Constraint-based Modeling and Symbolic Simulation of Hybrid Systems with HydLa and HyLaGI

*Yunosuke Yamada, Masashi Sato, Kazunori Ueda
Waseda University

CyPhy2019 Oct. 18

Talk Outline

We have developed:

- a *declarative* modeling language **HydLa**
  - that makes full use of the notion of ***constraints***
- its simulator **HyLaGI** based on ***symbolic*** computation
  - that opens up various applications

The talk will introduce

- design and features of HydLa and HyLaGI
- applications of simulation with symbolic parameters

Outline

Constraint Programming

- Users declare *constraints* (logical formulas with (in)equations) on variables
- Constraint solvers check consistency and compute *solutions* (= explicit form)

```
   5x+2y+3z == 19
&& x >= y >= 0
&& z >= 0
```

```
{{x->1, y->1, z->4},
 {x->2, y->0, z->3},
 {x->3, y->2, z->0}}
```

Key features:

- clear declarative meaning
- computing with partial information

Constraint Programming for Hybrid Systems

- Modeling of hybrid systems always involves constraints (i.e., differential equations)
- But, existing modeling languages come also with notions other than constraints.

Research question

Can we design a *declarative* hybrid system modeling language and system which is based *solely* on constraints and constraint solving?

- (cf. Modelica [1], Zélus [2], ...)

[1]Peter, F.: Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach. John Wiley & Sons (2014)
[2]Timothy, B., Marc, P.: Zélus: A synchronous language with ODEs. In: HSCC'13, 113–118 (2013)

A Constraint-based Language HydLa [3][4]

- Modeling hybrid systems
- Declaring constraints (temporal logic formulas)

Bouncing particle

```
INIT   <=> 7 < y < 12 & y' = 0.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y = 0 => y' = -4/5 * y'-).

INIT, (FALL << BOUNCE).
```

Module definition

Module declaration

[3] Ueda, K., Matsumoto, S., Takeguchi, A., Hosobe, H., Ishii, D.: HydLa: A High-Level Language for Hybrid Systems. In: LfSA2012, 3-17 (2012).
[4] Matsumoto, S.: Validated Simulation of Parametric Hybrid Systems Based on Constraints. Ph.D. thesis, Waseda University (2017)

A Constraint-based Language HydLa

Bouncing particle

```
INIT    <=> 7 < y < 12 & y' = 0.
FALL    <=> [](y'' = -10).
BOUNCE  <=> [](y = 0 => y' = -4/5 * y'-).

INIT, (FALL << BOUNCE).
```

Time derivative

e.g. **y'** is $\frac{d}{dt}y(t)$

Guard
("ask")

Left limit value

e.g. **y'-** is $\lim_{s \to t-0} y'(s)$

□-operator

e.g. **FALL** and **BOUNCE** are considered at all times; **INIT** is considered only when t = 0

Module priority

e.g. **FALL** is weaker than **BOUNCE**
What does "weak" mean?

Case 1 : When the particle is above the floor

After time 0, **INIT** and **BOUNCE** are ineffective constraints.

```
INIT   <=> 7 < y < 12 & y' = 0.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y = 0 => y' = -4/5 * y'-).
INIT, (FALL << BOUNCE).
```

Not with the □-operator

False

- At any time, HydLa adopts a maximal consistent set (MCS) of modules.
- In this case, the MCS is {**INIT, FALL, BOUNCE**}.
- The particle keeps falling.

Case 2 : When the particle collides the floor

```
INIT   <=> 7 < y < 12 & y' = 0.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y = 0 => y' = -4/5 * y'-).
```

True

```
INIT, (FALL << BOUNCE).
```

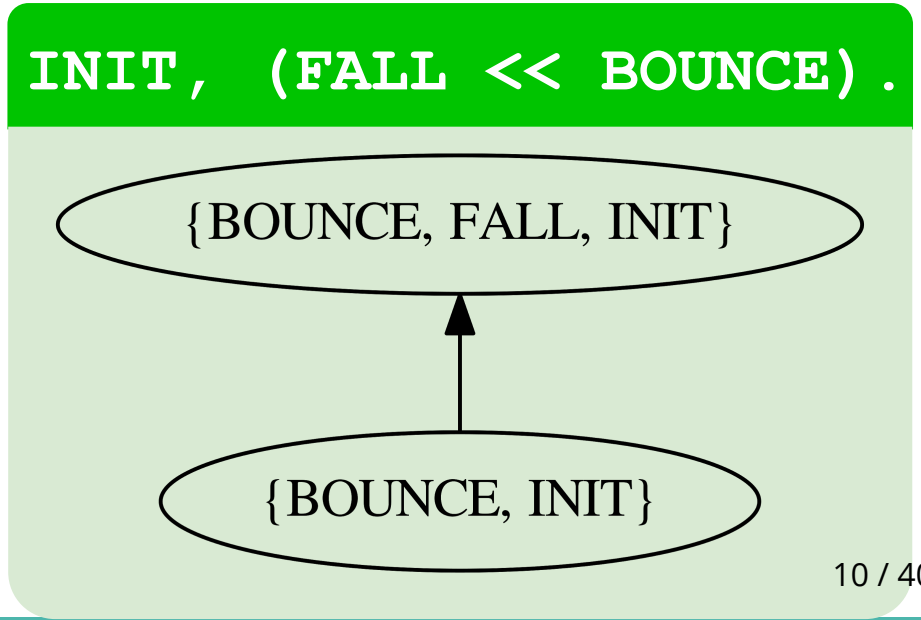
- `y'' = -10` and `y' = -4/5 * y'-` are inconsistent
- **`FALL`** is canceled because it is weaker than **`BOUNCE`**
- The MCS is **`{INIT, BOUNCE}`**

Constraint Hierarchy

Module priorities generate *constraint hierarchy,*
which is a poset

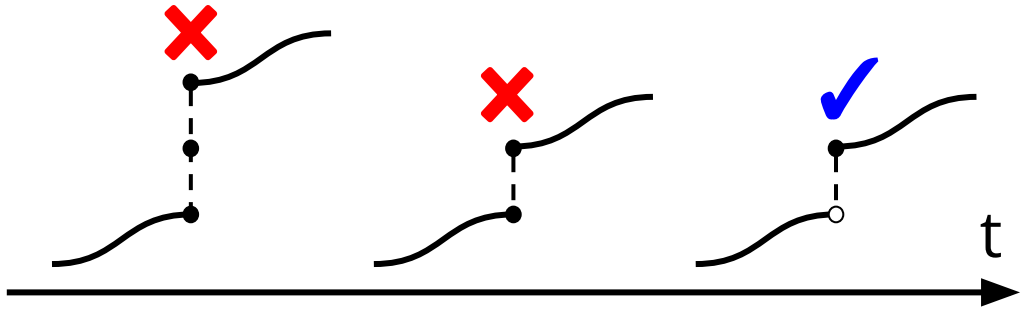
- whose elements are sets of modules and
- whose partial order is induced from module priority.

Constraint hierarchy is a *concise* representation of behaviors with *defaults* (e.g., falling) and *exceptions* (e.g., bouncing).



Differences between HA and HydLa (1)

- Variables are functions of (standard) time t (≥ 0)
  (cf. superdense time, hyperreal time)



expressive?

1. Multi-step jumps by cyber components
   … can be represented as *constraints (= relations) between initial and final values*
2. Simultaneous physical events (e.g. 3-body collision)
   … can be handled by *symbolic perturbation*

Differences between HA and HydLa (2)

- For translation from HA, we could employ a variable (taking discrete values) representing the current mode.
  - e.g., **mode =** k **=>** *(constraints of mode k)*
    (for k = 1, 2, ...)

Hybrid automata and HydLa are quite different,
and direct comparison is not easy

But we guess that their expressive power is not different in practice.

Outline

Rigorous Approaches to Hybrid Systems

Rigorous *numerical* simulators

Acumen [5] : Validated Numerics
Flow* [6] : Taylor model + Domain contraction
SpaceEX [7] : Template Polyhedra

Research question:

symbolic simulator?

Rigorous symbolic *theorem prover*

KeYmaera X [8] : differential dynamic logic

[5]Taha, W., et al.: Acumen: An open-source testbed for cyber-physical systems research. In: IoT360 2015 (2016)
[6]Chen, X., et al.: Flow*: An analyzer for non-linear hybrid systems. In: CAV 2013 (2013)
[7]Frehse G. et al.: SpaceEx: Scalable Verification of Hybrid Systems. In: CAV 2011 (2011)
[8] Fulton N., et al.: KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems. In: CADE25 (2015)

HyLaGI [9]

- A symbolic implementation of HydLa
- Features *nondeterministic* execution of models with symbolic parameters
  - based on automatic case analysis

Tossing of a Ball to a Ceiling

```
INIT <=> y = 9 & 3 < y' < 10.
FALL <=> [](y'' = -10).
BOUNCE <=> [](y = 10 => y' = -4/5 * y'-).

INIT, (FALL << BOUNCE).
```

[9]https://github.com/HydLa/HyLaGI

Simulation Result of Ball Tossed to a Ceiling

Case 1 : fall

---------Case 1---------
---------PP 1---------
t : 0
y : 9
y' : p[y, 1, 1]
y'' : -10
---------IP 2---------
t : 0->Infinity
y : 9+t^2*(-5)+t*p[y, 1, 1]
y' : t*(-10)+p[y, 1, 1]
y'' : -10
-parameter condition(Case1)-
p[y, 1, 1] : (3, 2*5^(1/2))

symbolic parameter

Case 2 : touch

---------Case 2---------
... (stuff deleted) ...
---------PP 4---------
t : (p[y, 1, 1]+(-1)*(-20
+p[y, 1, 1]^2)^(1/2))*1/10
y : 10
y' : (-20+p[y, 1, 1]^2)^(1/2)
y'' : -10
---------IP 6---------
t : (p[y, 1, 1]+(-1)*(-20
+p[y, 1, 1]^2)^(1/2))*1/10
->Infinity
y : 9+t^2*(-5)+t*p[y, 1, 1]
y' : t*(-10)+p[y, 1, 1]
y'' : -10
-parameter condition(Case2)-
p[y, 1, 1] : 2*5^(1/2)

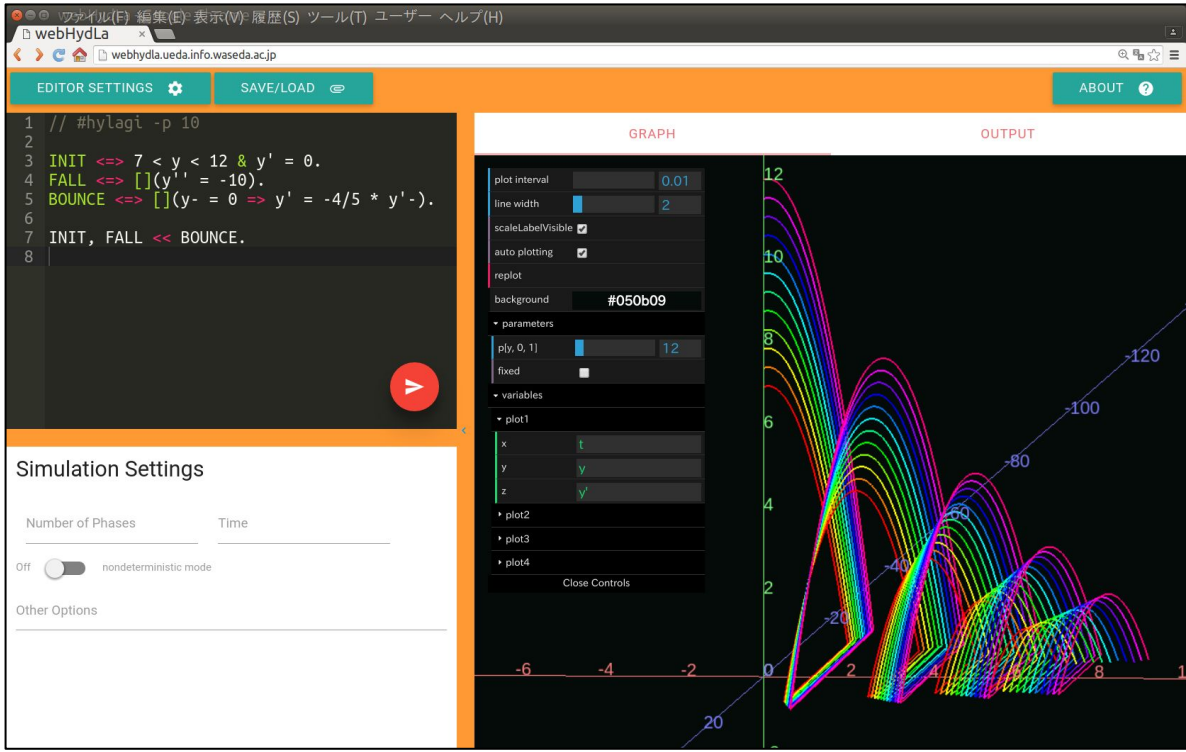
Case 3 : collide

---------Case 3---------
... (stuff deleted) ...
---------IP 7---------
t : (p[y, 1, 1]+(-1)*(-20
+p[y, 1 ,1]^2)^(1/2))*1/10
->Infinity
y : t^2*(-5)+63/5+p[y, 1, 1]
*(t+(-20+p[y, 1, 1]^2)^(1/2)*9/50)
+t*(-20+p[y, 1, 1]^2)^(1/2)*(-9)/5
+p[y, 1, 1]^2*(-9)/50
y' : t*(-10)+p[y, 1, 1]+(-20
+p[y, 1, 1]^2)^(1/2)*(-9)/5
y'' : -10
-parameter condition(Case3)-
p[y, 1, 1] : (2*5^(1/2), 10)

webHydLa[10]

Visualization tool for the understanding of results is desirable

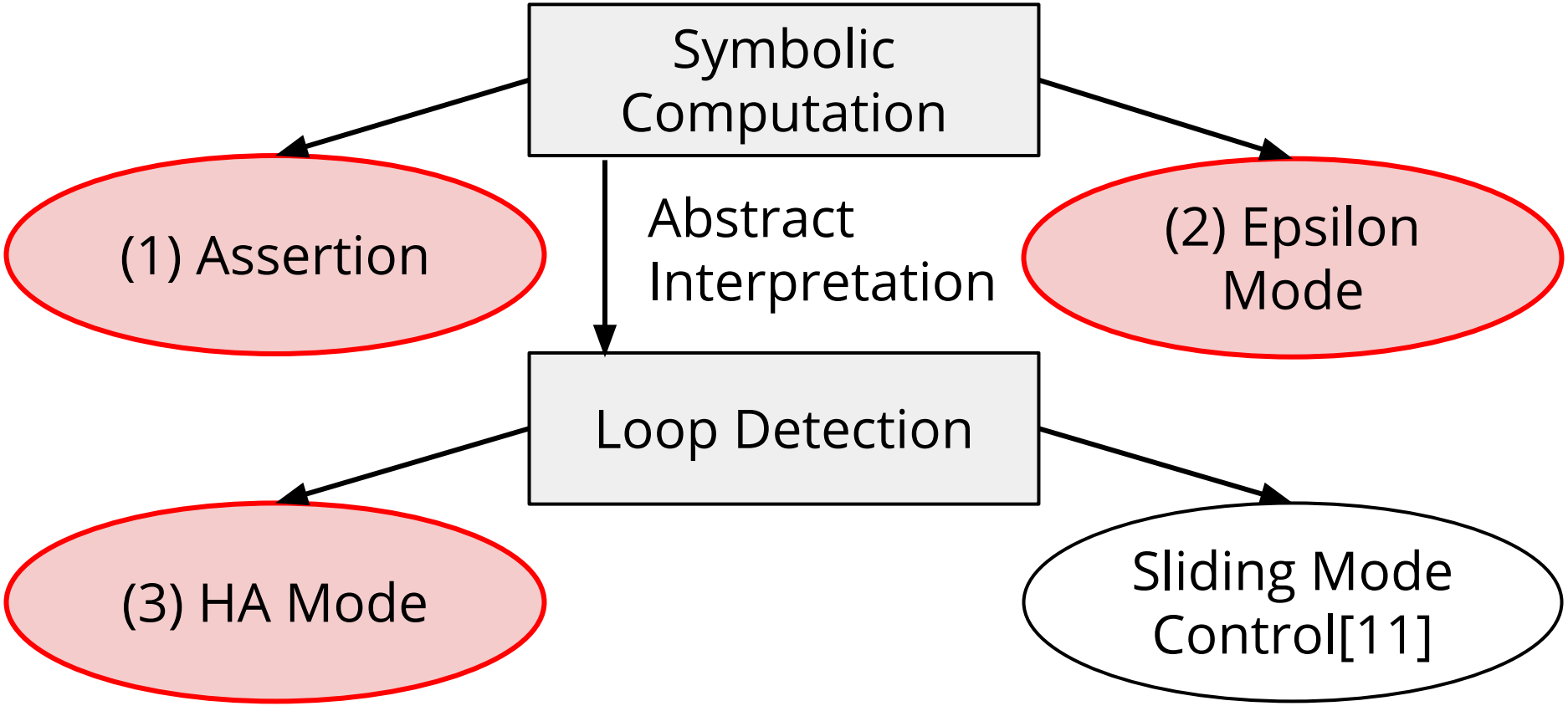
- Web IDE
- 2D and 3D visualization



[10]http://webhydla.ueda.info.waseda.ac.jp/

visualization of the simulation result of a bouncing particle

Various Functionalities of HyLaGI



[11]Betsuno, K., Matsumoto, S., Ueda, K.: Symbolic analysis of hybrid systems involving numerous discrete changes using loop detection. In: CyPhy 2016. 17–30. Springer (2017)

(1) Assertion

ASSERT can be used for bounded model checking

ASSERT(*G*) intuitively means $\Box(G)$ or $\Box(\neg G \Rightarrow \text{false})$

```
ASSERT(y >= 0)
```

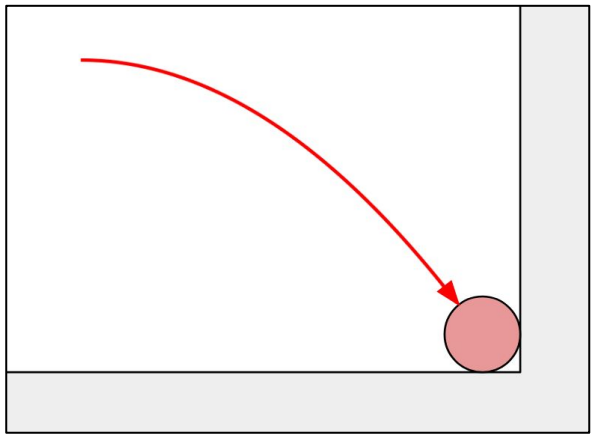
When `y < 0`, the assertion fails, and HyLaGI symbolically computes the range of parameters causing the failure

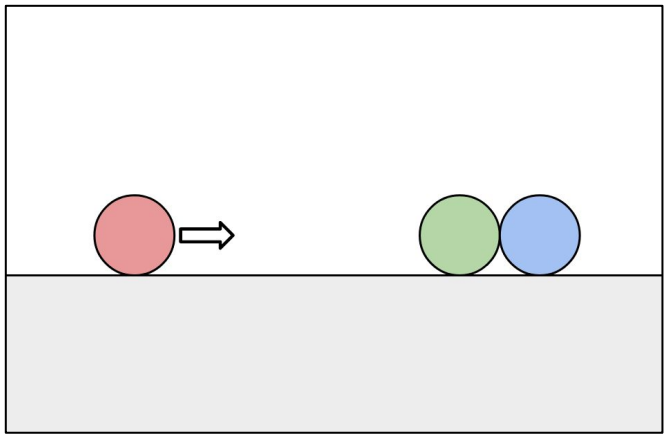
Examples are explained in the section of inverse problems

(2) Epsilon Mode

The simulation of hybrid systems may fall into a singular situation such as

1. A ball hits the wall and the floor at the same time
2. One of the two balls touching with each other is hit by the third ball[12]





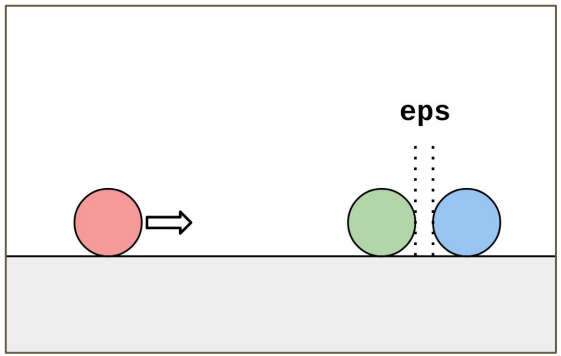
[12] Lee, E.A.: Constructive models of discrete and continuous physical phenomena. IEEE Access 2, 797–821 (2014)

(2) Epsilon Mode

These situations are considered the limits of non-singular situations such as

1. The ball hits the wall or the floor first
2. One of the two balls slightly apart is hit by the third ball

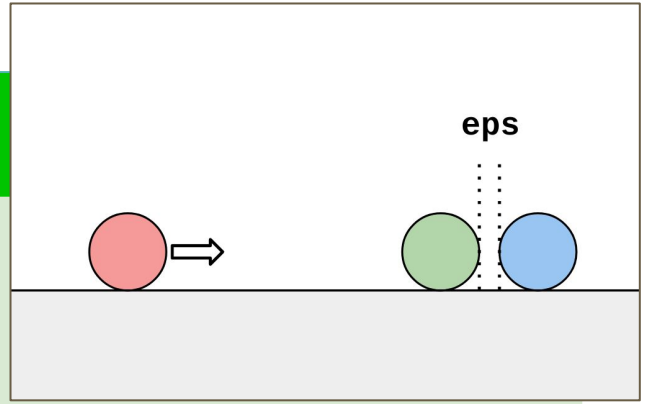
HyLaGI's epsilon mode [13] can take the limits of these models when the special symbolic parameter eps is used.



[13] Wakatsuki, Y., Masumoto, S., Ito, T., Wada, T., Ueda, K.: Model analysis by using micro errors in hybrid constraint processing system HyLaGI. In: The 32nd JSSST Annual Conference (2015 (in Japanese))

(2) Epsilon Mode

Three-body collision



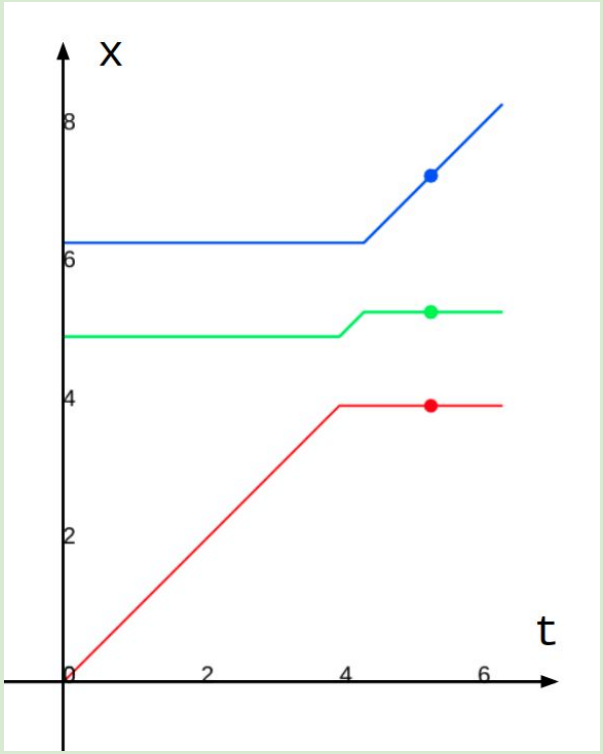

```
INIT <=> x1 = 0 & x2 = 5 & x3 = 6+eps
       & x1' = 1 & x2' = 0 & x3' = 0.
EPS  <=> 0 < eps < 0.1 & [](eps' = 0).
CONST(x) <=> [](x'' = 0).
COLLISION(xa, xb) <=>
   [](xa- = xb- - 1 => xa' = xb'- & xb' = xa'-).

INIT, EPS.
(CONST(x1),CONST(x2),CONST(x3))
   << (COLLISION(x1,x2), COLLISION(x2,x3)).
```

HyLaGI handles `eps` symbolically and is able to handle it as infinitesimal

(2) Epsilon Mode

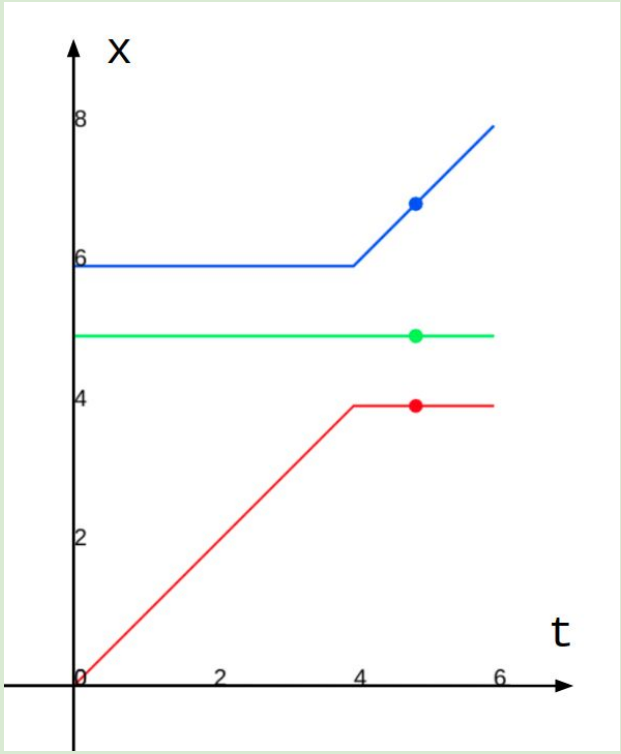
Three-body collision

The red ball hits first



Take the limit
(eps→0)

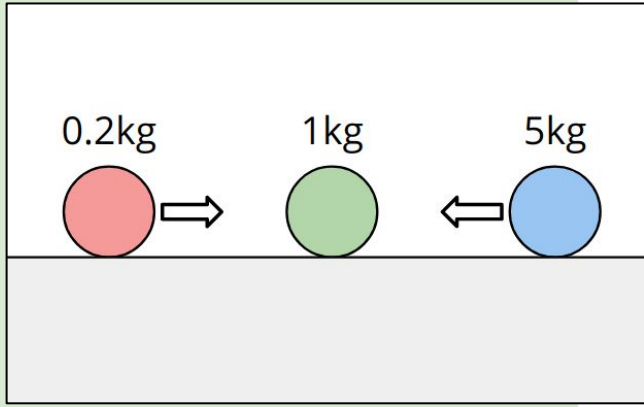
Newton’s cradle



(2) Epsilon Mode

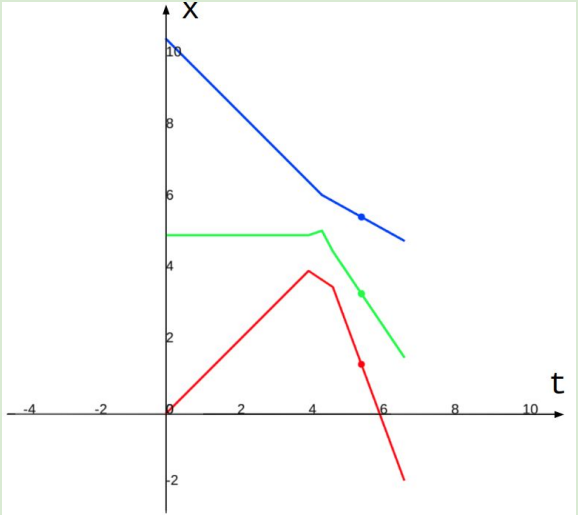
Collision of three bodies with different masses

```
INIT <=> x1 = 0 & x2 = 5 & x3 = 10+eps
         & x1' = 1 & x2' = 0 & x3' = -1.
EPS  <=> -0.1 < eps < 0.1 & [](eps' = 0).
MASS <=> [](m1 = 0.2 & m2 = 1 & m3 = 5).
CONST(x) <=> [](x'' = 0).
COLLISION(xa,ma,xb,mb) <=>
    [](xa- = xb- - 1 =>
       xa' = (xa'- *(ma-mb) + 2*mb*xb'-)/(ma+mb)
     & xb' = (xb'- *(mb-ma) + 2*ma*xa'-)/(ma+mb)).

INIT. EPS. MASS.
(CONST(x1),CONST(x2),CONST(x3))
   << (COLLISION(x1,m1,x2,m2), COLLISION(x2,m2,x3,m3)).
```

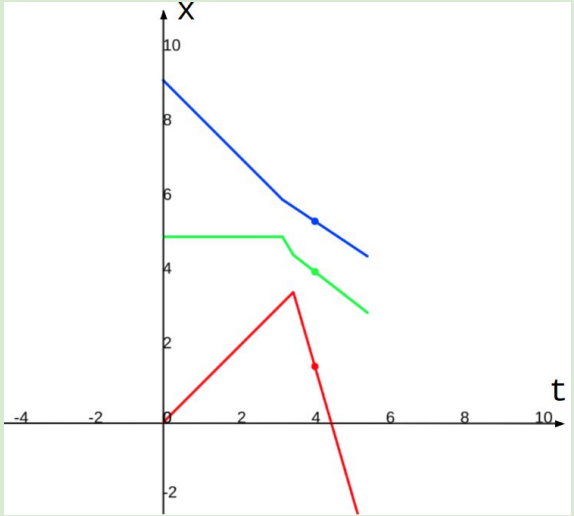


(2) Epsilon Mode

Collision of three bodies with different masses



If eps > 0,
The red one hits first



If eps < 0,
The blue one hits first

Left-hand and right-hand limits do not coincide

If the masses are equal, a two-sided limit exists

(3) Hybrid Automata Mode

HydLa users don't have to write automata

However, it is sometimes convenient to consider an automaton with explicit states

- for the analysis of systems
- for optimized simulation based on static analysis

The HA mode generates automata from HydLa programs by abstract interpretation [14]

- This analysis is based on the loop detection by symbolic computation

[14] Takeguchi, A., Wada, R., Matsumoto, S., Hosobe, H., Ueda, K.: An algorithm for converting hybrid constraint programs to hybrid automata. In: The 29nd JSSST Annual Conference, 2A-3 (2012 (in Japanese))

(3) Hybrid Automata Mode

Bouncing particle

```
INIT   <=> y > 0.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 0 => y' = -4/5 * y'-).

INIT, FALL << BOUNCE.
```

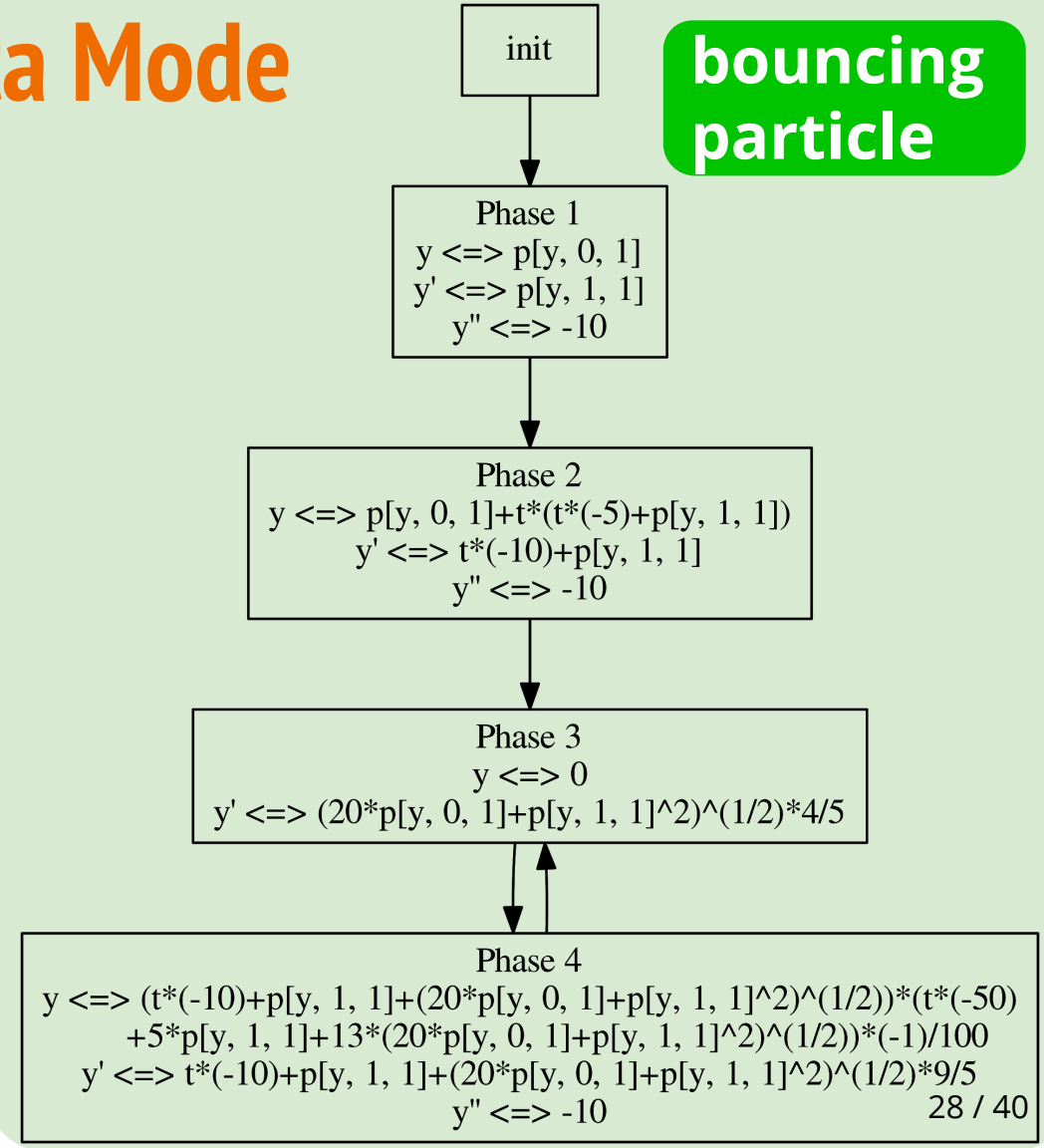
Initial values of `y` and `y'` must be parameterized

(No conditions on `y'` means that `y'` is fully parameterized)

(3) Hybrid Automata Mode

Applications of this functionality include

- LTL model checking
- Sliding mode control
- (Handling Zeno)



Outline

Discrete Asks and Continuous Asks

There are two categories of guarded constraints in HydLa

Thermostat

```
INIT   <=> p = 65 & sw = 0.
CONST  <=> [](sw' = 0).
ON     <=> [](sw = 1 => p' =  1).
OFF    <=> [](sw = 0 => p' = -2).
```

Continuous asks are enabled on intervals and determine continuous behavior

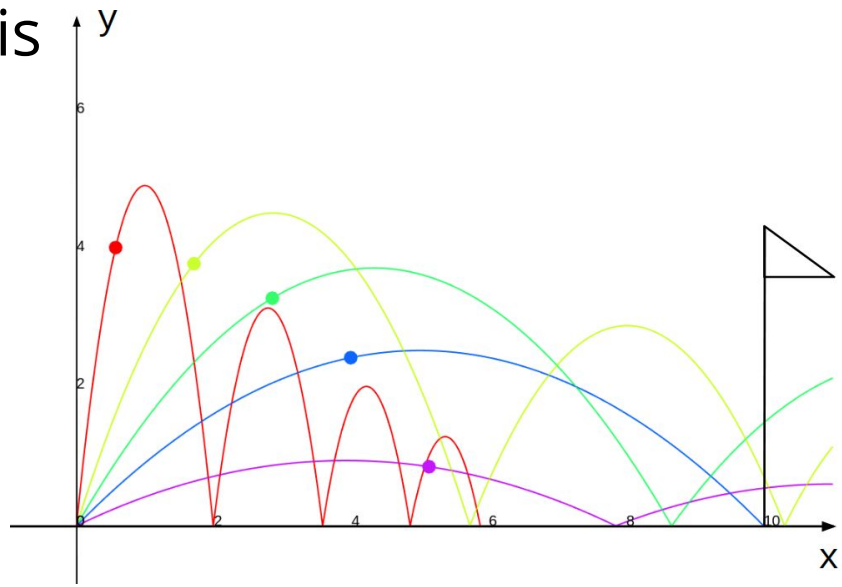
```
SWITCHON  <=> [](p- = 62 & sw- = 0 => sw = 1).
SWITCHOFF <=> [](p- = 68 & sw- = 1 => sw = 0).
```

Discrete asks are enabled only at time points and cause discrete changes

```
INIT, ON, OFF, CONST << (SWITCHON, SWITCHOFF).
```

Inverse Problems of Hybrid Systems

Inverse problems of hybrid systems are interesting because they involve case analysis



Can we hit hole-in-one?
Can we win the chicken race?

HyLaGI allows us to solve inverse problems of hybrid systems by using

- symbolic parameters
- assertion
- exhaustive search

Inverse Problems of Hybrid Systems

Hole-in-one

```
INIT     <=> x = 0 & y = 0
             & 1 < x' < 9 & y' = (100 - x'^2)^0.5.
AXCONST  <=> [](x'' = 0).
FALL     <=> [](y'' = -10).
BOUNCE   <=> [](y- = 0 => y' = -0.8*y'-).

ASSERT(!(y = 0 & 9.5 <= x <= 10)).
INIT, AXCONST, FALL << BOUNCE.
```

Search range

Negation of the goal state

- Modeling with parameter and **ASSERT**
- We write an assert condition as not reaching the cup

Inverse Problems of Hybrid Systems

Hole-in-one

Simulation result up to four bounces

Behavior	Parameter range
cup-in	[√((20-√39)*5/2), √((20+√39)*5/2)]
bounce, cup-in	[5/3*√((36-√935)/2), (5√14-10)/3]
bounce, bounce, cup-in	[5*√((244-√50511)/122), 5*√((61-6√86)*2/61)]
bounce, bounce, bounce, cup-in	[5/3*√((1476-√1952951)/82), 5/3*√((369-2√30134)*2/41)]
bounce, bounce, bounce, bounce	others

Persistent Consequent

HydLa allows us to use □-operators in consequent

```
[](x = 0 => [](y = 0)).
```

Once x becomes 0, y is always 0

We call it a persistent consequent

When the antecedent holds, the consequent is expanded as a constraint of the same strength

- Persistent consequent can be used to express irreversible changes such as object destruction

Inverse Problems of Hybrid Systems

Bouncing ball damaging the floor

```
INIT   <=> 5 < y < 10 & y' = 0 & dmg = 0.
FALL   <=> [](y'' = -10).
CONST  <=> [](dmg' = 0).
BOUNCE <=> [](y- = 0 => y' = -4/5 * y'-
                        & dmg = dmg- + y'-^2/100).
BREAK  <=> [](dmg >= 4 =>
                [](y'' = -10 & dmg' = 0)).

INIT, (FALL, CONST) << BOUNCE << BREAK.
ASSERT(!(y < 0)).
```

Assert condition states that the ball is not below the floor

Inverse Problems of Hybrid Systems

Bouncing ball damaging the floor

Simulation result up to five bounces

Behavior	Parameter range
bounce, bounce, bounce, bounce, bounce	(5, 7812500/968561)
bounce, bounce, bounce, bounce, break	[7812500/968561, 312500/36121)
bounce, bounce, bounce, break, through	[312500/36121, 12500/1281)
bounce, bounce, break, through	[12500/1281, 10)

Outline

Conclusion

Constraints allow natural modeling of hybrid systems with partial (uncertain) information

Constraint hierarchies allow concise modeling of default/exceptional behaviors

Symbolic simulation based on constraint satisfaction realizes various functionalities

- Nondeterministic execution
- Handling of infinitesimal quantities
- Construction of hybrid automata
- Solving inverse problems

Future work

Handling **difficult constraints**

- Nonlinear ODEs with parameters
- DAEs
- ODEs including many parameters

Dealing with **Zeno behavior** by loop detection

In-depth **expressive power comparison** with hybrid automata and other methods

Thank you for the attention!