AVERIST

Algorithmic VERifier for STability

Miriam García Soto
IMDEA Software Institute

Hybrid system

A dynamical system exhibiting a mixed discrete and continuous behavior.

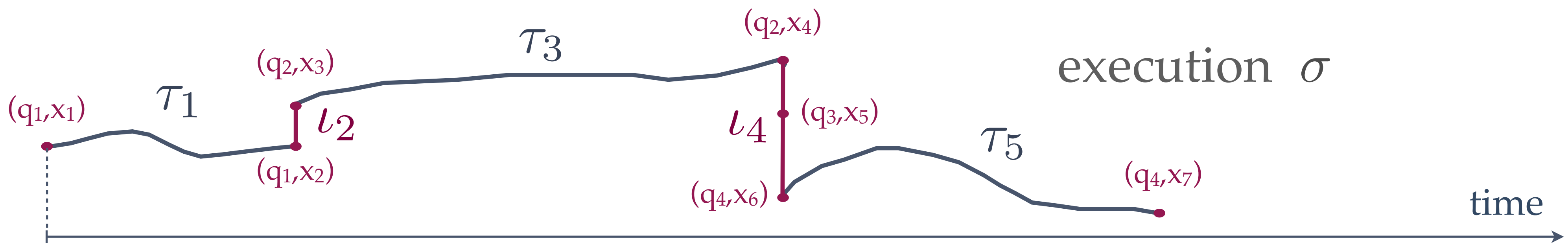
Pacemaker

Insuline pump

Polyhedral switched system (PSS)

$\mathcal{H} = (Q, X, \Sigma, \Delta)$

- Q finite set of control modes (discrete state space),
- $X = \mathbb{R}^n$ continuous state space,
- $\Sigma \subseteq Trans(Q, X)$ set of transitions and
- $\Delta \subseteq Traj(Q, X)$ set of trajectories.

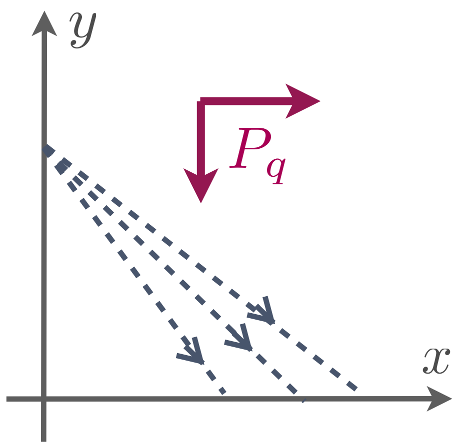


PSS: trajectories

$$\mathcal{H} = (Q, X, \Sigma, \Delta)$$

Polyhedral derivative

$$\frac{d}{dt}\tau(t) \in P_q$$



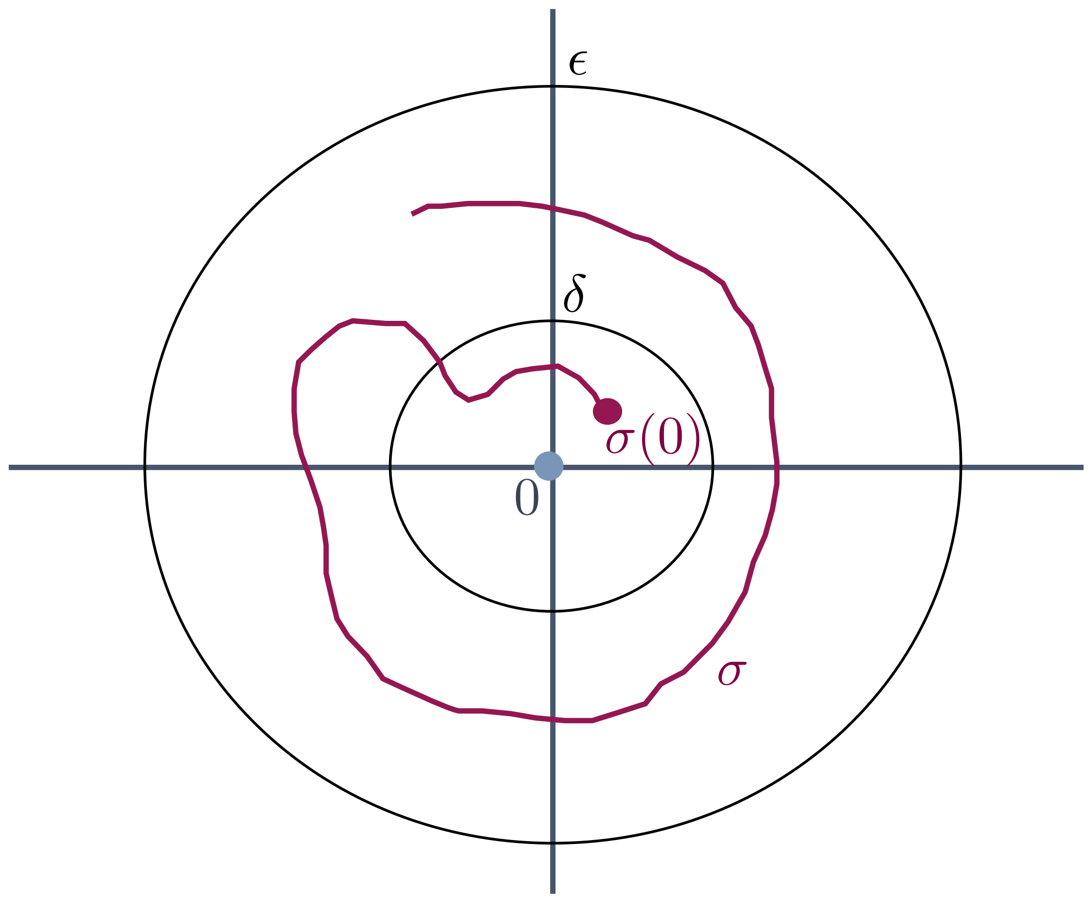
Stability

- Stability is a fundamental property in control system design and captures robustness of the system with respect to initial states or inputs.

- A system is stable when small perturbations in the input just result in small perturbations of the eventual behaviours.

- Classical notions of stability:
  - Lyapunov stability
  - Asymptotic stability

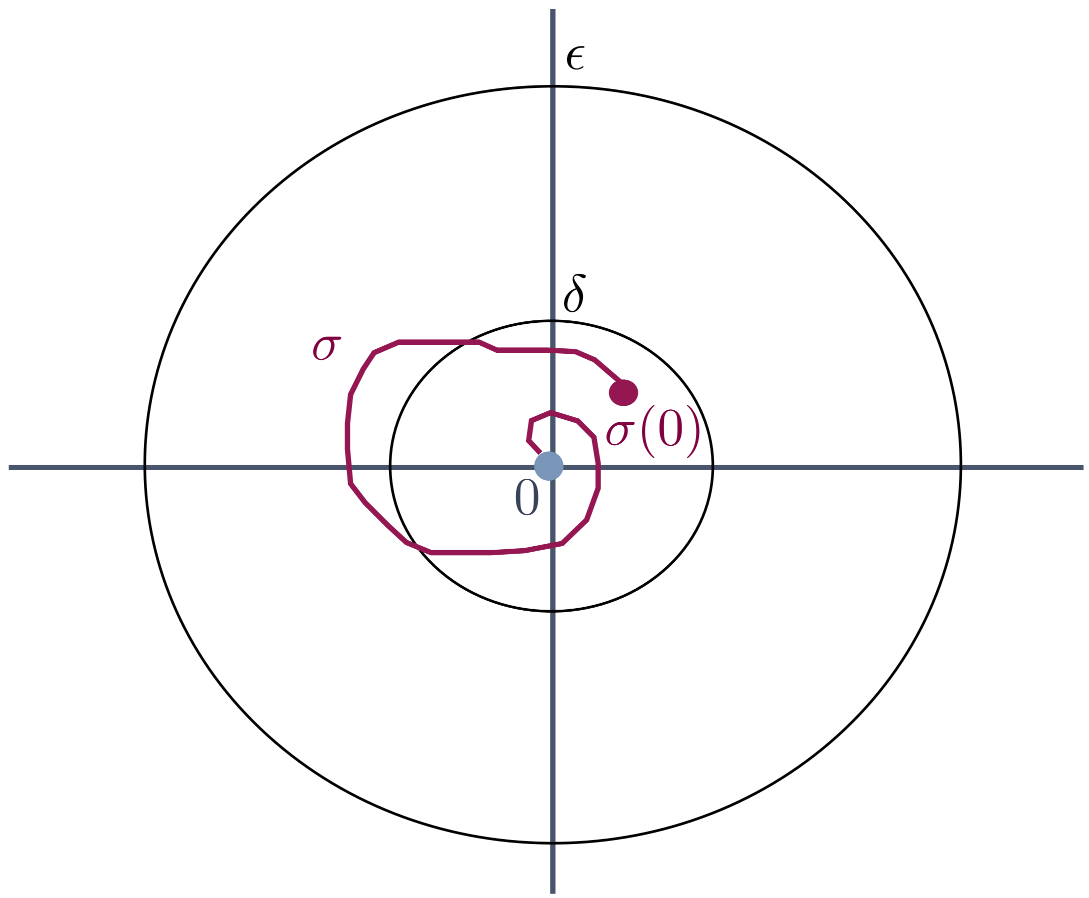
Lyapunov stability



The equilibrium point 0 is Lyapunov stable if

$$\forall \epsilon > 0 \; \exists \delta = \delta(\epsilon) > 0 : ||\sigma(0)|| < \delta \Rightarrow ||\sigma(t)|| < \epsilon \;\; \forall t \geqslant 0$$

Asymptotic stability



The equilibrium point 0 is asymptotic stable if it is Lyapunov stable and every execution converges to 0.

State of the art vs algorithmic approach

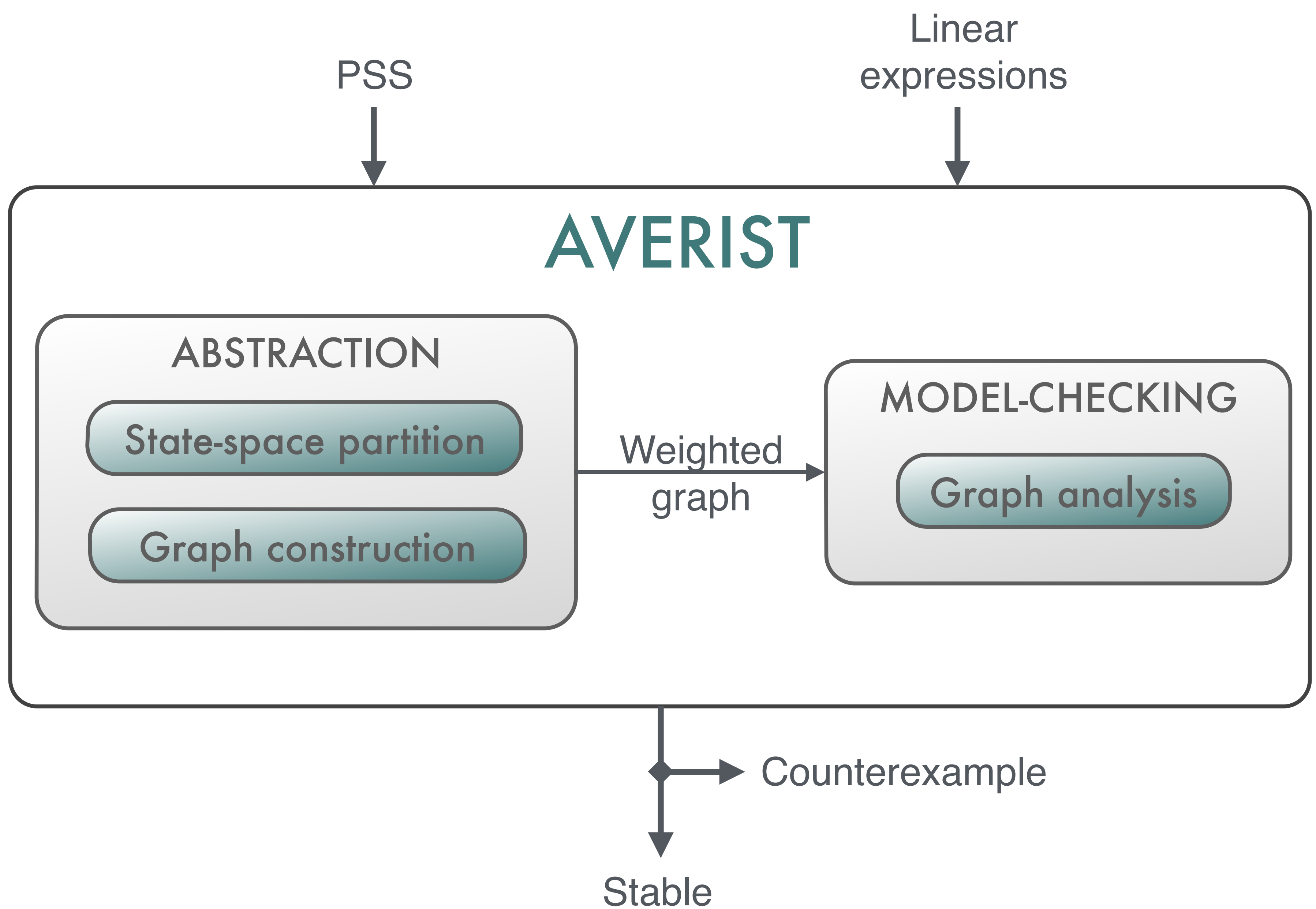
- Existence of Lyapunov function assures stability:
  - Choose a template L(x).
  - Check L(x) hold Lyapunov conditions.

State of the art vs algorithmic approach

- Existence of Lyapunov function assures stability:
  - Choose a template L(x).
  - Check L(x) hold Lyapunov conditions.

- Algorithmic approach implemented in `AVERIST`:
  - State space partition.
  - Abstract weighted graph construction.
  - Graph analysis.

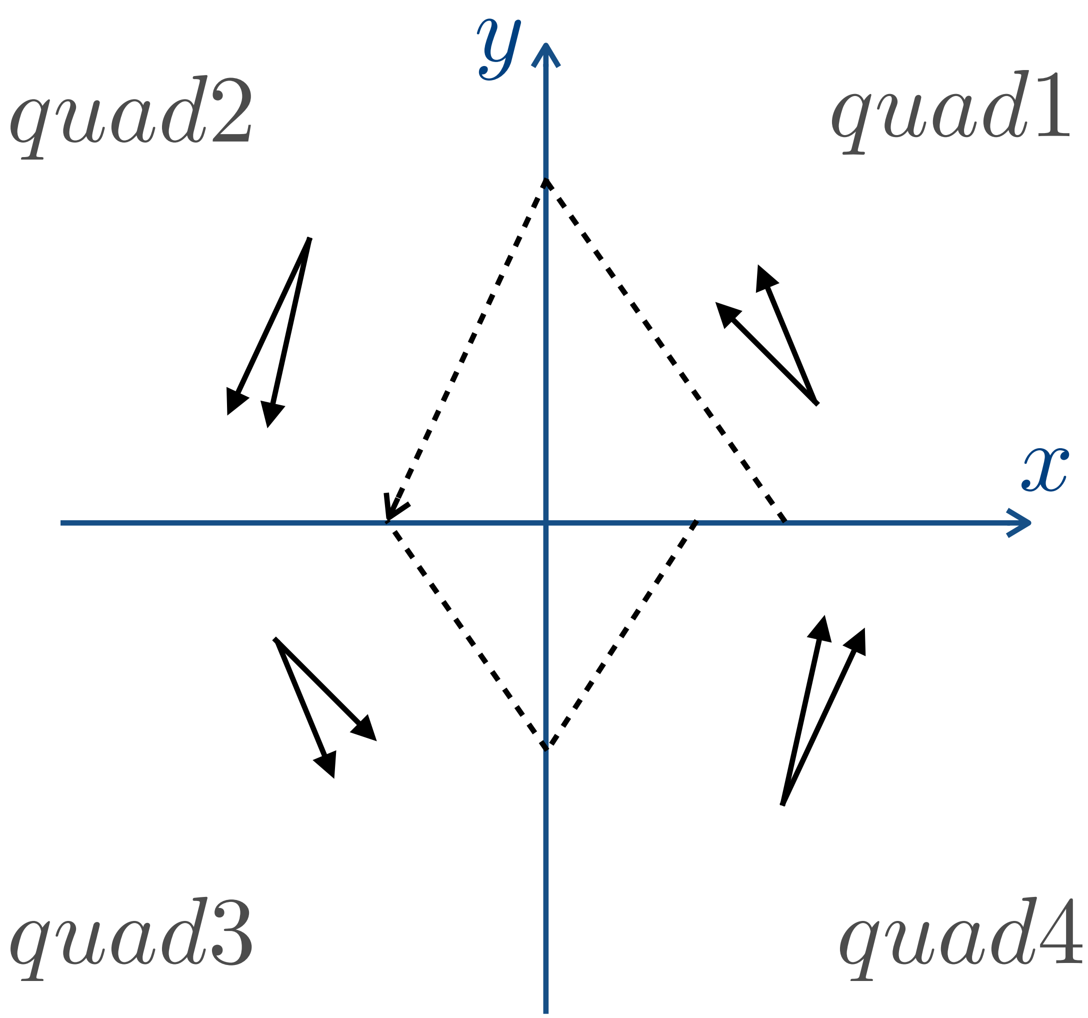
AVERIST architecture



AVERIST input grammar

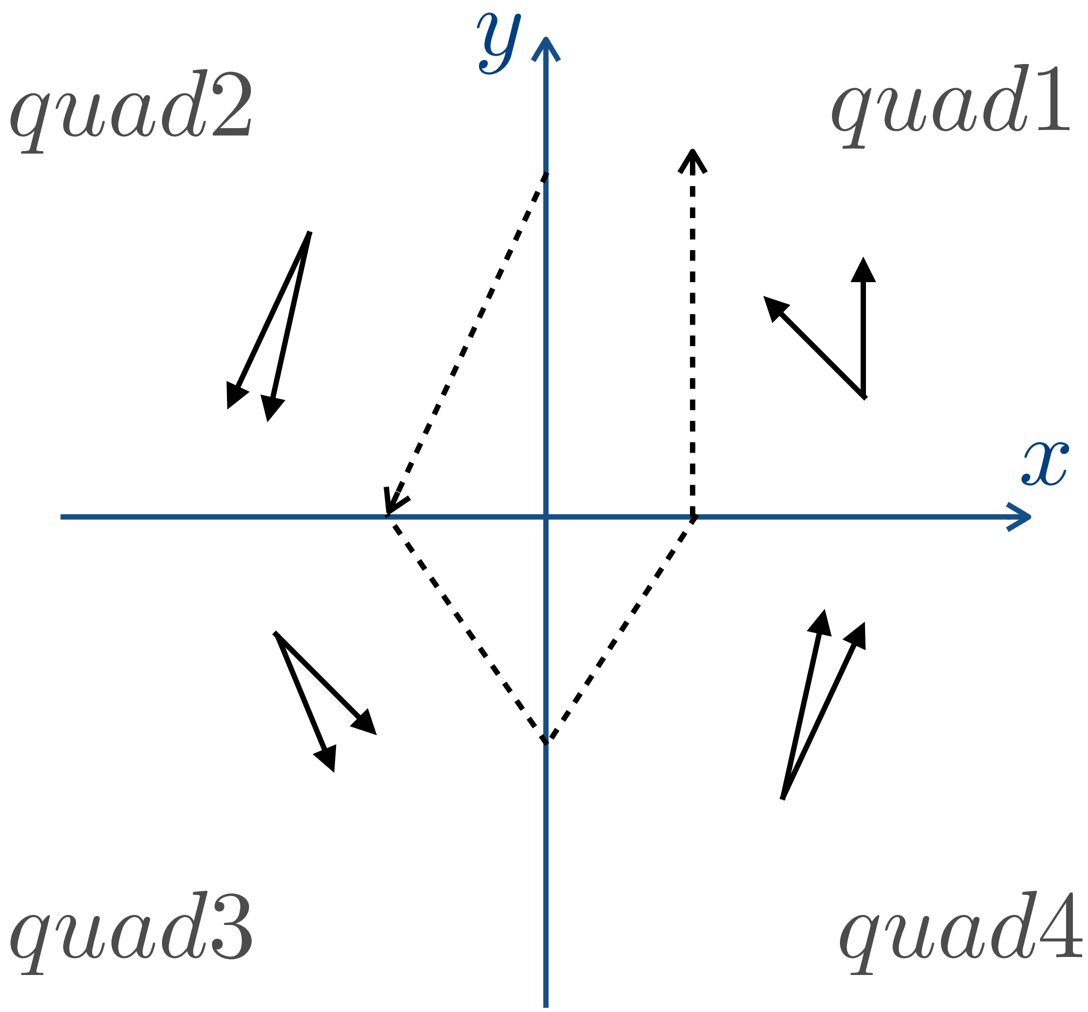
- Variables
  ```
  var:  x, y;
  ```
- Linear expressions
  ```
  x
  y
  x - y
  ```
- Locations
  ```
  location:  quad1, quad2, quad3, quad4;
  loc:  quad1;
  ```
  - Invariants
    ```
    inv:  x >= 0 AND y >= 0;
    ```
  - Dynamics
    ```
    dyn:  dx == -1 AND dy >=1 AND dy <= 2;
    ```
  - Guards
    ```
    guards:
        when y == 0 goto quad2;
    ```

2D stable polyhedral switched system



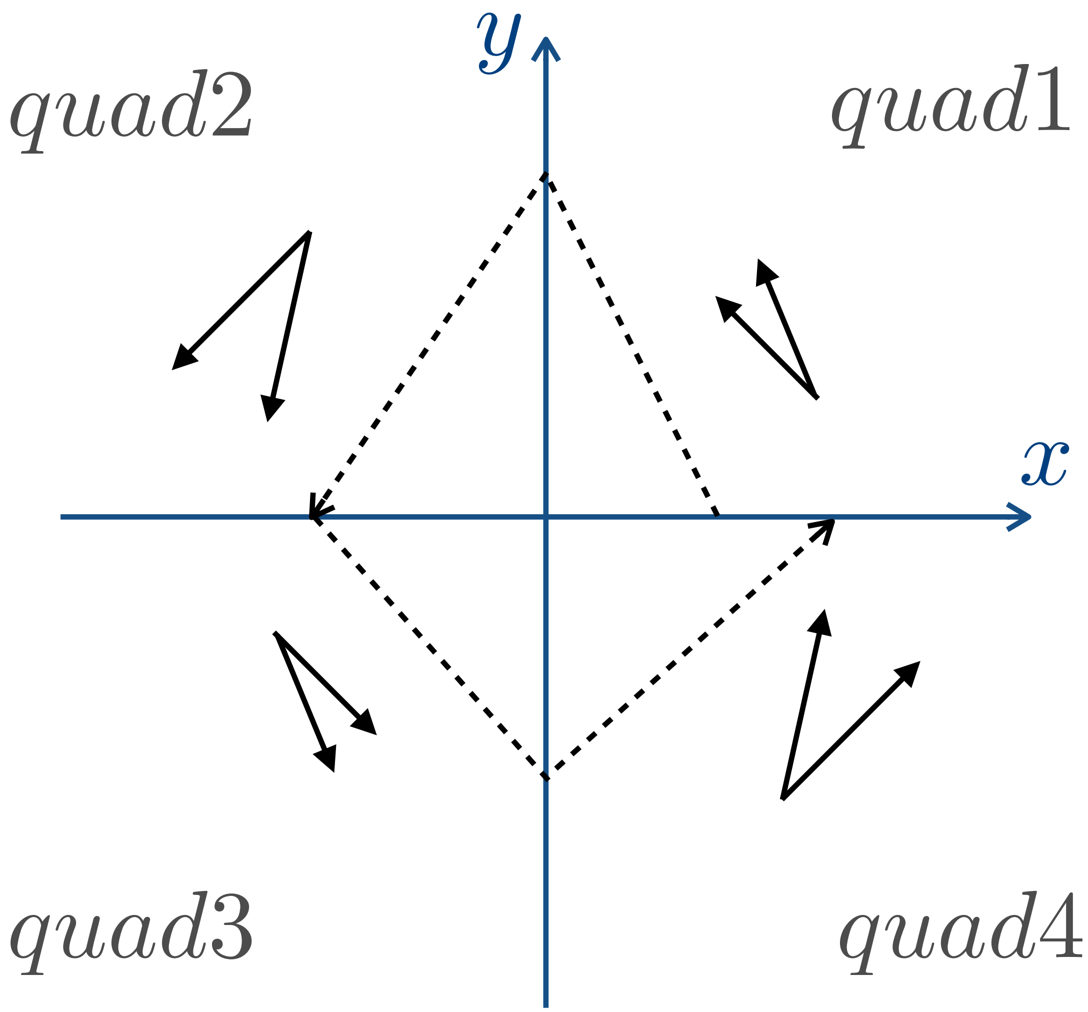

```
var :  x,y;
location:  quad1, quad2, quad3, quad4;
loc:  quad1;
    inv:  x >= 0 AND y >= 0;
    dyn:  dx == -1 AND dy >= 1 AND dy <= 2;
    guards:
        when x == 0 goto quad2;
loc:  quad2;
    inv:  x <= 0 AND y >= 0;
    dyn:  dx >= -2 AND dx <= -1 AND dy == -4;
    guards:
        when y == 0 goto quad3;
loc:  quad3;
    inv:  x <= 0 AND y <= 0;
    dyn:  dx == 1 AND dy <= -1 AND dy >= -2;
    guards:
        when x == 0 goto quad4;
loc:  quad4;
    inv:  x >= 0 AND y <= 0;
    dyn:  dx >= 1 AND dx <= 2 AND dy == 4;
    guards:
        when y == 0 goto quad1;
```

2D unstable polyhedral switched system
Blow-up




```
var :  x,y;
location:  quad1, quad2, quad3, quad4;
loc:  quad1;
    inv:  x >= 0 AND y >= 0;
    dyn:  dx >= -1 AND dx <= 0 AND dy == 1;
    guards:
        when x == 0 goto quad2;
loc:  quad2;
    inv:  x <= 0 AND y >= 0;
    dyn:  dx >= -2 AND dx <= -1 AND dy == -4;
    guards:
        when y == 0 goto quad3;
loc:  quad3;
    inv:  x <= 0 AND y <= 0;
    dyn:  dx == 1 AND dy <= -1 AND dy >= -2;
    guards:
        when x == 0 goto quad4;
loc:  quad4;
    inv:  x >= 0 AND y <= 0;
    dyn:  dx >= 1 AND dx <= 2 AND dy == 4;
    guards:
        when y == 0 goto quad1;
```

2D unstable polyhedral switched system Counterexample




```
var :  x,y;
location:  quad1, quad2, quad3, quad4;
loc:  quad1;
    inv:  x >= 0 AND y >= 0;
    dyn:  dx == -2 AND dy >= 1 AND dy <= 2;
    guards:
        when x == 0 goto quad2;
loc:  quad2;
    inv:  x <= 0 AND y >= 0;
    dyn:  dx >= -2 AND dx <= -1 AND dy == -2;
    guards:
        when y == 0 goto quad3;
loc:  quad3;
    inv:  x <= 0 AND y <= 0;
    dyn:  dx == 1 AND dy <= -1 AND dy >= -2;
    guards:
        when x == 0 goto quad4;
loc:  quad4;
    inv:  x >= 0 AND y <= 0;
    dyn:  dx >= 1 AND dx <= 2 AND dy == 2;
    guards:
        when y == 0 goto quad1;
```

AVERIST: dependencies

- 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.
- All included in the mathematical software system sage.

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