ハイブリッドシステムの記号的計算による解析

平石 邦彦

北陸先端科学技術大学院大学・情報科学研究科

内容

1. 形式的検証－モデル検査
2. オートマトンからハイブリッドオートマトンへ
3. ハイブリッドオートマトンの離散的抽象化
4. 記号的計算によるハイブリッドオートマトンの設計

内容

1. 形式的検証－モデル検査
2. オートマトンからハイブリッドオートマトンへ
3. ハイブリッドオートマトンの離散的抽象化
4. 記号的計算によるハイブリッドオートマトンの設計

形式的検証

- 目的：設計(design)の正しさを可能な限り初期の段階で保証する．
- 伝統的手法：シミュレーションとテスト．
  - すべてを調べ尽くしたのか？
  - どのぐらいのバグがまだ残っているか？
- 形式的検証(formal verification)
  - すべての“ふるまい”の網羅的検査．
  - 定理証明(theorem proving)，モデル検査(model checking)．

モデル検査

- モデル検査は，
  - 完全に自動的な方法であり，ユーザーの介入，補助は必要ない．必要な入力データを与え，モデル検査アルゴリズムを実行させれば，後は結果が出力されるのを待てばよい．

    → 定理証明はその実行過程において，専門的な知識を持つユーザーの介入が必要である．
  - 設計が正しくない場合は，反例を出力してくれる．それに基づきデバッグできる．

モデル検査の手順

1. モデル化：設計をモデル検査ツールが受け入れられる形式に変換する．場合によっては，簡略化，抽象化が必要．
2. 仕様記述：設計が満たすべき性質をツールが解釈できる形式で記述する．時相論理を用いる．
3. 検証：設計のモデルが仕様を満たしているかどうかを調べる．満たしていない場合には，誤ったふるまいを出力する．

モデル検査の手順

1. モデル化：設計をモデル検査ツールが受け入れられる形式に変換する．場合によっては，簡略化，抽象化が必要．
2. 仕様記述：設計が満たすべき性質をツールが解釈できる形式で記述する．時相論理を用いる．
3. 検証：設計のモデルが仕様を満たしているかどうかを調べる．満たしていない場合には，誤ったふるまいを出力する．

例：並行プログラム－ソースコード

P = m: **cogbegin** P_0 || P_1 **coend** m'.

P_0 :: l_0: **while** *True* **do**

NC_0: **wait**(*turn* = 0);

CR_0: *turn* := 1;

end while

l_0'.

P_1 :: l_1: **while** *True* **do**

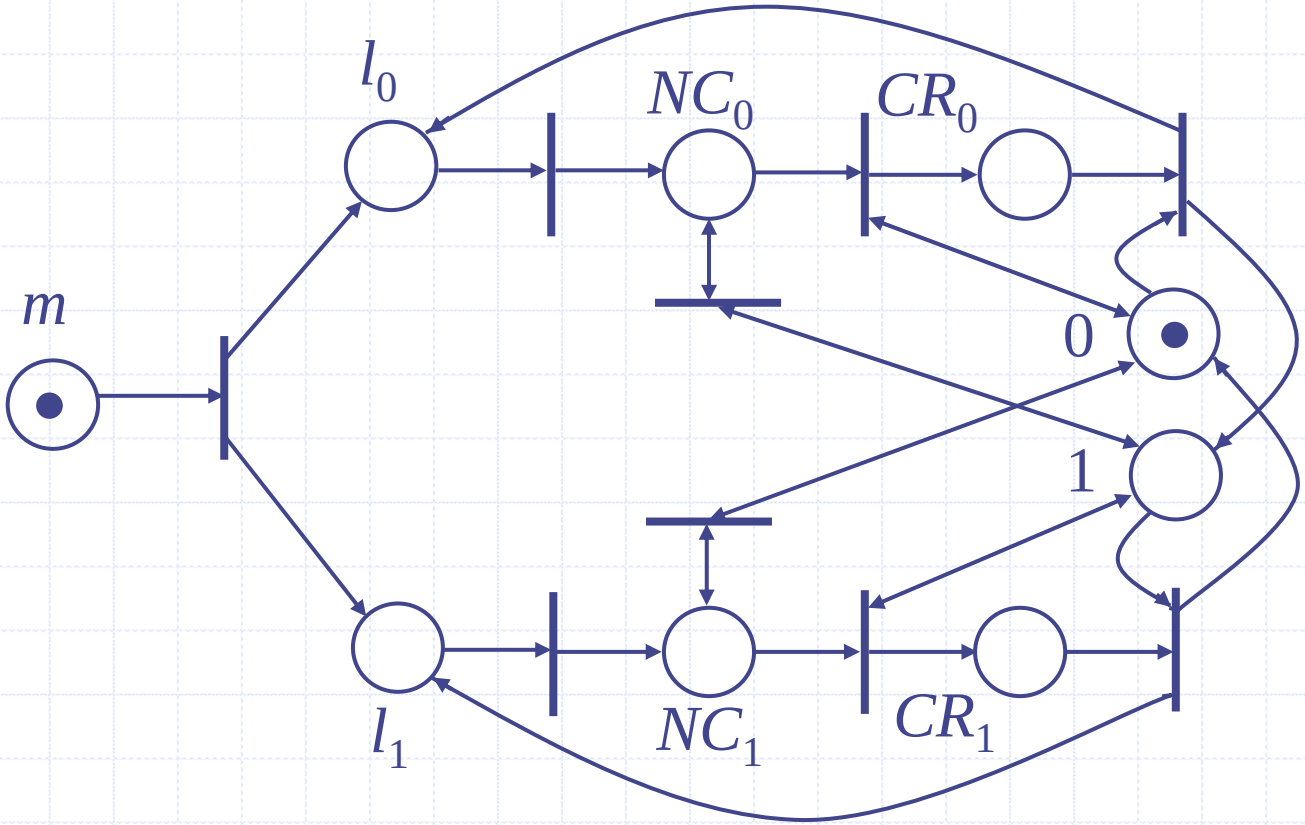
NC_1: **wait**(*turn* = 1);

CR_1: *turn* := 0;

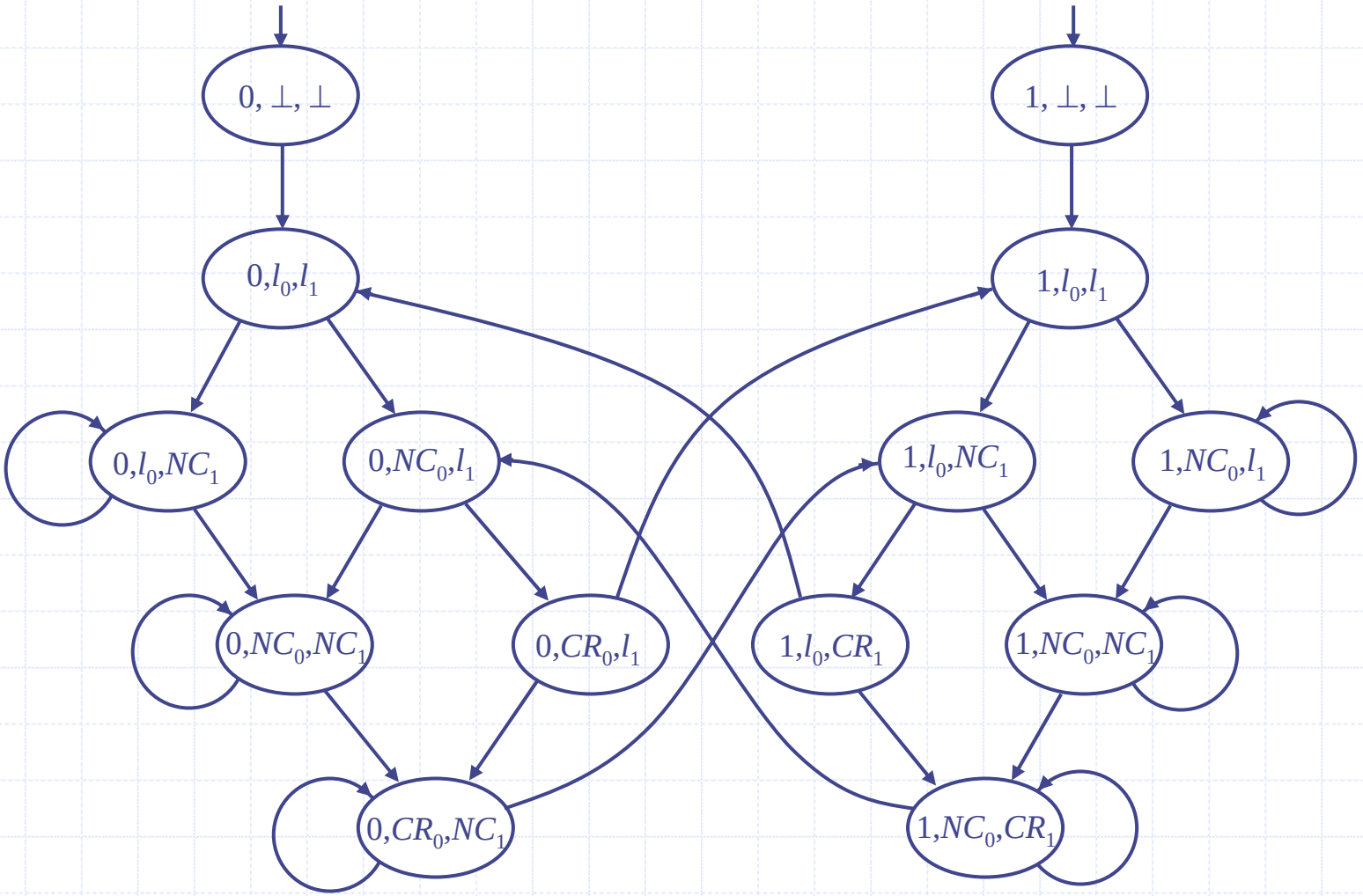
end while

l_1'.

例：並行プログラム－ペトリネット



例：並行プログラム ー オートマトン (クリプケ構造)



モデル検査の手順

1. モデル化：設計をモデル検査ツールが受け入れられる形式に変換する．場合によっては，簡略化，抽象化が必要．
2. **仕様記述：設計が満たすべき性質をツールが解釈できる形式で記述する．時相論理を用いる．**
3. 検証：設計のモデルが仕様を満たしているかどうかを調べる．満たしていない場合には，誤ったふるまいを出力する．

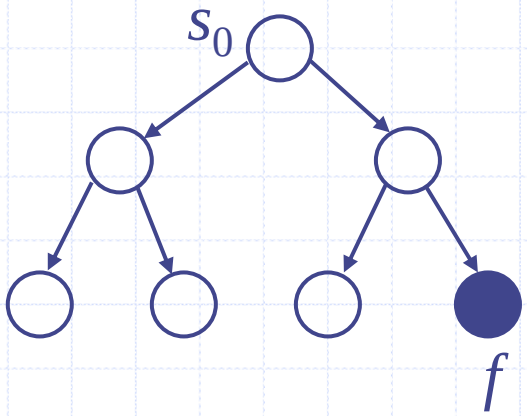
時相論理

- 時相論理(temporal logic)は離散状態システムにおける状態遷移の系列に対する性質の記述を行うための形式化である.
- 命題の真理値は時間軸上で変化してもよい. これにより, 以下のような性質を記述できる.
  - 望ましい状態に“将来”到達できる.
  - エラー状態に到達することはない.

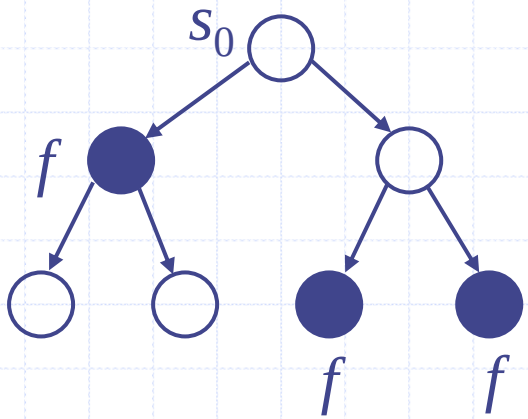
CTL(Computation Tree Logic) (1)

- **計算パス：状態遷移の無限系列** $s_0, s_1, s_2, \ldots$ **．**
- **パスオペレータ：**
  - E g : g **が（現在の状態からの）ある計算パスに対して真．**
  - A g : g **が（現在の状態からの）すべての計算パスに対して真．**
- **状態オペレータ：**
  - X f : f **が次の状態で真**(neXt)**．**
  - F f : f **がパス上のある状態で真** (Future).
  - G f : f **がパス上のすべての状態で真** (Globally).
  - f_1 U f_2 : f_2 **が真になるまではずっと** f_1 **が真**(Until).

CTL(Computation Tree Logic) (2)

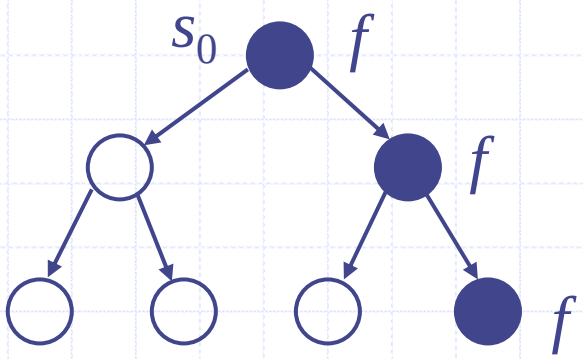


$<M, s_0> \models \mathrm{EF}\ f$

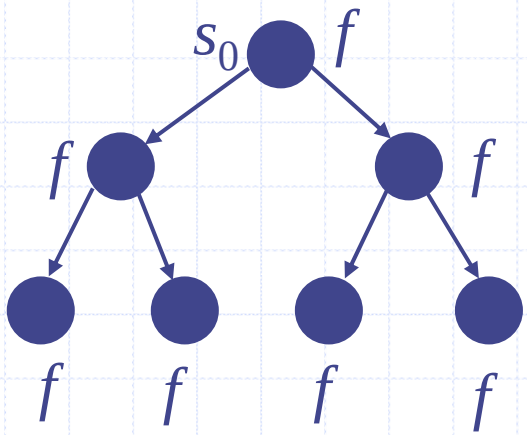


$<M, s_0> \models \mathrm{AF}\ f$

CTL(Computation Tree Logic) (3)



$<M, s_0> \models \mathrm{EG}\ f$



$<M, s_0> \models \mathrm{AG}\ f$

例：並行プログラム－*仕様*

- セーフティ（“悪いこと”は起こらない）.

$$\mathsf{AG}\neg(pc_0 = CR_0 \wedge pc_1 = CR_1)$$

- ライブネス（“良いこと”がいずれは起こる）.

$$\mathsf{AG}(pc_0 = NC_0 \rightarrow \mathsf{AF}(pc_0 = CR_0)) \wedge$$
$$\mathsf{AG}(pc_1 = NC_1 \rightarrow \mathsf{AF}(pc_1 = CR_1))$$

モデル検査の手順

1. モデル化：設計をモデル検査ツールが受け入れられる形式に変換する．場合によっては，簡略化，抽象化が必要．
2. 仕様記述：設計が満たすべき性質をツールが解釈できる形式で記述する．時相論理を用いる．
3. **検証：設計のモデルが仕様を満たしているかどうかを調べる．満たしていない場合には，誤ったふるまいを出力する．**

記号モデル検査アルゴリズム (1)

- 記号モデル検査は，与えられた時相論理式を真にする状態集合を論理式の形で計算していく手法である．
  - 原子命題: 与えられている．
  - EX f := $\exists \mathbf{x}'[f(\mathbf{x}') \wedge \mathbf{x} \rightarrow \mathbf{x}']$．ここで，$\mathbf{x}, \mathbf{x}'$ は状態変数ベクトルであり，$\rightarrow$ は1ステップの状態遷移関係．
  - E[f_1 U f_2] := 以下を満たすZの最小不動点

$$Z = f_2 \vee (f_1 \wedge \mathrm{EX}\ Z).$$

  - EG f :=以下を満たすZの最大不動点

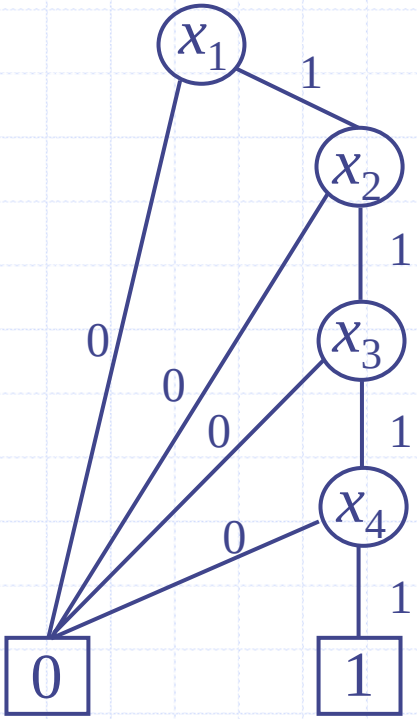
$$Z = f \wedge \mathrm{EX}\ Z.$$

  - 他のオペレータは上記のオペレータで表現できる．（例：EF f = E[$true$ U f], AG $f \equiv \neg\mathrm{EF}\neg f$）．

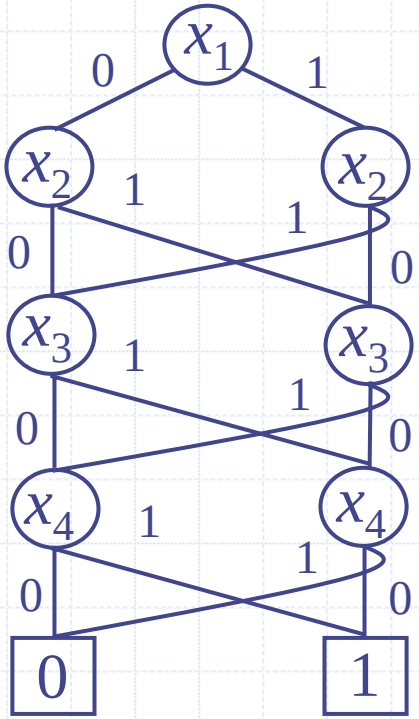
記号モデル検査アルゴリズム (2)

- $Z = f_2 \vee (f_1 \wedge \mathrm{EX}\, Z)$ **の計算：**
  - $Z_0 := \mathit{false}$;
  - $Z_{i+1} := f_2 \vee (f_1 \wedge \mathrm{EX}\, Z_i)$.
  - **もし，** $Z_{i+1} = Z_i$ **ならば停止．**
- $Z = f \wedge \mathrm{EX}\, Z$ **の計算：**
  - $Z_0 := \mathit{true}$;
  - $Z_{i+1} := f \wedge \mathrm{EX}\, Z_i$.
  - **もし，** $Z_{i+1} = Z_i$ **ならば停止．**
- **すべて論理演算→2分決定グラフによる効率的計算が可能．**

記号モデル検査アルゴリズム (3)



$x_1 \wedge x_2 \wedge x_3 \wedge x_4$



$x_1 \oplus x_2 \oplus x_3 \oplus x_4$

Reduced Ordered Binary Decision Diagram

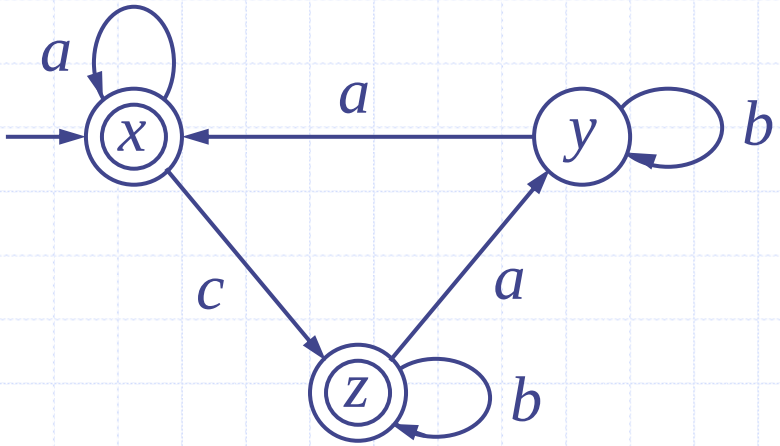
記号モデル検査アルゴリズム (4)

- **ツール**：SMV(Symbolic Model Verifier)
  http://www-2.cs.cmu.edu/~modelcheck/smv.html

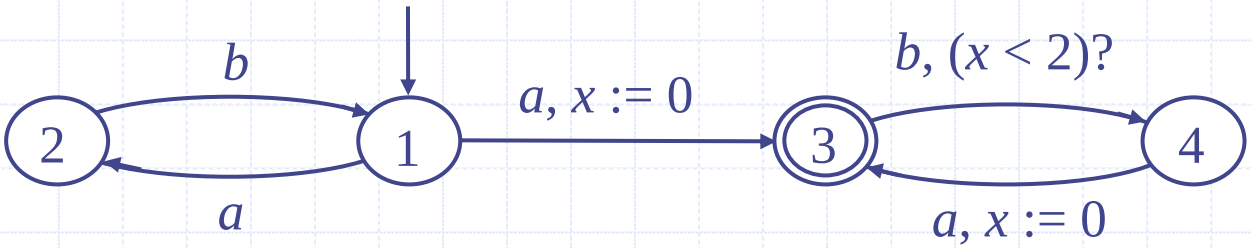
内容

1. 形式的検証－モデル検査
2. **オートマトンからハイブリッドオートマトンへ**
3. ハイブリッドオートマトンの離散的抽象化
4. 記号的計算によるハイブリッドオートマトンの設計

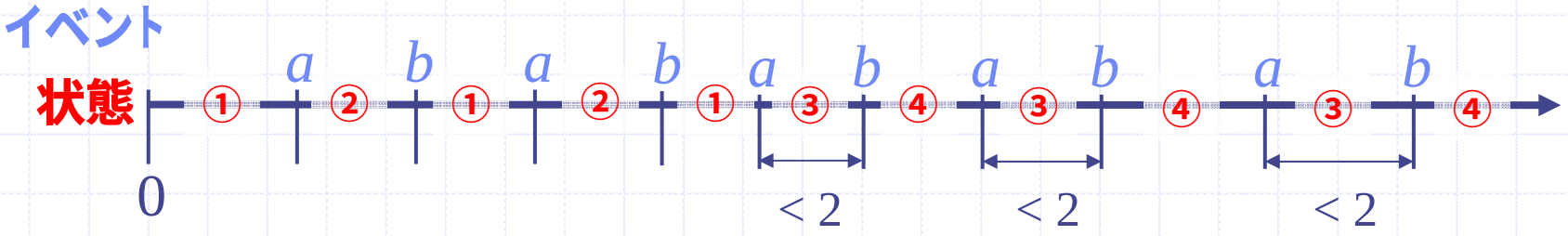
オートマトン



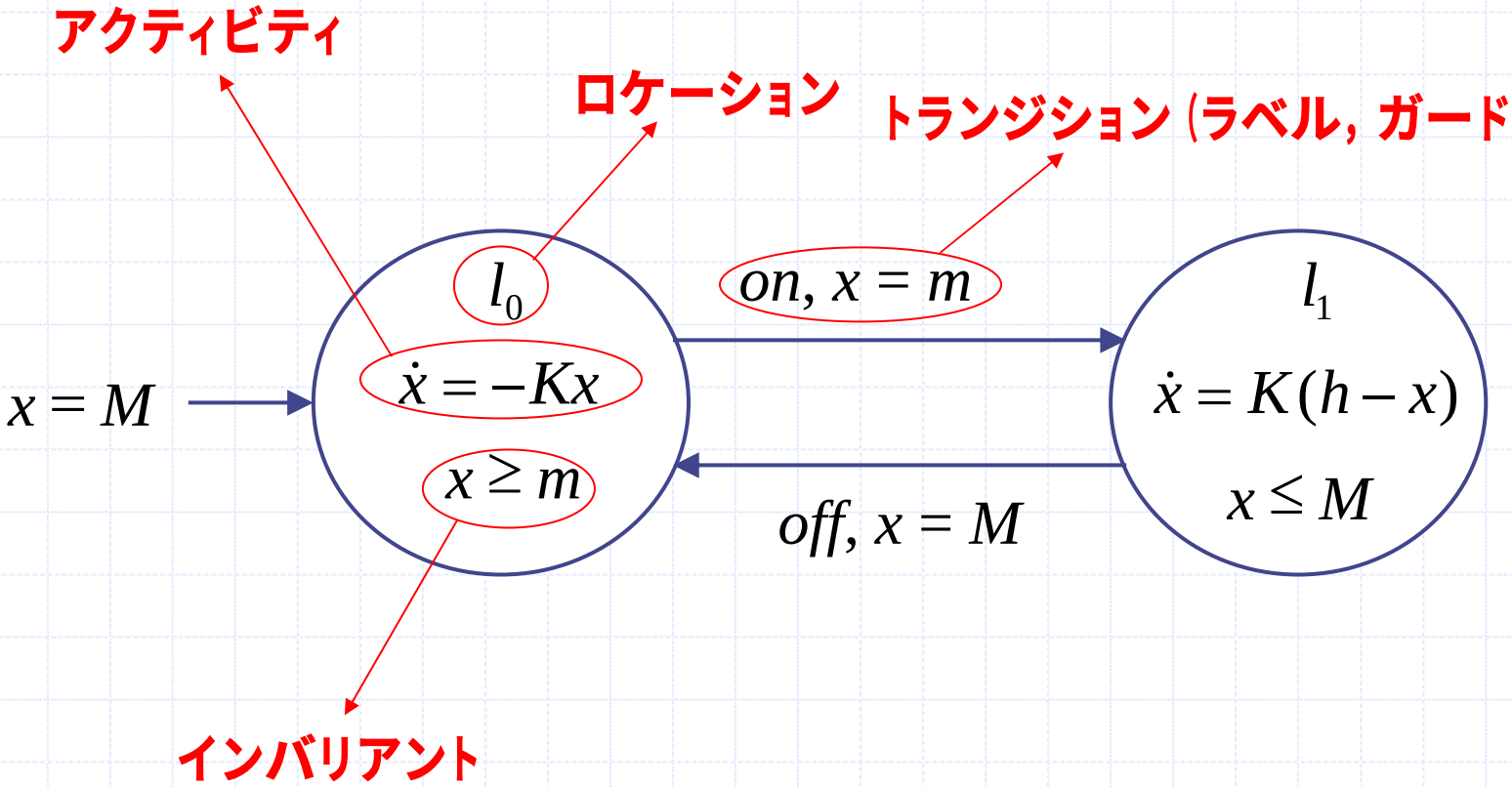
時間オートマトン



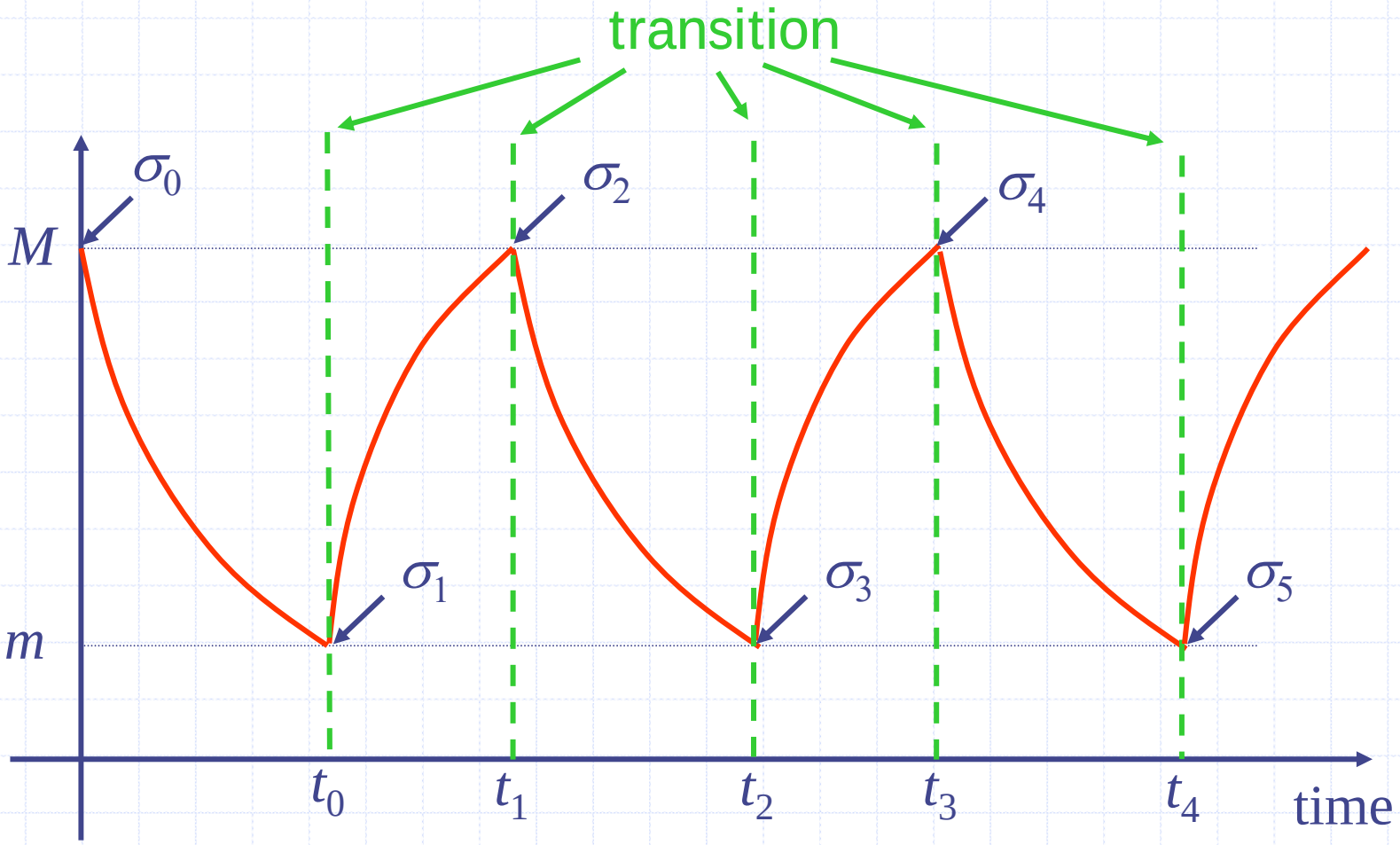
x: クロック



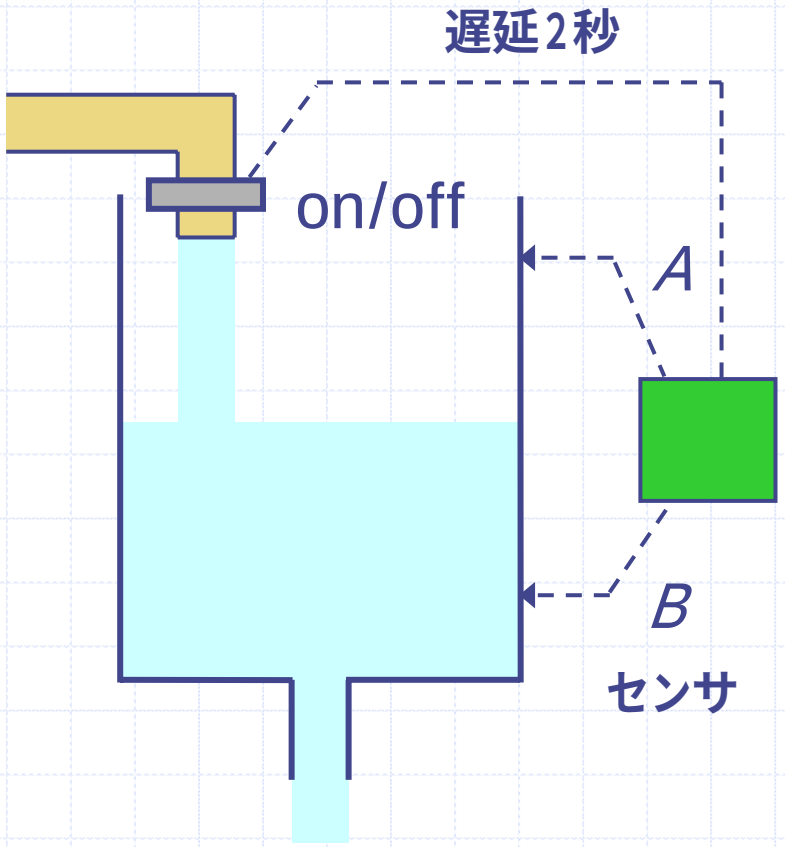
ハイブリッドオートマトン：サーモスタット (1)



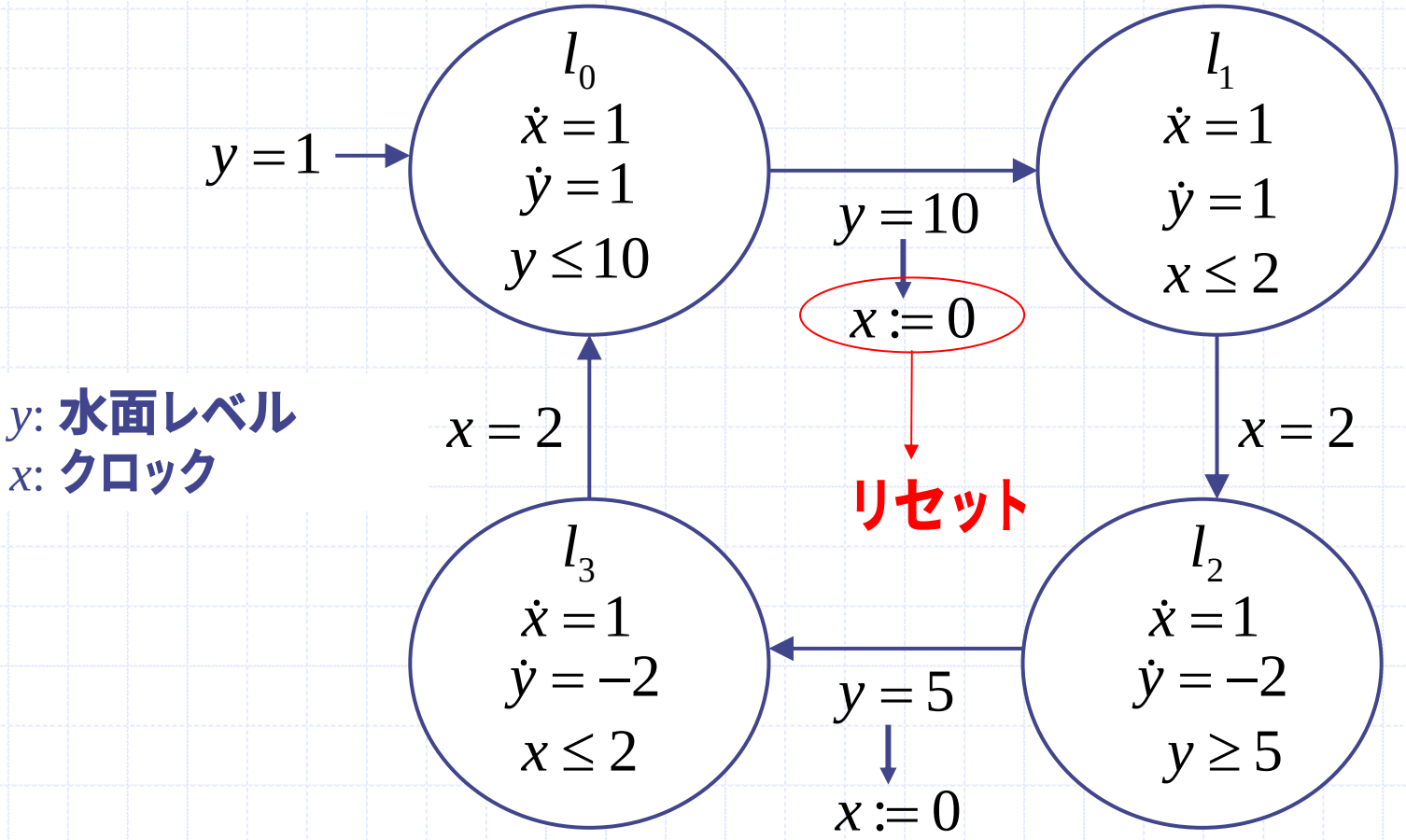
ハイブリッドオートマトン：*サーモスタット (2)*



線形ハイブリッドオートマトン：
水面レベルモニタ (1)



線形ハイブリッドオートマトン：
水面レベルモニタ (2)



y: **水面レベル**
x: **クロック**

内容

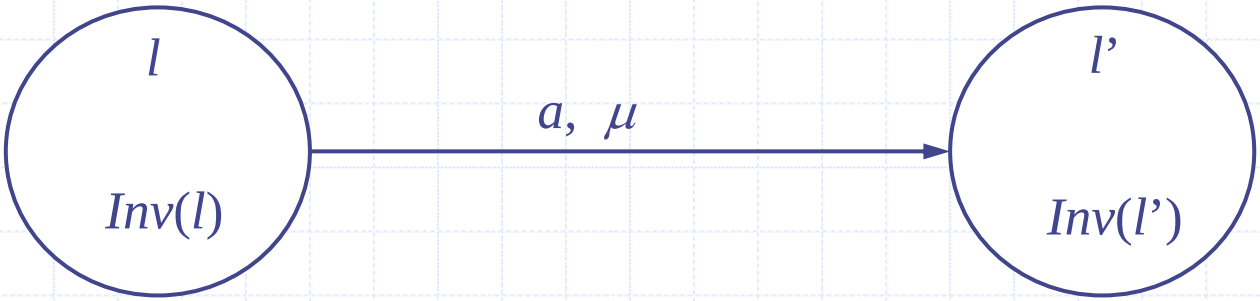
1. 形式的検証－モデル検査
2. オートマトンからハイブリッドオートマトンへ
3. **ハイブリッドオートマトンの離散的抽象化**
4. 記号的計算によるハイブリッドオートマトンの設計

ハイブリッドオートマトンのふるまい (1)

- ハイブリッドシステムの状態は (l, v) で表される．ここで，
  - l はロケーション，
  - v は各変数へ値を割り当てる関数．
- ハイブリッドオートマトン H のふるまいは，ラベル付遷移システム $LTS_H = (Q, \Sigma \cup \mathbf{R}^{\geq 0}, \rightarrow, Q_0)$ により与えられる．ここで，
  - Q は状態の集合，
  - Σ はラベルの集合，$Q_0 \subseteq Q$ は初期状態の集合，
  - 遷移関係 $\rightarrow$ は遷移ステップ関係と時間ステップ関係の2つからなる．

ハイブリッドオートマトンのふるまい (2)

遷移ステップ関係

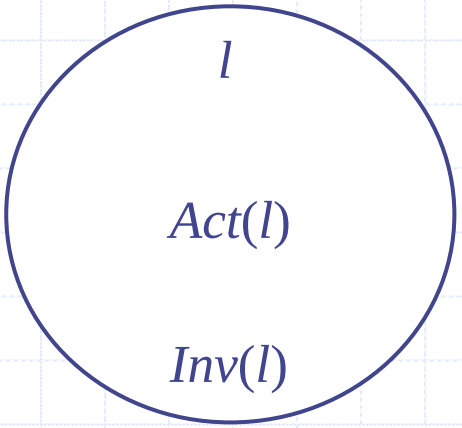


$$v \in Inv(l) \wedge v' \in Inv(l') \wedge (v, v') \in \mu$$

$$(l, v) \xrightarrow{a} (l', v')$$

ハイブリッドオートマトンのふるまい (3)

時間ステップ関係



$$f \in Act(l) \wedge f(0) = v \wedge \forall 0 \leq t' \leq t: f(t') \in Inv(l)$$

$$(l, v) \xrightarrow{t} (l, f(t))$$

離散的抽象化の必要性

- ハイブリッドオートマトン H から作られるラベル付遷移システム LTS_H は一般に非可算無限個の状態をもつ．また，1つの状態の次状態も一般に非可算無限個存在する．
- モデル検査アルゴリズムの対象は有限状態システムである．
- モデル検査アルゴリズムをハイブリッドオートマトンの検証に用いるために，LTS_H を与えられた時相論理式に関して等価な離散状態のラベル付遷移システムに変換する．

双模倣 (1)

- $LTS = (Q, \Sigma, \rightarrow, Q_0)$ をラベル付遷移システムとする．L を各状態に対し，そこで真となる原子命題（オペレータを含まない式）の集合を割り当てる関数とする．
- $\sim_L$ を状態の集合上の同値関係で以下を満たすものとする：任意の状態 $p, q \in Q$ および原子命題 a に対し，もし $p \sim_L q \wedge a \in L(p)$ ならば $a \in L(q)$.
- **定義．** 同値関係 $\sim_L$ は以下を満たすとき LTS の双模倣(bisimulation)であるという：$p \sim_L q$ であるとき，もし $p \rightarrow p'$ ならば，ある状態 $q' \in Q$ が存在して $q \rightarrow q' \wedge p' \sim_L q'$．
- **定理．** $\sim_L$ を TS の双模倣とする．任意のCTL論理式 f は，LTS において真ならば，かつそのときに限り，$LTS/\sim_L$ で真である．ここで，$LTS/\sim_L$ は LTS の同値関係 $\sim_L$ による商遷移システムを表す．

双模倣 (2)

Bisimulation Algorithm:

Initially $\pi := \{\ [a] \mid a \in AP\ \}$;

　　/* AP**は原子命題の集合,** $[a] := \{\ q \in Q \mid a \in L(q)\ \}$ */

while $\exists R, R' \in \pi : \emptyset \neq R \cap Pre(R') \neq R$ **do**

　　/* $Pre(R) := \{\ q \in Q \mid \exists p \in R: q \rightarrow p\ \}$ */

　　$R_1 := R \cap Pre(R')$;

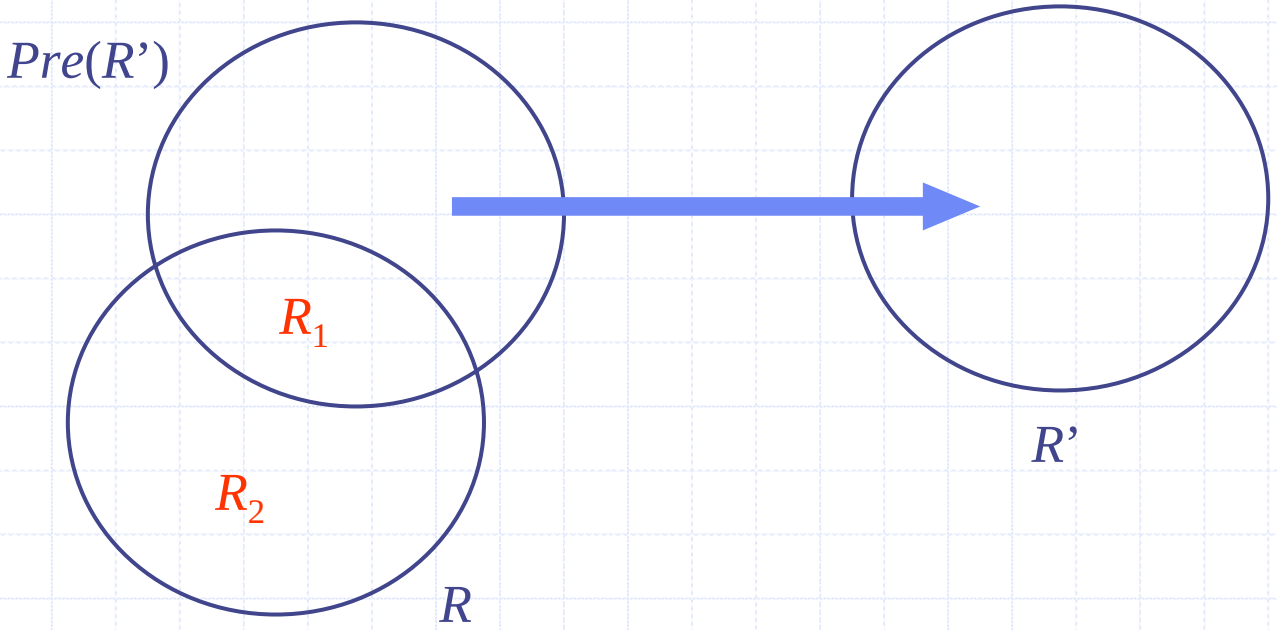
　　$R_2 := R - Pre(R')$;

　　$\pi := (\pi - \{\ R\ \}) \cup \{\ R_1, R_2\ \}$;

endwhile

Output π.

双模做 (3)



例題：水面レベルモニタ (1)

- $AP = \{ l_0, l_1, l_2, l_3, 1 \leq y \leq 12 \}$.
- $\pi_0 = \{$

  $\psi_{00} \equiv (l = 0 \wedge 1 \leq y \leq 12)$,

  $\psi_{10} \equiv (l = 1 \wedge 1 \leq y \leq 12)$,

  $\psi_{20} \equiv (l = 2 \wedge 1 \leq y \leq 12)$,

  $\psi_{30} \equiv (l = 3 \wedge 1 \leq y \leq 12)$,

  $\psi_{01} \equiv (l = 0 \wedge (0 \leq y < 1 \vee y > 12))$,

  $\psi_{11} \equiv (l = 1 \wedge (0 \leq y < 1 \vee y > 12))$,

  $\psi_{21} \equiv (l = 2 \wedge (0 \leq y < 1 \vee y > 12))$,

  $\psi_{31} \equiv (l = 3 \wedge (0 \leq y < 1 \vee y > 12))$

  $\}$.

例題：水面レベルモニタ (2)

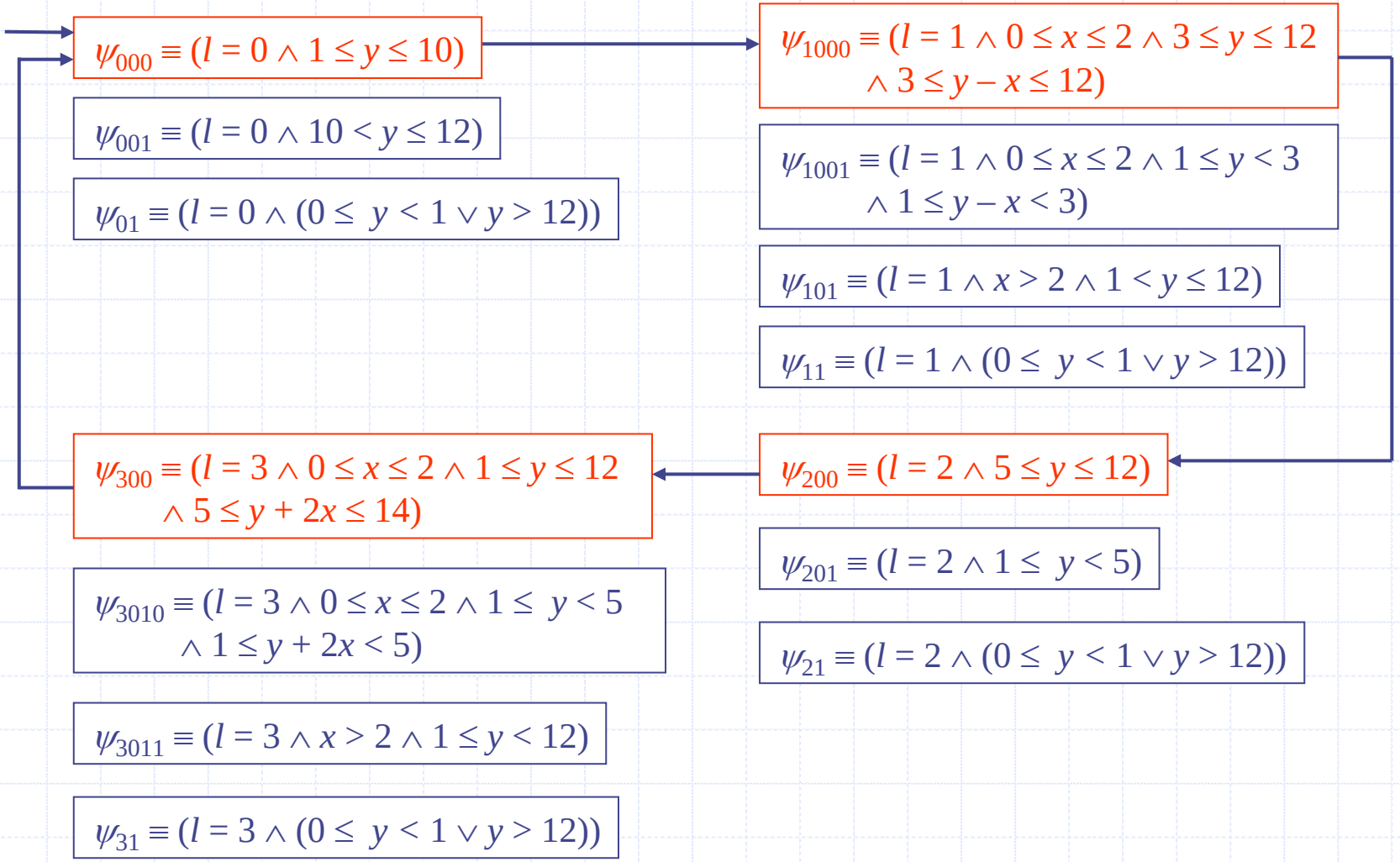
- $\psi_{00} \rightarrow \psi_{10} : \psi_{00}$ を $\psi_{000} \equiv (l = 0 \wedge 1 \leq y \leq 10)$ と $\psi_{001} \equiv (l = 0 \wedge 10 < y \leq 12)$ に分割. $\pi_1 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{10}, \psi_{11}, \psi_{20}, \psi_{21}, \psi_{30}, \psi_{31} \}$.
- $\psi_{10} \rightarrow \psi_{20} : \psi_{10}$ を $\psi_{100} \equiv (l = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12)$ と $\psi_{101} \equiv (l = 1 \wedge x > 2 \wedge 1 < y \leq 12)$ に分割. $\pi_2 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{100}, \psi_{101}, \psi_{11}, \psi_{20}, \psi_{21}, \psi_{30}, \psi_{31} \}$.
- $\psi_{20} \rightarrow \psi_{30} : \psi_{20}$ を $\psi_{200} \equiv (l = 2 \wedge 5 \leq y \leq 12)$ と $\psi_{201} \equiv (l = 2 \wedge 1 \leq y < 5)$ に分割. $\pi_3 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{100}, \psi_{101}, \psi_{11}, \psi_{200}, \psi_{201}, \psi_{21}, \psi_{30}, \psi_{31} \}$.
- $\psi_{100} \rightarrow \psi_{200} : \psi_{100}$ を $\psi_{1000} \equiv (l = 1 \wedge 0 \leq x \leq 2 \wedge 3 \leq y \leq 12 \wedge 3 \leq y - x \leq 12)$ と $\psi_{1001} \equiv (l = 1 \wedge 0 \leq x \leq 2 \wedge 1 \leq y < 3 \wedge 1 \leq y - x < 3)$ に分割. $\pi_4 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{1000}, \psi_{1001}, \psi_{101}, \psi_{11}, \psi_{200}, \psi_{201}, \psi_{21}, \psi_{30}, \psi_{31} \}$.

例題：水面レベルモニタ (3)

- $\psi_{30} \rightarrow \psi_{000} : \psi_{30}$ を $\psi_{300} \equiv (l = 3 \wedge 0 \leq x \leq 2 \wedge 1 \leq y \leq 12 \wedge 5 \leq y + 2x \leq 14)$ と $\psi_{301} \equiv (l = 3 \wedge 0 \leq x \leq 2 \wedge 1 \leq y < 5 \wedge 1 \leq y + 2x < 5) \vee (l = 3 \wedge x > 2 \wedge 1 \leq y < 12)$ に分割. $\pi_5 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{1000}, \psi_{1001}, \psi_{101}, \psi_{11}, \psi_{200}, \psi_{201}, \psi_{21}, \psi_{300}, \psi_{301}, \psi_{31} \}$.
- $\psi_{301} \rightarrow \psi_{001} : \psi_{301}$ を $\psi_{3010} \equiv (l = 3 \wedge 0 \leq x \leq 2 \wedge 1 \leq y < 5 \wedge 1 \leq y + 2x < 5)$ と $\psi_{3011} \equiv (l = 3 \wedge x > 2 \wedge 1 \leq y < 12)$ に分割. $\pi_6 := \{ \psi_{000}, \psi_{001}, \psi_{01}, \psi_{1000}, \psi_{1001}, \psi_{101}, \psi_{11}, \psi_{200}, \psi_{201}, \psi_{21}, \psi_{3010}, \psi_{3011}, \psi_{301}, \psi_{31} \}$.

  初期状態から到達可能な集合の双模倣が得られた.

例題：水面レベルモニタ (4)



ツール

- HyTech(The HYbrid TECHnology Tool)
  http://www-cad.eecs.berkeley.edu/~tah/HyTech/

決定不能性の障壁

- 同値類が有限個であるような双模倣が存在するとは限らない．
- したがって，Bisimulation Algorithmの停止性は保証されない．

内容

1. 形式的検証－モデル検査
2. オートマトンからハイブリッドオートマトンへ
3. ハイブリッドオートマトンの離散的抽象化
4. **記号的計算によるハイブリッドオートマトンの設計**

高信頼ハイブリッドシステム構築のために

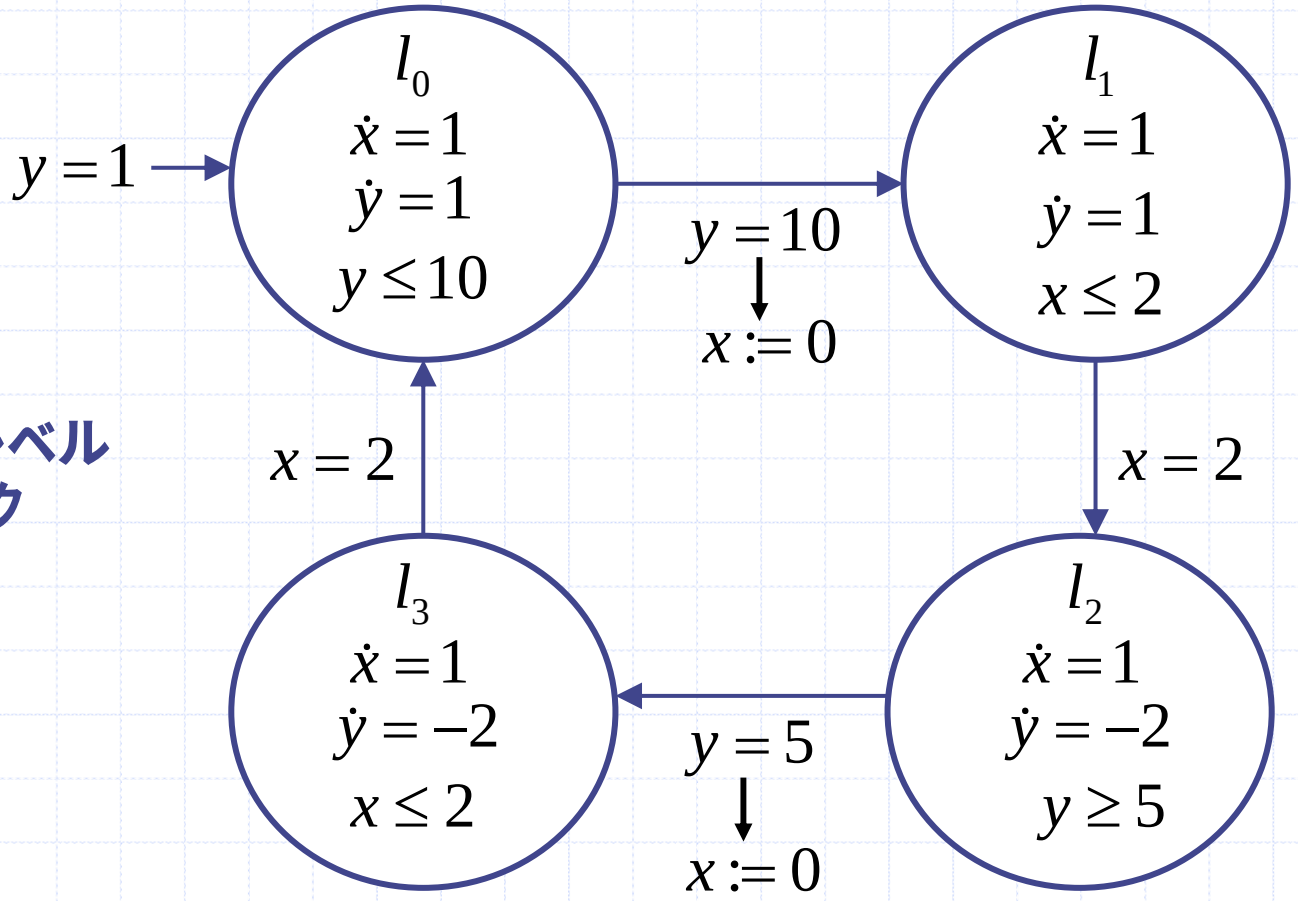
- 数値シミュレーション
- 定性シミュレーション
- モデル検査
- パラメータ設計
- 記号的シミュレーション

→ 制約論理プログラミング

制約論理プログラミング

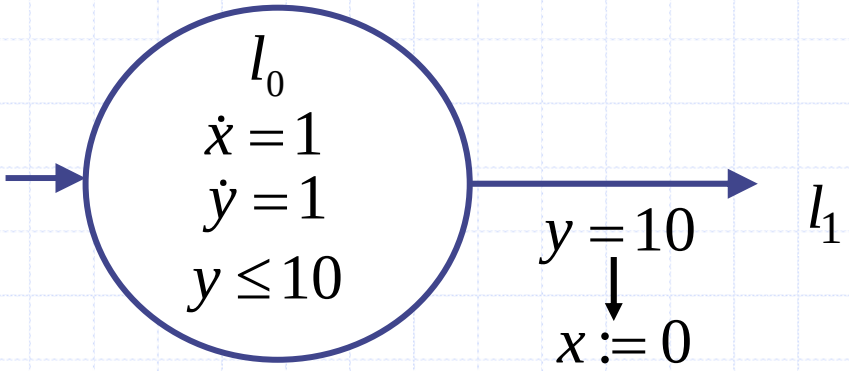
- **制約論理プログラミング**(CLP: Constraint Logic Programming)
  CLP = Prolog + Constraint Solver.
- Keyed CLP for HS =Prolog Interpreter
  - + Linear Constraint Solver
  - + Linear Optimizer
  - + Quadratic Optimizer
  - + Global Variables
  - + etc.

例題：水面レベルモニタ



y: **水面レベル**
x: **クロック**

単純なモデル化 (1)




```
l0(X, Y, Time):-
        X1 = X + D, Y1 = Y + D, D >= 0,
        Y1 = 10,
        l1(0, Y1, Time + D).
```

単純なモデル化 (2)

```
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).
```

単純なモデル化 (3)

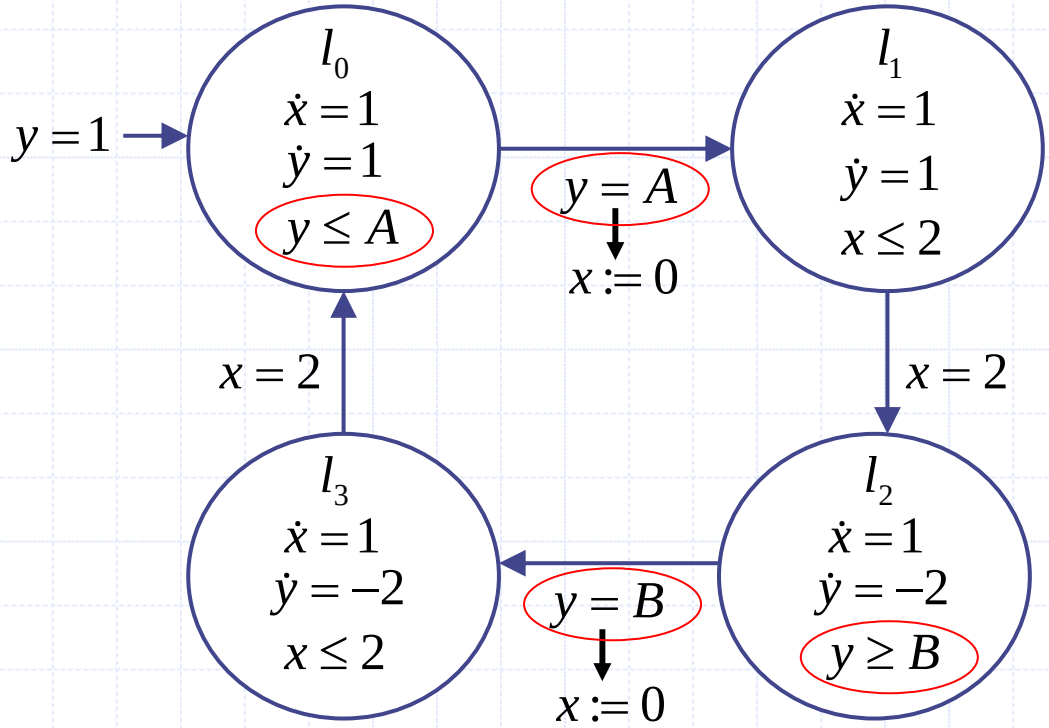
```
(trace)| ?- l0(X, 1, 0).
1-0) CALL : l0(X, 1, 0) ?
1-0) TRY  : l0(X, 1, 0) :- _5 = X + _6, _8 = 1 + _6, _6 >= 0, _8 = _10, l1(0, _8, 0 + _6)
1-0) SUC  : l0(X, 1, 0) :- _5 = X + _6, _8 = 1 + _6, _6 >= 0, _8 = _10, l1(0, _8, 0 + _6)
1-1) CONSTRAINT : _5 = X + _6
1-1) SUC  : _5 = X + _6
1-1) CONSTRAINT : _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  : _8 = _10
1-1) CALL : l1(0, _8, 0 + _6) ?
1-1) TRY  : l1(0, _14, _15) :- _16 = 0 + _17, _19 = _14 + _17, _17 >= 0, _16 = 2, l2(_16, _19, _15 + _17)
1-1) SUC  : l1(0, _14, _15) :- _16 = 0 + _17, _19 = _14 + _17, _17 >= 0, _16 = 2, l2(_16, _19, _15 + _17)
1-2) CONSTRAINT : _16 = 0 + _17
1-2) SUC  : _16 = 0 + _17
1-2) CONSTRAINT : _19 = _14 + _17
1-2) SUC  : _19 = _14 + _17
1-2) CONSTRAINT : _17 >= 0
1-2) SUC  : _17 >= 0
1-2) CONSTRAINT : _16 = 2
>>LIN(1) : 0  = -2  + _17
1-2) SUC  : 2 = 2
1-2) CALL : l2(2, _19, _15 + _17) ?
```

単純なモデル化 (4)

- ハイブリッドオートマトンの記述とCLPの記述は1対1に対応する.
- CLPの実行過程はハイブリッドオートマトンにおけるロケーション間の遷移に対応する. すなわち, CLPはハイブリッドオートマトンの動作を記号的にシミュレートする.
- ハイブリッドオートマトンの各変数はその値が時刻とともに変化する時間の関数である. 論理プログラミングでは変数の値は1つの実行パスに対して一意なので, 各時刻ごとに別の変数を用意しなければならない.
  - たとえば, l1(X, Y, Time) の変数X, Yは時刻Timeでロケーション1に入るときの変数 x, y の値を表す.
- このハイブリッドオートマトンの実行は停止しない. CLPの実行も停止しない.

パラメータ設計 (1)

- 問題：水面レベルモニタにおいて，水面の高さ y を $1 \le y \le 12$ に保つようにセンサの位置 A, B を定めよ．



パラメータ設計 (2)

- 3回目以降のふるまいは2回目と同じであり調べる必要はない.
  - ロケーション1に入るときの各変数の値を見ると,
  - 変数xは最初は未定，2回目以降は $x = 2$ であるが, ロケーション1から2への遷移のガードには x は出現せず，また $x := 0$ にリセットされるので, x の値は任意で構わない.
  - 変数 y については，最初は1であり，2回目以降はパラメータBに依存する値$(y = B - 4)$である.

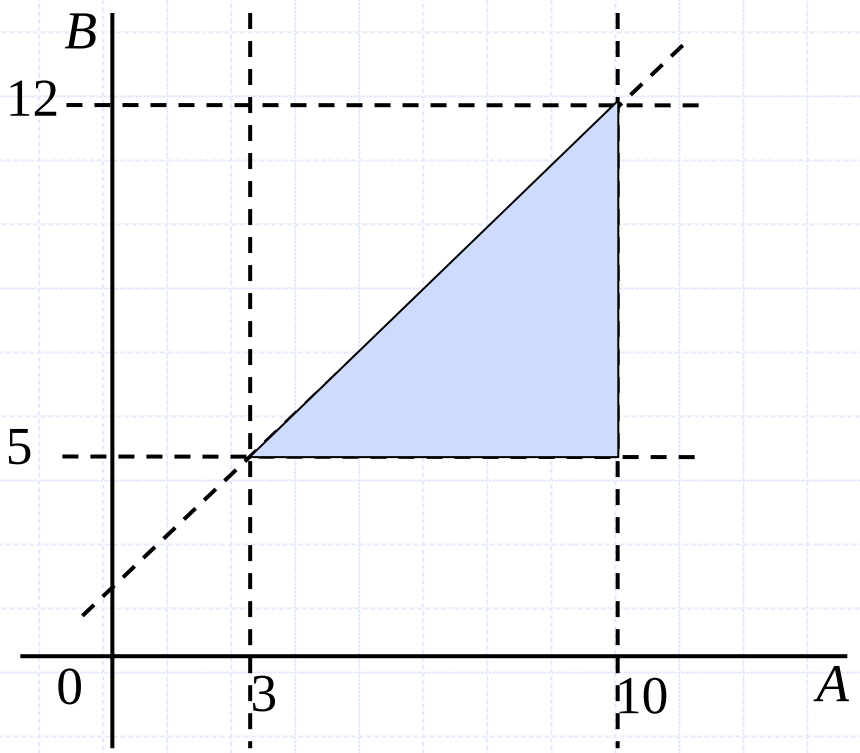
パラメータ設計 (3)

```
l0(X, Y, T, TT, [A, B]):-
        X1 = X + D, Y1 = Y + D, D >= 0,
        Y1 = A,
        spec(Y1),
        l1(0, Y1, T + D, TT, [A, B]).
・・・

l3(X, Y, T, TT, [A, B]):-
        X1 = X + D, Y1 = Y -2 * D, D >=
        0,
        X1 = 2,
        spec(Y1),
        l0_2(X1, Y1, T + D, TT, [A, B]).
```

```
l0_2(X, Y, T, TT, [A, B]):-
        X1 = X + D, Y1 = Y + D, D >= 0,
        Y1 = A,
        spec(Y1),
        l1_2(0, Y1, T + D, TT, [A, B]).
・・・

l3_2(X, Y, T, T + D, [A, B]):-
        X1 = X + D, Y1 = Y -2 * D, D >=
        0,
        X1 = 2,
        spec(Y1).

spec(Y) :- 1 <= Y, Y <= 12.
```

パラメータ設計 (4)

```
| ?- l0(X, 1, 0, TT, [A, B]).
X = ?
TT = 19 - _36 + 4 * _116
A = 10 - _36
B = 12 - _36 -2 * _116
 -2 * _116 - _36 >= -11
 2 * _116 + _36 >= 0
 -2 * _116 - _36 >= -7
 2 * _116 + _36 >= -4
 -2 * _116 - _36 >= -11
 2 * _116 + _36 >= 0
 -2 * _116 - _36 >= -7
 2 * _116 + _36 >= -4
0 <= _36  <= 9
0 <= _116

*** yes ***
```



パラメータ設計 (5)

- **制約条件を満たすパラメータA, Bの値は図の領域であり無限に存在する.**
- **その中で, ポンプのon/off回数の最小化(= サイクル1回りの時間最大化)を目的関数にしたときの最適解はつぎのように求められる.**

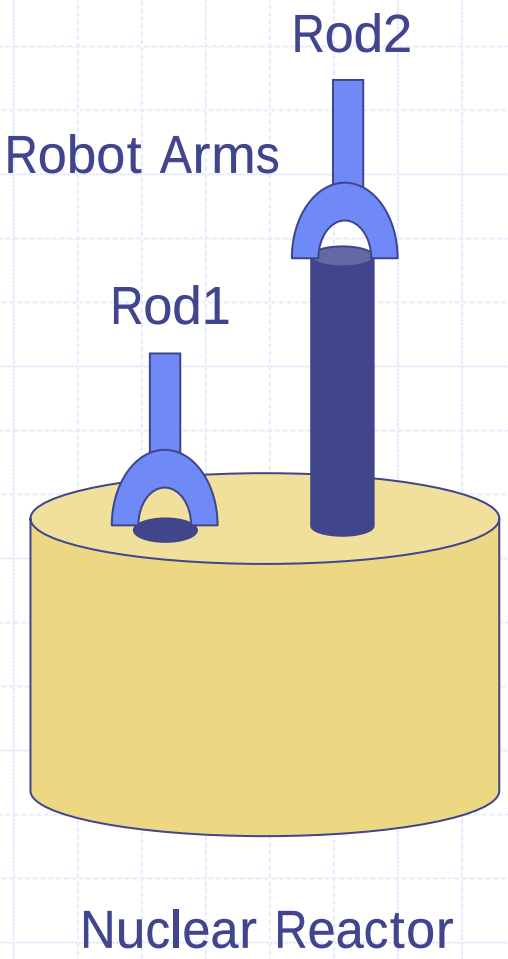
```
| ?- max(TT, l0(X, 1, 0, TT, [A, B])).
TT = 33
X = ?
A = 10
B = 5

*** yes ***
```

パラメータ設計問題：定式化

ハイブリッドシステムのパラメータ設計問題：

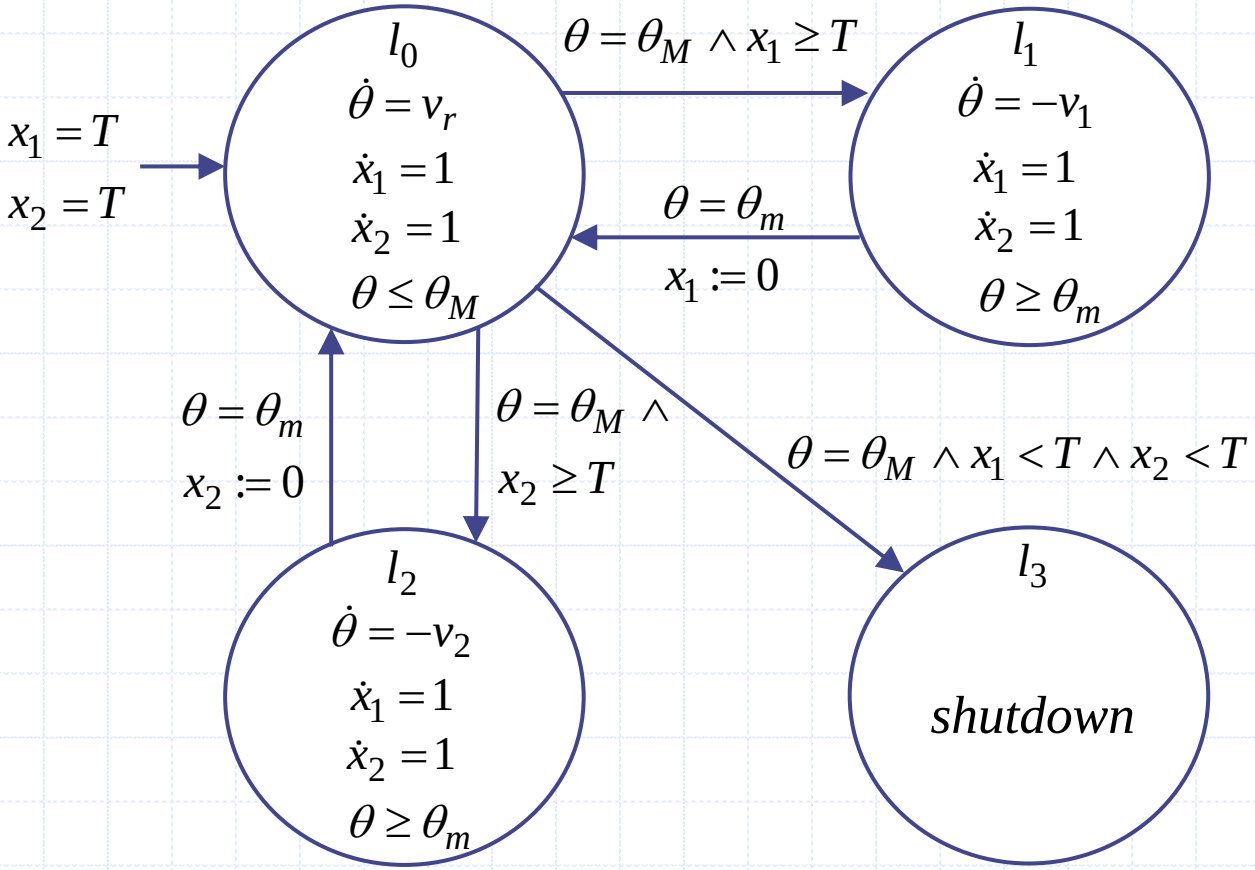
- Given: **パラメータ付きハイブリッドオートマトン，実時間**CTL**により記述された動作仕様．**
- Find: **動作仕様を満足するパラメータ値の集合．**

例題 (1)



- 反応炉の温度を2つの冷却棒で制御する.
- 冷却棒は1度使用してから一定時間(T)は使用できない.
- 制御不能な場合はシャットダウンする.

例題 (2)


l_0
$\dot{\theta} = v_r$
$\dot{x}_1 = 1$
$\dot{x}_2 = 1$
$\theta \le \theta_M$
$x_1 = T$
$x_2 = T$
$\theta = \theta_M \wedge x_1 \ge T$
l_1
$\dot{\theta} = -v_1$
$\dot{x}_1 = 1$
$\dot{x}_2 = 1$
$\theta \ge \theta_m$
$\theta = \theta_m$
$x_1 := 0$
$\theta = \theta_m$
$x_2 := 0$
$\theta = \theta_M \wedge$
$x_2 \ge T$
$\theta = \theta_M \wedge x_1 < T \wedge x_2 < T$
l_2
$\dot{\theta} = -v_2$
$\dot{x}_1 = 1$
$\dot{x}_2 = 1$
$\theta \ge \theta_m$
l_3
shutdown

例題 (3)

- **動作仕様**：$\mathrm{AG}_{\leq L}\,[l = l_1 \rightarrow \mathrm{EF}_{\leq 5}\, l = l_2]$
  時刻 L 以前でロケーションがl_1の任意の状態から，5単位時間内にl_2に遷移することが可能である．
- **問題**：**動作仕様を満たすパラメータ T の値を求めよ．**
- **解法**：**ハイブリッドオートマトンおよびCTL論理式から，パラメータの値を計算するCLPコードを生成する．**

例題 (4)

```
EF_l0(X1, X2, Temp, TT) :-
  D >= 0, vr(: Vr),
  EF_l0_next(X1 + D, X2 + D, Temp + Vr * D, TT + D).
EF_l0_next(X1, X2, Temp, TT) :-
  L(: L), TT <= L, TM(: Temp), T(: T), X1 >= T,
  EF_l1(X1, X2, Temp, TT).
EF_l0_next(X1, X2, Temp, TT) :-
  L(: L), TT <= L, TM(: Temp), T(: T), X2 >= T,
  EF_l2(X1, X2, Temp, TT).
EF_l1(X1, X2, Temp, TT) :-
  D >= 0, v1(: V1),
  EF_l1_next(X1 + D, X2 + D, Temp - V1 * D, TT + D).
EF_l1_next(X1, X2, Temp, TT) :-
  L(: L), TT <= L,
  EF2(X1, X2, Temp).
EF_l1_next(X1, X2, Temp, TT) :-
  L(: L), TT <= L, Tm(: Temp),
  EF_l0(0, X2, Temp, TT).
EF_l2(X1, X2, Temp, TT) :-
  D >= 0, v2(: V2),
  EF_l2_next(X1 + D, X2 + D, Temp - V2 * D, TT + D).
EF_l2_next(X1, X2, Temp, TT) :-
  L(: L), TT <= L,
  Tm(: Temp),
  EF_l0(X1, 0, Temp, TT).
```

```
EF2(X1, X2, Temp):-
  current_state(: [X1, X2, Temp]),
  range([T],
    (T(: T), current_state(: [X1, X2, Temp]),
      EF2_l1(X1, X2, Temp, 0)),
  [TMin], [TMax]),
  T(: T), negate(T, TMin, TMax).
…
negate(T, infeasible, _):- !.
negate(T, _, infeasible):- !.
negate(T, unbounded, unbounded):- !, fail.
negate(T, TMin, unbounded):- !, T < TMin.
negate(T, unbounded, TMax):- !, T > TMax.
negate(T, TMin, TMax):- !, (T > TMax; T < TMin).

go(T, L):-
  L(: L), T(: T), T >= 0,
  vr(: 6), v1(: 4), v2(: 3), TM(: 15), Tm(: 3), Temp(: 9),
  range ([T],
    (T(: T), Temp(: Temp), EF_l0(T, T, Temp, 0)),
  [TMin], [TMax]),
  negate(T, TMin, TMax).
```

例題 (5)

```
| ?- go(T, 30).
T = 7  - _18
0 < _18  <= 7

*** yes ***

| ?-
```

$L = 30$ **のとき,** $0 \leq T < 7$**が可能な** T **の範囲.**

Keyed CLP for HSの目的

- ハイブリッドシステムの設計，検証に必要なプリミティブを実装．
- 問題に応じて，それを解くCLPコードを自動生成．
- 将来的には以下を実装：
  - 非線形制約ソルバー
  - 数式処理アルゴリズム(Quantifier Elimination, …)
  - 計算幾何学アルゴリズム