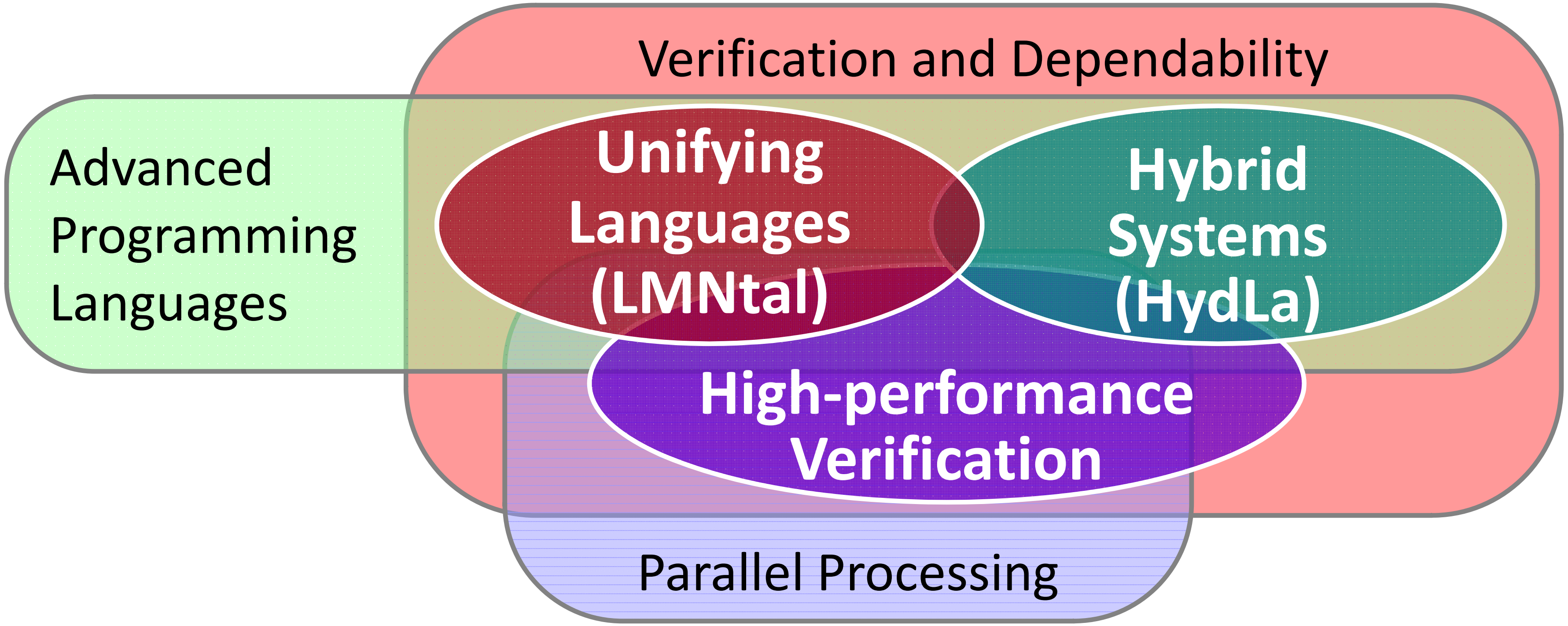
HydLa: A High-Level Language for Hybrid Systems

April 2012

Kazunori Ueda, Waseda University and NII
(with thanks to my students)

Computer Software, Vol.28, No.1 (2011), pp.306-311.

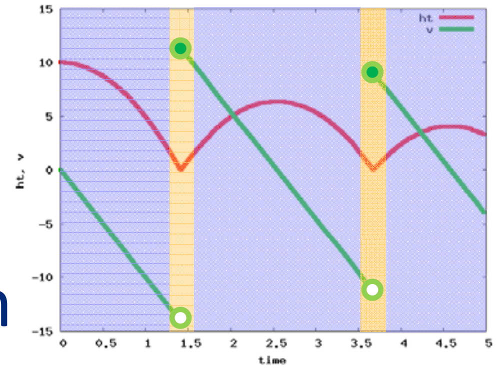
Research Groups and Their Relationship



- ✓ Three interrelated groups
- ✓ Three cross-cutting concerns

Hybrid systems

- Systems whose states can make both continuous and discrete changes



  Examples:
  - bouncing ball, billiard, . . .
  - thermostat + air conditioner + room
  - signal/crossing + roads/railroad + cars
  - (in general) Continuous systems with some components whose properties are described using case analysis
    - physical, biological, control, cyber-physical, etc.
- Related to CS, control engineering and apps.
- Programming language aspects rather unexplored

Challenges from the PL perspective

- Establish a high-level language
  - equipped with the notion of continuous time,
    - discrete-time systems could be dealt as infinite sequences
  - equipped with the notion of continuous changes,
    - in the true sense
  - that "correctly" handles uncertainties and errors of real values,
    - interval computation with conditional jumps
  - equipped with constructs for abstraction and parallel composition.

cf. Edward A. Lee, Cyber-Physical Systems - Are Computing Foundations Adequate? NSF Workshop on Cyber-physical Systems, 2006.

Constraint Programming

- A declarative paradigm in which a problem is described using (in)equations over continuous or discrete domains
  - requires no algorithms: constraint programming languages are often called modeling languages
  - the essence is computing with partial information
    - while AI+OR communities are most interested in constraint satisfaction
- Declarative description of hybrid systems = constraint programming of functions over time
  - logical implication (entailment) provides a mechanism for conditionals and synchronization

Example: $\Box(\underbrace{\text{e-stop} = 1}_{\text{(ask)}} \Rightarrow \underbrace{\text{speed}' = -4.0}_{\text{(tell)}})$

Existing Modeling Frameworks

- (more or less) procedural / state-based
  - Hybrid {Automata, Petri nets, I/O automata, Process Algebra} (models)
  - KeYmaera (languages)
- Constraint-based (domain = functions over time)
  - Hybrid CC (hybrid concurrent constraint language)
  - CLP(F) (constraint logic programming over real-valued functions)
  - Kaleidoscope '90 (discrete time)
  - HydLa (constraint hierarchy)

L. P. Carloni et al, Languages and Tools for Hybrid Systems Design, Foundations and Trends in Electronic Design Automation, Vol.1 (2006), pp.1-193.

HydLa : Overview and Features (1/2)

- Declarative (↔ procedural)
  - Minimizes new concepts and notations by employing popular math and logical notations
  - Describes systems using logic and hierarchy
- Constraint-based
  - Basic idea: defines functions over time using constraints including ODEs, and solves initial value problems
    - cf. streams and lists are defined by difference equations
  - Handles partial information properly
    - interval constraints fit well within HydLa

HydLa : Overview and Features (2/2)

- Features constraint hierarchies
  - It’s difficult to describe systems so that the constraints are consistent and well-defined.

    Example : bouncing ball, billiard, . . .
    - A ball normally falls by gravity (default), while it obeys the collision equation when it bounces (exception).
    - Frame problems occur in the description of complex systems
  - Want to define these properties concisely

Example 1 : Sawtooth function

constraint modules (rules)

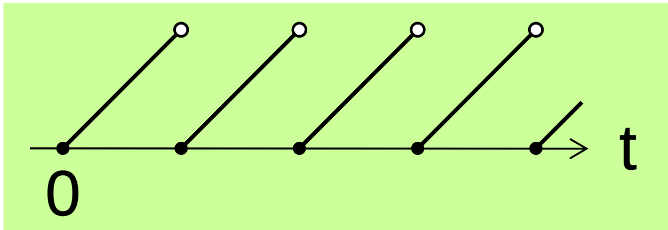
$$\begin{aligned}
&\text{INIT} \Leftrightarrow 0 \le f < 1.\\
&\text{INCREASE} \Leftrightarrow \Box(f' = 1).\\
&\text{DROP} \Leftrightarrow \Box(f^- = 1 \Rightarrow f = 0).\\
&\text{INIT, (INCREASE} << \text{DROP).}
\end{aligned}$$

guard

priority

- Describes properties at time 0
- Time argument is implicit

  $\Box(f' = 1)$ means $\forall t \ge 0\ (f'(t) = 1)$
- Family of sawtooth functions with the slope 1 and the range [0, 1)
- The value of f at a specific time point is just [0, 1) but all functions reach all values [0, 1) and oscillate.



Example 2 : Bouncing Ball

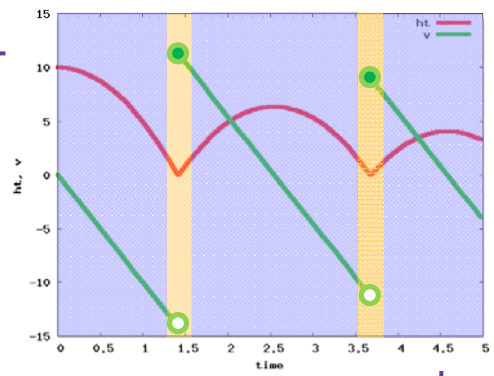
$$\text{INIT} \Leftrightarrow ht = 10 \wedge ht' = 0.$$

$$\text{PARAMS} \Leftrightarrow \Box(g = 9.8 \wedge c = 0.5).$$

$$\text{FALL} \Leftrightarrow \Box(ht'' = -g).$$

$$\text{BOUNCE} \Leftrightarrow \Box(ht- = 0 \Rightarrow ht' = -c \times (ht'-)).$$

$$\text{INIT, PARAMS, (FALL} << \text{BOUNCE)}.$$



- When the ball is not on the ground, {INIT,PARAMS,FALL,BOUNCE} is maximally consistent
- When the ball is on the ground, {INIT,PARAMS,BOUNCE} is maximally consistent
- Basic HydLa defines a program as the pair of (i) a poset of rule sets and (ii) rule definitions.

Syntax of Basic HydLa

(program)	P	::=	(RS, DS)
(rule sets)	RS	::=	poset of sets of R
(definitions)	DS	::=	set of D's with different LHS
(definition)	D	::=	R ⇔ C
(constraint)	C	::=	A | C ∧ C | G ⇒ C | □C | ∃x.C
(guard)	G	::=	A | G ∧ G
(atomic constraint)	A	::=	E ***relop*** E
(expression)	E	::=	***normal exp.*** | E′ | E–

function from R to C

Syntax of Basic HydLa: Comments

- A program is a pair of
  - poset of “sets of rules” (RS) and
  - rule definitions (DS).

  Example: {INIT,PARAMS,BOUNCE}
  $\prec$ {INIT,PARAMS,FALL,BOUNCE}
  - How to derive RS from << is beyond Basic HydLa
- HydLa / Basic HydLa is a language scheme in which the underlying constraint system is left unspecified
- ∃x . C realizes dynamic creation of variables
  - Example: creation and activation of new timers
  - ∃ is eliminated at runtime using Skolem functions

Semantics of Basic HydLa

- Declarative semantics (Ueda, Hosobe, Ishii, 2011)
  - What trajectories does a HydLa program denote?

- Operational semantics
  (Shibuya, Takata, Ueda, Hosobe, 2011)
  - How to compute the trajectories of a given HydLa program?

- Unlike many other programming languages, declarative semantics should come first since
  - completeness of the operational semantics can't be expected and
  - diverse execution methods could be explored

Declarative Semantics of Basic HydLa

- The purpose of a HydLa program is to define the constraints on a family of trajectories.

$$\overline{x}(t) = \{x_i(t)\}_{i \geq 1} \quad (t \geq 0)$$

- Declarative semantics, first attempt

$$\overline{x}(t) \models (RS, DS)$$

  - Works fine for programs not containing □ in the consequents of conditional constraints G ⇒ C [JSSST '08].

    Example: Systems with a fixed number of components and without delays

Declarative Semantics of Basic HydLa

- Not only trajectories, but also constraint sets defining the trajectories, change over time
  - Reason 1: change of maximally consistent sets
  - Reason 2: conditional constraints may discharge consequents (history sensitive)
    - When the consequent of a constraint starts with □, whether it's in effect or not depends on whether the corresponding guard held in the past
- Declarative semantics (refined)

$$\langle \bar{x}, Q \rangle \models (RS, DS)$$

$\boldsymbol{Q}(t)$: module definitions with dynamically added consequents

Preliminary: □-closure

- We identify a conjunction of constraints with a set of constraints.
- We regard a set of constraints as a function over time.
  - A constraint C in a program is regarded as a function C(0) = C, C(t) = { } (t>0) .
- □-closure * : Unfolds the topmost □-formulas dynamically and recursively.

Example: C = {f=0,□{f’=1}}

C*(0) = {f=0,f’=1,□{f’=1}}

C*(t) = {f’=1} (t>0)

Declarative Semantics

$$\langle \overline{x}, Q \rangle \models (MS, DS) \Leftrightarrow \text{(i)} \wedge \text{(ii)} \wedge \text{(iii)} \wedge \text{(iv)}$$

(i) $\forall M\,(Q(M) = Q(M)^*)$ — □-closure

(ii) $\forall M\,(DS(M)^* \subseteq Q(M)^*)$ — extensiveness

(iii) $\forall t\, \exists E \in MS\,($

$(\overline{x}(t) \Rightarrow \{Q(M)(t) \mid M \in E\})$ — satisfiability

$\wedge \neg \exists \overline{x}'\, \exists E' \in MS\,($
$\forall t' < t\,(\overline{x}'(t') = \overline{x}(t'))$
$\wedge E \prec E'$
$\wedge \overline{x}'(t) \Rightarrow \{Q(M)(t) \mid M \in E'\})$ — maximality

$\wedge \forall d\, \forall e\, \forall M \in E($
$(\overline{x}(t) \Rightarrow d) \wedge ((d \Rightarrow e) \in Q(M)(t))$
$\Rightarrow e \subseteq Q(M)(t)))$ — ⇒-closure

(iv) $Q(M)(t)$ at each t is the smallest set satisfying (i)-(iii)

Example 3 : Absence of back propagation

$$P = ((\text{Powerset}(\{D,E,F\}), \subsetneq), DS)$$

$$DS = \{\, D \Leftrightarrow y = 0,$$

$$E \Leftrightarrow \Box(y' = 1 \wedge x' = 0),$$

$$F \Leftrightarrow \Box(y = 5 \Rightarrow x = 1) \,\}$$

a. $y(t)=t$, $x(t)=1$ satisfies D, E, F at $0 \le t$.

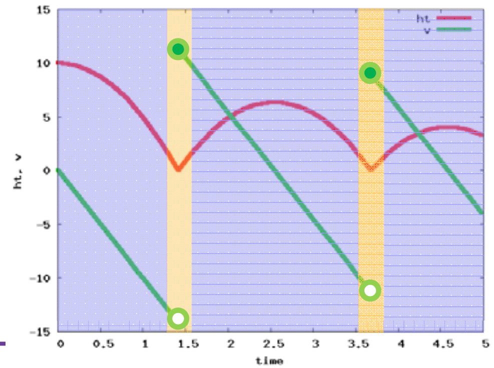
b. $y(t)=t$, $x(t)=2$ satisfies D, E, F at $0 \le t < 5$ and D, E at $t=5$. It again satisfies D, E, F at $t \ge 5$.

c. $y(t)=t$, $x(t)=2$ $(t<5)$, $x(t)=1$ $(t \ge 5)$ satisfies D, E, F at $0 \le t < 5$ and D, F at $t=5$. It again satisfies D, E, F at $t \ge 5$.

All of a., b. and c. satisfy local maximality and hence satisfy P.

Example 4 : Bouncing Ball, revisited

$$P = (RS,\ DS)$$
$$RS = (\{\{I,Pa,B\},\ \{I,Pa,F,B\}\},\ \{\{I,Pa,B\} < \{I,Pa,F,B\}\})$$
$$DS = \{\ \ I \Leftrightarrow ht{=}10 \wedge ht'{=}0,$$
$$Pa \Leftrightarrow \square(g{=}9.8 \wedge c{=}0.5),$$
$$F \Leftrightarrow \square(ht'' = -g),$$
$$B \Leftrightarrow \square(ht{-}{=}0 \Rightarrow ht'{=}{-}c{\times}(ht'{-}))\}$$



- ht and ht’ are not differentiable when bouncing
- However, to solve ODEs on ht and ht’, right continuity of ht and ht’ at the bouncing must be assumed
- To determine ht at the bouncing, left continuity of ht must be assumed as well. (cf. ht’ is determined from B.)

➔ Trajectories with differential constraints should assume both right and left continuity with higher priority.

Example 5 : Behaviors defined without ODEs

$$
\begin{aligned}
P &= (RS, DS) \\
RS &= (\{\{A,C\}, \{A,B,C\}\}, \{\{A,C\} < \{A,B,C\}\}) \\
DS &= \{\, A \Leftrightarrow f=0 \wedge \Box(f' = 1), \\
&\qquad B \Leftrightarrow \Box(g=0), \\
&\qquad C \Leftrightarrow \Box(f=5 \Rightarrow \exists a.(a=0 \wedge \Box(a'=1) \\
&\qquad\qquad \wedge \Box(a=2 \Rightarrow g=1))) \,\}
\end{aligned}
$$

- g is an impulse function that fires at time 7 (= 5+2).
  - an example of non-right-continuous functions

$$\Box(0.9<a \wedge a<1.1) \wedge \Box(a'=b)$$

- a is a set of all differentiable trajectories whose ranges are (0.9, 1.1) .

Example 6 : Zeno behavior

$P = (MS, DS)$

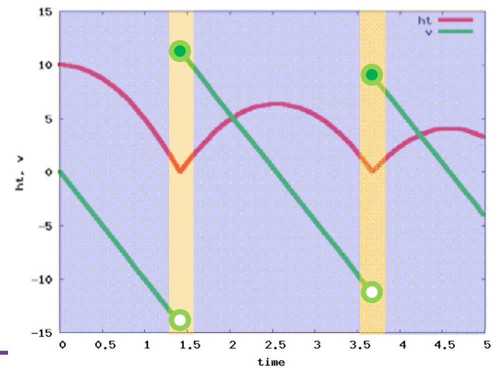
$RS = (\{\{I, Pa, B\}, \{I, Pa, F, B\}\}, \{\{I, Pa, B\} < \{I, Pa, F, B\}\})$

$DS = \{\ I \Leftrightarrow ht=10 \wedge ht'=0,$

$\quad Pa \Leftrightarrow \Box(g=9.8 \wedge c=0.5),$

$\quad F \Leftrightarrow \Box(ht'' = -g),$

$\quad B \Leftrightarrow \Box(ht- =0 \Rightarrow ht' = -c \times (ht'-))\}$



- This doesn't define a trajectory after the Zeno time.
- A rule for defining the trajectory after Zeno:

$\Box(ht- =0 \wedge ht'- =0 \Rightarrow \Box(ht=0))$

  - Checking of the guard condition would require a technique not covered by the current operational semantics.

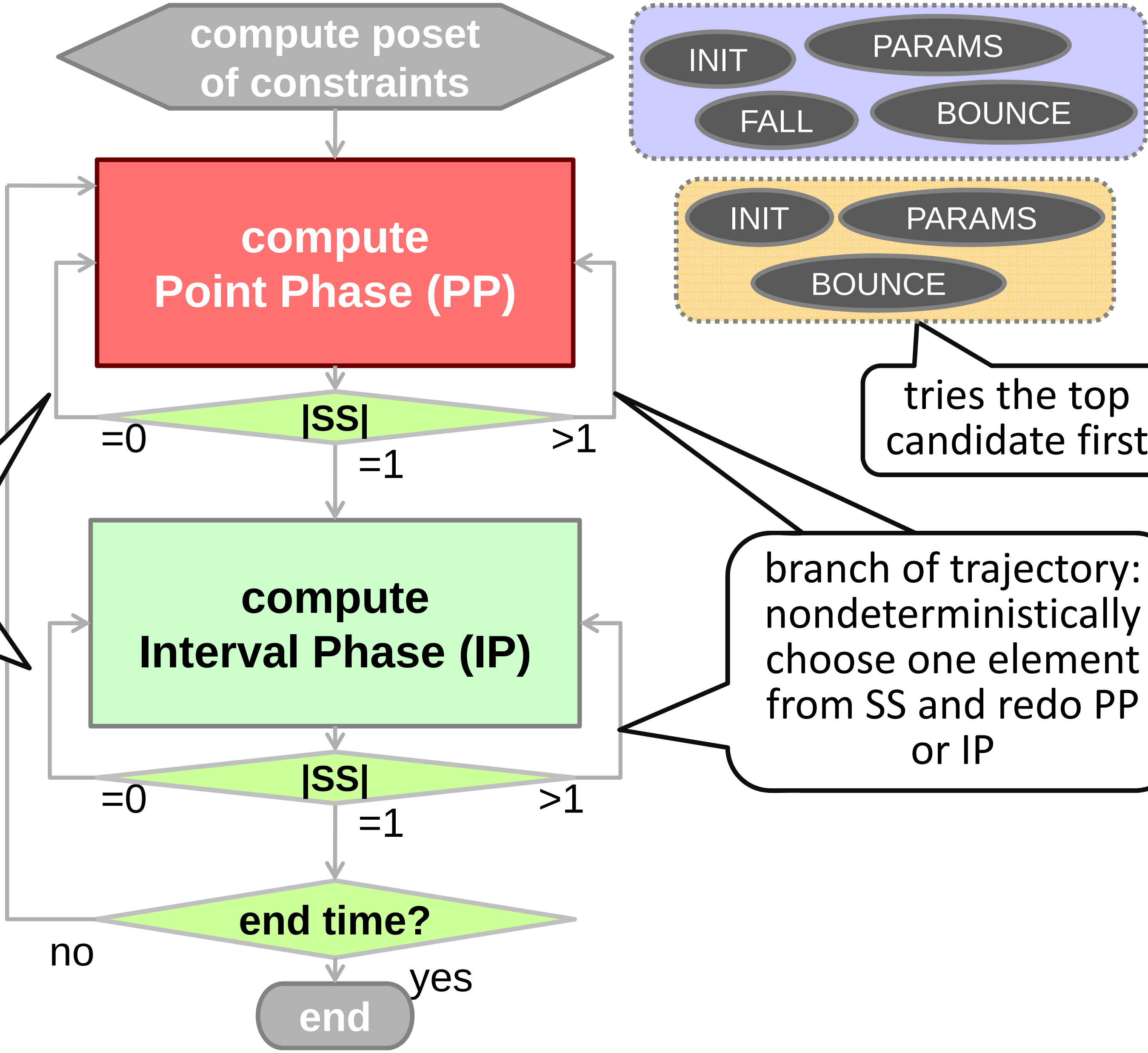
Execution algorithm

each phase updates the maximal consistent set and simulation time T

SS (store set) : set of possible stores

failure: choose the next candidate set and redo PP or IP

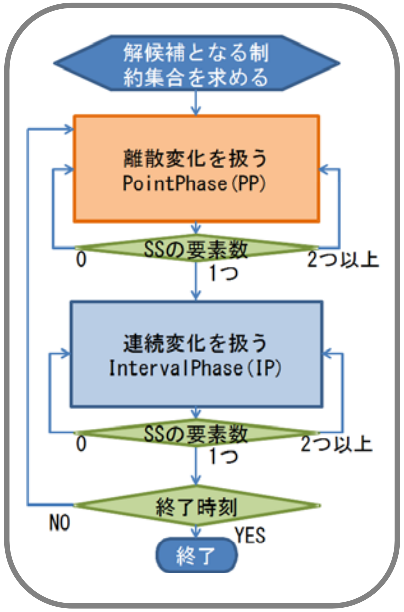
an element of SS represents a result of execution of PP or IP

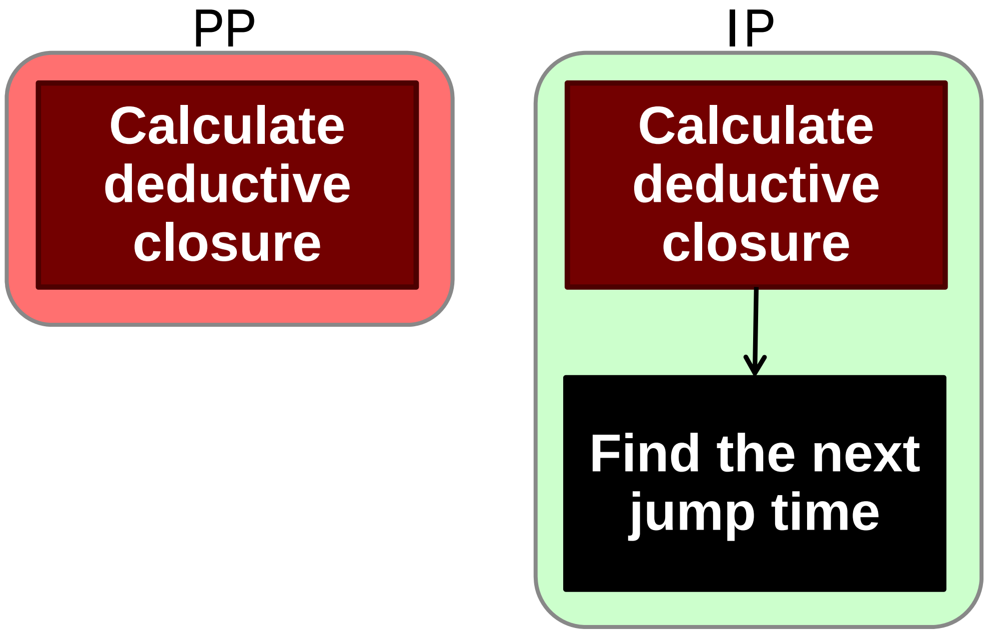


tries the top candidate first

branch of trajectory: nondeterministically choose one element from SS and redo PP or IP

Algorithm for Point Phase and Interval Phase





Closure calculation repeatedly checks the antecedents of conditional constraints

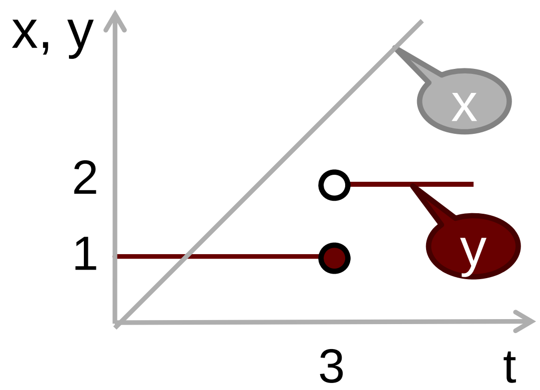
IP computes the next jump time (minimum of the following):

1. a conditional constraint becomes effective
2. a conditional constraint becomes ineffective
3. a ruled-out constraint becomes consistent with effective ones
4. the set of effective constraints becomes inconsistent

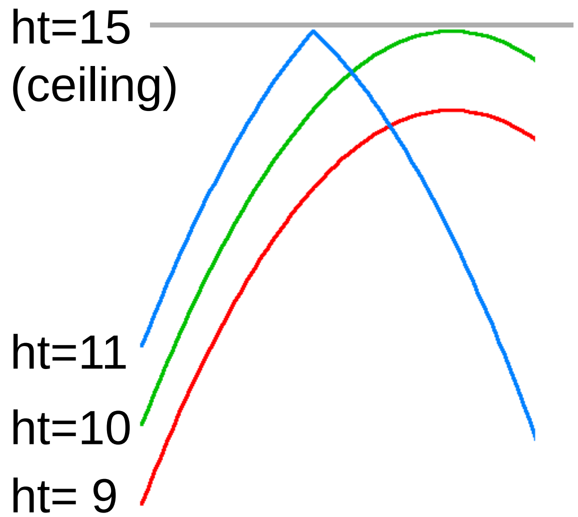
Execution algorithm should handle:

1. conditions that starts to hold "after" some time point
   - need to compute the greatest lower bound of the time interval

```
A ⇔ x=0.
B ⇔ □ (y=1).
C ⇔ □ (x'=1 ∧ (x>3 ⇒ y=2)).
A, (B << C).
```

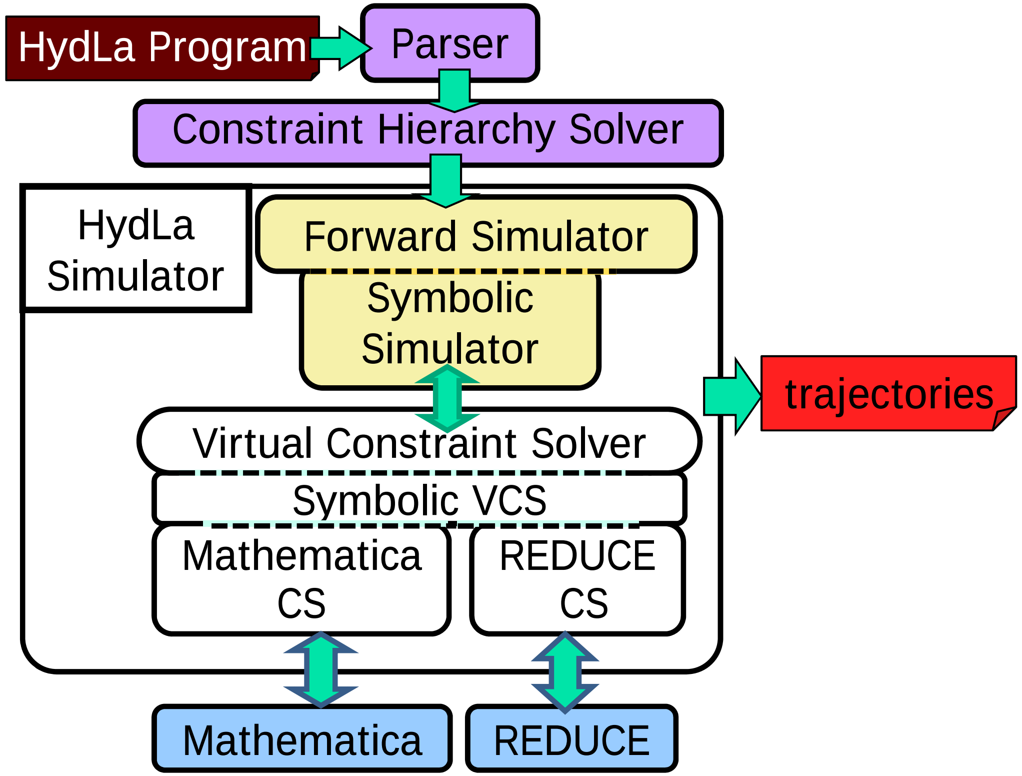


2. initial values given as intervals
   - could be divided into a subinterval that entails a guard and another that does not entail the guard
3. systems with parameters
   - needs symbolic computation



Hyrose: an implementation of HydLa

- implemented in C++
- 38,000 LOC
- Key features:
  - simulation with symbolic parameters
  - nondeterministic simulation



- http://www.ueda.info.waseda.ac.jp/hydla/

Example: Bouncing ball with 5<ht<15

```
#---------Case 1---------
#---------1---------
---------PP---------
time          : 0
ht            : pht
ht'           : 0
ht''          : (-49)/5

---------IP---------
time          : 0 ->
                1/7*10^(1/2)*pht^(1/2)
ht            : pht+(-49)/10*t^2
ht'           : (-49)/5*t
ht''          : (-49)/5
```

```
#---------2---------
---------PP---------
time          : 1/7*10^(1/2)*pht^(1/2)
ht            : 0
ht'           :
28/5*(2/5)^(1/2)*pht^(1/2)
ht''          : UNDEF

---------IP---------
...

#---------parameter condition---------
pht           : (5, 2205/338)


#---------Case 2---------
...
#---------parameter condition---------
pht           : [2205/338, 15)
```

Conclusion

- Defined Basic HydLa and gave a declarative semantics
  - now handles dynamically evolving systems
  - obtained through a lot of preliminary study (modeling examples, prototype implementation, etc.)
- Operational semantics is also developed
  - and evolved into a nondeterministic algorithm that allow uncertainties
  - however, completeness doesn't hold even for a very small class of ODEs [Henzinger '96]
- Modeling languages must be given a declarative semantics first to allow flexible execution
- Adopted simple notions of time and trajectory
  - adopting hybrid time is a topic of future work