修 士 論 文

ハイブリッドシステムの述語抽象化計算の高速化に関する研究

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

田中　祐輔

2009 年 3 月

修 士 論 文

ハイブリッドシステムの述語抽象化計算の高速化に関する研究

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　浅野哲夫 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

0710045 田中　祐輔

提出年月: 2009 年 2 月

目 次

図 目 次

第1章　はじめに

ハイブリッドシステムとは，微分方程式に代表される連続ダイナミクス，および，有限オートマトンに代表される離散ダイナミクスの両方を持ったシステムである [1]．組込み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ている．その例としては，自動車や自動運転システム，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，リアルタイム回路などがあり，制御理論や計算機科学の分野において非常に多くの研究がなされている．ハイブリッドシステムの解析に関する研究の一つとしてシステム検証があり，その中心的問題として安全性検証問題がある．安全性検証問題とは，初期状態から探索を始め，安全でない状態に到達するかどうかを判定する問題である．しかし，ハイブリッドシステムにおける安全性検証問題は一般的に決定不能であることが分かっている．そこで，ハイブリッドシステムでの到達可能な状態を近似的に計算する述語抽象化という手法が提案されている [2].

述語抽象化とは，無限状態システムを有限状態システムに変換するための手法である．述語により連続状態空間を抽象状態空間に分割することで，実数値間の遷移から領域間の遷移を考えることが出来る．抽象状態は述語を用いて，連続状態が述語を満たすかどうかで分割し，連続状態の上近似を計算する．したがって，元の連続状態空間では安全性が保証されているが，抽象状態空間では安全でない抽象状態に到達してしまう可能性がある．これは領域間の遷移を考えるので，遷移先が増加し，その結果，安全でない抽象状態に到達する可能性が存在するためである．その可能性を排除するために，反例を用いて抽象状態を精練する手法がある [3]．この反例による精練を用いて詳細に検査することで，安全でない抽象状態に到達する可能性を低減することが出来る．しかしながら，述語抽象化において述語の数の増加に伴い，抽象状態から抽象状態への遷移計算回数が指数関数的に増加するという問題が発生する．また，抽象状態への計算自体が複雑であることも問題点となる．実用的なシステムにおいては，述語の数が膨大になると予想されるため，これらの問題が実用的なシステムへの適用の妨げになると考えられる．

以上を踏まえ，本研究では，この問題の解決法を提案し，述語抽象化における安全性検証問題での抽象状態から抽象状態への遷移計算回数の低減を目的とする．抽象状態から抽象状態への遷移計算が指数関数的に増加するという問題に対しては，抽象状態の集合積を用いて，遷移先の状態の上近似をとることで計算回数の低減を図る．初期時刻の抽象状態から到達可能な状態は，初期時刻の抽象状態を包含する抽象状態の集合積を用いることで，上近似を取ることが出来る．抽象状態の集合積で到達可能な抽象状態を計算すること

で，全ての抽象状態から到達可能な抽象状態を計算する必要がなく，遷移計算回数の低減が考えられる．また，抽象状態から抽象状態への計算自体が複雑であるという問題に対しては，二分決定グラフ (BDD: Binary Decision Diagram) [6] [7] を適用することで計算の効率化を図る．抽象状態は 0-1 変数のベクトルで表現可能であり，抽象状態の遷移関係はブール関数で表現できるので，それらの計算に BDD を用いることができる．

本論文の構成は，第 2 章では本論文で用いた離散時間区分的線形システムの定義と例を示し，次にハイブリッドシステムにおける述語抽象化について述べる．第 3 章では，ハイブリッドシステムにおける述語抽象化の問題点に対する提案手法を述べ，第 2 章の例を用い提案手法の例を示す．第 4 章では，提案手法における抽象化の計算に BDD を用いて到達可能な抽象状態を計算し，従来法と提案手法の計算時間の比較を行い，検証する．

第2章　ハイブリッドシステム

2.1　ハイブリッドシステムの簡単な例

ハイブリッドシステムの簡単な例をサーモスタットを用いて示す．サーモスタットによりヒータの ON/OFF 制御が行われている部屋をモデル化する．室温はサーモスタットにより連続的に温度を計測し，ヒータの ON/OFF を切り替えることで制御される．図 2.1 にサーモスタットをハイブリッドシステムにモデル化したものを示す．変数 T で表される室温は各ロケーション ON と OFF における微分方程式に従う．ここで，室温の上限を 30，下限を 10 とする．

室温の初期値を 10 とすると，ロケーション ON において $\dot{T} = 2$ で室温が上昇していく．上昇した室温がアーク上の遷移条件 $T = 30$ を満たした場合に遷移が起こり，ロケーションが ON から OFF に切り替わる．ロケーション OFF においては $\dot{T} = -1$ で室温が低下していく．以降，アーク上の遷移条件を満たした場合，離散状態の遷移が起こり，室温が 10 度から 30 度の間に保たれる．図 2.2 に室温とロケーションの変化を示す．ここで，室温の変化率をハイブリッドシステムにおける連続ダイナミクス，ロケーション ON，OFF を離散ダイナミクスという．

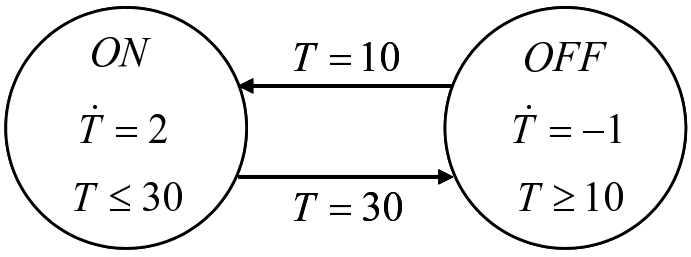

図 2.1: ハイブリッドシステムモデル

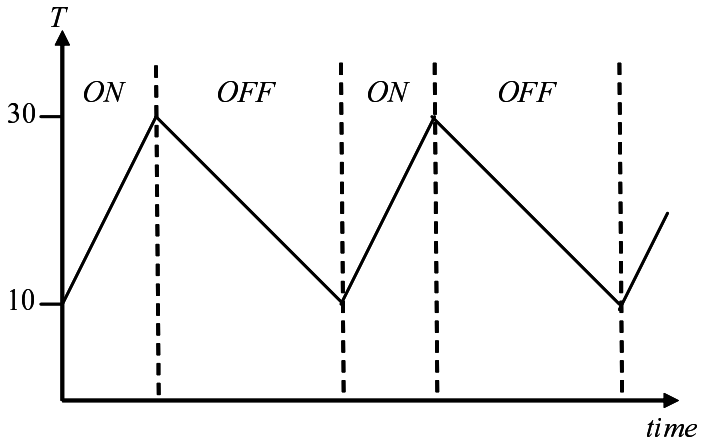

図 2.2: 室温とロケーションの変化

2.2 離散時間区分的線形システム

本論文では，ハイブリッドシステムのモデルとして，離散時間区分的線形システム

$$\begin{cases} x(t+1) = A_{I(t)}x(t) \\ I(t+1) = I(t) \text{ if } x(t+1) \in S_I \end{cases} \tag{2.1}$$

を考える．

ここで，$x(t) \in \mathcal{X} \subset \mathbf{R}^n$ は状態を表す．$\mathcal{X}$ は有界な閉凸多面体であり，$I(t) \in M := \{1, 2, \cdots, m\}$ はシステムのモードを示し，$I \neq J \in M$ に対して，$\bigcup_{I \in M} S_I = \mathcal{X}$ および $S_I \bigcap S_J = \emptyset$ を満足すると仮定する．モード i での $\mathcal{X}$ のダイナミクスは，行列 A_i を用いて表される．また，$\mathcal{X} \subset \mathbf{R}^n$ は連続的な状態空間を表す凸多面体を示す．

次に，線形式と線形述語を定義する．

【定義 1】 n 次元の線形式 $l : \mathbf{R}^n \to \mathbf{R}$ の集合を E_n と表記して，n 次元の線形述語を $\pi : \mathbf{R}^n \to \{0, 1\}$ の集合を L_n と表記する．線形式は $l(x) = \sum_{i=1}^{n} a_i x_i + a_{n+1}$ として定義し，線形述語は $\pi(x) = \sum_{i=1}^{n} a_i x_i + a_{n+1} \sim 0$ として定義する．ただし，$\sim \in \{\geq, >\}, \forall \pi \in \{1, \ldots, n+1\}$ に対して $a_i \in \mathbf{R}$ である．

次に，離散時間区分的線形システムでの集合的遷移を定義する．

【定義 2】 集合 $P \subseteq \mathcal{X}$ の遷移後継は $Post(P)$ を

$$Post(P) := \{y \in \mathcal{X} \mid y = A_i x, x \in P\}$$

として定義する．

2.3 離散時間区分的線形システムの例

例として，2 モード 2 状態の離散時間区分的線形システムとして，

$$x(t+1) = \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) \tag{2.2}$$

$$\alpha(t) = \begin{cases} \pi/3 & \text{if} \quad [1 \quad 0]x(t) < 0 \\ -\pi/3 & \text{if} \quad [1 \quad 0]x(t) \geq 0 \end{cases} \tag{2.3}$$

を考える．ここで，$\mathcal{X} \in [-2, 2] \times [-2, 2]$ とする．

初期状態を $x(0) = [1 \quad 1]^T$ とすると，$Post(x(0))$ は $x(1) = [-1.366 \quad -0.366]^T$ となる．同様に，$Post(x(1)), Post(x(2))$ は

$$Post(x(1)) = x(2) = [-0.366 \quad -1.366]^T$$

$$Post(x(2)) = x(3) = [0.999 \quad -1.106]^T$$

となる．図 2.3 に状態の軌道を示す.

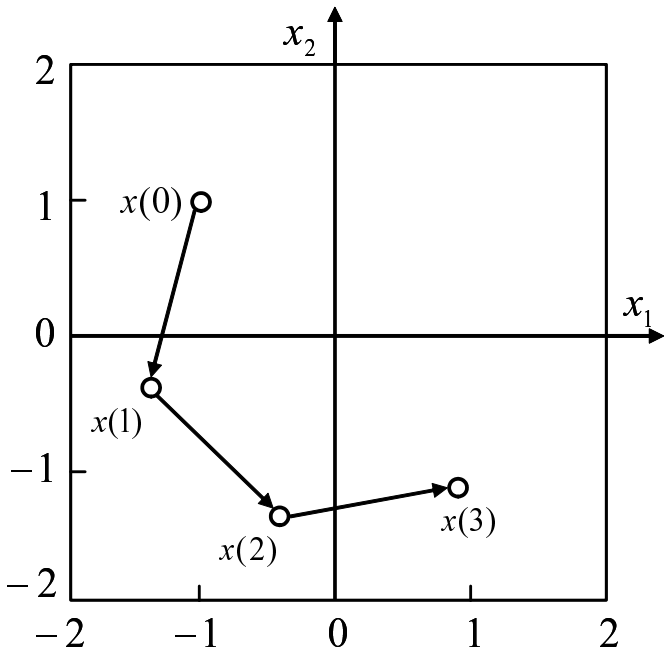

図 2.3: $x(0)$ から $x(3)$ への状態軌道

2.4 安全性検証問題

ハイブリッドシステムの解析に関する研究の１つとしてシステム検証があり，その中心的な問題として安全性検証がある．システムが安全性を満たすとは，すべての到達可能な状態が安全であることを意味する．よって，安全性検証は到達可能な状態の集合を計算することになる．次に，安全性検証問題の定義を示す.

【問題 1**】**離散時間区分的線形システム (2.1) に対して，到達可能な状態の集合 $Reach \subseteq \mathcal{X}$ は以下で定義される．

$$Reach^{(0)} := \{x \in \mathcal{X}_0\}$$

$$Reach^{(i+1)} := Post(Reach^{(i)}) \forall i \geq 0$$

$$Reach = \bigcup_{i \geq 0} Reach^{(i)}$$

ここで，$\mathcal{X}$ は初期状態集合である．安全でない状態の集合を $B_x \subseteq \mathcal{X}$ として定義する．したがって，安全性検証問題は，$Reach \cap B_x = \phi$ が成立するかどうかを判定する問題である．

しかし，ハイブリッドシステムでの安全性検証問題は一般的に決定不能であることが分かっている．そこで，安全であるための十分条件を求めるために，ハイブリッドシステムでの到達可能な状態を上近似で計算する述語抽象化という手法を用いる．述語抽象化を用いることで，実数値間の遷移から領域間の遷移を考えることが出来る．

2.5 述語抽象化

述語抽象化は複雑な無限状態システムから抽象的な有限状態システムを抽出する強力なテクニックである．述語とは，実数値が与えられた不等式を満たすかどうかで 0-1 変数を返す関数である．したがって，抽象状態は 0-1 変数のみで表現可能である．

ここで述語の数を k とすると，連続状態空間 $\mathcal{X} \subset \mathbf{R}^n$ を k 個の述語を用いて分割される．連続状態空間 $\mathcal{X}$ は述語を満たすか満たさないかの 2 通りによって分割されるため，抽象状態空間は述語の数が k 個であれば，2 通りの組合せを k 回掛け合わせた 2^k 個の抽象状態で表現できる．したがって，抽象状態における抽象状態の数は最大で 2^k 個となる．

また述語抽象化では，各抽象状態から到達可能な状態は凸多面体として計算される．初期抽象状態から到達可能な凸多面体と 0-1 変数で表された領域との積が空集合でなければ，その領域には初期抽象状態から到達可能な抽象状態であることがいえる．

以下に，線形述語によって表現できる抽象状態空間の定義を示す.

【定義 3**】** n 次元の離散時間区分的線形システムと n 次元の線形述語の k 次元ベクトル

$\Pi = (\pi_1, \pi_2, \ldots, \pi_k) \in {L_n}^k$ が与えられたとき，抽象状態は $\vec{b}$ として定義される．ここで，$\vec{b} \in \mathbf{B}^k$ である．

例として，第 2.2 節で示した離散時間区分的線形システムの例における連続状態空間の離散抽象化を示す．ここでは，以下の 6 個の述語を使う．

$$\Pi = (x_1 \leq 0, x_1 \leq 1, x_1 \geq -1, x_2 \leq 0, x_2 \leq 1, x_2 \geq -1)$$

図 2.4 に離散時間区分的線形システムでの連続状態空間と抽象状態空間を示す．抽象状態空間は 16 状態である．また，線形述語ベクトル Π の各ベクトル $\vec{b} \in \{0,1\}^k$ を用いると，連続状態空間の状態集合が計算できる．例えば，ベクトル $(0,1,1,0,0,1)$ に対し，述語の集合は

$$\{x_1 > 0, x_1 \leq 1, x_1 \geq -1, x_2 > 0, x_2 > 1, x_2 \geq 1\}$$

となる．これは，連続状態集合 $\{(x_1, x_2) \in \mathbf{R}^2 \mid 0 < x_1 \leq 1 \wedge 1 < x_2 \leq 2\}$ を表現する．

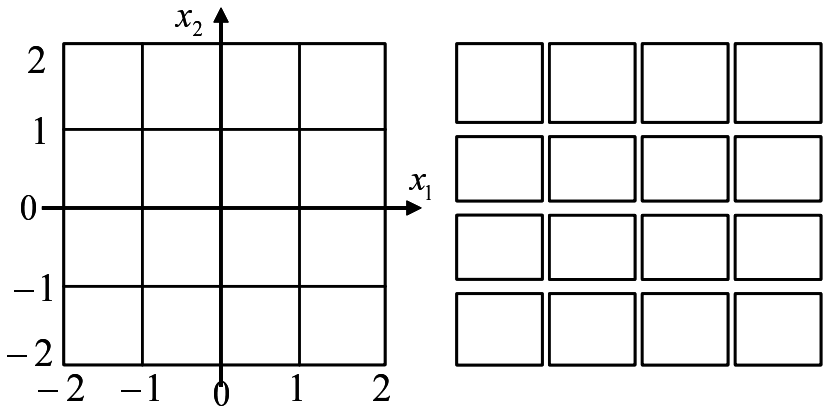

図 2.4: 離散時間区分的線形システムの例の連続状態空間と抽象状態空間

次に，具体関数を定義する．

【定義 4**】** n 次元の線形述語の k 次元ベクトル $\Pi = (\pi_1, \pi_2, \ldots, \pi_k) \in {L_n}^k$ に対して，具体化関数 $C_\Pi : \{0,1\}^k \to 2^{\mathbf{R}^n}$ は次のように定義される：

$$C_\Pi \vec{b} = \{x \in \mathbf{R}^n \mid \forall i \in \{1, \ldots, k\} : \pi_i(x) = b_i\}$$

次に述語抽象化を用いた離散時間区分的線形システムの遷移システムと述語抽象化における安全性検証問題を定義する．

【定義 5】線形述語のベクトル Π に対する離散時間区分的線形システムの抽象遷移システムは $TS_\Pi = (Q_\Pi, Q_{0\Pi}, \sum_\Pi, \delta_\Pi)$ として定義される．

1. TS_Π の状態は Q_Π
2. $Q_{0\Pi} = \left\{\vec{b} \in Q_\Pi \mid \exists x \in C_\Pi \vec{b} : x \in X_0\right\}$
3. $\sum_\Pi = C$，ただし，C は連続フローによる遷移を表す．
4. $\delta_\Pi \subseteq Q_\Pi \times \sum_\Pi \times Q_\Pi$ であり，以下で定義される：

$$\vec{b} \rightarrow^\Pi \vec{b'} :\Leftrightarrow x \in C_\Pi \vec{b} : \Phi(x,t) \in C_\Pi \vec{b'}$$

抽象状態 $\vec{b}$ と抽象状態の集合 $S \subseteq Q_\Pi$ の連続フローによる後継は，$Post^\Pi(\mathbf{b}), Post^\Pi(S)$ として表記され，以下のように定義される．

$$Post^\Pi \vec{b} = \left\{\vec{b'} \in Q_\Pi \mid \vec{b} \rightarrow^\Pi \vec{b'}\right\}$$

$$Post^\Pi(S) = \left\{\vec{b'} \in Q_\Pi \mid \exists \vec{b} \in S : \vec{b} \rightarrow^\Pi \vec{b'}\right\}$$

以下に 2.2 節の例を用いて，抽象状態の遷移関係の例を示す．

2.2 節の例における連続状態空間は 6 個の述語を用いることで，16 の抽象状態に分割することができる．また，図 2.5 のように分割された各抽象状態に S_1 から S_{16} のラベルを付ける．

図 2.5 は抽象状態 $S_5 = \{(0,1,1,0,0.1)\}, C_\Pi(\vec{b_5}) = \{x_1, x_2 \in \mathbf{R} \mid 0 < x_1 \leq 1 \wedge 1 < x_2 \leq 2\}$ から到達可能な状態への遷移を示したものである．S_5 から到達可能な状態は P である．凸多面体 P を抽象化すると，抽象状態 $\{S_1, S_2, S_3, S_6\}$ が得られる (図 2.6 参照)．

したがって，抽象状態 S_5 から到達可能な抽象状態は抽象状態 $\{S_1, S_2, S_3, S_6\}$ となる．

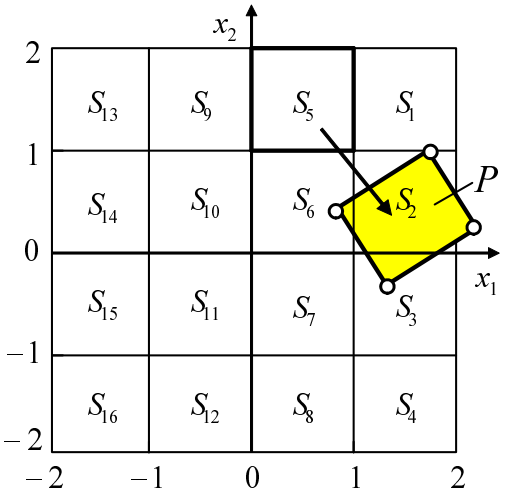

図 2.5: 抽象状態 S_7 から到達可能な状態 P

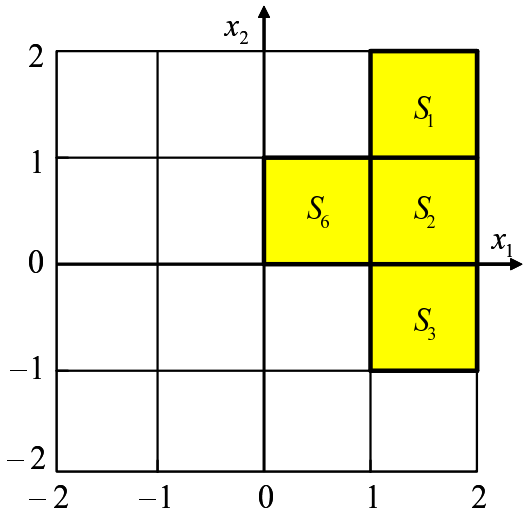

図 2.6: 抽象状態 S_7 から到達可能な抽象状態

次に，抽象状態空間においての安全性検証問題の定義を示す．**【問題 2】**離散時間区分的線形システムと述語ベクトル Π に対して，到達可能な抽象状態の集合 $Reach_{\Pi}$ は以下のように定義される．

$$Reach_{\Pi}{}^{(0)} = Q_{0\Pi}$$

$$Reach_{\Pi}{}^{(i+1)} = Post^{\Pi}(Reach_{\Pi}{}^{(i)}), \forall i \geq 0$$

$$Reach_{\Pi} = \bigcup_{i \geq 0} Reach_{\Pi}{}^{(i)}$$

ここで，$C_{\Pi}(\vec{b})$ は初期抽象状態集合である．安全でない状態の集合 B_{Π} は以下で定義する．

$$B_{\Pi} = \left\{ (\vec{b}) \in Q_{\Pi} \mid C_{\Pi}(\vec{b}) \cap B \neq \phi \right\}$$

したがって，抽象状態空間における安全性検証問題は，$Reach_{\Pi} \cap B_{\Pi} = \phi$ が成立するかどうかを判定する問題である．

抽象状態は連続状態の上近似をとるため，問題 2 で安全であると判定されれば，問題 1 においても安全であると判定できる．

2.6 反例による精錬

ハイブリッドシステムの安全性検証に述語抽象化を用いるが，抽象状態は連続状態の上近似をとるため，問題 1 では安全であると判定されているが，問題 2 では安全でないと判定してしまう可能性がある．この問題は，領域間の遷移を考えるので，遷移先が増加し，その結果，安全でない抽象状態に到達する可能性が存在するためである．

その可能性を排除するために，反例を用いて抽象状態を精錬する手法がある．反例とは安全でない抽象状態に到達するトレースである．以下に述語抽象化における反例による精錬 (CEGAR: CountEexample Guided Abstraction Refinement) の流れを示す [4].

1. まず，述語を用いて，離散時間区分的線形システムから抽象遷移システムを構成して，深さ優先探索により，初期状態から安全でない状態を探索する．
2. 次に，探索結果により，以下の二つの場合がある．
   (a) もし (安全でない状態へ到達する) 反例がなければ，安全性が成り立つと判断できる．これで CEGAR を終わる．
   (b) もし反例があれば，その反例が元の離散時間区分的線形システムが存在するかどうかを判定する．
      i. もし反例が元の離散時間区分的線形システムに存在するならば，安全性が成り立たないと判断できる．これで CEGAR を終わる．

ii. もし反例が元の離散時間区分的線形システムに存在しない (反例が偽反例) ならば，新たに，追加する述語を選択して，1. に戻る．

述語抽象化は任意の凸多面体を述語を用いて上近似で表現するが，CEGAR を行い反例を詳細に調査することで，安全性が成り立つか否かを厳密に判断することができる．

2.7 問題点

述語抽象化の特徴として，連続状態空間を最大で 2^k 個の抽象状態に分割することが挙げられる．k は述語の数である．述語の数が膨大となる大規模なシステムでは述語抽象化において，ここで，次の問題点が挙げられる．

- 抽象状態の増加に伴い，抽象状態から到達可能な抽象状態への遷移計算回数が k に関して指数関数的に増加する．

実際のシステムでは，述語の数が膨大になることが予想されるので，以上の問題点が実用的なシステムへの適用の妨げとなっている．述語の数に応じて抽象状態の数が増加すると安全でない抽象状態へ到達する可能性も増加すると考えられ，結果，反例による精練の計算回数も増加すると考えられる．この抽象状態の数の増加に対して，問題 2 の計算時間が膨大になると考えられる．したがって，これらの問題に対する解決法として，反例による精練が行われる前の段階で抽象状態から到達可能な抽象状態への遷移計算における計算回数を低減する方法を考える．

また，問題 2 における $Post^{\Pi}$ の効率化を行うために BDD を用いる．第 4 章では，$Post^{\Pi}$ に BDD を用いて提案手法の実装を行う．

第3章　集合積を用いた遷移計算の効率化

ハイブリッドシステムの述語抽象化における問題点を以下に示す.

• 抽象状態の増加に伴い，抽象状態から到達可能な抽象状態への遷移計算 $Post^{\Pi}$ 回数が述語の数 k に関して指数関数的に増加する.

本章では，この問題点に対する解決手法として，集合の包含関係を利用した手法を提案する．集合の包含関係を利用することで，$Post^{\Pi}$ の計算回数の低減を図る.

3.1　概要

述語抽象化において到達可能な抽象状態は第 2 章で定義した $Post^{\Pi}$ の計算で求めることができる．ハイブリッドシステムの述語抽象化において安全性検証問題を解くには各抽象状態から $Post^{\Pi}$ の計算を繰り返し，安全でない抽象状態に到達するかどうかを調べる．しかし，抽象状態の数が述語の数に比例して増加し，結果として $Post^{\Pi}$ の計算回数が指数関数的に増加する.

本章ではこの問題の解決のため，集合の包含関係を利用した提案手法を示す．図 3.1 は集合の包含関係の概念図を表す.

状態の集合 A と B が与えられたとき，$Post(A)$ および $Post(B)$ が計算できる．このとき，$Post(A \bigcap B) \subseteq Post(A) \bigcap Post(B)$ が成立する．言い換えると，$Post(A)$, $Post(B)$ を用いることで，$Post(A \bigcap B)$ の上近似が計算できる.

この包含関係を利用して，各抽象状態から到達可能な抽象状態を抽象状態の集合の積を用いて表現できる．この抽象状態の集合を拡大抽象状態と呼ぶことにする．到達可能な抽象状態の上近似は，拡大抽象状態から到達可能な抽象状態の集合積を用いて計算することができる．したがって，全ての抽象状態から $Post^{\Pi}$ の計算を必要とせず，$Post^{\Pi}$ の計算回数を低減することができる．さらに，格子状の離散時間区分的線形システムにおいては拡大抽象状態の取り方が容易になるので，さらなる $Post^{\Pi}$ の計算回数の低減が可能である.

提案手法は抽象状態の上近似による遷移計算であるが，安全でない抽象状態に到達しなければ，安全性が成り立つと判断できる．また，安全でない抽象状態に到達した場合は，2.6 節で説明した反例による精練を行い，詳細に検査すればよい.

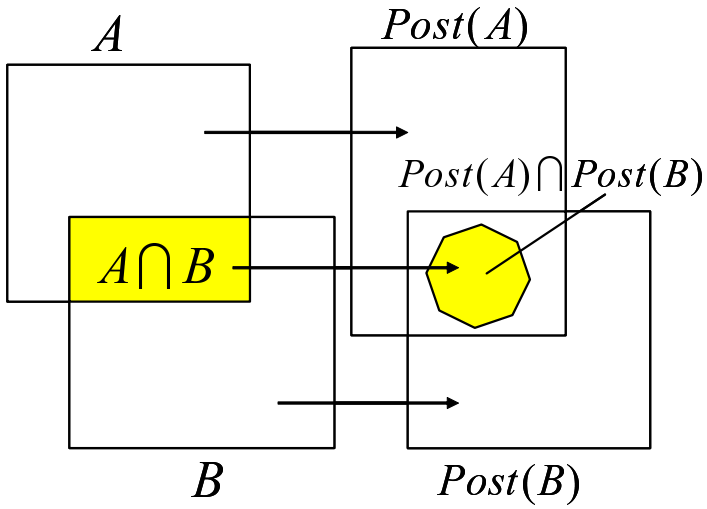

図 3.1: 集合の包含関係

3.2 集合の包含関係を用いた遷移計算

はじめに，本節の定理で用いる定義を以下に示す [5].

上記の集合の包含関係を用いた遷移計算では，$f(R_1 \cap R_2) \subseteq f(R_1) \cap f(R_2)$ の関係において等号が成り立つことが望ましい．そのための条件について考察する f を線形写像としたとき，まず $\ker f$ を $\ker f := \{x \in \mathbf{R}^n \mid f(x) = 0\}$ として定義する．W_1, W_2 は $\mathbf{R}^n$ の部分集合とする．このとき，$W_1 + W_2$ を $W_1 + W_2 := \{w_1 + w_2 \mid w_1 \in W_1, w_2 \in W_2\}$ として定義する．

また，$W_1 + \ker f$ は，$W_1 + \ker f = \{w_1 + x \mid w_1 \in W_1, x \in \ker f\}$ となるので，以下の関係が成り立つ．

$$f(W_1 + \ker f) = f(W_1) \tag{3.1}$$

このとき，次の定理が成立する．

《定理 2》 $f : \mathbf{R}^n \to \mathbf{R}^n$ が線形写像とし，R_1, R_2 を $\mathbf{R}^n$ の部分集合とする．このとき，$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成立する必要十分条件は $(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \mathrm{kerf})$ が成立することである．

（証明）

(必要性)

$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成り立つならば，$(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つことを示す．まず，$(R_1 \cap R_2) + \ker f \subseteq (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つことを示す．

$$(R_1 \cap R_2) + \ker f \subseteq (R_1 + \ker f)$$
$$(R_1 \cap R_2) + \ker f \subseteq (R_2 + \ker f)$$

となるので，$(R_1 \cap R_2) + \ker f \subseteq (R_1 + \ker f) \cap (R_2 + \ker f)$ は明らかに成り立つ．

次に，$(R_1 \cap R_2) + \ker f \supseteq (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つことを示す．$\forall r \in (R_1 + \ker f) \cap (R_2 + \ker f)$ に対して，

$$f(r) \in f(R_1) \cap (R_2) = f(R_1 \cap R_2) \tag{3.2}$$

が成り立つ．よって，

$$r \in (R_1 \cap R_2) + \ker f \tag{3.3}$$

となる．したがって，$(R_1 \cap R_2) + \ker f \supseteq (R_1 + \ker f) \cap (R_2 + \ker f)$ は成り立つ．

以上より，$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成り立つならば，$(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つ．

(十分性)

$(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つならば，$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成り立つことを示す．

まず，$f(R_1 \cap R_2) \subseteq f(R_1) \cap f(R_2)$ が成り立つことを示す．

$$f(R_1 \cap R_2) \subseteq f(R_1)$$
$$f(R_1 \cap R_2) \subseteq f(R_2)$$

となるので，$f(R_1 \cap R_2) \subseteq f(R_1) \cap f(R_2)$ は明らかに成り立つ．

次に，$f(R_1 \cap R_2) \supseteq f(R_1) \cap f(R_2)$ が成り立つことを示す．

$f(r) \in f(R_1) \cap f(R_2)$ とすると，

$$r \in (R_1 + \ker f) \cap (R_2 + \ker f) = (R_1 \cap R_2) + \ker f \tag{3.4}$$

が成り立つ．よって，

$$f(r) \in f(R_1 \cap R_2) \tag{3.5}$$

となる．

したがって，$f(R_1 \cap R_2) \supseteq f(R_1) \cap f(R_2)$ は成り立つ．

以上より，$(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \ker f)$ が成り立つならば，$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成り立つ．

以上の必要性，十分性の証明より，$f(R_1 \cap R_2) = f(R_1) \cap f(R_2)$ が成立する必要十分条件は $(R_1 \cap R_2) + \ker f = (R_1 + \ker f) \cap (R_2 + \mathrm{ker f})$ が成立することであることが示される．

定理 2 より，この必要十分条件は連続状態空間においては成り立つが，抽象状態空間においては必ずしも成り立たない．2.2 節の例を用いて，この必要十分条件が成り立たない例を示す．

2.2 節の例において抽象状態 $S_7 = (0,1,1,1,1,1), C_\Pi(S_7) = \{x_1, x_2 \in \mathbf{R} \mid 0 < x_1 \leq 1 \wedge -1 \leq x_2 \leq 0\}$ から到達可能な抽象状態の上近似を拡大抽象状態を用いて計算する．図 3.2 に計算に求める抽象状態 S_7 を示す．

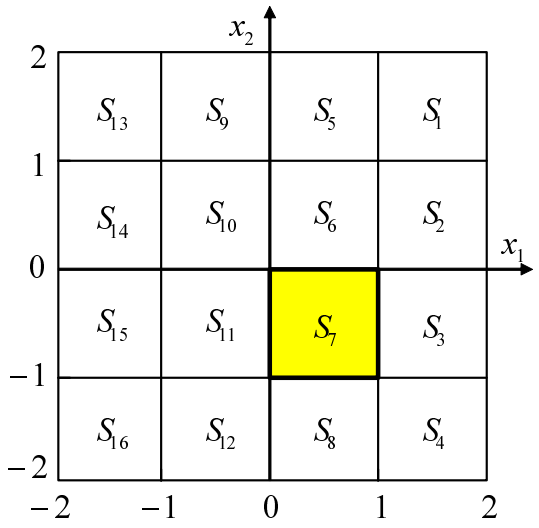

図 3.2: 初期抽象状態 S_7

抽象状態を包含する拡大抽象状態は 2.2 節の例において，各モードで各 x 軸毎に分割する．ここでは，$[1 \quad 0]x(t) \geq 0$ をモード 1，$[1 \quad 0]x(t) < 0$ をモード 2 とする．x_1 軸においては，$\Pi_{x_1} = (x_1 > 0, x_1 \leq 1, x_1 \geq -1)$ の述語を用いて 4 個の拡大抽象状態に分割することができる．x_2 軸においては，$\Pi_{x_2} = (x_2 \leq 0, x_2 \leq 1, x_2 \geq -1)$ の述語を用いて 4 個の拡大抽象状態に分割することができ，この場合は述語 $x_1 \leq 0$ でモードが分けられるので，図 3.3 のように 8 個の拡大抽象状態に分割される．したがって，全体で 12 個の拡大抽象状態が生成される．図 3.3 に 2.2 節の例での離散時間区分的線形システムにおける拡大抽象状態を示す．

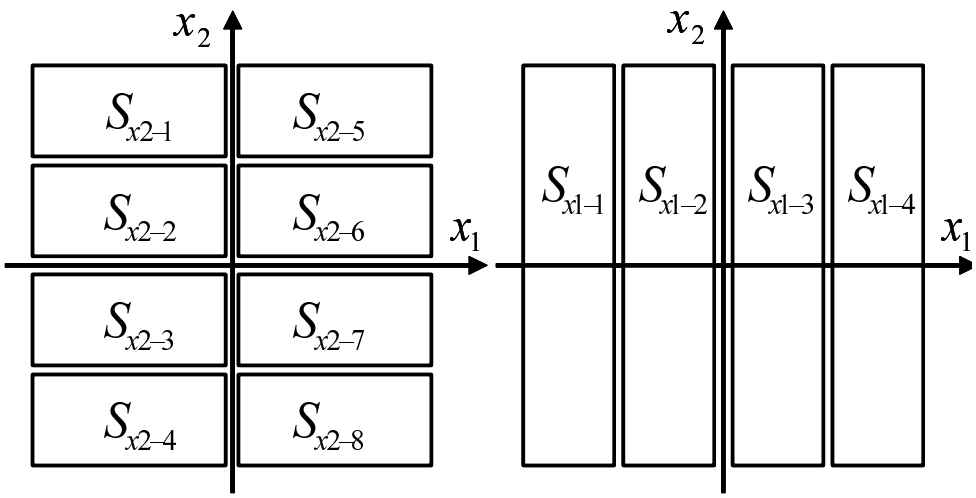

図 3.3: 離散時間区分的線形システムにおける拡大抽象状態

S_7 を包含する拡大抽象状態は,
$S_{x1-3} = \{(0,0,1,1,1,1),(0,1,1,1,1,1)\}, C_\Pi(S_{x1-3}) = \{x_1, x_2 \in \mathbf{R} \mid 0 < x_1 \leq 1 \wedge -2 \leq x_2 \leq 2\}$ と $S_{x2-7} = \{(0,1,1,1,1,0),(0,1,1,1,1,1),(0,1,1,0,0,1),(0,1,1,0,1,1)\}, C_\Pi(S_{x2-7}) = \{x_1, s_2 \in \mathbf{R} \mid 0 < x_1 \leq 2 \wedge -1 \leq x_2 \leq 0\}$ の 2 つである.

図 3.3 は S_{x1-3} から到達可能な状態 P_1，図 3.4 は S_{x2-7} から到達可能な状態 P_2 を示す.

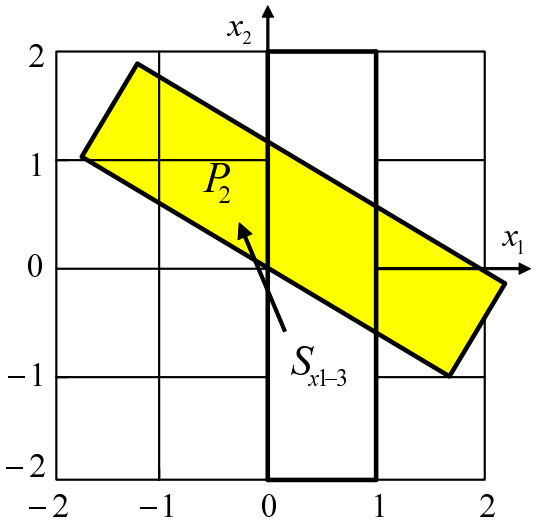

図 3.4: 拡大抽象状態 S_{x1-3} から到達可能な状態 P_2

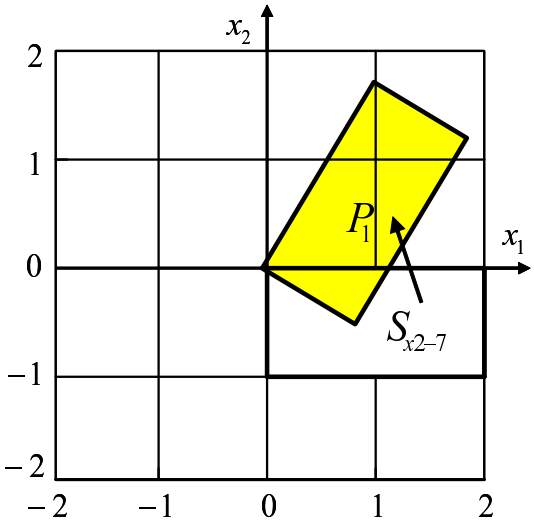

図 3.5: 拡大抽象状態 S_{x2-7} から到達可能な状態 P_1

P_1, P_2 を抽象化すると，S_{x1-3} から到達可能な抽象状態は $S_2, S_3, S_5, S_6, S_7, S_9, S_{10}, S_{13}, S_{14}$ の 9 個であり，S_{x2-3} から到達可能な抽象状態は $S_1, S_2, S_3, S_5, S_6, S_7$ の 6 個となる．

したがって，$\{S_2, S_3, S_5, S_6, S_7, S_9, S_{10}, S_{13}, S_{14}\} \bigcap \{S_1, S_2, S_3, S_5, S_6, S_7\} = \{S_2, S_3, S_5, S_6, S_7\}$ が成立する．しかし，S_7 から到達可能な抽象状態は S_2, S_3, S_6, S_7 となり，抽象状態 S_5 には到達しない．図 3.6 に抽象状態 S_7 から到達可能な抽象状態を示す．

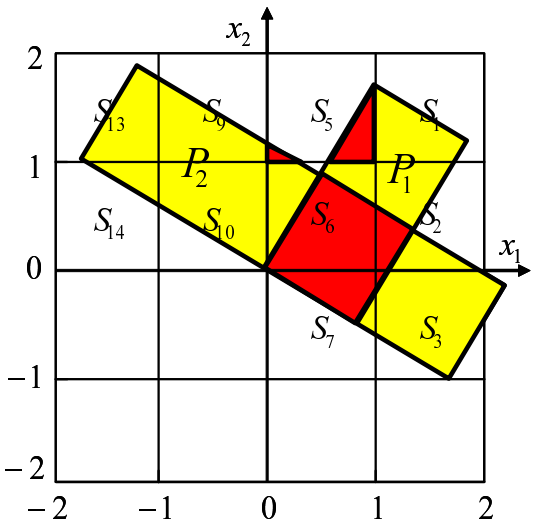

図 3.6: 抽象状態 S_7 から到達可能な抽象状態

この問題は拡大抽象状態の和集合が凸集合でないことに起因する．よって，抽象状態空間においても拡大抽象状態の和集合 $S_{x1-3} \bigcup S_{x2-7}$ が凸集合である場合は定理 2 の必要十分条件が成り立つ．

しかし，抽象状態空間において凸集合となるような拡大抽象状態の和集合の取り方は遷移計算回数を大幅に低減することはない．したがって，抽象状態空間においては集合積を用いて遷移計算を行う場合は上近似となる．

3.3 集合積による近似法

ハイブリッドシステムの述語抽象化において，$Post^{\Pi}$ の計算回数は述語の数に対して指数関数的に増加する．そこで，3.1 節で説明した集合の包含関係を利用して，3.2 節では $Post^{\Pi}$ の計算回数を低減する手法を提案した．拡大抽象状態の集合積からの $Post^{\Pi}$ の計算により，抽象状態からの $Post^{\Pi}$ の上近似を取る．従来手法では $Post^{\Pi}$ の計算を各抽象状態から行っていたが，提案手法では $Post^{\Pi}$ の計算を行う抽象状態を包含する拡大抽象状態を求めて，それぞれの拡大抽象状態から $Post^{\Pi}$ を計算し，積集合をとることで $Post^{\Pi}$ の計算回数の低減を図る．

提案手法を用いた $Post^{\Pi}(S_k)$ の計算アルゴリズムを以下に示す．

[$Post^{\Pi}(S_k)$ **の計算アルゴリズム**]

Step 1. 抽象状態 S_k を選択する．

Step 2. 抽象状態 S_k を包含する拡大抽象状態 S_i, S_j を選択する．

Step 3. 選択した拡大抽象状態 S_i, S_j から $Post^{\Pi}$ の計算を行う．

Step 4. $Post^{\Pi}(S_i) \bigcap Post^{\Pi}(S_j)$ を計算する．

問題 2 において，このアルゴリズムを繰り返し計算する．従来手法より計算回数を低減するので，結果，問題 2 の計算時間を短縮できる．

$Post^{\Pi}$ の計算アルゴリズムにおいて，以下の定理 3 と定理 4 が成り立つ．

《定理 3**》**抽象状態の集合積を用いるならば，遷移計算に使用する抽象状態の数は最大で $2k$ である．ここで，k は述語の数である．

（証明）まず，任意の抽象状態 S を考える．一つの述語に対して，その述語が真あるいは偽であるような集合に対応する 2 個の拡大抽象状態が存在する．S はこれらの抽象状態の積で表現可能である．したがって，抽象状態の集合積を用いる場合，遷移計算に使用する抽象状態の数は各述語に対して形成される 2 つの抽象状態の足し合わせた $2k$ となる．

また，格子状の離散時間線形システムにおける拡大抽象状態の個数については次の定理が成り立つ．

《定理 4**》**離散時間線形システムにおいて拡大抽象状態の集合積を用いるならば，$Post^{\Pi}$ 計算に使用する拡大抽象状態の数は $\sum_{i=1}^{n} k_i + 1$ である．ここで，k_i を各 $x_i, i = 1, 2, \ldots, n$ 毎の述語の数とする．

（証明）x_i 軸で表現できる抽象状態は $k_i + 1$ となる．全ての抽象状態はこれらの抽象状態

の集合積を取ることで表現できるので，$Post^{\Pi}$ の計算に使用する抽象状態の数は全ての次元の抽象状態を足し合わせた $\sum_{i=1}^{n} k_i + 1$ となる．

以下に，2.2 節の例において拡大抽象状態の積集合を用いた提案手法を $Post^{\Pi}$ の計算に適用した例を示す．

まず，$Step1$. において，抽象状態 $S_{13} = (1,1,0,0,0,1), C_{\Pi}(\vec{b}_{13}) = \{x_1, x_2 \in \mathbf{R} \mid -2 \leq x_1 < -1 \wedge 1 < x_2 \leq 2\}$ を選択する．提案手法を用いて，S_{13} から到達可能な抽象状態の上近似を計算する．

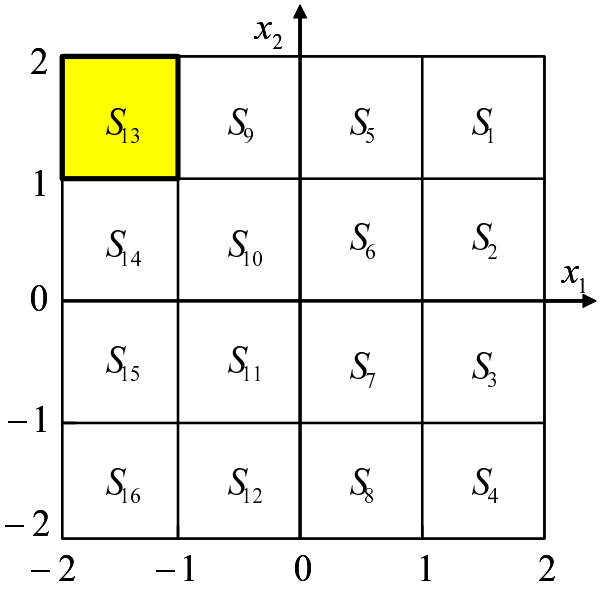

図 3.7: 抽象状態 S_{13}

$Step2$. では，$Post^{\Pi}(S_{13})$ を計算するために S_{13} を包含する拡大抽象状態を選択する．ここでは，拡大抽象状態は図 3.3 で与えることにする．この場合，$Post^{\Pi}(S_{13})$ の計算に用いる拡大抽象状態は，図 3.3 における拡大抽象状態 S_{x1-1}, S_{x2-1} である．

拡大抽象状態 S_{x1-1}, S_{x2-1} はそれぞれ，
$S_{x1-1} = \{(1,1,0,0,0,1),(1,1,0,0,1,1),(1,1,0,1,0,0),(1,1,0,1,1,1)\}, C_{\Pi}(\vec{b}_{x_1-1}) = \{x_1, x_2 \in \mathbf{R} \mid -2 \leq x_1 < -1 \wedge -2 \leq x_2 \leq 2\}$ と
$S_{x2-1} = \{(1,1,0,0,0,1),(1,1,1,0,0,1)\}, C_{\Pi}(\vec{b}_{x_2-1}) = \{x_1, x_2 \in \mathbf{R} \mid -2 \leq x_1 \leq 0 \wedge 1 < x_2 \leq 2\}$ である．

$Step3.$ において，S_{x_1-1} から $Post^{\Pi}$ の計算を行う．$Post^{\Pi}$ の計算により，$Post^{\Pi}(S_{x_1-1})$ は $\{S_4, S_8, S_{11}, S_{12}, S_{15}, S_{16}\}$ となる．S_{x_1-1} の到達可能な状態 P_1 と $Post^{\Pi}(S_{x_1-1})$ を図 3.8 と図 3.9 に示す.

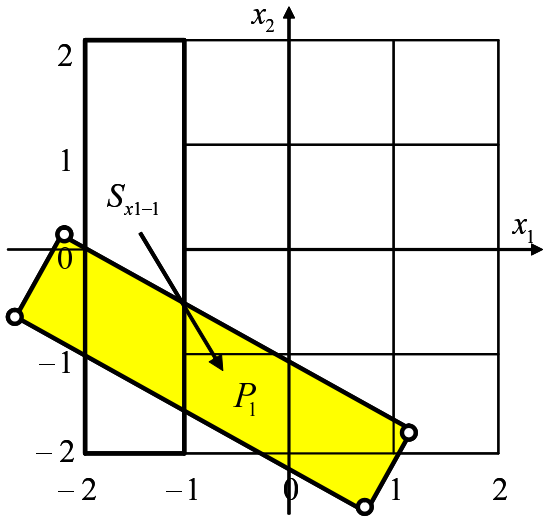

図 3.8: 抽象状態 S_{x_1-1} から到達可能な状態 P_1

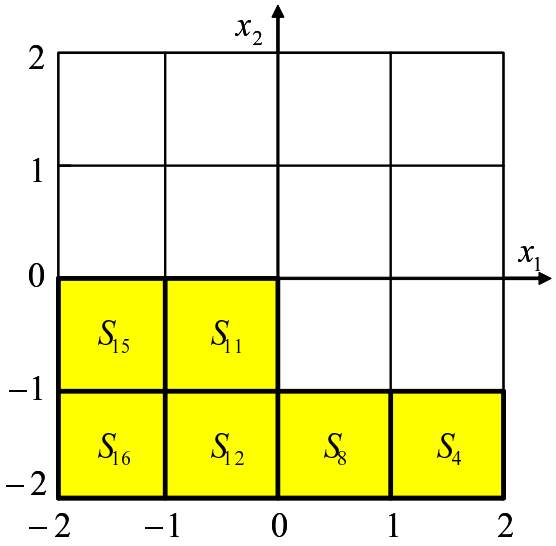

図 3.9: 抽象状態 $Post^{\Pi}(S_{x_1-1})$

次に，S_{x_2-1} から $Post^{\Pi}$ の計算を行う．$Post^{\Pi}$ の計算により，$Post^{\Pi}(S_{x_2-1})$ は $\{S_{10}S_{13}, S_{14}, S_{15}, S_{16}\}$ となる．S_{x_2-1} の到達可能な状態 S_2 と $Post^{\Pi}(S_{x_2-1})$ を図 3.10 と図 3.11 に示す．

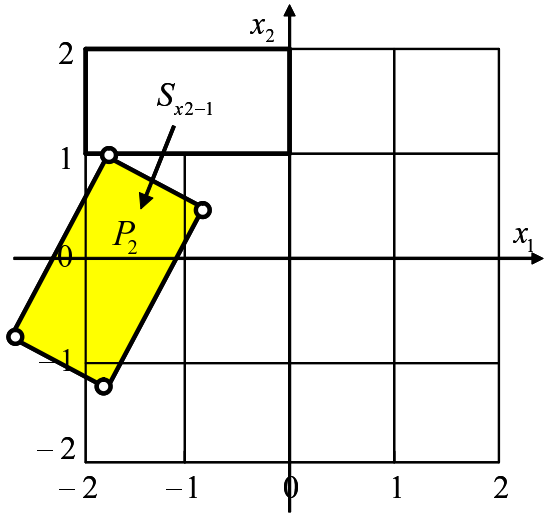

図 3.10: 抽象状態 S_{x_2-1} から到達可能な状態 P_2

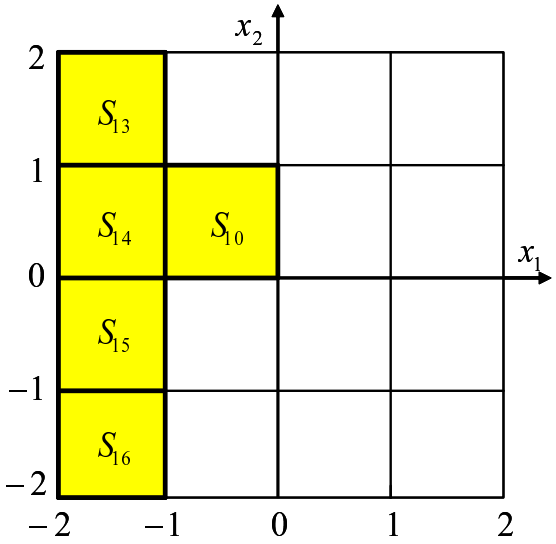

図 3.11: 抽象状態 $Post^{\Pi}(S_{x_2-1})$

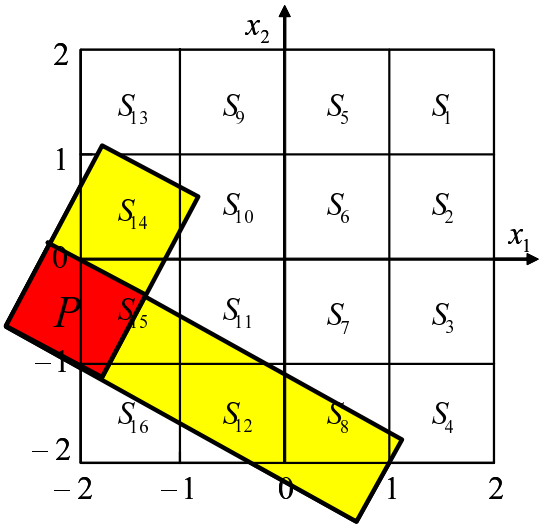

図 3.12: P_1 と P_2 の積集合 P

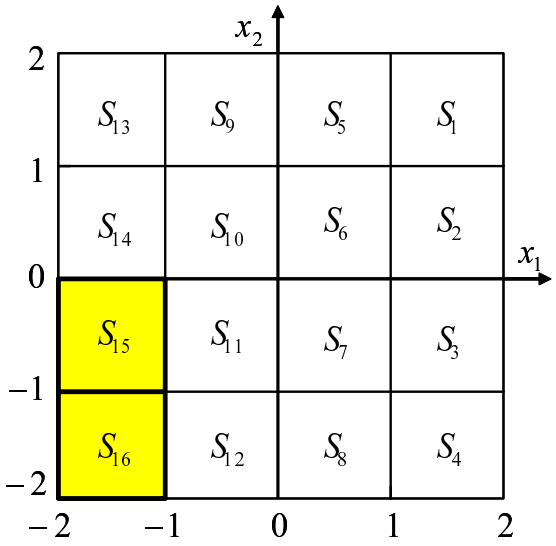

図 3.13: 抽象状態 S_{13} から到達可能な抽象状態

$Step4.$ で $Post^{\Pi}(S_{x_1-1})$ と $Post^{\Pi}(S_{x_2-1})$ の積集合をとると，$Post^{\Pi}(S_{x_1-1}) \bigcap Post^{\Pi}(S_{x_2-1})$ は $\{S_{15}, S_{16}\}$ となる．したがって，抽象状態 S_{13} から到達可能な抽象状態の上近似である $Post^{\Pi}(S_{13})$ は $\{S_{15}, S_{16}\}$ である．

従来の $Post^{\Pi}$ の計算手法を用いると，16 の抽象状態を必要とするのに対して，提案手法を用いると，12 の抽象状態の集合で $Post^{\Pi}$ の計算を行える．

述語の数が膨大となる大規模なシステムにおいては，$Post^{\Pi}$ の計算回数の低減が顕著になる．

第4章　BDDによる実装

ハイブリッドシステムの述語抽象化における安全性検証問題では抽象状態から抽象状態の遷移をを計算し，安全でない抽象状態に到達するか否かを考える．第3章では，抽象状態から抽象状態への遷移計算の回数を低減する方法を提示した．しかし，述語抽象化においては，抽象状態から到達可能な抽象状態への遷移計算が膨大な数になるため，効率的な実装方法が必要である..

そこで，BDD(二分決定グラフ) を用いて抽象化の計算の効率化を図る [6, 7]．BDD とは，ブール代数を表現するのに用いられるデータ構造である．抽象状態は0-1変数で表現可能であり，抽象状態の遷移はブール代数で表現するので，抽象状態から到達可能な抽象状態の決定にBDDを用いることが出来，抽象化の計算の効率化を図る．

また，従来手法と提案手法とを離散時間区分的線形システムと線形システムにおいて $Post^{\Pi}$ による計算時間と遷移数の比較を行う．

4.1　BDDを用いた遷移計算

第2章の問題2に対する計算時間短縮のため，遷移計算の回数を低減する手法を第3章にて提示した．ここでは，BDD を用いて，3.3 節の $Post^{\Pi}$ の計算アルゴリズムで全ての抽象状態からの $Post^{\Pi}$ を計算するアルゴリズムを示す．以下に，BDD を用いた遷移計算アルゴリズムを示す.

[BDD **を用いた遷移計算アルゴリズム**]

Step 1. 述語を用いて，全ての抽象状態と拡大抽象状態を与える．ここで，述語の数は k 個とする．述語によって分割された状態の領域を S_j，$j = 1, 2, \cdots, n$ とし，S_j に対応する抽象状態 (ブール関数表現) を b_{S_j} とする．S_j の適当な和集合を S_i^e，$i = 1, 2, \cdots, n_e$ とし，拡大抽象状態 (ブール関数表現) を $b_{S_i^e}$ とする．拡大抽象状態から到達可能な抽象状態の集合は $T_i = \phi$, $i = 1, 2, \cdots, n_e$ とする．

Step 2. $i = 1$ とする．

Step 3. $Post(S_i^e)$ を計算する．

Step 4. 以下のサブステップで $Post(S_i^e)$ との積が空集合でない抽象状態を計算する．

Step 4-1. $j = 1$ とする．

Step 4-2. $Post(S_i^e) \bigcap S_j$ を計算する．

Step 4-3. $Post(S_i^e) \bigcap S_j \neq \phi$ ならば，$T_i := T_i \cup \{S_j\}$ とする．

Step 4-4. $j := j + 1$ に更新して，Step 4-2 へ戻る．$j = n$ ならば，Step 5 へ．

Step 5. $i := i + 1$ に更新して，Step 3 に戻る．$i = n_e$ ならば，Step 6 へ．

Step 6. $\wedge_i(b_{S_i^e} \rightarrow (\vee_i b_{S_j}))$ を計算する．

次に，アルゴリズムの説明をする．拡大抽象状態は，3.3 節の例においては図 3.3 で示すとおり，抽象状態の和集合であり，全ての抽象状態が拡大抽象状態の積で表現可能であるという性質を満たす．Step 2 では，拡大抽象状態から $Post$ を計算する．Step 4 において，Step 2 で計算した $Post$ と各抽象状態との積を計算しており，積が空集合でない抽象状態の集合を 3.3 節の $Post^{\Pi}$ 計算アルゴリズムの Step 3 における $Post^{\Pi}$ として計算している．Step 6 においては，拡大抽象状態の $Post^{\Pi}$ の積を計算しており，3.3 節の $Post^{\Pi}$ 計算アルゴリズムの Step 4 に対応している．$b_{S_i^e} \rightarrow (\vee_i b_j)$ は拡大抽象状態 S_k から到達可能な抽象状態の集合 T_i への状態遷移のブール関数表現である．$b_{S_i^e} \rightarrow (\vee_i b_{S_j})$ を $i = 1$ から $i = n_e$ までの論理積を計算することで，各抽象状態から到達可能な抽象状態への状態遷移の上近似を計算している．

3.3 節の例において，この計算アルゴリズムの例を示す．$i = 2$ で計算する．
Step 1 では，
$S_1^e = S_{x1-1}$ として，$b_{S_1^e} = \{(1,1,0,0,0,1),(1,1,0,0,1,1),(1,1,0,1,0,0),(1,1,0,1,1,1)\}$，
$S_2^e = S_{x2-1}$ として，$b_{S_2^e} = \{(1,1,0,0,0,1),(1,1,1,0,0,1)\}$ とする．Step 3 では，$Post(S_1^e)$ の計算を行う．Step 4 では，$Post(S_1^e) \bigcap S_j$ を計算し，$T_1 = \{S_4, S_8, S_{11}, S_{12}, S_{15}, S_{16}\}$ となる．また，S_2^e からの計算も同様に行い，$T_2 = \{S_{10}, S_{13}, S_{14}, S_{15}, S_{16}\}$ となる．そして，Step 6 にて $\wedge_2(b_{S_2^e} \rightarrow (\vee_2 b_{S_j}))$ を計算すると，
$\{(1,1,0,0,0,1,1,1,0,1,1,1),(1,1,0,0,0,1,1,1,0,1,1,0)\}$ が得られる．これは，$S_1^e \bigcap S_2^e$ から $Post(S_1^e) \bigcap Post(S_2^e)$ への状態遷移をブール関数で表現しており，$b_{S_1^e \bigcap S_2^e} = (1,1,0,0,0,1)$ ならば，$Post^{\Pi}(b_{S_1^e \bigcap S_2^e}) = (1,1,0,1,1,1),(1,1,0,1,1,0)$ となることを意味している．

BDD を用いて遷移計算の効率化を行うことで，問題 2 における計算時間のさらなる短縮を図る．

4.2 実装例

離散時間区分的線形システムにおいて従来手法と提案手法での $Post^{\Pi}$ の計算時間を比較する．2.2 節の例を用いて全ての抽象状態から $Post^{\Pi}$ の計算を行う．

計算に用いた環境は以下のとおりである．

•CPU：Intel Pentium D(3GHz)

•OS：Windows XP Professional

実装には凸多面体操作のライブラリである NewPolka [8] ，BDD ライブラリの bddlib [9] を用いた．

従来手法を用いた全ての抽象状態から $Post^{\Pi}$ の計算時間は 0.0930 sec となり，提案手法を用いた全ての拡大抽象状態から $Post^{\Pi}$ の計算時間は 0.0780 sec となる従来手法と提案手法との計算時間を比較すると，提案手法が 0.015sec 計算時間が短いことがわかる．

次に以下の線形システムを考える．

$$x(t+1) = \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) \tag{4.1}$$

$$\alpha(t) = \pi/3 \tag{4.2}$$

線形システムにおいて従来手法と提案手法での $Post^{\Pi}$ の計算時間を比較する．2.2 節の例を用いて各抽象状態から $Post^{\Pi}$ の計算を行う．線形システムではモード遷移がないため，離散時間区分的線形システムのようにモード毎に拡大抽象状態を分割する必要がない．したがって，この例では図 4.1 に示すとおり，拡大抽象状態の数は 8 個となり，離散時間区分的線形システムより $Post^{\Pi}$ の計算回数が低減できる．

次に，線形システムにおいて従来法と提案手法での $Post^{\Pi}$ の計算時間を比較する．

従来法を用いて，全ての抽象状態から $Post^{\Pi}$ の計算時間は 0.0940 sec となり，提案手法を用いて，全ての拡大抽象状態から $Post^{\Pi}$ の計算時間は 0.0630 sec となる．線形システムにおいて，従来手法と提案手法との計算時間を比較すると，提案手法が 0.031sec 計算時間が短いことが分かる．

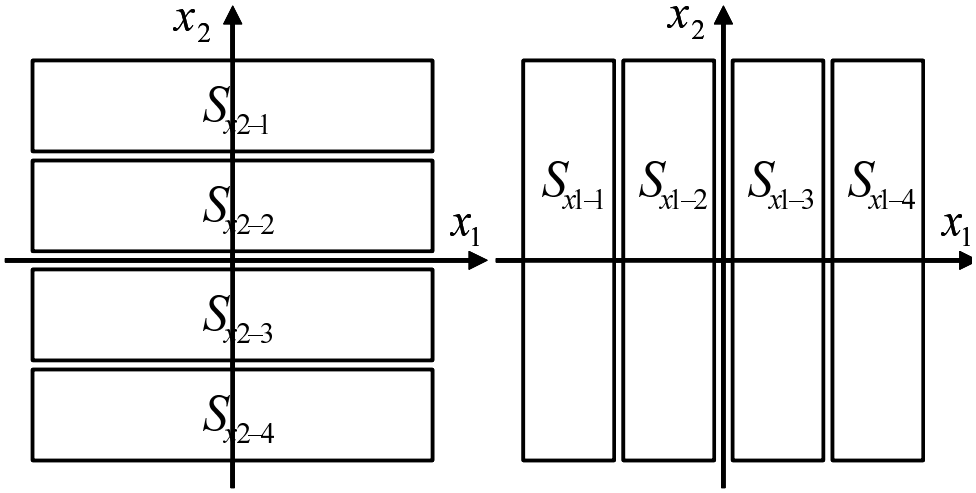

図 4.1: 線形システムにおける拡大抽象状態

表.1 は線形システムにおいて述語の数に応じて従来手法と提案手法とを比較した表である．拡大抽象状態の集合積を用いて上近似を取っているため，$Post^{\Pi}$ の計算による遷移数は提案手法のほうが多くなっている．しかし，$Post^{\Pi}$ の計算時間は提案手法のほうが短くなっており，述語の数の増加につれて，その差が顕著になっている．

図 4.2 に従来手法と提案手法の計算時間の比較を表したグラフを示す．従来手法においては，述語の数の増加に伴い計算時間が指数関数的に増加しているが，提案手法においては，計算時間は線形的に増加していることが分かる．

図 4.3 に従来手法と提案手法の遷移数の比較を表したグラフを示す．従来法と比べて，近似法は遷移数が多くなっている．しかし，増加の比率も述語の数に関係なく約 1.17 倍と誤差は小さく収まっている．

表 1：従来法と近似法による比較

各軸の述語数	計算時間 (sec)		B/A	遷移数		D/C
	従来法A	近似法B		従来法C	近似法D	
3	0.094	0.078	0.829787	56	64	1.142857
4	0.141	0.109	0.773050	97	109	1.123711
5	0.234	0.141	0.602564	140	164	1.171429
6	0.359	0.204	0.568245	189	213	1.126984
7	0.547	0.234	0.427788	236	280	1.186441
8	0.891	0.360	0.404040	329	385	1.170213
9	1.312	0.421	0.320884	396	458	1.156566
10	1.922	0.626	0.325702	481	565	1.174636
11	2.656	0.733	0.275979	568	664	1.169014
12	3.625	0.985	0.271724	665	781	1.174436
13	5.047	1.187	0.235189	784	928	1.183673
14	6.781	1.454	0.214423	889	1041	1.170979
15	8.859	1.625	0.183429	992	1172	1.181452
16	11.875	2.266	0.190821	1157	1365	1.179775

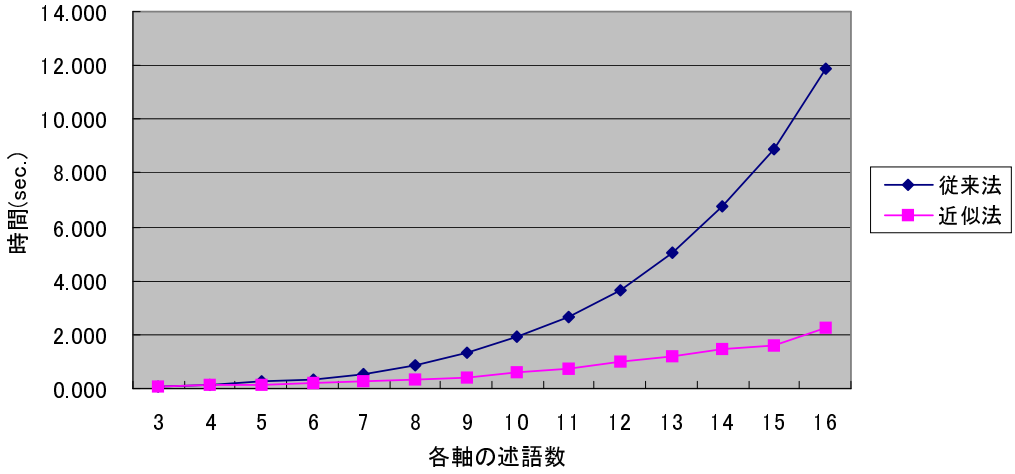

図 4.2: 従来法と近似法による計算時間の比較

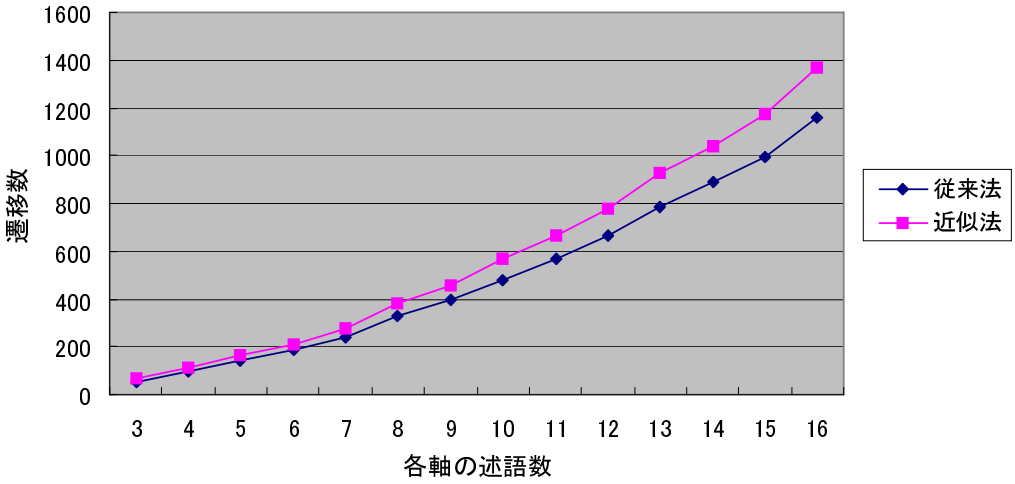

図 4.3: 従来法と近似法による遷移数の比較

4.3　考察

2.2 節の例を用いて離散時間区分的線形システムの述語抽象化における遷移計算を従来手法と提案手法とで実装した．従来手法での計算時間と提案手法での計算時間を比較した結果，提案手法での計算時間が従来手法での計算時間より短いことが分かった．

また，さらなる計算回数の低減を目指し，線形システムにおいても実装し，従来手法と提案手法の比較を行った．従来手法での計算時間と提案手法での計算時間を比較した結果，提案手法での計算時間が従来手法での計算時間より短いことが分かった．

モード遷移のない線形システムにすることで，拡大抽象状態が減少し，遷移計算時間をさらに短縮することができた．線形システムにおいては，述語の数の増加に伴い，従来手法では計算時間が指数関数的に増加している．各抽象状態からの遷移数は，述語の増加に伴い，提案手法のほうが従来手法より多くなっているが，増加の比率は述語の数に関係なく約 1.17 倍と比率の差は小さく収まっている．

第5章　まとめ

本研究では，ハイブリッドシステムの述語抽象化の実用化に向け，集合の包含関係を利用した新しい計算手法を提案した．さらに，BDD を用いて実装することで，計算時間が低減されることを数値例により確認した．以上から，提案手法は実用的な観点から有用であると考えられる．今後は，大規模システムへの適用のため，格子状ではない離散時間区分的線形システムでの計算を検証していく必要がある．

謝辞

本研究に対して，終始ご熱心に御指導頂いた平石邦彦教授に感謝の意を表し，心より御礼申し上げます．本研究進めるにあたり，貴重な助言を頂いた小林孝一助教に深く感謝致します．また，大学院での生活において様々な面で支えていただいた崔舜星氏をはじめとする，システム基礎講座の皆様に感謝します．

参考文献

[1] 井村　順一，東　俊一：ハイブリッドシステムの制御-I：総論．システム情報制御学会 Vol.51, No.5, pp230-237, 2007.

[2] Rajeev Alur, Thao Dang, Franjo Ivancic : Conterexample-guided predicateabstraction of hybrid system. Theoretical Computer Science 354, no.2, pp.250-271, 2006.

[3] 加藤　位明, 陸田　陽介, 山根　智：述語抽象化とその精練による確率線形ハイブリッドオートマトンの到達可能解析手法．IEICE Technical Report CST2006-4,2006-06.

[4] 山根　智：組込みシステムのフォーマルメソッドにおけるハイブリッドシステムの使用記述と形式的検証．Fundamentals Review Vol.2 No.1 pp.22-34 2008-7.

[5] 布川昊，中山　弘隆，谷野　哲三：線形代数と凸解析，コロナ社 (1996).

[6] 石浦　菜岐佐：BDD とは，情報処理，Vol.34, No.5, pp.585-592,1993.

[7] 湊　真一：計算機上での BDD の処理技法, Vol.34, No.5, pp.593-599,1993.

[8] http://www.inrialpes.fr/pop-art/people/bjeannet/newpolka/index.html

[9] http://www-2.cs.cmu.edu/ modelcheck/bdd.html