修 士 論 文

ハイブリットシステムにおける
パラメータ設計

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

石川　礼

2004 年 3 月

修 士 論 文

ハイブリットシステムにおける パラメータ設計

指導教官　平石邦彦 教授

審査委員主査　平石邦彦 教授
審査委員　金子峰雄 教授
審査委員　宮地 充子助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

210003 石川　礼

提出年月: 2004 年 2 月

目 次

第1章　はじめに

ハイブリットシステムとは，連続的，および，離散的状態遷移の両方を持ったシステムである [1]．分散システムや組み込み制御システムの解析や設計に有効であり，多くのアプリケーションに対する数学的モデルとして使われる．その例としては，自動車や自動車運転システム，航空交通管理システム，生産システム，化学プロセス，ロボット工学，プラント制御，リアルタイム通信ネットワーク，そして，リアルタイム回路などがあり，制御理論や理論計算機の部門において非常に多くの研究がなされている．ハイブリットシステムでは，連続系および離散系事象単独では観測できないような複雑な現象が発生することが知られており，そのため，高信頼性ハイブリットを設計するための方法論の構築が要望されている．

ハイブリットシステムにおける従来研究は，システムの設計が与えられた設計仕様を満たしているかどうかを検証するものがほとんどであった．しかし，その逆問題としてのパラメータ設計問題，すなわち，与えられた動作仕様を満たすようにパラメータの値を決定する問題についての研究はそれほど多くない．最適なパラメータの値をアルゴリズム的に計算する手法を確立することにより，試行錯誤的に行われていた設計プロセスを効率化することが可能となる．

本研究では，ハイブリットシステムにおけるパラメータ設計問題を，連続および離散変数上の制約条件を述語とする一階述語論理式を真にする変数の値を決定する問題として定式化する．そして，論理的制約と数値的制約の両方をもつような制約充足問題を解くための処理系である制約論理プログラミング (CLP:Constraint Logic Programing) に着目し，それを用いたパラメータ設計問題の解法について検討する．具体的には，時相論理式により記述されたシステムの動作仕様を与えたとき，それを満たすようなパラメータの値を CLP により計算する方法を提案する．

CLP とは，Prolog に代表される論理プログラミングの各種の制約充足判定を行うソルバーを組み込んだ処理系のことである．CLP を用いる利点として 3 つあげられる [2]．1 つめは, ハイブリットシステムの形式化であるハイブリットオートマトンの実行過程と CLP の操作的意味論の類似性である．しかしながら，ハイブリットシステムを CLP で解析しようとする従来研究はごく少数に限られる [2]，[4]，[5]，[6]．2 つめは，CLP を用いることにより，ハイブリットオートマトンの動作を記号的にシミュレートすることができることである 3 つめは,CLP では, 達成したい目標において，各対象の間にどのような関係が成立するか，ということに着目するだけでよいということである．つまり，プログラムには，各対象の関係を制約という形で記述すればよい．本研究では，実数体上の線形不等式制約

に対するソルバーを備えた CLP 言語である Keyed CLP を用いる．
時相論理 (Temporal logic) は，離散システムの状態遷移に関する様々な性質を記述するために用いられる [3]．時相論理は，同時プログラムの検証における仕様記述に用いることを目的とし，1977 年に Pnueli によって最初に提案されて以来様々なバーションが存在する．本研究では，モデル検査などで用いられる CTL(Computation Tree logic) の各時相オペレータに対し，それが真になるまでの最大時間を指定した論理を用いる．最大時間の指定は，CLP による計算を有限ステップで終了させるために設けた．
本論文の構成は，第 2 章でハイブリットシステムの定義について述べ, 次に線形ハイブリットシステムの例を述べる. 第 3 章では, 制約論理プログラミング (CLP:Constraint Logic Programing) とはどのようなものかを述べ,CLP を用いハイブリットシステムを表現した例を示す．第 4 章では，パラメータ設計問題を CTL(Computation Tree logic) により記述した動作仕様を満足するようなパラメータの値を決定するとして定式化する．まず，各 CTL オペレータに対し，それを満足するパラメータの値をどのように CLP で解くかを示し，さらに，ネストした CTL オペレータに対するパラメータ値の計算法を提案する．

第2章　ハイブリットシステム

2.1　ハイブリットシステムの定義

ハイブリットシステム $H = (Loc, Var, Lab, Edg, Act, Inv)$ は6つの要素からなっている [1].

- ロケーションと呼ばれる頂点の有限集合 Loc.
- 実数値をとる変数の有限集合 Var. その変数に対する付値 v とは，それぞれの変数 $x \in Var$ に実数値 $V(x) \in R$ を割り当てる関数である.
- 同期レベルの有限集合 Lab.　stutter ラベル $\tau \in Lab$ が含まれる.
- 遷移と呼ばれる辺の有限集合 Edg. それぞれの遷移 $e = (l, a, \mu, l')$ はもとのロケーション $l \in Loc$，目的のロケーション $l' \in loc$，同期ラベル $a \in Lab$，そして，遷移関係 $\mu \subseteq V^2$ からなっている．また，それぞれのロケーション $l \in Loc$ に対し，(l, τ, Id, l) という形の stutter 遷移が許される．ここで，$Id = \{(v, v) | v \in V\}$
  状態 (l, v) においてある付値 $v' \in V$ に対して $(v, v') \in \mu$ ならば, 遷移 e は, 生起後継である.
- それぞれのロケーション $l \in Loc$ にアクティビティの集合を割り当てる関数 Act．それぞれアクティビティは非負実数 $R^{\geq 0}$ への関数. それぞれのロケーションのアクティビティは時間に対して不変であるとする．すなわち，すべてのロケーション $l \in Loc$. アクティビティ $f \in Act(l)$, そして非負実数 $t \in R^{\geq 0}$ に対して，$(f+t)(t') \in Act$．ここで，すべての $t' \in R^{\geq 0}$ に対して $(f+t)(t') = f(t+t')$．なお，すべてのロケーション $l \in Loc$，アクティビティ $f \in Act(l)$．そして，変数 $x \in Var$ に対して $f^x(t) = f(t)(x)$ のような $R^{\geq 0}$ から R への関数を f^x と書く.
- 関数 Inv は，それぞれのロケーション $l \in Loc$ にインバリアント $Inv(l) \subseteq V$ を割り当てる.

ハイブリットシステム H が時間決定性であるとは，それぞれのロケーション $l \in Loc$ とそれぞれの付値 $v \in V$ に対して，$f(0) = v$ を満たすような高々一つのアクティビティ $f \in Act(l)$ が存在するときである．このような f を $\varphi_l[v]$ と書く.

2.2 線形ハイブリットシステム

変数の集合Var上の線形項とは，整数係数をもつVarの変数の線形結合である．そして，Var上の線形式とは，Var上の線形項からなる不等式のブール結合である．
時間決定性のハイブリットシステム$H = (Loc, Var, Lab, Edg, Act, Inv)$は，アクティビティ，インバリアント，そして，遷移関係が変数の集合Var上の線形表現で定義されている場合, 線形である．

1. すべてのロケーション$l \in Loc$に対し，アクティビティ$Act(l)$は，$\dot{x} = k_x$という形の微分方程式の集合で定義されており，それぞれの変数に対して微分方程式が与えられ，$k_x \in R$は，定数である．したがって，すべての付値$v \in V$，変数$x \in Var$，そして，非負実数$t \in R^{\geq 0}$に対して，

$$\varphi_l^x[v](t) = v(t) + k_x t$$

   である．k_xを$Act(l, x)$と書き，ロケーションlにおける変数xのレート，または，変化率という．
2. すべてのロケーション$l \in Loc$に対しインバリアント$Inv(l)$はVar上の線形式ψで定義される．すなわち，

$$v = Inv(l) iff v(\psi).$$

3. すべての遷移$e \in Edg$に対し，遷移関係はμは非決定性の割り当てガード付き集合で定義される．

$$\psi \Rightarrow x := [\alpha_x, \beta_x] | x \in Var$$

   これは, 次の条件を意味する．

$$(v, v') \in iff v(\psi) \wedge \forall_x \in Var.\, v(\alpha) \leq v'(x) \leq v(\beta).$$

   ここで，ψは線形式であり，ガードという．α_x，β_xは線形項である，$\alpha_x = \beta_x$の時，遷移eの後の変数xの更新された値を表すために$\mu(e, x) = \alpha_x$と書く．

2.3 線形ハイブリットシステムの例

2.3.1 水面レベルモニタ

タンクの水面は，水面を連続的に計測しポンプのスイッチをon/offを切り替えて制御されている．水面は線形な関数によって変化する．ポンプがoffの時は，変数yで書かれた水面は，毎秒2インチの速さで下降する．また，ポンプがonの時には毎秒1インチの速さで上昇する．初期状態として水面は1インチの高さであり，ポンプはonになっている

と仮定する．そこで，水面を 1 から 12 インチまでの高さに保ちたい．しかし，モニタがポンプの状態を切り替える命令を出してから実際，ポンプが off になるまでに 2 秒間の遅れが生じる．したがって，モニタは水面が 1 インチを下回る前にスイッチが on になるように命令を出し，12 インチに達する前にスイッチを off にするような命令を出さなければならない．
図 2.1 の線形ハイブリットシステムでは，水面が 5 と 10 インチになったときにモニタはポンプに on/off を切り替えるように命令を出す．また，ロケーション 0 と 1 では，ポンプは on で，ロケーション 2 と 3 では off である．クロック x は遅れの時間を表すために使われている．

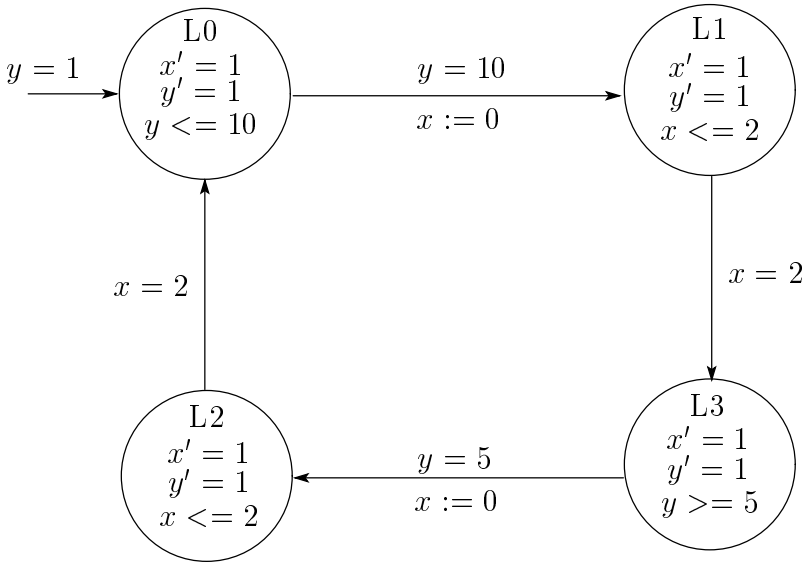

図 2.1: 水面レベルモニタ

2.3.2 温度制御システム

反応タンクの冷却を行うシステムを表したものである．冷却を行うのは 2 つの制御ロッドである．タンクは冷却を行わないと毎秒 v_r だけ温度が上昇する．温度が上限 θ_M に達したらどちらか一方の制御ロッドで冷却を開始する．制御ロッドの選択は非決定的に行われる．2 つの制御ロッドは異なる冷却性能を持ち，それぞれ毎秒 v_1,v_2 だけ温度を下げる，タンクの温度が下限 θ_m に達したら冷却は終了する．各ロッドは使用すると T 秒間は使用できない．もし，θ_m から θ_M の間に温度が制御できない場合はシャットダウンすることになる．
以下に温度制御システムを示す.

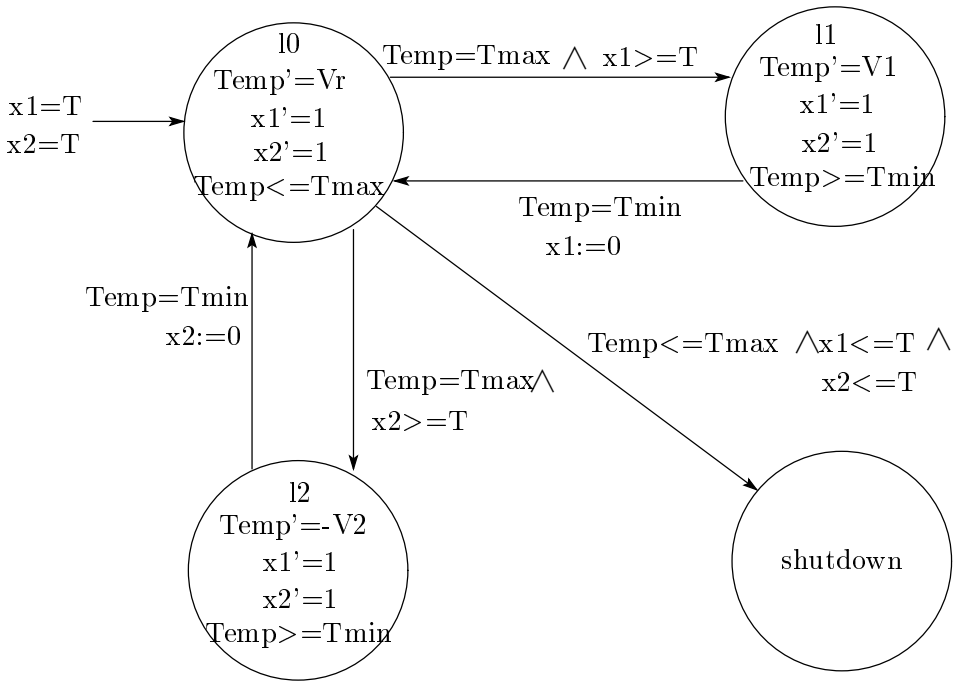

図 2.2: 温度制御システム

第3章　制約論理プログラミング

3.1　制約論理プログラミングとは

制約論理プログラミングとは，prolog に代表される論理プログラミング言語 (Constraint Losic Proguram) に各種の制約充足判定を行うソルバーを組み込んだ処理系のことである，CLP の最大の特徴は，達成したい目標において，各対象の間にどういう関係が成立するか，ということだけに着目するだけでよく，プログラムにはそれを制約として記述するだけでよい，つまり, 制約を用いることにより，問題が宣言的に記述できるということである．このことは，目標を達成する方法については触れなくすむので，プログラムを書く手間がはぶけるという利点にもなる [2]．CLP は，数値的な制約と論理的な制約の両方を扱うことができる言語である．また，CLP は，ハイブリットオートマトンの非決定的な動きに対応できる．本研究では実数体上の線形不等式制約に対するソルバーを備えた CLP 言語である Keyed CLP を用いる．

3.2　水面レベルモニタ

タンクの水面は，水面を連続的に計測しポンプのスイッチを on/off を切り替えて制御されている．水面は線形な関数によって変化する．ポンプが off の時は, 変数 y で書かれた水面は，毎秒 2 インチの速さで下降する．また，ポンプが on の時には毎秒 1 インチの速さで上昇する．初期状態として水面は 1 インチの高さであり，ポンプは on になっていると仮定する．そこで，水面を 1 から 12 インチまでの高さに保ちたい．しかし，モニタがポンプの状態を切り替える命令を出してから実際，ポンプが off になるまでに 2 秒間の遅れが生じる．したがって，モニタは水面が 1 インチを下回る前にスイッチが on になるように命令を出し，12 インチに達する前にスイッチを off にするような命令を出さなければならない．
水面を決められた範囲内に納めることができるような A，B の値を求める．また，A は水面の上限を示し，B は水面の下限を示す．以下に，このハイブリットシステムを CLP で記述した例とオートマトンの対応を以下に示す．また，プログラム全体は付録に示す．

```
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]).

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

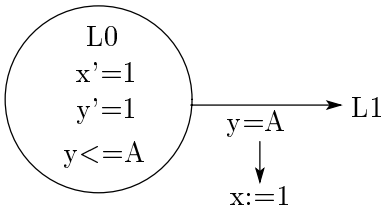

図 3.1: 水面レベルモニタ

L0 におけるオートマトンで，L0 はラベルを表し，x'，y' は変化率を表す．$y <= A$ とは，変数が満たすべき条件を表している．x と y の変化率は 1 であるので $x1 = x + D$,$y1 = y + D$ と書くことができる．水面の高さ y が A に等しくなったとき，x の値を 0 にリセットして状態は L0 から L1 に遷移される．spec(Y) のこの範囲は制御の目的を示している．同じようにそれぞれのロケーションについてプログラムを組む． 上で示したハイブリットオートマトンとプログラムから，以下のことがわかる．

- ハイブリットオートマトンと CLP の記述は 1 対 1 に対応する．
- CLP の実行ステップはハイブリットオートマトンにおける離散状態遷移に対応している．
- ハイブリット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り，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，各時間ごとに別の変数を用意しなければならない．

実行結果を以下に示す．(付録：A.1.1 参照)

```
| ?- l0(0, 1, 0, TT, [A, B]).
TT = 13  - _35 + _49
A = 10  - _35
B = 12  - _35 -2 * _49
```

```
11  -2 * _49 - _35 >= 0
 2 * _49 + _35 >= 0
7  -2 * _49 - _35 >= 0
4  + 2 * _49 + _35 >= 0
0 <= _35  <= 9
0 <= _49
```

__35、__49 という数字は変数を表している．
この実行結果は，水面の最大値 A と最小値 B の取りうる範囲を表している．

```
| ?- max(TT, l0(0, 1, 0, TT, [A, B])).
TT = 16.5
A = 10
B = 5
```

この結果から，TT を最大化することによって最適なパラメータである $A = 10$，$B = 5$ を得ることができる．なぜならば，TT は事象の切り替わりを表しているため，TT を最大化することによって最適なパラメータが求まる．

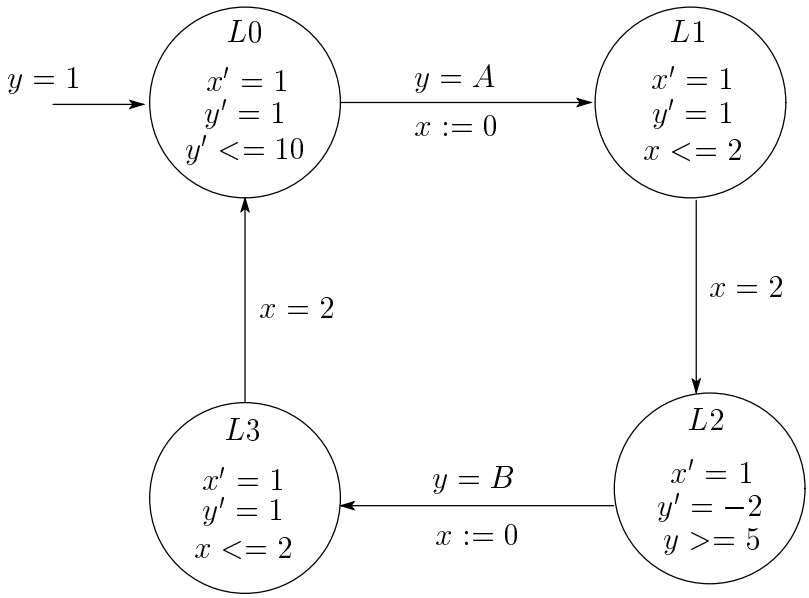

図 3.2: 水面レベルモニタ

第4章　パラメータ設計問題

4.1　CTLについて

本研究では，モデル検査 [3] などで用いられる CTL(Computation Tree Logic) の各時相オペレータに対し，それが真になるまでの最大時間を指定した論理を用いる．最大時間の指定は，CLP による計算を有限ステップで終了させるために設けたものである．

4.2　CTL-formula

- 状態，拡張状態：ロケーションの状態を表す離散変数 l および，連続変数 $x_1, \cdots x_n$ を合わせて状態変数という．状態変数の値のベクトルを状態という．状態変数に時刻 t を加えた $q = (l; x_1, \cdots, x_n; t)$ を拡張状態という．拡張状態 q のロケーションを $LOC(q)$，q の時刻を $TIME(q)$ で表す．
- 原子命題：原子命題をハイブリットオートマトン状態変数上に定義される線形不等式 (例:$l = l_0, x_1 + x_2 >= 5$) により与える．原子命題の有限集合を AP とし，各ロケーション l_i に対して，$l = l_i$ は原子命題であるとする．
- ハイブリットオートマトンは連続的な状態を許すので，next time オペレータを除いた論理式を用いる．
  - 状態論理式
    - 原子命題は状態論理式である．
    - f_1,f_2 が状態式であるならば，$\neg f_1$,$f_1 \wedge f_2$，$f_1 \vee f_2$,$f_1 \rightarrow f_2$ は状態論理式である．
    - g がパス論理であるならば,E_g，A_g はパス論理式である．
  - パス論理式
    - f_1，f_2 が状態論理式ならば，Ff，Gf，$f_1 \cup f_2$ はパス論理式である．

以下に状態論理式の解釈を示す．

- 解釈

* EFf（将来 f が真となるパスが存在する）
* AFf（任意のパス上で, 将来 f が真になる）
* EGf（常に f が真のパスが存在する）
* AGf（任意のパス上で常に f が真となる）
* E$[f_1 \cup f_2]$ (f_2 が真になるまではずっと f_2 が真のパスが存在する)

CTL 論理式は，以下に示すように，オペレータ EG および E $\cup$により他のオペレータはすべて表現できる．

EF$f \equiv$ E[true$\cup$ f]
AF$f \equiv \neg$EG$\neg f$
AG$f \equiv \neg$EF$\neg f$
A$[f \cup g] \equiv \neg[\neg g \cup (\neg f \wedge \neg g)] \wedge \neg$EG$\neg g$

- 時間限定システム：ハイブリットオートマトンから遷移システム $TS = (Q, \rightarrow Q_0, \psi)$ が定義できる．ここで，Q は拡張状態の集合，$R \subseteq Q \times Q$ は遷移関係であり，連続的な状態遷移と離散的な状態遷移の両方を含む．Q_0 は初期状態に時刻 0 を加えた拡張状態の集合を割り当てる関数である．初期状態 $q_0 \in Q_0$ からの状態遷移の軌跡 $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots$ を実行パスと呼ぶ．
  最大時間 L を与えたときの L-時間限定遷移システムとは $TS_{\leq L} = (Q_{\leq L}, \rightarrow_{\leq L}, Q_0, \psi)$ ここで，$Q_{\leq L}$ とは，時刻が L 以前は TS と等しく，時刻が L になったらそれ以降は時間が進行しないように制限した遷移システムである．(ただし，時間が進行しない離散的状態遷移は許される．)
  最大時間 L を与えたときの CTL 論理式 f($f_{\leq L}$ と書く) の真偽値を時間限定システム $TS_{\leq T}$ で与える．たとえば，$AF_{\leq 10} f$ は現在に状態から 10 単位時間以内に必ず f が真になるという性質を表す．

4.3 モデルの離散化

$TS_{\leq L}$ の各拡張状態は連続的な状態変化による非加算無限個の次状態を含むが，$TS_{\leq L}$ の状態遷移を離散的状態遷移に置き換えることにより，CLP での計算を可能にする．
原子命題の真偽値が変化する時点の状態を境界状態といい，CTL 論理式の真偽判定には，境界状態間の離散的状態遷移のみを考慮すればよい．
TS$_{\leq L}$ 上の実行パスから通過する境界状態の系列が得られる．このような，境界状態の系列全体の集合を $\Pi(TS_{\leq L})$ 上に同値関係$\equiv$を導入する．
2 つの境界状態の系列 $\pi_1 = b_0^{(1)}, b_1^{(1)}, \cdots, b_k^{(1)}$ および $\pi_2 = b_0^{(2)}, b_1^{(2)}, \cdots, b_k^{(2)}$ が，$\psi(b_i^{(1)}) \equiv \psi(b_i^{(2)})$，$i$=1,2,$\cdots$,$k$ を満たすならば，$\pi_1 \equiv \pi_2$ である．このとき，$\Pi(TS_{\leq L})$ の各同値類における境界状態の集合を境界領域という．境界領域は有限個の線形不等式により記述でき

る．
これらの結果を用いることで，$\mathrm{TS}_{\leq L}$ から境界領域間の遷移を表す離散的遷移システムを構成することができ，これをオカレンスツリーとよぶ．定義より，オカレンスツリー上の各境界領域に対し，その次の境界条件は有限個である．
以下に温度制御システムのオカレンスツリーを示す．ここで，$v_i = 6$，$v_1 = 4$，$v_2 = 3$，Tmax=15，Tmin=3，Temp=9 であり，T をパラメータとする．また，原子命題は，各ロケーションに対する $l = l_i$ に加え，Temp≤10 を与える．境界領域を定義する式は以下の通りである．
(0)$l = l_0$,$x_1 = T$, $x_2 = T$, Temp=9
(1)$l = l_0$,$x_1 = T + 1/6$, $x_2 = T + 1/6$, Temp=10
(2)$l = l_1$,$x_1 = T + 1$, $x_2 = T + 1$, Temp=15
(3)$l = l_1$,$x_1 = T + 9/4$, $x_2 = T + 9/4$, Temp=10
(4)$l = l_0$,$x_1 = 0$, $x_2 = T + 4$, Temp=3
(5)$l = l_0$,$x_1 = 7/6$, $x_2 = T + 31/6$, Temp=10

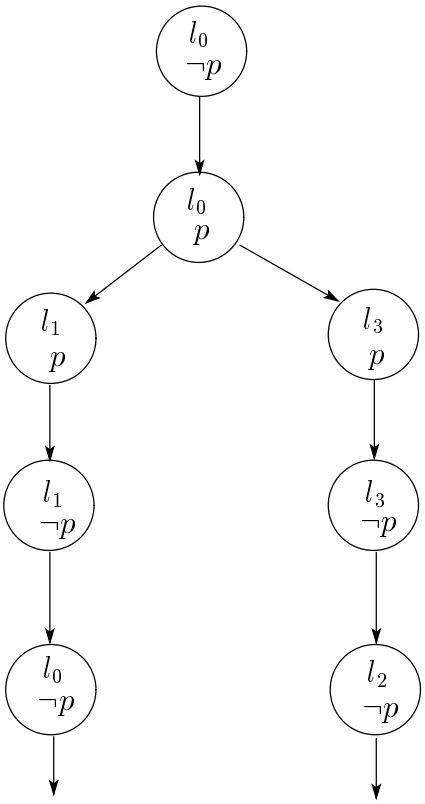

図 4.1: 温度制御システムのオカレンスツリー

第5章　CLPによるパラメータ計算

5.1　CLPの動作

CLPは，オカレンスツリーの各ノードに対応する境界領域を逐次計算することができる．CLPは1つの実行パスに沿ってツリーの各ノードに対応する状態集合を連立不等式の集合として計算している．遷移状態が満たされないならば，バックトラックを行う．または，目的とするノードに達したならば，バックトラックを行い他の選択肢を実行する．
CLPによって判定できるのは，ツリーにおいて，指定した境界領域に到達可能かどうかだけである．

5.2　CTL論理式の計算

4.2節で示した状態論理式の例とともにオカレンスツリーを示し, その例となるハイブリットオートマトンを示す.

5.2.1　EFf

温度制御の動作仕様とハイブリットオートマトンを示す.
反応タンクの冷却を行うシステムを表したものである. 冷却を行うのは2つの制御ロッドである. タンクは冷却を行わないと毎秒v_rだけ温度が上昇する. 温度が上限θ_Mに達したらどちらか一方の制御ロッドで冷却を開始する. 制御ロッドの選択は非決定的に行われる.2つの制御ロッドは異なる冷却性能を持ち, それぞれ毎秒v_1,v_2だけ温度を下げる. タンクの温度が下限$theta_m$に達したら冷却は終了する. 各ロッドは使用するとT秒間は使用できない. もし,θ_mからθ_Mの間に温度が制御できない場合はシャットダウンすることになる.
EFfの解釈は「将来fが真になるパスが存在する.」である．原子命題はfは$f = l3$を与えた．シャットダウンすれば成功であるとする．このような条件を満たすようなTmax, Tminの値を求める．

以下に，最大時刻Lを200としたときの実行結果を以下に示す．(付録：A.2.1参照)

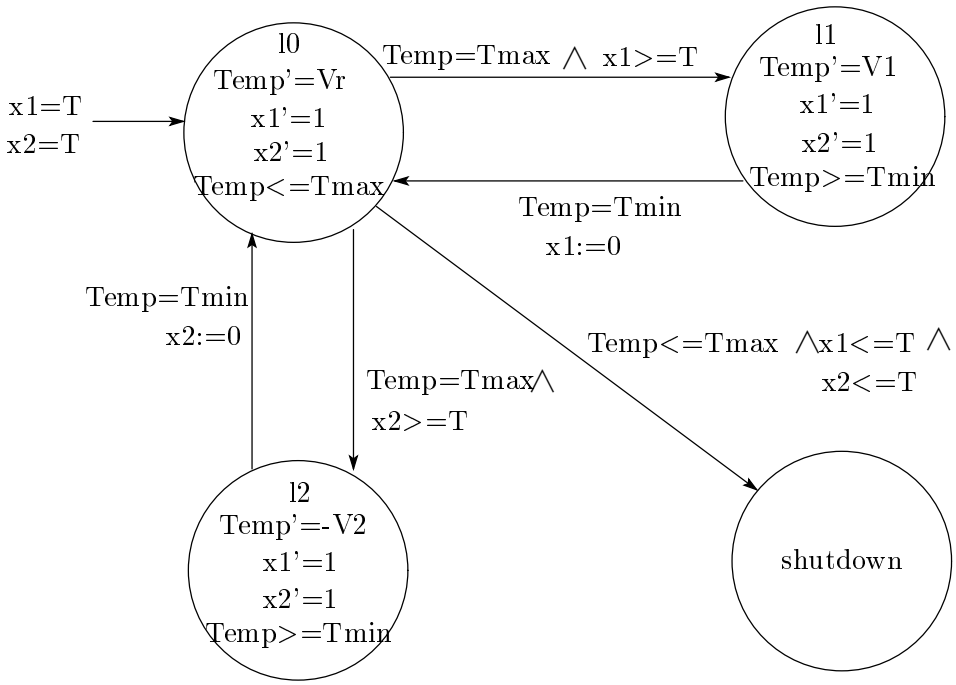

図 5.1: 温度制御システム

```
| ?- max(A, go(A, B, 500, 200)).
A = 1309.523708
B = 500.000000
```

A は Tmax を表し，B は Tmin を表している．
図の斜線部の範囲が，解の範囲となる．

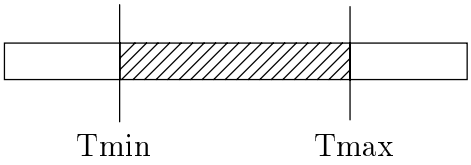

図 5.2: 解の範囲

5.2.2 AFf

使用するシステムは温度制御システムである．AFf の解釈は「任意のパス上で将来 f が真になる.」であるので，$f = l0, l1, l2$ を与えた，AF=$(l = l0, l1, l2)$ となるような，つまり，シャットダウンしないような Tmax，Tmin の値の範囲を求める．
しかし，CLP では，任意のパスで，というのは探索することができない，そこで，問題の変換を行う．

$$AFf \equiv \neg E(\neg Ff) = \neg E(G\neg f)$$

任意のパスで将来 f が真となる． ⇔ 「あるパスで将来 ¬ f が真になる」ことはない．と変換することができる．そのときの f は，

$$\neg f \equiv (l = l3)$$

とする．
この問題のシステムでは，シャットダウンする Tmax，Tmin の値を求めることになる．
ここで，得られた Tmax，Tmin の値は，将来 ¬ f が真となるパスに対する解であるので，得られた解の範囲以外であるならば将来 f が真となる Tmax，Tmin の値の範囲となる．
最大時刻を L としたときの実行結果を以下に示す．(付録：A.2.1 参照)

```
| ?- max(A, go(A, B, 500, 200)).
A = 1309.523708
B = 500.000000
```

A は Tmax を表し，B は Tmin を表している．
図の斜線部の範囲が，解の範囲となる．

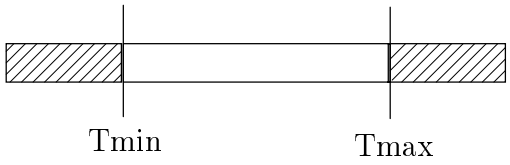

図 5.3: 解の範囲

5.2.3 EGf

この問題を解くために用いるハイブリットオートマトンを以下に示す．温度制御システムと基本的は動作は同じであるが，このシステムではロケーション $l2, l3$ に遷移された場合にシャットダウンすることになる．EGf の解釈は「常に f が真のパスが存在する.」である．$f = l0, l1$ を与えた，EG$(l = l0, l1)$ となるように常にロケーション $l0, l1$ を遷移する Tmax，Tmin の値を求める．最大時刻を L を 200 としたときの実行結果を以下に示す.

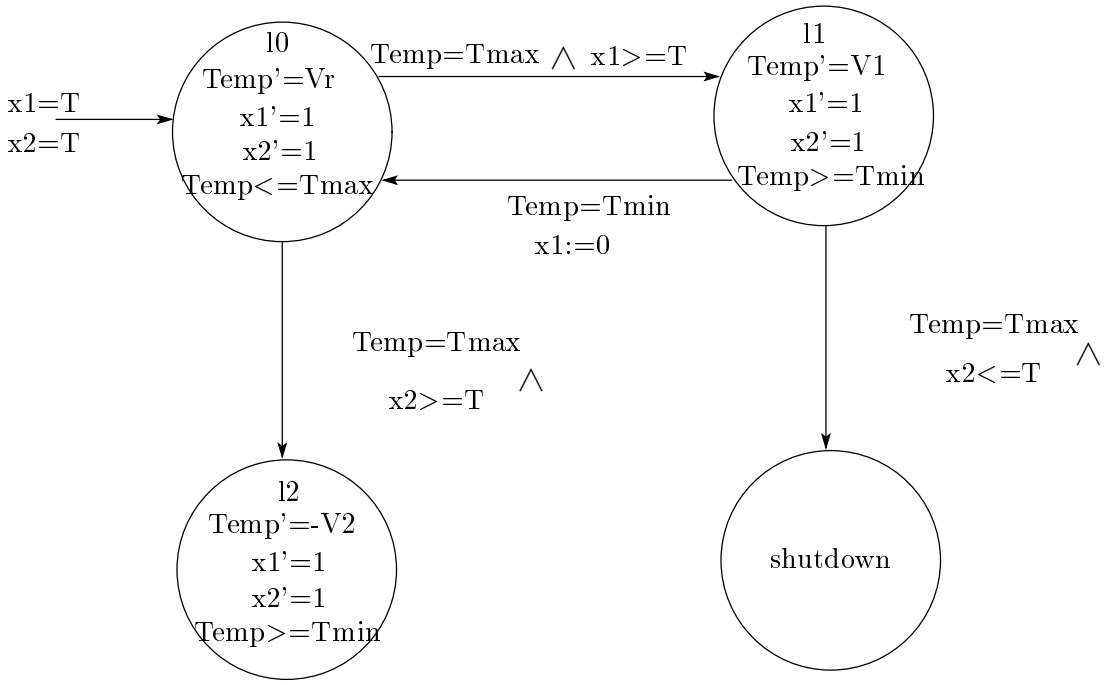

図 5.4: 温度制御システム 2

```
| ?- max(A, go(A, B, 500, 200)).
A=7300
B=500-_14
0<=_14<=500
```

A は Tmax を表し，B は Tmin を表している．また，__14 は変数を表す.

5.2.4 AGf

温度制御システムを用いた．AGf の解釈は「任意のパス上で常に f が真である.」である．$f = l0, l1, l2$ を与えた．AG(l=$l0, l1, l2$) となるような，つまり，シャットダウンしないような Tmax，Tmin の値範囲を求める．
しかし，この例題も AFf と同様に問題の変換をする必要がある．

$$AGf \equiv \neg E(\neg Gf) = \neg E(F\neg f)$$

任意のパス上で常に f が真となる． ⇔ 「あるパス上で将来¬ fが真となる」ことはない．
と変換することができる．そのときの命題は，

$$\neg f \equiv (l = l3)$$

とする.
この問題のシステムでは，シャットダウンする Tmax，Tmin の値を求めることになる．ここで，得られた Tmax，Tmin の値は，将来¬ f が真となるパスに対する解であるので，得られた解の範囲以外であるならば将来 f が真となる Tmax，Tmin 値となる．
以下に実行結果を示す．(付録：A.2.1 参照)
最大時刻 L を 200 としたときの実行結果を以下に示す.

```
| ?- max(A, go(A, B, 500, 200)).
A = 1309.523708
B = 500.000000
```

A は Tmax を表し，B は Tmin を表している．
図の斜線部の範囲が，解の範囲となる．

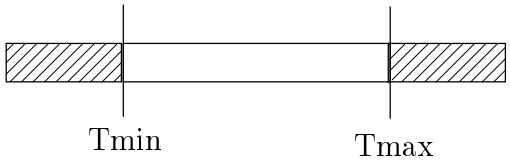

図 5.5: 解の範囲

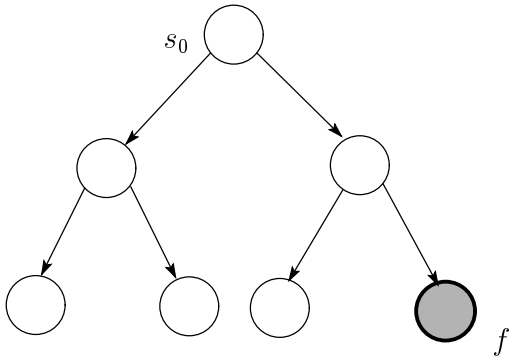

図 5.6: EFf

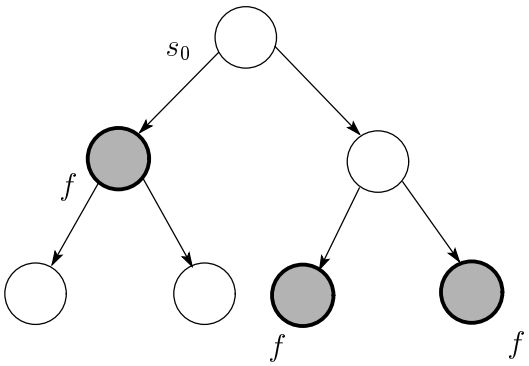

図 5.7: AFf

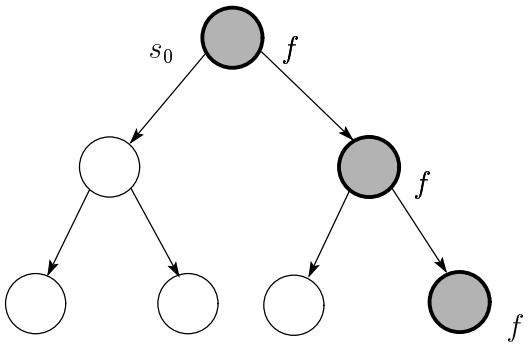

図 5.8: EGf

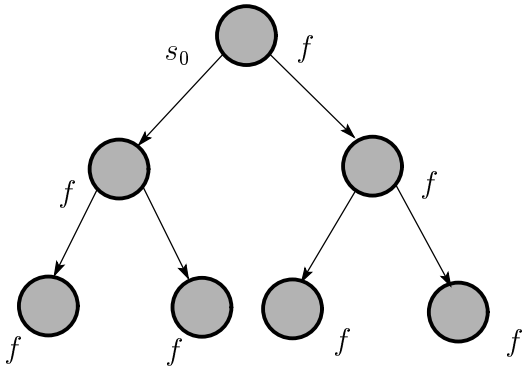

図 5.9: AGf

5.3 ネストしたCTL式に対する計算

4.2節で述べたように，CTL式は変換することにより，EF$f \equiv$ E[$f_1 \cup f_2$] とEGf を用いて計算することができる．
したがって，E[$f_1 \cup f_2$] とEGf の計算方法を与えればよい．具体的には，以下の計算を行えばよい．

- EGf の計算：オカレンスツリー上で f が真となる状態のみをたどり，設定した最大時刻 L になるまでのパスを探索する．
- E[$f_1 \cup f_2$] の計算：オカレンスツリー上で f_1 が真となる状態のみをたどり，設定した最大時刻L以前に f_2 が真の状態になるパスを探索する．

次に，時相オペレータがネストした場合を考える．パラメータを $(r_1, r_2, \cdots, r_m)$ とする．f および状態式 q に対して f を真にするパラメータの集合を $R(f,q)$ とする．すなわち，時相論理式 f が状態 q において「真」は原子命題 $(r_1, r_2, \cdots, r_m) \in R(f,q)$ と同値である．時相論理式 f を逐次，原子命題 $(r_1, r_2, \cdots, r_m) \in R(f,q)$ に置き換えて計算することにより，任意の時相オペレータが扱える．否定 $\neg\ f$ を含む場合，$R(f,q)$ の補集合である $R(f,q)^c$ を計算する必要がある．$R(f,q)^c$ を効果的に求めることは一般的に難しく，また，現状のKeyed CLPの機能では求めることができない。そこで，$R(f,q)^c$ を以下の集合で近似する方法を採用する。$R(f,q)$ における各パラメータ r_i の最小値，最大値をそれぞれ r_i*，r_i^* とする。
$R'(f,q)^c := \{(r_1, r_2, \cdots, r_m) | \vee_i (r_i \leq r_i * \vee r_i^* \leq r_i)\}$
とおくと，$R'(f,q)^c \subseteq R(f,q)^c$ が成り立つ。
よって，最終的に得られるパラメータ集合に対する振る舞いは時相論理式を満たす。

5.3.1 温度制御システムを用いた例

温度制御システムにおいて，$AG_{\leq L}[l = l_1 \rightarrow EF_{\leq 5} l = l_2]$
(時刻L以前でロケーションが l_1 の任意の状態から，5単位時間以内に l_2 に遷移することが可能である．) を満たすパラメータ T を計算するプログラムの一部を示す．プログラムはこの使用を同等な
$\neg\ EF_{\leq L}[l = l_1 \wedge \neg\ EF_{\leq 5} l = l2]$
として計算する．

```
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, 0). /*Checking EF_{<= 5} l = l2} */
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, TT):-
current_state(: [X1, X2, Temp, TT]),
range([T],
(T(: T), current_state(: [X1, X2, Temp, TT]), EF2_l1(X1, X2, Temp, TT)),
[TMin], [TMax]),
T(: T), negate(T, TMin, TMax), solve.
```

```
....


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

$EF\_\_l0()$ は $EF_{\leqq L}[l = l_1 \wedge \rightarrow EF_{\leqq 5} l = l_2]$ を満たす T の値を計算する．この中で，$EF2()$ を呼び出しているが，これは，$EF_{\leqq 5} l = l_2$ を満たす T の値を計算する $EF2\_\_l1()$ を max()，min() の中で呼び出して T の最大値，最小値を求め，それを用いて否定の近似を計算している．
range() は Keyed CLP がもつ線形目的関数の最適化を行う高階述語である．

```
range([var_1, ..., var_n], goal, [Min_1, ..., Min_n], [Max_1, ..., Max_n])
```

の実行では，goal を満たす制約条件下で各変数 var_i の最小値 Min_i, 最大値 Max_i の値を求める．goal に複数の代替案が存在する場合は，すべての場合を探索し，最小値，最大値を求める．
最後に，go(T, L) では $EF_{\leq L}[l = l_1 \wedge \rightarrow EF_{\leq 5} l = l_2]$ を満たす T を求め，その否定を計算している．
以下に最大時間を L としたときの実行結果を示す．

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

パラメータ T が満たすべき制約条件式が出力される．__18 はシステムが生成した変数であり，$0 < T \leq 7$ が可能な T の範囲となる．

5.4 考察

提案手法では，ある動作仕様を達成するパラメータ領域の否定を計算する場合に近似的手法を用いている．

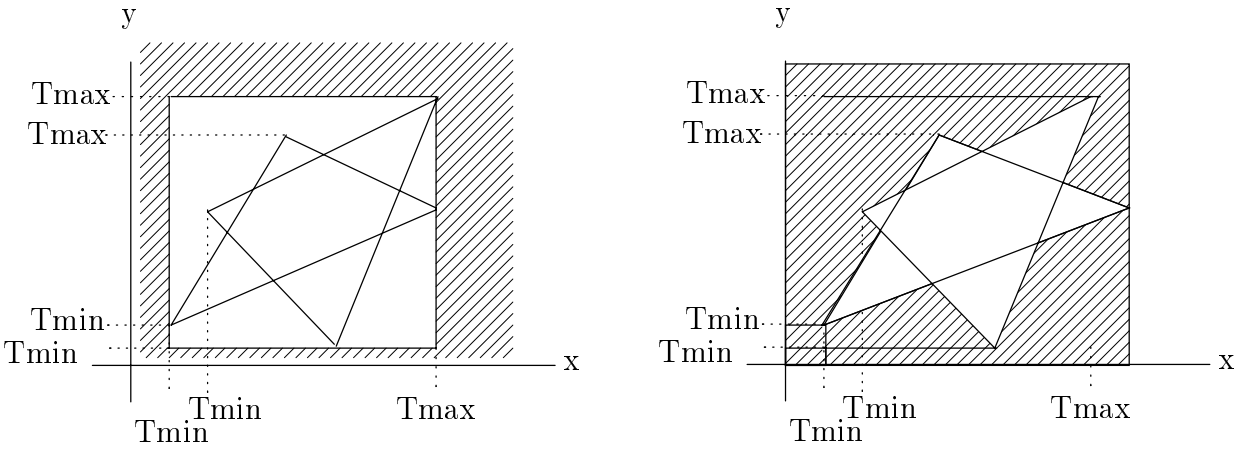

図 5.10:

上の図に示したように，本来ならば右側の図のような解の範囲としたいのだが，最大，最小をとっているために近似となっており, 左側の図のように解の範囲が制限されてしまう．この点に関しては，今後，改良していく必要がある．
本稿で提案した手法は，与えられた最大時間内に起こる離散的状態遷移の回数が有限ならば，必ず停止する．このようなハイブリットオートマトンに対しては，最大時間内という制限付きながら，動作仕様を満たすパラメータ設定を常に求めることができた．

第6章　まとめ

本研究では，ハイブリットシステムにおけるパラメータ設計問題を，連続および離散変数上の制約条件を述語とする一階述語論理式を真にする変数の値を決定する問題として定式化する．そして，論理的制約と数値的制約の両方をもつような制約充足問題を解くための処理系である制約論理プログラミング (CLP:Constraint Losic Programing) に着目し，それを用いたパラメータ設計問題について提案した．
今後は，数式処理および，幾何学の成果を取り入れ，近似ではない厳密な設計の解法について検討していきたい．

謝辞

本研究に対して終始御指導頂いた，平石邦彦教授に深謝の意を表します．また，ゼミ等で貴重なアドバイスを頂いた，宋少秋助手に感謝いたしま.。

付 録A　プログラム

以下に本研究で用いたプログラムを示す.

A.1　制約論理プログラミング

A.1.1　水面レベルモニタ

```
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]).

l1(X, Y, T, TT, [A, B]):-
   X1 = X + D, Y1 = Y + D, D >= 0,
   X1 = 2,
   spec(Y1),
   l2(X1, Y1, T + D, TT, [A, B]).

l2(X, Y, T, TT, [A, B]):-
   X1 = X + D, Y1 = Y - 2 * D, D >= 0,
   Y1 = B,
   spec(Y1),
   l3(0, Y1, T + D, TT, [A, B]).

l3(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.
```

A.1.2 温度制御システム

```
l0(S, X1, X2, Temp, TT) :-
D >= 0, param_vr(: Vr),
X1new = X1 + D, X2new = X2 + D, Tempnew = Temp + Vr * D,
(shutdown(S, X1new, X2new, Tempnew);
l0_next(S, X1new, X2new, Tempnew, TT + D)).

l0_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_A(: A), Temp <= A,
write("STEP: "), write(S), write(" finished."), nl, fail.
l0_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_A(: Temp),
param_T(: T), X1 >= T,
write("STEP: "), write(S), write(" -> l1 "), nl,
l1(S + 1, X1, X2, Temp, TT).
l0_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_A(: Temp),
param_T(: T), X2 >= T,
write("STEP: "), write(S), write(" -> l2 "), nl,
l2(S + 1, X1, X2, Temp, TT).

shutdown(S, X1, X2, Temp):-
param_A(: Temp),
param_T(: T), eps(: EPS), X1 <= T - EPS, X2 <= T - EPS,
write("STEP: "), write(S), write(" shutdown."), nl.

l1(S, X1, X2, Temp, TT) :-
D >= 0,param_v1(: V1),
l1_next(S, X1 + D, X2 + D, Temp - V1 * D, TT + D).

l1_next(S, X1, X2, Temp, TT) :-
```

```
param_L(: L), TT > L,
param_B(: B), Temp >= B,
write("STEP: "), write(S), write(" finished."), nl, fail.
l1_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
write("STEP: "), write(S), write(" -> l0 "), nl,
l0(S + 1, 0, X2, Temp, TT).

l2(S, X1, X2, Temp, TT) :-
D >= 0,param_v2(: V2),
l2_next(S, X1 + D, X2 + D, Temp - V2 * D, TT + D).

l2_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_B(: B), Temp >= B,
write("STEP: "), write(S), write(" finished."), nl, fail.
l2_next(S, X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
write("STEP: "), write(S), write(" -> l0 "), nl,
l0(S + 1, X1, 0, Temp, TT).

/* グローバル変数 */

eps(: 0.00001).
param_T(: _).
param_vr(: _).
param_v1(: _).
param_v2(: _).
param_A(: _).
param_B(: _).
param_L(: _).


go(A, B, Temp, L):-
param_T(: 80),
param_vr(: 34),
```

```
param_v1(: 25),
param_v2(: 10),
param_A(: A),
param_B(: B),
B >= 0, A >= B, Temp <= A, Temp >= B,
param_L(: L),
l0(0, 80, 80, Temp, 0).
```

A.2 CTL

A.2.1 EFf,AFf, AGf

```
l0(X1, X2, Temp, TT) :-
D >= 0, param_vr(: Vr),
X1new = X1 + D, X2new = X2 + D, Tempnew = Temp + Vr * D,
(l3(X1new, X2new, Tempnew);
l0_next(X1new, X2new, Tempnew, TT + D)).

l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_A(: A), Temp <= A,
write(" finished."), nl, fail.
l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_A(: Temp),
param_T(: T), X1 >= T,
write(" -> l1 "), nl,
l1(X1, X2, Temp, TT).
l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_A(: Temp),
param_T(: T), X2 >= T,
write("STEP: "), write(" -> l2 "), nl,
l2(X1, X2, Temp, TT).

l3(X1, X2, Temp):-
```

```
param_A(: Temp),
param_T(: T), eps(: EPS), X1 <= T - EPS, X2 <= T - EPS,
write(" shutdown."), nl.

l1(X1, X2, Temp, TT) :-
D >= 0,param_v1(: V1),
l1_next(X1 + D, X2 + D, Temp - V1 * D, TT + D).

l1_next(X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_B(: B), Temp >= B,
write(" finished."), nl, fail.
l1_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
write(" -> l0 "), nl,
l0(0, X2, Temp, TT).

l2(X1, X2, Temp, TT) :-
D >= 0,param_v2(: V2),
l2_next(X1 + D, X2 + D, Temp - V2 * D, TT + D).

l2_next(X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_B(: B), Temp >= B,
write(" finished."), nl, fail.
l2_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
write(" -> l0 "), nl,
l0(X1, 0, Temp, TT).

/* グローバル変数 */

eps(: 0.00001).
param_T(: _).
param_vr(: _).
param_v1(: _).
```

```
param_v2(: _).
param_A(: _).
param_B(: _).
param_L(: _).

/* Go */

go(A, B, Temp, L):-
param_T(: 80),
param_vr(: 34),
param_v1(: 25),
param_v2(: 10),
param_A(: A),
param_B(: B),
B >= 0, A >= B, Temp <= A, Temp >= B,
param_L(: L),
l0(80, 80, Temp, 0).
```

A.2.2 EGf

```
l0(X1, X2, Temp, TT) :-
D >= 0, param_vr(: Vr),
X1new = X1 + D, X2new = X2 + D, Tempnew = Temp + D * Vr,
l0_next(X1new, X2new, Tempnew, TT + D).

l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_A(: A), Temp <= A,
write(" finished."), nl, fail.
l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_A(: Temp),
param_T(: T), X1 >= T,
write(" -> l1 "), nl,
l1(X1, X2, Temp, TT).
l0_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
```

```
param_A(: Temp),
param_T(: T), X2 >= T,
write(" -> l3 "), nl,
l3(X1, X2, Temp, TT).


l1(X1, X2, Temp, TT) :-
D >= 0,param_v1(: V1),
l1_next(X1 + D, X2 + D, Temp - V1 * D, TT + D).

l1_next(X1, X2, Temp, TT) :-
param_L(: L), TT > L,
param_B(: B), Temp <= B,
write(" finished."), nl, fail.
l1_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
param_T(: T),
write(" -> l0 "), nl,
l0(0, X2, Temp, TT).
l1_next(X1, X2, Temp, TT) :-
param_L(: L), TT <= L,
param_B(: Temp),
param_T(: T),
write(" -> l2 "), nl,
l2(X1, 0, Temp, TT).

l3(X1, X2, Temp, TT) :- shutdown.

l2(X1, X2, Temp, TT) :- shutdown.

shutdown :- write("NG."), nl.


/* グローバル変数 */

param_T(: _).
param_vr(: _).
```

```
param_v1(: _).
param_A(: _).
param_B(: _).
param_L(: _).


go(A, B, Temp, L):-
param_T(: 80),
param_vr(: 34),
param_v1(: 25),
param_A(: A),
param_B(: B),
B >= 0, A >= B, Temp <= A, Temp >= B,
param_L(: L),
l0(80, 80, Temp, 0).
```

A.3 AGEFf

A.3.1 温度制御システム

```
EF_l0(X1, X2, Temp, TT, [T]) :-
D >= 0, param_vr(: Vr),
  write("EF_l0."),nl,
EF_l0_next(X1 + D, X2 + D, Temp + Vr * D, TT + D, [T]).

EF_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_TM(: TM), Temp <= TM, fail.
EF_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Temp),
X1 >= T,
  write("EF_l0_next-l1."), nl,
EF_l1(X1, X2, Temp, TT, [T]).
EF_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Temp),
```

```
X2 >= T,
  write("EF_l0_next-l2."), nl,
EF_l2(X1, X2, Temp, TT, [T]).
EF_l0_next(X1, X2, Temp, TT, [T]):-
param_L(: L), TT <= L,
param_TM(: Temp),
eps(: EPS), X1 <= T - EPS, X2 <= T - EPS,
  write("EF_l0_next-l3."), nl,
EF_l3(X1, X2, Temp, TT, [T]).

EF_l1(X1, X2, Temp, TT, [T]) :-
D >= 0,param_v1(: V1),
  write("EF_l1."), nl,
EF_l1_next(X1 + D, X2 + D, Temp - V1 * D, TT + D, [T]).

EF_l1_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Tm), Temp >= Tm, fail.
EF_l1_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L - 10,
  write("EF_l1_next-EF2."), nl,
EF2(T, X1, X2, Temp, TT).
EF_l1_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_Tm(: Temp),
  write("EF_l1_next-l0."), nl,
EF_l0(0, X2, Temp, TT, [T]).

EF_l2(X1, X2, Temp, TT, [T]) :-
D >= 0,param_v2(: V2),
  write("EF_l2."), nl,
EF_l2_next(X1 + D, X2 + D, Temp - V2 * D, TT + D, [T]).

EF_l2_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Tm), Temp >= Tm, fail.
EF_l2_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
```

```
param_Tm(: Temp),
  write("EF_l2_next-l0."), nl,
EF_l0(X1, 0, Temp, TT, [T]).

EF_l3(X1, X2, Temp, TT, [T]):- fail.

EF2(T, X1, X2, Temp, TT):-
current_state(: [X1, X2, Temp, TT]),
max(T,
(T >= 0, current_state(: [X1, X2, Temp, TT]), EF2_l1(X1, X2, Temp, TT, [T])),
TMax),
    write("EF2_1."), nl,
(unify(TMax, infeasible);
min(T,
(T >= 0, current_state(: [X1, X2, Temp, TT]), EF2_l1(X1, X2, Temp, TT, [T])),
TMin),
    write("EF2_2."), nl,
(T < TMin;
(unify(TMax, unbounded); TMax < T)
)
), !.

EF2_l0(X1, X2, Temp, TT, [T]) :-
D >= 0, param_vr(: Vr),
  write("EF2_l0."), nl,
EF2_l0_next(X1 + D, X2 + D, Temp + Vr * D, TT + D, [T]).

EF2_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_TM(: TM), Temp <= TM, fail.
EF2_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Temp),
X1 >= T,
  write("EF2_l0_next-l1."), nl,
EF2_l1(X1, X2, Temp, TT, [T]).
EF2_l0_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
```

```
param_TM(: Temp),
X2 >= T,
  write("EF2_l0_next-l2."), nl,
EF2_l2(X1, X2, Temp, TT, [T]).
EF2_l0_next(X1, X2, Temp, TT, [T]):-
param_L(: L), TT <= L,
param_TM(: Temp),
eps(: EPS), X1 <= T - EPS, X2 <= T - EPS,
  write("EF2_l0_next-l3."), nl,
EF2_l3(X1, X2, Temp, TT, [T]).

EF2_l1(X1, X2, Temp, TT, [T]) :-
D >= 0,param_v1(: V1),
  write("EF2_l1."), nl,
EF2_l1_next(X1 + D, X2 + D, Temp - V1 * D, TT + D, [T]).

EF2_l1_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Tm), Temp >= Tm, fail.
EF2_l1_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_Tm(: Temp),
  write("EF2_l1_next-l0."), nl,
EF2_l0(0, X2, Temp, TT, [T]).

EF2_l2(X1, X2, Temp, TT, [T]) :-
D >= 0,param_v2(: V2),
  write("EF2_l2."), nl,
EF2_l2_next(X1 + D, X2 + D, Temp - V2 * D, TT + D, [T]).

EF2_l2_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Tm), Temp >= Tm, fail.
EF2_l2_next(X1, X2, Temp, TT, [T]) :-
param_L(: L), TT <= L,
param_Tm(: Temp),
  write("EF2_l2_next-l0."), nl,
EF2_l0(X1, 0, Temp, TT, [T]).
```

```
EF2_l3(X1, X2, Temp, TT, [T]).

/* Result */

result(T, infeasible, _):- !, fail.
result(T, unbounded, unbounded):- !.
result(T, unbounded, TMin):- !, T < TMin.
result(T, TMax, unbounded):- !, T > TMax.
result(T, TMax, TMin):- !, (T > TMax; T < TMin).

/* グローバル変数 */

eps(: 0.00001).
param_T(: _).
param_vr(: _).
param_v1(: _).
param_v2(: _).
param_TM(: _).
param_Tm(: _).
param_Temp(: _).
param_L(: _).

current_state(: _).

go(T, L):-
param_vr(: 6), param_v1(: 4), param_v2(: 3),
param_TM(: 15), param_Tm(: 3), param_Temp(: 10),
param_L(: L), T >= 0,
max(T, (T >= 0, param_Temp(: Temp), EF_l0(T, T, Temp, 0, [T])), TMax),
min(T, (T >= 0, param_Temp(: Temp), EF_l0(T, T, Temp, 0, [T])), TMin), !,
result(T, TMax, TMin).
```

A.3.2 水面レベルモニタ

```
EF_l0(X, Y, TT, [T]) :-
D >= 0, param_v1(: V1),
```

```
EF_l0_next(X + D, Y + V1 * D, TT + D, [T]).

EF_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT > L,
param_TM(: TM), Y <= TM, fail.
EF_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Y),
EF_l1(0, Y, TT, [T]).
EF_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Y),
EF_l2(0, Y, TT, [T]).

EF_l1(X, Y, TT, [T]) :-
D >= 0,param_v2(: V2),
EF_l1_next(X + D, Y + V2 * D, TT + D, [T]).

EF_l1_next(X, Y, TT, [T]) :-
param_L(: L), TT > L, fail.
EF_l1_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L - 10,
EF2(T, 0, Y, TT).
EF_l1_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
X = 2,
EF_l3(0, Y, TT, [T]).

EF_l2(X, Y, TT, [T]) :- fail.

EF_l3(X, Y, TT, [T]):-
D >= 0,param_v3(: V3),
EF_l3_next(X + D, Y - V3 * D, TT + D, [T]).

EF_l3_next(X, Y, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Y), Y >= Tm, fail.
EF_l3_next(X, Y, TT, [T]) :-
```

```
param_L(: L), TT <= L,
param_Tm(: Y),
EF_l0(0, Y, TT, [T]).

EF2(T, X, Y, TT):-
current_state(: [X, Y, TT]),
max(T,
(T >= 0, current_state(: [X, Y, TT]), EF2_l1(X, Y, TT, [T])),
TMax),
(unify(TMax, infeasible);
min(T,
(T >= 0, current_state(: [X, Y, TT]), EF2_l1(X, Y, TT, [T])),
TMin),
(T < TMin;
(unify(TMax, unbounded); TMax < T)
)
), !.


EF2_l0(X, Y, TT, [T]) :-
D >= 0, param_v1(: V1),
EF2_l0_next(X + D, Y + V1 * D, TT + D, [T]).

EF2_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT > L,
param_TM(: TM), Y <= TM, fail.
EF2_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Y),
EF2_l1(0, Y, TT, [T]).
EF2_l0_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
param_TM(: Y),
EF2_l2(0, Y, TT, [T]).

EF2_l1(X, Y, TT, [T]) :-
D >= 0,param_v2(: V2),
EF2_l1_next(X + D, Y + V2 * D, TT + D, [T]).
```

```
EF2_l1_next(X, Y, TT, [T]) :-
param_L(: L), TT > L, fail.
EF2_l1_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
X = 2,
EF2_l3(0, Y, TT, [T]).

EF2_l2(X, Y, TT, [T]).

EF2_l3(X, Y, TT, [T]):-
D >= 0,param_v3(: V3),
EF2_l3_next(X + D, Y - V3 * D, TT + D, [T]).

EF2_l3_next(X, Y, TT, [T]) :-
param_L(: L), TT > L,
param_Tm(: Y), Y >= Tm, fail.
EF2_l3_next(X, Y, TT, [T]) :-
param_L(: L), TT <= L,
param_Tm(: Y),
EF2_l0(0, Y, TT, [T]).


/* Result */

result(T, infeasible, _):- !, fail.
result(T, unbounded, unbounded):- !.
result(T, unbounded, TMin):- !, T > TMin.
result(T, TMax, unbounded):- !, T < TMax.
result(T, TMax, TMin):- !, (T < TMax; T > TMin).

/* グローバル変数 */

param_T(: _).
param_v1(: _).
param_v2(: _).
param_v3(: _).
param_TM(: _).
```

```
param_Tm(: _).
param_T(: _).
param_L(: _).

current_state(: _).

go(T, L):-
param_v1(: 1), param_v2(: 1), param_v3(: 2),
param_TM(: 10), param_Tm(: 1),
param_L(: L), T >= 0,
max(T, (T >= 0, EF_l0(0, T, 0, [T])), TMax),
min(T, (T >= 0, EF_l0(0, T, 0, [T])), TMin), !,
result(T, TMax, TMin).
```

参考文献

[1] R.Alue et al, The algorithmic analysis of hybrid systems, Theoretical Computer Science 138, pp3–34, 1995.

[2] 溝口文雄, 古川康一, J-L.Lassez 制約論理プログラミング, 共立出版, 1989.

[3] E.M.Clarke, Jr. et al. Model checking, The MIT Press, 1999.

[4] L.Urbina. Analysis of hybrid systems in CLP(R), Lecture Notes in Computer Science 1118, 451-467, 1996.

[5] L.Urbina. The generalized railroad crossing, its symbolic analysis in CLP(R), Lecture Notes in Computer Science 1118, 565-657, 1996.

[6] T. J Hickey and D. K Wittenberg. Modeling hybrid system using analytic logic programming, Technical Report, Dep. Computer Science, Brandeis, 2002.