AVERIST
Algorithmic Verifier for Stability of Linear Hybrid Systems

Miriam García Soto and Pavithra Prabhakar

HSCC, April 2018

institute imdea software

Kansas State University

AVERIST

- Formal **stability verification** of hybrid systems
- **Classes considered:**
  - polyhedral hybrid systems (**PHS**)
  - linear hybrid systems (**LHS**)
- **Techniques implemented:**
  - Counterexample Guided Abstraction Refinement (**CEGAR**) for state-space reduction
  - Hybridization for dynamics simplification

Input & Stability property

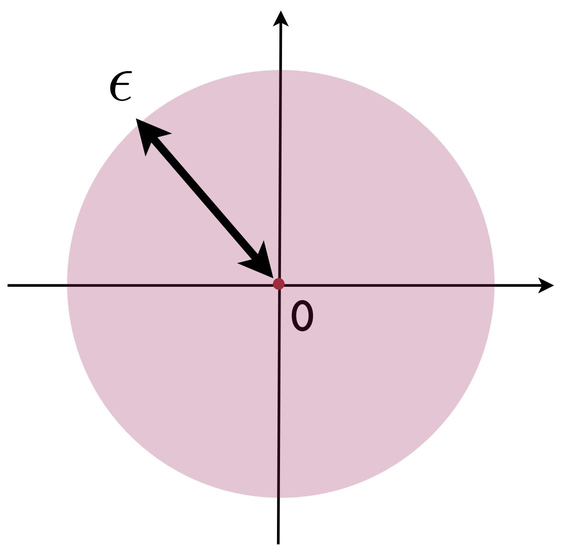
```
var: x,y;
location: quad1, quad2, quad3, quad4;

loc: quad1;
        inv: x>=0 AND y>=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad4;

loc: quad2;
        inv: x<=0 AND y>=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad1;

loc: quad3;
        inv: x<=0 AND y<=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad2;

loc: quad4;
        inv: x>=0 AND y<=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad3;
```

- A system is **Lyapunov stable** with respect to the **equilibrium point 0** if for every $\varepsilon > 0$ there exists $\delta > 0$ such that for every execution σ starting from $B_\delta(0)$, $\sigma(t) \in B_\varepsilon(0)$, for all time t.

Input & Stability property

```
var: x,y;
location: quad1, quad2, quad3, quad4;

loc: quad1;
        inv: x>=0 AND y>=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad4;

loc: quad2;
        inv: x<=0 AND y>=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad1;

loc: quad3;
        inv: x<=0 AND y<=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad2;

loc: quad4;
        inv: x>=0 AND y<=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad3;
```

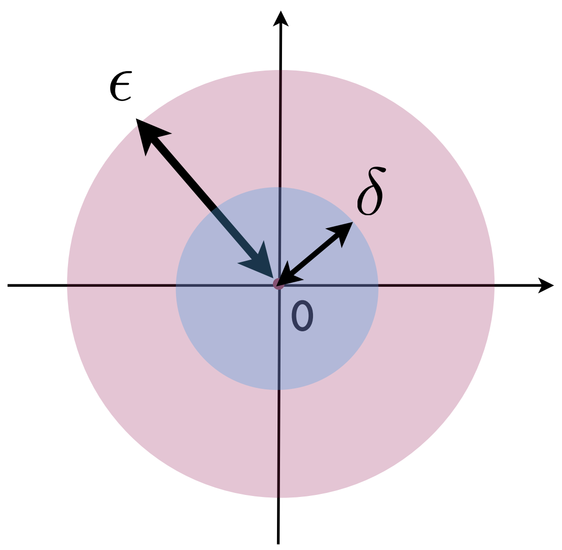
- A system is **Lyapunov stable** with respect to the **equilibrium point 0** if for every $\varepsilon > 0$ there exists $\delta > 0$ such that for every execution σ starting from $B_\delta(0)$, $\sigma(t) \in B_\varepsilon(0)$, for all time t.



Input & Stability property

```
var: x,y;
location: quad1, quad2, quad3, quad4;

loc: quad1;
        inv: x>=0 AND y>=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad4;

loc: quad2;
        inv: x<=0 AND y>=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad1;

loc: quad3;
        inv: x<=0 AND y<=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad2;

loc: quad4;
        inv: x>=0 AND y<=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad3;
```

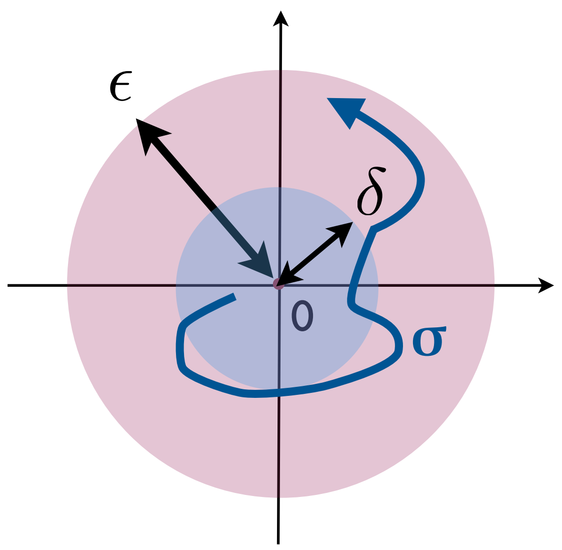
- A system is **Lyapunov stable** with respect to the **equilibrium point 0** if for every $\varepsilon > 0$ there exists $\delta > 0$ such that for every execution σ starting from $B_\delta(0)$, $\sigma(t) \in B_\varepsilon(0)$, for all time t.



Input & Stability property

```
var: x,y;
location: quad1, quad2, quad3, quad4;

loc: quad1;
        inv: x>=0 AND y>=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad4;

loc: quad2;
        inv: x<=0 AND y>=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad1;

loc: quad3;
        inv: x<=0 AND y<=0;
        dyn: dx==y AND dy==-4*x;
        guards:
                when y==0 goto quad2;

loc: quad4;
        inv: x>=0 AND y<=0;
        dyn: dx==10*y AND dy==-x;
        guards:
                when x==0 goto quad3;
```

- A system is **Lyapunov stable** with respect to the **equilibrium point 0** if for every $\varepsilon > 0$ there exists $\delta > 0$ such that for every execution σ starting from $B_\delta(0)$, $\sigma(t) \in B_\varepsilon(0)$, for all time t.



Stability verification

State-of-the-art: Lyapunov’s second method

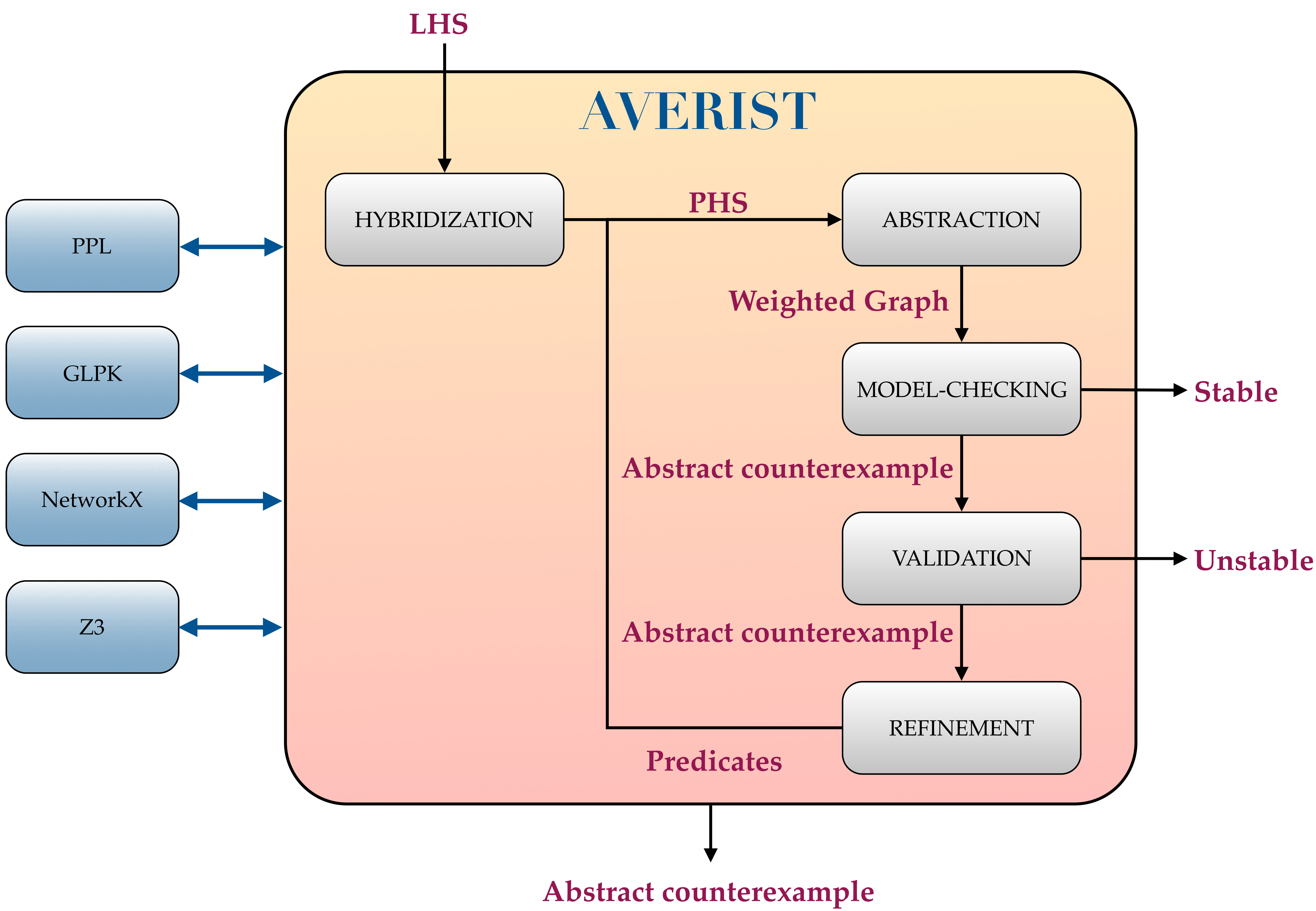
Template based search

- Choose a **template**
- Encode **Lyapunov** function **conditions** as constraints
- Solve using **sum-of-squares** programming tools

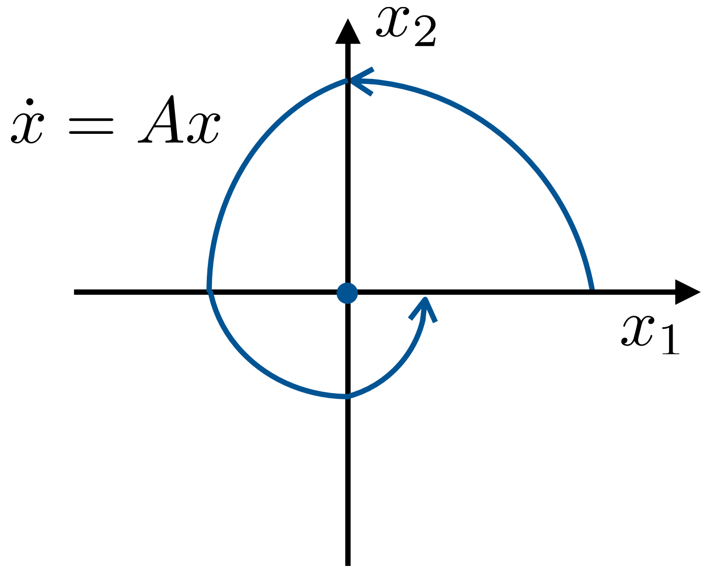
CEGAR approach

- Constructs an **abstract weighted graph** from the hybrid system and a state space partition
- **Systematically iterates** over the abstract systems
- **Returns** a **counterexample** in the case that the **abstraction fails**
- The **counterexample** can be used to **guide** the choice of the **next abstraction**

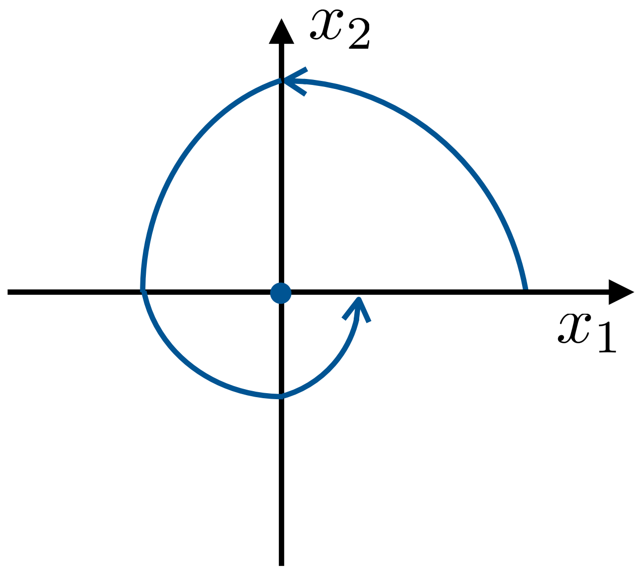
AVERIST diagram



Hybridization

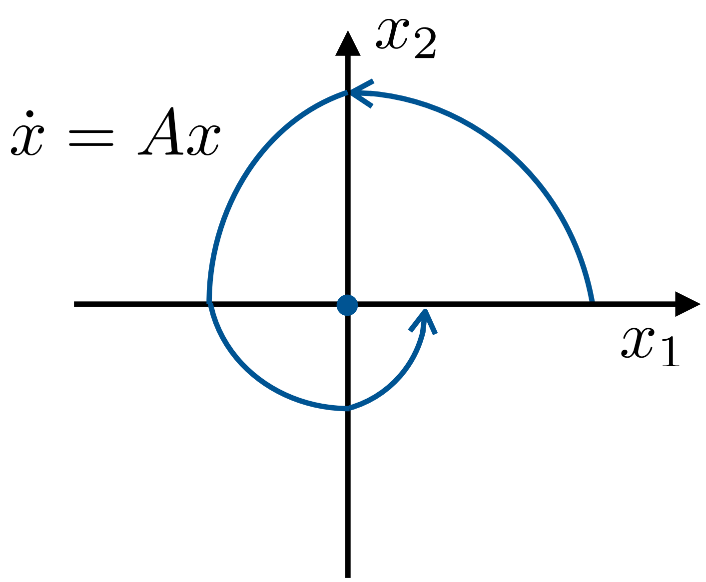


Linear hybrid system

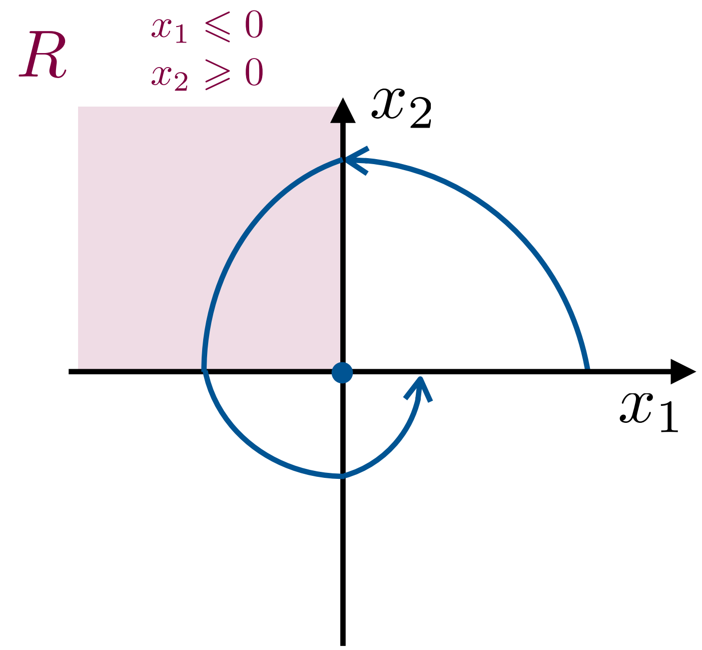


Polyhedral hybrid system

Hybridization

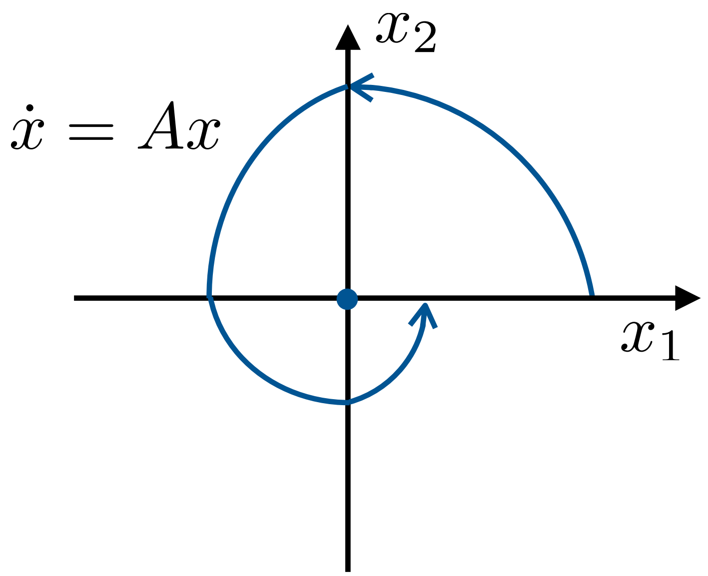


Linear hybrid system

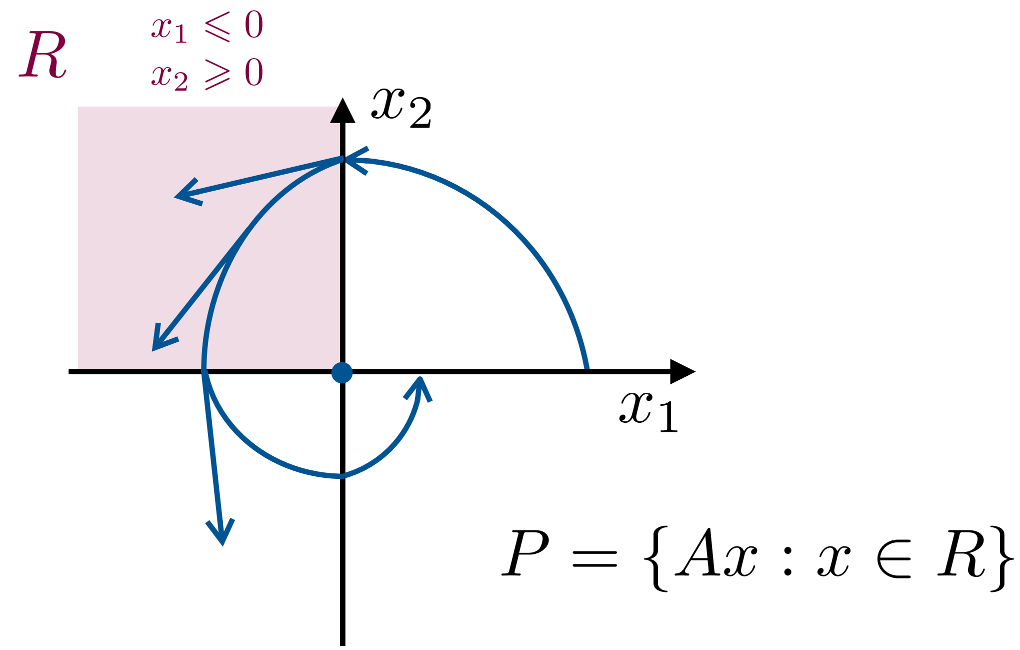


Polyhedral hybrid system

Hybridization

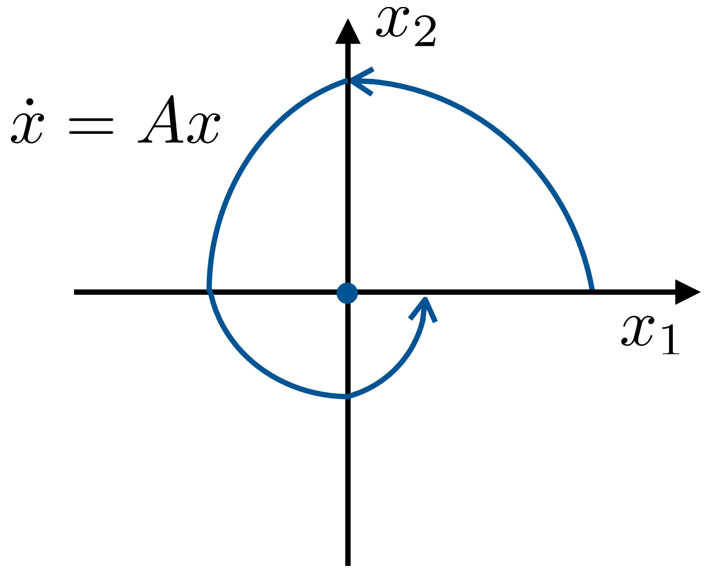


Linear hybrid system

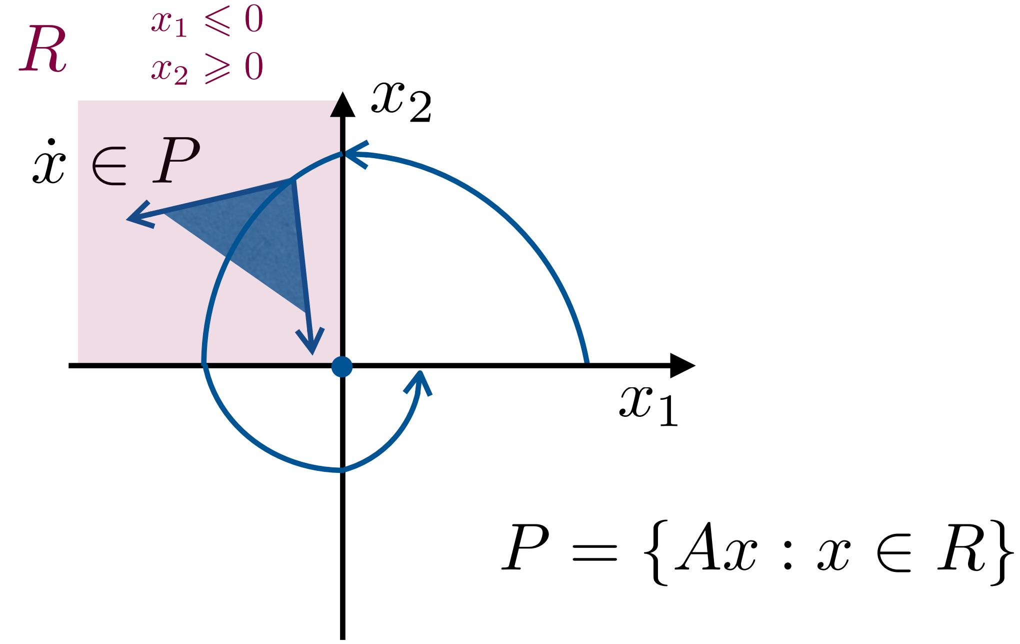


Polyhedral hybrid system

Hybridization



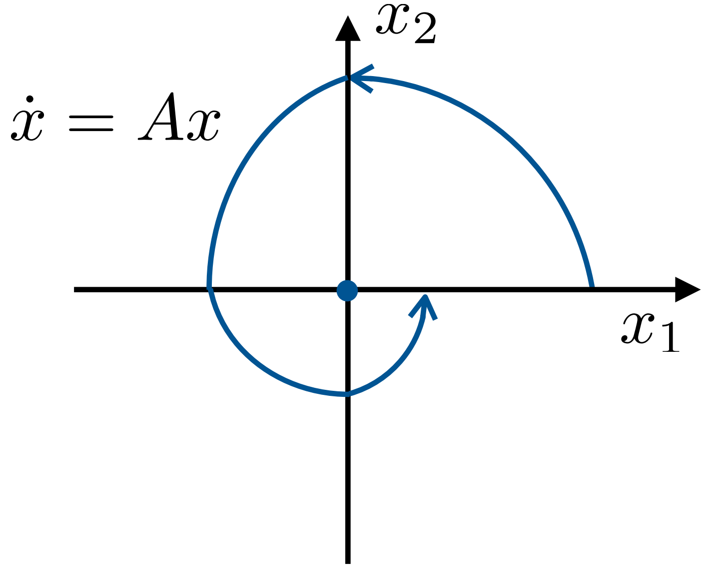
Linear hybrid system



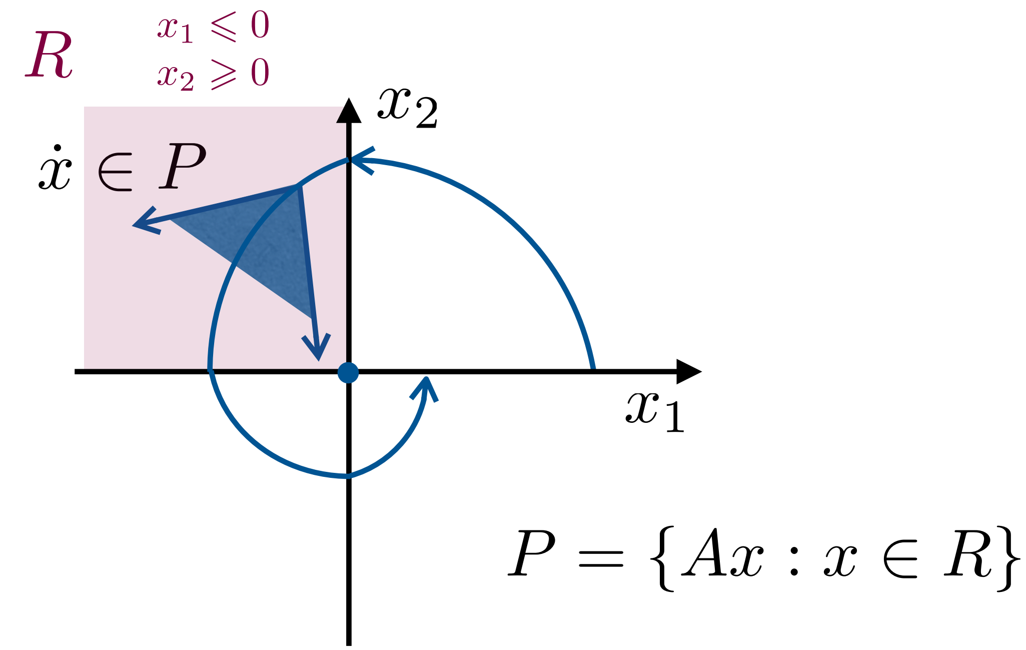
Polyhedral hybrid system

P is defined as a ***convex polyhedron*** *using* ***PPL****.*

Hybridization



Linear hybrid system



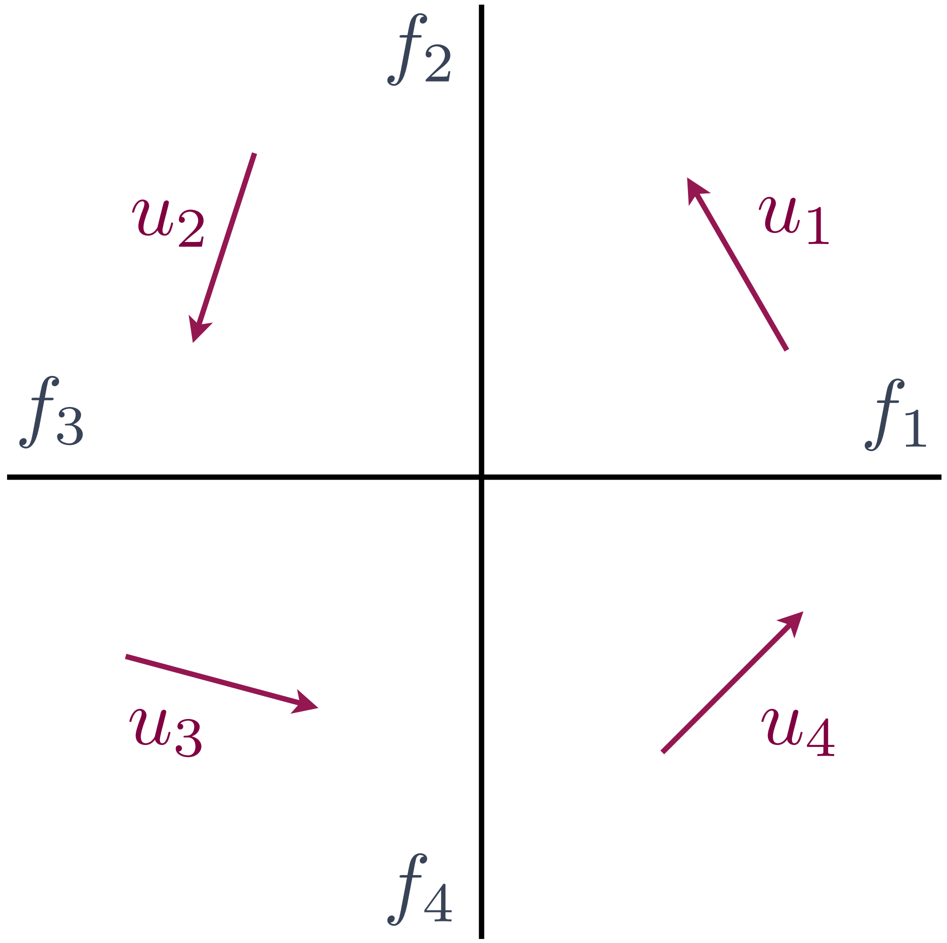
Polyhedral hybrid system

P is defined as a ***convex polyhedron*** *using* ***PPL****.*

Theorem - Hybridization

If the hybridized polyhedral hybrid system is Lyapunov stable then the original linear hybrid system is Lyapunov stable.

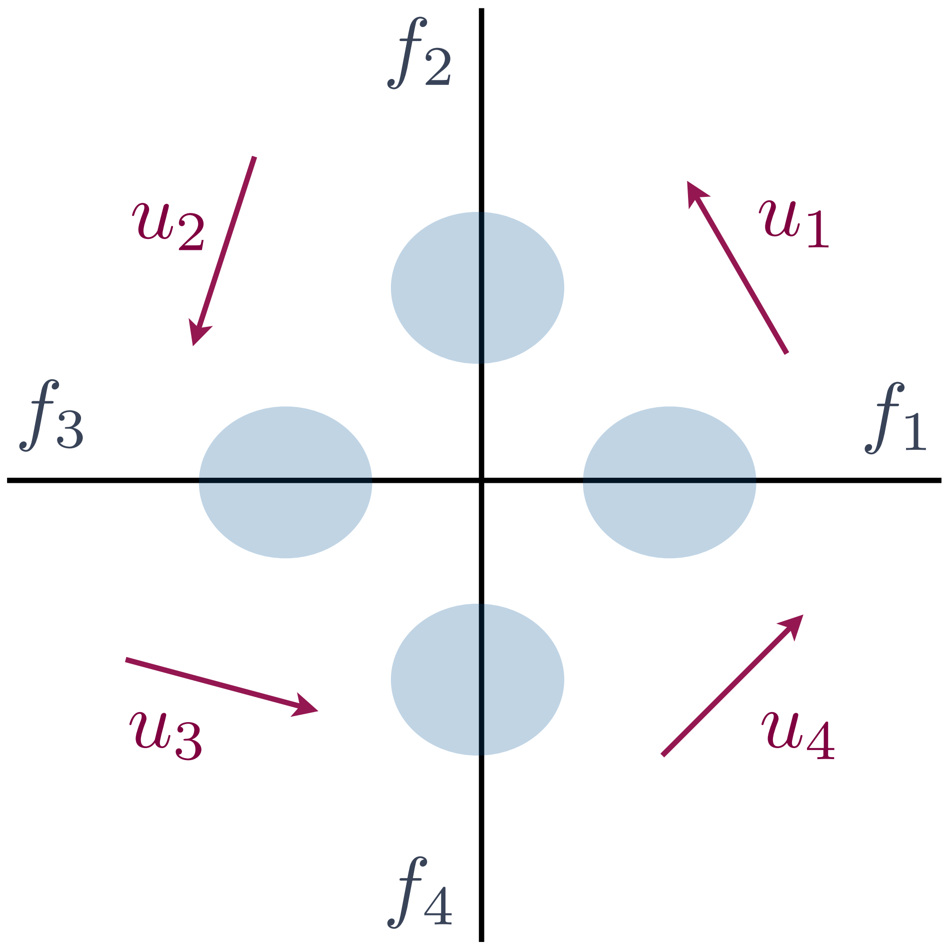
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

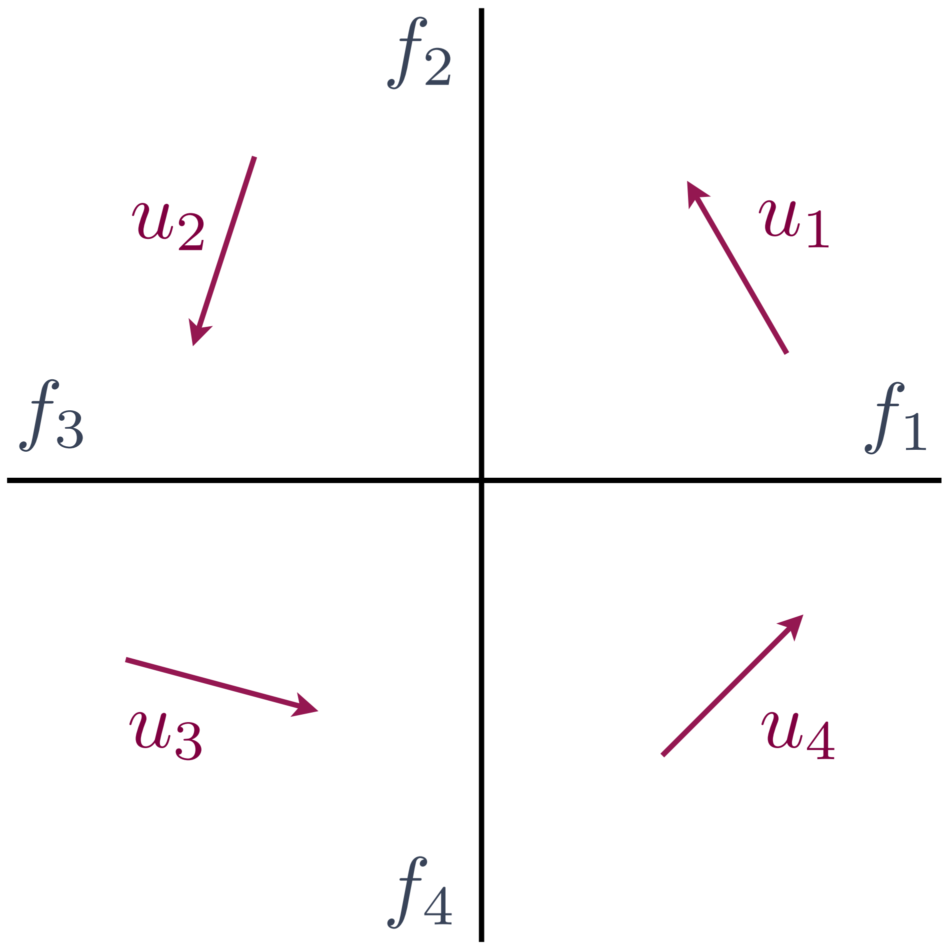
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

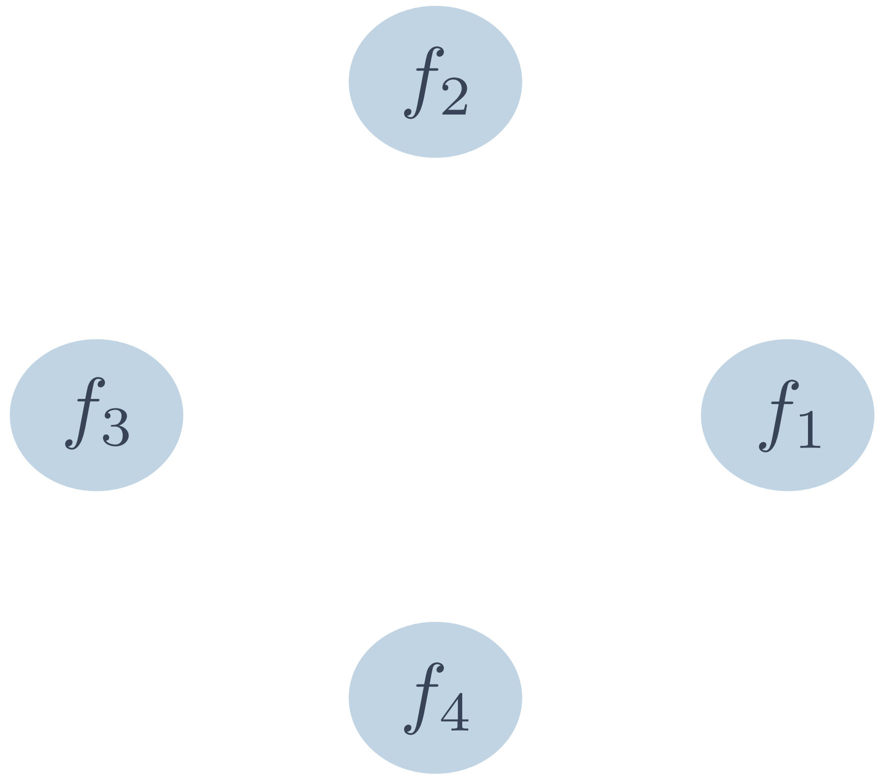
Quantitative Predicate Abstraction



Concrete system

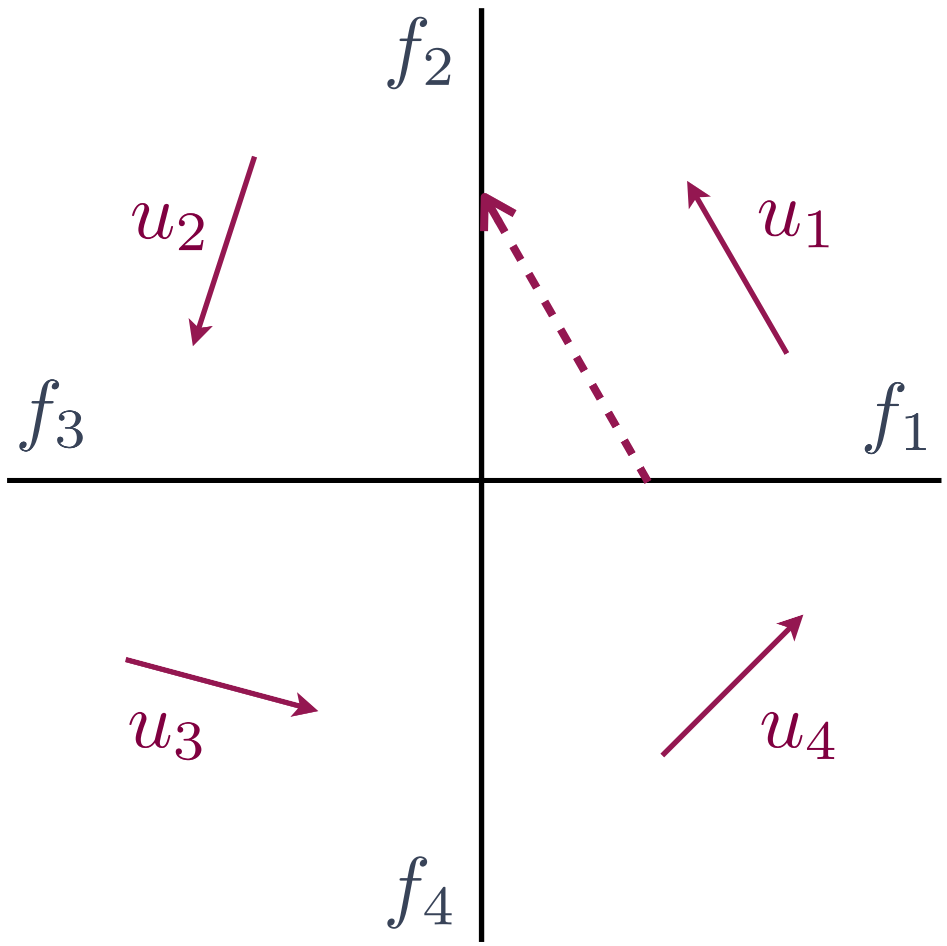
Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

$\Longrightarrow$

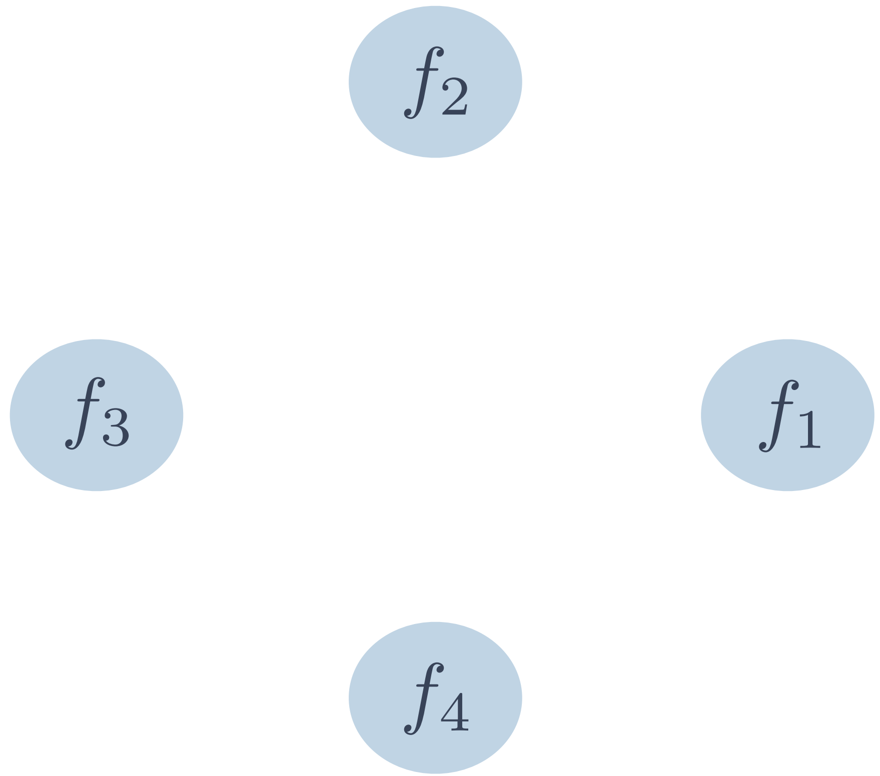


Abstract system

Quantitative Predicate Abstraction



$\Longrightarrow$

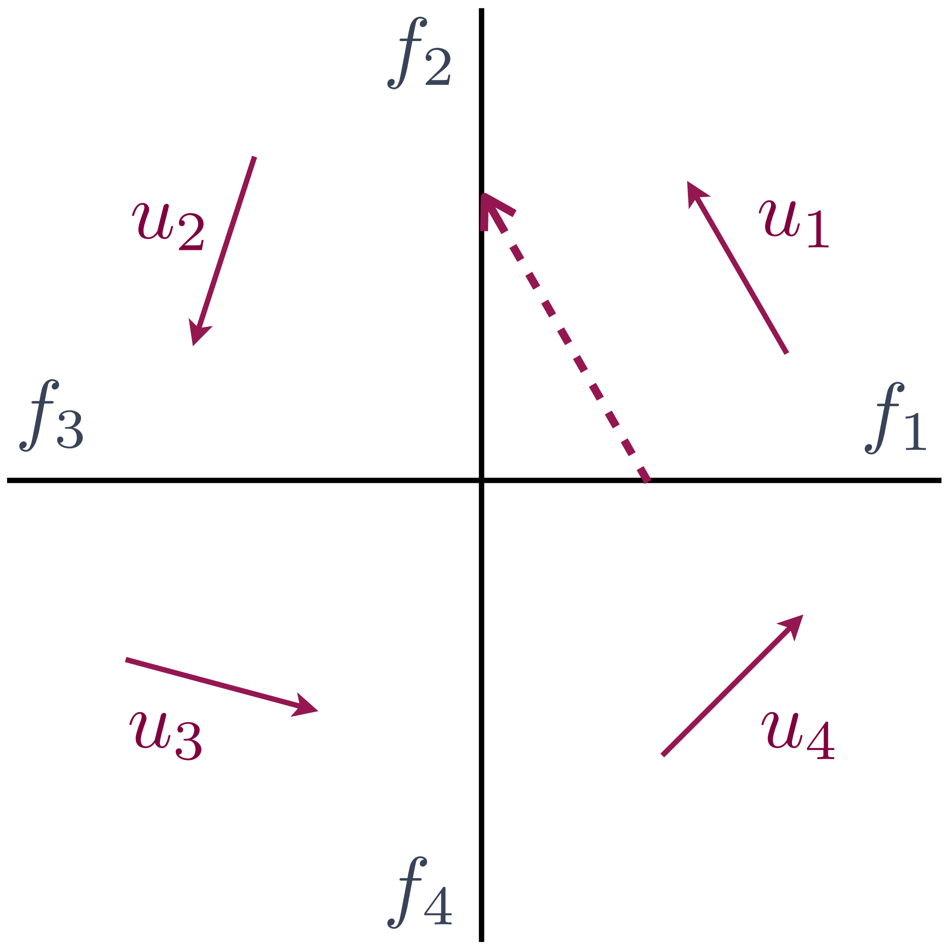


Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

Abstract system

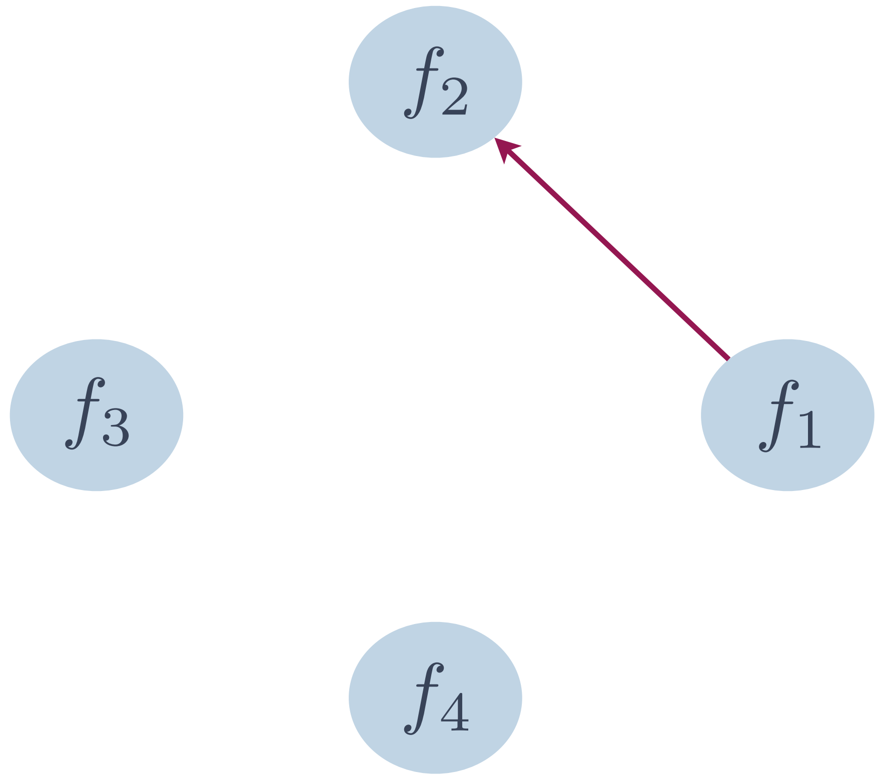
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

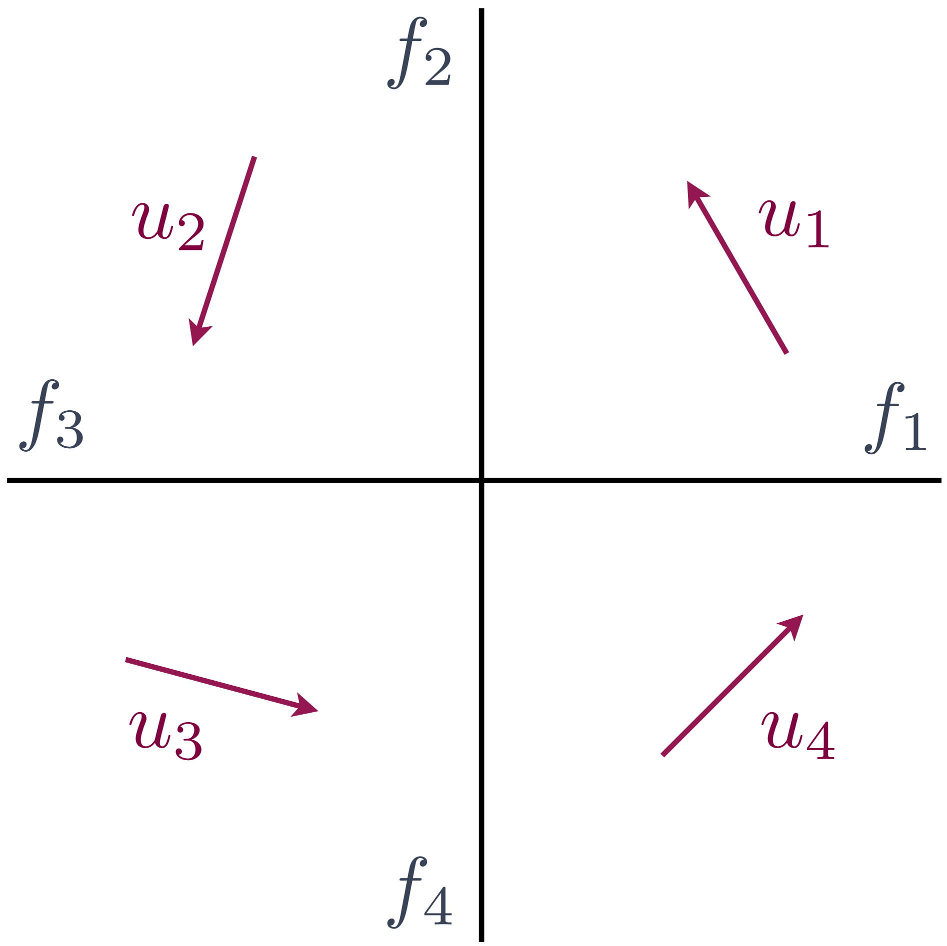
$\Longrightarrow$



Abstract system

An edge between facets indicates the existence of an execution.

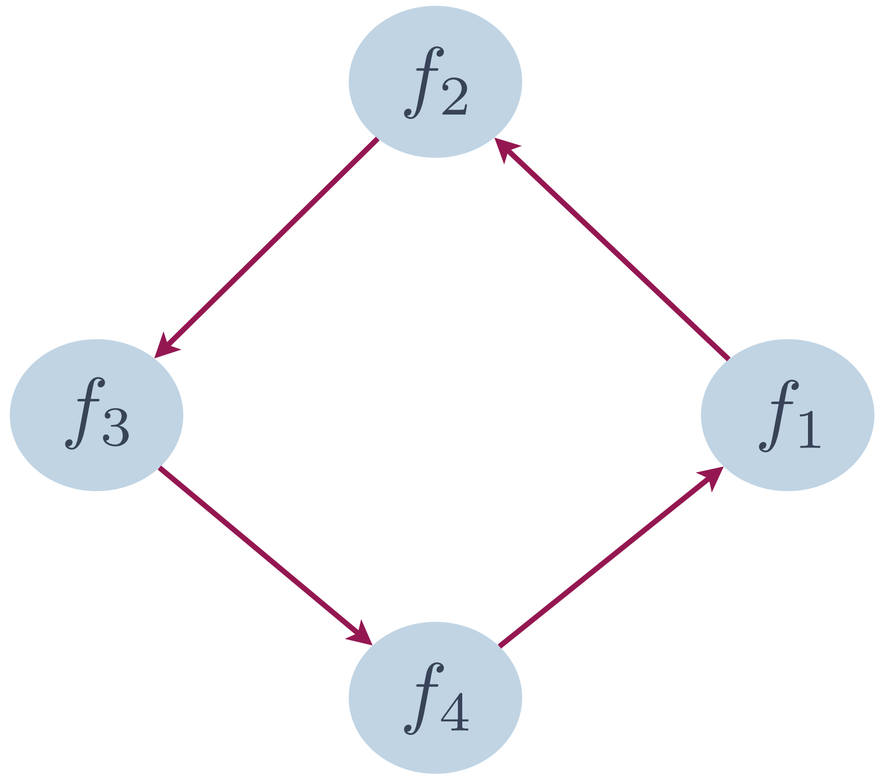
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

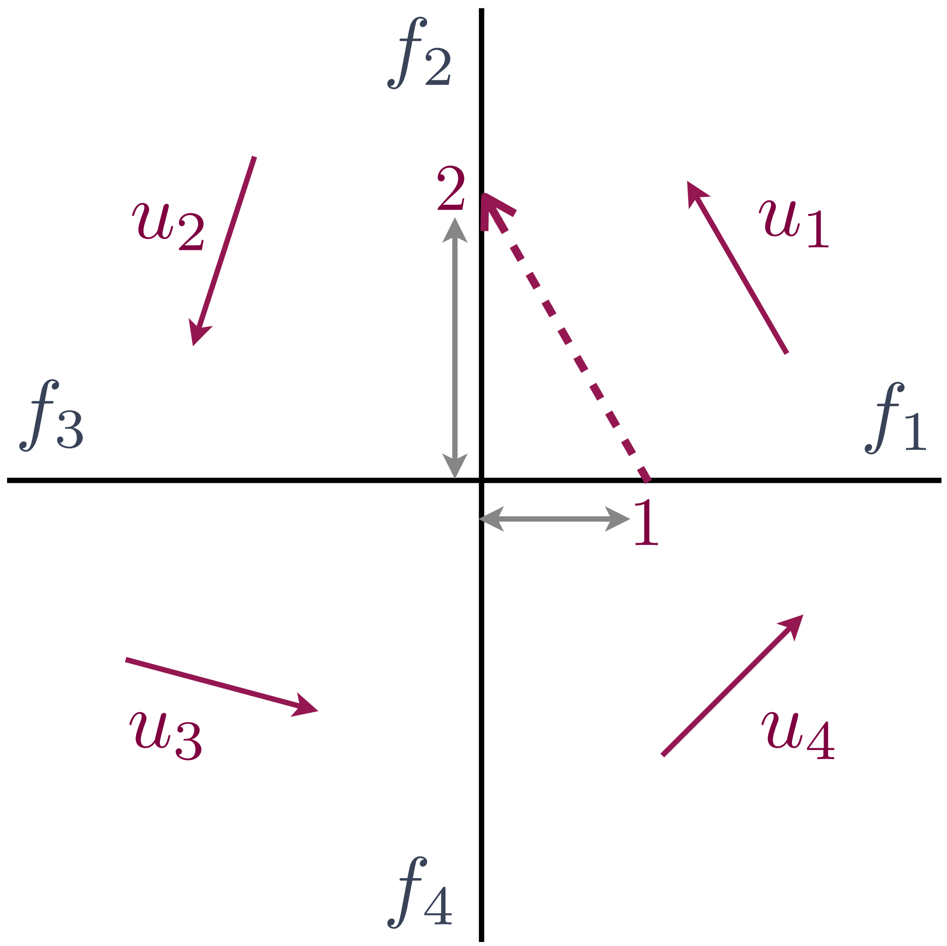
$\Longrightarrow$



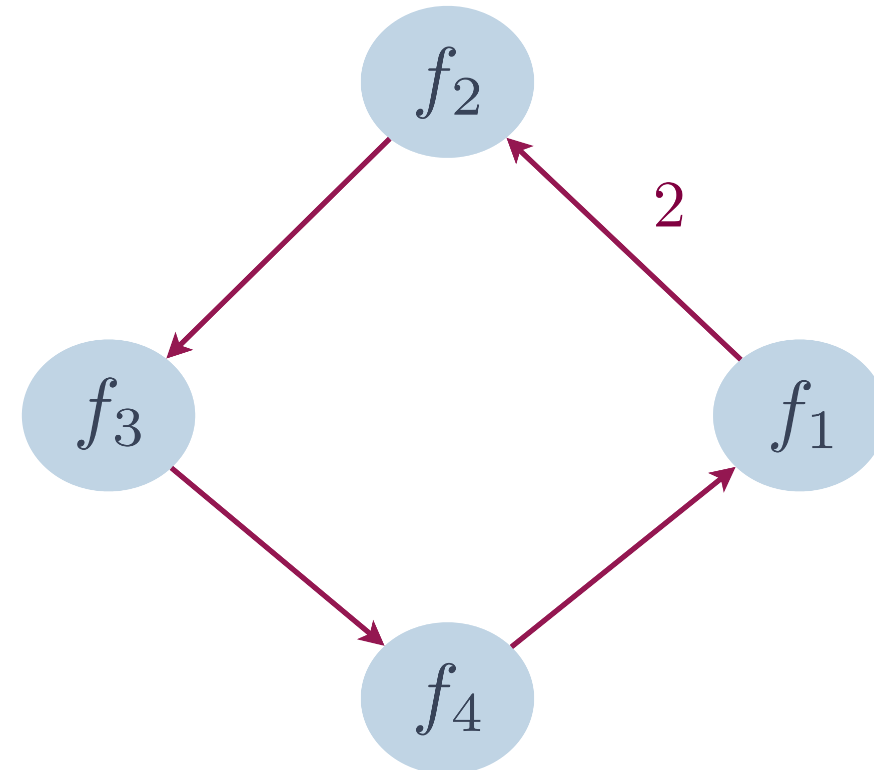
Abstract system

An edge between facets indicates the existence of an execution.

Quantitative Predicate Abstraction



$\Longrightarrow$



Concrete system

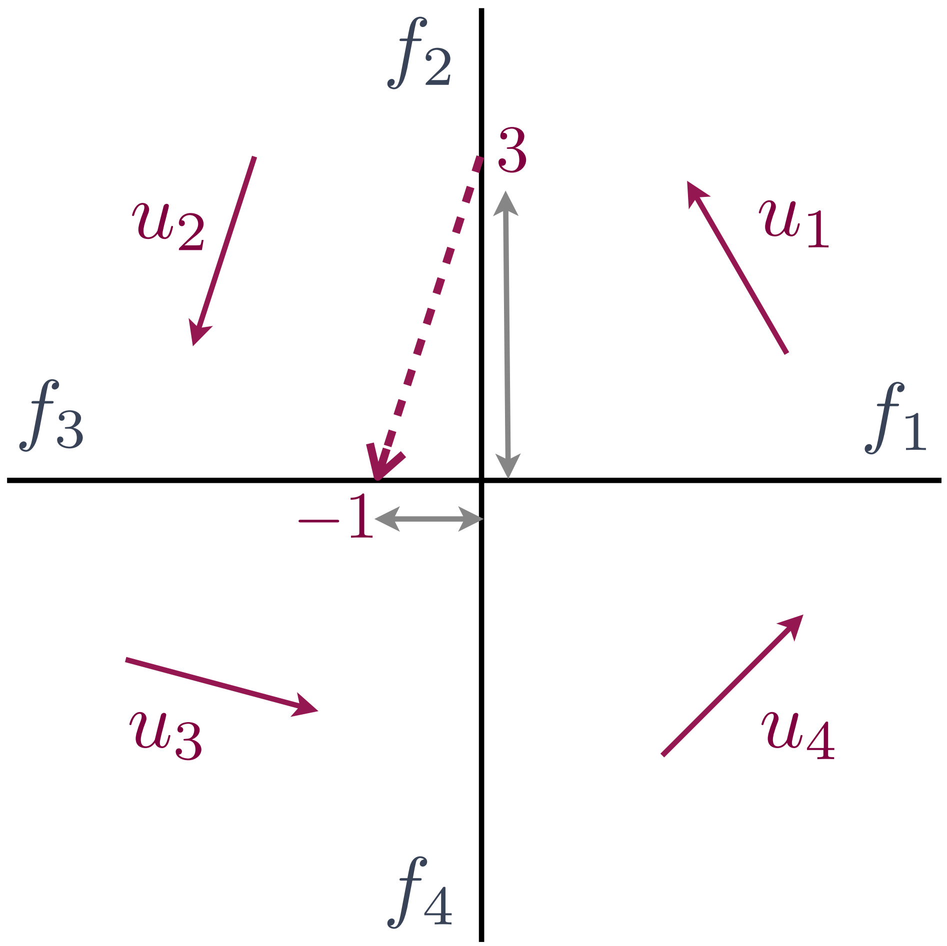
Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

Abstract system

An edge between facets indicates the existence of an execution.

Weights capture information about distance to the equilibrium point along the executions.

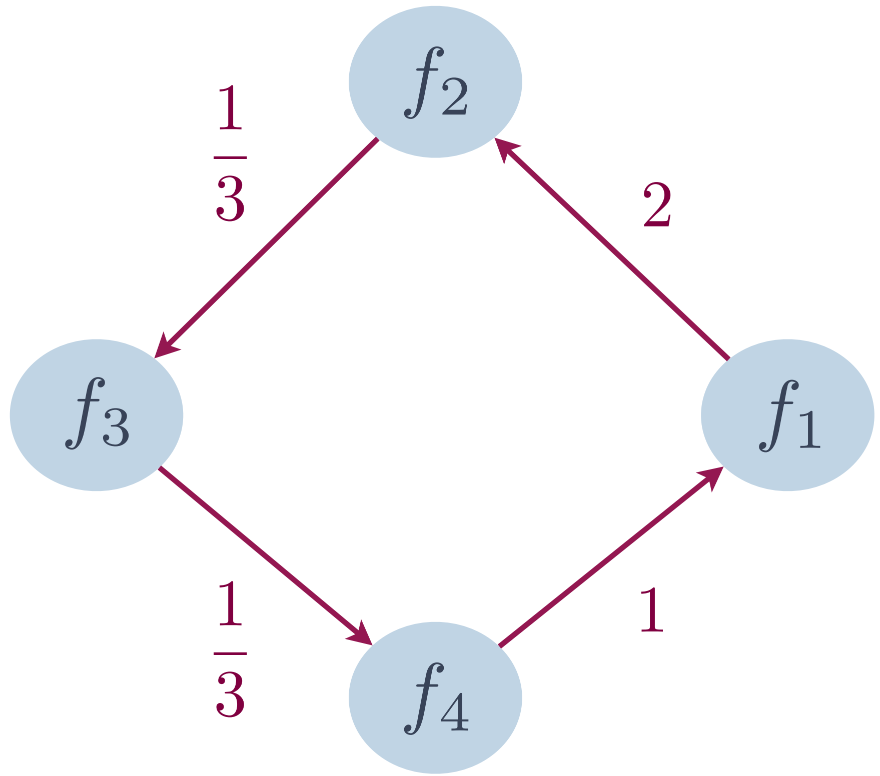
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

$\Longrightarrow$

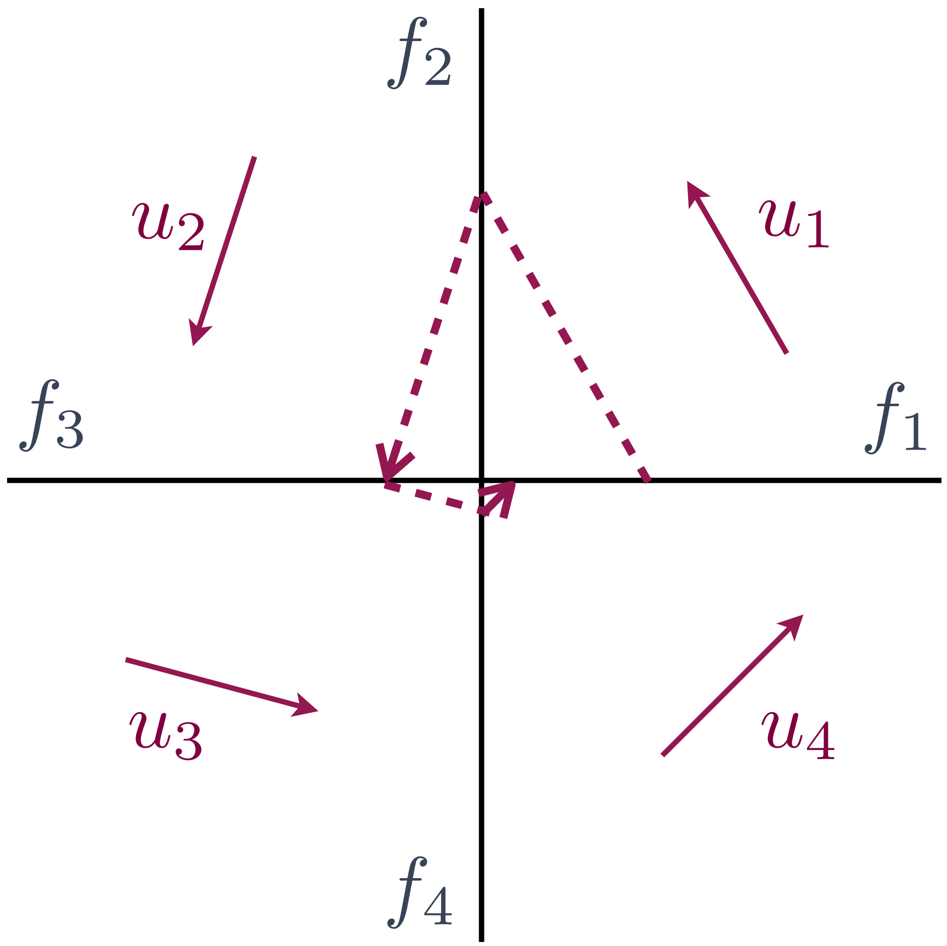


Abstract system

An edge between facets indicates the existence of an execution.

Weights capture information about distance to the equilibrium point along the executions.

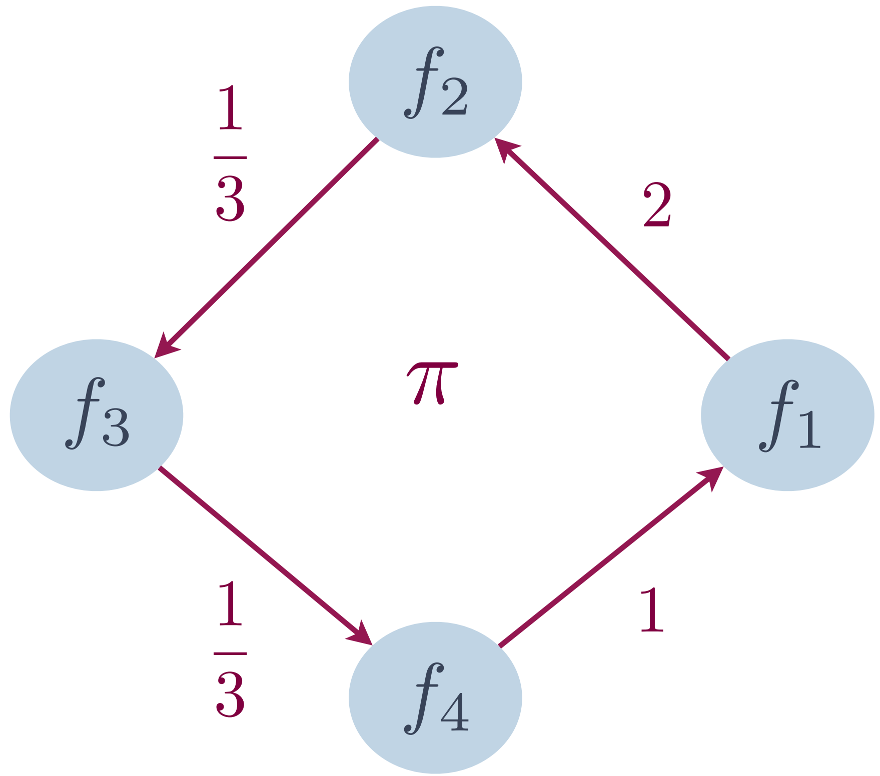
Quantitative Predicate Abstraction



Concrete system

Facets $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$

$\Longrightarrow$



Abstract system

$$W(\pi) = 2 \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot 1 = \frac{2}{9} < 1$$

An edge between facets indicates the existence of an execution.

Weights capture information about distance to the equilibrium point along the executions.

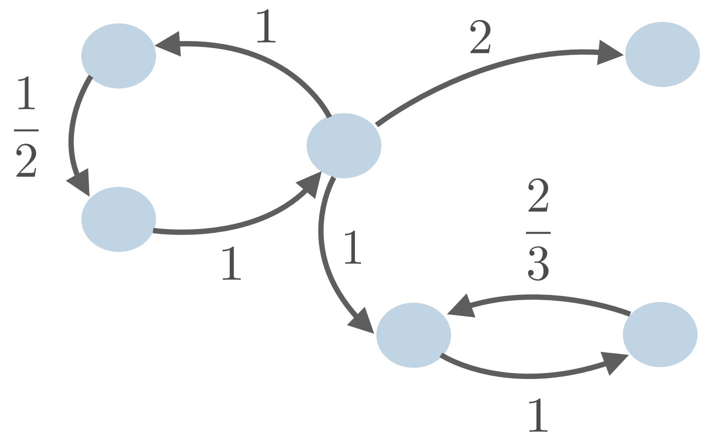
Model-checking

Theorem - Model-checking

A polyhedral hybrid system is Lyapunov stable if

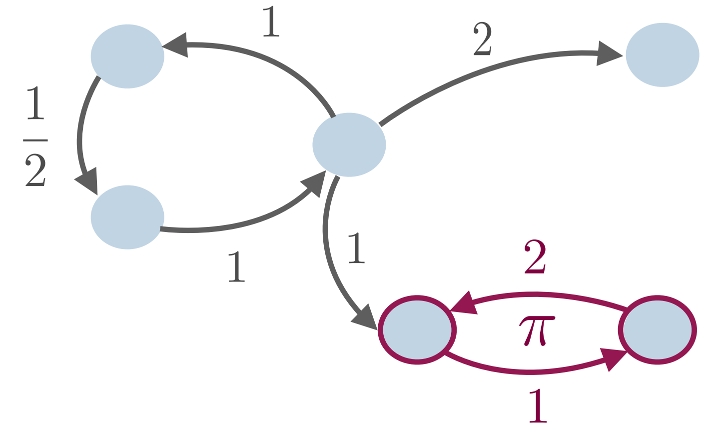
- the abstract weighted graph has no edges with infinite weights, and
- no cycles with product of edge weights greater than 1

Abstract system



Every cycle has weight smaller than 1 => Hybrid system is stable => Stop

Abstract system



There is a cycle, π, with weight greater than 1 => π is an abstract counterexample => Validation

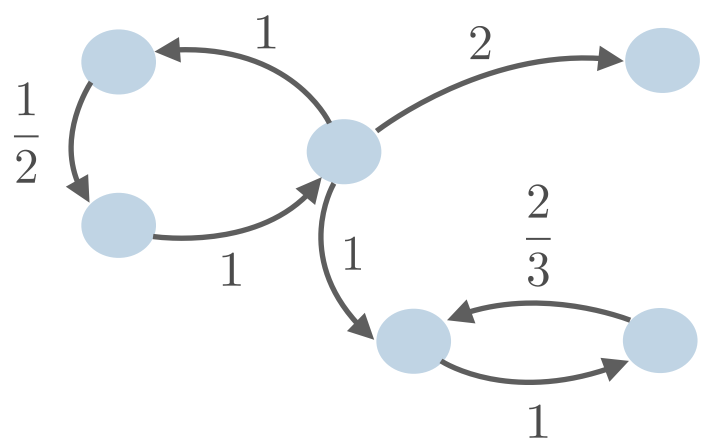
Model-checking

Theorem - Model-checking

A polyhedral hybrid system is Lyapunov stable if

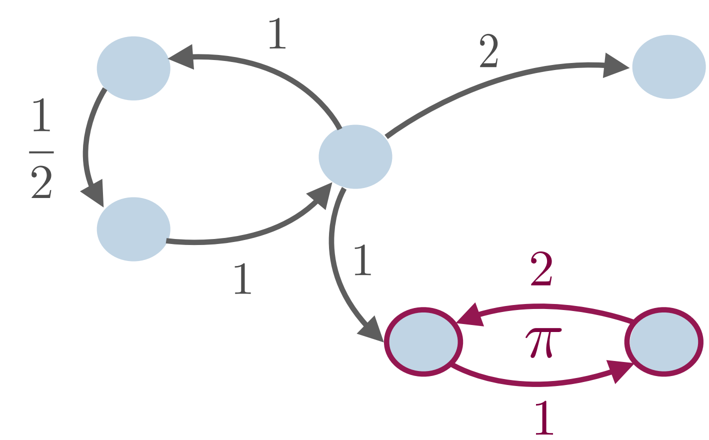
- the abstract weighted graph has no edges with infinite weights, and
- no cycles with product of edge weights greater than 1

Abstract system



Every cycle has weight smaller than 1 => Hybrid system is stable => Stop

Abstract system



There is a cycle, π, with weight greater than 1 => π is an abstract counterexample => Validation

*Adaptation of **Bellman-Ford algorithm** included in **NetworkX** package.*

Validation

- **Abstract counterexample** $\pi = f_1 \longrightarrow f_2 \longrightarrow f_3 \longrightarrow \ldots \longrightarrow f_1$
- **Validation** checks if π is valid, that is, corresponds to an infinite execution in the hybrid system which follows the edges and weights of π and diverges

Theorem - Validation

A counterexample $f_1 \longrightarrow f_2 \longrightarrow f_3 \longrightarrow \ldots \longrightarrow f_1$ is valid

$$\Longleftrightarrow$$

$$\exists\, \alpha > 1, \exists\, x_1 \in f_1, \ldots, x_k \in f_k, x_{k+1} \in f_1$$

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \ldots \longrightarrow x_k \longrightarrow x_{k+1}, \; x_{k+1} = \alpha x_1$$

Validation

- **Abstract counterexample** $\pi = f_1 \longrightarrow f_2 \longrightarrow f_3 \longrightarrow \ldots \longrightarrow f_1$
- **Validation** checks if π is valid, that is, corresponds to an infinite execution in the hybrid system which follows the edges and weights of π and diverges

Theorem - Validation

A counterexample $f_1 \longrightarrow f_2 \longrightarrow f_3 \longrightarrow \ldots \longrightarrow f_1$ is valid

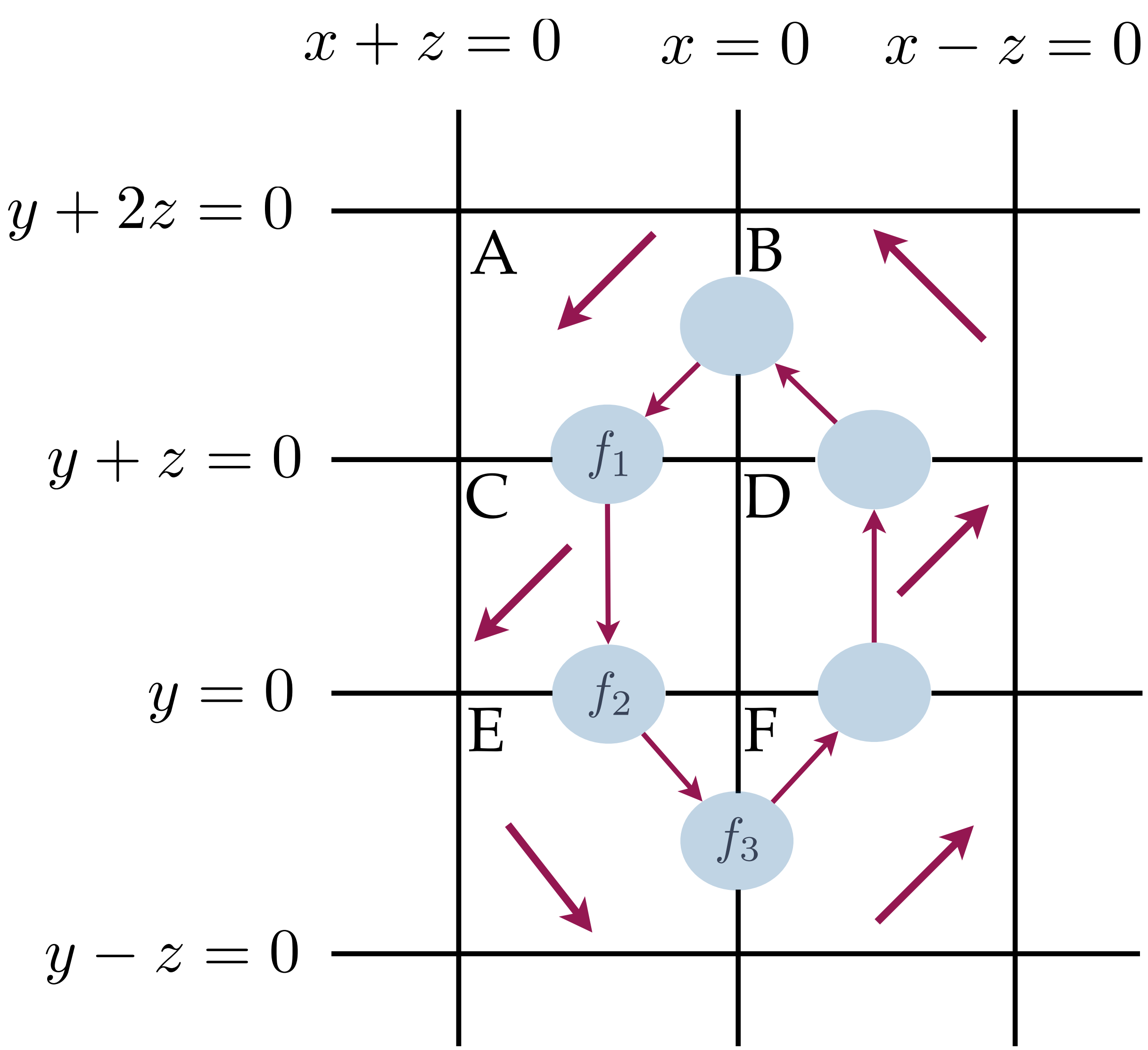
$$\Longleftrightarrow$$

$$\exists\, \alpha > 1,\ \exists\, x_1 \in f_1, \ldots, x_k \in f_k, x_{k+1} \in f_1$$

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow \ldots \longrightarrow x_k \longrightarrow x_{k+1},\ x_{k+1} = \alpha x_1$$

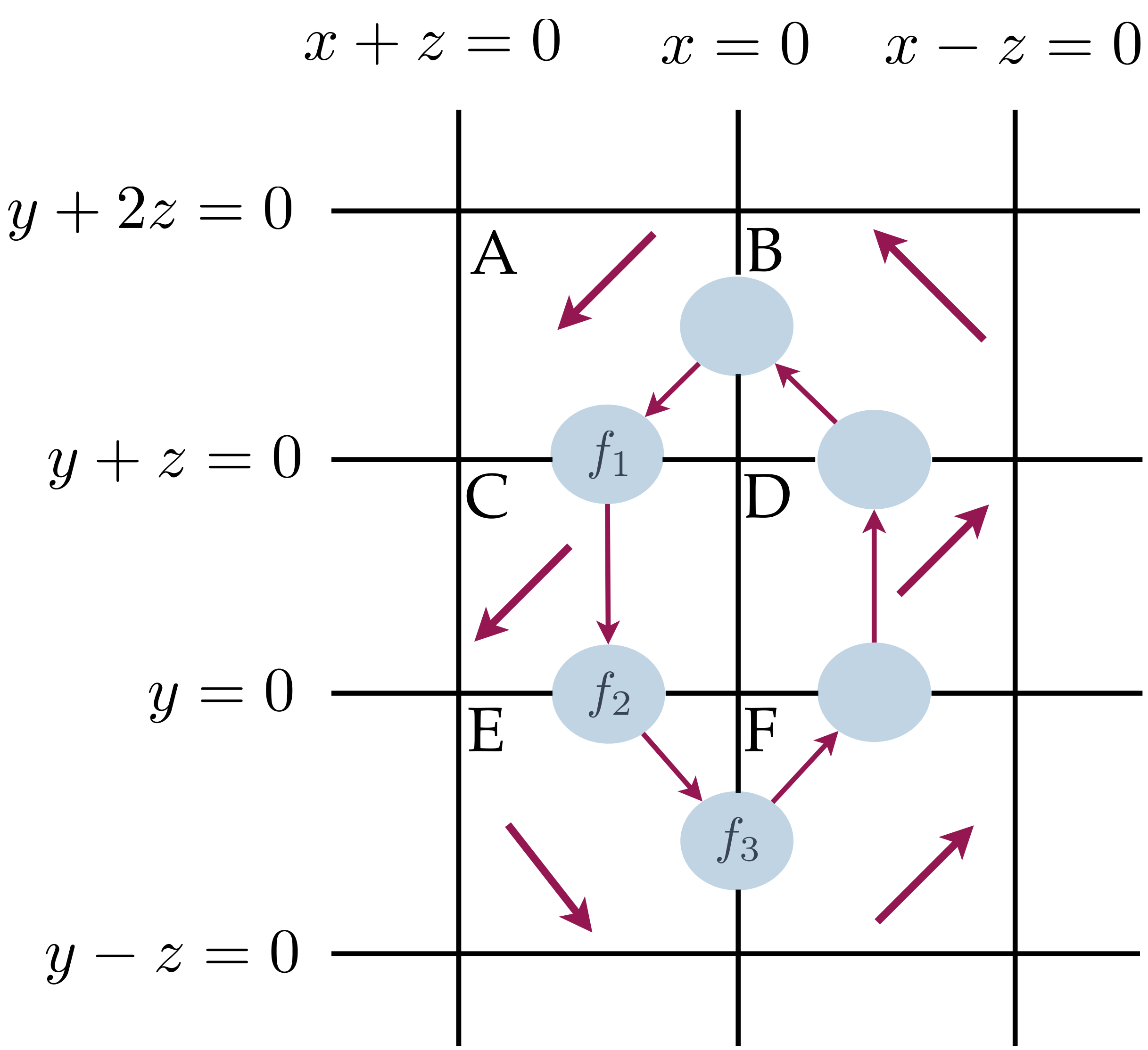
*Encoded as an **SMT formula** and solved with **Z3**.*

Refinement

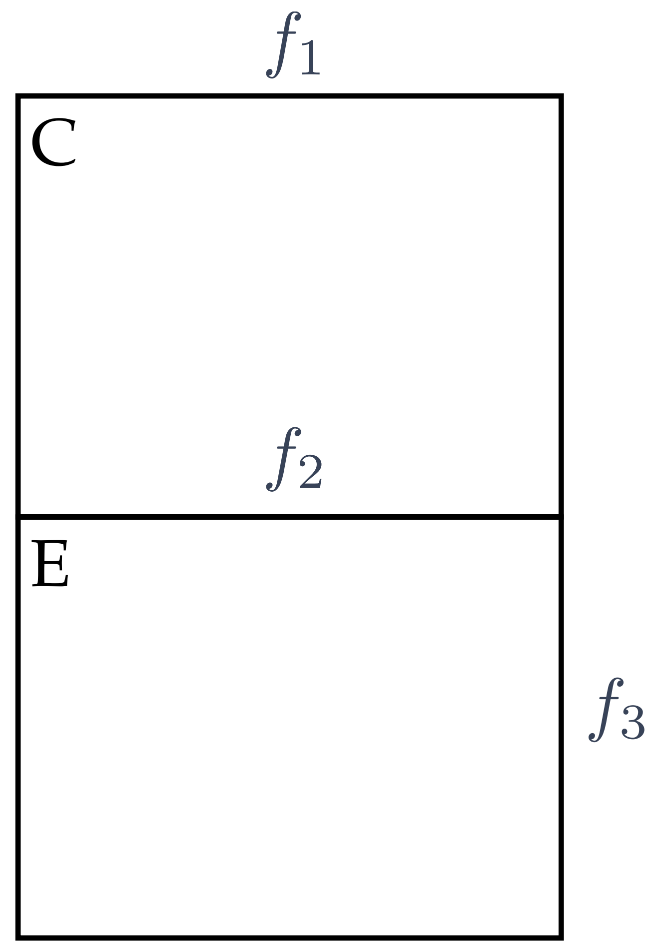


Spurious counterexample

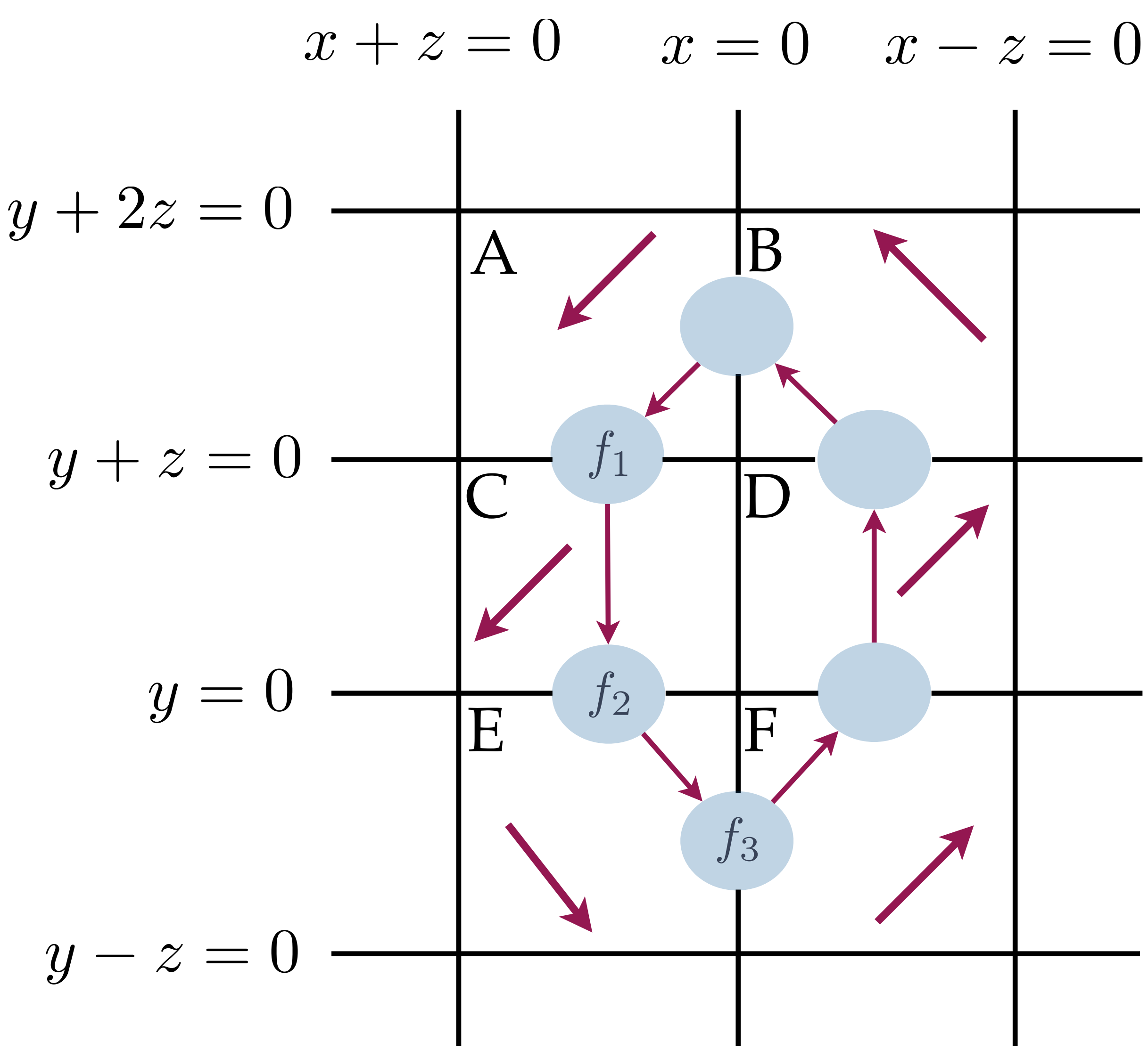
Refinement



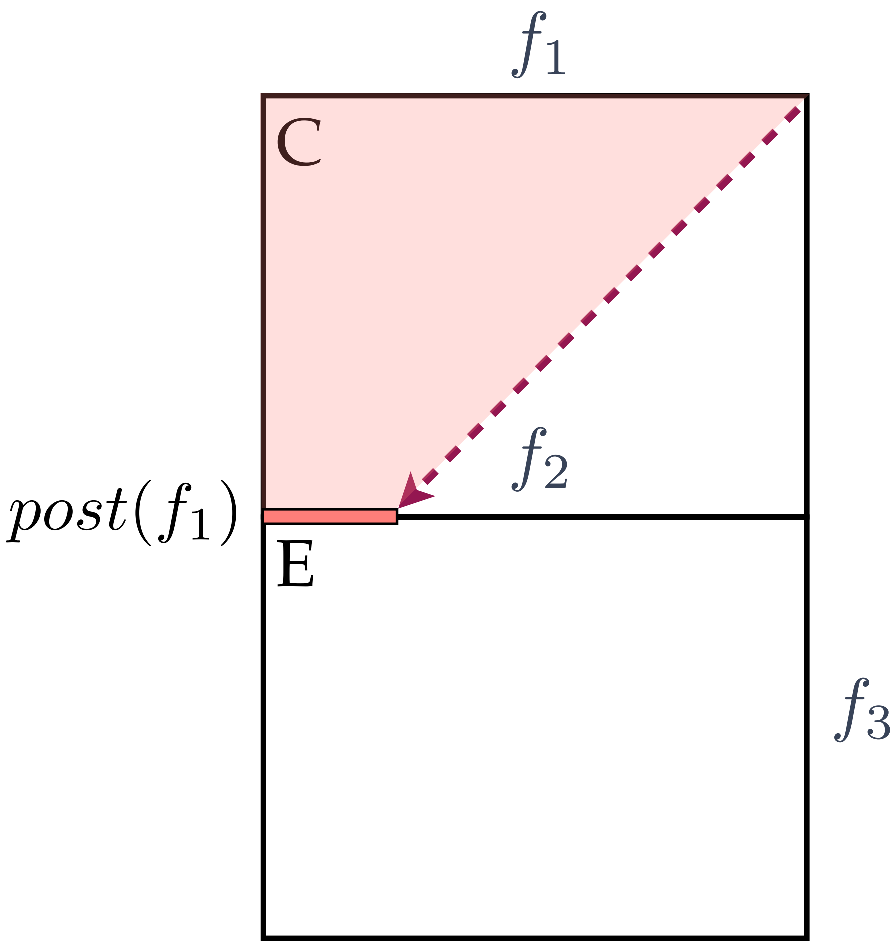
Spurious counterexample



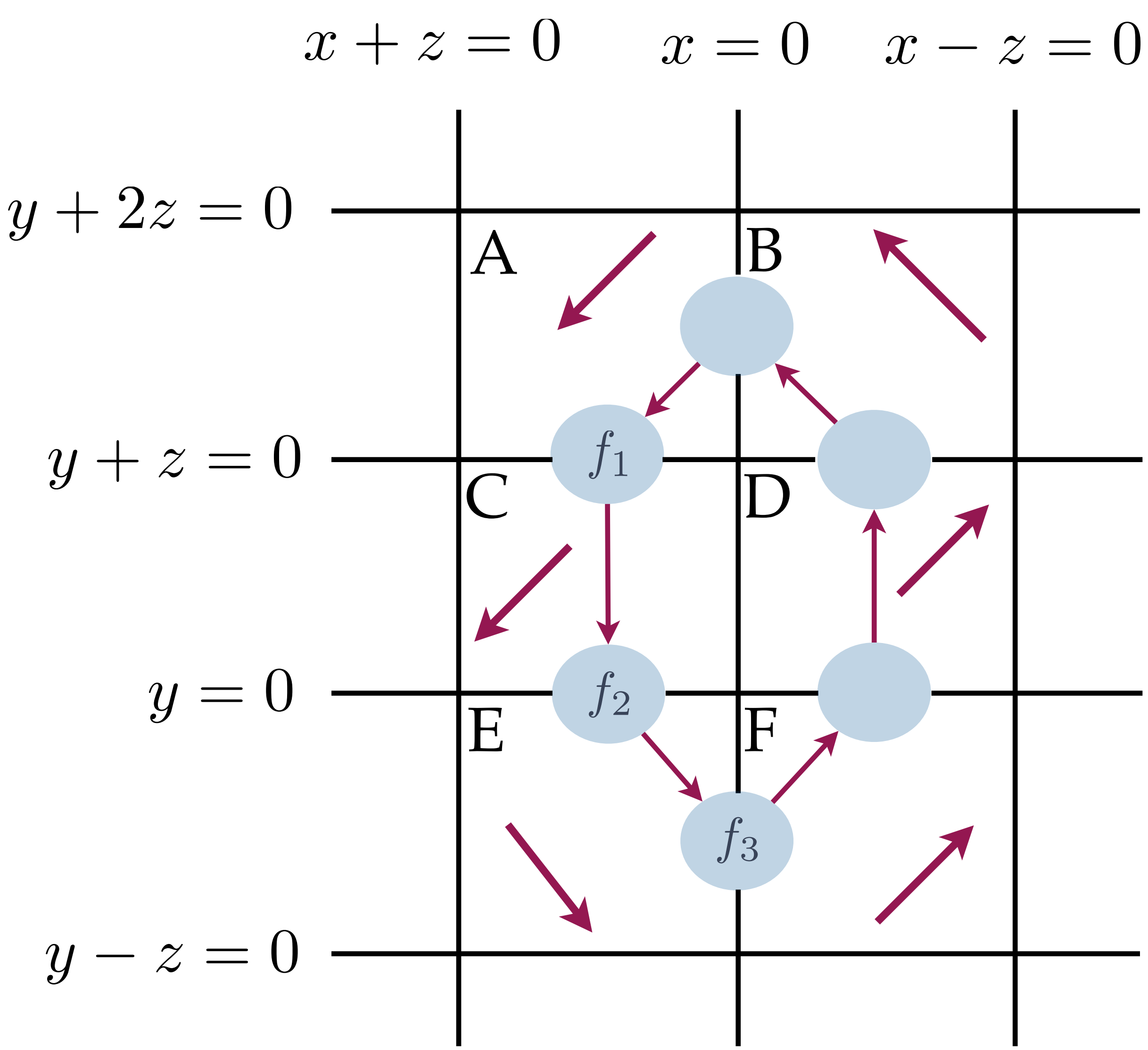
Refinement



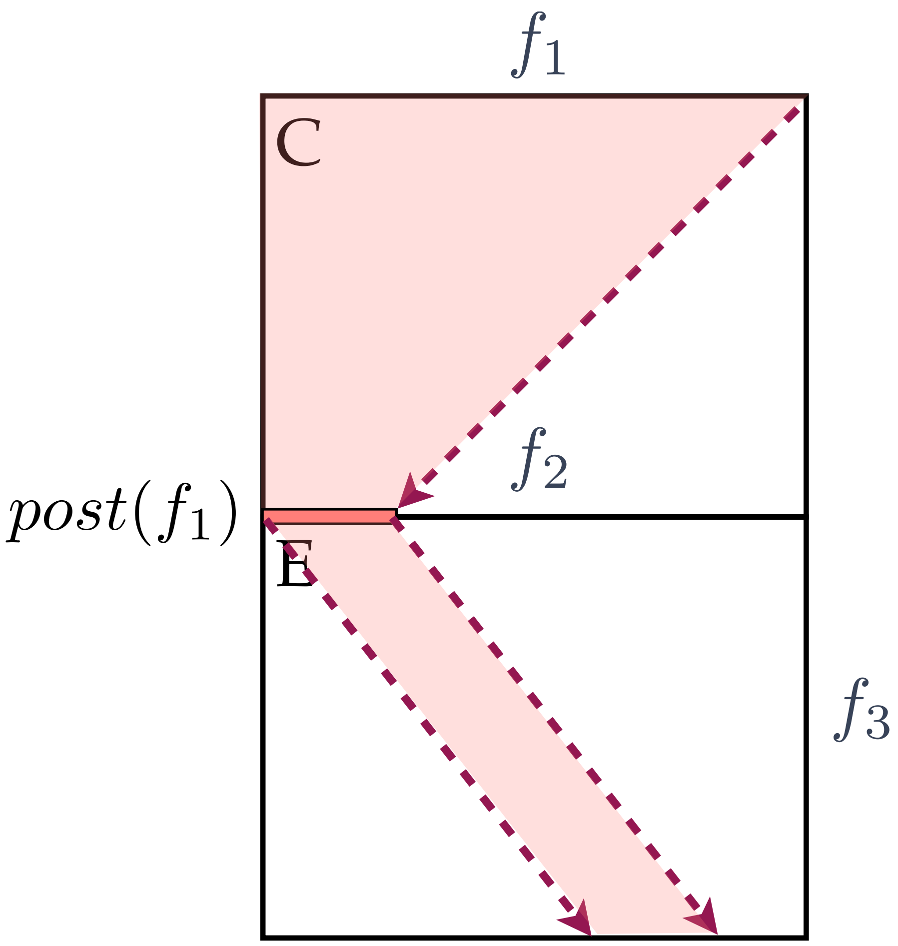
Spurious counterexample



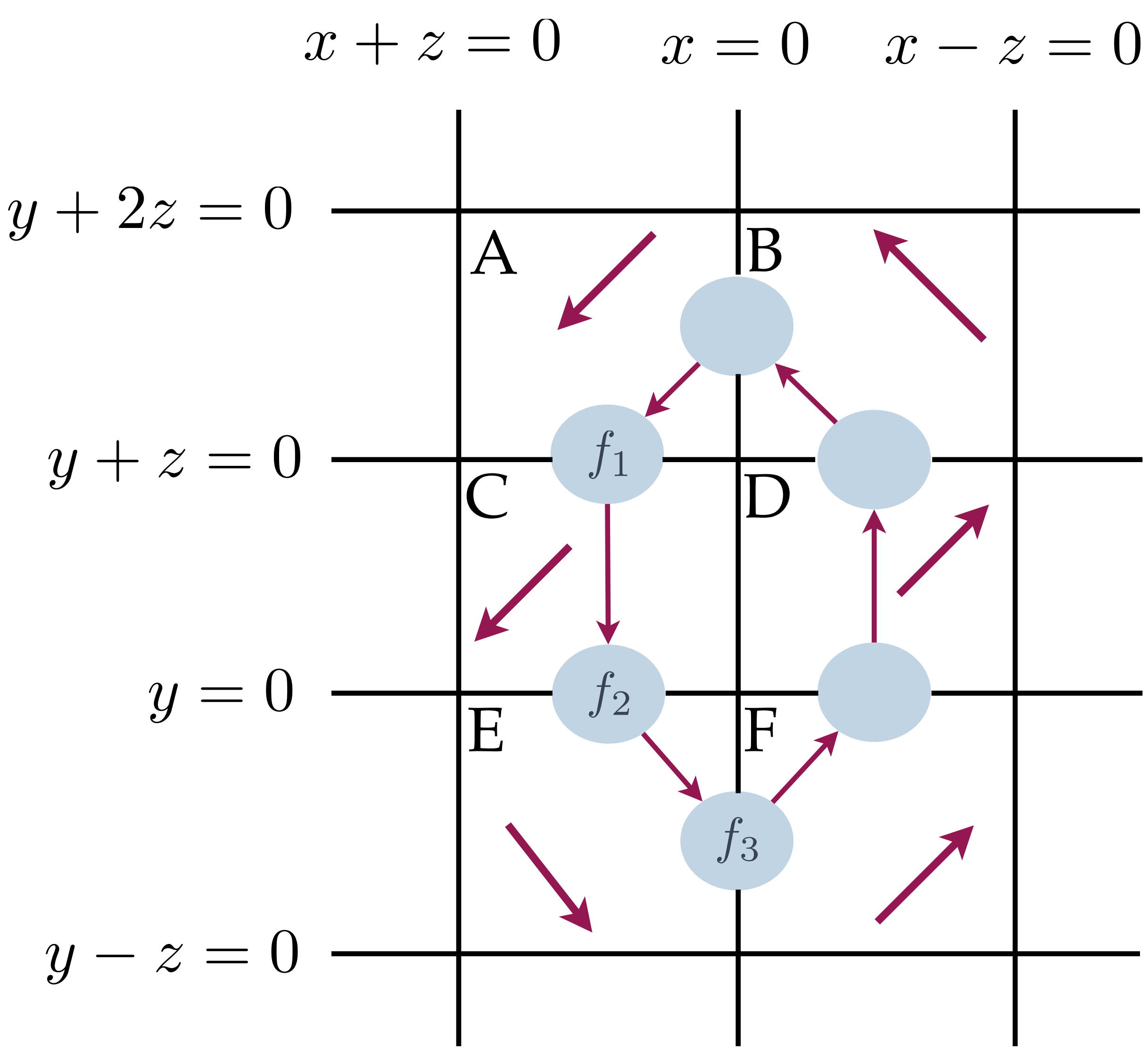
Refinement



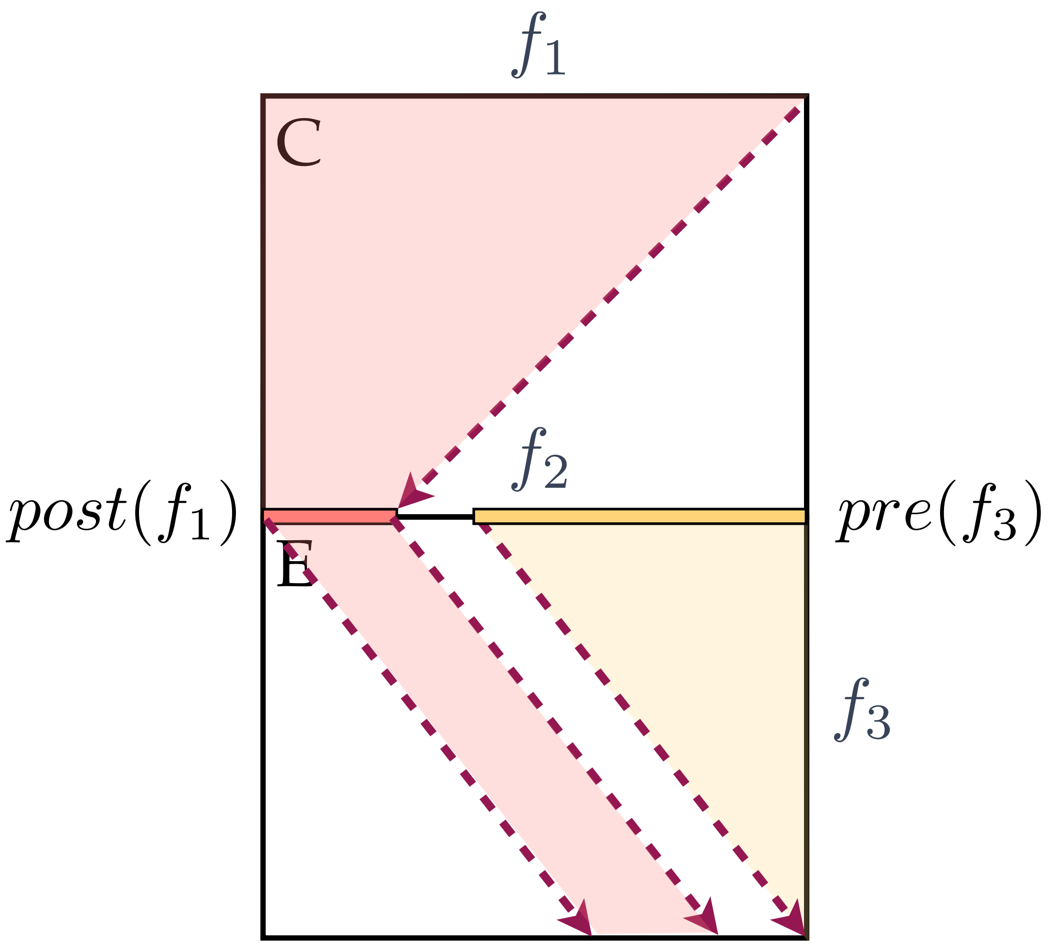
Spurious counterexample



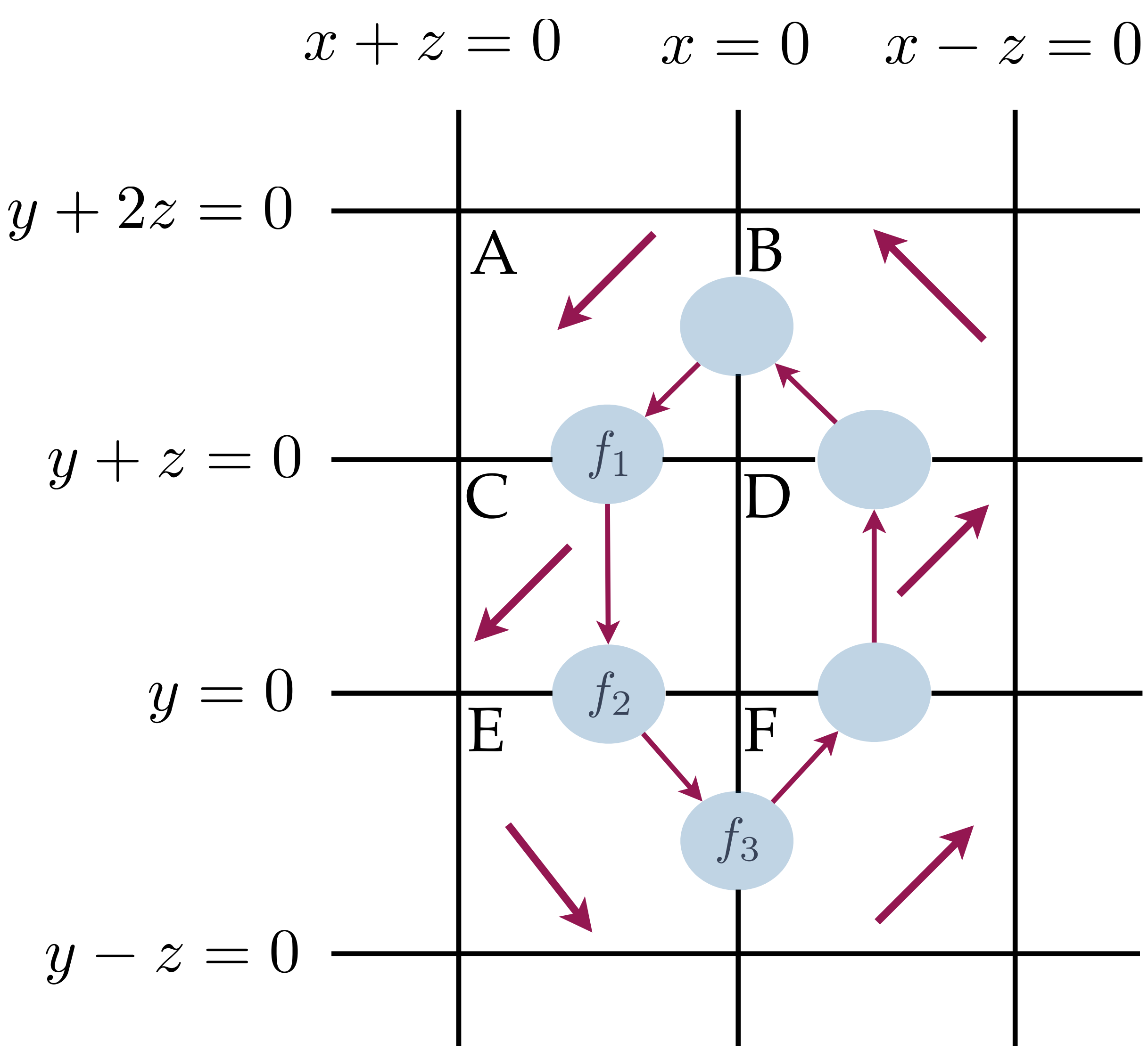
Refinement



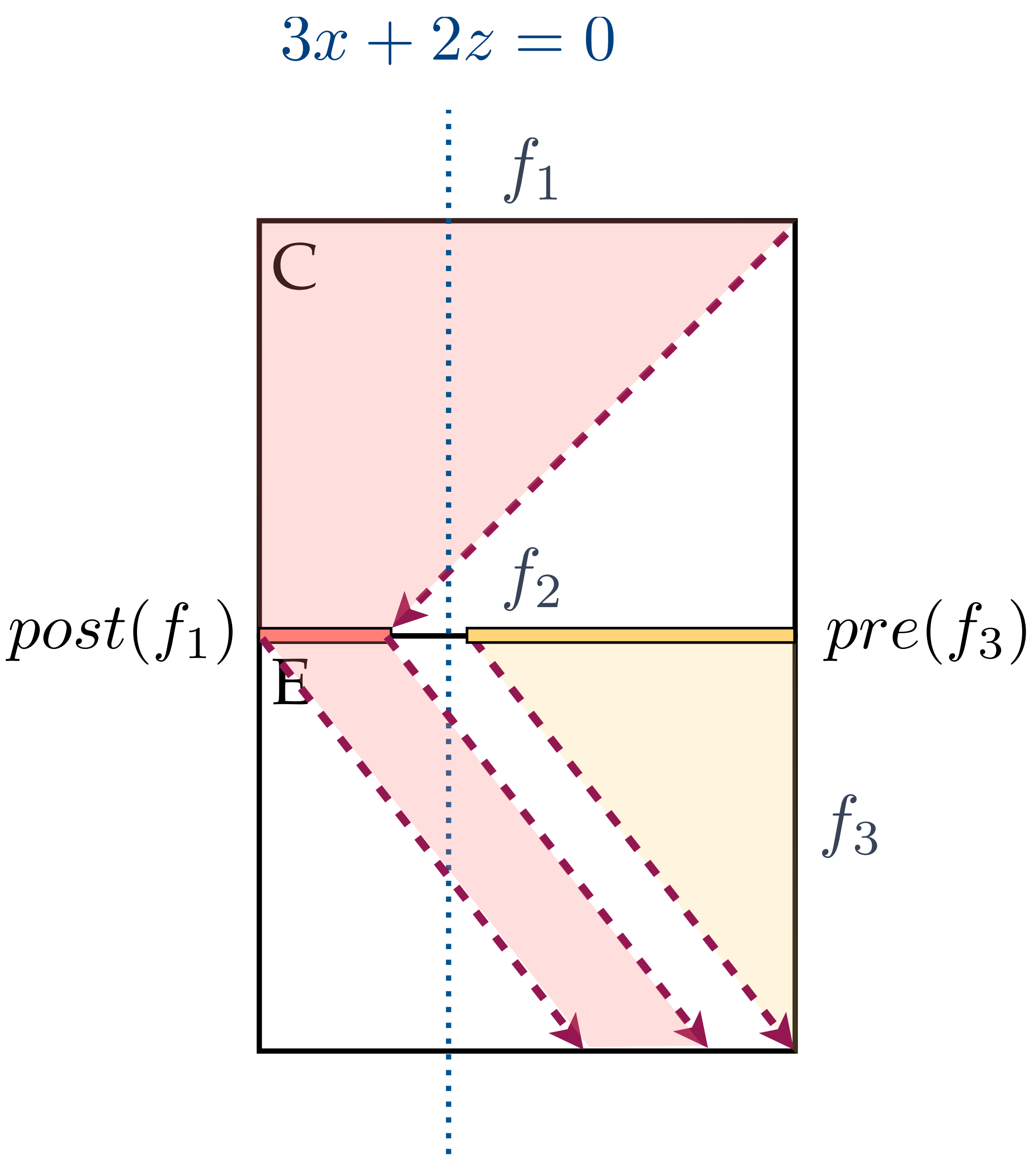
Spurious counterexample



Refinement

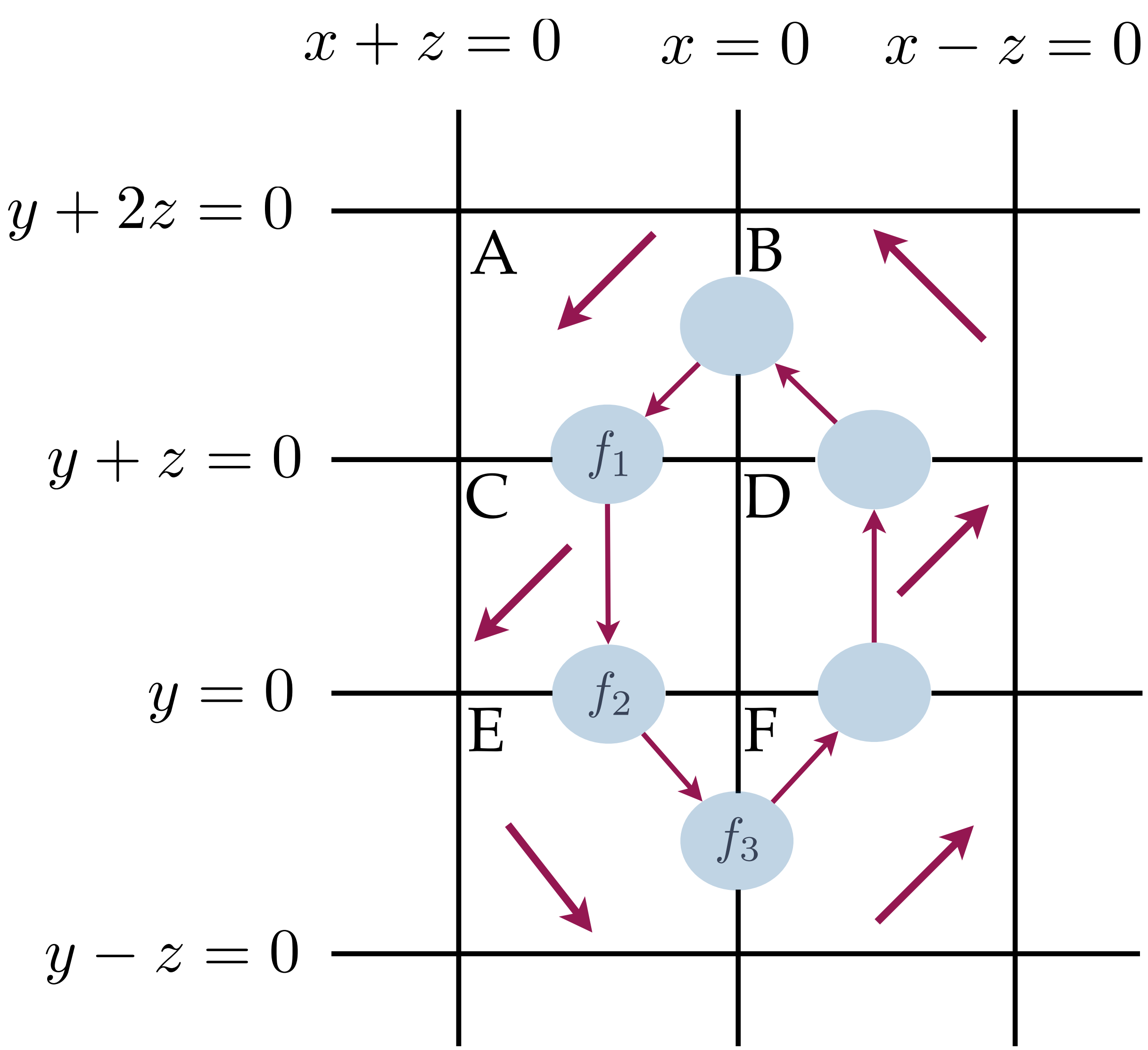


Spurious counterexample

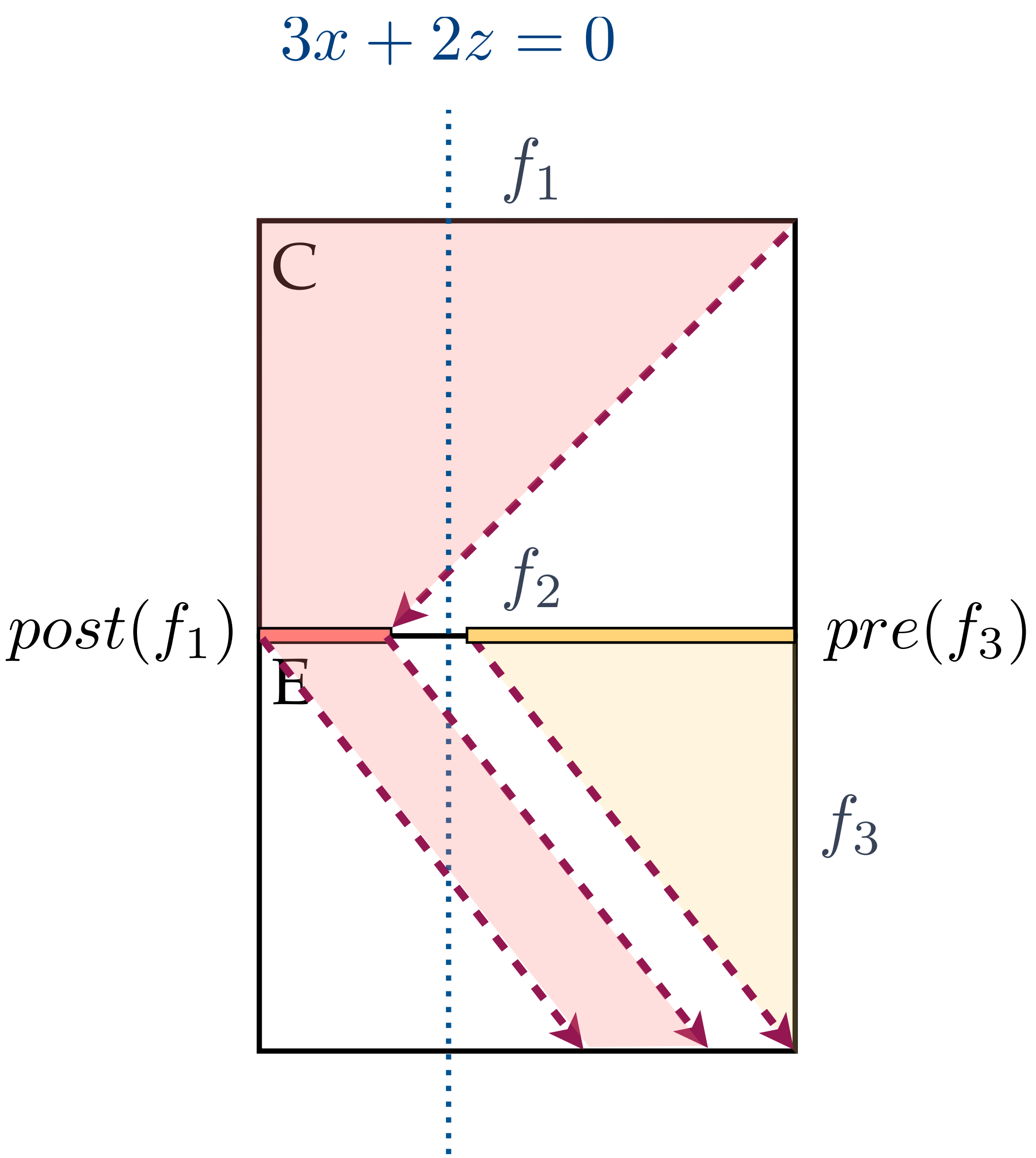


Separation predicate

Refinement



Spurious counterexample



Separation predicate

Post *and* ***pre-rechability*** *computations by means of Parma Polyhedral Library (**PPL**).*
Separation predicate *candidates are the linear constraints of the polyhedra to be separated.*

Running AVERIST

Hybrid Automaton type

linear

Given predicates

none

Uniform predicates grid ratio

0

Predicates from automaton

none

Maximum CEGAR iterations

2

CEGAR refinement type

selected

CEGAR predicates grid ratio

0

Optimization solver

GLPK solver

Hybrid automaton

```
var: x,y;
location: quad1, quad2, quad3, quad4;

loc: quad1;
    inv: x>=0 AND y>=0;
    dyn: dx==y AND dy==-4*x;
    guards:
        when y==0 goto quad4;

loc: quad2;
    inv: x<=0 AND y>=0;
    dyn: dx==10*y AND dy==-x;
    guards:
        when x==0 goto quad1;

loc: quad3;
    inv: x<=0 AND y<=0;
    dyn: dx==y AND dy==-4*x;
    guards:
        when y==0 goto quad2;

loc: quad4;
    inv: x>=0 AND y<=0;
    dyn: dx==10*y AND dy==-x;
    guards:
        when x==0 goto quad3;
```

Clear Send

AVERIST details

- Implemented in **Python**
- Parma Polyhedra Library (**PPL**) to manipulate polyhedral sets
- **GLPK** solver to compute the weights
- **NetworkX** Python package to define and analyse graphs
- Run through the mathematical software system **sage**

http://software.imdea.org/projects/averist/index.html

Experimental Comparison

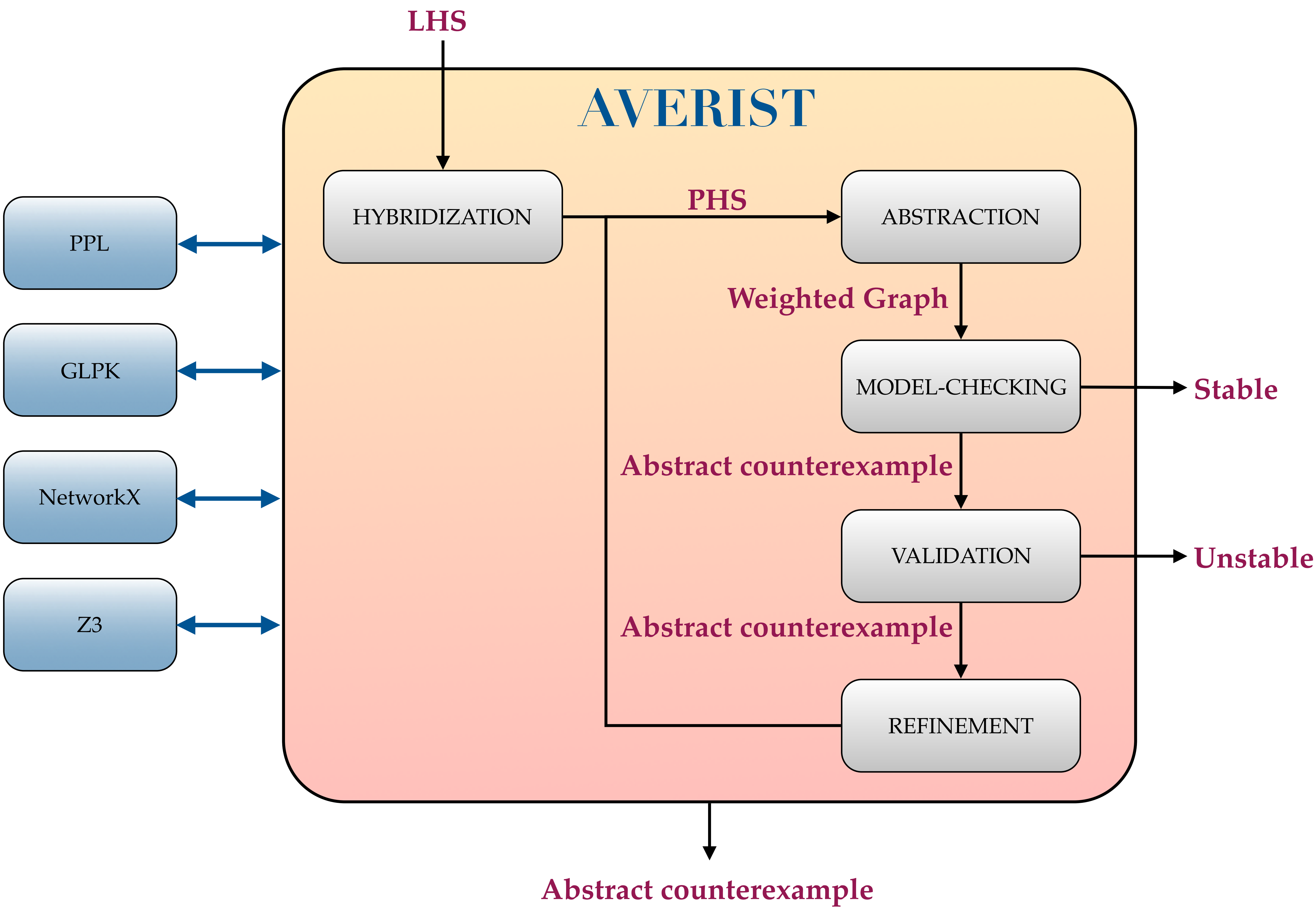
		AVERIST			STABHYLI		
Dimension/ name		Regions	Runtime	Proved Stability	Degree	LF found	Runtime
2D	AS1	129	31	Yes	6	Yes	8
	SS4 1	9	<1	Yes	8	–	452
	SS8 1	17	<1	Yes	6	–	443
	SS16 1	33	1	Yes	4	–	177
3D	AS 4	147	194	Yes	6	–	410
	SS4 4	771	484	Yes	2	Yes	75
	SS8 4	771	470	Yes	2	Yes	15
	SS16 4	771	568	Yes	2	Yes	138
4D	AS 7	81	625	Yes	2	–	12
	SS4 7	81	119	Yes	2	–	101
	SS8 7	153	234	Yes	2	–	1071
	SS16 7	297	533	Yes	2	–	339
	AS 9	–	out	No	4	Yes	34
	SS4 9	81	125	Yes	4	–	105
	SS8 9	153	247	Yes	2	–	16

- Averist proves stability in many more cases than Stabhyli
- Stabhyli can handle nonlinear systems
- Averist is more robust to numerical issues
- Underlying algorithms are highly parallelizable

Conclusion

- Averist implements an algorithmic approach for stability verification of linear and polyhedral hybrid systems
- Alternate approach to template based search
- Can sometimes conclude instability and return counterexamples
- Fully automated and parallelizable
- **Future work:**
  - Develop heuristics for scalability
  - Extend to nonlinear system

Questions?



http://software.imdea.org/projects/averist/index.html