Solving Optimization Problems on Hybrid Systems by Graph Exploration

Kunihiko HIRAISHI

School of Information Science,

Japan Advanced Institute of Science and Technology

Aim

- Tool Support for Verification and Optimization of Hybrid Dynamical Systems.
- The tool KCLP-HS is based on Constraint Logic Programming Technology, and can handle
  - Logical constraints
  - Numerical constraints (linear inequalities)
  - Depth-first search of logical alternative by backtracking
  - Linear/Quadratic optimizer
  - Operations on convex polyhedra (intersection, projection, convex-hull, negation, emptyness, etc)

An optimization problem on Hybrid Systems

- The set of variables are partitioned into state variables $x_i(t)$, input variables $u_i(t)$, and parameters p_i.
- We assume that the dynamics of the system is uniquely determined by $x(t) = [x_i(t)]$, $u(t) = [u_i(t)]$, and $p = [p_i]$.
- Then the objective is to find values of $u(t)$ and p such that (i) the run (trajectory) satisfies given constraints and (ii) they minimize a given objective function of the form

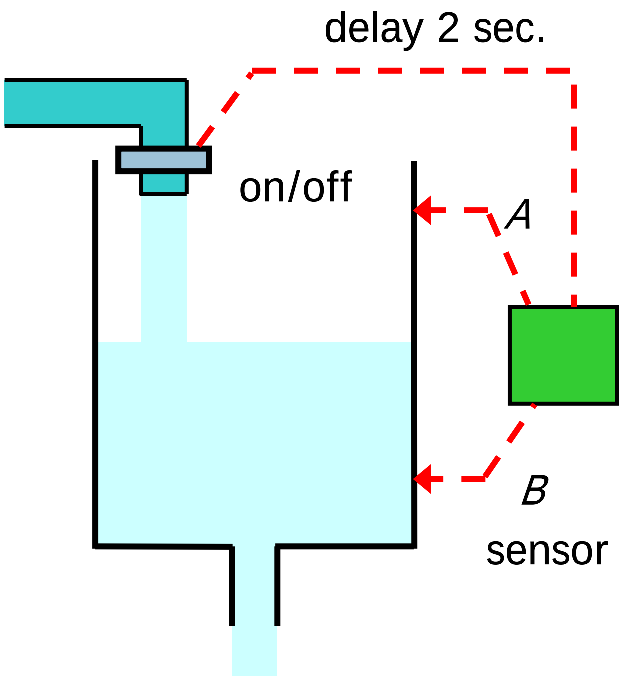
$$J(\pi) := \int_0^h N(x(t), u(t), p, t)dt$$

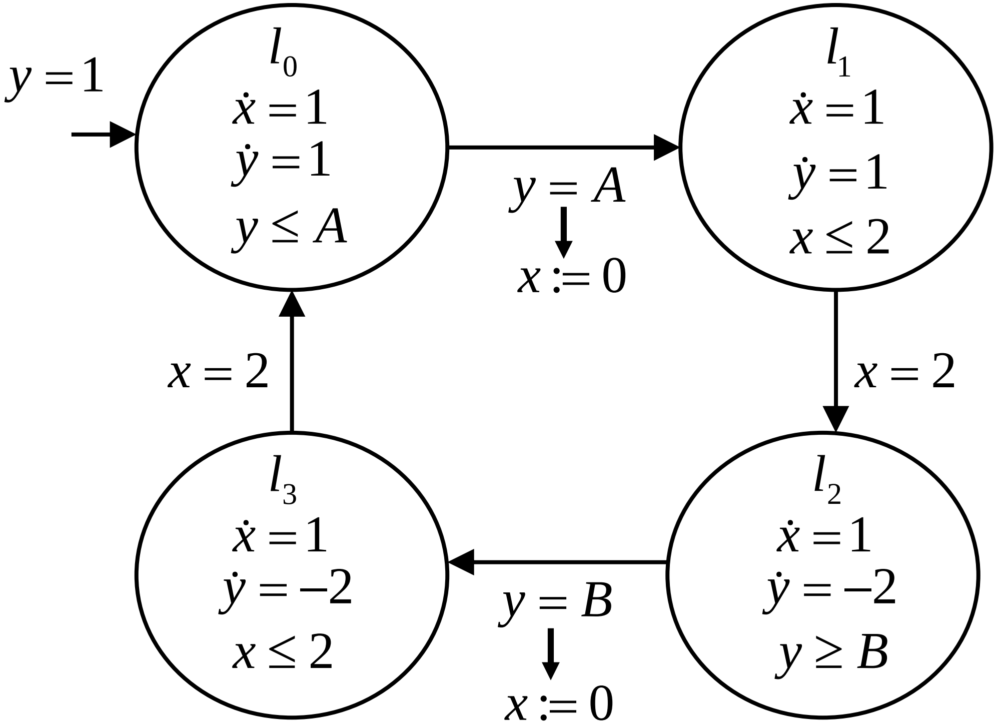
where $N(\cdot) \geq 0$.

Problem1:
Parameter design of a class of linear HA

- Linear Hybrid Automata: the continuous dynamics is in the form $\dot{x} = k$.
- No inputs (autonomous system)
- Each formula in activities and invariants has no disjunction.

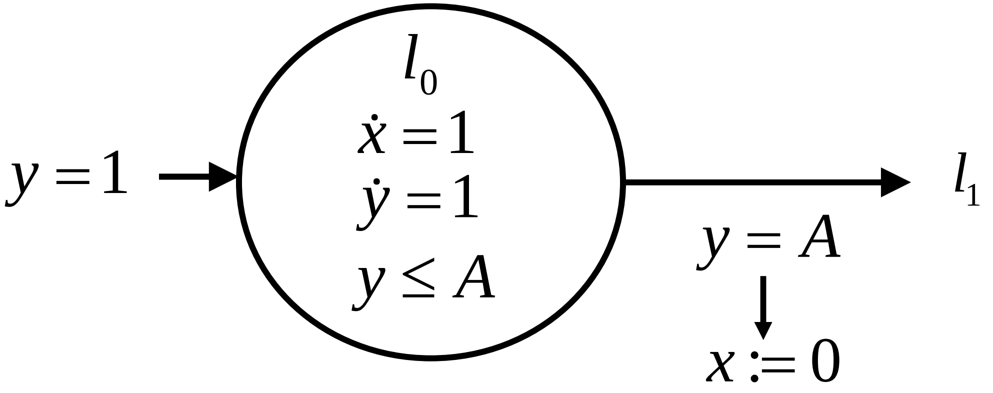
Example.

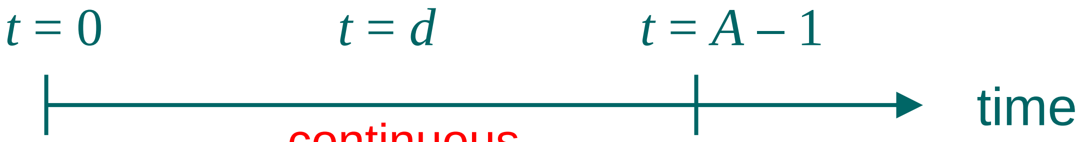




Find values of A and B so that $1 \le y \le 12$ always holds and the number of discrete transitions during a fixed time interval is minimized.

Simulating linear HA by CLP





continuous dynamics

continuous state

$$\boxed{\begin{array}{l} x(t) = ? \\ y(t) = 1 \end{array}}$$

$$\boxed{\begin{array}{l} x(t) = x + d \\ y(t) = y + d \end{array}}$$

$$\boxed{\begin{array}{l} y(t) = A \rightarrow \\ x(t) = 0 \end{array}}$$

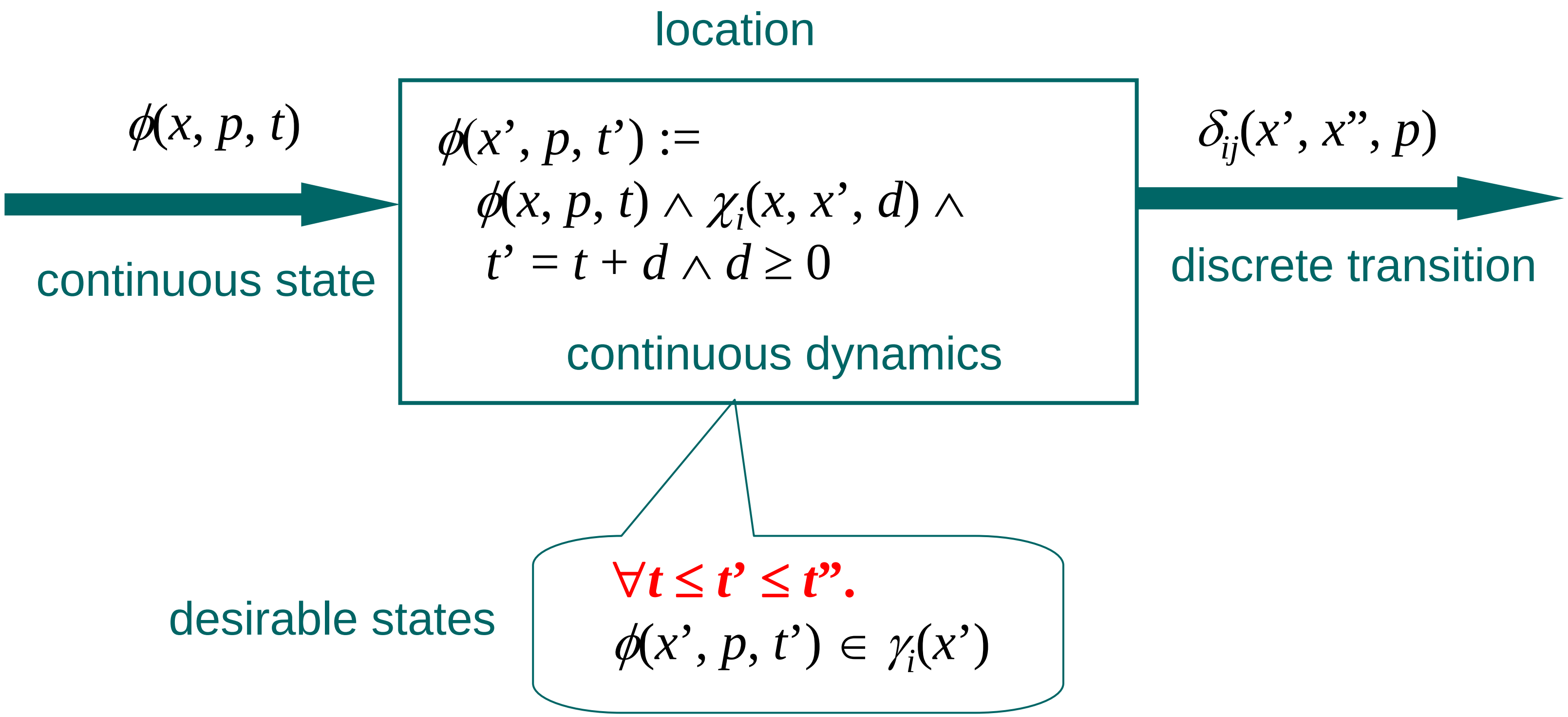
discrete transition

$$\boxed{\forall t \geq 0.\ 1 \leq y(t) \leq 12}$$

spec.

Simulating linear HA by CLP

In general,



location

$\phi(x, p, t)$

continuous state

$\phi(x', p, t') :=$
$\phi(x, p, t) \wedge \chi_i(x, x', d) \wedge$
$t' = t + d \wedge d \geq 0$

continuous dynamics

$\delta_{ij}(x', x'', p)$

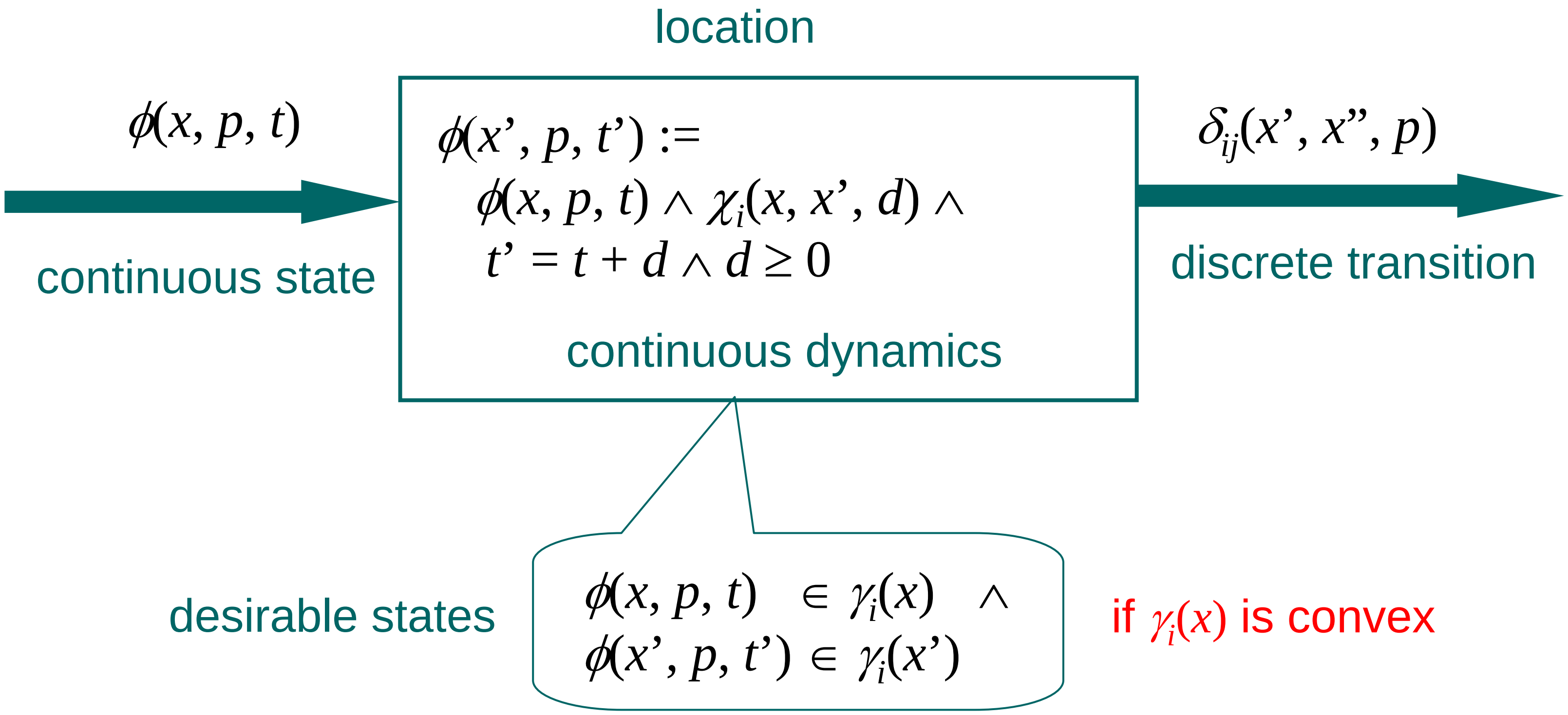
discrete transition

desirable states

$\forall t \leq t' \leq t''.$
$\phi(x', p, t') \in \gamma_i(x')$

where $\phi(x, p, t)$, $\chi_i(x, x', d)$ and $\delta_{ij}(x', x'', p)$ are convex polyhedra.

Simulating linear HA by CLP



location

$\phi(x, p, t)$

continuous state

$\phi(x', p, t') :=$
$\phi(x, p, t) \wedge \chi_i(x, x', d) \wedge$
$t' = t + d \wedge d \geq 0$

continuous dynamics

$\delta_{ij}(x', x'', p)$

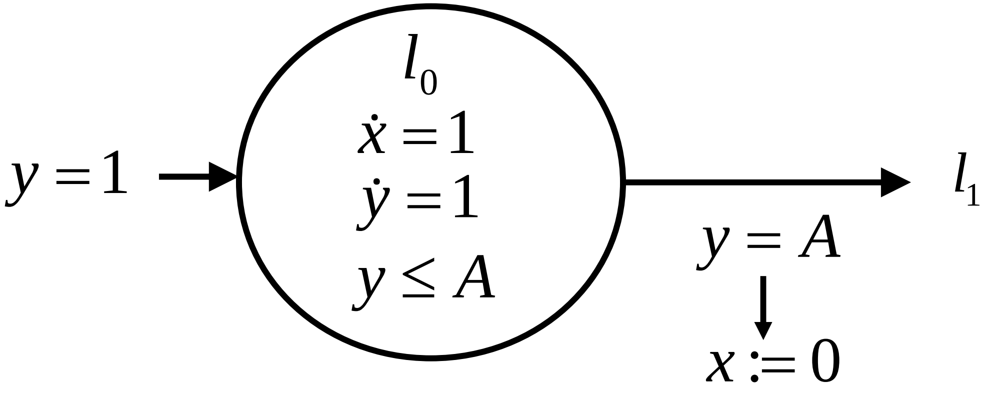
discrete transition

desirable states

$\phi(x, p, t) \in \gamma_i(x) \wedge$
$\phi(x', p, t') \in \gamma_i(x')$

if $\gamma_i(x)$ is convex

Simulating linear HA by CLP



```
l0(X, Y, A, B, T):-
        inv(X, Y),
        X1 = X + D, Y1 = Y + D,
        T1 = T + D, D >= 0,
        Y1 = A, X2 = 0, Y2 = Y1,
        inv(X1, Y1),
        l1(0, Y1, A, B, T1).
inv(X, Y):- 1 <= Y, Y <= 12.
```

$\phi(x, p, t)$

$\phi(x, p, t) \in \gamma_0(x)$

$\chi_0(x, x', d)$

$t' = t + d, d \geq 0$

$\delta_{01}(x', x'', p)$

$\phi(x', p, t') \in \gamma_0(x')$

$\Rightarrow$ next location

Simulating linear HA CLP

Goal

l0(X, 1, A, B, 0)

Constraints

$\Theta_0 = \{\ \}$

Simulating linear HA CLP

Goal

```
l0(X, 1, A, B, 0)
```

Constraints

$\Theta_0 = \{ \}$

↕ unification

```
l0(X, Y, A, B, T):-
        inv(X, Y),
        X1 = X + D, Y1 = Y + D,
        T1 = T + D, D >= 0,
        Y1 = A, X2 = 0, Y2 = Y1,
        inv(X1, Y1),
        l1(0, Y1, A, B, T1).
```

Simulating linear HA CLP

Goal

```
l0(X, 1, A, B, 0)
```

Constraints

$\Theta_0 = \{ \}$

↕ unification

```
l0(X, 1, A, B, 0):-
        inv(X, 1),
        X1 = X + D, Y1 = 1 + D,
        T1 = 0 + D, D >= 0,
        Y1 = A, X2 = 0, Y2 = Y1,
        inv(X1, Y1),
        l1(0, Y1, A, B, T1).
```

Simulating linear HA CLP

Goal

```
inv(X, 1),
X1 = X + D, Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

$\Theta_0 = \{ \}$

unification

```
inv(X, 1):- 1 <= 1, 1 <= 12.
```

Simulating linear HA CLP

Goal

```
1<= 1, 1 <= 12,
X1 = X + D, Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

$\Theta_0 = \{ \}$

Simulating linear HA CLP

Goal

```
1<= 1, 1 <= 12,
X1 = X + D, Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

$\Theta_0 = \{ \}$

Simulating linear HA CLP

Goal

X1 = X + D, Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).

Constraints

$\Theta_0 = \{ \}$

1<= 1, 1 <= 12 is true

Simulating linear HA CLP

Goal

```
X1 = X + D, Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

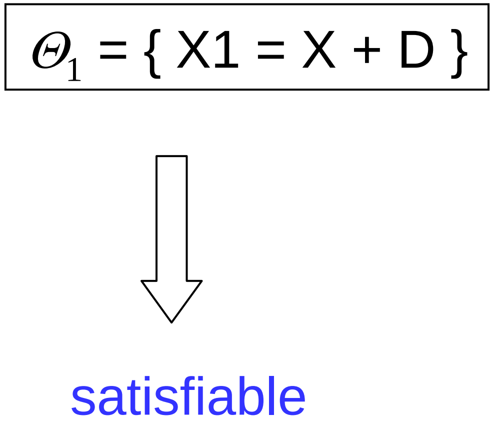
$\Theta_0 = \{ \}$

Simulating linear HA CLP

Goal

```
Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints



Θ_1 = { X1 = X + D }

satisfiable

Simulating linear HA CLP

Goal

```
Y1 = 1 + D,
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

Θ_1

Simulating linear HA CLP

Goal

```
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

$\Theta_2 = \Theta_1 \cup \{ Y1 = 1 + D \}$

⇓

satisfiable

Simulating linear HA CLP

Goal

```
T1 = 0 + D, D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

Constraints

$\boxed{\Theta_2}$

Simulating linear HA CLP

Goal

```
D >= 0,
Y1 = A, X2 = 0, Y2 = Y1,
inv(X1, Y1),
l1(0, Y1, A, B, T1).
```

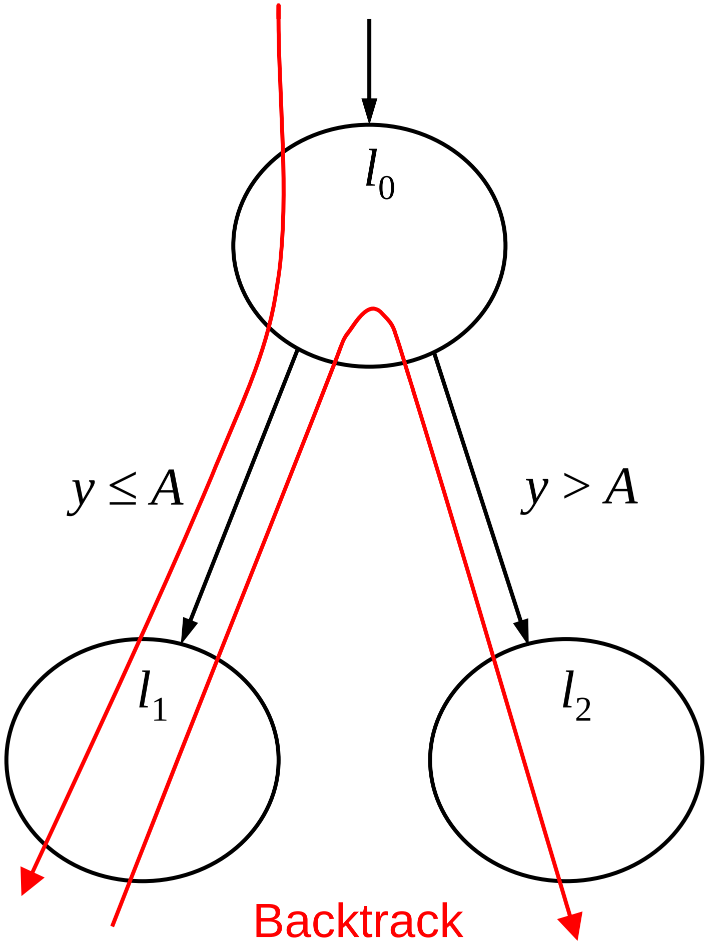
Constraints

$$\Theta_3 = \Theta_2 \cup \{ \mathrm{T1} = 0 + \mathrm{D} \}$$

⇩

satisfiable

Conditional branches



```
l0(X, Y, A, T) :-
  ...,
  (Y1 <= A,
     l1(X1, Y1, T1);
   Y1 > A,
     l2(X1, Y1, T1)
  ).
```

OR

CLP code for optimization

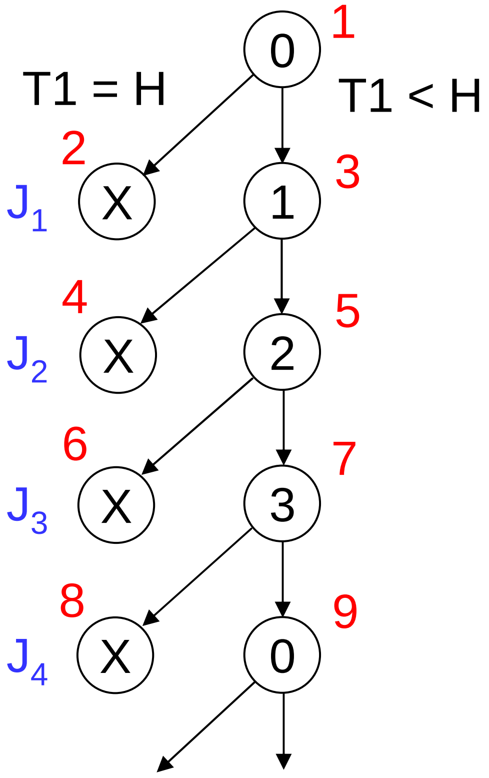
```
loc0(X, Y, A, B, T, H, W, J):-
        W <= @optval,
        inv(0, X2, Y2),
        X1 = X + D, Y1 = Y + D,
        T1 = T + D, D >= 0,
        (T1 = H,
                inv(0, X1, Y1), J = W;
         T1 < H,
                inv(0, X1, Y1),
                Y1 = A, X2 = 0, Y2 = Y1,
                loc1(X2, Y2, A, B, T1, H, W + 1, J)
        ).
loc1(X, Y, A, B, T, H, W, J):-  ...

go(H, A, B, Z):-
        min(J, loc0(0, 1, A, B, 0, H, 0, J)), project([A, B], Z).
```

H : the length of the period
W: the accumulated number of discrete transitions
J : the value of the objective function

minimize J subject to loc0(...)

@optval



@optval keeps the temporal optimal value, e.g., @optval at step 7 is min { J_1, J_2, J_3 }.

This is used for *pruning branches*.

Depth-first search by backtracking

Results

```
| ?- go(60, A, B, Z).
A = 10
B = 5
Z = [(-1) * 5 = -5, (-1) * 10 = -10]

*** yes ***

0.0700 sec.
```

$$A = 10,\ B = 5$$

```
| ?- go(30, A, B, Z).
A = 10  - _284
B = 5.666667  -2 * _284 -0.333333 * _287
Z = [(-3) * B + 6 * A >= 43, 1 * B >= 5, (-1) * A >= -10]
_284 >= 0
_287 >= 0
-0.666667 =  - _293 -2 * _284 -0.333333 * _287
_293 >= 0

*** yes ***

0.0300 sec.
```

$$-3B + 6A \geq 43 \wedge B \geq 5 \wedge A \leq 10$$

Problem 2:

Piecewise linear systems with control inputs

$$x(t+1) = 0.8\begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

$$\alpha(t) = \begin{cases} \pi/3 & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) \geq 0 \\ -\pi/3 & \text{if } \begin{bmatrix} 1 & 0 \end{bmatrix} x(t) < 0 \end{cases}$$

$$x(t) \in \begin{bmatrix} -10 & 10 \end{bmatrix} \times \begin{bmatrix} -10 & 10 \end{bmatrix}, u(t) \in \begin{bmatrix} -1 & 1 \end{bmatrix}$$

Optimal control problem

Given the initial state x_0 and the final state x_f at time h, find control inputs at each time step that minimize the quadratic objective function

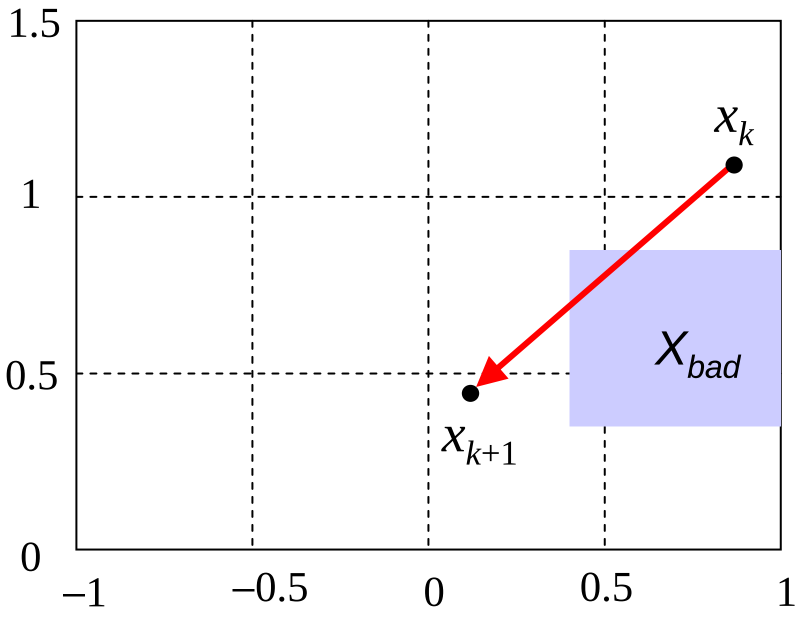
$$J_h(x_0, x_f, u) = \sum_{t=0}^{h-1} \|u(t)\|_{Q_1}^2 + \|x(t) - x_f\|_{Q_2}^2$$

and the run does not intersect a region X_{bad} given by a set of convex polyhedra, where

$$X_{bad} = \{(x_1, x_2) \mid 0.4 < x_1 \leq 10 \wedge 0.3 < x_2 < 0.8\},$$
$$x_0 = [-1, 1.5]^T, x_f = [0, 0]^T.$$

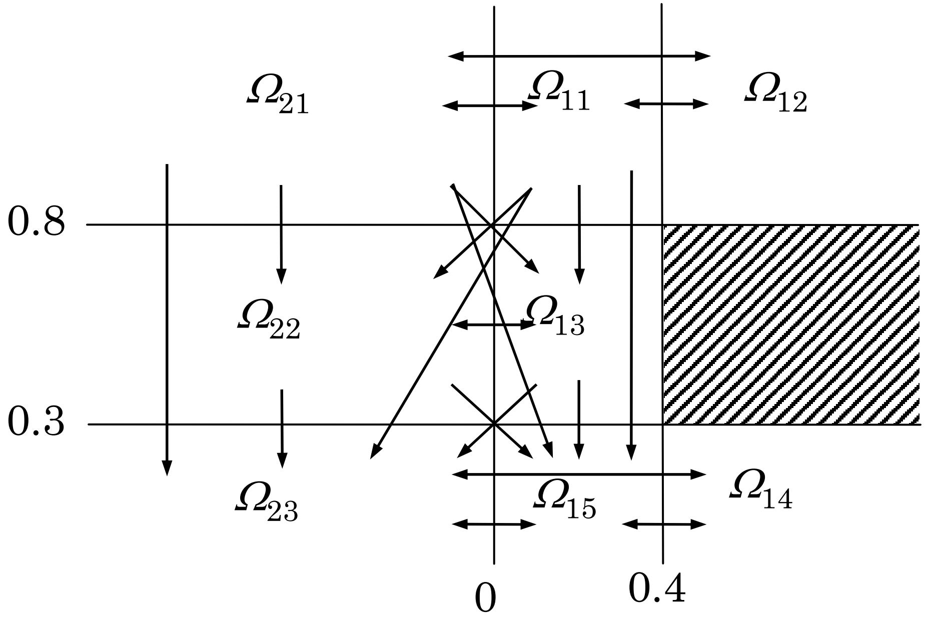
Optimal control problem

We do not allow trajectories such that



State-space partition

1. We partition the state space into subregions.
2. Define discrete transitions between the subregions in such a way that they do not violate the requirement.

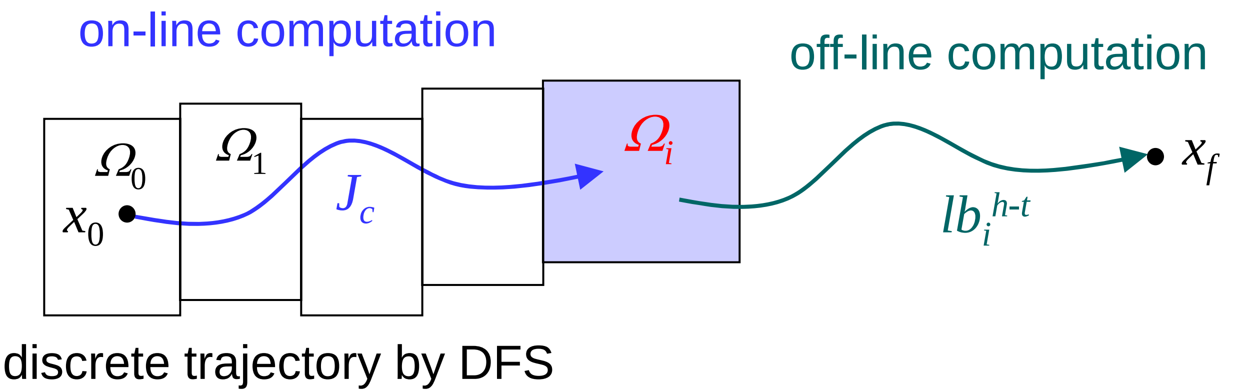


Lower-bounds of J_h

By off-line computation, we can compute the minimum value lb_i^r of J_r, $r < h$ for runs *from any point in the current region* Ω_i to the final state x_f, i.e.,

$$lb_i^r := \min_{x \in \Omega_i} J_r(x, x_f, u), r < h$$

Let J_c be the minimum J_t form x_0 to any point in Ω_i. Then $J_c + lb_i^{h\text{-}t}$ is a lower bound of J_h.

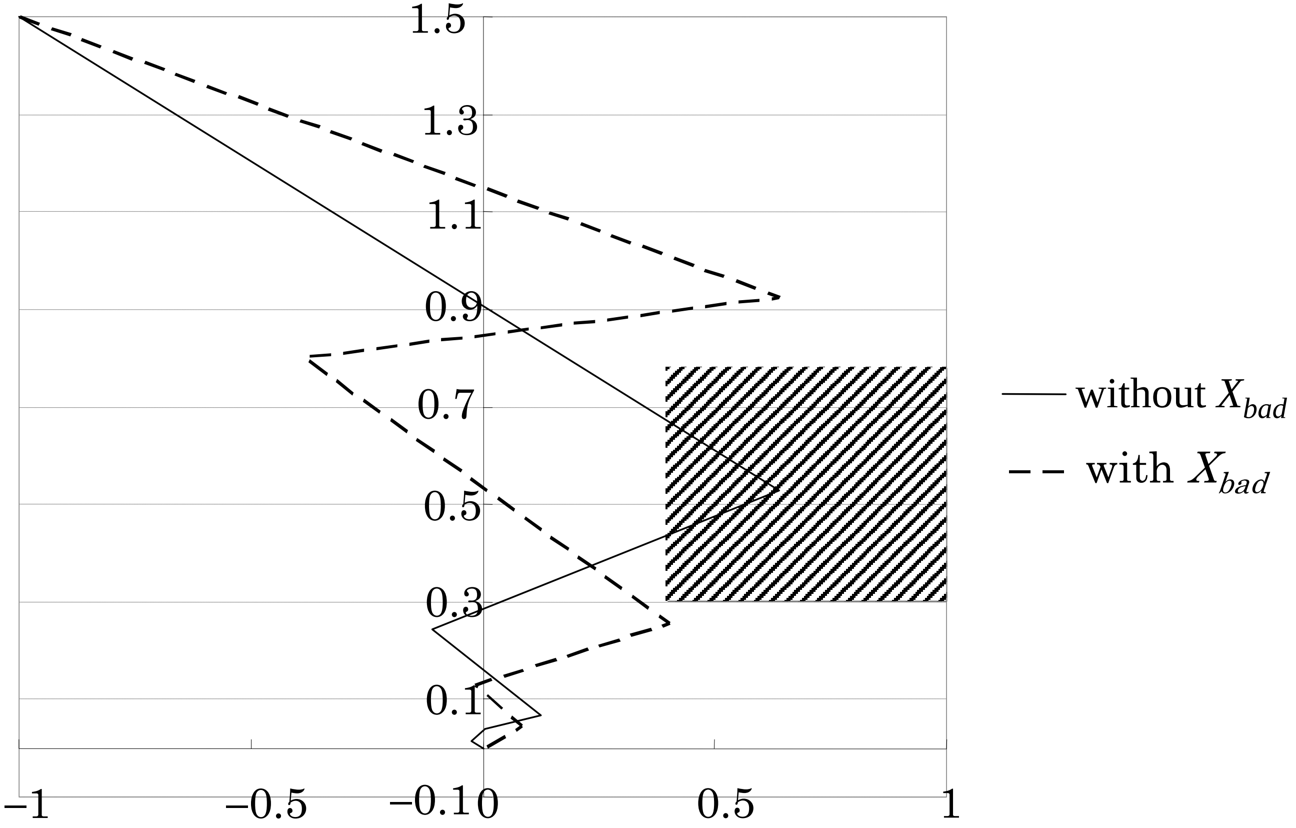


CLP code for optimization

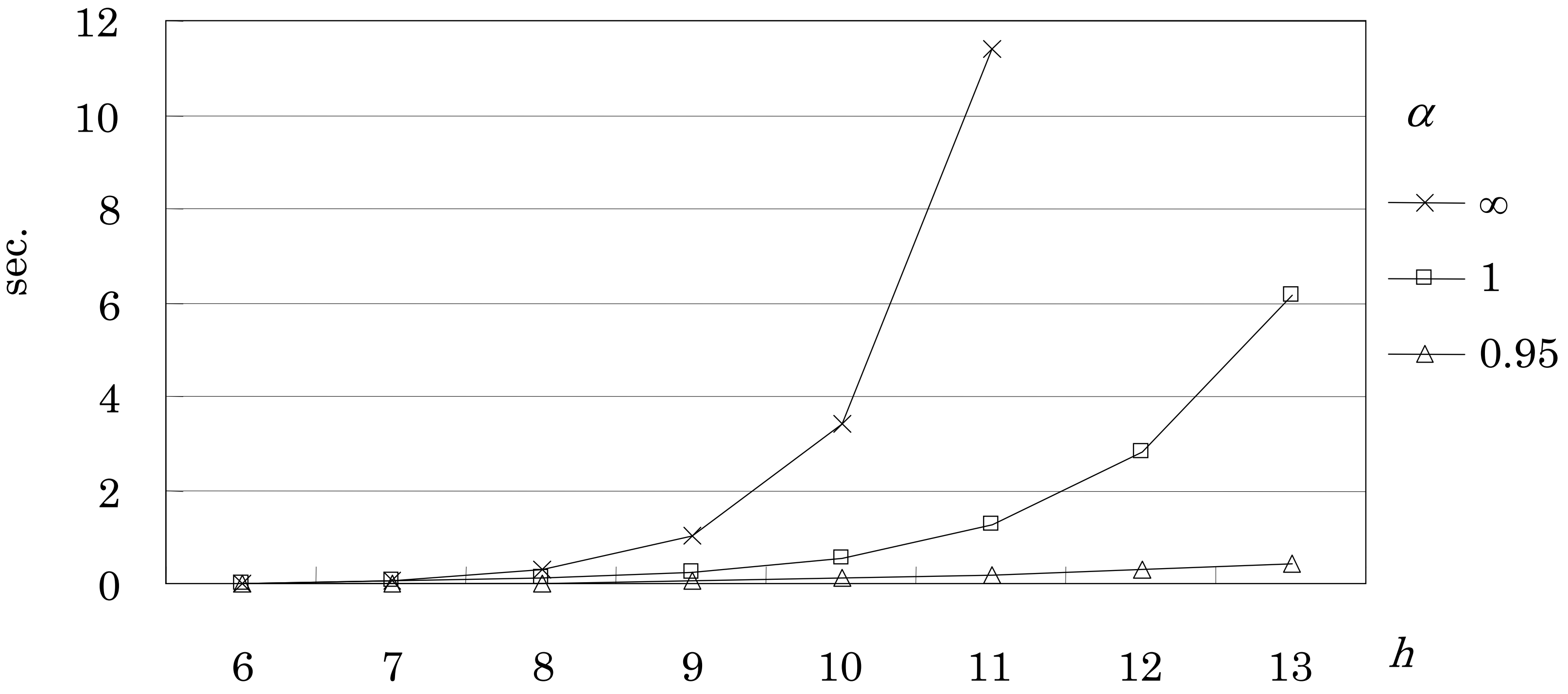
```
exec(Loc, T, W, J, H):-
    T = H, !,
    qmin_list(W, [], J), save_solution(J).
exec(Loc, T, W, J, H):-
    lb(Loc, H – T, LB), qmin_list(W, [], JC, nobind),
    JC + LB <= 0.95 * @optval,          { Lower bound $\le \alpha J_{temp}$ }
    x1(T : X1), x2(T : X2), !,
    edge(Loc, LocN),                    { get the next location LocN }
    inv(LocN, X1, X2),                             { invariant of LocN}
    update(LocN, T, W, W1),             {update the continuous state}
    exec(LocN, T + 1, W1, J, H).           { recursive call of exec() }
```

We do not consider a node if it probably generates a run with J_h close to the temporary optimal value.

Results



Results



Future work

Incorporating scenario with optimization

- In practical application, exact optimality is not necessarily required. Moreover, operators of the system have knowledge on the desirable *scenario*, i.e., how the plant should behave and should be operated. Such knowledge can be described formally by logical constraints. By incorporating the notion of scenario, we will be able to compute semi-optimal solutions more efficiently.
- As the next step of this research, we are planning to describe a scenario in the form of a temporal logic formula and compute an optimal solution among all solutions that force the system to behave like the scenario. Such procedures is implemented by KCLP-HS.