HyLaGIの主要関数

- CheckConsistencyPP
  - PP(離散変化時)において与えた制約が成り立つのか判定
- CheckConsistencyIP
  - 与えられた制約に含まれる微分方程式を解く
  - IP(連続変化時)において与えた制約が成り立つのか判定
- FindMinTime・CompareMinTime
  - 次の離散変化の時刻を１つのモジュールのガード条件を解いて求める
  - 求めた離散変化の時刻を比べて最小の時刻を求める

HyLaGIの主要関数

- CheckConsistencyPP
  - 記号定数の条件CPと制約Sを同時に満たす制約S内の変数が存在するような記号定数の条件CP_tmpを求める
  - CP_tmpがなければ (false, CP) を返す
    - 記号定数が範囲の中の任意の値で制約を満たさない.
  - CP_tmpが与えた記号定数の条件と一致するならば (true, CP) をそのまま返す
    - 記号定数が範囲の中の任意の値で制約を満たす.
  - CP_tmpが与えた記号定数の条件と一致しないならば (true, CP_tmp) か (false, CP∧¬CP_tmp)を非決定的に返す
    - 記号定数の範囲の中に制約を満たす値と制約を満たさない範囲がある.

Input: 制約ストア S, 記号定数の条件 CP
Output: 充足可能性, 記号定数の条件

$V := GetVariables(S)$
$CP_{tmp} := \exists V(S \wedge CP)$
if $CP_{tmp} = false$ **then**
  return $(false, CP)$
else if $CP_{tmp} = CP$ **then**
  return $(true, CP)$
else
  return $GetElement(\{(true, CP_{tmp}),$
    $(false, CP \wedge \neg CP_{tmp})\})$
end if

HyLaGIの主要関数

- 例
  - 最初のPP
    - CP = 9 <= py <= 11 (py: yの記号定数)
    - S = {y = py, y’ = 10, y’’ = -10}
    - V = {y, y’, y’’}
    - CP_tmp = ∃V(S∧CP) = 9 <= py <= 11
    - CP_tmp = CP なので (true, 9<=py<=11)を返す.
  - ２回目のPP
    - CP = 10 <= py <= 11
    - S = {y = 15, y’ = 2(-50+5py)^(1/2),
      y’’ = -10, y’ = -0.8y’-}
    - V = {y, y’, y’’}
    - CP_tmp = ∃V(S∧CP) = (py = 10)
    - (true, py=10)か(false, 10<py<=11)を返す.

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

Input: 制約ストア S, 記号定数の条件 CP
Output: 充足可能性, 記号定数の条件

$V := GetVariables(S)$
$CP_{tmp} := \exists V(S \wedge CP)$
if $CP_{tmp} = false$ **then**
  return $(false, CP)$
else if $CP_{tmp} = CP$ **then**
  return $(true, CP)$
else
  return $GetElement(\{(true, CP_{tmp}), (false, CP \wedge \neg CP_{tmp})\})$
end if

HyLaGIの主要関数

CheckConsistencyIP

- CheckConsistencyPPとほとんど同じ処理を行なっている.
- 異なった処理を行なっている以下の２つ
  - はじめに制約Sに含まれる微分方程式を解いて時刻の式S_tを導出する
  - tが正の近傍のときに記号定数の条件CPとS_tを同時に満たす制約S内の変数が存在するような記号定数の条件CP_tmpを求める.

Input: 制約ストア S，記号定数の条件 CP
Output: 充足可能性，記号定数の条件

$S_t := SolveDifferentialEquation(S)$
$V := GetVariables(S_t)$
$CP_{tmp} := \exists V(Inf\{t \mid \exists_t(S_t \land t > 0)\} = 0) \land CP)$
if $CP_{tmp} = false$ **then**
  return $(false, CP)$
else if $CP_{tmp} = CP$ **then**
  return $(true, CP)$
else
  return $GetElement(\{(true, CP_{tmp}),$
    $(false, CP \land \neg CP_{tmp})\})$
end if

HyLaGIの主要関数

- 例
  - 最初のIP
    - S = {y = py, y' = 10, y'' = -10}
    - S内の微分方程式を解くと
      S_t = {y = py + 10t – 5t^2, y' = 10 - 10t, y'' = -10}
    - CP = 9 <= py <= 11
    - CP_tmp = 9 <= py <= 11
    - CP_tmp = CP なので
      (true, 9 <= py <= 11)

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

Input: 制約ストア S，記号定数の条件 CP
Output: 充足可能性，記号定数の条件

$S_t := \mathit{SolveDifferentialEquation}(S)$
$V := \mathit{GetVariables}(S_t)$
$CP_{tmp} := \exists V(\mathit{Inf}\{t \mid \exists_t(S_t \wedge t > 0)\} = 0) \wedge CP)$
if $CP_{tmp} = \mathit{false}$ **then**
  return (false, CP)
else if $CP_{tmp} = CP$ **then**
  return (true, CP)
else
  return $\mathit{GetElement}(\{(\mathit{true}, CP_{tmp}),$
    $(\mathit{false}, CP \wedge \neg CP_{tmp})\})$
end if

HyLaGIの主要関数

- FindMinTime・CompareMinTime
  - FindMinTime
    - CheckConsistencyIPで導出した時刻の式とガード条件から
      ガード条件を満たす最小時刻tを計算する.
    - 最小時間 t は必ず 0 より大きい.
  - CompareMinTime
    - FindMinTimeで求めた最小時刻t同士を比較することで
      最小離散変化の時刻を導出する.
    - 記号定数の条件によって最小離散時間が変化する場合はそれぞれの場合の
      記号定数の条件を求める.

HyLaGIの主要関数

➢例

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

- CheckConsistencyIPで導出した時刻の式は以下のようになる.
  - y = py + 10t – 5t^2, y’ = 10 - 10t, y’’ = -10　(py: y の記号定数)
- 上の式とBOUNCEのガード条件を組み合わせて最小時間を計算(FindMinTime)
  - BOUNCE: (y- = 15) ∧ (y = py + 10t – 5t^2) => t = 1- (py/5 - 2)^(1/2)
- 記号定数の条件と最小離散時間を計算(CompareMinTime)
  - py >= 10 の場合は t = 1- (py/5 - 2)^(1/2) で天井に接触する.
  - py < 10 の場合は t = ∞ で離散変化が起こらない.

sageMathでの実装

- CheckConsistencyPP
  - 最初のPP
    - CP = 9 <= py <= 11 (py: yの記号定数)
    - S = {y = py, y' = 10, y'' = -10}

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

```
== Check Consistency PP by QE ========================================
==== input ======
constr store    :       [[y == p_y, dy == 10, ddy == -10]]
symbolic params :       [[p_y >= 9, p_y <= 11]]
# tmp_parameters is modified
==== output =====
consistency     :       [True, False]
updated params  :       [[py - 9 >= 0 /\ py - 11 <= 0], []]
==== measure ====
time            :       0.0954190000000147
```

sageMathでの実装

➢例

- 最初のPP
  - CP = 9 <= py <= 11 (py: yの記号定数)
  - S = {y = py, y’ = 10, y” = -10}
  - V = {y, y’, y”}
  - CP_tmp = ∃V(S∧CP) = 9 <= py <= 11
  - CP_tmp = CP なので (true, 9<=py<=11)を返す.
- ２回目のPP
  - CP = 10 <= py <= 11
  - S = {y = 15, y’ = 2(-50+5py)^(1/2),
    y” = -10, y’ = -0.8y’-}
  - V = {y, y’, y”}
  - CP_tmp = ∃V(S∧CP) = (py = 10)
  - (true, py=10)か(false, 10<py<=11)を返す.

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

Input: 制約ストア S, 記号定数の条件 CP
Output: 充足可能性, 記号定数の条件

$V := \mathit{GetVariables}(S)$
$CP_{tmp} := \exists V(S \wedge CP)$
if $CP_{tmp} = \mathit{false}$ **then**
  return (false, CP)
else if $CP_{tmp} = CP$ **then**
  return (true, CP)
else
  return $\mathit{GetElement}(\{(\mathit{true}, CP_{tmp}), (\mathit{false}, CP \wedge \neg CP_{tmp})\})$
end if

sageMathでの実装

- CheckConsistencyPP
  - p5の例題を動かす
  - ２回目のPP
    - CP = 10 <= py <= 11
    - S = {y = 15, y' = 2(-50+5py)^(1/2),
      y'' = -10, y' = -0.8y'-}

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

```
== Check Consistency PP by QE ==========================================
==== input ======
constr store    :       [[dy == 2*sqrt(5*p_y - 50), y == 15, ddy == -10, dy == -4/5*dy]]
symbolic params :       [[p_y >= 10, p_y <= 11]]
```

- この入力だとQEを行うところで止まる.
  - y' = 2(-50+5py)^(1/2) を py について解いて式変形することで回避可能

sageMathでの実装

- CheckConsistencyPP
  - p5の例題を動かす
  - ２回目のPP
    - CP = 10 <= py <= 11
    - S = {y = 15, y’ = 2(-50+5py)^(1/2), y” = -10, y’ = -0.8y’-}

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

```
== Check Consistency PP by QE ==========================================
==== input ======
constr store     :        [[p_y == 1/20*dy^2 + 10, y == 15, ddy == -10, dy == -4/5*dy]]
symbolic params :        [[p_y >= 10, p_y <= 11]]
# tmp_parameters is modified
==== output =====
consistency      :        [True, False]
updated params   :        [[py - 10 = 0], [py - 10 > 0 /\ py - 11 <= 0]]
==== measure ====
time             :       1.7195779999999985
```

sageMathでの実装

➢例

- 最初のPP
  - CP = 9 <= py <= 11 (py: yの記号定数)
  - S = {y = py, y’ = 10, y’’ = -10}
  - V = {y, y’, y’’}
  - CP_tmp = ∃V(S∧CP) = 9 <= py <= 11
  - CP_tmp = CP なので (true, 9<=py<=11)を返す.
- 2回目のPP
  - CP = 10 <= py <= 11
  - S = {y = 15, y’ = 2(-50+5py)^(1/2), y’’ = -10, y’ = -0.8y’-}
  - V = {y, y’, y’’}
  - CP_tmp = ∃V(S∧CP) = (py = 10)
  - (true, py=10)か(false, 10<py<=11)を返す.

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

Input: 制約ストア S, 記号定数の条件 CP
Output: 充足可能性, 記号定数の条件

$V := GetVariables(S)$
$CP_{tmp} := \exists V(S \wedge CP)$
if $CP_{tmp} = false$ **then**
  return $(false, CP)$
else if $CP_{tmp} = CP$ **then**
  return $(true, CP)$
else
  return $GetElement(\{(true, CP_{tmp}), (false, CP \wedge \neg CP_{tmp})\})$
end if

sageMathでの実装

- CheckConsistencyIP
  - 同様の例題を動かす
  - １回目のIP
    - CP = 10 <= py <= 11
    - S = {y = py, y’ = 10, y’’ = -10}

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

- 入力
  - 微分方程式の集合 [-10 = diff(dy,1), dy= diff(y ,1)]
    - diff(y,1): yの１階微分　diff(dy,1): dyの１階微分
    - sageMathでは2階微分を含む微分方程式を解くことができないため, 中間変数を用意する.
  - 関数の初期値の集合 [py, 10]
  - 解きたい関数の集合 [y, dy]
- 出力として時刻の式の集合 [y(t) == -5t^2 + 10t + p_y, dy(t) = -10*t + 10]

sageMathでの実装

- 例
  - 最初のIP
    - $S = \{y = py,\ y' = 10,\ y'' = -10\}$
    - S内の微分方程式を解くと
      S_t = {y = py + 10t – 5t^2, y' = 10 - 10t, y'' = -10}
    - CP = 9 <= py <= 11
    - CP_tmp = 9 <= py <= 11
    - CP_tmp = CP なので
      (true, 9 <= py <= 11)

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

Input: 制約ストア S, 記号定数の条件 CP
Output: 充足可能性, 記号定数の条件

$S_t := \mathit{SolveDifferentialEquation}(S)$
$V := \mathit{GetVariables}(S_t)$
$CP_{tmp} := \exists V(\mathit{Inf}\{t \mid \exists_t(S_t \wedge t > 0)\} = 0) \wedge CP)$
if $CP_{tmp} = \mathit{false}$ **then**
  return (false, CP)
else if $CP_{tmp} = CP$ **then**
  return (true, CP)
else
  return $\mathit{GetElement}(\{(\mathit{true}, CP_{tmp}),$
    $(\mathit{false}, CP \wedge \neg CP_{tmp})\})$
end if

sageMathでの実装

- CheckConsistencyIP

```
INIT   <=> 9 <= y <= 11 & y' = 10.
FALL   <=> [](y'' = -10).
BOUNCE <=> [](y- = 15 => y' = -0.8y'-).

INIT, FALL << BOUNCE.
```

- 入力
  - 時刻の式の集合にt=0を代入した集合 [y == p_y, dy = 10, ddy = -10]
    - sageMath で Inf を表現するために t = 0 を代入
    - mathematicaによる実装でも同様の処理を行なっている

```
== Check Consistency IP by QE ==========================================
==== input ======
constr store     :       [[y == p_y, dy == 10, ddy == -10]]
symbolic params  :       [[p_y >= 9, p_y <= 11]]
[ddy, dy, y] [y = p_y /\ dy = 10 /\ ddy = -10 /\ p_y >= 9 /\ p_y <= 11]
# tmp_parameters is modified
==== output =====
consistency      :       [True, False]
updated params   :       [[py - 9 >= 0 /\ py - 11 <= 0], []]
==== measure ====
time             :       4.315759999999997
```