

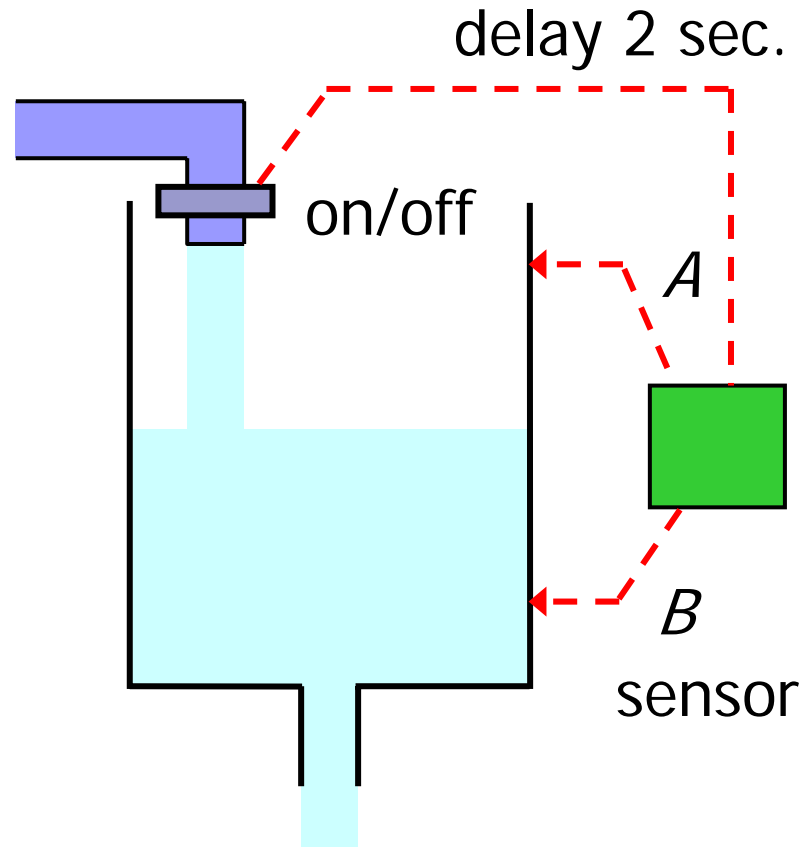
Solving Problems on Hybrid Systems by Constraint Satisfaction

Kunihiko Hiraishi

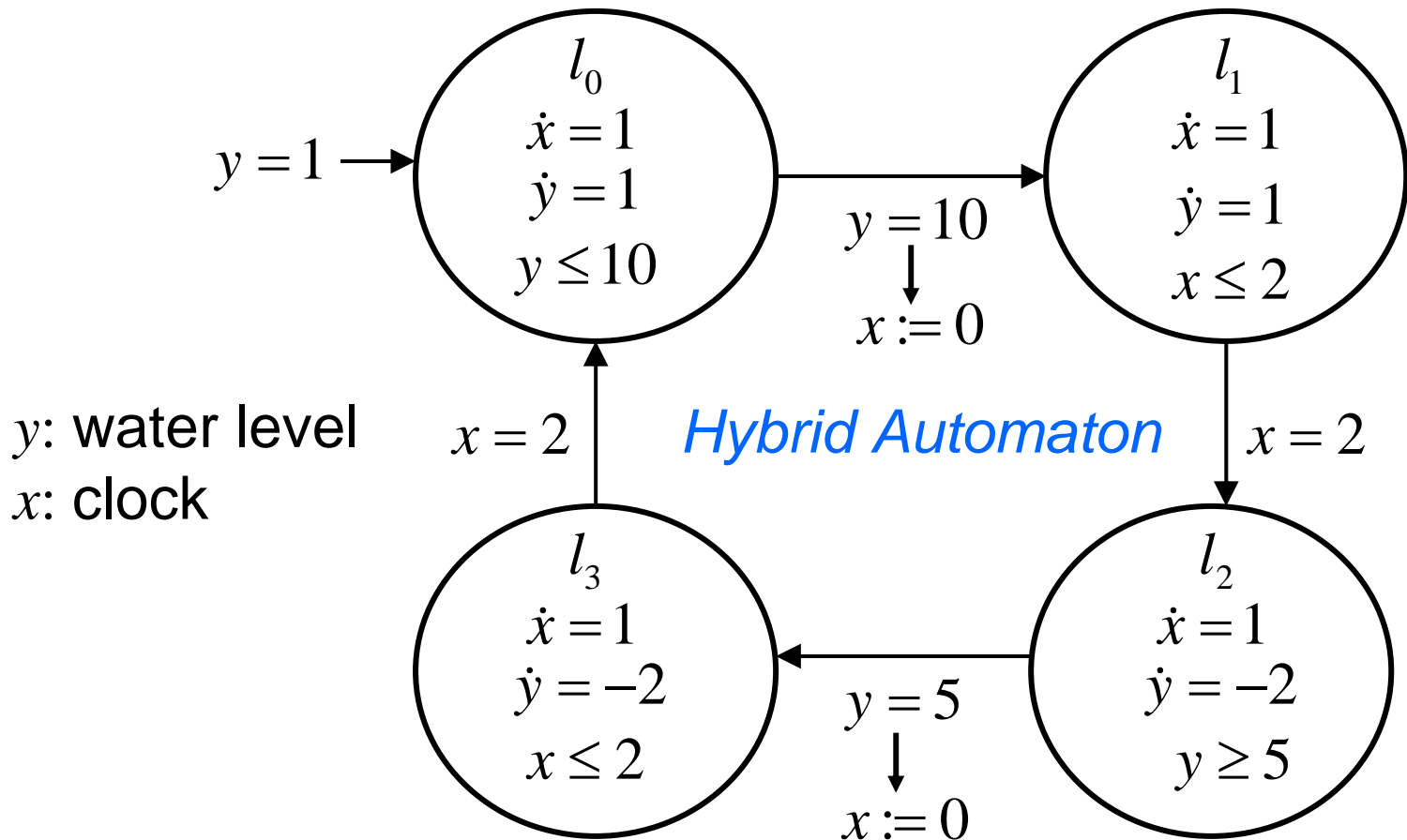
School of Information Science,

Japan Advanced Institute of Science and Technology

Example: Water-level monitor



Example: Water-level monitor



Computer tools for hybrid systems

- Real-time systems: KRONOS,UPPAAL, ...
- Hybrid systems: HyTech, CheckMate, d/dt, MLD (Mixed Logical Dynamical) systems with MIQP(Mixed Integer Quadratic Programming), ...

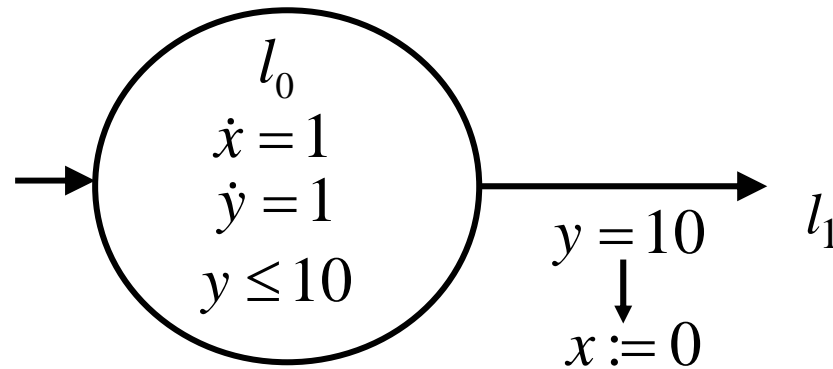
Aim and Method

- We would like to develop new computer tools for
 - optimization (including optimal control problems),
 - parameter design,
 - and also for verification and reachability analysis.
- We use symbolic computation techniques:
 - CLP (Constraint Logic Programming) → **KCLP-HS**
 - QE (Quantifier Elimination) → **SyNRAC**

KCLP-HS

- CLP = Prolog + Constraint Solver.
- Keyed CLP for HS =
 - ▶ Prolog Interpreter +
 - ▶ Linear Constraint Solver (simplex method) +
 - ▶ Linear/Quadratic Optimizer +
 - ▶ Global variables (keyed predicates) +
 - ▶ Convex Polyhedral Library (POLKA).

Modeling HA by KCLP-HS



$I_0(X, Y, \text{Time})$:-

$X_1 = X + D, Y_1 = Y + D, D \geq 0,$

$Y_1 = 10,$

$I_1(0, Y_1, \text{Time} + D).$

Modeling by CLP

```
I0(X, Y, Time):-  
    X1 = X + D, Y1 = Y + D, D >= 0,  
    Y1 = 10,  
    I1(0, Y1, Time + D).  
I1(X, Y, Time):-  
    X1 = X + D, Y1 = Y + D, D >= 0,  
    X1 = 2,  
    I2(X1, Y1, Time + D).  
I2(X, Y, Time):-  
    X1 = X + D, Y1 = Y - 2 * D, D >= 0,  
    Y1 = 5,  
    I3(0, Y1, Time + D).  
I3(X, Y, Time):-  
    X1 = X + D, Y1 = Y - 2 * D, D >= 0,  
    X1 = 2,  
    I0(X, Y, Time + D).
```


Execution trace

(trace)| ?- l0(X, 1, 0).

1-0) CALL : l0(X, 1, 0) ?

1-0) TRY : l0(X, 1, 0) :- _5 = X + _6, _8 = 1 + _6, _6 >= 0, _8 = 10, l1(0, _8, 0 + _6)

1-0) SUC : l0(X, 1, 0) :- _5 = X + _6, _8 = 1 + _6, _6 >= 0, _8 = 10, l1(0, _8, 0 + _6)

1-1) CONSTRAINT : _5 = X + _6

1-1) SUC : _5 = X + _6

1-1) CONSTRAINT : _8 = 1 + _6

1-1) SUC : _8 = 1 + _6

1-1) CONSTRAINT : _6 >= 0

1-1) SUC : _6 >= 0

1-1) CONSTRAINT : _8 = 10

1-1) SUC : 10 = 10

1-1) CALL : l1(0, 10, 0 + _6) ?

1-1) TRY : l1(0, 10, 9) :- _15 = 0 + _16, _18 = 10 + _16, _16 >= 0, _15 = 2, l2(_15, _18, 9 + _16)

1-1) SUC : l1(0, 10, 9) :- _15 = 0 + _16, _18 = 10 + _16, _16 >= 0, _15 = 2, l2(_15, _18, 9 + _16)

1-2) CONSTRAINT : _15 = 0 + _16

1-2) SUC : _15 = 0 + _16

1-2) CONSTRAINT : _18 = 10 + _16

1-2) SUC : _18 = 10 + _16

1-2) CONSTRAINT : _16 >= 0

1-2) SUC : _16 >= 0

Computation process

Goal

$I0(X, 1, 0)$

Computation process

$I0(X, 1, 0)$



unification

$I0(X, Y, Time):-$

$X1 = X + D, Y1 = Y + D, D \geq 0,$

$Y1 = 10,$

$I1(0, Y1, Time + D).$

Computation process

$I0(X, 1, 0)$



unification

$I0(X, 1, 0):-$

$X1 = X + D, Y1 = 1 + D, D \geq 0,$

$Y1 = 10,$

$I1(0, Y1, 0 + D).$

Computation process

Goal

$$X1 = X + D, Y1 = 1 + D, D \geq 0, Y1 = 10, I1(0, Y1, D)$$

Computation process

Goal

$$X1 = X + D, Y1 = 1 + D, D \geq 0, Y1 = 10, I1(0, Y1, D)$$

Computation process

$I1(0, Y1, D)$

Constraints

$$\begin{aligned} X1 &= X + D, \\ Y1 &= 1 + D, \\ D &\geq 0, \\ Y1 &= 10 \end{aligned}$$

Computation process

$I1(0, Y1, D)$

Constraints

$X1 = X + D,$
 $Y1 = 1 + D,$
 $D \geq 0,$
 $Y1 = 10$

Constraint Solver



TRUE

Computation process

Goal

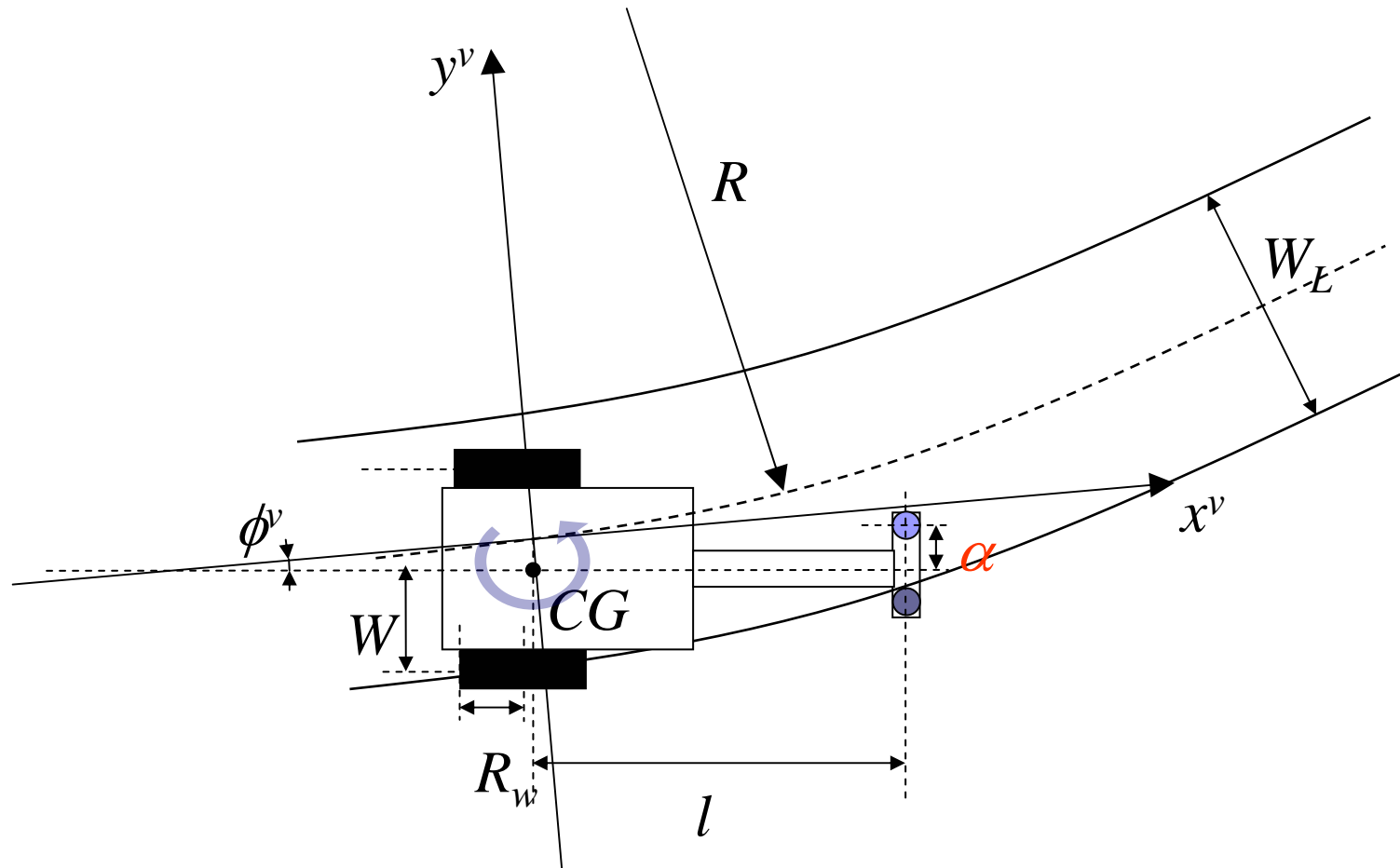
$I1(0, 10, D)$

with

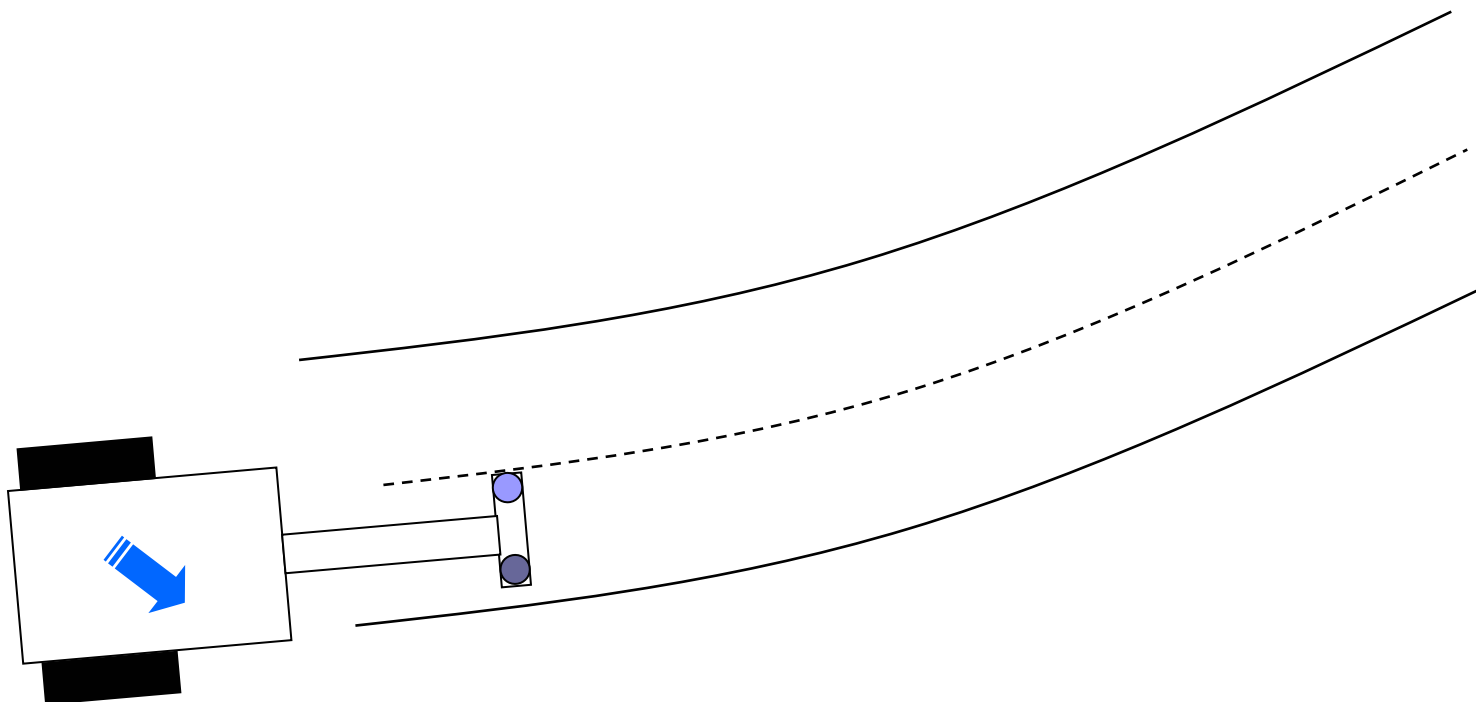
Constraints

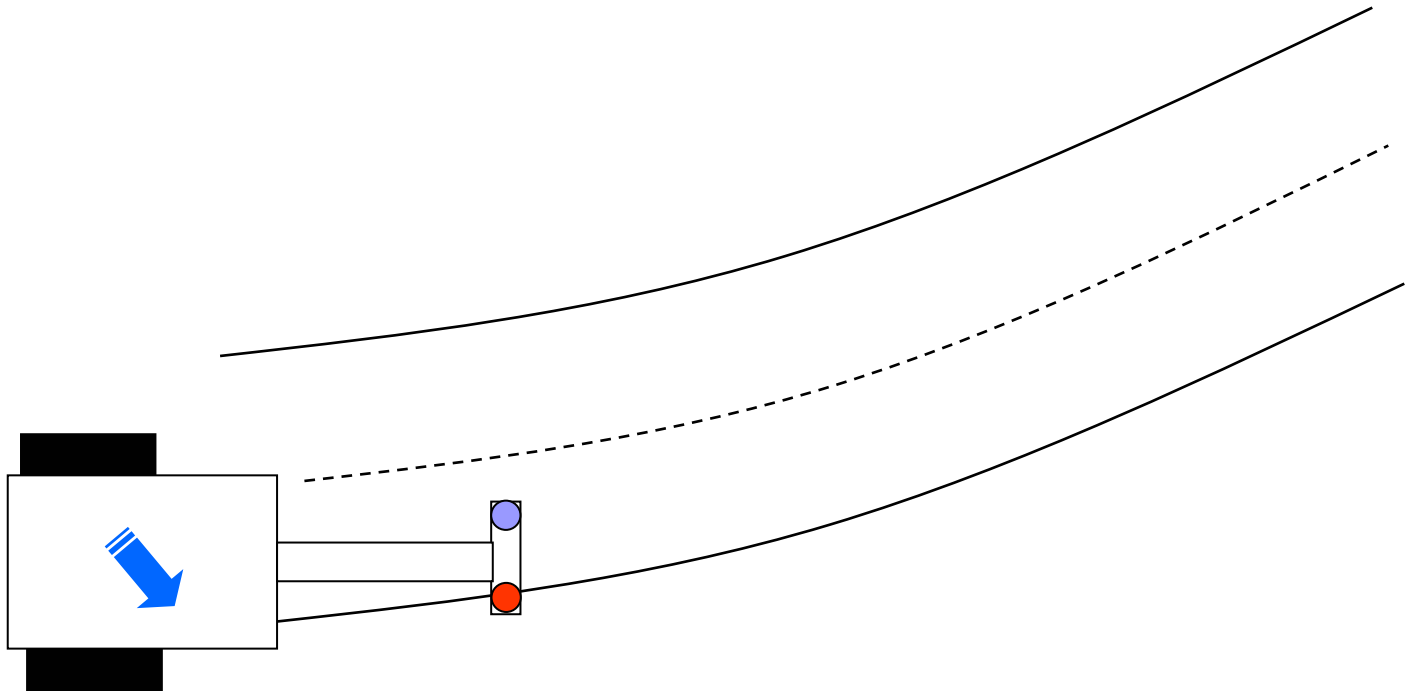
$X1 = X + D, Y1 = 1 + D, D \geq 0$

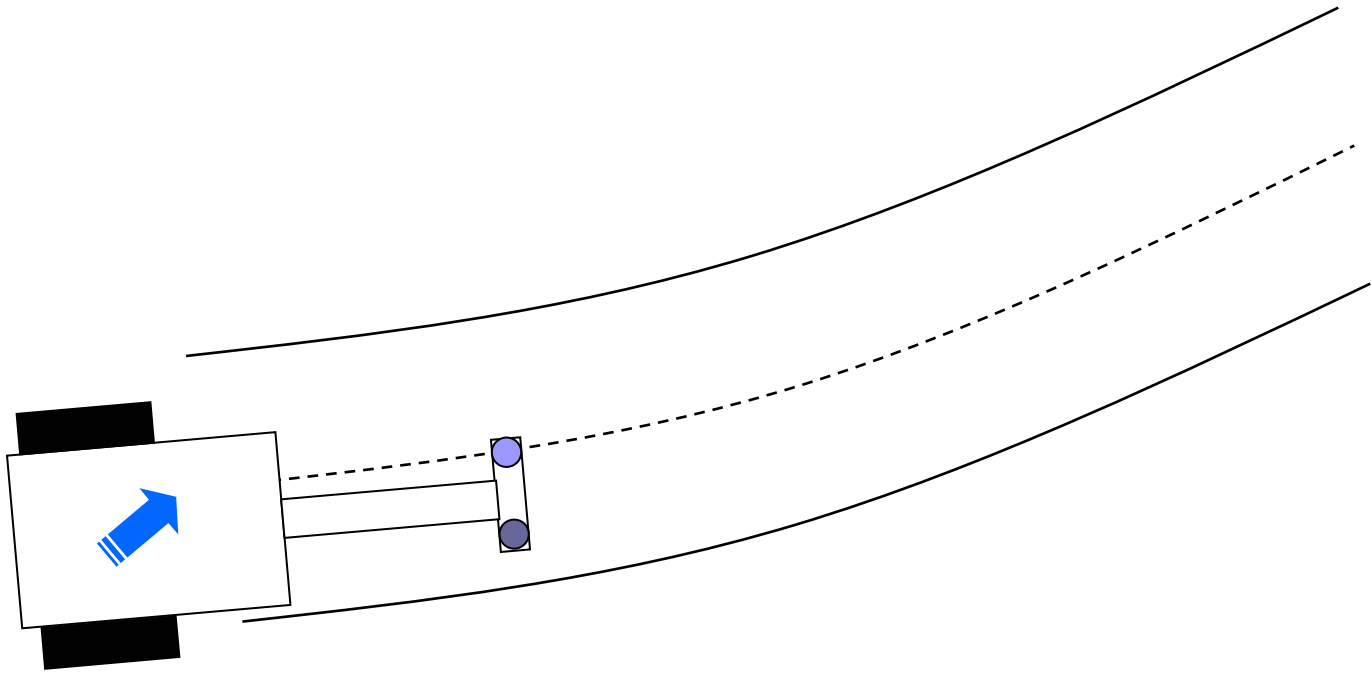
Example: Two-wheeled vehicle

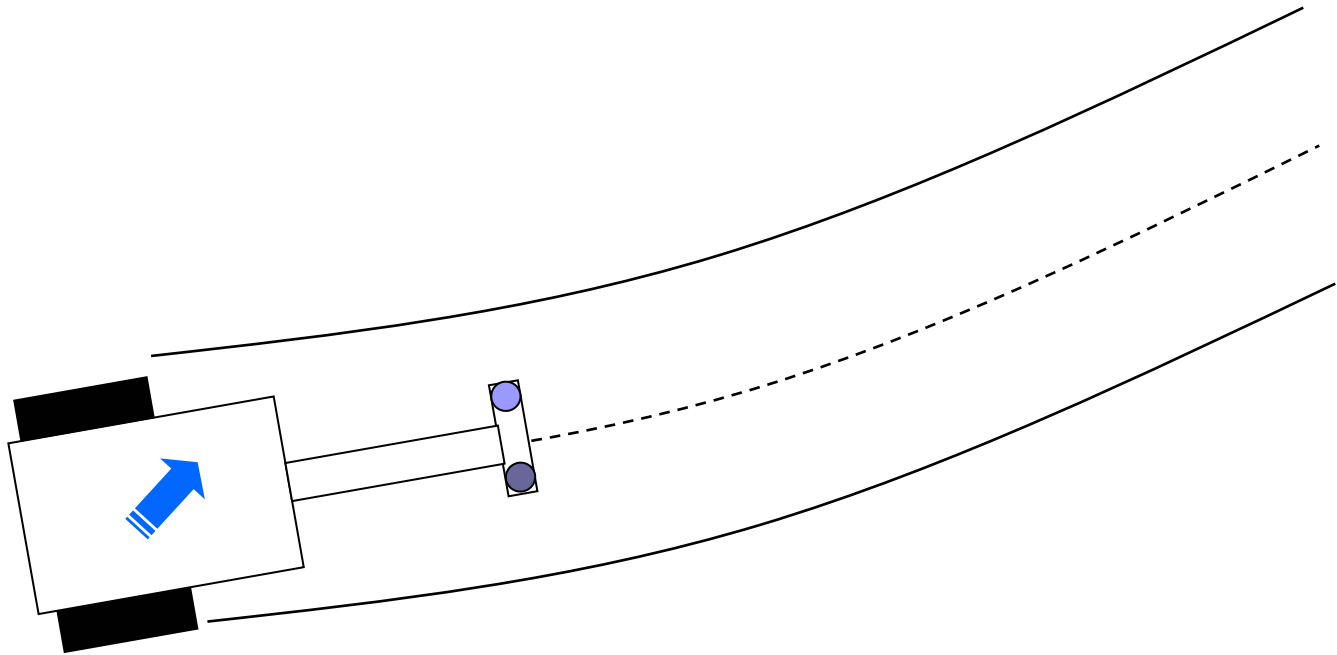


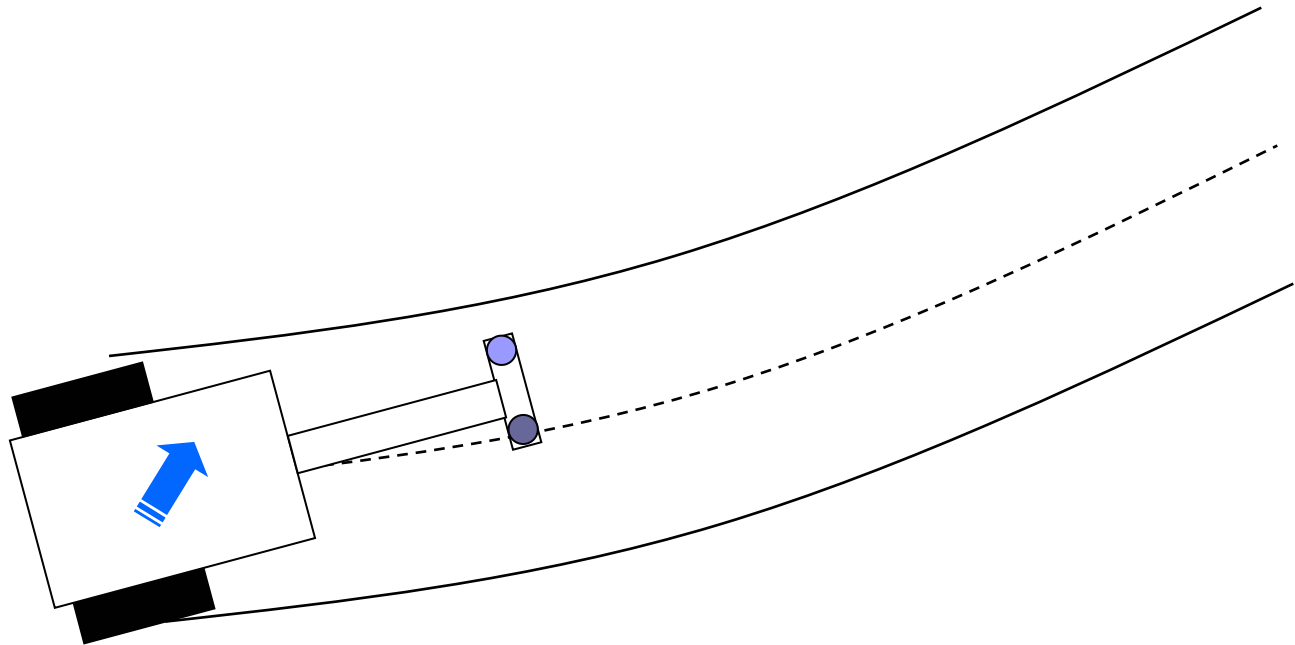
[Konaka 2004]

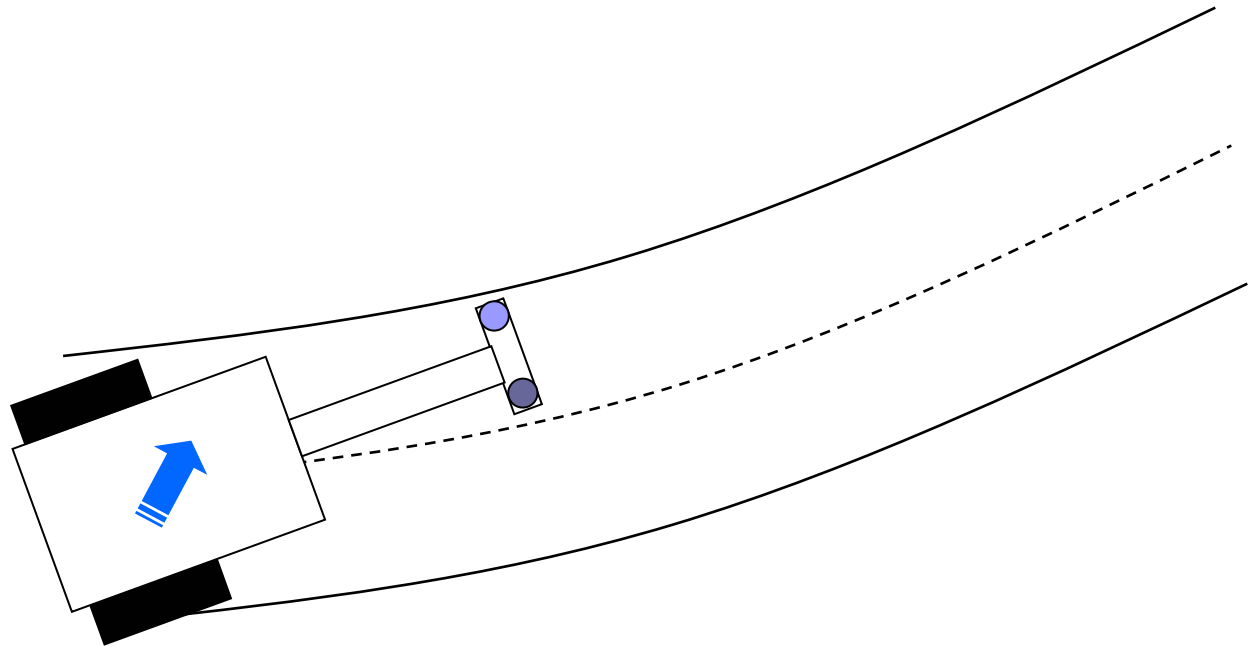


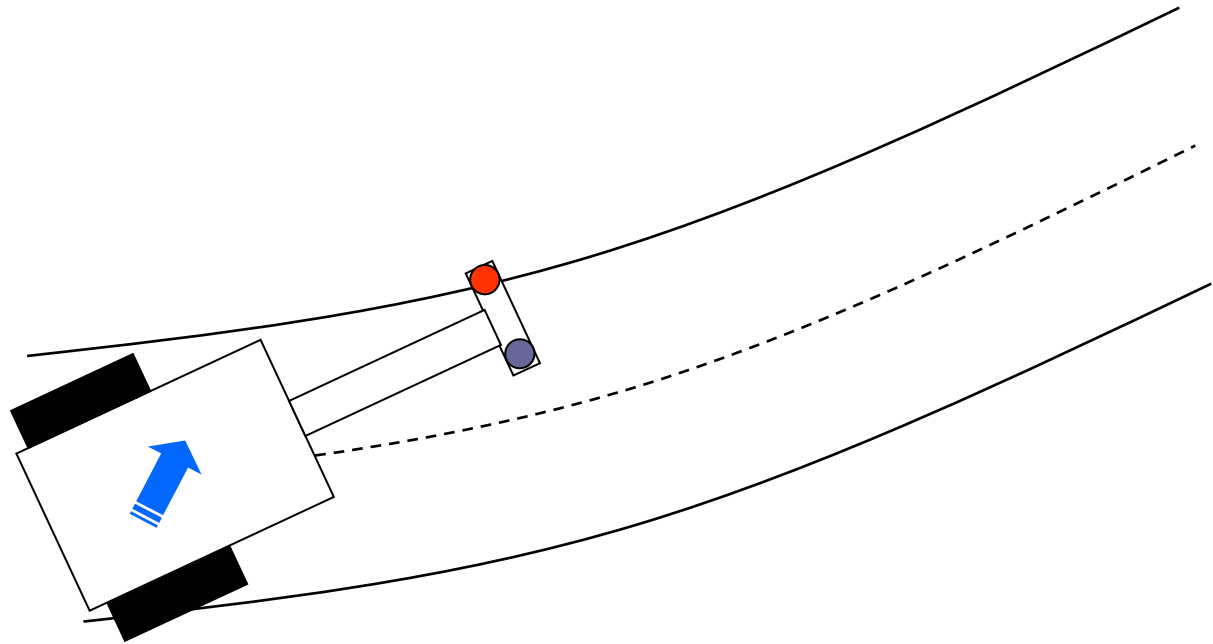


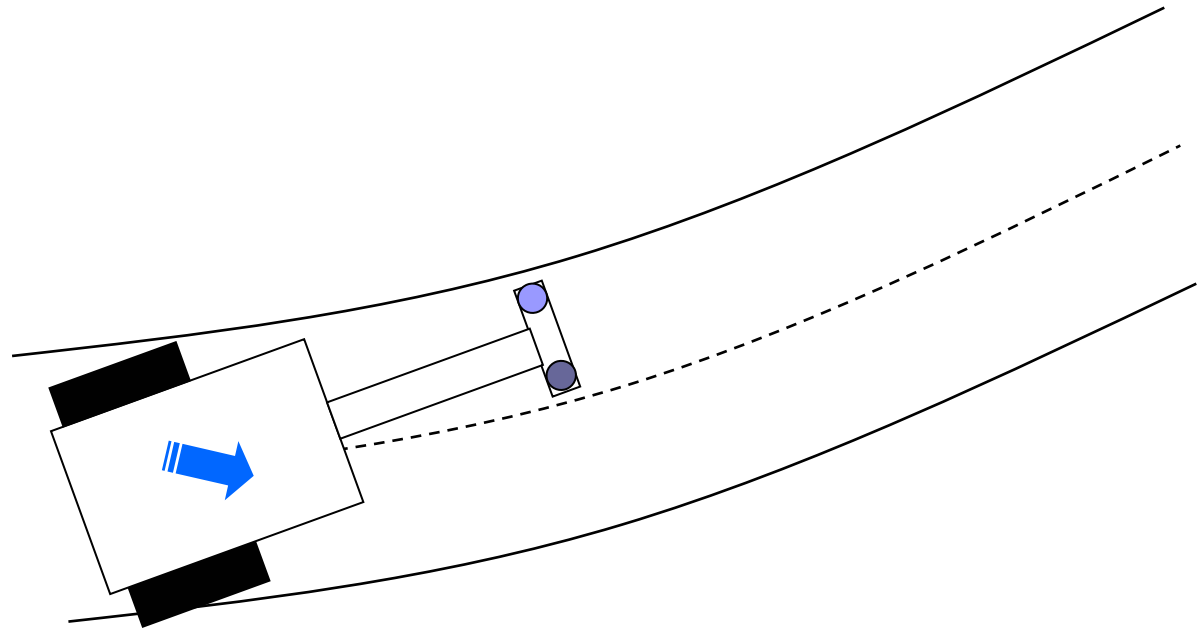


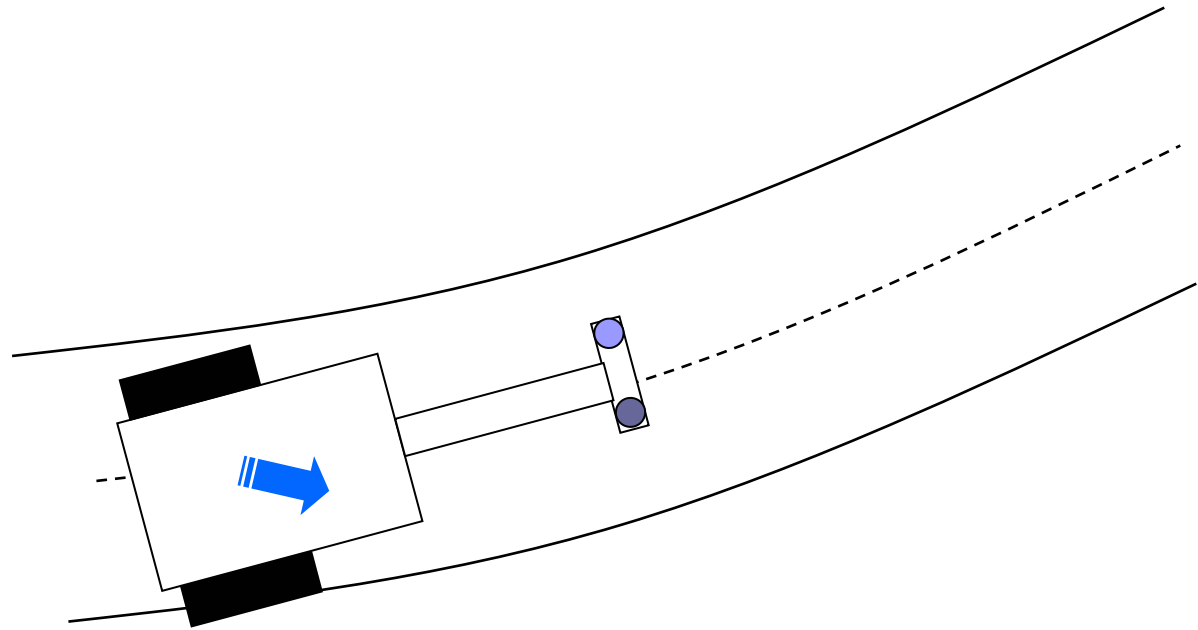


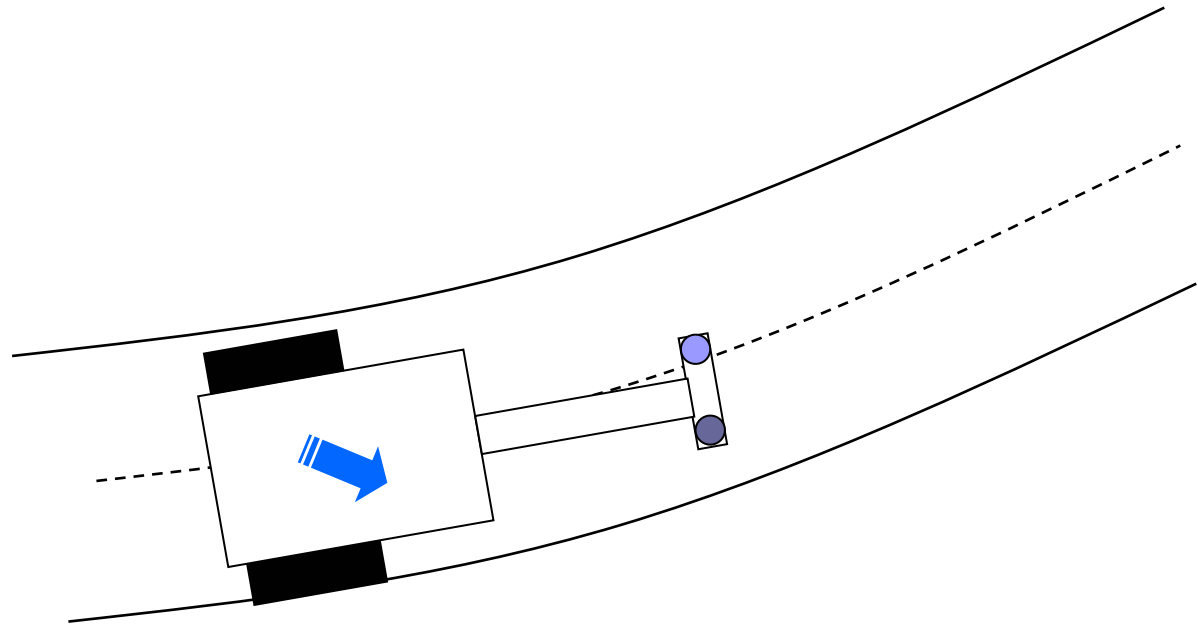


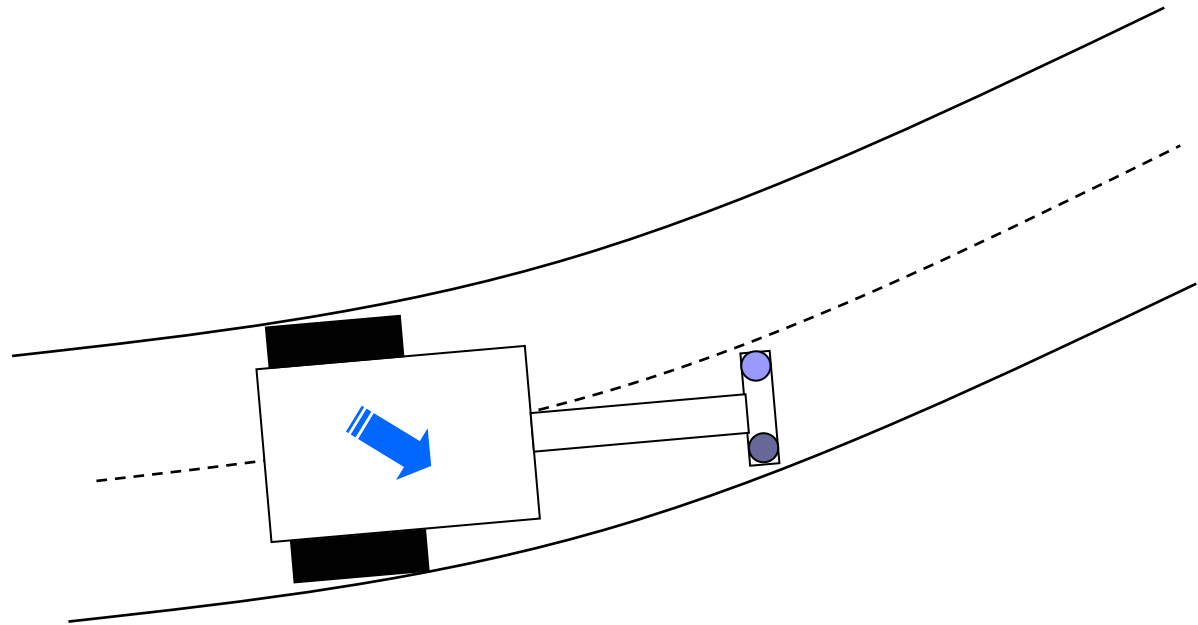


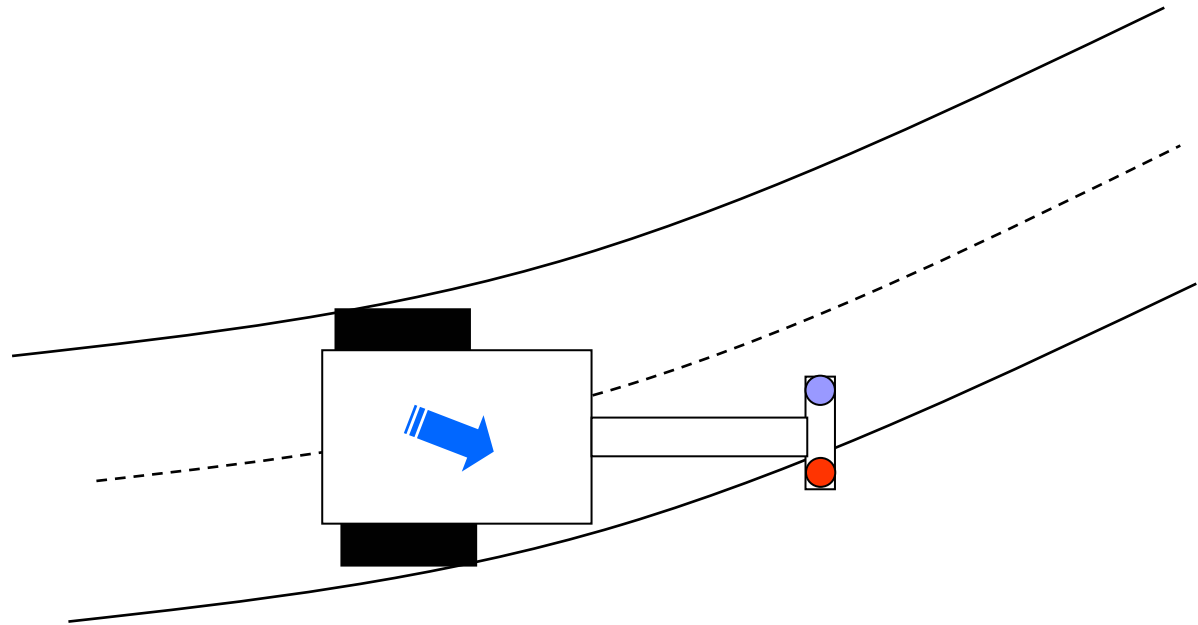


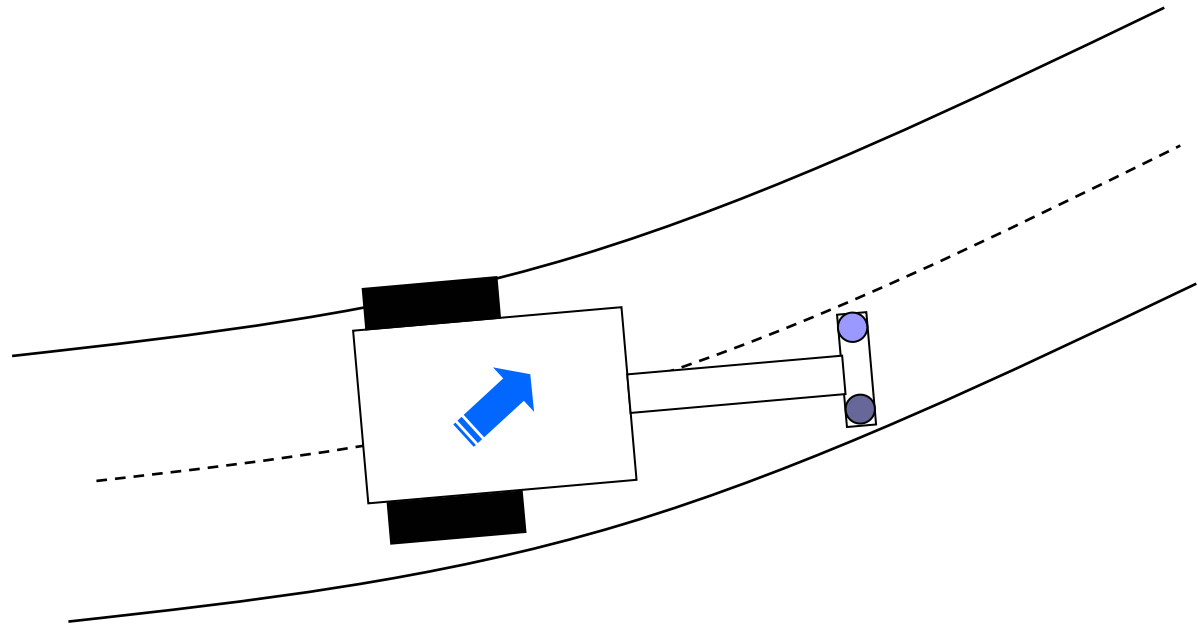




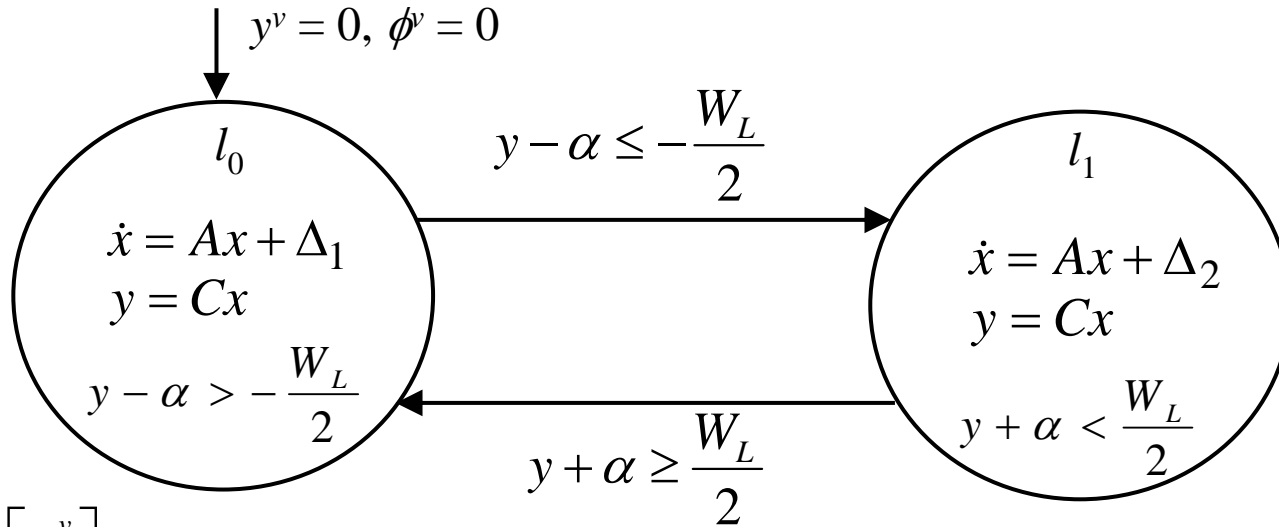








Hybrid automaton



$$x = \begin{bmatrix} y^v \\ \phi^v \end{bmatrix}$$

$$A = \begin{pmatrix} 0 & a_1 \\ 0 & 0 \end{pmatrix}, \Delta_1 = \begin{pmatrix} 0 \\ a_2 \end{pmatrix}, \Delta_2 = \begin{pmatrix} 0 \\ a_3 \end{pmatrix}$$

$$C = (1 \quad l)$$

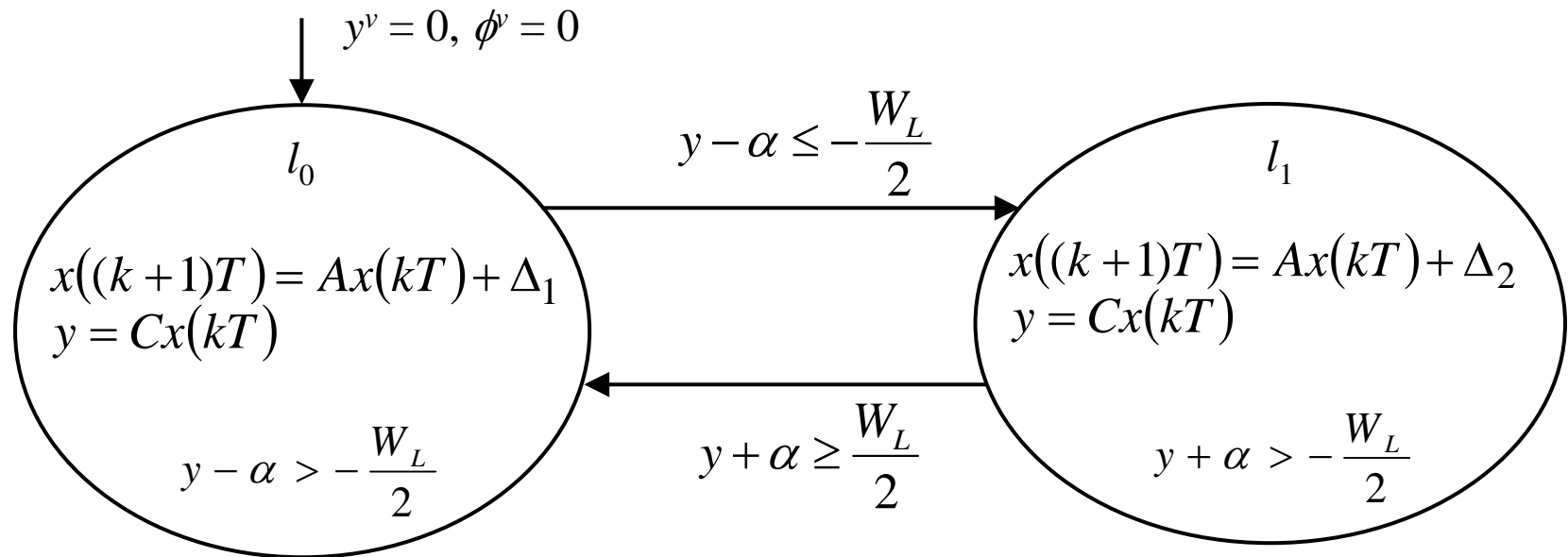
$$a_1 = \frac{R_w}{2} (w_H + w_L)$$

$$a_2 = -\frac{R_w}{2} \left(\frac{w_H + w_L}{R} + \frac{w_H - w_L}{W} \right)$$

$$a_3 = -\frac{R_w}{2} \left(\frac{w_H + w_L}{R} - \frac{w_H - w_L}{W} \right)$$

Continuous Time (differential equations)

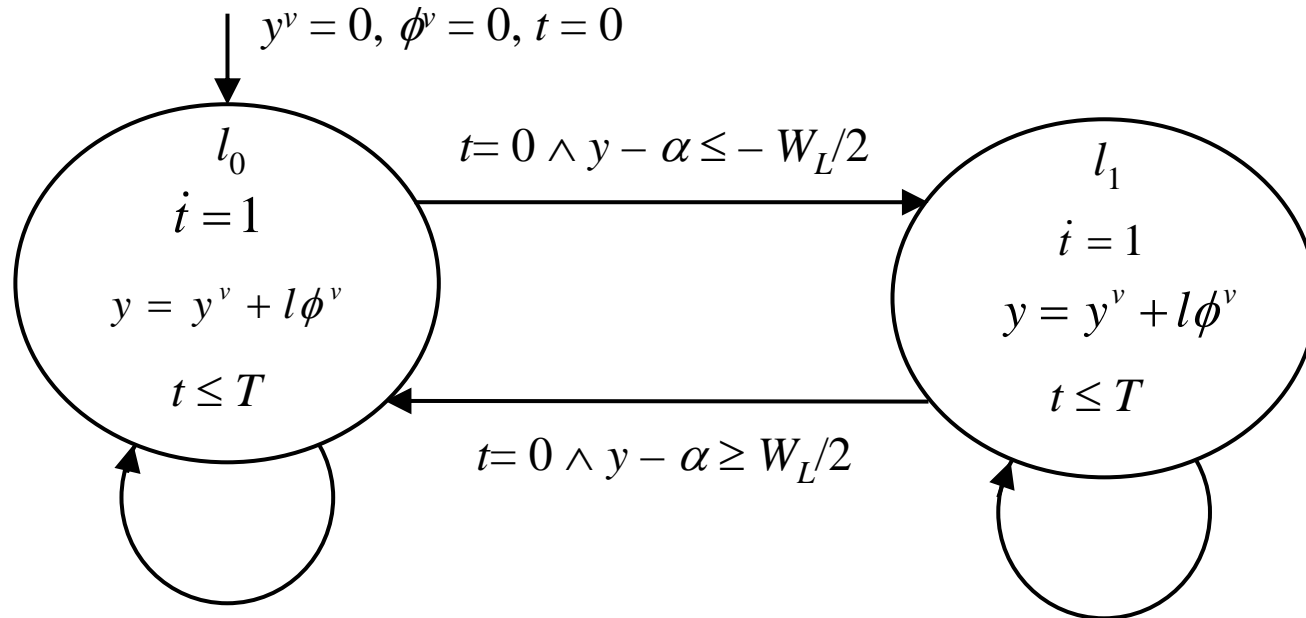
Hybrid automaton



$$x = \begin{bmatrix} y^v \\ \phi^v \end{bmatrix}, A = \begin{pmatrix} 1 & a_1 T \\ 0 & 1 \end{pmatrix}, C = (1 \quad l), \Delta_1 = \begin{pmatrix} -\frac{a_1 a_2}{2} T^2 \\ a_2 T \end{pmatrix}, \Delta_2 = \begin{pmatrix} -\frac{a_1 a_3}{2} T^2 \\ a_3 T \end{pmatrix}$$

Discrete Time (difference equations)

Hybrid automaton



$$\begin{aligned}
 &t = T \wedge y - \alpha > -W_L/2 \rightarrow \\
 &y^v := y^v + a_1 T \phi^v - (a_1 a_2 T^2)/2, \\
 &\phi^v := \phi^v + a_2 T, \\
 &t := 0
 \end{aligned}$$

$$\begin{aligned}
 &t = T \wedge y - \alpha < W_L/2 \rightarrow \\
 &y^v := y^v + a_1 T \phi^v - (a_1 a_3 T^2)/2, \\
 &\phi^v := \phi^v + a_3 T, \\
 &t := 0
 \end{aligned}$$

HA with one clock variable

Parameter design problem

Find the value of α that minimizes

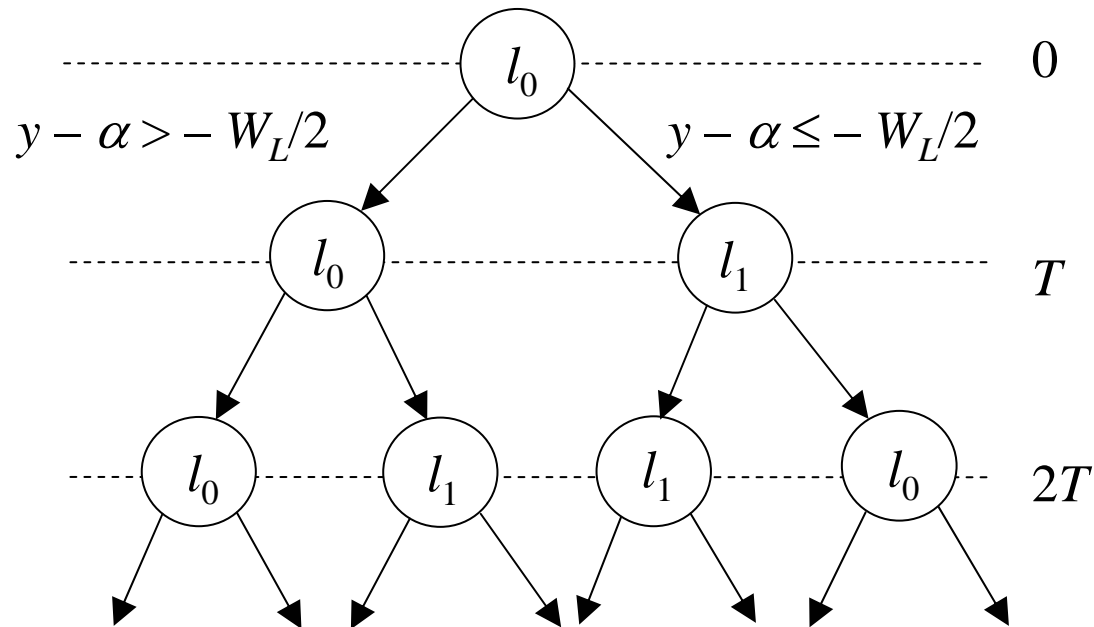
$$J = \sum_{k=1}^{k_f} \left(x(k)^T x(k) + \lambda |l(k) - l(k-1)| \right) \rightarrow \textit{Minimize}$$

(Quadratic objective function)

CLP code

```
I1(X1, X2, T, J, J):- FinalTime(: FT), T >= FT, !.  
I1(X1, X2, T, J, J1):- (optval, !, J <= @optval; true),  
    Alpha(: Alpha), X1 + 0.215 * X2 - Alpha > - 0.05,  
    I1(X1 + 0.022765 * X2 + 0.000781, X2 - 0.06864,  
        T + 0.1, J + X1 * X1 + X2 * X2, J1).  
I1(X1, X2, T, J, J1):- (optval, !, J <= @optval; true),  
    Alpha(: Alpha), X1 + 0.215 * X2 - Alpha <= - 0.05,  
    Lambda(: L), I2(X1, X2, T, J + L, J1).  
I2(X1, X2, T, J, J):- FinalTime(: FT), T >= FT, !.  
I2(X1, X2, T, J, J1):- (optval, !, J <= @optval; true),  
    Alpha(: Alpha), X1 + 0.215 * X2 + Alpha < 0.05,  
    I2(X1 + 0.022765 * X2 - 0.00061, X2 + 0.053464,  
        T + 0.1, J + X1 * X1 + X2 * X2, J1).  
I2(X1, X2, T, J, J1):- (optval, !, J <= @optval; true),  
    Alpha(: Alpha), X1 + 0.215 * X2 + Alpha >= 0.05,  
    Lambda(: L), I1(X1, X2, T, J + L, J1).
```

Search strategy



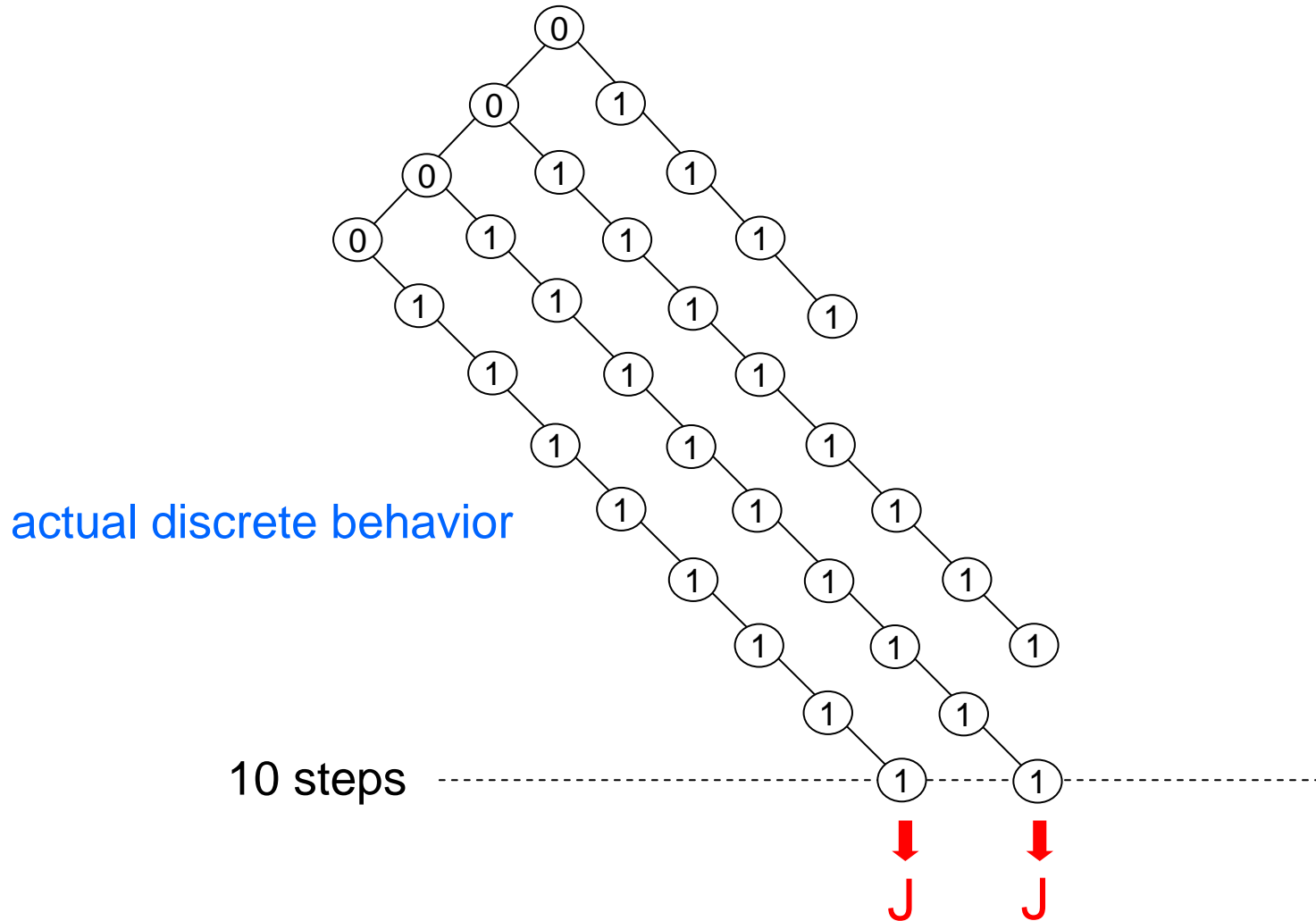
go(F, L, A, Z, J):-

FinalTime(: F), Lambda(: L), Alpha(: A),

min(J, l1(0, 0, 0, 0, J)),

fourier([A], Z). ← projecting constraints to those on variable A

Search strategy



Comparing with tentative optimal value

```
I1(X1, X2, T, J, J1):- (optval, !, J <= @optval; true),  
    Alpha(: Alpha), X1 + 0.215 * X2 - Alpha > - 0.05,  
    I1(X1 + 0.022765 * X2 + 0.000781, X2 - 0.06864,  
    T + 0.1, J + X1 * X1 + X2 * X2, J1).
```

If the tentative optimal value is obtained,
then compare it with the current value of J .

Operations on convex polyhedra

polka_constraints2poly()

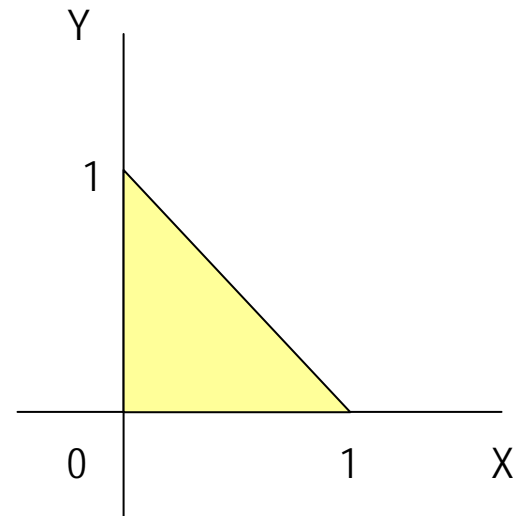


`[[X, Y], [X + Y <= 1, X >= 0, Y >= 0]]`

CLP constraints



polka_poly2constraints()



convex polyhedron



POLKA Library

(<http://www.irisa.fr/prive/Bertrand.Jeannet/newpolka.html>)
supports intersection, emptiness test, change of
dimension, convex hull etc of convex polyhedra.

CLP code using POLKA

```
l1( _, _, _, P, _, _, _):- polka_poly_is_empty(P), !, fail.
l1(T, _, _, P, P, J, J):- FinalTime(: FT), T >= FT, !.
l1(T, X1, X2, P, P1, J, J1):- (optval, !, entailed(J <= @optval); true),
    polka_constraints2poly([[A], [X1 + 0.215 * X2 - A > - 0.05]], PN),
    polka_poly_intersection(P, PN, PNN),
    l1(T + 0.1, X1 + 0.022765 * X2 + 0.000781, X2 - 0.06864,
        PNN, P1, J + X1 * X1 + X2 * X2, J1).
l1(T, X1, X2, P, P1, J, J1):- (optval, !, entailed(J <= @optval); true),
    polka_constraints2poly([[A], [X1 + 0.215 * X2 - A <= - 0.05]], PN),
    polka_poly_intersection(P, PN, PNN),
    Lambda(: L), l2(T, X1, X2, PNN, P1, J + L, J1).
l2( _, _, _, P, _, _, _):- polka_poly_is_empty(P), !, fail.
l2(T, _, _, P, P, J, J):- FinalTime(: FT), T >= FT, !.
l2(T, X1, X2, P, P1, J, J1):- (optval, !, entailed(J <= @optval); true),
    polka_constraints2poly([[A], [X1 + 0.215 * X2 + A < 0.05]], PN),
    polka_poly_intersection(P, PN, PNN),
    l2(T + 0.1, X1 + 0.022765 * X2 - 0.00061, X2 + 0.053464,
        PNN, P1, J + X1 * X1 + X2 * X2, J1).
l2(T, X1, X2, P, P1, J, J1):- (optval, !, entailed(J <= @optval); true),
    polka_constraints2poly([[A], [X1 + 0.215 * X2 + A >= 0.05]], PN),
    polka_poly_intersection(P, PN, PNN),
    Lambda(: L), l1(T, X1, X2, PNN, P1, J + L, J1).
```

Example: optimal control problem

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

$$y(t) = [1 \quad 0]x(t)$$

$$\alpha(t) = \begin{cases} \pi/3 & ([1 \quad 0]x(t) \geq 0) \\ -\pi/3 & ([1 \quad 0]x(t) < 0) \end{cases} \quad \text{SWITCH}$$

$$x(t) \in [-10 \quad 10] \times [-10 \quad 10]$$

$$u(t) \in [-1 \quad 1] \quad \leftarrow \text{control input}$$

$$J = \sum_{t=0}^{T-1} \|u(t) - u_f\|_{Q_1}^2 + \|x(t) - x_f\|_{Q_2}^2 \rightarrow \text{Minimize}$$

[Bemporad and Morari 1999]

CLP code

`l1(T, L, J):- FinalTime(: F), T > F, !, qmin_list(L, [], J), ← quadratic optimizer
rec(0, RES), record(0, [J, RES]).`

`l1(T, L, J):- X1(T : X1), X1 < 0, l2(T, L, J).` ↓ checking the current value of J

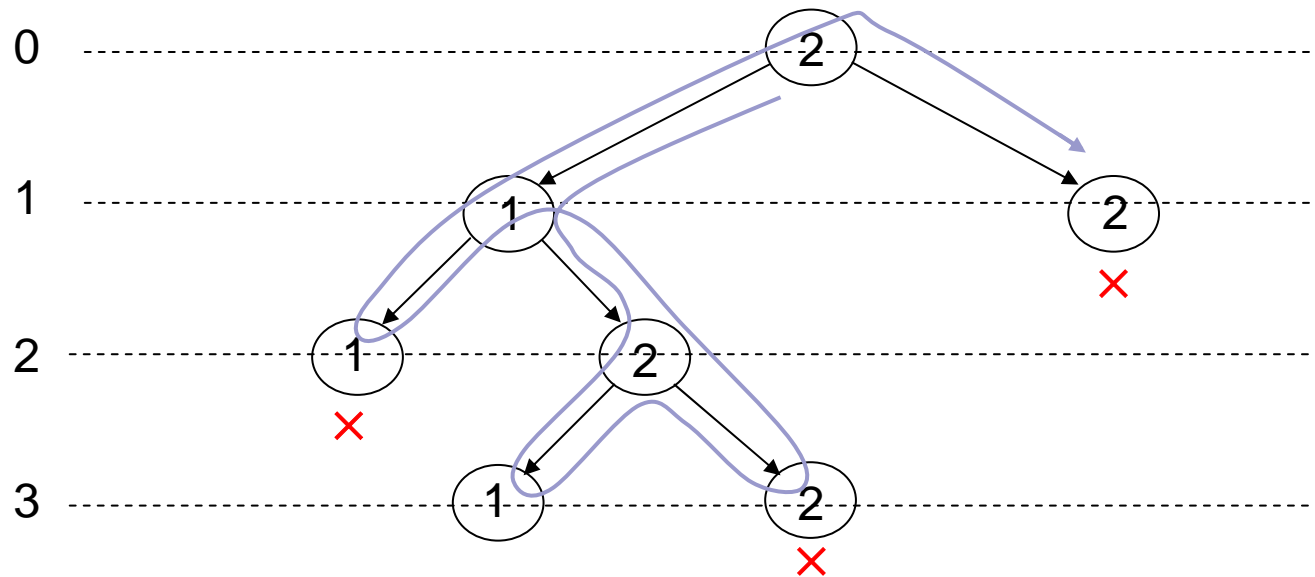
`l1(T, L, J):- (optval, !, qmin_list(L, [], JTEMP, _), JTEMP <= @optval; true),
X1(T : X1), X1 >= 0, X2(T : X2), U(T : U), LOC(T : 1),
A11(1 : A11), A12(1 : A12), A21(1 : A21), A22(1 : A22),
X1(T + 1 : A11 * X1 + A12 * X2),
X2(T + 1 : A21 * X1 + A22 * X2 + U), QU(: QU), UU = U * QU,
l1(T + 1, [X1, X2, UU | L], J).`

`l2(T, L, J):- FinalTime(: F), T > F, !, qmin_list(L, [], J),
rec(0, RES), record(0, [J, RES]).`

`l2(T, L, J):- X1(T : X1), X1 >= 0, l1(T, L, J).`

`l2(T, L, J):- (optval, !, qmin_list(L, [], JTEMP, _), JTEMP <= @optval; true),
X1(T : X1), X1 < 0, X2(T : X2), U(T : U), LOC(T : 2),
A11(2 : A11), A12(2 : A12), A21(2 : A21), A22(2 : A22),
X1(T + 1 : A11 * X1 + A12 * X2),
X2(T + 1 : A21 * X1 + A22 * X2 + U), QU(: QU), UU = U * QU,
l2(T + 1, [X1, X2, UU | L], J).`

Search tree by KCLP-HS



Execution result

| ?- go(10).

JOPT = 2.837877

Time = 0: Loc = 2, X1 = -1.000000, X2 = 1.000000, U = -0.672764

Time = 1: Loc = 1, X1 = 0.292820, X2 = 0.420056, U = -0.220506

Time = 2: Loc = 2, X1 = -0.173895, X2 = 0.150389, U = -0.112616

Time = 3: Loc = 1, X1 = 0.034634, X2 = 0.068017, U = -0.029554

Time = 4: Loc = 2, X1 = -0.033270, X2 = 0.021649, U = -0.020139

Time = 5: Loc = 1, X1 = 0.001690, X2 = 0.011570, U = -0.003290

Time = 6: Loc = 2, X1 = -0.007340, X2 = 0.002509, U = -0.003876

Time = 7: Loc = 2, X1 = -0.001198, X2 = 0.002213, U = -0.000943

Time = 8: Loc = 1, X1 = 0.001054, X2 = 0.000772, U = -0.000974

Time = 9: Loc = 2, X1 = -0.000113, X2 = 0.000065, U = -0.000104

Time = 10: Loc = 1, X1 = 0.000000, X2 = 0.000000, U = 0.000000

*** yes ***

0.0300 sec.

| ?-

CPU Time (sec.) for $\lambda = 0.25$.

	10	20	30
#Steps			
MLD+MIQP	0.3	16	955.2
KCLP-HS	< 0.01	0.01	0.01
SyNRAC	0.36	1.41	2.77

[Konaka 2004]

CPU Time (sec.) for $\lambda = 0.5$.

	10	20	30
#Steps			
MLD+MIQP	0.34	4.25	53.98
KCLP-HS	< 0.01	< 0.01	0.01
SyNRAC	0.34	1.36	2.70

[Konaka 2004]

Conclusion

- We have shown several examples of design and verification problems on hybrid systems, and demonstrate how to solve them by using KCLP-HS and SyNRAC.
- The tools can be used not only for verification, but also for design and optimization. This is one of advantages of them over existing verification tools.
- There are instances of problems (particularly parameter design problems) in which computation by the tools is much faster than existing approaches such as mixed integer quadratic programming based on MLD system representation.