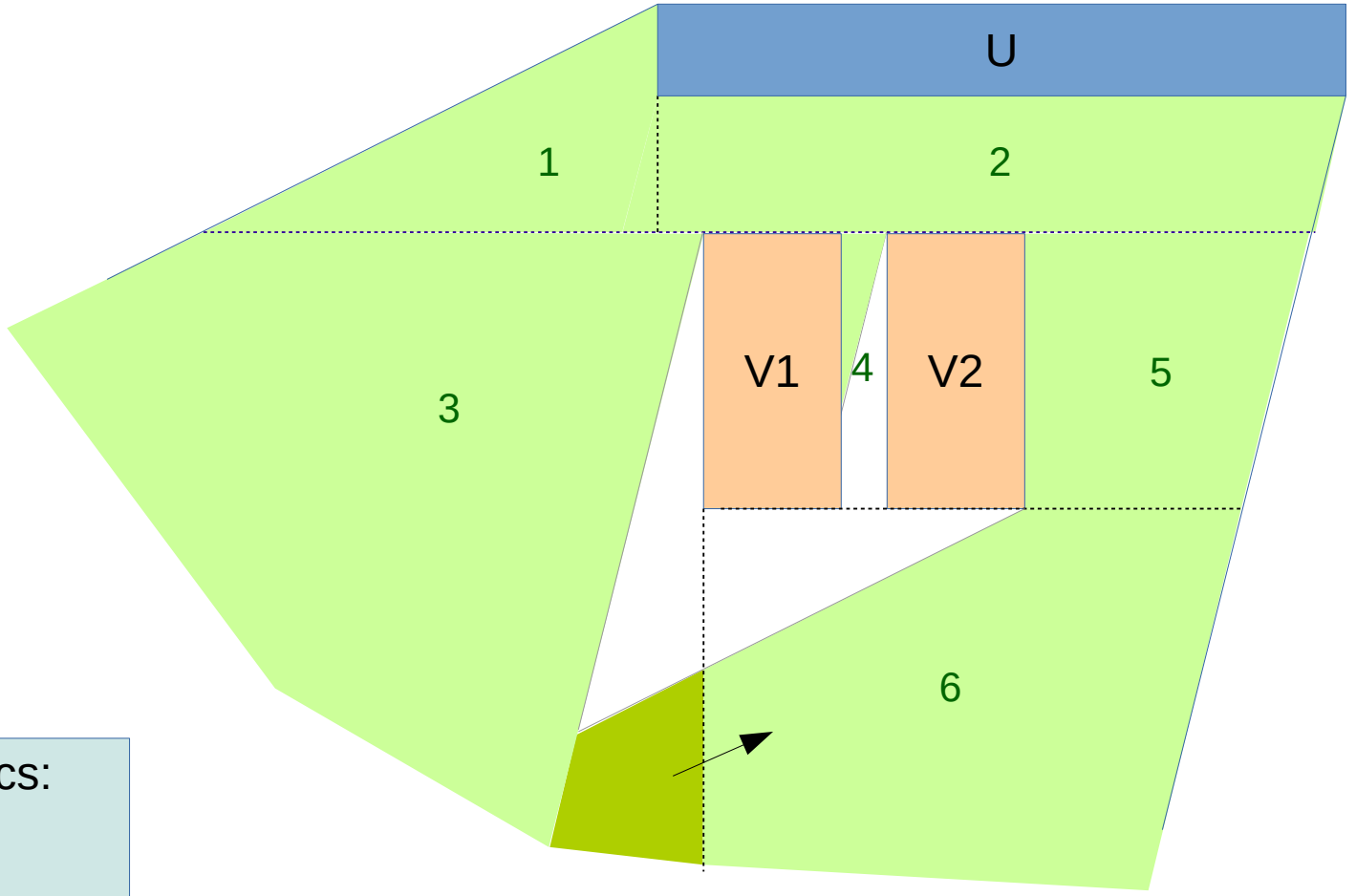


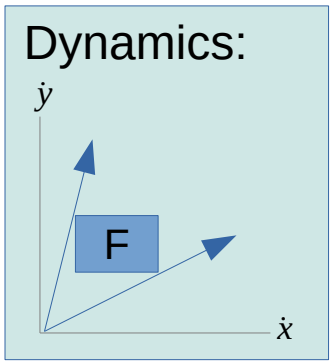
Computing RWA(U,V)



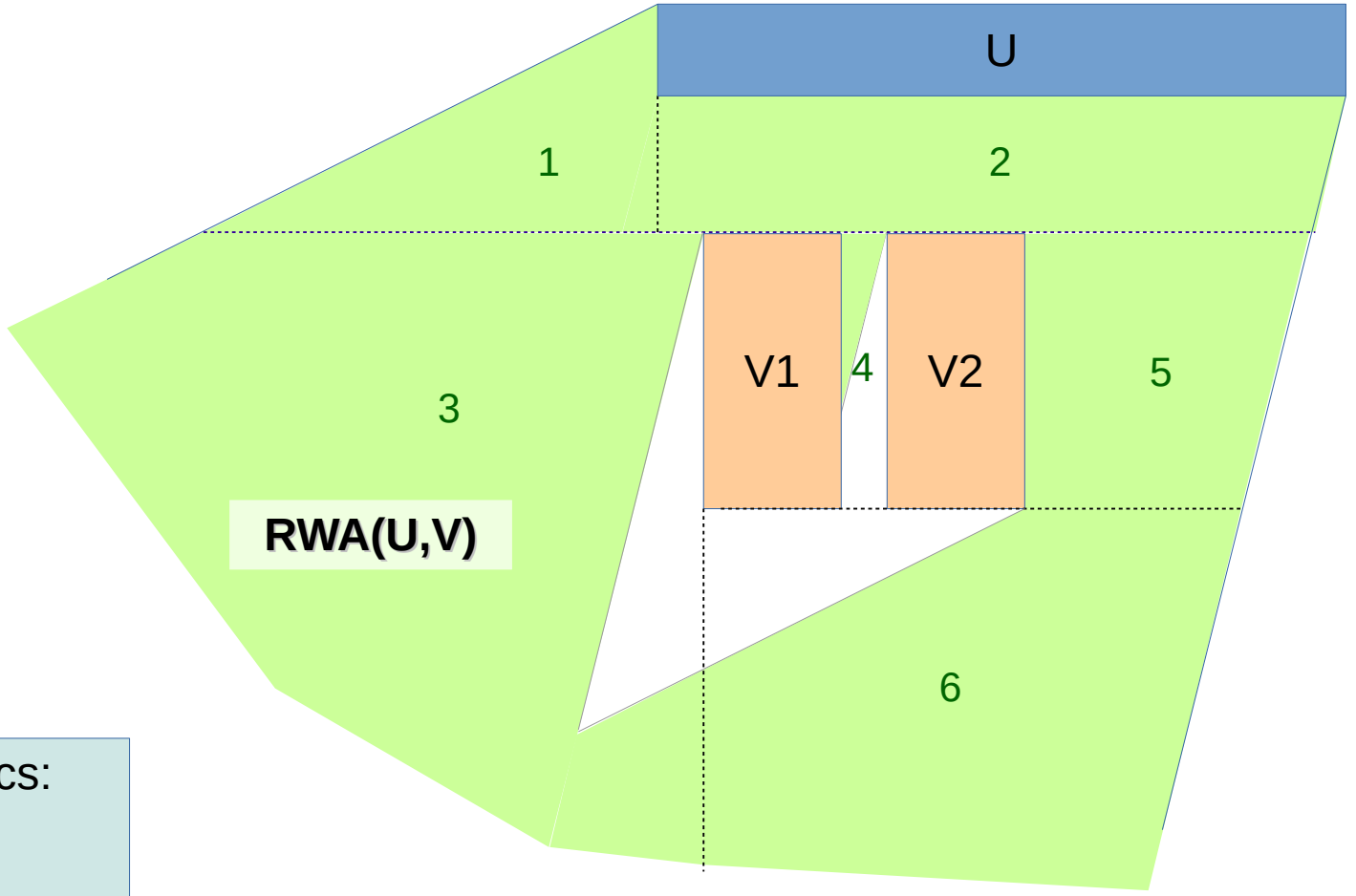


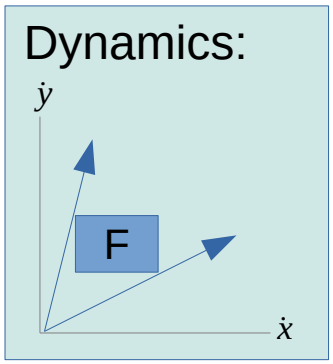
Computing RWA(U,V)





Computing RWA(U,V)





Algorithm for RWA(U, V)

$$\mu R\,.\,U \cup \bigcup_{P \in [\![\overline{V} \cap U_{\swarrow}]\!]} \bigcup_{P' \in [\![R]\!]} \Big(P \cap \big(bndry(P, P') \cap P'_{\swarrow} \big)_{\swarrow} \Big),$$

where

$$bndry(P, P') = (P \cap cl(P')) \cup (cl(P) \cap P').$$

and [[A]] is the representation of A as a finite set of convex polyhedra

- Interesting implementation issues
- More info in [Benerecetti et al., TCS 2013]