Development of design methodologies and support environments of high-reliability embedded systems based on hybrid models

研究題名

ハイブリッドモデルによる組込みシステムの高信頼性設計方法論の構築と支援環境の開発

課題番号：１４５８０３６８

平成１４年度～平成１６年度科学研究費補助金（基盤研究（Ｃ）（2））研究成果報告書

平成１７年３月

研究代表者　　山根　智
（金沢大学工学部，教授）

研究題名

ハイブリッドモデルによる組込みシステムの高信頼性設計方法論の構築と支援環境の開発

課題番号：１４５８０３６８

平成１４年度～平成１６年度科学研究費補助金（基盤研究（Ｃ）（2））研究成果報告書

平成１７年３月

研究代表者　　山根　智
（金沢大学工学部）

はしがき

研究組織

研究代表者：山根　智（金沢大学工学部）

交付決定額（配布額）　　（金額単位：千円）

	直接経費	間接経費	合計
平成１４年度	１０００	０	１０００
平成１５年度	６００	０	６００
平成１６年度	６００	０	６００
総計	２２００	０	２２００

研 究 発 表

1. 学会誌等

(1)S. Yamane:``Refinement Theory of Embedded systems based on Hybrid models'', The 2002 International Conference on Information and Knowledge Engineering, pp.455-461, CSREA Press, 2002

(2)S. Yamane:``Formal development methodology of hybrid systems based on both control theory and computer science'', The 2002 International Conference on Information and Knowledge Engineering, pp.469-475, CSREA Press, 2002

(3)山根 智:"ハイブリッドオートマトンによるリアルタイムソフトウェアの仕様記述とスケジューラビリティ検証",電子情報通信学会研究報告 SS2002-18, pp.19-24, 2002

(4)S. Yamane, T. Yamanokuchi: ``Formal Verification of schedulability of real-time software'', 1st International Forum on Information and Computer Technology, pp.209-214, 2003.

(5)Y. Tachi, S. Yamane: ``Refinement Verification of Hybrid Automata for Embedded Systems'', 1st International Forum on Information and Computer Technology, pp.203-208, 2003.

(6)山根 智:" ハイブリッドシステムのモジュールの仕様記述と検証の手法", 情報処理学会論文誌, Vol.44, No.3, pp.867-914, 2003.

(7)山根 智、中村 一博:" 実時間システムのための近似手法に基づいた記号

モデル検査器の開発と評価”, 電子情報通信学会, J86-D1, pp. 232-247, 2003.
(Satoshi Yamane, Kazuhiro Nakamura: Development and evaluation of symbolic model checker based on approximation for real-time systems. Systems and Computers in Japan 35(10): 83-101 (2004))

(8)山根　智：招待講演　”計算機科学の立場からのハイブリッドシステムの形式的検証手法の動向”, 第 16 回回路とシステム軽井沢ワークショップ, pp. 237-242, 電子情報通信学会, 2003

(9)S. Yamane: ``Formal refinement verification method of real-time systems with discrete probability distributions'', 2nd International Workshop on Automatic Verification of Infinite-State Systems, Naval Research Laboratory, pp. 202 -215, 2003.

(10)山根　智:”離散確率分布を持つリアルタイムシステムの詳細化検証手法”, 電子情報通信学会研究報告 SS2002-31, pp. 25-30, 2003

(11)中野　善幸, 山根　智：” 時間弱模倣検証とスケジュラビリティ検証の統合化によるリアルタイムソフトウェアの詳細化設計手法”, 電子情報通信学会研究報告 DC2003-08, pp. 19-26, 2003

(12)S. Yamane: ``Formal Probabilistic Refinement Verification Method of Embedded Real-Time Systems'', Proc. IEEE Workshop on Software Technologies for Future Embedded Systems, pp. 79-82, 2003.

(13)山根 智 :”離散確率分布を持つリアルタイムシステムの詳細化検証手法”, 情報処理学会論文誌, Vol. 44, No. 8, pp. 2189-2199, 2003.

(14)S. Yamane: ``Deductive Schedulability Verification Methodology of Real-Time Software using both Refinement Verification and Hybrid

Automata'', pp. 527-533, Porc. IEEE 27th COMPSAC, 2003.

(15) S. Yamane :``Probabilistic Timed Simulation Verification and its application to Stepwise Refinement of Real-Time Systems'', Eighth Asian Computing Science Conference, Lecture Notes in Computer Science 2896, pp. 276-290, Springer-Verlag, 2003.

(16) S. Yamane :``Deductive Verification of Probabilistic Real-Time Systems'', Proc. Third International Workshop on Assurance in Distributed Systems and Networks (ADSN 2004), pp. 622-627, IEEE Computer Society, 2004

(17) 山根 智：招待講演" ハイブリッドシステムの形式的検証手法"，システム制御情報学会大会，pp. 37-44, 2004

(18) 山根 智 :" 離散確率分布を持つリアルタイムシステムの確率時間双模倣関係と確率時間時相論理式の保存"，情報処理学会論文誌，Vol. 45, No. 5, pp. 2189-2199, 2004.

(19) 山根 智 :" 離散確率分布を持つリアルタイムシステムの確率時間時相論理式の演繹的検証手法"，情報処理学会論文誌，Vol. 45, No. 6, pp. 2189-2199, 2004.

(20) 山根 智："離散確率分布を持つリアルタイムシステムの確率時間時相論理式の演繹的検証手法"、電子情報通信学会研究報告 SS2003-31, pp. 25-30, 2004

(21) S. Yamane, T. Kanatani: Deductive Probabilistic Verification Methods for Embedded and Ubiquitous Computing. Lecture Notes in Computer Science 3207, pp. 183-195 , Springer-Verlag, 2004.

(22)陸田　陽介、山根　智："確率線形ハイブリッドオートマトンの記号的到達可能性解析手法"、電子情報通信学会研究報告 CAS2004, pp.7-12, 2004

(23)山崎　貴史、山根　智："非線形ハイブリッドオートマトンの近似解析による到達可能性解析検証手法"、電子情報通信学会研究報告 CAS2004, pp.13-17, 2004

(24)S. Yamane ：５章："The automatic verification system for real-time systems using symbolic model-checking", AMAST Series in Computing, Vol.10, World Scientific, 2005(in press)

２．　口頭発表

なし

３．　出版物

なし

目　次

1 はじめに

マイクロプロセッサの普及に伴い, ゲートウェイや携帯電話などのインターネット関連機器及び自動車やプラント制御などの制御システムといった組込み型システムが多数開発されている [1, 2]. 組込み型システムはアナログ環境に組み込まれたデジタルな実時間システムであり, ハイブリッドシステム（いわゆる, アナログ動作とデジタル動作が混在するシステム）[2] と考えることができて, 信頼性が保証できる設計方法論の構築が要望されている分野である. なぜならば, 組込み型システムのバグは, 人命に関わったり, 莫大な金銭的な損失を引き起こすからである.

本研究の目的は, ハイブリッドシステムの観点から, 組込み型システムの信頼性が保証できる設計方法論を構築することである. 本質的には, 組込み型システムはアナログ動作とデジタル動作が混在している. 従来の研究では, アナログ動作を抽象化してデジタル動作のみを対象とする手法がほとんどであった. なぜならば, 従来の計算機科学には, アナログ動作を対象とする考え方は存在しなかった. しかし, 最新の研究では, 画期的なブレークスルーがあり, アナログ動作とデジタル動作の両方が仕様記述できるハイブリッドオートマトン及びそのモデル検査などの形式的検証手法が開発されている. しかし, 既存研究には, 以下の重要な観点が考慮されていない.

1. 組込み型システムの設計作業は, 抽象度の高い仕様から, 実装可能な抽象度の低い仕様へと段階的に詳細化しながら, 進められる. このために, 抽象度の高い仕様が抽象度の低い仕様に正しく実現されていることを保証することが非常に重要である. 既存のハイブリッドシステムの研究では, このような詳細化検証設計方法論の研究が存在しない.
2. 組込み型システムは大規模であり, 設計や検証において, モジュール単位の仕様記述手法と検証

手法が要求される. 既存のハイブリッドシステムの研究では, このようなモジュール毎の仕様記述と検証の研究が存在しない.

3. プリエンプションを有する, リアルタイムオペレーティングシステムはハイブリッドシステムとしてモデル化される必要があることが知られている. 既存研究では, ハイブリッドシステムとしてモデル化された, リアルタイムソフトウェアのモデル検査が行われている. しかし, ハイブリッドシステムとしてモデル化された, リアルタイムソフトウェアのスケジューラビリティ検証は存在しない.
4. 既存の制御システムの仕様は制御理論として記述されており, 形式的検証ができない. 制御理論に基づく制御仕様からハイブリッドモデルへの統合的な手法は存在しない.
5. 最近のハイブリッドシステムはソフトリアルタイム性を有していたり, 分散化しており, モデルに確率を加える必要がある. そこで, 確率ハイブリッドオートマトンが開発されている. しかし, 既存研究では, 確率ハイブリッドオートマトンのサブクラスのみが検証可能であり, 実用上から重要なクラスの検証器は開発さていない.

本研究では, 上記の重要な指摘より, 以下の研究を実施する.

1. ハイブリッドシステムの演繹的な詳細化検証理論を開発する.
2. ハイブリッドシステムのモジュール毎の仕様記述と検証の手法を開発する.
3. リアルタイムソフトウェアをハイブリッドシステムとしてモデル化して, そのスケジューラビリティ検証を行う.
4. 制御理論に基づく制御仕様からハイブリッドモデルへの統合的な手法を開発する.

5. 確率線形ハイブリッドオートマトンの検証理論を開発する.

本研究の主要な独創性と学術的意義は以下のとおりである.

1. ハイブリッドシステムのモジュール記述のために開発した仕様記述言語である, フェーズ遷移モジュールは, モジュール性, アナログ動作とデジタル動作の混在記述及び無限な状態数を有する高度な表現能力といった特徴を有しており, ハイブリッドシステムの一般的な計算モデルとなりうる. このような計算モデルは従来, 提案されていない.
2. フェーズ遷移モジュールの実装可能性（receptiveness）の演繹的検証手法及びモジュール単位の演繹的詳細化検証手法は, 実用レベルの組込み型システムの開発に本手法を適用可能とする, 重要なスケールアップ技術であり, 既存研究には存在しない. 既存の形式的手法の多くは仕様記述言語や検証手法などの提案のみで終わっているものが多い. しかし, 本研究では, 定理証明器上に設計手法を実装して, 上記のスケールアップ技術により, 実用レベルの組込み型システムに適用・評価するものであり, この適用・評価は大きな意義がある.

また, 関連研究に対する本研究の位置付けは以下のとおりである. 最初のハイブリッドシステムの研究は, 1992 年に Z.Manna や A.Pnueli らにより, アナログ動作とデジタル動作をオートマトンの枠組みで扱い, そのオートマトン上で時相論理による証明理論を開発したものである [3]. 次に, 1993 年に, R.Alur や T.A.Henzinger らがハイブリッドオートマトンのモデル検査手法 [4] を開発して, 1996 年に Lynch らがハイブリッド I/O オートマトンの模倣関係の検証ルール [5] を開発した. その他の多数の研究は, ほとんど, この路線の踏襲であり, 我々の研究のように, 設計方法論へ発展させて完成

させるものはほとんど存在しない. さらに, モジュール性記述及び Assume-Guarantee style によるスケールアップ技術を基礎として, 定理証明器上に設計手法を実装して, 実用レベルの組込み型システムに適用して評価するものも存在しない.

本報告書の構成は以下のとおりである. まず, 2章では, ハイブリッドシステムの詳細化検証理論を提案する. 次に, 3章では, ハイブリッドシステムのモジュール記述による定理証明手法を提案する. 次に, 4章では, ハイブリッドモデルによるリアルタイムスケジューラの演繹的検証の事例を示す. 次に, 5章では, ハイブリッドモデルに基づく制御システムの開発手法を提案する. 次に, 6章では, 確率ハイブリッドシステムのモデル化と検証の手法を提案する. 最後に, まとめと今後の課題を述べる.

2　ハイブリッドシステムの詳細化検証理論

2.1　まえがき

組込み型システムのほとんどはアナログ環境に組み込まれたデジタルな実時間システムであり, アナログ動作とデジタル動作が混在している [1, 2]. また, 自動車や飛行機などの多くの組込み型システムは safety-critical な状況で動作するので, 形式的な開発方法論による信頼性保証が重要であり, 従来より多数の信頼性保証のための研究がなされている. 本章では, 形式的な開発方法論を実現するための基礎となる, ハイブリッドオートマトンの詳細化検証理論について述べる.

従来のハイブリッドシステムの詳細化検証に関する主要な研究には以下がある：

1. 1993 年に, Henzinger らが Pnueli らのフェーズ遷移システム [3] の詳細化検証の公理系を提案した [6]. これは, ハイブリッド時相論理を詳細化しながら, アナログ動作の詳細化を構成するものであり, フェーズ遷移システムの詳細化検証理論の構築である.
2. 1995 年に, Henzinger が Alur らのハイブリッドオートマトン [4] の双模倣関係の検証アルゴリズムを提案した [7]. これは, ハイブリッドオートマトンの自動検証の基礎となる双模倣関係を計算するアルゴリズムを開発したものである.
3. 1996 年に, Lynch らがハイブリッド I/O オートマトンの模倣関係の公理系を提案した [5]. これは, 抽象的なモデルであるハイブリッド I/O オートマトンの詳細化検証を可能とする基礎理論である.
4. 2000 年に, Alur や Henzinger らが Charon と Masaccio といったコンポーネントベースのハイブリッドシステムの詳細化検証を基礎とした開発方法論を提案している [8, 9]. Alur や Henzinger

らの方法論は現実のハイブリッドシステムの構造や動作を詳細化検証するものであるが, ソフトウェア構造を考慮したものではない.

5. 2001 年に, Henzinger らは Giotto と呼ばれる抽象的な組込み型プログラマモデルを開発しており, 組込み型プログラムを周期タスクなどから, どのように構成すべきかを定義している [10]. しかし, 上流工程からどのように Giotto を構成すべきかを論じていない. さらに, Giotto が上流工程を正しく詳細化しているかどうかを検証する手法も提案していない.

我々は, 詳細化写像の概念を用いて, アナログ動作ばかりでなく, デジタル動作も含めた詳細化検証理論を提案する.

以降の本論文の構成は以下のとおりである. 2 − 2 では, ハイブリッドシステムの仕様記述言語を定義する. 2 − 3 では, ハイブリッドシステムの詳細化検証理論を提案する. 最後に, 2 − 4 では, まとめと今後の課題を述べる.

2.2 ハイブリッドシステムの仕様記述言語

まず, 仕様記述言語である線形ハイブリッドオートマトン [26] を定義する :

Definition 1 (線形ハイブリッドオートマトン)

線形ハイブリッドオートマトンは, $\mathrm{A} = (\vec{x}, V, local, inv, dif, E, act, L, syn, E_{\Theta})$ の *10* つ組で定義される. ここで,

1. データ変数

有限なベクタ $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ は実数値のデータ変数である. $\vec{x}$ のサイズ n は A の次元と呼

ばれる.

データの状態はn-次元の実数空間R^nの点$\vec{s} = \{s_1, \ldots, s_n\}$であり, 各データ変数$x_i$に$s_i \in \mathrm{R}$を割り付ける関数である. 凸状データ領域は$\mathrm{R}^n$の凸状多面体である. データ領域は凸状データ領域の有限な和集合である. データ領域は結合されていて凸状閉包性を持つならばほとんど凸状である. 凸状のデータの述語は$\vec{x}$上の凸状の線形の論理式である. 凸状のデータの述語pは凸状のデータ領域$\ll p \gg \subseteq \mathrm{R}^n$である. ここで, $p[\vec{x} := \vec{s}]$が*true*のときに限り$\vec{s} \in \ll p \gg$である.

任意のデータ変数x_iに対して, x_iの第一階導関数を$\dot{x}_i$と表現する. 第一階導関数の包含はR^nの中の凸状の多面体である. 速度を表す述語は$\dot{\vec{x}}$上の凸状の線形の論理式である. 速度を表す述語rは第一階導関数の包含$\ll r \gg \subseteq \mathrm{R}^n$を定義する. ここで, $r[\dot{\vec{x}} := \vec{s}]$が*true*のときに限り$\vec{s} \in \ll r \gg$である.

任意のデータ変数x_iに対して, 状態遷移後のx_iの新しい値を$x_{i\prime}$と表記する. アクションを表す述語$q = (\vec{y}, q\prime)$は, 更新された変数の集合$\vec{y} \subseteq \vec{x}$と, データ変数$\vec{x}$と遷移後の更新変数$\vec{y}$の和集合の上の凸状の線形な論理式$q\prime$から構成される. アクションを表す述語$q$の閉包$\{q\}$は凸状の線形な論理式$q\prime \wedge \bigwedge_{x \in \vec{x} \setminus \vec{y}} (x\prime = x)$である：すなわち, 更新されない全データ変数は変化しないままである. アクションを表す述語qは状態変数から凸状のデータ領域への関数$\ll q \gg$である：すべてのデータ状態$\vec{s}, \vec{s}\prime \in \mathrm{R}^n$に対して, $\{q\}[\vec{x}, \vec{x}\prime := \vec{s}, \vec{s}\prime]$が*true*のときに限り$\vec{s} \in \ll q \gg (\vec{s})$である. もしデータ領域$\ll q \gg (\vec{s})$が非空ならば, アクションを表す述語$q$はデータ状態の中で実行可能である.

2. 制御ロケーション

ノードの有限集合 V は制御ロケーションである.

線形ハイブリッドオートマトン A の状態 $(v, \vec{s})$ は制御ロケーション $v \in V$ とデータ状態 $\vec{s} \in \mathrm{R}^n$ から構成される. 領域 $R = \bigcup_{v \in V}(v, S_v)$ は各制御ロケーションのデータ領域 $S_v \subseteq \mathrm{R}^n$ を集めたものである. 状態を表す述語 $\phi = \bigcup_{v \in V}(v, p_v)$ は各制御ロケーションのデータ述語 p_v を集めたものである. 状態を表す述語 ϕ は $\ll \phi \gg = \bigcup_{v \in V}(v, \ll p_v \gg)$ である. 状態を表す述語が存在するならば領域 R は線形である.

領域 $(v, S) \cup \bigcup_{v\prime \neq v}(v\prime, \emptyset)$ を (v, S) と書き, $(v, p) \cup \bigcup_{v\prime \neq v}(v\prime, false)$ を (v, p) と書く. 状態を表す述語を書くとき, 制御ロケーションの集合 V 上の範囲をとるロケーションカウンタ l を使う. ロケーション制約 $l = v$ は状態を表す述語 $(v, true)$ を示す. 状態を表す述語としてデータ述語を使うとき, データ述語 p は $\bigcup_{v \in V}(v, p)$ を示す. 2 つの状態を表す述語 $\phi = \bigcup_{v \in V}(v, p_v)$ と $\phi\prime = \bigcup_{v \in V}(v, p_v\prime)$ に対して, $\neg\phi = \bigcup_{v \in V}(v, \neg p_v)$, $(\phi \vee \phi\prime) = \bigcup_{v \in V}(v, p_v \vee p_v\prime)$, $(\phi \wedge \phi\prime) = \bigcup_{v \in V}(v, p_v \wedge p_v\prime)$ である.

3. ローカル変数

local はローカル変数の有限集合であり, 制御ロケーション V と有限なベクタ $\vec{x}$ の中で観測できない要素から構成される. ローカル変数は他のプロセスから観測できない変数である.

4. ロケーション不変性

各制御ロケーション $v \in V$ に凸状のデータ述語 $inv(v)$, つまり v の不変式を割り付ける関数である. オートマトン A の制御は不変式 $inv(v)$ が *true* である限り v に留まるので, 不変式 $inv(v)$ は一つのロケーションから別のロケーションへのシステムの前進を強制するために使われる. $\vec{s} \in \ll inv(v) \gg$ ならば状態 $(v, \vec{s})$ は存在しえる. A の存在しえる状態を Σ_A として, その状態述

語を $\bigcup_{v \in V}(v, inv(v))$ とする.

5. 連続的アクティビティ

各制御ロケーション $v \in V$ に速度を表す述語 $dif(v)$, つまり v のアクティビティを割り付ける関数は dif である. アクティビティはデータ変数の値が変化する速度を制約する. つまり, オートマトンの制御がロケーション v に存在するとき, すべてのデータ変数の第一導関数は導関数の包含 $\ll dif(v) \gg$ 内に存在する.

6. 遷移

エッジの有限多重集合 E は遷移と呼ばれる. 各遷移 $(v, v\prime)$ は遷移元ロケーション $v \in V$ と遷移先ロケーション $v\prime \in V$ を示す. 各ロケーション $v \in V$ に対して, 遷移 $e_v = (v, v)$ が存在する.

7. 離散的アクション

各遷移 $e \in E$ にアクション述語 $act(e)$ を割り付けるラベリング関数 act は e のアクションである. アクション $act(e)$ が実行可能のときにのみ, オートマトンの制御は $e = (v, v\prime)$ により, ロケーション v からロケーション $v\prime$ へ進む. もし $act(e)$ がデータの状態 $\vec{s}$ において実行可能ならば, すべてのデータ変数の値は非決定的に $\vec{s}$ からデータ領域 $\ll act(e) \gg (\vec{s})$ のある点に変化する. アクション述語は共有変数によってハイブリッドオートマトンを同期するために使われる.

8. 同期ラベルとラベリング関数

同期ラベルの有限集合は L であり, 各遷移 $e \in E$ に同期ラベルの部分集合を割り付けるラベリング関数は syn である. 集合 L は A のアルファベットと呼ばれる. 同期ラベルは 2 つのオートマトンの並列合成を定義するために使われる. つまり, 2 つのオートマトンが同期ラベル a を共有

するならば, 一方のオートマトンの a-遷移が他方のオートマトンの a-遷移によって遂行されなければならない.

9. エントリーエッジ

エントリーエッジ E_Θ は遷移元のロケーションは存在しないが, エントリーロケーション $v_i \in V$ がある. E_Θ は $act(E_\Theta)$ によりラベル付けされている.

■

次に, ハイブリッドオートマトンの並列合成を定義する.

Definition 2 (線形ハイブリッドオートマトンの並列合成) 2つの線形ハイブリッドオートマトン $A_1 = (\vec{x}_1, V_1, local_1, inv_1, dif_1, E_1, act_1, L_1, syn_1, E_{\Theta 1})$ と $A_2 = (\vec{x}_2, V_2, local_2, inv_2, dif_2, E_2, act_2, L_2, syn_2, E_{\Theta 2})$ が与えられたとする. ここで, A_1 と A_2 の次元は各々 n_1 と n_2 である. A_1 と A_2 の並列合成は, $A = (\vec{x}_1 \cup \vec{x}_2, V_1 \times V_2, local_1 \cup local_2, inv, dif, E, act, L_1 \cup L_2, syn, E_\Theta)$ である.

1. $V_1 \times V_2$ の各ロケーション $(v, v\prime)$ は不変式 $inv(v, v\prime) = inv_1(v) \wedge inv_2(v\prime)$ と $dif(v, v\prime) = dif_1(v) \wedge dif_2(v\prime)$ を持つ.

2. $E_\Theta = E_{\Theta 1} \wedge E_{\Theta 2}$ である.

3. 以下のいずれかのときに限り, E は $e = ((v_1, v_1\prime), (v_2, v_2\prime))$ を含む:

(a) $v_1 = v_1\prime$, かつ, $L_1 \cap syn_2(e_2) = \emptyset$ を有する $e_2 = (v_2, v_2\prime) \in E_2$ が存在する.

このとき, $act(e) = act_2(e_2)$ かつ $syn(e) = syn_2(e_2)$ である.

(b) $L_1 \cap syn_2(e_2) = \emptyset$ を持つ $e_2 = (v_2, v_2\prime) \in E_2$ が存在して, かつ, $v_2 = v_2\prime$ である.

このとき, $act(e) = act_1(e_1)$ かつ $syn(e) = syn_1(e_1)$ である.

(c) $syn_1(e_1) \cap L_2 = syn_2(e_2) \cap L_1$ を持つ $e_1 = (v_1, v_1\prime) \in E_1$ と $e_2 = (v_2, v_2\prime) \in E_2$ が存在する.

このとき, $act_1(e_1) = (\vec{y_1}, q_1\prime)$ かつ $act_2(e_2) = (\vec{y_2}, q_2\prime)$ ならば, $act(e) = (\vec{y_1} \cup \vec{y_2}, q_1\prime \wedge q_2\prime)$ かつ $syn(e) = syn_1(e_1) \cup syn_2(e_2)$ である.

■

2.3 詳細化検証手法

本論文における詳細化検証では, ハイブリッドオートマトンをフェーズ遷移システムに変換して, 詳細化検証の公理系により行う.

2.3.1 準備

まず, システム変数の有限集合 Var を考える. システム変数は整数や実数などの型を持つ. ここで, $\mathbf{R}^n$ を実数の集合とする. 状態は $\sigma : Var \to \mathbf{R}^n$ であり, Var の中の変数の型整合解釈である. 状態の集合を Σ_V とする. なお, 変数の型としては整数やブール値などがあるが, いずれも実数の部分集合なので, 状態を $\sigma : Var \to \mathbf{R}^n$ とする.

時間は非負実数 R^+ によってモデル化される. 区間 $I = [a, b)$ は $a \leq t < b$ であるような点 $t \in R^+$ の集合である. ここで, $a, b \in R^+$ かつ $a < b$ である. $I = [a, b)$ を区間とする. 関数 $f : I \to \mathbf{R}^n$ は, 以下の条件を満たすならば, I 上で各部分がスムーズである :

1. a において, f の右極限とすべての右導関数が存在する.

2. すべての点 $t \in I$ において, f の右極限と左極限, すべての右導関数と左導関数が存在して, f

は右または左から連続である.

3. b において, f の左極限とすべての左導関数が存在する.

2つの関数 $f, g : I \to \mathrm{R}^n$ は有限の点以外のすべてにおいて一致するならば, I 上において区別不能である. もし, 各部分がスムーズな2つの関数が開区間 I 上で区別不能ならば, 2つの関数は I における, すべての極限と導関数において一致する.

変数 V 上のフェーズ $P =< I, f >$ は以下の順序対から構成される：

1. 区間 $I = [a, b)$ である.
2. 関数 $f_x : I \to \mathrm{R}^n$ の型整合族は $f = \{f_x | x \in Var\}$ である. なお, 関数 $f_x : I \to \mathrm{R}^n$ は区間 $I = [a, b)$ 上で各部分がスムーズであり, 各点 $t \in I$ に変数 $x \in Var$ の値を割り付ける.

フェーズ P はすべての実数値の時間 $t \in I$ に状態 $f(t) \in \Sigma_V$ を割り付ける.

以下では, フェーズ P の左終端の極限状態 $\overrightarrow{P} \in \Sigma_V$ を

$$\overrightarrow{P} = lim_{t \to a}\{f(t) | a < t < b\} \tag{1}$$

と書き, フェーズ P の右終端の極限状態 $\overleftarrow{P} \in \Sigma_V$ を

$$\overleftarrow{P} = lim_{t \to b}\{f(t) | a < t < b\} \tag{2}$$

と書く.

2.3.2 フェーズ遷移システム

まず, Pnueli らのフェーズ遷移システム [3] を拡張した, フェーズ遷移システムを定義する. このフェーズ遷移システムは観測可能な変数と観測不能な変数を区別するものである.

Definition 3 (フェーズ遷移システム)

フェーズ遷移システムは, $\mathrm{M} = (Var, L, \Phi, \Theta, \mathcal{T})$ の 5 つ組で定義される. ここで,

1. Var はシステム変数の有限集合である.
2. $L \subseteq Var$ はローカル変数であり, システムの外部からは観測できない変数である.
3. Φ は Var 上のフェーズの普遍式の有限集合である. 各フェーズ普遍式 $\phi \in \Phi$ は表明式 $\rho_\phi(Var, \dot{Var})$ によって表現される. ここで, $\dot{Var}$ はシステム変数 Var の導関数である.
4. Θ は初期条件である. これは, すべての初期状態を特徴付ける表明である.
5. $\mathcal{T}$ は状態遷移の有限集合である. 各状態遷移 $\tau \in \mathcal{T}$ は表明式 $\rho_\tau(Var, Var\prime)$ によって表現される.

■

$\overline{P} = P_0, P_1, P_2, \ldots$ は無限なフェーズ列である. ここで, すべての $i \geq 0$ に対して, $P_i =< [a_i, a_{i+1}), f_i >$ である.

フェーズ列が以下の条件を満たすフェーズ列 $\overline{P} = P_0, P_1, P_2, \ldots$ と等しければ, このフェーズ列はフェーズ遷移システム M の計算である :

1. 初期条件 : もし $P_0 = [a, b)$ ならば Θ は a で成り立つ.
2. 連続動作 : すべての $0 \leq i < |\overline{P}|$ に対して, P_i が ϕ-フェーズであるようなフェーズ普遍式 $\rho_\phi \in \Phi$

が存在する.

3. 離散的状態遷移：すべての $0 \leq i < |\overrightarrow{P}| - 1$ に対して, $\rho_\tau(\overrightarrow{P_i}[Var], \overrightarrow{P_{i+1}}[Var])$ が成り立つような状態遷移 $\tau \in \mathcal{T}$ が存在する.

4. 発散性：$\overrightarrow{P}$ は発散する.

次に, ハイブリッドオートマトンからフェーズ遷移システムへの変換方法を定義する.

Definition 4 (ハイブリッドオートマトンの変換方法)

線形ハイブリッドオートマトン $\mathbf{A} = (\vec{x}, V, local, inv, dif, E, act, L, syn, E_\Theta)$ は, 以下のように, フェーズ遷移システム $\mathbf{M} = (Var, L, \Phi, \Theta, \mathcal{T})$ に変換される：

1. $Var = \pi \cup \vec{x}$

   ただし, π はロケーション V を表現するロケーション変数である. つまり, システム変数の有限集合 Var はロケーション変数 π とデータ変数 $\vec{x}$ である.

2. $L = local$

   つまり, ローカル変数 L はハイブリッドオートマトンのローカル変数 $local$ であり, $local \subseteq Var$ である.

3. $\Phi = \{\phi_{l_i} | l_i \in V\}$

   任意の $l_i \in V$ に対して,

   $$\rho_{l_i} = inv(l_i) \wedge (\phi_{l_i} = l_i) \wedge dif(l_i)$$

   である.

4. $\Theta = act(E_\Theta) \wedge (\pi = l_i)$

ここで, l_i はエントリーロケーションである.

5. $\mathcal{T} = \{\tau_{l_i,l_j} | (l_i, l_j) \in E\}$

ここで, $e = (l_i, l_j) \in E$ に対して,

$$\rho_{(l_i,l_j)} : act(e) \wedge \pi = l_i \wedge \pi\prime = l_j$$

である.

なお, 同期ラベルの集合 L とラベリング関数 syn は状態遷移の名前 $label \in L$ を定義するものであり, $syn(e) = label$ である.

■

2.3.3 詳細化検証の公理系

まず, 詳細化検証の基本概念を定義する. 図 1 − 1 の (1) と (2) のように, 抽象化仕様及び具体化仕様が与えられたとする. ハイブリッドシステムの詳細化検証では, フェーズ $\rho_\phi(Var, \dot{V}ar)$ と状態遷移 $\rho_\tau(Var, Var\prime)$ の詳細化を考慮する必要がある. 本論文では, 具体化仕様が抽象化仕様を詳細化するとは, 抽象化仕様のフェーズが具体化仕様のフェーズを包含して, すべての具体化仕様の状態遷移に対して, ある抽象化仕様の状態遷移が存在することを意味する. この詳細化検証の基本概念のイメージを図 1 − 1 (3) に示す. ただし, T はマスタークロックである. また, x は現在の x を意味して, $x\prime$ は次の時点の x を意味する.

ハイブリッドシステムは, $\mathrm{M} = (Var, L, \Phi, \Theta, \mathcal{T})$ の 5 つ組で定義される. $L \subseteq Var$ はローカル変数であり, システムの外部からは観測できない変数である. ハイブリッドシステムは分散並行システムの一種であり, プロセス代数 [15] と同様に, 観測可能な動作における等価性や模倣性などが重要であ

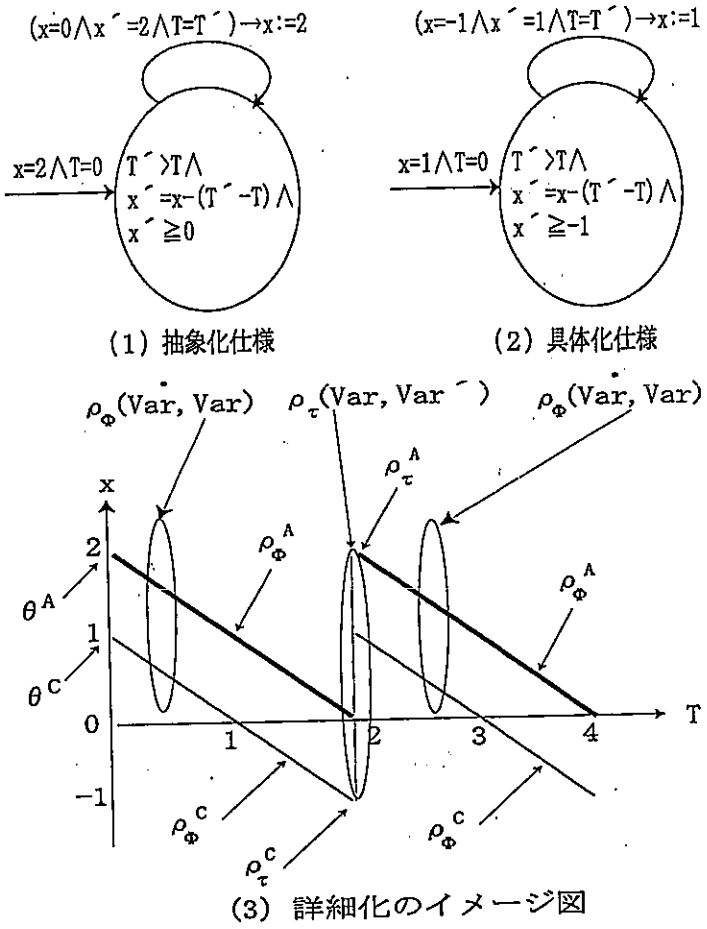


図 1: 詳細化検証の基本概念図

る. ゆえに, 本論文においても観測可能な動作における詳細化検証手法に焦点を当てる.

抽象化仕様 M^A の観測可能なフェーズが具体化仕様 M^C の観測可能なフェーズを包含して, すべての具体化仕様 M^C の状態遷移に対して, ある抽象化仕様 M^A の状態遷移が存在するならば, M^C は M^A を詳細化すると呼び, $\mathrm{M}^C \sqsubseteq \mathrm{M}^A$ と表記する. 以下に, 詳細化検証ルールを定義する :

Definition 5 (詳細化検証ルール)

$\mathrm{M}^C = (Var^C, L^C, \Phi^C, \Theta^C, \mathcal{T}^C)$ が $\mathrm{M}^A = (Var^A, L^A, \Phi^A, \Theta^A, \mathcal{T}^A)$ を観測可能な動作において詳細化していることを検証するルールを以下に示す :

なお, M^A の観測可能な変数 (つまり, $Var^A - L^A$) は M^C の観測可能な変数 (つまり, $Var^C - L^C$) である.

1. $\quad \Theta^C \rightarrow \Theta^A[\alpha]$
2. $\forall \phi^C \in \Phi^C$ に対して以下が成り立つ :
   $\rho_{\phi^C} \rightarrow \bigvee_{\phi^A \in \Phi^A} \rho_{\phi^A}[\alpha]$
3. $\forall \tau^C \in \mathcal{T}^C$ に対して以下が成り立つ :
   $\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$
4. ---
5. $\quad \mathrm{M}^C \sqsubseteq \mathrm{M}^A$

これは詳細化を証明するルールであり, ある α が存在して, 1 と 2 を満たすとき, 5 のように $\sqsubseteq$ が定義されることを意味する. ここで, ルールの各行は, 以下を意味する : 1 行目は初期条件 Θ^C ならば初期条件 $\Theta^A[\alpha]$ であることを意味する. 2 行目は ρ_{ϕ^C} のフェーズ普遍式が $\rho_{\phi^A}[\alpha]$ のフェーズ普遍式に包含されることを意味する. 3 行目は ρ_{τ^C} の状態遷移が抽象的な状態遷移 $\rho_{\tau^A}[\alpha]$ により説明されることを意味する. 4 行目は前提と結論を分離するラインである. 5 行目は結論 $\mathrm{M}^C \sqsubseteq \mathrm{M}^A$ であり, 具体化仕様が抽象化仕様を正しく詳細化していることを意味する.

ここで, α は L^A の式を Var^C の式に置き換える写像を意味する. なお, $X_\alpha(Var^C)$ により, Var^C の式を変数 $X \in L^A$ に割り当てることを表現する.

以下のように, α はフェーズ上の写像を与える：

P^C は M^C の計算の中に現れるフェーズとする. この P^C を具体化フェーズと呼ぶ. 具体化フェーズ P^C に対応する抽象化フェーズが $P^A = m_\alpha(P^C)$ であることは, P^A の中の任意の抽象化仕様のローカル変数 $X \in L^A$ の値が式 $X_\alpha(Var^C)$ の値であることを意味する. ここで, m_α は M^C から M^A への詳細化写像と呼ぶ. この詳細化写像は *Abadi* らの詳細化写像 *[12]* である.

■

2.4 むすび

本章では, 詳細化写像の概念を用いて, アナログ動作ばかりでなく, デジタル動作も含めた詳細化検証理論を提案した. 今後の研究課題しては以下が考えられる：

1. 大規模システムの詳細化検証を実現するために, Assume-Guarantete 方式の詳細化検証理論へ拡張する.
2. ハイブリッドシステムの開発では, ハイブリッドオートマトンだけではなく, UML などの複雑な仕様記述言語が使用されるので, それらの仕様記述に対する詳細化検証理論を開発する.

3 ハイブリッドシステムのモジュール演繹的検証

3.1 まえがき

ハイブリッドシステムはアナログ環境に組み込まれたデジタルな実時間システムである. 自動車や飛行機などの多くのハイブリッドシステムは safety-critical な状況で動作するので, 形式的な仕様記述と検証による正当性保証が重要である. 基本的なハイブリッドシステムの動作はフェーズ列である [3]. 各フェーズ内では, 状態遷移が発生するまで, システムの状態は, ダイナミックな制御法則（例えば, 微分方程式で表現される法則）に従って連続的に進化する. 状態遷移は連続的な状態の進化を分離する状態の変化である. ハイブリッドシステムの代表的な仕様記述言語及び計算モデルとして, フェーズ遷移システム [3] やハイブリッドオートマトン [4], ハイブリッド I/O オートマトン [5] が提案されている. まず, フェーズ遷移システム [3] は時相論理を演繹的検証するための計算モデルであり, 安全性や活性の検証ルールが開発されている. また, ハイブリッドオートマトン [4] は時間オートマトンをハイブリッドシステムに拡張した仕様記述言語であり, 時相論理の充足性に関する自動検証, いわゆるモデル検査が可能である. また, ハイブリッド I/O オートマトン [5] は模倣関係を検証するための計算モデルであり, 模倣関係の検証ルールが開発されている. しかし, 上記の演繹的検証手法や自動検証手法では, 多数の並行動作するハイブリッドオートマトンのカルテジアン積によりシステムを構成する必要があり, オートマトンの数の指数オーダーで状態組み合わせ爆発を発生させるために, 大規模ハイブリッドシステムの検証が困難である.

ハイブリッドシステムの検証とは, ハイブリッドシステムがある性質を充足するかどうかを判定することである. 一般的に, 検証すべき性質はハイブリッドシステムを構成する, すべてのモジュール

に直接, 関係するとは限らない. すなわち, 検証すべき性質があるモジュールの変数のみからなる表明式の場合が少なくない [13]. こういう場合では, ハイブリッドシステム全体を検証対象とする必要がなく, 対象とするモジュールのみを検証すればよい. しかし, 対象とするモジュールがその他のどんなモジュールと相互作用しても検証性質を満たすこと及び物理的に実装可能であることを保証する必要がある.

以上を踏まえて, 本論文では, 大規模ハイブリッドシステムの検証を実現するために, モジュール単位の形式的な仕様記述と検証を実現する. すなわち, ハイブリッドシステムを構成する, モジュール単位に仕様記述して, すべてのモジュールのカルテジアン積を構成しないで, 検証性質に直接関係するモジュール部分の安全性や活性を検証することによってシステム全体の検証を保証する手法を実現する. 本章で提案する手法を実現するために, 以下を開発する：

1. ハイブリッドシステムのモジュール単位の仕様記述を可能とするために, フェーズ遷移システム [3] を拡張して, フェーズ遷移モジュールを開発する.
2. ハイブリッドシステムのすべてのモジュールのカルテジアン積を構成しないで, モジュールの演繹的検証手法を開発する.
3. さらに, モジュールの検証を実現するために, モジュールの実装可能性, すなわち receptiveness の充足性の検証を実現する.

本章と同様な狙いの既存研究は存在しない. 類似研究としては, 以下の２つが存在する：１つ目は, ソフトウェアとハードウェアのコデザインにおける, 同期と非同期の計算のための統一的なフレームワークとして提案された状態遷移システム, いわゆるハイブリッドモジュール [14] の研究がある. ハ

イブリッドモジュール [14] の研究では, モジュール単位の仕様記述手法及び receptiveness の自動検証手法を提案している. しかし, 無限な状態数を有する状態遷移システムの検証には対応していないし, 時相論理式の検証問題には言及していない. 2つ目は, ハイブリッドシステムの模倣関係などの検証のために提案されたハイブリッドI/Oオートマトン [5] の研究がある. ハイブリッドI/Oオートマトン [5] の研究では, 模倣関係の検証ルール及び receptiveness の形式化を提案している. しかし, 安全性, 活性や receptiveness の検証手法を提案していないし, ハイブリッドI/Oオートマトンはハイブリッドシステムのトレース列の形式化であり, 我々のモデルよりも意味的なモデルである. ゆえに, 本研究はハイブリッドモジュール [14] 及びハイブリッドI/Oオートマトン [5] の研究と異なる.

以降の本章の構成は以下のとおりである. 3－2節ではモジュール単位の仕様記述を可能とするフェーズ遷移モジュールを提案する. 3－3節ではモジュール演繹的検証手法を提案する. 3－4節では提案手法の事例により有効性を示す. 最後に, 3－5節ではまとめと今後の課題を述べる.

3.2 フェーズ遷移モジュールによるモジュール単位の仕様記述

本節では, フェーズ遷移モジュールを定義する. ハイブリッドシステムのモジュールの仕様記述言語としてフェーズ遷移モジュールを定義する. 本論文では, 多数のモジュール（いわゆる並行プロセス）が並列動作して, ハイブリッドシステムが構成されるとする.

3.2.1 フェーズ遷移モジュールの定義

フェーズ遷移モジュールは, システム変数の集合を共有変数とローカル変数に区別することによって, フェーズ遷移システムを拡張する. まず, システム変数の有限集合を考える. システム変数は整

数や実数などの型を持つ. 我々はシステム変数にその型の値を割り付ける解釈として状態 s を定義する. すべての状態の集合を Σ とする.

Definition 6 (フェーズ遷移モジュール)

フェーズ遷移モジュールは, $\mathrm{FTM} = (V, \Theta, \mathcal{T}, \mathcal{A}, \Pi)$ の 5 つ組で定義される. ここで,

1. V : システム変数の有限集合. 集合 $V = D \cup I$ は, 離散変数の集合 $D = \{u_1, \ldots, u_n\}$ と連続変数の集合 $I = \{x_1, \ldots, x_m\}$ に分類できる. 離散変数は任意の型がありえるが, 連続変数は実数型である. さらに, リセットされないマスタクロック $T \in I$ を導入する. また, 集合 V は共有変数 (V^S) とローカル変数 (V^P) に分類できて, $V = V^P \cup V^S$ である. ローカル変数 (V^P) はモジュールのみが参照更新できる変数であり, 共有変数 (V^S) はモジュールの外部からも参照更新ができる変数である. ただし, マスタクロック $T \in I$ は共有変数である.
2. Θ : 初期条件. これは, すべての初期状態を特徴付ける表明である. $\Theta \rightarrow T = 0$ が要求される.
3. $\mathcal{T}$: 状態遷移の有限集合. 各状態遷移 $\tau \in \mathcal{T}$ は関数 $\tau : \Sigma \rightarrow 2^{\Sigma}$ であり, 各状態 $s \in \Sigma$ に次状態 τ-*successor* $\tau(s) \subseteq \Sigma$ を写像する. 状態遷移 τ に関連する関数は表明 $\rho_\tau(V, V\prime)$ により表現される. $\rho_\tau(V, V\prime)$ は状態遷移関係と呼び, 状態 $s \in \Sigma$ を τ-*successor* $s\prime \in \tau(s)$ に関係付ける. ただし, 状態 s は V の型整合解釈であり, 各変数 $v \in V$ に値 $s[v]$ を割り付ける. なお, システム変数の値は s の中の値や $s\prime$ の中の値として参照する. 任意の $\tau \in \mathcal{T}$ に対して, $\rho_\tau \rightarrow T\prime = T$ が要求される.
4. $\mathcal{A}$: アクティビティの有限集合. 各アクティビティ $\alpha \in \mathcal{A}$ はアクティビティ関係

$$p_\alpha \rightarrow I(t) = F^\alpha(V^0, t)$$

によって表現される. ここで, p_α は α の活性条件と呼ばれる, D 上の述語である. 活性条件 α が

s 上で成り立つならば, アクティビティ α は活性と言われる. $I = \{x_1, \ldots, x_m = T\}$ のとき, ベクトル式 $I(t) = F^\alpha(V^0, t)$ は以下の個々の式の集合の略記である：

$x_i(t) = F^\alpha{}_i(V^0, t)$ $(i = 1, \ldots, m)$

この式は, アクティビティ α に応じた, 連続的変化のフェーズの中の連続変数の進化を定義する. パラメータ V^0 はフェーズの開始点における, すべてのシステム変数の初期値を表現する. 任意の $\alpha \in \mathcal{A}$ に対して, 以下が要求される：

(a) $F^\alpha{}_i(V^0, 0) = x^0{}_i$ $(i = 1, \ldots, m)$

(b) $F^\alpha{}_T(V^0, t) = F^\alpha{}_m(V^0, t) = T^0 + t$

すなわち, $F^\alpha{}_i(V^0, 0)$ は x_i の初期値と一致して, マスタークロック T の長さ t の進化の効果は T に t が加算される. なお, t はフェーズが連続する時間である.

5. Π ：時間前進条件. これは時間前進の制限を表現するために使われる. つまり, 状態に割り付けられた変数の制約条件である.

■

具体的なハイブリッドシステムの仕様記述においては, 進化を表現する関数 $F^\alpha(V^0, t)$ は微分方程式 $\dot{x}_j = g^\alpha{}_j(V)$ $(j = 1, \ldots, m)$ の集合によって表現される. この場合, $F^\alpha(V^0, t)$ は微分方程式の解として得られる.

Example 1 (フェーズ遷移モジュールの例)

図 2 *(1)* のように, フェーズ遷移モジュール $\mathbf{FTM} = (V, \Theta, T, \mathcal{A}, \Pi)$ が与えられたとする. 図 2 *(1)* では, 状態遷移の枝は, " 条件/動作 " を意味しており, " /動作 " は無条件に動作することを意味し

て，"条件/"は動作無しに状態遷移することを意味する．例えば，$x=1/$ は $x=1$ ならば動作なしで状態遷移することを意味して，$x=-1/x:=1$ は $x=-1$ ならば x を -1 に更新して状態遷移することを意味する．ここで，各要素は以下のとおりである：

1. $V=D\cup I$ であり，$D=\emptyset$ と $I=\{x,T\}$ である．また，$V=V^P\cup V^S$ であり，$V^P=\{x\}$ と $V^S=\{T\}$ である．

2. $\Theta: x=1\wedge T=0.$

3. $\tau\in\mathcal{T}$ は以下のとおりである：

$$\rho_\tau: x=-1\wedge x\prime=1\wedge T\prime=T$$

4. $\alpha\in\mathcal{A}$ は以下のとおりである：

$$\underbrace{\left(true\right)}_{p}\rightarrow\underbrace{\left(x=x^0-t\right)}_{F(x^0,t)}$$

ただし，$x^0=1$ であり，t は Definition1*(4)(b)* の t であり，状態に存在する時間である．

5. $\Pi: x\geq -1.$ なぜならば，状態に割り当てられた変数の制約条件は $x\geq -1$ だからである．

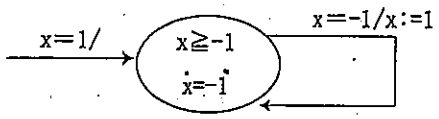
図2 *(2)* のように，フェーズ遷移モジュールは動作する．

■

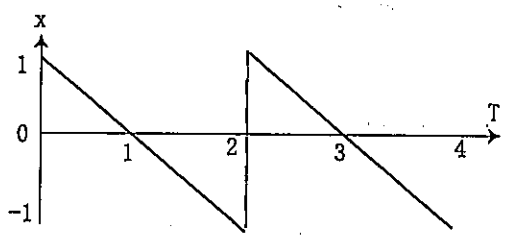
また，フェーズ遷移モジュール $\mathbf{FTM}=(V,\Theta,\mathcal{T},\mathcal{A},\Pi)$ に対して，以下のように，状態遷移 $\mathcal{T}$ を $\mathcal{T}_H$ に拡張する：

$\mathcal{T}_H=\mathcal{T}\cup\mathcal{T}_\Phi$，ここで $\mathcal{T}_\Phi=\{\tau_\alpha|\alpha\in\mathcal{A}\}$.

任意の $\alpha\in\mathcal{A}$ に対して，τ_α の状態遷移関係は以下のように与えられる：



(1) フェーズ遷移モジュールの仕様記述例



(2) フェーズ遷移モジュールの動作

図 2: フェーズ遷移モジュールの例

$$\rho_{\tau_\alpha} : \exists \triangle > 0. \left(\begin{array}{c} D\prime = D \wedge p_\alpha \wedge I\prime = F^\alpha(V, \triangle) \\ \wedge \\ \forall t \in [0, \triangle). \Pi(D, F^\alpha(V, t)) \end{array} \right)$$

ρ_{τ_α} は α-フェーズの開始点と終了点におけるシステム変数の可能な値を特徴付ける. ここで, $V = (D, I)$ はフェーズの開始点の値を示して, $V\prime = (D\prime, I\prime)$ はフェーズの終了点の値を示す. 上記論理式では, 正の時間の増加である, フェーズの長さ $\triangle$ を仮定している. 上記論理式は以下を意味している : 離散変数の値は保存されて $(D\prime = D)$, 活性条件 p_α は現在成り立っており, フェーズの終了点における連続変数の値は $F^\alpha(V, \triangle)$ により与えられて, $0 \leq t \leq \triangle$ の任意の t に対して時間前進条件 Π は成り立つ.

ここで, 拡張されたフェーズ遷移モジュール $\mathrm{FTM} = (V, \Theta, \mathcal{T}_H, \mathcal{A}, \Pi)$ の動作列は, 以下の条件を満たす無限の状態列 $\sigma : s_0, s_1, \ldots$ である :

1. $s_0 \models \Theta$ である.

2. 任意の $j \geq 0$ に対して, $s_{j+1} \in \tau(s_j)$ であるような $\tau \in \mathcal{T}_H$ が存在する.

ある状態がFTMの中の状態列に現れるならば, その状態はFTMのアクセス可能な状態と呼ばれる.

さらに, FTMの動作列のマスタークロック T の値が無限に大きくなるならば, 動作列はFTMの計算と呼び, $Comp(\mathrm{FTM})$ と表記する.

また, フェーズ遷移モジュール $\mathrm{FTM} = (V, \Theta, \mathcal{T}_H, \mathcal{A}, \Pi)$ は, モジュールの外部の環境（ここで環境とは, 対象としているモジュール以外はすべて環境である）の状態遷移 τ_E を追加することにより, 閉じたシステム $\mathrm{S_{FTM}} = (V^*, \Theta, \mathcal{T} \cup \{\tau_E\}, \mathcal{A}, \Pi)$ として解釈できる. ここで, V^* はモジュールと環境の変数の集合である. また, τ_E を表現する状態遷移関係 ρ_{τ_E} は以下のように定義される：

$$\rho_{\tau_E} : \Big(\bigwedge_{v \in V^P} (v = v\prime) \Big) \wedge (T = T\prime)$$

なお, $Comp(\mathrm{FTM}) = Comp(\mathrm{S_{FTM}})$ である.

3.2.2 フェーズ遷移モジュールの並列合成

次に, プロセス代数 [15] の並列合成と同様に, フェーズ遷移モジュールの並列合成は以下のように定義される.

Definition 7 (フェーズ遷移モジュールの並列合成)

2つのフェーズ遷移モジュール $\mathrm{FTM}_1 = (V_1, \Theta_1, \mathcal{T}_1, \mathcal{A}_1, \Pi_1)$ と $\mathrm{FTM}_2 = (V_2, \Theta_2, \mathcal{T}_2, \mathcal{A}_2, \Pi_2)$ が与えられたとき, 並列合成 $\mathrm{FTM}_1 \parallel \mathrm{FTM}_2$ はフェーズ遷移モジュール $\mathrm{FTM} = (V, \Theta, \mathcal{T}, \mathcal{A}, \Pi)$ として

以下のように定義される．ここで，

1. $V = V_1 \cup V_2$, $D = D_1 \cup D_2$, $I = I_1 \cup I_2$, $V^P = {V_1}^P \cup {V_2}^P$, $V^S = {V_1}^S \cup {V_2}^S$

2. $\Theta = \Theta_1 \wedge \Theta_2$

3. $\mathcal{T}$ は以下のように定義される：

(a) ${\mathcal{T}_1}$ の中の遷移 τ に対して，そのラベルが ${\mathcal{T}_2}$ に現れないとき

$$\rho_\tau \wedge \Big(\bigwedge_{v \in {V_2}^P} (v = v\prime) \Big)$$

また，${\mathcal{T}_2}$ の中の遷移 τ に対して，そのラベルが ${\mathcal{T}_1}$ に現れないとき

$$\rho_\tau \wedge \Big(\bigwedge_{v \in {V_1}^P} (v = v\prime) \Big)$$

(b) $\tau_1 \in \mathcal{T}_1$ と $\tau_2 \in \mathcal{T}_2$ が同じラベルを持つとき

$$\rho_{\tau_1} \wedge \rho_{\tau_2}$$

4. $\mathcal{A} = \mathcal{A}_1 \wedge \mathcal{A}_2$

5. $\Pi = \Pi_1 \wedge \Pi_2$

■

Example 2 (フェーズ遷移モジュールの並列合成の例)

図3 *(1)* のように，2つのフェーズ遷移モジュール $\mathrm{FTM}_1 = (V_1, \Theta_1, \mathcal{T}_1, \mathcal{A}_1, \Pi_1)$ と $\mathrm{FTM}_2 = (V_2, \Theta_2, \mathcal{T}_2, \mathcal{A}_2, \Pi_2)$ が与えられたとき，並列合成のフェーズ遷移モジュール $\mathrm{FTM} = (V, \Theta, \mathcal{T}, \mathcal{A}, \Pi)$ を定義する．図3 *(2)* に並列合成のフェーズ遷移モジュールを示す．

■

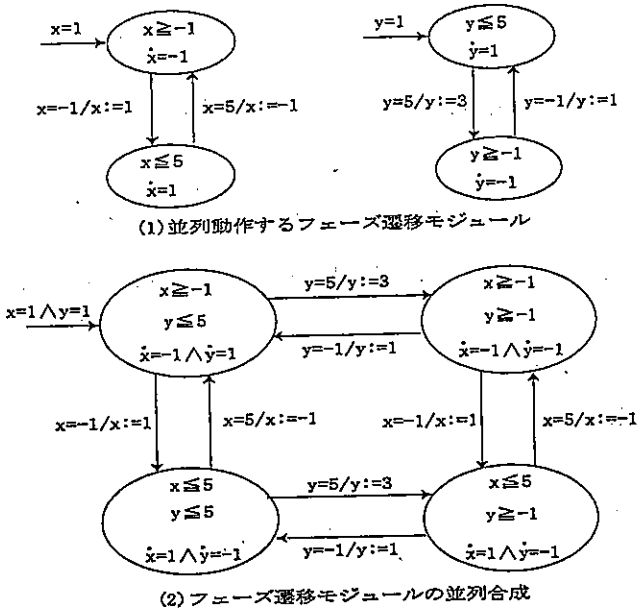


図 3: フェーズ遷移モジュールの並列合成の例

3.3 モジュールの演繹的検証手法

本論文では, 検証性質が直接関与する, モジュールのみを検証するといったモジュール演繹的検証を提案する. その場合, 対象とするモジュールがその他のどんなモジュールと相互作用しても検証性質を満たすこと及び物理的に実装可能であることを保証する必要がある. その他のどんなモジュールと相互作用しても検証性質を満たすことを保証するために, モジュールと環境の遷移に関して検証を行う. また, 物理的に実装可能であることを保証する性質は receptiveness [16] という性質である. 本節では, フェーズ遷移モジュールに関して, 安全性や活性のモジュール演繹的検証手法を開発する. 最初に receptiveness [16] の演繹的検証を定義して, 次にモジュール演繹的検証手法を提案する.

ハイブリッドシステムはリアクティブシステムであり, 環境と相互作用して, 環境の任意の動作に正しく反応しなければならない. なお, ハイブリッドシステムのモジュールでは, モジュール以外の

すべての部分が環境となる. モジュールが環境の任意の動作に反応することはモジュールと環境との無限ゲームとして形式化できて, 無限ゲームにおいてモジュールが環境に勝つとき, モジュールは環境に対して正しく動作すると考える [17]. このような, モジュールが環境に対して正しく動作する条件を receptiveness と呼び, receptiveness はモジュールが物理的に実装可能な条件である.

3.3.1 Receptiveness の演繹的検証

ハイブリッドシステムのモジュールの動作の正当性を保証する概念として, 並列演算閉包性が存在し, Nonzeno を保証する receptiveness を使用する [16, 18]. モジュールは, 以下のいずれかの場合に receptive であると言う.

1. モジュールの経過時間が発散する. つまり, モジュールの状態遷移が無限回起きると, モジュールの経過時間も無限に大きくなる.
2. モジュールの状態遷移が有限回起きる. つまり, 有限または無限の時間に, モジュールが有限回動作する.

Definition 8 (Receptiveness の定義)

receptiveness はモジュールと環境（対象としているモジュール以外のモジュール）との無限ゲームの下で, モジュールに対して定義される. ここで, モジュール及び環境は以下のように動作する：モジュールが状態遷移 τ_{con} を行うか, 時間経過 $\triangle_{mod}$ （モジュールの遷移 τ_α による経過時間）を行う. 環境が状態遷移 τ_E や *external* 遷移 τ_{ext} を行うか, 時間経過 $\triangle_{env}$（環境の遷移 τ_α による経過時間）を行う. なお, *external* 遷移 τ_{ext} とは, 環境により制御されるモジュール内の遷移である. ただし, モ

ジュールの状態遷移 τ は, モジュールが提案する状態遷移 τ_{cont} と *external* 遷移 τ_{ext} の和集合である (つまり, $\tau = \tau_{con} \cup \tau_{ext}$). また, 状態遷移 τ_{con} は *controlled* 遷移と呼ばれることがある. なお, *external* 遷移は *I/O* オートマトン *[18]* の入力イベントによる遷移に対応し, *controlled* 遷移は *I/O* オートマトン *[18]* の出力イベントによる遷移に対応する. 無限ゲームにおいては, 以下のように, 次の動作が決定される :

1. もし環境が状態遷移 τ_E や *external* 遷移 τ_{ext} を提案するならば, 環境の状態遷移 τ_E や *external* 遷移 τ_{ext} が選択される. これは環境の動作である.
2. もし環境が時間経過 $\triangle_{env}$ を提案してモジュールが状態遷移 τ_{con} を提案するならば, モジュールの状態遷移 τ_{con} が選択される. これはモジュールの動作である.
3. もし環境とモジュールがともに時間経過 $\triangle_{mod}, \triangle_{env}$ を提案するならば, 小さい時間経過 $min\{\triangle_{mod}, \triangle_{env}\}$ が選択される. これは, $\triangle_{mod} \leq \triangle_{env}$ ならばモジュールの動作であり, そうでなければ環境の動作である.

その様子を図 4 に示す. 簡単に言えば, 上記無限ゲームに従って動作するモジュールの経過時間が発散するか, またはモジュールの状態遷移 τ が有限回ならば, モジュールは勝つ. 形式的には, 任意の環境に対して, モジュールが勝つ戦略を持つならば, そのモジュールは *receptive* である. なお, モジュールの戦略とは, モジュールの任意の到達可能な状態に対して, *controlled* 遷移か $\triangle_{mod}$ 遷移を割り付ける関数である. ■

フェーズ遷移モジュールが receptive であることを示すために, 我々は, Manna らがクロック遷移モジュールの中で開発した Receptiveness ダイヤグラム [16] を使用する.

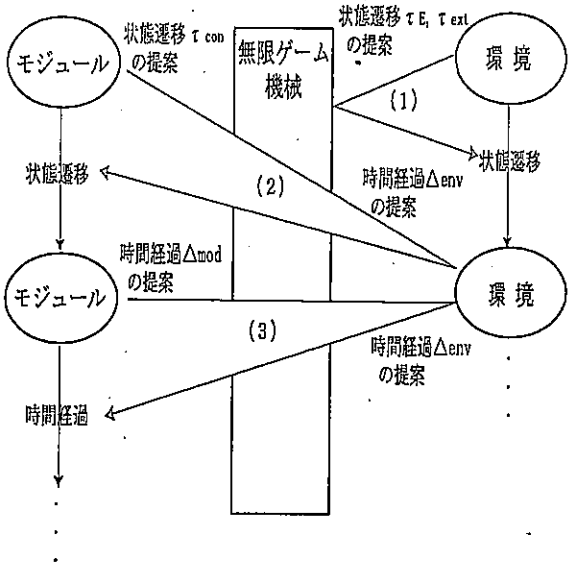


図 4: 無限ゲームの概念図

Definition 9 (Receptiveness ダイヤグラム)

フェーズ遷移モジュール FTM の *Receptiveness* ダイヤグラムは有向グラフ $\mathcal{D}$ である．ここで，任意のノードは表明 φ_n とランキング関数 δ_n によりラベル付けされている．なお，ランキング関数 δ_n はフェーズ遷移モジュール FTM のローカル変数 V^P にのみ依存しており，その値域は整礎順序集合である．なお，ランキング関数 δ_n は，ある到達する状態と現在の状態との距離を表示する関数であり，状態遷移が有限回しか発生しないことを示すときに使う．*Receptiveness* ダイヤグラム $\mathcal{D}$ には，以下のような第一階の検証条件が存在する：

1. 初期条件：

   $(T = t_0) \rightarrow \bigvee_{n \in Nodes(\mathcal{D})} \varphi_n$．ここで，$t_0$ は *Skolem* 定数であり，$Nodes(\mathcal{D})$ は有向グラフ $\mathcal{D}$ のノードの集合である．

2. モジュールの動作：

有向グラフ $\mathcal{D}$ の任意のノードに対して，*(I)* τ_α 遷移（モジュールの時間経過）又はランキング関数 δ_n を減少させる状態遷移 τ_{con}（モジュールが提案する状態遷移），または，*(II)* $t_0+\epsilon$ を超えて時間経過する τ_α 遷移（モジュールの時間経過）がラベル付けられている．ここで，t_0 は *Skolem* 定数であり，ϵ は 0 より大きな定数である．

なお，*(I)* はモジュールが有限回動作する場合を示して，*(II)* はモジュールの経過時間が発散する場合を示す．上記 *(I)* において，ランキング関数 δ_n を減少させる状態遷移 τ_{con} の理由は，*(I)* がモジュールの有限回動作する場合の条件を意味しているからである．

形式的には以下のように表現できる：

$$\{\varphi_n\}\ \tau_\alpha\ \exists\{T \geq t_0+\epsilon\}$$

$$\vee$$

$$\bigvee_{e_m=<n,m>\in COout(n)} \{\varphi_n \wedge \delta_n = u\}\ \tau_{con}(e_m)\ \exists\{\varphi_m \wedge \delta_m \prec u\}$$

ここで，$COout(n)$ は τ_α 遷移（モジュールの時間経過）か状態遷移 τ_{con}（モジュールが提案する状態遷移）によりラベル付けされる，ノード n から出るノードの集合である．

3. 環境の動作：

すべてのノード n と τ_α，環境 τ_E か *external* 遷移 τ_{ext} の $\tau\prime$ に対して，$\tau\prime$ により φ_n が保存される，または，$\tau\prime$ が $\tau\prime(n)$ の中の一つに遷移する．形式的には以下のように表現できる：

$$\{\varphi_n \wedge \delta_n = u\}\ \tau\prime\ \{(\varphi_n \wedge \delta_n \preceq u) \vee \Big(\bigvee_{m\in N(\tau\prime,n)} (\phi_m \wedge \delta_m \preceq u)\Big)\}$$

ここで，$N(\tau\prime,n)$ は，状態遷移 $\tau\prime$ により n から m に遷移するようなノード m の集合である．た

だし, $\tau I = \tau_E \cup \tau_{ext}$ である.

■

Receptiveness ダイヤグラムの第一階の検証条件のすべてが恒真ならば, Receptiveness ダイヤグラムは FTM-恒真と呼ぶ. Receptiveness ダイヤグラムが FTM-恒真ならば, FTM は receptive である. この証明の方針は, Manna らの実時間システムの演繹的検証に関する論文の中の証明 [16] と同様に, 以下のとおりである : Receptiveness ダイヤグラムが FTM-恒真ならば, モジュールが勝つ戦略を持つことを示す. すなわち, 恒真な検証条件により, モジュールの経過時間が発散するか, または, モジュールの状態遷移が有限回起きることを示す. 紙面の都合上から, この証明の詳細は省略する.

以下に, Receptiveness の証明例を示す.

Example 3 (Receptiveness の証明例)

Receptiveness の証明を簡単な例を用いて説明する.

図 5 は, フェーズ遷移モジュールにより, アルコールランプの動作を仕様記述したものである.

アルコールランプでは, 離散変数 x により, $lightoff$ はランプが消えていること, $lighton$ はランプがついていること, out はアルコールがないことを示す. また, 連続変数 y により, アルコールの量を示す. $x = lightoff$ ではアルコールが揮発してアルコール量が減少して, $x = lighton$ ではアルコールが燃えてアルコール量が減少する. また, 離散変数 z により, on は人間がアルコールランプを着けること, off は消すこと, $full$ はアルコールを入れることを示す. なお, 状態遷移の枝は, 条件/動作を意味しており, /動作は無条件に動作することを意味して, 条件/は動作無しに状態遷移することを意味する. 以上のアルコールランプの *Rcecptiveness* を証明する. 以下に証明図を示す.

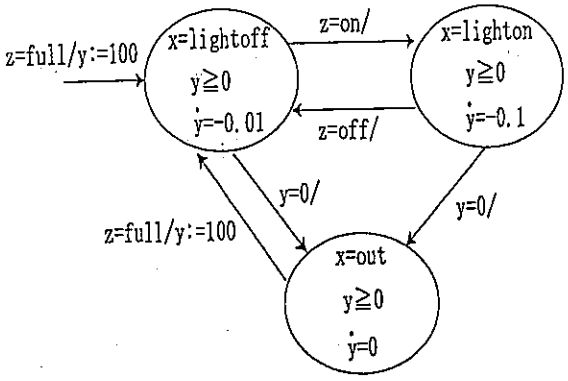


図 5: アルコールランプの仕様記述例

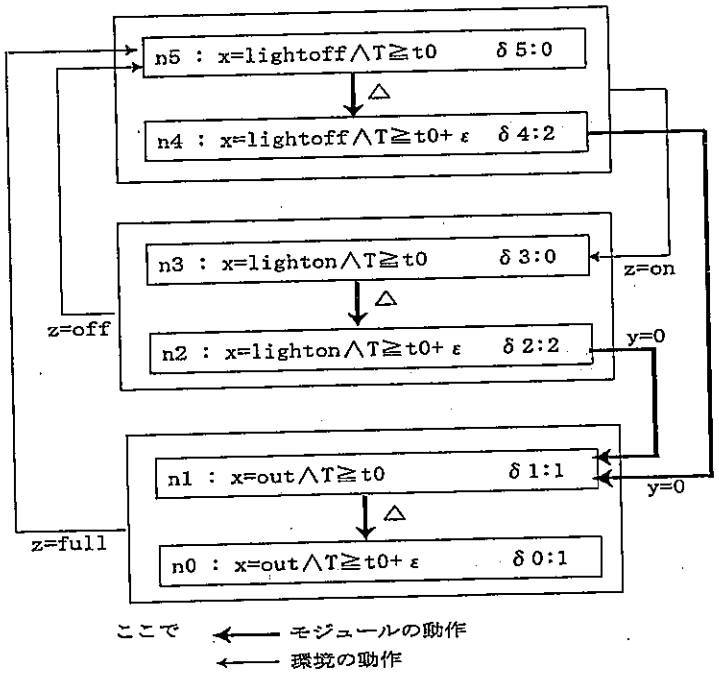


図 6: アルコールランプの Receptiveness の証明図

図6を以下に説明する．t_0 はアルコールランプが動作を開始した瞬間のマスタークロック T の値である．モジュールの動作は太線の矢印で示し，環境の動作は細線の矢印で示す．

アルコールランプの *receptiveness* の証明図は以下のように FTM-恒真である：

1. 初期条件：初期ノード n_5 では，$\triangle$ が選ばれるならば，$t_0 + \epsilon$ 以上になる．
2. モジュールの動作：ノード n_4, n_3, n_2, n_1 では，$\triangle$ または *controlled* 遷移が選ばれる．$\triangle$ が選ばれるならば，$t_0 + \epsilon$ 以上になる．*controlled* が選ばれるならば，ランキング関数 $\delta_i (i = 0, \ldots, 4)$ は減少する．
3. 環境の動作：*external* または τ_E の動作は，n_5 または n_4 から n_3 への遷移，n_1 または n_0 から n_5 への遷移，n_3 または n_2 から n_5 への遷移であり，ランキング関数 $\delta_i (i = 0, \ldots, 5)$ は減少する．

よって，アルコールランプが *Receptiveness* を満たすことが証明できた．

■

3.4 フェーズ遷移モジュールの演繹的検証手法

3.4.1 時相論理

まず，時相論理 [19] の部分集合であり，公理系で証明する時相論理 [20] の構文を定義する．

Definition 10 (時相論理の構文)

時相論理式の集合は，以下の規則で生成される論理式の集合のうち最小のものである：

1. 各原子命題 P は論理式である．ただし，原子命題はフェーズ遷移モジュールのシステム変数の集

合への値の割り当てにより定義される表明式である.

2. p が論理式ならば, $\Box\, p$ は論理式である.

3. p と q, r が論理式ならば, $\Box\ (p \rightarrow (qWr))$ は論理式である.

4. p と r が論理式ならば, $\Box\ (p \rightarrow \Diamond r)$ は論理式である.

■

時相論理式の直感的意味としては以下のとおりである: $\Box\, p$ は常に p が成り立つことを意味する. $\Box\ (p \rightarrow (qWr))$ は常に p ならば r が成り立つまで q が成り立つことを意味する. $\Box\ (p \rightarrow \Diamond r)$ は常に p ならばいつかは r が成り立つことを意味する.

次に, 時相論理の意味を形式的に定義する.

Definition 11 (時相論理の意味)

状態 s と論理式 p に対して, p が s 上で成り立つことを $s \models p$ と書く. 状態の無限列 $\sigma = s_0, s_1, s_2, \cdots$ をモデルとする. モデル σ に対して, 位置 $j \geq 0$ で p が成り立つことを $\sigma, j \models p$ と書く. モデル上における論理式の充足関係は, 以下のように定義できる:

1. $\sigma, j \models p$ *iff* $s_j \models p$.

2. $\sigma \models \Box\, p$ *iff* すべての j に対して $s_j \models p$ である.

3. $\sigma \models \Box\ (p \rightarrow (qWr))$ *iff* すべての i に対して $s_i \models p$ ならば, すべての $j \geq i$ に対して $s_j \models r$, またはある $k \geq i$ に対して $s_k \models r$ かつすべての j $(i \leq j < k)$ に対して $s_j \models q$ である.

4. $\sigma \models \Box\ (p \rightarrow \Diamond r)$ *iff* すべての i に対して $s_i \models p$ ならば, $j \geq i$ に対して $s_j \models r$ である.

FTM の任意の状態列 σ に対して $\sigma \models \varphi$ ならば, 時相論理式 φ は FTM 上で恒真であると呼ばれる. 表明式 p が FTM の任意のアクセス可能な状態で成り立つならば, 表明式 p は FTM の状態恒真であると呼ばれる.

3.4.2 演繹的検証ルール

本節では, フェーズ遷移モジュールに関する時相論理式の演繹的検証ルールを定義する. フェーズ遷移システムの安全性や活性の検証ルールは Kesten や Manna, Pnueli によって開発されている [20]. 本論文では, Kesten や Manna, Pnueli が開発した検証ルールをフェーズ遷移モジュールへ拡張する. フェーズ遷移システムの検証ルールとフェーズ遷移モジュールの検証ルールは同一の形式をしているが, 以下の 2 点が異なる :

1. 表明式はフェーズ遷移モジュールから構成する.
2. 検証ルールはフェーズ遷移モジュール及びその環境の状態遷移に関係する.

文献 [20] では, 安全性として不変性と waiting-for 性質の検証ルールが提案されており, 本論文では紙面の都合上から, 不変性の検証ルールのみを以下に示す. また, 他の安全性や活性の検証についても不変性と同様に, フェーズ遷移システムの安全性や活性の検証ルールをフェーズ遷移モジュールに拡張して検証ルールを構成できる.

まず, フェーズ遷移モジュール $\mathrm{FTM} = (V, \Theta, \mathcal{T}, \mathcal{A}, \Pi)$ が不変性 $\Box p$ を充足することを検証するための不変性の検証ルールを定義する.

Definition 12 (不変性の検証ルール)

表明 φ と p に対して, 以下である.

$$\frac{\begin{array}{l}1.\ \Theta \rightarrow \varphi \\ 2.\ \varphi \rightarrow p \\ 3.\ \rho_\tau \wedge \varphi \rightarrow \varphi\prime \quad (\forall \tau \in \mathcal{T}_H \cup \{\tau_E\})\end{array}}{4.\ \Box p}$$

これは不変性を証明するルールであり, 各記号の意味は以下のとおりである：

1. Θ は初期条件である.

2. p は不変的に成り立つ論理式である.

3. φ は p より強い表明式であり, 実際の検証では φ の不変性を証明する.

4. ρ_τ は状態遷移関数 τ を意味する表明式である.

5. $\varphi\prime$ は状態遷移 ρ_τ 後の φ であり, φ の中の変数に $\prime$ を付けたものである.

また, ルールの各行は, 以下を意味する：

1. 1行目は初期条件 Θ ならば表明 φ が成り立つことを意味する.

2. 2行目は表明 φ ならば表明 p が成り立つことを意味する.

3. 3行目は以下のとおりである：$\rho_\tau \wedge \varphi \rightarrow \varphi\prime$ は $\{\varphi\}\tau\{\varphi\}$ を意味する. ゆえに, 3行目は, $\forall \tau \in \mathcal{T}_H \cup \{\tau_E\}$ に対して, 状態遷移 τ の前に φ が成り立つならば状態遷移 τ の後にも φ が成り立つことを意味する.

4. 4行目は結論 $\Box p$ であり, $\Box$ は常に成り立つことを意味する時相オペレータである.

■

明らかに, この不変性の検証ルールの1行目と3行目は, φ がフェーズ遷移モジュールの不変式であることを示す. すなわち, φ はフェーズ遷移モジュールのすべての計算の上で成り立つ. また, 2行目より, φ ならば p なので, p もフェーズ遷移モジュールの不変式である.

次に, 不変性の検証ルールの健全性と完全性を証明する.

まず, 不変性の検証ルールの健全性とは, 不変性の性質が証明可能ならば, その性質は恒真であることを意味する. この健全性を示すためには, 以下を証明すればよいことが知られている [20, 21].

Theorem 1 (不変性の検証ルールの健全性)

receptiveness を満たすフェーズ遷移モジュール FTM 上において, 不変性の検証ルールの前提条件が状態恒真であるならば, 論理式 $\Box p$ は恒真である.

Proof 1 前提条件 *1.* が状態恒真であるので, 初期状態において φ は成り立つ. 前提条件 *3.* が状態恒真であるので, 初期状態から遷移規則を適用すると, すべての FTM の状態において φ は成り立つ. ここで, FTM は *receptiveness* を満たすので, 物理的に実装可能なすべての状態において φ は成り立つ. また, 前提条件 *2.* が状態恒真であるので, 物理的に実装可能なすべての状態において p は成り立つ. ゆえに, $\Box p$ は恒真である.

■

次に, 不変性の検証ルールの完全性とは, 不変性の性質が恒真ならば, その性質は証明可能であることを意味する. この完全性を示すためには, 以下を証明すればよいことが知られている [19, 20, 21]. (とりわけ, Manna らの文献 [19] の 2.5 に詳しく説明がある.)

Theorem 2 (不変性の検証ルールの完全性)

receptiveness を満たすフェーズ遷移モジュール FTM 上において, 論理式 $\Box p$ が恒真ならば, 不変性の検証ルールの前提条件が状態恒真であるような表明式 φ が存在する.

Proof 2 証明の基本的なアイディアは状態 s がアクセス可能な状態であるときのみに限り状態 s で成り立つ表明式 acc を構成することである. このことより, acc を φ と見なすときに, $\Box p$ が恒真ならば検証ルールの前提条件は状態恒真であることを示す. ここでは, 相対完全性のみを証明するので, すべての恒真な表明式を証明することを仮定しながら, 前提条件の状態恒真性を示せば十分である.

次のような表明式 acc を構成したと仮定する：

$s \models acc$ *iff* s はアクセス可能な状態である

なお, FTM は *receptive* なので, アクセス可能な状態は物理的に実装可能な状態である.

以下に, φ の代わりに acc が 3 つの前提条件を状態恒真にすることを示す：

1. $acc \rightarrow p$：

   $\Box p$ が恒真であると仮定することによって, 状態列に出現する任意の状態は p を満たす. FTM は *receptive* なので, すべてのアクセス可能な状態がある状態列に出現して, すべてのアクセス可能な状態が p を満たす. acc はアクセス可能な状態を特徴付けるので, この前提条件は成り立つ.

2. $\Theta \rightarrow acc$：

   明らかに, Θ を満足する, すべての状態は初期状態であり, アクセス可能な状態である. 結局, そのような状態は acc を満たさなければならない.

3. 任意の $\tau \in \mathcal{T}_H \cup \{\tau_E\}$ 及び U と $\widetilde{U}$ のすべての値に対して $\rho_\tau(U, \widetilde{U}) \wedge acc(U) \rightarrow acc(\widetilde{U})$：

U と $\widetilde{U}$ は $\rho_\tau(U,\widetilde{U})$ と $acc(U)$ の両方が真 *(true)* であるような値のリストとする．以下に，$acc(\widetilde{U})$ が真 *(true)* であることを示す：

s と $\widetilde{s}$ は $s[V]=U$ と $\widetilde{s}[V]=\widetilde{U}$ である，2つの任意の状態である．ただし，V はシステム変数の有限集合である．$acc(U)=true$ なので，$s \models acc$ である．acc によって，s はアクセス可能な状態である．$\rho_\tau(U,\widetilde{U})=true$ より，$<s,\widetilde{s}> \models \rho_\tau(V,V\prime)$ であり，$\widetilde{s}$ は s の $\tau-successor$ である．s はアクセス可能な状態なので，$\widetilde{s}$ はアクセス可能な状態であり，$\widetilde{s} \models acc$ となり，$acc(\widetilde{U})=true$ となる．

以上の *(1)*〜*(3)* より，表明式 acc が構成できることが示せた．

ゆえに，検証ルールの完全性が証明できた．

■

3.5 鉄道踏み切りによる検証事例

3.5.1 鉄道踏み切りの事例

本論文では，フェーズ遷移モジュールにより，鉄道踏み切りの事例を仕様記述して，モジュール演繹的検証を行う．図7に示すように，鉄道踏み切りは $Train$, $Gate$ と $Controller$ の3つのフェーズ遷移モジュールから構成される．

以下では，図7の詳細を説明する：変数 x は $Gate$ から $Train$ の距離を表現する．$\dot{x}$ は時間に関する一階微分であり，$Train$ の速度を表現する．最初，$u=far$ では，$Train$ は $Gate$ から離れており，48と52の間の速度で移動する．次に，$Train$ が $Gate$ に接近するとき，踏み切りから距離1000に置かれているセンサが $Train$ を検知して，$Controller$ に信号 app を送信する．それから，$Train$ は $u=near$ に

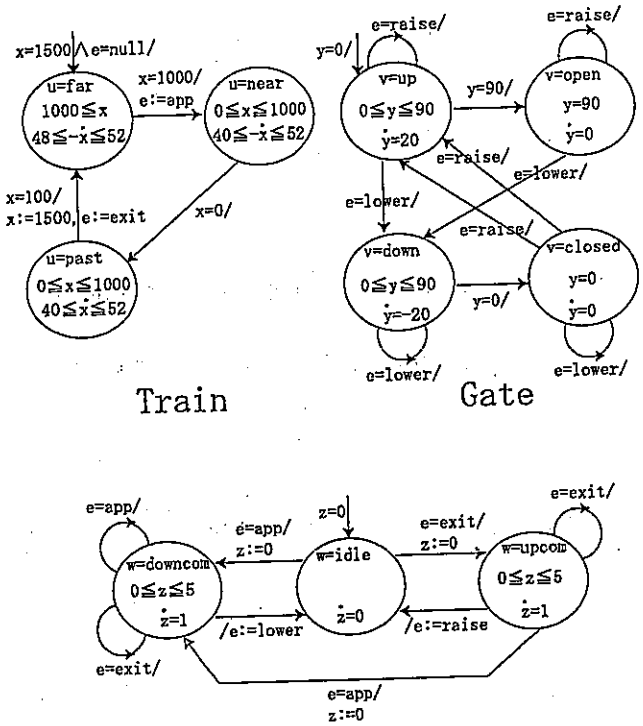


図 7: 鉄道踏み切りの事例

なり, 40 と 52 の間の速度で移動する. *Controller* が $w = idle$ で信号 *app* を受信するならば, 5 時刻以内に *Gate* に信号 *lower* を送信する. ここで, *Controller* の遅延は変数 z で計測する. もし *Gate* が開くならば, 速度 20 度により, 90 度から 0 度へ下がる. ここで, *Gate* の位置は変数 y で表現する. 踏み切りから 100 過ぎたところにある第 2 のセンサは *Controller* に信号 *exit* を送信して, 5 時刻以内に *Gate* に信号 *raise* を送信する. 我々は, *Train* 間の距離は 1500 と仮定する.

3.5.2 鉄道踏み切りの検証

モジュール検証を保証するために, まず, 各モジュールの receptiveness を検証する必要がある. 図 7 の鉄道踏み切りを構成する *Train*, *Gate* と *Controller* の 3 つのフェーズ遷移モジュールは, receptive であるかどうかを検証すると, すべて receptive であった. 以下の図 8 に, *Train* の receptiveness の証明図を示す. この証明図では, 遷移はすべてモジュールの動作であり, △ または controlled 遷移が選ばれる. △ が選ばれるならば, $t_0 + \epsilon$ 以上になる. controlled が選ばれるならば, ランキング関数 $\delta_i (i = 0, \ldots, 5)$ は減少する. ゆえに, receptiveness の証明図は正しいので, *Train* は receptive である.

次に, モジュール演繹的検証について説明する. 図 7 の踏み切りの事例では, 例えば, 以下のような検証性質がある :

1. $\Box\ (u = past \rightarrow w = idle \vee w = downcom)$
2. $\Box\ (u = far \rightarrow \Diamond (v = closed))$
3. $\Box \neg\ (u = past \wedge v = up \wedge w = downcom)$

上記 (3) の検証性質は *Train*, *Gate* と *Controller* の 3 つに直接関係するが, (1) は *Train* と *Controller*

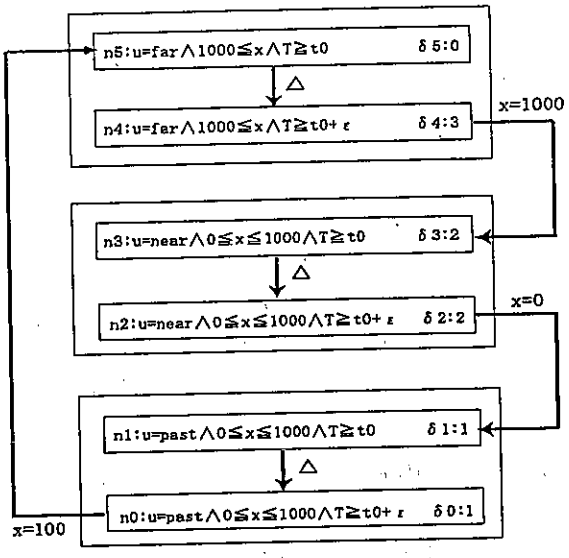


図 8: Train の receptiveness の証明図

にのみ直接関係して, (2) は $Train$ と $Gate$ にのみ直接関係する. ゆえに, (1) と (2) はモジュール演繹的検証が可能である.

以下に, $\Box\ (u = past \rightarrow w = idle \vee w = downcom)$ の検証を示す.

まず, $Train$ と $Controller$ の並列合成から構成されるフェーズ遷移モジュール $\mathbf{FTM} = (V, \Theta, \mathcal{T}, \mathcal{A}, \Pi)$ を定義する. ここで, 各要素は以下のとおりである :

1. $V = D \cup I$ であり, $D = \{u, w, e\}$ と $I = \{x, z, T\}$ である. また, $V = V^P \cup V^S$ であり, $V^P = \{u, x, w, z\}$ と $V^S = \{e, T\}$ である.

2. $\Theta : u = far \wedge w = idle \wedge x = 1500 \wedge z = 0 \wedge e = null \wedge T = 0.$

3. 各 $\tau_i \in \mathcal{T} (i = 1, \ldots, 33)$ と τ_E は以下のとおりである :

   (1) $\rho_{\tau_1} : u = far \wedge w = idle \wedge x \geq 1000 \wedge z = 0 \wedge e = null \wedge T \geq 0 \wedge u\prime = near \wedge w\prime =$

$idle \wedge x\prime = 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(2) $\rho_{\tau_2} : u = near \wedge w = idle \wedge x = 0 \wedge z = 0 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = idle \wedge x\prime = 0 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(3) $\rho_{\tau_3} : u = past \wedge w = idle \wedge x = 100 \wedge z = 0 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = idle \wedge x\prime = 1500 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0.$

(4) $\rho_{\tau_4} : u = far \wedge w = downcom \wedge x = 1000 \wedge 0 \leq z \leq 5 \wedge (e = app \vee exit) \wedge T > 0 \wedge u\prime = near \wedge w\prime = downcom \wedge x\prime = 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(5) $\rho_{\tau_5} : u = near \wedge w = downcom \wedge x = 0 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = downcom \wedge x\prime = 0 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(6) $\rho_{\tau_6} : u = past \wedge w = downcom \wedge x = 100 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = downcom \wedge x\prime = 1500 \wedge z\prime \geq 0 \wedge e\prime = exit \wedge T\prime > 0.$

(7) $\rho_{\tau_7} : u = far \wedge w = upcom \wedge x = 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime = upcom \wedge x\prime = 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(8) $\rho_{\tau_8} : u = near \wedge w = upcom \wedge x = 0 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime = upcom \wedge x\prime = 0 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0.$

(9) $\rho_{\tau_9} : u = past \wedge w = upcom \wedge x = 100 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime = upcom \wedge x\prime = 1500 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0.$

(10) $\rho_{\tau_{10}} : u = far \wedge w = idle \wedge x \geq 1000 \wedge z \geq 0 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime = upcom \wedge x\prime \geq 1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0.$

(11) $\rho_{\tau_{11}} : u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime = idle \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = raise \wedge T\prime > 0.$

(12) $\rho_{\tau_{12}} : u = far \wedge w = idle \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = downcom \wedge x\prime \geq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(13) $\rho_{\tau_{13}} : u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = idle \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0.$

(14) $\rho_{\tau_{14}} : u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = downcom \wedge x\prime \geq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(15) $\rho_{\tau_{15}} : u = near \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime = upcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0.$

(16) $\rho_{\tau_{16}} : u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge (e = exit \vee app) \wedge T > 0 \wedge u\prime = near \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = raise \wedge T\prime > 0.$

(17) $\rho_{\tau_{17}} : u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0.$

(18) $\rho_{\tau_{18}} : u = near \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(19) $\rho_{\tau_{19}} : u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(20) $\rho_{\tau_{20}} : u = past \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$upcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0.$

(21) $\rho_{T21} : u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0.$

(22) $\rho_{T22} : u = past \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(23) $\rho_{T23} : u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge (e = app \vee exit) \wedge T > 0 \wedge u\prime = past \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0.$

(24) $\rho_{T24} : u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0.$

(25) $\rho_{T25} : u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = downcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(26) $\rho_{T26} : u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime = downcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0.$

(27) $\rho_{T27} : u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(28) $\rho_{T28} : u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0.$

(29) $\rho_{T29} : u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0.$

(30) $\rho_{\tau_{30}}$: $u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime =$ $past \wedge w\prime = downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$.

(31) $\rho_{\tau_{31}}$: $u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime =$ $upcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$.

(32) $\rho_{\tau_{32}}$: $u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime =$ $near \wedge w\prime = upcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$.

(33) $\rho_{\tau_{33}}$: $u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime =$ $past \wedge w\prime = upcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$.

(34) ρ_{τ_E} : $u = u\prime \wedge w = w\prime \wedge x = x\prime \wedge z = z\prime \wedge T = T\prime$.

4. 各 $\alpha_i \in \mathcal{A}(i = 1, \ldots, 9)$ は以下のとおりである：

(1) $\rho_{\tau_{\alpha_1}}$: $\exists \triangle > 0.((u\prime = u = far \wedge w\prime = w = idle \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq$ $1500 - 48 \times \triangle) \wedge \forall t \in [0, \triangle).(u = far \rightarrow x \geq 1000))$.

(2) $\rho_{\tau_{\alpha_2}}$: $\exists \triangle > 0.((u\prime = u = near \wedge w\prime = w = idle \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq$ $1000 - 40 \times \triangle) \wedge \forall t \in [0, \triangle).(u = near \rightarrow 0 \leq x \leq 1000))$.

(3) $\rho_{\tau_{\alpha_3}}$: $\exists \triangle > 0.((u\prime = u = past \wedge w\prime = w = idle \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge \forall t \in$ $[0, \triangle).(u = past \rightarrow 0 \leq x \leq 1000))$.

(4) $\rho_{\tau_{\alpha_4}}$: $\exists \triangle > 0.((u\prime = u = far \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq 1500 -$ $48 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = far \rightarrow x \geq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$.

(5) $\rho_{\tau_{\alpha_5}} : \exists\triangle > 0.((u\prime = u = near \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq 1000 - 40 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = near \rightarrow 0 \leq x \leq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$.

(6) $\rho_{\tau_{\alpha_6}} : \exists\triangle > 0.((u\prime = u = past \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = past \rightarrow 0 \leq x \leq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$.

(7) $\rho_{\tau_{\alpha_7}} : \exists\triangle > 0.((u\prime = u = far \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq 1500 - 48 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = far \rightarrow x \geq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$.

(8) $\rho_{\tau_{\alpha_8}} : \exists\triangle > 0.((u\prime = u = near \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq 1000 - 40 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = near \rightarrow 0 \leq x \leq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$.

(9) $\rho_{\tau_{\alpha_9}} : \exists\triangle > 0.((u\prime = u = past \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = past \rightarrow 0 \leq x \leq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$.

5. $\Pi : (u = far \rightarrow x \geq 1000) \wedge (u = near \rightarrow 0 \leq x \leq 1000) \wedge (u = past \rightarrow 0 \leq x \leq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)$.

なぜならば, 状態に割り当てられた変数の制約条件は上記だからである.

次に, 以下の公理系により, 不変性 $\Box\, p = \Box\, (u = past \rightarrow w = idle \vee w = downcom)$ を検証する.

表明 φ と p に対して, 公理系は以下である.

1. $\Theta \rightarrow \varphi$

2. $\varphi \rightarrow p$

3. $\rho_\tau \wedge \varphi \rightarrow \varphi\prime \quad (\forall \tau \in \mathcal{T}_H \cup \{\tau_E\})$

---

4. $\Box p$

以下では, $\varphi = p$ とする.

1. Θ=$(u = far \wedge w = idle \wedge x = 1500 \wedge z = 0 \wedge e = null \wedge T = 0)$ 及び $\varphi = p$=$(u = past \rightarrow w = idle \vee w = downcom)$ なので, $\Theta \rightarrow \varphi$ は成り立つ.

2. $\varphi = p$ なので, $\varphi \rightarrow p$ は成り立つ.

3. $\forall \tau \in \mathcal{T}_H \cup \{\tau_E\}$ に対して, 以下のように, $\rho_\tau \wedge \varphi \rightarrow \varphi\prime$ は成り立つ.

   (1) $\rho_{\tau_1} \wedge \varphi \rightarrow \varphi\prime$:

   $u = far \wedge w = idle \wedge x \geq 1000 \wedge z = 0 \wedge e = null \wedge T \geq 0 \wedge u\prime = near \wedge w\prime = idle \wedge x\prime =$

   $1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

   $\wedge$

   $(u = past \rightarrow w = idle \vee w = downcom)$

   $\rightarrow$

   $(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

   上記論理式は成り立つ.

   (2) $\rho_{\tau_2} \wedge \varphi \rightarrow \varphi\prime$:

   $u = near \wedge w = idle \wedge x = 0 \wedge z = 0 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime = idle \wedge x\prime =$

$0 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(3) $\rho_{\tau_3} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = idle \wedge x = 100 \wedge z = 0 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime = idle \wedge x\prime =$

$1500 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(4) $\rho_{\tau_4} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = downcom \wedge x = 1000 \wedge 0 \leq z \leq 5 \wedge (e = app \vee exit) \wedge T > 0 \wedge u\prime =$

$near \wedge w\prime = downcom \wedge x\prime = 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(5) $\rho_{\tau_5} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = downcom \wedge x = 0 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$downcom \wedge x\prime = 0 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$,

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(6) $\rho_{\tau_6} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = downcom \wedge x = 100 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$downcom \wedge x\prime = 1500 \wedge z\prime \geq 0 \wedge e\prime = exit \wedge T\prime > 0$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(7) $\rho_{\tau_7} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = upcom \wedge x = 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$upcom \wedge x\prime = 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(8) $\rho_{\tau_8} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = upcom \wedge x = 0 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$upcom \wedge x\prime = 0 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(9) $\rho_{\tau_9} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = upcom \wedge x = 100 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$upcom \wedge x\prime = 1500 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(10) $\rho_{\tau_{10}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = idle \wedge x \geq 1000 \wedge z \geq 0 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime = upcom \wedge x\prime \geq$

$1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(11) $\rho_{\tau_{11}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$idle \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = raise \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(12) $\rho_{\tau_{12}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = idle \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$downcom \wedge x\prime \geq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(13) $\rho_{\tau_{13}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$idle \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(14) $\rho_{\tau_{14}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$downcom \wedge x\prime \geq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(15) $\rho_{\tau_{15}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$upcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(16) $\rho_{\tau_{16}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge (e = exit \vee app) \wedge T > 0 \wedge u\prime =$

$near \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = raise \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(17) $\rho_{\tau_{17}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(18) $\rho_{\tau_{18}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(19) $\rho_{T19} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(20) $\rho_{T20} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$upcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(21) $\rho_{T21} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(22) $\rho_{T22} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = idle \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(23) $\rho_{T23} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge (e = app \vee exit) \wedge T > 0 \wedge u\prime =$

$past \wedge w\prime = idle \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = lower \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(24) $\rho_{T24} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge z\prime = 0 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(25) $\rho_{T25} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$downcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(26) $\rho_{T26} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = downcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$downcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(27) $\rho_{T27} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(28) $\rho_{\tau_{28}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(29) $\rho_{\tau_{29}} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = app \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = app \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(30) $\rho_{\tau_{30}} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = downcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$downcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(31) $\rho_{\tau_{31}} \wedge \varphi \rightarrow \varphi\prime$:

$u = far \wedge w = upcom \wedge x \geq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = far \wedge w\prime =$

$upcom \wedge x\prime \geq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(32) $\rho_{\tau_{32}} \wedge \varphi \rightarrow \varphi\prime$:

$u = near \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = near \wedge w\prime =$

$upcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(33) $\rho_{\tau_{33}} \wedge \varphi \rightarrow \varphi\prime$:

$u = past \wedge w = upcom \wedge 0 \leq x \leq 1000 \wedge 0 \leq z \leq 5 \wedge e = exit \wedge T > 0 \wedge u\prime = past \wedge w\prime =$

$upcom \wedge 0 \leq x\prime \leq 1000 \wedge 0 \leq z\prime \leq 5 \wedge e\prime = exit \wedge T\prime > 0$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(34) $\rho_{\tau_E} \wedge \varphi \rightarrow \varphi\prime$:

$u = u\prime \wedge w = w\prime \wedge x = x\prime \wedge z = z\prime \wedge T = T\prime$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(35) $\rho_{\tau_{a_1}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = far \wedge w\prime = w = idle \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq 1500 - 48 \times \triangle) \wedge \forall t \in$

$[0, \triangle).(u = far \rightarrow x \geq 1000))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(36) $\rho_{\tau_{\alpha_2}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = near \wedge w\prime = w = idle \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq 1000 - 40 \times$

$\triangle) \wedge \forall t \in [0, \triangle).(u = near \rightarrow 0 \leq x \leq 1000))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(37) $\rho_{\tau_{\alpha_3}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = past \wedge w\prime = w = idle \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge \forall t \in$

$[0, \triangle).(u = past \rightarrow 0 \leq x \leq 1000))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(38) $\rho_{\tau_{\alpha_4}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = far \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq 1500 -$

$48 \times \triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = far \rightarrow x \geq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(39) $\rho_{\tau_{\alpha_5}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = near \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq 1000 - 40 \times$

$\triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = near \rightarrow 0 \leq x \leq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(40) $\rho_{\tau_{\alpha_6}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = past \wedge w\prime = w = downcom \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge (z =$

$\triangle) \wedge \forall t \in [0, \triangle).((u = past \rightarrow 0 \leq x \leq 1000) \wedge (w = downcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(41) $\rho_{\tau_{\alpha_7}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = far \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (1500 - 52 \times \triangle \leq x \leq 1500 - 48 \times$

$\triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = far \rightarrow x \geq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(42) $\rho_{\tau_{\alpha_8}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = near \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (1000 - 52 \times \triangle \leq x \leq 1000 - 40 \times$

$\triangle) \wedge (z = \triangle) \wedge \forall t \in [0, \triangle).((u = near \rightarrow 0 \leq x \leq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

(43) $\rho_{\tau_{\alpha_9}} \wedge \varphi \rightarrow \varphi\prime$:

$\exists \triangle > 0.((u\prime = u = past \wedge w\prime = w = upcom \wedge e\prime = e) \wedge (40 \times \triangle \leq x \leq 52 \times \triangle) \wedge (z =$

$\triangle) \wedge \forall t \in [0, \triangle).((u = past \rightarrow 0 \leq x \leq 1000) \wedge (w = upcom \rightarrow 0 \leq z \leq 5)))$

$\wedge$

$(u = past \rightarrow w = idle \vee w = downcom)$

$\rightarrow$

$(u\prime = past \rightarrow w\prime = idle \vee w\prime = downcom)$

上記論理式は成り立つ.

以上により, 不変性 $\Box\, p = \Box\, (u = past \rightarrow w = idle \vee w = downcom)$ が成り立つことが検証できた.

このことは, *Train*, *Gate* と *Controller* の 3 つのフェーズ遷移モジュールの並列合成が必要であった従来の検証問題が, 本論文の提案手法により, *Train* と *Controller* の 2 つのフェーズ遷移モジュールの並列合成の検証問題に還元できたことを意味する.

本章で提案したモジュール演繹的検証手法の効果に関しては, 一般的には, 検証で証明すべき式の数が指数オーダーで削減できると考えられる. これは, モジュール数に対して, 指数的に状態数と遷移数が増加するためである. この理論的な裏付けは, この検証問題が PSPACE 完全であること [22] に起因する. しかし, 現実的には, モジュール演繹的検証する場合に, 環境の影響や receptiveness の検証を考慮する必要がある.

1. 環境の影響に関しては, 環境の状態遷移 (ρ_{τ_E}) と環境の経過時間 (ρ_{τ_α}) を考慮する必要がある. 環境の経過時間 (ρ_{τ_α}) はモジュールの経過時間 (ρ_{τ_α}) と等しいので, 証明すべき式を増加させない. なぜならば, 環境が時間経過するときはモジュールも時間経過して, その時間経過の長さは無限ゲームに従って決まるからである. 環境の状態遷移 (ρ_{τ_E}) では, 証明すべき式が一つ増える.

以上より, 環境の影響による証明すべき式の数は一つ増加する.

2. receptiveness の検証に関しては, モジュールの数の証明が必要である. 個々のモジュールの receptiveness の検証では, モジュールの状態数と状態遷移数との和で証明すべき式が必要である. 以上より, receptiveness の検証による証明すべき式の数は（モジュールの数）×（モジュールの状態数と状態遷移数の和）に比例して増加すると考えられる.

以上より, 従来手法と比較して提案手法では, 証明すべき式の数が指数オーダーで削減できるが, 一方, モジュールの数やモジュールの状態数と状態遷移数の多項式オーダーで増加すると考えられる. つまり, 提案手法はモジュール数が多いシステムの検証で大きな効果を発揮する. また, 提案手法では, 証明すべき式そのものの長さが短くなる.

具体的には, 提案手法で証明すべき式の数は, 安全性証明の 43 個の式と receptiveness の検証の 31 個の式で, 合計 74 個の式である. 従来手法で証明すべき式の数は, 安全性証明の 258 個の式である. つまり, 提案手法により, 証明すべき式の数が 30 % になった. ゆえに, 提案手法は大いに有益であるといえる.

今後はさらに, 実用レベルの事例により, モジュール単位の仕様記述と演繹的検証手法の効果を実験的に裏付ける必要がある.

3.6 むすび

本章では, 大規模ハイブリッドシステムの検証を実現するために, フェーズ遷移モジュールを開発して, その receptiveness の検証を行うことにより, モジュール演繹的検証を実現した. 演繹的証明は

数学的訓練が必要な作業であり, 大規模システムへの適用は困難である. しかし, 本論文のように, モジュール演繹的検証を適用すれば, 大規模システムの検証問題が小規模システムへの検証問題に還元できて, 検証可能なシステムの規模が大きくなる.

本章のアプローチとは異なるものとして, ハイブリッドシステムの仕様が有限状態なハイブリッドオートマトンで構成できた場合のモデル検査手法 [4] がある. しかし, モデル検査アルゴリズムの停止性が保証されているクラスは Rectangular なハイブリッドオートマトン [23] と O-minimal なハイブリッドオートマトン [24] のみであり, さらに, 仕様の表現能力が弱い.

ハイブリッドシステムの検証においては, 演繹的検証の重要性は大きいと考えられる. 以上より, 演繹的検証を実用化するために, 今後の研究課題として, 以下が重要である.

1. 表明の自動生成や証明式の決定可能手続きの発見, GUI の向上などにより, 演繹的証明作業の効率的な計算機支援を実現する.
2. 抽象実行などの抽象化技術を適用することにより, フェーズ遷移モジュールの自動検証を実現する.

4　ハイブリッドモデルによるスケジューラの演繹的検証

4.1　まえがき

近年, マイクロプロセッサの 99 %以上は組込み型システムに使われており, 組込み型システムのほとんどはリアルタイムソフトウェアが動作している. リアルタイムソフトウェアは非常に重要な計算機ソフトウェアパラダイムである. リアルタイムソフトウェアはリアルタイムオペレーティングシステム上で動作して, 品質保証が難しいソフトウェアである. なぜならば, リアルタイムソフトウェアはタイミング制約やリソース割当てなどの条件が厳しくて, スケジューラビリティ, 安全性と活性などを保証しなければならないからである [25].

本章では, リアルタイムソフトウェアの設計には, 以下の難しさがあると考える.

1. 一般的に, リアルタイムソフトウェアはプリエンプティブスケジューリングで動作するので, 仕様記述が容易でない. 仕様記述の手法としては, 扱いが困難なハイブリッドオートマトン [26] を用いざるをえない. なぜならば, あるタスクは優先度の高いタスクに割り込まれて, その優先度の高いタスクの実行が終了して, 実行が継続されるからである. すなわち, あるタスクの実行時間は, 実行中では微分係数 1 で増加して, 停止中では微分係数 0 で増加するとして仕様記述する必要があるためである. なお, 常に微分係数が 1 の場合は時間オートマトンである.
2. 一般的に, リアルタイムソフトウェアはリアルタイムオペレーティングシステム上で動作するので, スケジューリング可能であることが必要である [27] . スケジューリング可能とは, タスクの最悪応答時間がタスクのデッドライン以下であることを意味する.
3. 最近では, リアルタイムオペレーティングシステムの改造が行われており, 仕様に対するリアル

タイムオペレーティングシステムの正当性を保証する必要がある場合が多く存在する.

本章では, 上記の設計の困難さを解決するために, 上記のそれぞれに対して, 以下の手法を提案する.

1. まず, リアルタイムソフトウェアの仕様記述言語としてハイブリッドオートマトン [26] を拡張する.
2. 次に, 仕様に対するリアルタイムオペレーティングシステムの正当性を検証するために, 仕様とリアルタイムオペレーティングシステムとの並列合成が元の仕様を正しく詳細化していることを検証する. これを実現するために, ハイブリッドオートマトンの演繹的詳細化検証手法を開発する. そして, 既存の手法である, M. Joseph らの最悪応答時間計算法 [27] を使って, 最悪応答時間を計算して, 最悪応答時間がデッドライン以下であることをチェックする. 以降では, これをスケジューラビリティ検証と呼ぶ.

以上の本手法により, 仕様記述と検証がハイブリッドオートマトン上で統一的に実現できる. なお, 単一 CPU で, 周期タスクが固定優先度のプリエンプティブスケジューリングで動作するリアルタイムソフトウェアを想定して, ハイブリッドオートマトンにより, 仕様記述して, 仕様に対するリアルタイムオペレーティングシステムの正当性を検証する事例を示す.

近年, 形式的手法により, リアルタイムソフトウェアを仕様記述して, その正当性を検証する研究は多くなされている. 本章では, 実用上重要な固定優先度のプリエンプティブスケジューリングに着目する. 代表的な研究では, 計算モデル及び時間制約記述の観点から, 以下のように分類できる.

1. 計算モデルとしては, オートマトンまたはプロセス代数に分類できる.

2. 時間制約記述としては, タイミング制約記述（例えば, 時間オートマトン）または蓄積時間記述（例えば, ハイブリッドオートマトン）に分類できる.

具体的には, 以下のような重要な既存研究が存在する.

1. 1996年, ペンシルベニア大学の Alur らは, 初めて, ハイブリッドオートマトンでプリエンプティブスケジューリングを仕様記述した [26]. また, モデル検査手法により, 安全性と活性を検証した. しかし, 仕様に対するリアルタイムオペレーティングシステムの正当性を検証するためには, 満たすべきすべての性質を漏れなく, 時相論理式で記述する必要がある.
2. 2000年, ハネウエル社の Vestal は, Alur らのハイブリッドオートマトンに制約を加えて, 到達可能解析が決定可能なハイブリッドオートマトンのクラスを提案して, 実用的なリアルタイムソフトウェアとプリエンプティブスケジューリングの仕様記述と検証を行った [28]. しかし, オペレーティングシステムを仕様記述しないで, タスクのみを仕様記述して検証しているために, オペレーティングシステムの正当性を検証していない.
3. 1999年, ブエノスアイレス大学の Braberman らは, 時間オートマトンでプリエンプティブスケジューリングを仕様記述して, 初めて, その正当性を自動検証した [29]. 彼らは, 最大と最小のタイミング制約により, プリエンプティブスケジューリングにおける, タスクの実行時間を大き目に近似して仕様記述と検証を行った.
4. 2002年, VERIMAG の Sifakis らは, 優先度を付けて拡張した時間オートマトンを提案して, リアルタイムソフトウェアの仕様記述を行った [30]. Braberman と同様に, タスクの実行時間を近似的に扱っている.

5. 1999年, ライセスター大学のLiuらは, TLA(Temporal Logic in Action) により, リアルタイムソフトウェアとプリエンプティブスケジューリングの仕様記述と演繹的検証を行った [31]. 彼らは, タスク実行可能状態, タスク実行中状態, タスク実行停止状態とタスク実行再開状態のタイミング制約記述により, プリエンプティブスケジューリングとそのリアルタイムソフトウェアを仕様記述した. しかし, タスクの実行時間をタイミング制約記述で仕様記述したために, 大き目の値と小さ目の値でタスクの実行時間を近似して仕様記述と検証を行った. つまり, タイミング制約記述では, プリエンプティブスケジューリングは正確には仕様記述できない.

プリエンプティブスケジューリングは, 優先度の高いタスクが優先度の低いタスクの実行権を横取りするので, 蓄積時間に関する仕様記述が必要となり, 時間オートマトンなどのタイミング制約記述では近似的かつ複雑となる. 一方, ハイブリッドオートマトンで仕様記述すると, 蓄積時間に関する仕様記述は厳密かつ簡単にできる. しかし, ハイブリッドオートマトンの検証コストは大きいので, 大規模なシステムの検証ができない. 本論文では, ハイブリッドオートマトンの詳細化検証理論を開発することにより, ハイブリッドオートマトンの検証を可能とする. これにより, 仕様記述の複雑さを回避して, 仕様に対するスケジューラの正当性が検証できる. Liuらの手法 [31] では, タスクの実行時間をタイミング制約記述で仕様記述するために, 大き目の値と小さ目の値でタスクの実行時間を挟んで仕様記述を行った. このために, 数行にわたる論理式により, タスクの実行時間を仕様記述している. 提案手法では, 微分係数を0と1で切り替えることにより, 一行未満で記述できる.

また, ソフトウェア開発における本手法の位置づけとしては, 仕様記述レベルで, 仕様がリアルタイムオペレーティングシステム上で実装可能であることを検証するという大きな意義がある. ゆえに, 本章の手法により, 開発の手戻りが防げる.

以降の本章の構成は以下のとおりである．4－2節では，リアルタイムソフトウェアの仕様記述を述べる．4－3節では，リアルタイムソフトウェアの検証手法を提案する．4－4節では，検証事例を示す．最後に，4－5節では，まとめと今後の課題を述べる．

4.2 リアルタイムソフトウェアの仕様記述

本節では，ハイブリッドオートマトンを導入して，リアルタイムソフトウェアの仕様記述を示す．

4.2.1 ハイブリッドオートマトン

まず，Alurらの線形ハイブリッドオートマトン [26] を拡張したハイブリッドオートマトンを定義する．Alurらの線形ハイブリッドオートマトンでは，観測可能な変数と観測不能な内部変数を区別していないので，詳細化検証には適さない．そこで，本論文では，観測可能な変数と観測不能な内部変数を区別して，Alurらの線形ハイブリッドオートマトンを拡張する．

Definition 13 (ハイブリッドオートマトン)

ハイブリッドオートマトンは，$\mathbf{A} = (\vec{x}, V, local, inv, dif, E, act, Label, syn, E_\Theta)$ の *10* 組で定義される．ここで，

1. データ変数

有限なベクタ $\vec{x} = (x_1, x_2, \ldots, x_n)$ は実数値のデータ変数である．$\vec{x}$ のサイズ n は A の次元と呼ばれる．

データの状態は n-次元の実数空間 R^n の点 $\vec{s} = (s_1, \ldots, s_n)$ であり，各データ変数 x_i に実数値

$s_i \in R$ を割り付ける関数により定義される．凸状のデータの領域は R^n の凸状の多面体である．データの領域は凸状のデータの領域の有限な和集合である．凸状のデータの述語は $\vec{x}$ 上の凸状の線形な論理式である．凸状のデータの述語 p は凸状のデータの領域 $\ll p \gg \subseteq R^n$ を定義する．ここで，$p[\vec{x} := \vec{s}]$ が $true$ のときに限り $\vec{s} \in \ll p \gg$ である．

任意のデータ変数 x_i に対して，x_i の第一階導関数を $\dot{x}_i$ と表現する．第一階導関数の包含は R^n の中の凸状の多面体である．速度の述語は $\dot{\vec{x}}$ 上の凸状の線形な論理式である．速度の述語 r は第一階導関数の包含 $\ll r \gg \subseteq R^n$ を定義する．ここで，$r[\dot{\vec{x}} := \vec{s}]$ が $true$ のときに限り $\vec{s} \in \ll r \gg$ である．

任意のデータ変数 x_i に対して，状態遷移後の x_i の新しい値を $x_i\prime$ と表記する．アクションの述語 $q = (\vec{y}, q\prime)$ は，更新された変数の集合 $\vec{y} \subseteq \vec{x}$ とその更新された変数のデータの述語である凸状の線形な論理式 $q\prime$ から構成される．アクションの述語 q を表す表明式 $\{q\}$ は凸状の線形な論理式 $q\prime \wedge \bigwedge_{x \in \vec{x} \setminus \vec{y}} (x\prime = x)$ である．すなわち，更新されないデータ変数は変化しないままである．アクションの述語 q は状態変数から凸状のデータの領域への関数 $\ll q \gg$ である：すべてのデータの状態 $\vec{s}, \vec{s}\prime \in R^n$ に対して，$\{q\}[\vec{x}, \vec{x}\prime := \vec{s}, \vec{s}\prime]$ が $true$ のときに限り $\vec{s} \in \ll q \gg (\vec{s})$ である．もしデータの領域 $\ll q \gg (\vec{s})$ が非空ならば，アクションの述語 q はデータの状態の中で実行可能である．

2. 制御ロケーション

ノードの有限集合 V は制御ロケーションである．

ハイブリッドオートマトン $\mathbf{A}$ の状態 $(v, \vec{s})$ は制御ロケーション $v \in V$ とデータの状態 $\vec{s} \in R^n$ から構成される．領域 $\mathbf{R} = \bigcup_{v \in V} (v, S_v)$ は各制御ロケーションのデータの領域 $S_v \subseteq R^n$ を集めたも

のである．A の状態の述語 $\phi = \bigcup_{v \in V}(v, p_v)$ は各制御ロケーションのデータの述語 p_v を集めたものである．A の状態の述語 ϕ は領域 $\ll \phi \gg = \bigcup_{v \in V}(v, \ll p_v \gg)$ を定義する．R を定義する状態の述語が存在するならば，領域 R は線形である．

領域 $(v, S) \cup \bigcup_{v' \neq v}(v', \emptyset)$ を (v, S) と書き，$(v, p) \cup \bigcup_{v' \neq v}(v', false)$ を (v, p) と書く．状態の述語を書くとき，制御ロケーションの集合 V 上の範囲をとるロケーションカウンタ l を使う．ロケーション制約 $l = v$ は状態の述語 $(v, true)$ を示す．状態の述語としてデータの述語を使うとき，データの述語 p は $\bigcup_{v \in V}(v, \mathrm{p})$ を示す．2 つの状態の述語 $\phi = \bigcup_{v \in V}(v, \mathrm{p_v})$ と $\phi' = \bigcup_{v \in V}(v, \mathrm{p_v}')$ に対して，$\neg\phi = \bigcup_{v \in V}(v, \neg \mathrm{p_v})$, $(\phi \vee \phi') = \bigcup_{v \in V}(v, \mathrm{p_v} \vee \mathrm{p_v}')$, $(\phi \wedge \phi') = \bigcup_{v \in V}(v, \mathrm{p_v} \wedge \mathrm{p_v}')$ である．

3. ローカル変数

local はローカル変数の有限集合であり，制御ロケーション V と有限なベクタ $\vec{x}$ の中で観測できない要素から構成される．ローカル変数は他のオートマトンから観測できない変数である．

4. ロケーション不変性

各制御ロケーション $v \in V$ に凸状のデータの述語 $inv(v)$，つまり v の不変式を割り付ける関数である．オートマトン A の制御は不変式 $inv(v)$ が *true* である限り v に留まるので，不変式 $inv(v)$ は一つのロケーションから別のロケーションへのシステムの前進を強制するために使われる．$\vec{s} \in \ll inv(v) \gg$ ならば状態 $(v, \vec{s})$ は存在しえる．A の存在しえる状態を Σ_A として，その状態の述語を $\bigcup_{v \in V}(v, inv(v))$ とする．

5. 連続的アクティビティ

各制御ロケーション $v \in V$ に速度の述語 $dif(v)$，つまり v のアクティビティを割り付ける関数

は dif である．アクティビティはデータ変数の値が変化する速度を制約する．つまり，オートマトンの制御がロケーション v に存在するとき，すべてのデータ変数の第一導関数は導関数の包含 $\ll dif(v) \gg$ 内に存在する．

6. 遷移

エッジの有限多重集合 E は遷移と呼ばれる．各遷移 $(v, v\prime)$ は遷移元ロケーション $v \in V$ と遷移先ロケーション $v\prime \in V$ を示す．各ロケーション $v \in V$ に対して，遷移 $e_v = (v, v)$ が存在する．

7. 離散的アクション

各遷移 $e \in E$ にアクションの述語 $act(e)$ を割り付けるラベリング関数 act は e のアクションである．アクション $act(e)$ が実行可能のときにのみ，オートマトンの制御は $e = (v, v\prime)$ により，ロケーション v からロケーション $v\prime$ へ進む．もし $act(e)$ がデータの状態 $\vec{s}$ において実行可能ならば，すべてのデータ変数の値は非決定的に $\vec{s}$ からデータの領域 $\ll act(e) \gg (\vec{s})$ のある点に変化する．

8. 同期ラベルとラベリング関数

同期ラベルの有限集合は $Label$ であり，各遷移 $e \in E$ に同期ラベルの部分集合を割り付けるラベリング関数は syn である．集合 $Label$ は A のアルファベットと呼ばれる．同期ラベルは 2 つのオートマトンの並列合成を定義するために使われる．つまり，2 つのオートマトンが同期ラベル a を共有するならば，一方のオートマトンの a-遷移が他方のオートマトンの a-遷移によって遂行されなければならない．

9. エントリーエッジ

エントリーエッジ E_Θ は遷移元のロケーションは存在しないが，エントリーロケーション $v_i \in V$ が

ある. E_Θ は $x_1 = c_1 \wedge x_2 = c_2 \wedge \ldots \wedge x_n = c_n$ によりラベル付けされている. ただし, $c_1, c_2, \ldots, c_n$ は実数値である.

■

次に, リアルタイムソフトウェアは多数のタスクとスケジューラの並列合成なので, ハイブリッドオートマトンの並列合成を定義する.

Definition 14 (ハイブリッドオートマトンの並列合成)

2つのハイブリッドオートマトン $A_1 = (\vec{x}_1, V_1, local_1, inv_1, dif_1, E_1, act_1, Label_1, syn_1, E_{\Theta 1})$ と $A_2 = (\vec{x}_2, V_2, local_2, inv_2, dif_2, E_2, act_2, Label_2, syn_2, E_{\Theta 2})$ が与えられたとする. ここで, A_1 と A_2 の次元は各々 n_1 と n_2 である. A_1 と A_2 の並列合成は, $A = (\vec{x}_1 \cup \vec{x}_2, V_1 \times V_2, local_1 \cup local_2, inv, dif, E, act, Label_1 \cup Label_2, syn, E_\Theta)$ である.

1. $V_1 \times V_2$ の各ロケーション $(v, v\prime)$ は不変式 $inv(v, v\prime) = inv_1(v) \wedge inv_2(v\prime)$ とアクティビティ $dif(v, v\prime) = dif_1(v) \wedge dif_2(v\prime)$ を持つ.
2. $E_\Theta = E_{\Theta 1} \times E_{\Theta 2}$ である. すなわち, エントリーロケーション (v_{1i}, v_{2i}) である. ここで, $v_{1i} \in V_1$ は A_1 のエントリーロケーションであり, $v_{2i} \in V_2$ は A_2 のエントリーロケーションである. また, E_Θ は $(x_{11} = c_{11} \wedge \ldots \wedge x_{1n_1} = c_{1n_1}) \wedge (x_{21} = c_{21} \wedge \ldots \wedge x_{2n_2} = c_{2n_2})$ によりラベル付けされている. ただし, $\vec{x}_1 = (x_{11}, x_{12}, \ldots, x_{1n_1})$, $\vec{x}_2 = (x_{21}, x_{22}, \ldots, x_{2n_2})$, $c_{11}, \ldots, c_{1n_1}$, $c_{21}, \ldots, c_{2n_2}$ は実数値である.
3. 以下のいずれかのときに限り, E は $e = ((v_1, v_2), (v_1\prime, v_2\prime))$ を含む:

(a) $v_1 = v_1\prime$, かつ, $Label_1 \cap syn_2(e_2) = \emptyset$ を有する $e_2 = (v_2, v_2\prime) \in E_2$ が存在する.

このとき, $act(e) = act_2(e_2)$ かつ $syn(e) = syn_2(e_2)$ である.

(b) $Label_1 \cap syn_2(e_2) = \emptyset$ を持つ $e_2 = (v_2, v_2\prime) \in E_2$ が存在して, かつ, $v_2 = v_2\prime$ である.

このとき, $act(e) = act_1(e_1)$ かつ $syn(e) = syn_1(e_1)$ である.

(c) $syn_1(e_1) \cap Label_2 = syn_2(e_2) \cap Label_1$ を持つ $e_1 = (v_1, v_1\prime) \in E_1$ と $e_2 = (v_2, v_2\prime) \in E_2$ が存在する.

このとき, $act_1(e_1) = (\vec{y_1}, q_1\prime)$ かつ $act_2(e_2) = (\vec{y_2}, q_2\prime)$ ならば, $act(e) = (\vec{y_1} \cup \vec{y_2}, q_1\prime \wedge q_2\prime)$ かつ $syn(e) = syn_1(e_1) \cup syn_2(e_2)$ である.

■

4.2.2 仕様記述

本論文で対象とするリアルタイムソフトウェアは複数の周期タスクと固定優先度のプリエンプティブスケジューリングで構成されるとする. 以下のように, ハイブリッドオートマトンにより, リアルタイムソフトウェアを仕様記述する.

1. 個々の周期タスクとスケジューラを独立にハイブリッドオートマトンで仕様記述して, 並列合成により, リアルタイムソフトウェアを構成する.
2. 個々の周期タスクでは, 実行時間と経過時間を表す変数をそれぞれ一つづつ使う. 実行時間の変数では, そのタスクの実行中は微係数が 1 で, そのタスクの停止中は微係数が 0 とする. これにより, プリエンプションを表現する. これがハイブリッドオートマトンを使う理由である. また, タスクの初期状態から初期状態に戻るまでを 1 周期とする. そして, 周期定数やデッドライン定

数, 実行時間定数を与えて, それら定数と実行時間を表す変数との大小関係によりタイミング制約を表現して, 状態遷移を制御する.

3. 優先度を状態遷移関係で表現するので, 固定優先度のスケジューリングポリシーのみが仕様記述できる. つまり, 仕様記述できるスケジューリングポリシーはレートモノトニックとデッドラインモノトニックである.

以下では, 仕様記述例を示す.

[周期タスクの仕様記述例]

まず, ハイブリッドオートマトンによる, 周期タスクの仕様記述例を示す. ここでは, 周期タスクの周期, 実行時間とデッドラインが決まっているとする. 2つの周期タスクの仕様記述例を図9に示す.

1. タスクの時間に関する定数は以下である：$T1$ と $T2$ は周期, $E1$ と $E2$ は実行時間, $D1$ と $D2$ はデッドラインである. なお, $E1 \leq D1 \leq T1$ と $E2 \leq D2 \leq T2$ である.
2. タスクの各ロケーションは以下である：sleep1 と sleep2 はタスクの実行準備中のロケーション, wait1 と wait2 はタスクの実行待ちのロケーション, execute1 と execute2 はタスクの実行中のロケーション, ready1 はプリエンプトの準備ロケーション, preempt2 はプリエンプティブロケーションである.
3. タスクの各イベントは以下である：arrive1 と arrive2 はタスクの実行要求の信号が到着したことを意味するイベントである. begin1 と begin2 はタスクの実行開始を意味するイベントである. end1 と end2 はタスクの実行終了を意味するイベントである. suspend と resume はオペレーティングシステムが発行するイベントであり, suspend はタスク2の実行を停止するイベントであり,

resume はタスク 2 の実行を再開するイベントである.

4. タスクの各クロックは以下である：$x1$ と $x2$ は各々タスク 1 と 2 の実行時間を計測する. $t1$ と $t2$ は各々タスク 1 と 2 の実行要求の信号が到着してからの経過時間を計測する. $\dot{x}1$ は $x1$ の時間に対する微分であり, $\dot{x}1 = 0$ はタスク 1 が停止していることを意味して, $\dot{x}1 = 1$ はタスク 1 が実行していることを意味している.

なお, タスク 1 がタスク 2 より優先度が高いとする. タスク 2 はタスク 1 に CPU を横取りされるので, execute2 から preempt2 への状態遷移が存在する. ここでは, レートモノトニックスケジューリングと考えて, $T1 < T2$ とする.

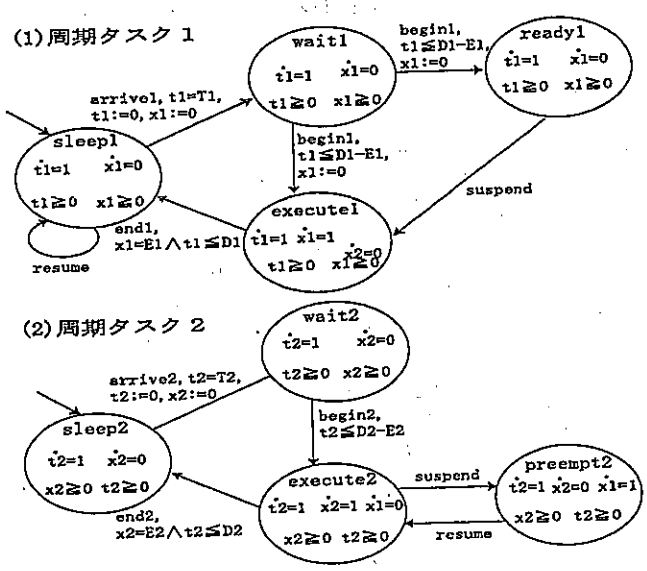


図 9: 周期タスクの仕様記述例

[プリエンプティブスケジューラの仕様記述例]

次に, 上記の２つの周期タスクを制御する, プリエンプティブスケジューラを仕様記述する. 図１１に, プリエンプティブスケジューラの仕様を図示する. ここでは, 固定優先度のプリエンプティブスケジューリングの一種であるレートモノトニックスケジューリングと考える [33].

プリエンプティブスケジューラの各ロケーションは以下である : idle はどのタスクも実行していないロケーション, task1 はタスク１が実行しているロケーション, task2 はタスク２が実行しているロケーション, preempt はタスク２が CPU を横取りされているロケーション, preemptexe はタスク１が実行しているロケーションである.

プリエンプティブスケジューラの各クロックは図１０の周期タスクと全く同じである. ただし, task1 ロケーションでは, タスク２は実行しないので, $x2 = 0$ である. また, スケジューラの各イベントは図１０と同様である.

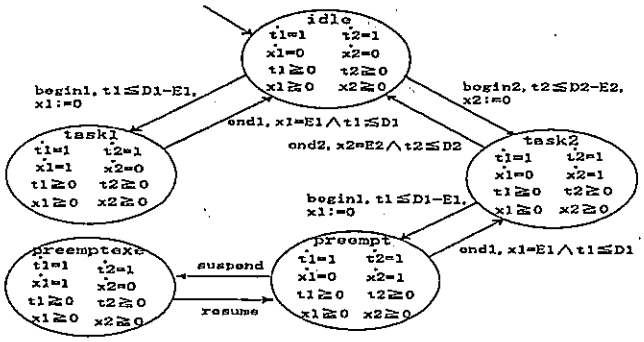


図 10: preemptive スケジューラの仕様記述

[タイミングダイアグラム例]

次に, タイミングダイアグラムにより, タスク１とタスク２の動作を図１１に例示する. 図１１では, タスク１の周期 $T1$, 実行時間 $E1$, デッドライン $D1$, タスク２の周期 $T2$, 実行時間 $E2$, デッドラ

イン $D2$ とする. ここでは, タスク 1 がタスク 2 より優先度が高くて, $T1 < T2$ である. 図の中で, 点線部分では, タスク 2 の実行中にタスク 1 がプリエンプティブに実行されて, タスク 1 の実行終了後に, タスク 2 が再開されている.

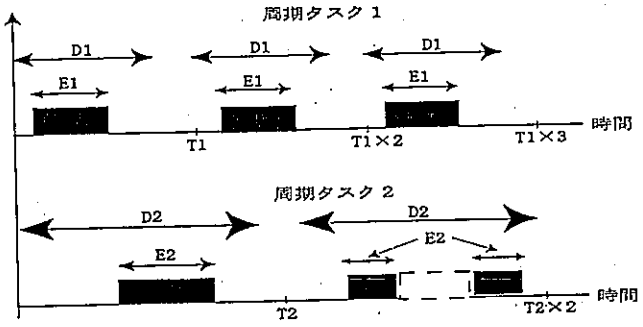


図 11: タイミングダイアグラム

4.3 リアルタイムソフトウェアの正当性検証手法

本節では, 以下の 2 つの手法により, リアルタイムソフトウェアの正当性検証を行う.

1. 仕様とリアルタイムオペレーティングシステムとの並列合成が元の仕様を正しく詳細化していることを検証する. これを実現するために, ハイブリッドオートマトンの演繹的詳細化検証手法を開発する.
2. M. Joseph らの提案した最悪応答時間計算法 [27] を使って, 最悪応答時間を計算して, 最悪応答時間がデッドライン以下であることをチェックして, スケジューラビリティを検証する.

以下では, まず, ハイブリッドオートマトンの演繹的詳細化検証手法を説明して, 次に, スケジューラビリティ検証手法を説明する.

4.3.1 ハイブリッドオートマトンの詳細化検証手法

本節では, 仕様がリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすことの検証を詳細化検証で行う. 本論文における詳細化検証では, ハイブリッドオートマトンをフェーズ遷移システムに変換して, 詳細化検証の公理系により行う [34]. この検証では, 周期タスクとスケジューラの並列合成が周期タスクを詳細化していることを検証する.

[準備]

まず, システム変数の有限集合 Var を考える. システム変数は整数や実数などの型を持つ. ここで, R^n を実数の集合とする. 状態は $\sigma : Var \rightarrow R^n$ であり, Var の中の変数の型整合解釈である. 状態の集合を Σ_V とする. なお, 変数の型としては整数やブール値などがあるが, いずれも実数の部分集合なので, 状態を $\sigma : Var \rightarrow R^n$ とする.

時間は非負実数 R^+ によってモデル化される. 区間 $I = [a, b)$ は $a \leq t < b$ であるような点 $t \in R^+$ の集合である. ここで, $a, b \in R^+$ かつ $a < b$ である. $I = [a, b)$ を区間とする. 関数 $f : I \rightarrow R^n$ は, 以下の条件を満たすならば, I 上で各部分がスムーズである :

1. a において, f の右極限とすべての右導関数が存在する.
2. すべての点 $t \in I$ において, f の右極限と左極限, すべての右導関数と左導関数が存在して, f は右または左から連続である.
3. b において, f の左極限とすべての左導関数が存在する.

2つの関数 $f, g : I \rightarrow R^n$ は有限の点以外のすべてにおいて一致するならば, I 上において区別不能である. もし, 各部分がスムーズな2つの関数が開区間 I 上で区別不能ならば, 2つの関数は I における, すべての極限と導関数において一致する.

変数 V 上のフェーズ $P =< I, f >$ は以下の順序対から構成される：

1. 区間 $I = [a, b)$ である.
2. 関数 $f_x : I \rightarrow R^n$ の型整合族は $f = \{f_x | x \in Var\}$ である. なお, 関数 $f_x : I \rightarrow R^n$ は区間 $I = [a, b)$ 上で各部分がスムーズであり, 各点 $t \in I$ に変数 $x \in Var$ の値を割り付ける.

フェーズ P はすべての実数値の時間 $t \in I$ に状態 $f(t) \in \Sigma_V$ を割り付ける.

以下では, フェーズ P の左終端の極限状態 $\overrightarrow{P} \in \Sigma_V$ を

$$\overrightarrow{P} = lim_{t \rightarrow a}\{f(t) | a < t < b\} \tag{3}$$

と書き, フェーズ P の右終端の極限状態 $\overleftarrow{P} \in \Sigma_V$ を

$$\overleftarrow{P} = lim_{t \rightarrow b}\{f(t) | a < t < b\} \tag{4}$$

と書く.

[フェーズ遷移システム]

まず, Pnueli らのフェーズ遷移システム [3] を拡張した, フェーズ遷移システムを定義する. このフェーズ遷移システムは観測可能な変数と観測不能な変数を区別するものである.

Definition 15 (フェーズ遷移システム)

フェーズ遷移システムは, $\mathrm{M} \coloneqq (Var, L, \Phi, \Theta, T)$ の5つ組で定義される. ここで,

1. Var はシステム変数の有限集合である.

2. $L \subseteq Var$ はローカル変数であり, システムの外部からは観測できない変数である.

3. Φ は Var 上のフェーズの不変式の有限集合である. 各フェーズ不変式 $\phi \in \Phi$ は表明式 $\rho_\phi(Var, \dot{Var})$ によって表現される. ここで, $\dot{Var}$ はシステム変数 Var の導関数である.

4. Θ は初期条件である. これは, すべての初期状態を特徴付ける表明である.

5. $\mathcal{T}$ は状態遷移の有限集合である. 各状態遷移 $\tau \in \mathcal{T}$ は表明式 $\rho_\tau(Var, Var\prime)$ によって表現される.

■

$\overline{P} = P_0, P_1, P_2, \ldots$ は無限なフェーズ列である. ここで, すべての $i \geq 0$ に対して, $P_i =< [a_i, a_{i+1}), f_i >$ である.

フェーズ列が以下の条件を満たすフェーズ列 $\overline{P} = P_0, P_1, P_2, \ldots$ と等しければ, このフェーズ列はフェーズ遷移システム M の計算である :

1. 初期条件 : もし $P_0 = [a, b)$ ならば Θ は a で成り立つ.

2. 連続動作 : すべての $0 \leq i < |\overline{P}|$ に対して, P_i が ϕ-フェーズであるようなフェーズ不変式 $\rho_\phi \in \Phi$ が存在する.

3. 離散的状態遷移 : すべての $0 \leq i < |\overline{P}| - 1$ に対して, $\rho_\tau(\overrightarrow{P_i}[Var], \overleftarrow{P_{i+1}}[Var])$ が成り立つような状態遷移 $\tau \in \mathcal{T}$ が存在する.

4. 発散性 : $\overline{P}$ は発散する.

次に, ハイブリッドオートマトンからフェーズ遷移システムへの変換方法を定義する.

Definition 16 (ハイブリッドオートマトンの変換方法)

ハイブリッドオートマトン $\mathrm{A} = (\vec{x}, V, local, inv, dif, E, act, Label, syn, E_\Theta)$ は，以下のように，フェーズ遷移システム $\mathrm{M} = (Var, L, \Phi, \Theta, T)$ に変換される：

1. $Var = \pi \cup \vec{x}$

   ただし，π は制御ロケーション V を表現するロケーション変数である．つまり，システム変数の有限集合 Var はロケーション変数 π とデータ変数 $\vec{x}$ である．

2. $L = local$

   つまり，ローカル変数 L はハイブリッドオートマトンのローカル変数 $local$ であり，$local \subseteq Var$ である．

3. $\Phi = \{\phi_{l_i} | l_i \in V\}$

   ここで，任意の $l_i \in V$ に対して，

   $$\rho_{\phi_{l_i}} : inv(l_i) \wedge (\pi = l_i) \wedge dif(l_i)$$

   である．

4. $\Theta = (x_1 = c_1 \wedge \ldots \wedge x_n = c_n) \wedge (\pi = l_i) \wedge inv(l_i)$

   ここで，l_i はエントリーロケーションであり，$(x_1 = c_1 \wedge \ldots \wedge x_n = c_n)$ はエントリーエッジ E_Θ のラベルである．

5. $T = \{\tau_{(l_i,l_j)} | (l_i, l_j) \in E\}$

   ここで，$e = (l_i, l_j) \in E$ に対して，

   $$\rho_{\tau_{(l_i,l_j)}} : act(e) \wedge \pi = l_i \wedge \pi\prime = l_j$$

   である．

なお, 同期ラベルの集合 $Label$ とラベリング関数 syn は状態遷移の名前 $label \in Label$ を定義するものであり, $syn(e) = label$ である.

■

[詳細化検証の公理系]

周期タスクと並列合成は, それぞれ $\mathrm{M} = (Var, L, \Phi, \Theta, \mathcal{T})$ の 5 つ組で定義される. $L \subseteq Var$ はローカル変数であり, システムの外部からは観測できない変数である.

周期タスク M^A の観測可能なフェーズが並列合成 M^C の観測可能なフェーズを包含して, すべての並列合成 M^C の状態遷移に対して, ある周期タスク M^A の状態遷移が存在するならば, M^C は M^A を詳細化すると呼び, $M^C \sqsubseteq M^A$ と表記する. 以下に, 詳細化検証ルールを定義する :

Definition 17 (詳細化検証ルール)

並列合成 $M^C = (Var^C, L^C, \Phi^C, \Theta^C, \mathcal{T}^C)$ が周期タスク $M^A = (Var^A, L^A, \Phi^A, \Theta^A, \mathcal{T}^A)$ を観測可能な動作において詳細化していることを検証するルールを以下に示す :

なお, M^A の観測可能な変数 (つまり, $Var^A - L^A$) は M^C の観測可能な変数 (つまり, $Var^C - L^C$) である.

1. $\Theta^C \rightarrow \Theta^A[\alpha]$
2. $\forall \phi^C \in \Phi^C$ に対して以下が成り立つ :
   $\rho_{\phi^C} \rightarrow \bigvee_{\phi^A \in \Phi^A} \rho_{\phi^A}[\alpha]$
3. $\forall \tau^C \in \mathcal{T}^C$ に対して以下が成り立つ :
   $\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$
4. ---
5. $M^C \sqsubseteq M^A$

これは詳細化を証明するルールであり, ある α が存在して, 1., 2. と 3. を満たすとき, 5. のように $\sqsubseteq$ が定義されることを意味する. ここで, ルールの各行は, 以下を意味する : 1 行目は初期条件

Θ^C ならば初期条件 $\Theta^A[\alpha]$ であることを意味する．2行目は $\rho_{\phi C}$ のフェーズ不変式が $\rho_{\phi A}[\alpha]$ のフェーズ不変式に包含されることを意味する．3行目は $\rho_{\tau C}$ の状態遷移が $\rho_{\tau A}[\alpha]$ の状態遷移により説明されることを意味する．4行目は前提と結論を分離するラインである．5行目は結論 $M^C \sqsubseteq M^A$ であり，並列合成が周期タスクを正しく詳細化していることを意味する．つまり，仕様がリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすことを意味する．

ここで，α は L^A の式を Var^C の式に置き換える写像を意味する．なお，$X_\alpha(Var^C)$ により，Var^C の式を変数 $X \in L^A$ に割り当てることを表現する．

以下のように，α はフェーズ上の写像を与える：

P^C は M^C の計算の中に現れるフェーズとする．この P^C をリアルタイムソフトウェアのフェーズと呼ぶ．リアルタイムソフトウェアのフェーズ P^C に対応する周期タスクのフェーズが $P^A = m_\alpha(P^C)$ であることは，P^A の中の周期タスクの任意のローカル変数 $X \in L^A$ の値が式 $X_\alpha(Var^C)$ の値であることを意味する．ここで，m_α は M^C から M^A への詳細化写像と呼ぶ．この詳細化写像は *Abadi* らの詳細化写像 *[12]* である．

■

[仕様に対するオペレーティングシステムの正当性]

ここでは，上記の詳細化検証の公理系により，仕様がリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすことを検証する手法を定義する．

Definition 18 (仕様に対する正当性検証)

仕様に対するオペレーティングシステムの正当性を検証する手法を定義する．つまり，仕様がリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすことを検証する手法を定義する．

以下が成り立つならば, 仕様はリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすと言える：

$$M^C \sqsubseteq M^A.$$

ただし, M^A は仕様のフェーズ遷移システム, M^C は仕様とリアルタイムオペレーティングシステムの並列合成のフェーズ遷移システムである. なお, 上記では, 仕様は周期タスクのことである.

■

$M^C \sqsubseteq M^A$ は, 適当な α により, M^C が M^A を詳細化していることを表しており, M^C が M^A により模倣されることを意味する. つまり, M^A の観測可能な動作が M^C の観測可能な動作を包含することを意味する. ただし, M^A は仕様のフェーズ遷移システム, M^C は仕様とリアルタイムオペレーティングシステムの並列合成のフェーズ遷移システムである. ゆえに, $M^C \sqsubseteq M^A$ を満たせば, 仕様はリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすと言える.

また, $M^C \sqsubseteq M^A$ が成り立たない場合としては, 例えば, スケジューラの仕様が周期タスクの仕様 M^A を完全に包含する場合などが考えられる. この場合は, 周期タスクの仕様がスケジューラの仕様に包含されて, 実装された並列合成がもとの周期タスクの仕様よりも大きな観測可能な動作を行うので, 並列合成はもとの周期タスクの仕様を満たさないと言える.

4.3.2 スケジューラビリティ検証

ここでは, 既存のスケジューリング理論を使って, スケジューラビリティを検証する.

まず, 既存のスケジューリング理論上のスケジューラビリティ検証を説明する [27]. タスク i の周

期を Ti, 実行時間を Ei, デッドラインを Di とする $(i = 1, .., n)$. Joseph らがタスク i の最悪応答時間 Ri を定義した. タスク i の $n+1$ 番目の応答時間 $Ri^{(n+1)}$ は以下のように定義される：

$$Ri^{(n+1)} = Ei + \sum_{j=1}^{i-1} \lceil \frac{Ri^{(n)}}{Tj} \rceil \times E_j \tag{5}$$

ここで, $Ri^{(0)}$ を Ei として, $Ri = \lim_{n\to\infty} Ri^{(n)}$ とする. ただし, $\lceil \frac{Ri^{(n)}}{Tj} \rceil$ は $\frac{Ri^{(n)}}{Tj}$ より大きいかまたは等しい最小の整数である.

Definition 19 (スケジューラビリティ検証)

リアルタイムソフトウェアは以下の条件が満たされるならば, スケジューラブルである.

すべての周期タスク i に対して, $Ri \leq Di(i=1,..,n)$ が成り立つ.

ただし, Ri は最悪応答時間, Di はデッドラインである.

■

4.4 正当性検証の事例

まず, 仕様である周期タスクのフェーズ遷移システムを構成して, 次に, 周期タスクとスケジューラの並列合成のフェーズ遷移システムを構成して, 最後に, 仕様に対するオペレーティングシステムの正当性とスケジューラビリティを検証する.

4.4.1 周期タスクのフェーズ遷移システムの構成

まず, ハイブリッドオートマトンの並列合成により, 周期タスクのハイブリッドオートマトンを構成する. 周期タスクのハイブリッドオートマトンを図１２に示す. なお, ここでは周期タスクが仕様である.

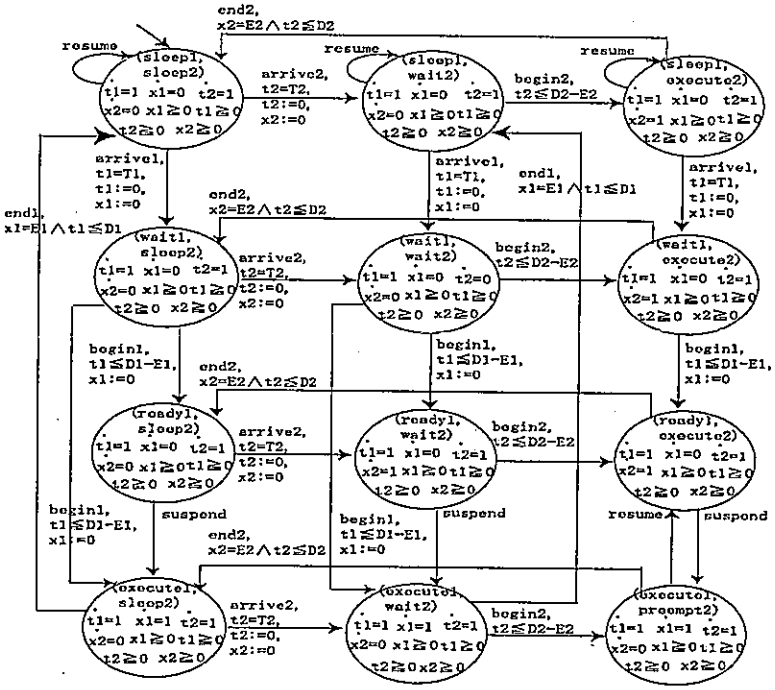


図 12: 周期タスクのハイブリッドオートマトン

次に, 図１２の周期タスクのハイブリッドオートマトンから, フェーズ遷移システムを構成する. 周期タスクのフェーズ遷移システム $M^A = (Var^A, L^A, \Phi^A, \Theta^A, \mathcal{T}^A)$ を構成する.

次に, 図１２の周期タスクのハイブリッドオートマトンから, フェーズ遷移システムを構成する. 周期タスクのフェーズ遷移システム $M^A = (Var^A, L^A, \Phi^A, \Theta^A, \mathcal{T}^A)$ を構成する.

1. $Var^A = \{t_1, t_2, x_1, x_2, {\pi_1}^A, {\pi_2}^A\}$.

2. $L^A = \{{\pi_1}^A, {\pi_2}^A\}$.

3. $\Phi^A = \{\phi_{(sleep_1,sleep_2)}, \phi_{(sleep_1,wait_2)}, \phi_{(sleep_1,execute_2)}, \phi_{(wait_1,sleep_2)}, \phi_{(wait_1,wait_2)},$

$\phi_{(wait_1,execute_2)}, \phi_{(ready_1,sleep_2)}, \phi_{(ready_1,wait_2)}, \phi_{(ready_1,execute_2)}, \phi_{(execute_1,sleep_2)},$

$\phi_{(execute_1,wait_2)}, \phi_{(execute_1,preempt_2)}\}$.

(a) ${\rho_{\phi_{(sleep_1,sleep_2)}}}^A : {\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} =$

$1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

(b) ${\rho_{\phi_{(sleep_1,wait_2)}}}^A : {\pi_1}^A = sleep_1 \wedge {\pi_2}^A = wait_2 \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} =$

$1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

.....

.....

4. $\Theta^A : {\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge t_1 = 0 \wedge t_2 = 0 \wedge x_1 = 0 \wedge x_2 = 0 \wedge \dot{t_1} = 0 \wedge \dot{t_2} = 0 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

5. $\mathcal{T}^A = \{{\tau_{(sleep_1,sleep_2)(sleep_1,wait_2)}}^A, {\tau_{(sleep_1,wait_2)(sleep_1,execute_2)}}^A,$

${\tau_{(sleep_1,execute_2)(sleep_1,sleep_2)}}^A, {\tau_{(sleep_1,execute_2)(wait_1,execute_2)}}^A,$

${\tau_{(wait_1,sleep_2)(wait_1,wait_2)}}^A, ...\}$.

(a) ${\rho_{\tau_{(sleep_1,sleep_2)(sleep_1,wait_2)}}}^A : {\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge {\pi_1}^{A\prime} = sleep_1 \wedge {\pi_2}^{A\prime} = wait_2 \wedge 0 \leq$

$t_1 \wedge t_2 = T_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge 0 \leq t_1\prime \wedge t_2\prime = 0 \wedge 0 \leq$

$x_1\prime \wedge x_2\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$

(b) ${\rho_{\tau_{(sleep_1,wait_2)(sleep_1,execute_2)}}}^A : {\pi_1}^A = sleep_1 \wedge {\pi_2}^A = wait_2 \wedge {\pi_1}^{A\prime} = sleep_1 \wedge {\pi_2}^{A\prime} = execute_2 \wedge 0 \leq$

$t_1 \wedge t_2 \leq D_2 - E_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge 0 \leq t_1\prime \wedge t_2\prime \leq$

$D_2 - E_2 \wedge 0 \leq x_1\prime \wedge x_2\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 1$

.....

.....

4.4.2 並列合成のフェーズ遷移システムの構成

まず, 周期タスクとプリエンプティブスケジューラの並列合成のハイブリッドオートマトンを構成する. なお, 状態数が少ないのは, 並列合成により, 不変性などの条件が矛盾した状態が構成されたためである. 並列合成のハイブリッドオートマトンを図１３に示す.

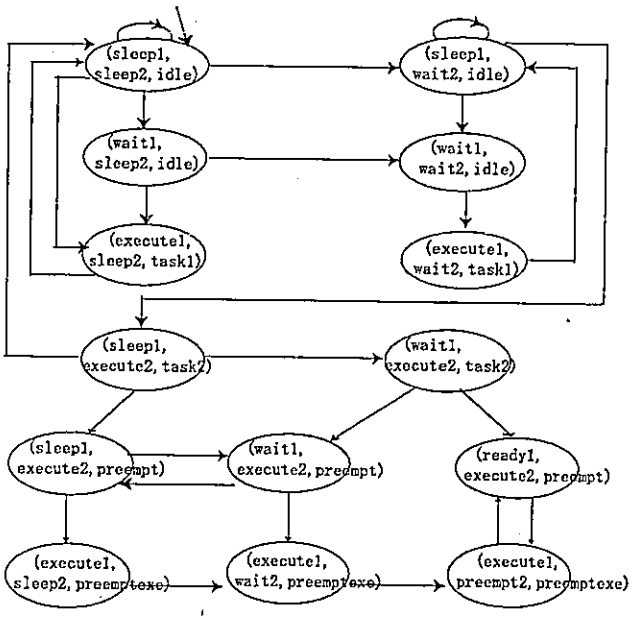


図 13: 並列合成のハイブリッドオートマトン

次に, 図１３の並列合成のハイブリッドオートマトンから, フェーズ遷移システムを構成する. リアルタイムソフトウェアのフェーズ遷移システム $M^C = (Var^C, L^C, \Phi^C, \Theta^C, T^C)$ を構成する.

1. $Var^C = \{t_1, t_2, x_1, x_2, {\pi_1}^C, {\pi_2}^C, {\pi_3}^C\}$.

2. $L^C = \{{\pi_1}^C, {\pi_2}^C, {\pi_3}^C\}$.

3. $\Phi^C = \{\ \phi_{(sleep_1,sleep_2,idle)}, \phi_{(sleep_1,wait_2,idle)}, \phi_{(wait_1,sleep_2,idle)}, \phi_{(wait_1,wait_2,idle)},$

$\phi_{(execute_1,sleep_2,task_1)}, \phi_{(execute_1,wait_2,task_1)}, \phi_{(sleep_1,execute_2,task_2)}, \phi_{(wait_1,execute_2,task_2)},$

$.....\}$.

(a) ${\rho_{\phi_{(sleep_1,sleep_2,idle)}}}^C : {\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

(b) ${\rho_{\phi_{(sleep_1,wait_2,idle)}}}^C : {\pi_1}^C = sleep_1 \wedge {\pi_2}^C = wait_2 \wedge {\pi_3}^C = idle \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

.....

.....

4. $\Theta^C : {\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge t_1 = 0 \wedge t_2 = 0 \wedge x_1 = 0 \wedge x_2 = 0 \wedge \dot{t_1} = 0 \wedge \dot{t_2} = 0 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$

5. $\mathcal{T}^C = \{{\tau_{(sleep_1,sleep_2,idle)(wait_1,sleep_2,idle)}}^C, {\tau_{(sleep_1,sleep_2,idle)(sleep_1,wait_2,idle)}}^C,$

${\tau_{(wait_1,sleep_2,idle)(wait_1,wait_2,idle)}}^C, {\tau_{(sleep_1,wait_2,idle)(sleep_1,execute_2,task_2)}}^C, ...\}$.

(a) ${\rho_{\tau_{(sleep_1,sleep_2,idle)(wait_1,sleep_2,idle)}}}^C : {\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^{C\prime} = wait_1 \wedge {\pi_2}^{C\prime} = sleep_2 \wedge {\pi_3}^{C\prime} = idle \wedge t_1 = T_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge t_1\prime = 0 \wedge 0 \leq t_2\prime \wedge x_1\prime = 0 \wedge 0 \leq x_2\prime \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$

(b) ${\rho_{\tau_{(sleep_1,sleep_2,idle)(sleep_1,wait_2,idle)}}}^C : {\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^{C\prime} = sleep_1 \wedge {\pi_2}^{C\prime} = wait_2 \wedge {\pi_3}^{C\prime} = idle \wedge 0 \leq t_1 \wedge t_2 = T_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} =$

$$0 \wedge 0 \leq t_1\prime \wedge t_2\prime = 0 \wedge 0 \leq x_1\prime \wedge x_2\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$$

.....

.....

4.4.3 仕様に対するオペレーティングシステムの検証例

次に, 仕様に対するリアルタイムオペレーティングシステムの検証例を示す.

まず, 詳細化検証の公理系により, $M^C = (Var^C, L^C, \Phi^C, \Theta^C, T^C)$ が $M^A = (Var^A, L^A, \Phi^A, \Theta^A, T^A)$ を観測可能な動作において詳細化していることを検証する.

ここで, α を以下のように定義する :

$\alpha : ({\pi_1}^A, {\pi_2}^A) \rightarrow$

$$
\begin{array}{ll}
if & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (sleep_1, sleep_2, idle) \\
then & (sleep_1, sleep_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (wait_1, sleep_2, idle) \\
then & (wait_1, sleep_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (execute_1, sleep_2, state_1) \\
then & (execute_1, sleep_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (sleep_1, wait_2, idle) \\
then & (sleep_1, wait_2, idle) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (wait_1, wait_2, idle) \\
then & (wait_1, wait_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (execute_1, wait_2, state_1) \\
then & (execute_1, wait_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (sleep_1, execute_2, state_2) \\
then & (sleep_1, execute_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (wait_1, execute_2, state_2) \\
then & (wait_1, execute_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (ready_1, execute_2, preempt) \\
then & (ready_1, execute_2) \\
elseif & (\pi_1{}^C, \pi_2{}^C, \pi_3{}^C) = (execute_1, preempt_2, preemptexe) \\
then & (execute_1, preempt_2)
\end{array}
$$

ただし, $state_1$ は $task_1$ または $preemptexe$, $state_2$ は $task_2$ または $preempt$ である.

1. $\Theta^C \rightarrow \Theta^A[\alpha]$ の検証:

Θ^C : ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge t_1 = 0 \wedge t_2 = 0 \wedge x_1 = 0 \wedge x_2 = 0 \wedge \dot{t_1} = 0 \wedge \dot{t_2} = 0 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ 及び Θ^A : ${\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge t_1 = 0 \wedge t_2 = 0 \wedge x_1 = 0 \wedge x_2 = 0 \wedge \dot{t_1} = 0 \wedge \dot{t_2} = 0 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ である.

また, $\Theta^A[\alpha]$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge t_1 = 0 \wedge t_2 = 0 \wedge x_1 = 0 \wedge x_2 = 0 \wedge \dot{t_1} = 0 \wedge \dot{t_2} = 0 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ である.

以上より, 観測可能な動作において,

$$\Theta^C \rightarrow \Theta^A[\alpha]$$

は成り立つ.

2. $\forall \phi^C \in \Phi^C$ に対して以下が成り立つことの検証 :

$$\rho_{\phi^C} \rightarrow \bigvee_{\phi^A \in \Phi^A} \rho_{\phi^A}[\alpha]$$

(a)${\rho_{\phi_{(sleep_1, sleep_2, idle)}}}^C$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ 及び ${\rho_{\phi_{(sleep_1, sleep_2)}}}^A$: ${\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ である.

また, ${\rho_{\phi_{(sleep_1, sleep_2)}}}^A[\alpha]$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge \dot{x_2} = 0 \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0$ である.

以上より, 観測可能な動作において,

$${\rho_{\phi_{(idle_1, idle_2, idle)}}}^C \rightarrow {\rho_{\phi_{(idle_1, idle_2)}}}^A[\alpha]$$

は成り立つ.

(b)${\rho_{\phi_{(wait_1,sleep_2,idle)}}}^C$: ${\pi_1}^C = wait_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ 及び ${\rho_{\phi_{(wait_1,sleep_2)}}}^A$: ${\pi_1}^A = wait_1 \wedge {\pi_2}^A = sleep_2 \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ である.

また, ${\rho_{\phi_{(wait_1,sleep_2)}}}^A[\alpha]$: ${\pi_1}^C = wait_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge 0 \leq t_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0$ である.

以上より, 観測可能な動作において,

$${\rho_{\phi_{(wait_1,sleep_2,idle)}}}^C \rightarrow {\rho_{\phi_{(wait_1,sleep_2)}}}^A[\alpha]$$

は成り立つ.

.....

.....

3. $\forall {\tau^C} \in \mathcal{T}^C$ に対して以下が成り立つことの検証:

$$\rho_{\tau^C} \rightarrow \bigvee_{\tau^A \in \mathcal{T}^A} \rho_{\tau^A}[\alpha]$$

(a)${\rho_{\tau_{(sleep_1,sleep_2,idle)(wait_1,sleep_2,idle)}}}^C$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^C{}_I = wait_1 \wedge {\pi_2}^C{}_I = sleep_2 \wedge {\pi_3}^C{}_I = idle \wedge t_1 = T_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge {t_1}_I = 0 \wedge 0 \leq {t_2}_I \wedge {x_1}_I = 0 \wedge 0 \leq {x_2}_I \wedge \dot{t_1}_I = 1 \wedge \dot{t_2}_I = 1 \wedge \dot{x_1}_I = 0 \wedge \dot{x_2}_I = 0$ 及び ${\rho_{\tau_{(sleep_1,sleep_2)(wait_1,sleep_2)}}}^A$: ${\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge {\pi_1}^A{}_I = wait_1 \wedge {\pi_2}^A{}_I = sleep_2 \wedge t_1 = T_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge {t_1}_I = 0 \wedge 0 \leq {t_2}_I \wedge {x_1}_I = 0 \wedge 0 \leq {x_2}_I \wedge \dot{t_1}_I = 1 \wedge \dot{t_2}_I = 1 \wedge \dot{x_1}_I = 0 \wedge \dot{x_2}_I = 0$

である.

また, ${\rho_{\tau_{(sleep_1,sleep_2)(wait_1,sleep_2)}}}^A[\alpha]$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^C{}_I = wait_1 \wedge {\pi_2}^C{}_I =$

$sleep_2 \wedge {\pi_3}^C{}\prime = idle \wedge t_1 = T_1 \wedge 0 \leq t_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge {t_1}\prime = 0 \wedge 0 \leq {t_2}\prime \wedge {x_1}\prime = 0 \wedge 0 \leq {x_2}\prime \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$ である.

以上より, 観測可能な動作において,

$$\rho_{\tau(sleep_1,sleep_2,idle)(wait_1,sleep_2,idle)}{}^C \rightarrow \rho_{\tau(sleep_1,sleep_2)(wait_1,sleep_2)}{}^A[\alpha]$$

は成り立つ.

(b)$\rho_{\tau(sleep_1,sleep_2,idle)(sleep_1,wait_2,idle)}{}^C$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^C{}\prime = sleep_1 \wedge {\pi_2}^C{}\prime = wait_2 \wedge {\pi_3}^C{}\prime = idle \wedge 0 \leq t_1 \wedge t_2 = T_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge 0 \leq {t_1}\prime \wedge {t_2}\prime = 0 \wedge 0 \leq {x_1}\prime \wedge {x_2}\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$ 及び $\rho_{\tau(sleep_1,sleep_2)(sleep_1,wait_2)}{}^A$: ${\pi_1}^A = sleep_1 \wedge {\pi_2}^A = sleep_2 \wedge {\pi_1}^A{}\prime = sleep_1 \wedge {\pi_2}^A{}\prime = wait_2 \wedge 0 \leq t_1 \wedge t_2 = T_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge 0 \leq {t_1}\prime \wedge {t_2}\prime = 0 \wedge 0 \leq {x_1}\prime \wedge {x_2}\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$ である.

また, $\rho_{\tau(sleep_1,sleep_2)(sleep_1,wait_2)}{}^A[\alpha]$: ${\pi_1}^C = sleep_1 \wedge {\pi_2}^C = sleep_2 \wedge {\pi_3}^C = idle \wedge {\pi_1}^C{}\prime = sleep_1 \wedge {\pi_2}^C{}\prime = wait_2 \wedge {\pi_3}^C{}\prime = idle \wedge 0 \leq t_1 \wedge t_2 = T_2 \wedge 0 \leq x_1 \wedge 0 \leq x_2 \wedge \dot{t_1} = 1 \wedge \dot{t_2} = 1 \wedge \dot{x_1} = 0 \wedge \dot{x_2} = 0 \wedge 0 \leq {t_1}\prime \wedge {t_2}\prime = 0 \wedge 0 \leq {x_1}\prime \wedge {x_2}\prime = 0 \wedge \dot{t_1}\prime = 1 \wedge \dot{t_2}\prime = 1 \wedge \dot{x_1}\prime = 0 \wedge \dot{x_2}\prime = 0$ である.

以上より, 観測可能な動作において,

$$\rho_{\tau(sleep_1,sleep_2,idle)(sleep_1,wait_2,idle)}{}^C \rightarrow \rho_{\tau(sleep_1,sleep_2)(sleep_1,wait_2)}{}^A[\alpha]$$

は成り立つ.

.....

.....

以上の (1), (2), (3) より, $M^C \sqsubseteq M^A$ が成り立つことが検証できた.

すなわち, 仕様に対するリアルタイムオペレーティングシステムの正当性が検証できた.

4.4.4 スケジューラビリティ検証例

次に, すべての周期タスク i に対して, $Ri \leq Di(i = 1, 2)$ が成り立つことを検証する.

Joseph の手法 [27] の (3) 式により, 最悪応答時間 Ri を計算する. ここで, タスク 1 の周期, 実行時間とデッドラインを, それぞれ $T1 = 300$, $E1 = 30$, $D1 = 250$ として, 優先度は 1 とする. タスク 2 の周期, 実行時間とデッドラインを, それぞれ $T2 = 500$, $E1 = 100$, $D1 = 450$ として, 優先度は 2 とする. なお, タスク 1 はタスク 2 より優先度が高い.

では, 以下に, $R1 \leq D1$ を検証する.

1. まず, $Ri^{(n+1)}$ の計算が収束するかどうかを判定する. Joseph の手法 [27] の判定式は以下である :

$$\sum_{j=1}^{i} (E_j \times \frac{M_i/T_j}{M_i}) \leq 1$$

ここで, $M_i = LCM(\{T_1, T_2, \ldots, T_i\})$ である.

この例では,

$$(30 \times \frac{1500/300}{1500}) \leq 1$$

が成り立つ. ゆえに, $R1^{(n+1)}$ の計算は収束する.

2. 次に, $Ri^{(n+1)}$ を計算する. Joseph の手法 [27] の計算式は以下である :

$R(t, t\prime, i) = if\ Comp([t, t\prime), i) = 0\ then\ t\prime$

$else\ R(t\prime, t\prime + Comp([t, t\prime), i))$

ここで, $R(t, t\prime, i)$ は区間 $[t, t\prime)$ の優先度 i の応答時間, $Comp([t, t\prime), i)$ は優先度 i より高いタスクの計算時間である. なお, $Comp([t, t\prime), i)$ は以下である：

$$Comp([t, t\prime), i) = \sum_{j=1}^{i-1} Inputs([t, t\prime), j) \times Ej$$

ただし, $Inputs([t, t\prime), j) = \lceil t\prime/T_j \rceil - \lceil t/T_j \rceil$ である.

この例では,

$R(0, 30, 1) = if\ Comp([0, 30), 1) = 0\ then\ 30$

$\qquad\qquad else\ R(30, 30 + Comp([0, 30), 1))$

である. 各項の計算は以下のとおりである：

$Comp([0, 30), 1) = Inputs([0, 30), 1) \times 30 = 1 \times 30 = 30$

$R(30, 60, 1) = if\ Comp([30, 60), 1) = 0\ then\ 60$

$\qquad\qquad else\ R(60, 60 + Comp([30, 60), 1), 1) \qquad = 60$

$R(60, 60, 1) = if\ Comp([60, 60), 1) = 0\ then\ 60$

$\qquad\qquad else\ R(60, 60 + Comp([60, 60), 1), 1) \qquad = 60$

以上より, $R(0, 30, 1) = 60$ である.

以上より, $R1 \leq D1$ が検証できた.

また, $R2 \leq D2$ も同様に検証できる. 紙面の都合上より, $R2 \leq D2$ の検証の詳細は省略する.

以上より, すべての周期タスク i に対して, $Ri \leq Di (i = 1, 2)$ が成り立つことが検証できた. すなわち, すべての周期タスク i はスケジューラブルである.

4.5 むすび

本章では, まず, リアルタイムソフトウェアをハイブリッドオートマトンで仕様記述して, 次に, 最悪応答時間がデッドライン以下であることをチェックして, 次に, ハイブリッドオートマトンの演繹的詳細化検証手法により, 仕様がリアルタイムオペレーティングシステム上で動作しても元の仕様を満たすことを検証した. また, 周期タスクが固定優先度のプリエンプティブスケジューリングで動作するリアルタイムソフトウェアを想定して, ハイブリッドオートマトンにより, 仕様記述して, スケジューラビリティを検証する事例を示した. ハイブリッドオートマトンによるリアルタイムソフトウェアの仕様記述と詳細化検証による正当性検証としては, 本論文は世界で初めての試みである.

今後の課題として, 現在, 以下の研究を進めている：

1. 本論文では, 簡単なリアルタイムソフトウェアを対象として仕様記述と検証を行った. 実際のリアルタイムソフトウェアの特徴をより捉えた仕様記述が必要である. 現在, リソースアクセスやバッファなどの重要なリアルタイムソフトウェアの特徴を組み込んだ仕様記述を行っている. さらに, リアルタイムプログラムから仕様記述への自動変換も検討している.
2. 本論文では, 手作業により, 詳細化検証を行った. 現在, 定理証明器 [35, 36] 上に, 詳細化検証を実装している.
3. 演繹的検証は写像の発見などの手作業が多いが, 自動検証で問題となる状態数組み合わせ爆発にある程度対処できる. 今度は, 演繹的検証と自動検証との融合により, 効率化を図る予定である.

5 ハイブリッドシステムによる開発方法論

5.1 まえがき

ハイブリッドシステムはアナログ環境に組み込まれたデジタルな実時間システムであり, アナログ動作とデジタル動作が混在している [2]. 自動車や飛行機などの多くのハイブリッドシステムはsafety-criticalな状況で動作するので, 形式的な開発方法論による信頼性保証が重要である. 従来のハイブリッドシステムの開発方法論としては, 制御理論のアプローチから, $Matrix_x$ [37] や $MATLAB$ [38] などのCADツールにより, ブロック線図を設計して, その動作をシミュレーションにより確認するものである. しかし, この方法論では, 設計の正当性の確認が困難であったり, デジタルな動作がうまく表現できない. 一方, 近年, 計算機科学のアプローチから, ハイブリッドシステムを形式的に開発する方法論が研究されている. すなわち, アナログ動作を表現する制御方程式により, 状態遷移システムを拡張した, フェーズ遷移システム [3] やハイブリッドオートマトン [4], ハイブリッドI/Oオートマトン [5] を用いて, ハイブリッドシステムを仕様記述して, 検証するといった形式的手法である. しかし, この手法では, デジタルな動作の表現の中に, アナログな動作の表現を埋め込むために, アナログ動作が支配的なハイブリッドシステムの仕様記述には適さない.

そこで, 本論文では, アナログ動作が支配的なハイブリッドシステムの形式的開発方法論を提案する. 本論文は, 既知のハイブリッドオートマトンなどの計算機科学の技術と既知の $Matrix_x$ などの制御の技術, つまり異分野の技術, を統合化することによって, ハイブリッドシステムの体系的な開発手法を構築して, 実験的に有効性を示す. このような試みは, 既存研究にはまったく存在せず, 新規性がある. 提案する形式的開発方法論では, 抽象度の高い仕様から低い仕様までを階層的に仕様記述する.

抽象度の高い仕様では, $Matrix_x$ を中心的に使いながら, ハイブリッドシステムの構造を仕様記述する. また, 抽象度の低い仕様では, ハイブリッドオートマトンを中心的に使いながら, ハイブリッドシステムの動作を仕様記述する. 本論文で提案するハイブリッドシステムの形式的開発方法論は以下から構成される：

1. まず, $Matrix_x$ とハイブリッドオートマトンにより, ハイブリッドシステムを階層的に仕様記述する.
2. 次に, $Matrix_x$ からハイブリッドオートマトンを構成することにより, ハイブリッドオートマトンの並列合成としてハイブリッドシステムを表現する.
3. 最後に, 近似手法により, ハイブリッドオートマトンから線形ハイブリッドオートマトンを構成することによって, モデル検査により, ハイブリッドシステムの正当性を検証する.

本章と同様な狙いの既存研究は存在しない. 類似研究としては, 以下の２つが存在する：１つ目では, T. Villa らが自動車エンジン制御システムをハイブリッドオートマトンの並列合成としてモデル化して, ハイブリッドオートマトンを線形ハイブリッドオートマトンに近似して, モデル検査を実現している [39]. ２つ目では, T. Stauner らが自動車サスペンション制御システムをハイブリッドオートマトンの並列合成としてモデル化して, 上記と同様に, ハイブリッドオートマトンを線形ハイブリッドオートマトンに近似して, モデル検査を実現している [40]. 上記の２つの既存研究は, ハイブリッドオートマトンのモデル検査手法に関する研究であり, どのようにハイブリッドシステムをハイブリッドオートマトンにモデル化するかの手法に言及していない. 一方, 本研究では, 既知のハイブリッドオートマトンなどの計算機科学の技術と既知の $Matrix_x$ などの制御の技術, つまり異分野の技術,

を統合化することによって構築した形式的開発方法論により, ハイブリッドシステムをハイブリッドオートマトンへとモデル化する手法を主要な対象としている. ゆえに, 本研究は上記の既存研究と異なる.

以降の本章の構成は以下のとおりである. 5－2節では, ハイブリッドシステムの形式的開発方法論を提案する. 5－3節では, 事例により提案手法の有効性を示す. 最後に, 5－4節では, まとめと今後の課題を述べる.

5.2 ハイブリッドシステムの形式的開発方法論

ここでは, ハイブリッドシステムの形式的開発方法論を定義する. 形式的開発方法論では, 抽象度の高い仕様から低い仕様までを階層的に仕様記述する. 抽象度の高い仕様では, $Matrix_x$ を中心的に使いながら, ハイブリッドシステムの構造を仕様記述する. 一方, 抽象度の低い仕様では, ハイブリッドオートマトンを中心的に使いながら, ハイブリッドシステムの動作を仕様記述する. すなわち, ハイブリッドシステムの構造を段階的に詳細化しながら, ハイブリッドシステムの動作を仕様記述する.

形式的開発方法論は以下のとおりであり, その概念図を図１４に示す：

1. まず, 構造的概観図により, ハイブリッドシステムの構造を定義する. なお, 構造的概観図は $Matrix_x$ のブロック線図を基本的に用いるが, 以下の点が特徴である：

   (a) 構造的概観図では, 構成要素間の構造的な関係のみを表現する.

   (b) 外乱や外部からの入出力のみを表現して, 構成要素間を流れる内部の操作量は表現しない.

2. 次に, 上記１. の構造図の構成要素を段階的に詳細化して, $Matrix_x$, ハイブリッドオートマトン

または並列ハイブリッドオートマトンにより，その詳細を仕様記述する．

3. 次に，上記２．を繰り返して，$Matrix_x$，ハイブリッドオートマトンまたは並列ハイブリッドオートマトンの階層構造により，ハイブリッドシステムを表現する．

4. 次に，上記３．で作られた，最下層の $Matrix_x$，ハイブリッドオートマトンまたは並列ハイブリッドオートマトンに対して，以下の各処理を行う：

   (a) $Matrix_x$ の変換：$Matrix_x$ からハイブリッドオートマトンに変換する．そして，近似手法により，ハイブリッドオートマトンから線形ハイブリッドオートマトンを構成する．

   (b) ハイブリッドオートマトンの近似：近似手法により，ハイブリッドオートマトンから線形ハイブリッドオートマトンを構成する．

5. 最後に，上記４．で作られた線形ハイブリッドオートマトンの並列合成を構成して，モデル検査により，ハイブリッドシステムの検証を行う．

上記の形式的開発方法論では，$Matrix_x$ からハイブリッドオートマトンへの変換手法及びハイブリッドオートマトンから線形ハイブリッドオートマトンへの近似手法が重要であり，以降で詳細に議論する．簡単に言えば，以下のとおりである：$Matrix_x$ からハイブリッドオートマトンへの変換手法では，$Matrix_x$ のブロック線図より制御方程式を構成して，その制御方程式により支配されるハイブリッドオートマトンを構成する．一方，ハイブリッドオートマトンから線形ハイブリッドオートマトンへの近似手法では，モデル検査アルゴリズムを実現するために，制御方程式を線形近似する．

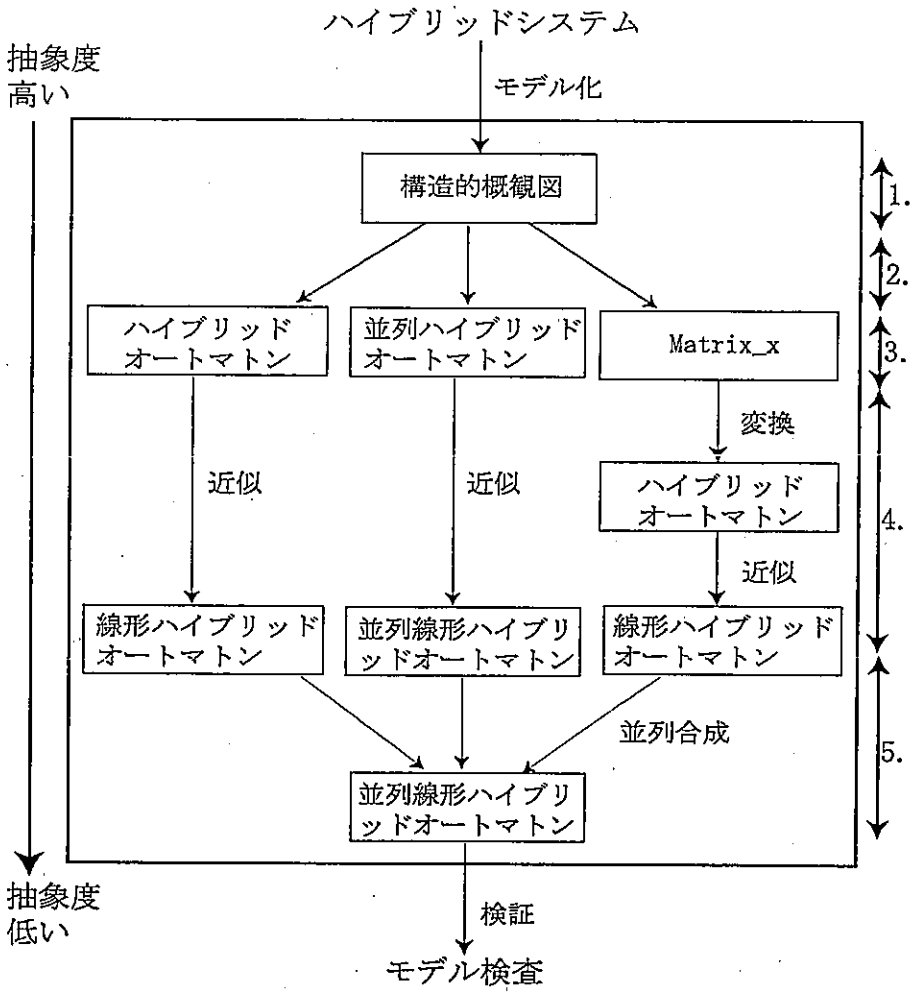


図 14: ハイブリッドシステムの形式的開発方法論

5.2.1 $Matrix_x$ からハイブリッドオートマトンへの変換手法

$Matrix_x$ からハイブリッドオートマトンへの変換手法では, $Matrix_x$ のブロック線図より, 制御を表現する微分方程式を構成して, その微分方程式により支配されるハイブリッドオートマトンを構成する.

[$Matrix_x$ のブロック線図とその微分方程式の構成]

まず, $Matrix_x$ のブロック線図を定義して, それが表現する微分方程式を定義する. なお, ブロック線図とそれが表現する微分方程式との関係は制御理論において知られている [41].

Definition 20 ($Matrix_x$ のブロック線図とその微分方程式の構成) $Matrix_x$ の代表的なブロック線図は以下の図１５のとおりである.

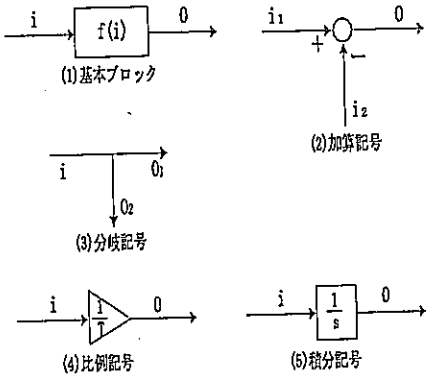


図 15: $Matrix_x$ のブロック線図の構成要素

各ブロック線図の要素から, それが表現する微分方程式は以下のように構成される：

1. 基本ブロック：$O = f(i)$.

2. 加算記号：$O = i_1 - i_2$.

3. 分岐記号 : $O_1 = i,\ O_2 = i.$

4. 比例記号 : $O = \frac{1}{T} i.$

5. 積分記号 : $O = \oint i dt.$

■

[ハイブリッドオートマトンの定義]

ハイブリッドオートマトンは実数値の変数の有限ベクタ X とラベル付きグラフ (V, E) から構成される. $\dot{X}$ によって, X の中の変数の一階微分のベクタを表現する. エッジ E は離散的なアクションを表現しており, 事前と事後の有限ベクタ X の値の制約がラベル付けされる. ノード V は連続的動作を表現しており, X と $\dot{X}$ の制約がラベル付けされる. ハイブリッドオートマトンの状態は瞬間的なアクション及び時間経過 (微分方程式に従う動作) により変化する.

次に, ハイブリッドオートマトンを定義する.

Definition 21 (ハイブリッドオートマトン)

ハイブリッドオートマトンは, $\mathbf{A} = (X, V, inv, init, flow, E, upd, jump, L, sync)$ の 10 つ組で定義される.

1. X は実数 $\vec{R}$ 上の変数の集合 $\{x_1, \ldots, x_n\}$ である. 割り当て s は実数 $\vec{R}$ の中の点であり, s の中の変数 x_i の値は s の i 番目の要素 s_i である.

本章の以降では, 以下の表記を用いる : Φ が X の中の変数上の述語のとき, $\Phi[X := s]$ が $true$ となる割り当て s の集合を表示するために, 《Φ》 と記す.

2. V はロケーションの有限集合である．ハイブリッドオートマトン A の状態は (v, s) である．ここで，$v \in V$ はロケーションであり，s は割り当てである．

3. inv はロケーションの集合 V から X の中の変数上の述語への写像である．$inv(v)$ はロケーション v の不変的性質である．ロケーション v にあるとき，$s \in \langle\!\langle inv(v) \rangle\!\rangle$ である．

4. $init$ は V の中のロケーションから X 上の述語への写像である．$init$ は v の初期条件と呼ばれる．$s \in \langle\!\langle init(v) \rangle\!\rangle \cap \langle\!\langle inv(v) \rangle\!\rangle$ ならば (v, s) は初期状態である．

5. $flow$ は V の中のロケーションから $X \cup \dot{X}$ 上の述語への写像である．ここで，$\dot{X} = \{\dot{x}_1, \ldots, \dot{x}_n\}$ 及び $\dot{x}_i = \frac{dx_i}{dt}$ である．なお，$\frac{dx_i}{dt}$ は x_i の時間 t による微分を意味する．ロケーション v にあるとき，$flow(v)$ を満足する微分可能な関数に従って，変数は変化する．形式的には，以下の条件を満たす微分可能な関数 $\rho : [0, \delta] \to \vec{R}^n$ が存在するときのみに限り，δ 時間ステップ関係 $\xrightarrow{\delta}$ は $(v, s) \xrightarrow{\delta} (v, s')$ として定義される：

(a) $\rho(0) = s$ かつ $\rho(\delta) = s'$

(b) $t \in [0, \delta]$ に対して $\rho(t) \in \langle\!\langle inv(v) \rangle\!\rangle$ である．

(c) $t \in [0, \delta]$ に対して $flow(v)[X, \dot{X} := \rho(t), \dot{\rho}(t)]$ が $true$ である．ここで，$\dot{\rho}(t) = (\dot{\rho}_1(t), \ldots, \dot{\rho}_n(t))$ である．

6. $E \subseteq V \times V$ は状態遷移関係である．

7. upd は E から X のベキ集合への写像である．状態遷移 e が起こるとき，$upd(e)$ の中の変数はその値を変えることができる．

8. $jump$ は E からジャンプ条件への写像である．状態遷移 e のジャンプ条件 $jump(e)$ は $X \cup upd'(e)$ 上の述語である．ここで，$upd(e) = \{y_1, \ldots, y_k\}$ に対して，$upd'(e) = \{y_1', \ldots, y_k'\}$ である．y_i'

は状態遷移後の y_i の値である．以下の条件を満たすときのみに限り，遷移ステップ関係 $\xrightarrow{e}$ は $(v,s) \xrightarrow{e} (v\prime, s\prime)$ として定義される：

(a) $e = (v, v\prime)$

(b) $s \in \langle\!\langle inv(v) \rangle\!\rangle$ かつ $s\prime \in \langle\!\langle inv(v\prime) \rangle\!\rangle$

(c) $jump(e)[X, upd\prime(e) = s, s\prime[upd(e)]]$ は *true* である．ここで，$s\prime[upd(e)]$ は $s\prime$ を $upd(e)$ の中の変数に制限することを意味する．

(d) $x_i \in X \setminus upd(e)$ に対して $s_i = s_i\prime$ である．

9. L は並列合成を定義するために使われる同期ラベルである．

10. $sync$ は E から $L \cup \{\tau_A\}$ への写像である．$sync(e)$ は状態遷移 e の同期ラベルである．ラベル τ_A は並列合成のときに同期しない内部遷移である．

■

[$Matrix_x$ からハイブリッドオートマトンへの変換]

次に，$Matrix_x$ からハイブリッドオートマトンへの変換方法を示す．

本章では，$Matrix_x$ で表現する仕様の部分はアナログ動作が支配的であると考えて，1つの状態を持つハイブリッドオートマトンを構成する．また，このハイブリッドオートマトンは他のハイブリッドオートマトンと同期して，値が設定される可能性があると考える．しかし，$Matrix_x$ はアナログ動作しか記述できないので，他のハイブリッドオートマトンと同期して値が設定されるといったデジタル動作や $Matrix_x$ で制御する制御量の初期値は自然言語で記述する．

$Matrix_x$ からハイブリッドオートマトンへの変換は以下のとおりである：

1. まず, Definition 1 の方法により, $Matrix_x$ のブロック線図より, それが意味する微分方程式を構成する.

2. 次に, 微分方程式を基礎として, ハイブリッドオートマトンを構成する. ハイブリッドオートマトンは, $\mathrm{A} = (X, V, inv, init, flow, E, upd, jump, L, sync)$ の 10 つ組で定義される.

(a) $flow$ は (1) で構成した微分方程式である.

(b) X は微分方程式に存在する変数の集合とする.

(c) V は制御法則 $flow$ により支配される一つのロケーション $v \in V$ をとる.

(d) inv はロケーションから微分方程式への写像とする.

(e) $init$ はロケーションから x 上の述語への写像であり, x 上の述語はこのオートマトンの初期条件である. $init$ は自然言語で記述された初期条件を使う.

(f) $E \subseteq V \times V$ は状態遷移関係であり, 以下のように定義される :

i. 他のハイブリッドオートマトンと同期するとき

$$(v, v) \in E$$

ii. 他のハイブリッドオートマトンと同期しないとき

$$E = \emptyset$$

(g) upd は E から X のべき集合への写像であり, 状態遷移 $e \in E$ が起こるときに $upd(e)$ の中の変数を更新する. 他のハイブリッドオートマトンと同期して値が設定されるときは $(v, v) \in E$ が存在して, $upd(e)$ に値の設定を割り当てる.

(h) $jump$ は E からジャンプ条件への写像である. 他のハイブリッドオートマトンと同期して値が設定される条件をジャンプ条件とする.

(i) Lは同期ラベルであり, このハイブリッドオートマトンの状態遷移と同期する, 他のハイブリッドオートマトンの状態遷移のラベルを指定する.

(j) $sync$はEから$L \cup \{\tau_A\}$への写像であり, 上記の同期ラベルL及び内部のラベル$\{\tau_A\}$をEに割り付ける.

次に, $Matrix_x$からハイブリッドオートマトンへの変換方法の正当性を示す.

Theorem 3 (ハイブリッドオートマトンへの変換の正当性)

$Matrix_x$により, アナログ動作を仕様記述して, かつ, 自然言語により, それが他のハイブリッドオートマトンと同期して値が設定されるといったデジタル動作や制御量の初期値を仕様記述した, ハイブリッドシステムが上記の変換により得られたハイブリッドオートマトン$\mathrm{A} = (X, V, inv, init, flow, E, upd, jump, L, sync)$と等しい.

Proof 3 まず, 変換においては, ハイブリッドオートマトンAのロケーションVは1個の要素を持つ集合とする.

次に, $init$は自然言語で定義された初期条件と等しい述語で定義する.

次に, X, inv, $flow$は$Matrix_x$と等しい微分方程式 *[41]* から定義する.

最後に, E, upd, $jump$, L, $sync$は, 他のハイブリッドオートマトンと同期してある条件下で値が設定されるといった自然言語によるデジタル動作の記述と等しく定義する.

以上より, $Matrix_x$と自然言語記述した仕様から変換されたハイブリッドオートマトンは仕様と等しい.

■

最後に, $Matrix_x$ からハイブリッドオートマトンへの変換例を示す.

Example 4 ($Matrix_x$ からハイブリッドオートマトンへの変換)

図１６（１）のブロック線図を例として, $Matrix_x$ からハイブリッドオートマトンへの変換例を示す. また, デジタル動作の自然言語仕様としては, 以下のとおりである：

「$Matrix_x$ は他のハイブリッドオートマトンと同期ラベル set_f で同期して無条件に f が値 sp に設定される. また, 初期条件は $f = sp$ である.」

1. まず, ブロック線図から微分方程式を構成する. 図１６（１）のすべての線に名前を付けて（２）を作成する. 図１６（２）に対して, 以下のように代数方程式を作る：

   $a = \frac{1}{T}h$, $b = a - d$, $c = \oint b dt$, $c = f$, $e = f$, $d = \frac{1}{T}e$.

   以上の代数方程式を解くと, 図１６（３）の微分方程式 $\dot{f} = \frac{1}{T}(h - f)$ が得られる.

2. 次に, 微分方程式からハイブリッドオートマトン $\mathbf{A} = (X, V, inv, init, flow, E, upd, jump, L, sync)$ を構成する.

   (a) $flow$ は (1) で構成した微分方程式 $\dot{f} = \frac{1}{T}(h - f)$ である.

   (b) X は微分方程式に存在する変数の集合 $X = \{f, h\}$ とする.

   (c) V は制御法則 $flow$ により支配される一つのロケーション $idle \in V$ をとる.

   (d) inv はロケーションから微分方程式への写像であり, $inv(idle) = \dot{f} = \frac{1}{T}(h - f)$ とする.

   (e) $init$ はロケーションから x 上の述語への写像であり, x 上の述語はこのオートマトンの初期条件であり, $init(idle) = f = sp$ である.

   (f) $E \subseteq V \times V$ は状態遷移関係であり, 他のハイブリッドオートマトンと同期するので, $(idle, idle) \in$

Eである.

(g) $(idle, idle) \in E$が存在して, $upd(idle)$に値設定$f := sp$を割り当てる.

(h) $jump$はEからジャンプ条件への写像であり, この場合は無条件なので, $jump(e) = true$である.

(i) Lは同期ラベルであり, このハイブリッドオートマトンの状態遷移と同期する, 他のハイブリッドオートマトンとの同期ラベルはset_fなので, $L = \{set_f\}$となる.

(j) $sync$はEから$L \cup \{\tau_A\}$への写像であり, 同期ラベル$L = \{set_f\}$及び内部のラベル$\emptyset$をEに割り付ける.

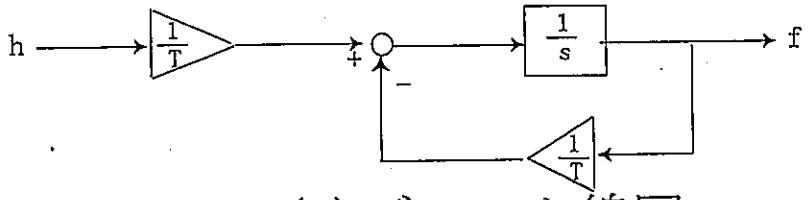
以上より, 図16（4）のハイブリッドオートマトンが構成できた.

■

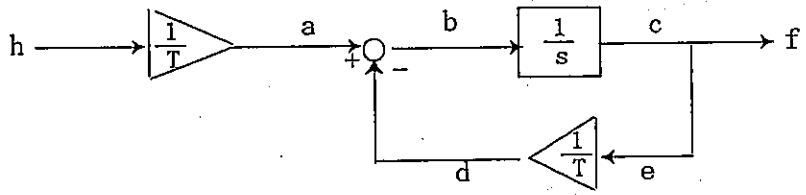
5.2.2 ハイブリッドオートマトンから線形ハイブリッドオートマトンへの近似手法

ハイブリッドオートマトンから線形ハイブリッドオートマトンへの近似手法では, モデル検査アルゴリズム [42] を実現するために, 微分方程式などを線形近似する. ここで, 線形ハイブリッドオートマトンとは, 微分方程式の右辺が定数である場合を指す. この近似手法は, 文献 [43, 44] の手法を適用する. この近似手法は, 基本的には, 以下の二つのステップからなる:

1. まず, ハイブリッドオートマトンのロケーションに関しては, ロケーションの非線形な不変的性質を分割して, 複数の新しいロケーションの線形な不変的性質に分割されて, 線形ハイブリッドオートマトンのロケーションとなる.



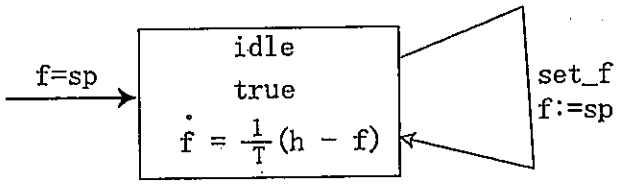
(1) ブロック線図



(2) 新たなブロック線図

$$\dot{f} = \frac{1}{T}(h - f)$$

(3) 微分方程式



(4) ハイブリッドオートマトン

図 16: $Matrix_x$ のブロック線図とハイブリッドオートマトン

2. 次に, ハイブリッドオートマトンの微分方程式は線形微分方程式により近似されて, 線形ハイブリッドオートマトンの微分方程式となる. さらに, ハイブリッドオートマトンの初期条件やジャンプ条件などは弱められて（条件が弱められるとは, 条件を表現する表明式の制約が緩くなり, より多くの動作が可能となることを意味する）, 線形ハイブリッドオートマトンの初期条件やジャンプ条件などとなる.

以下に, 文献 [43, 44] に従って, 近似手法を定義する.

まず, ハイブリッドオートマトンの両立性の概念を定義する.

Definition 22 (ハイブリッドオートマトンの両立性)

ハイブリッドオートマトン $\mathrm{A} = (X_A, V_A, inv_A, init_A, flow_A, E_A, upd_A, jump_A, L_A, sync_A)$ とハイブリッドオートマトン $\mathrm{B} = (X_B, V_B, inv_B, init_B, flow_B, E_B, upd_B, jump_B, L_B, sync_B)$ は, 以下の条件を満たすときのみに限り, A と B は両立するという：

1. 変数, ロケーション, 状態遷移と同期ラベルの集合が同じである. つまり, $X_A = X_B$, $V_A = V_B$, $E_A = E_B$, $L_A = L_B$ である.
2. すべての状態遷移 e に対して, $sync_A(e) = sync_B(e)$ である.

■

次に, ハイブリッドオートマトンの近似の概念を定義する.

Definition 23 (ハイブリッドオートマトンの近似)

両立なハイブリッドオートマトン A と B に対して, 以下の条件を満たすときのみに限り, B は A の

近似であるという. なお, B は A の近似であることを $A \preceq B$ と記述する：

1. すべてのロケーション v に対して, $《inv_A(v)》 \subseteq 《inv_B(v)》$ である.

2. すべてのロケーション v に対して, $《init_A(v)》 \subseteq 《init_B(v)》$ である.

3. すべてのロケーション v と割り付け s に対して, $fl_A(v)(s) \subseteq fl_B(v)(s)$ である.

4. すべての状態遷移 e に対して, $jump_A(e)$ ならば $jump_B(e)$ である.

5. すべての状態遷移 e に対して, $upd_A(e) = upd_B(e)$ である.

ここで, $fl_A(v)(s) = \{d | s \in 《inv_A(v)》$ かつ $flow_A(v)[X_A, \dot{X}_A = s, d]$ は $true$ である $\}$ であり,

$fl_B(v)(s) = \{d | s \in 《inv_B(v)》$ かつ $flow_B(v)[X_B, \dot{X}_B = s, d]$ は $true$ である $\}$ である.

■

次に, ハイブリッドオートマトンの近似の正当性を示す.

Theorem 4 (ハイブリッドオートマトンの近似)

両立なハイブリッドオートマトン A と B に対して, $A \preceq B$ ならば $reach(A) \subseteq reach(B)$ である. ただし, $reach(A)$ は A の到達可能な状態集合である.

Proof 4 文献 *[43, 44]* に証明があり, ここでは省略する.

■

次に, ハイブリッドオートマトンから線形ハイブリッドオートマトンへの近似の例を示す.

Example 5 (線形ハイブリッドオートマトンへの近似例)

近似手法により, 以下のように, 図１６（４）のハイブリッドオートマトンから図１８の線形ハイブリッドオートマトンを構成する：

1. まず, ハイブリッドオートマトンのロケーション *idle* と不変的性質 $\dot{f} = \frac{1}{T}(h - f)$ を分割して, 線形な不変的性質を構成する. ここでは, $T = 2$ として, $\dot{f} = \frac{1}{T}(h - f)$ を分割する. 図１７のように, ロケーション *idle* と不変的性質 $\dot{f} = \frac{1}{T}(h - f)$ は分割される.

ロケーション	不変的性質	フロー条件
A	h-f∈(-∞,-10]	$\dot{f}$∈(-∞,-5]
B	h-f∈[-10,-6]	$\dot{f}$∈[-5,-3]
C	h-f∈[-6,0]	$\dot{f}$∈[-3,0]
D	h-f∈[0,6]	$\dot{f}$∈[0,3]
E	h-f∈[6,20]	$\dot{f}$∈[3,10]
F	h-f∈[20,∞)	$\dot{f}$∈[10,∞)

図 17: ロケーションの分割例

１個のロケーション *idle* が $A, \ldots, F$ といった６個のロケーションに分割されて, それらのロケーションは線形な不変的性質を持つ.

2. 分割された６個のロケーションでは, すべて自己ループの状態遷移及びロケーション間の状態遷移を持って, それらの状態遷移には同期ラベル set_f と初期条件 $f := sp$ が付いている.

3. 分割された６個のロケーションは, 本来, 一つのロケーションなので, それらのすべてのロケーション間を自由に状態遷移できる必要がある. ゆえに, すべてのロケーション間に内部遷移 *True*

を作る.

以上より, 図１６（４）のハイブリッドオートマトンから, 図１８の線形ハイブリッドオートマトンが得られる. 図１８の線形ハイブリッドオートマトンは, 図１６（４）のハイブリッドオートマトンの不変的性質 inv, 初期条件 $init$, フロー fl, ジャンプ条件 $jump$ を緩めて近似されたものである. この近似による不正確さの度合いは f と h の具体的な関数による. 例えば, 近似では $f-h$ は-∞ から ∞ の範囲を動くが, f と h の具体的な関数によればそうでないかもしれない. しかし, この近似の不正確さに関わらず, Theorem 2 より, 近似したハイブリッドオートマトンがより多くの到達可能な状態集合を持つので, 到達可能に関する検証結果は正しい.

■

5.2.3 線形ハイブリッドオートマトンのモデル検査手法

線形ハイブリッドオートマトンのモデル検査器として, $HYTECH$ が開発されている [45]. $HYTECH$ では, 線形ハイブリッドオートマトンが時相論理 TCTL で記述した性質を充足するかどうかを自動判定できる. 一般的には, このモデル検査の問題は決定不能であるが, 多くの実用問題で有用であることが知られている. $Matrix_x$ から構成した線形ハイブリッドオートマトンやハイブリッドオートマトンから構成した線形ハイブリッドオートマトンの並列合成を作成して, 時相論理 TCTL により充足すべき安全性の性質を記述して, $HYTECH$ でモデル検査する.

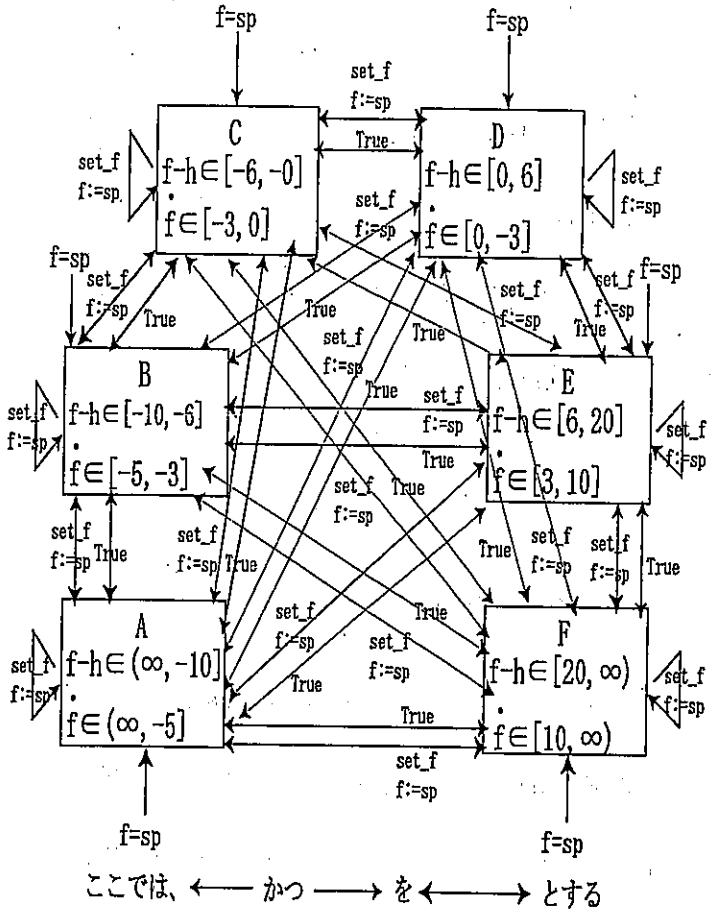


図 18: 線形ハイブリッドオートマトンへの近似例

5.3 ハイブリッドシステムの形式的開発の事例

本節では, 自動車制御システムを事例とする. 自動車制御システムとしては, エンジン制御やブレーキ制御などが存在しているが, ここでは, シャシーレベルを制御する EHC(Electronic Height Control) を事例とする [46].

5.3.1 構造的概観図

まず, EHC の構造的概観図を図１９に示す. EHC は４つのホイールの気圧サスペンションによりシャシーレベルを調節する. compressor がホイールのサスペンションに空気を吹き込むことによってシャシーレベルは増加して, escape valve が空気を抜くことによってシャシーレベルは減少する. センサはシャシーレベルの変化を計測するが, 高周波の外乱はフィルタによって滑らかにされる. 制御論理は, あるサンプリング速度で, シャシーレベルの変化を読み込んで, シャシーレベルを制御する.

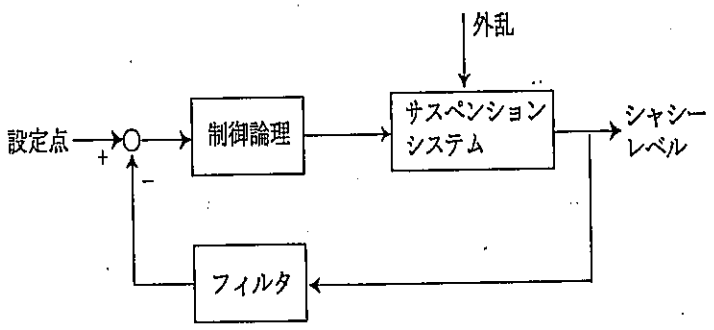


図 19: 構造的概観図

5.3.2 構造的概観図の詳細化

次に, 構造的概観図を段階的に詳細化して開発したものを図２０に示す. その内容は以下のとおりである.

1. 制御論理は制御モードハイブリッドオートマトン, コントローラハイブリッドオートマトンとタイマハイブリッドオートマトンの３つの並列合成に詳細化される.
2. フィルタはフィルタの $Matrix_x$ のブロック線図に詳細化される.
3. サスペンションシステムと外乱はサスペンションシステム＋外乱のハイブリッドオートマトンに詳細化される.

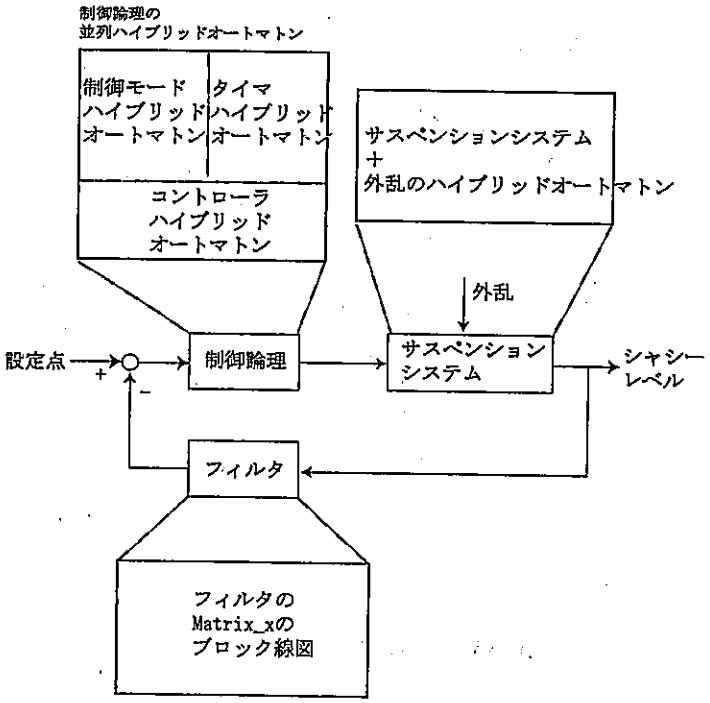


図 20: 構造的概観図の詳細化

5.4 制御論理

制御論理の詳細化は制御モードハイブリッドオートマトン, タイマハイブリッドオートマトンとコントローラハイブリッドオートマトンの３つの並列合成であり, 各ハイブリッドオートマトンは図２１, 図２２及び図２３のとおりである.

まず, 図２１の制御モードハイブリッドオートマトンの特徴は以下のとおりである：

1. $BEND$ は車がカーブ走行中かどうかを意味するブール変数であり, E_ON はエンジンがオンかどうかを意味するブール変数であり, V は車の現在のスピードを意味する変数である.
2. *bend* 又は *not_bend* は車がカーブ走行開始又は終了することを表現するイベントである. つまり, *bend* は $BEND := true$ を意味して, *not_bend* は $BEND := false$ を意味する.
3. *v_zero* 又は *not_v_zero* は V をゼロ又はゼロ以外に設定することを表現するイベントである. つまり, *v_zero* は $V := 0$ を意味して, *not_v_zero* は $V := a$ （ただし, $a \neq 0$）を意味する.
4. *e_on* 又は *not_e_on* はエンジンをオン又はオフすることを表現するイベントである. つまり, *e_on* は $E\_ON := true$ を意味して, *not_e_on* は $E\_ON := false$ を意味する.
5. *cm* は制御モードを意味する変数である.

次に, 図２２のタイマハイブリッドオートマトンの特徴は以下のとおりである：

1. *nc* は同期ラベルである.
2. サンプル時間 *t_sample* が過ぎると, 自己ループしてタイマをゼロにする.
3. タイマをゼロにするときに, 同期ラベル *nc* により他のハイブリッドオートマトンと同期する. つ

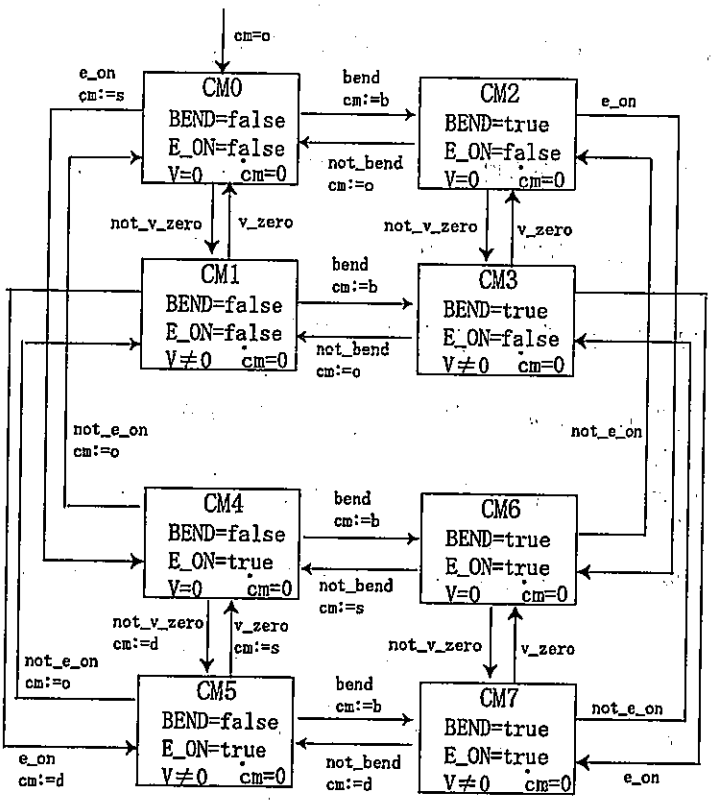


図 21: 制御モードハイブリッドオートマトン

まり, サンプル時間 t_sample 毎に, コントローラハイブリッドオートマトンなどを駆動する.

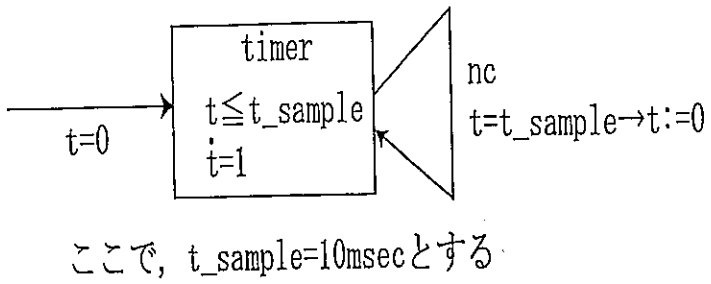
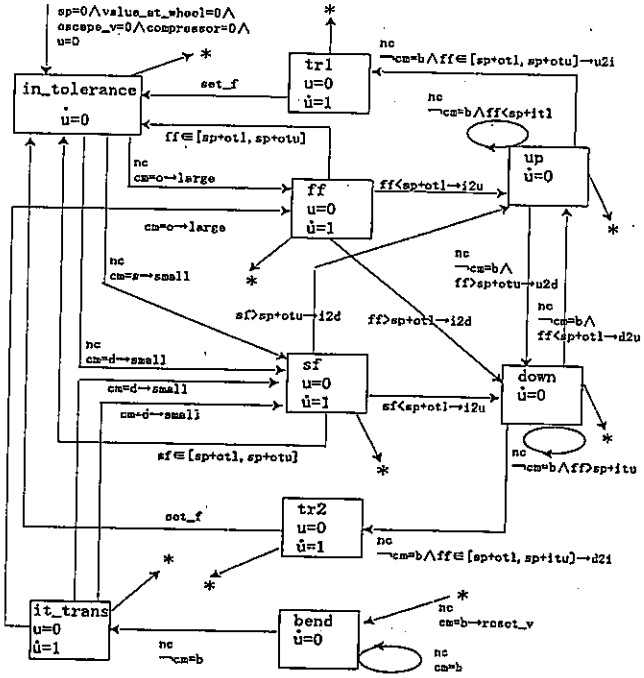


図 22: タイマハイブリッドオートマトン

次に, 図２３のコントローラハイブリッドオートマトンの特徴は以下のとおりである：

1. *in_torelance*, *bend*, *down* 及び *up* の各状態は, 各々, 初期状態, 車がカーブ走行状態, 下り走行状態及び昇り走行状態を意味する. なお, *it_trans* は *in_torelance* と意味は同じであるが, ハイブリッドオートマトンの状態遷移を簡単にするために, *in_torelance* と分離したものである.
2. シャシーレベルが設定点より大幅にずれている時はフィルタの状態 ff に遷移して, そうでないときはフィルタの状態 sf に遷移する.
3. 同期ラベル *nc* により, タイマハイブリッドオートマトンと同期する.
4. 同期ラベル *set_f* により, 後述するフィルタのハイブリッドオートマトンと同期する.
5. シャシーレベルの設定点 sp において, $[sp+otl, sp+otu]$ は外側の許容範囲であり, $[sp+itl, sp+itu]$ は内側の許容範囲である.



ここで

(1)largeは、otl:=-40, otu:=20, itl:=-6, itu:=16 である。
(2)smallは、otl:=-10, otu:=10, itl:=-6, itu:=6 である。
(3)reset_vは、valve_at_wheel:=0, escape_v:=0, compressor:=0 である。
(4)i2uは、valve_at_wheel:=1, compressor:=1 である。
(5)i2dは、valve_at_wheel:=1, escape_v:=1 である。
(6)u2dは、compressor:=0, escape_v:=1 である。
(7)d2uは、compressor:=1, escape_v:=0 である。
(8)u2iは、compressor:=0, valve_at_wheel:=0 である。
(9)d2iは、escape_v:=0, valve_at_wheel:=0 である。

図 23: コントローラハイブリッドオートマトン

5.5 フィルタ

フィルタが詳細化されたフィルタの $Matrix_x$ のブロック線図は以下の図２４である．このブロック線図は，図１６（4）のハイブリッドオートマトンに変換できる．さらに，図１８のように，このハイブリッドオートマトンは線形ハイブリッドオートマトンに変換できる．

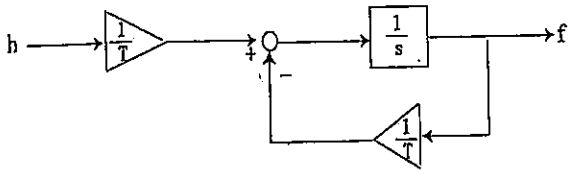


図 24: フィルタの $Matrix_x$ のブロック線図

5.6 サスペンションシステム

サスペンションシステムと外乱が詳細化されたサスペンションシステム＋外乱のハイブリッドオートマトンは車が走行する環境に依存しており，簡単には仕様記述できない．ここでは，車の走行環境はカーブの存在のみを考慮して，最初，ノーマル状態 (normal) であり，カーブ (bend) になるとカーブ中状態 (in_bend) になり，カーブを抜ける (not_bend) とノーマル状態 (normal) になるとする．このような場合の一例として，図２５を考える．

5.6.1 検証

以上により，本事例における EHC は上記の５つの線形ハイブリッドオートマトンが並列動作するハイブリッドシステムとして開発できた．HYTECH による検証では，５つの線形ハイブリッドオー

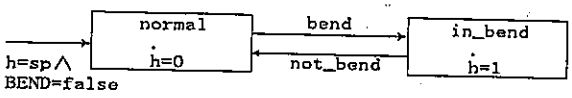


図 25: サスペンションシステム+外乱のハイブリッドオートマトン

トマトンの並列合成が TCTL 記述の検証性質を充足するかどうかを判定する. 代表的な検証性質としては, 以下が考えられる :

1. シャシーレベルは常に $[-47, 27]$ の範囲でなけらばならない. これは, TCTL 式で表現すると, $\forall\Box(-47 \leq sp \leq 27)$ である. なお, $\forall\Box p$ は $\neg\exists\Diamond\neg p$ である.
2. compressor と escape valve は決して同時に動作しない. これは, TCTL式で表現すると, $\neg\forall\Box(compressor = 1 \wedge escape\_v = 1)$ である.
3. 車がカーブ走行状態のときはシャシーレベルは変化しない. これは, TCTL 式で表現すると, $\neg\forall\Box(BEND = true \wedge compressor = 1) \wedge \neg\forall\Box(BEND = true \wedge escape\_v = 1)$ である.

今回, 本来の仕様である非線形ハイブリッドオートマトンから, その不変的性質 *inv*, 初期条件 *init*, フロー *fl*, ジャンプ条件 *jump* を緩めて近似された線形ハイブリッドオートマトンを検証の対象とした. この近似により, 効率的に実装された検証器 HYTECH [45] で検証が行えた.

HYTECH [45] により, 以上の検証性質を検証した. 一般的には, この検証問題は決定不能である [4] が, 上記事例に関しては検証できた. HYTECH により, 多くの現実的な検証問題が検証できたことが報告されている [45].

この事例では検証に要した計算時間とメモリ量は以下のとおりである :

1. $\forall\Box(-47 \leq sp \leq 27)$

   計算時間：62分, メモリ量：128MB

2. $\neg\forall\Box(compressor = 1 \wedge escape\_v = 1)$

   計算時間：58分, メモリ量：128MB

3. $\neg\forall\Box(BEND = true \wedge compressor = 1) \wedge \neg\forall\Box(BEND = true \wedge escape\_v = 1)$

   計算時間：73秒, メモリ量：35MB

なお, 動作マシンはUNIXマシンSun Sparcstattion 20(Main memory 128MB)である.

以上により, 本論文で提案した形式的開発方法論により設計した, ハイブリッドシステムの正当性が検証できることを確認した. これより, 提案した方法論の有効性が示せた.

5.7 むすび

本論文は, 既知のハイブリッドオートマトンなどの計算機科学の技術と既知の$Matrix_x$などの制御の技術, つまり異分野の技術, を統合化することによって, ハイブリッドシステムの体系的な開発手法を構築して, 実験的に有効性を示した. このような試みは, 既存研究にはまったく存在せず, 新規性がある. 具体的には, アナログ動作が支配的なハイブリッドシステムを対象として, 以下のような段階的に詳細化する形式的開発方法論を提案した：

1. 抽象度の高い仕様では, $Matrix_x$を中心的に使いながら, ハイブリッドシステムの構造を仕様記述する.

2. 一方, 抽象度の低い仕様では, ハイブリッドオートマトンを中心的に使いながら, ハイブリッドシ

ステムの動作を仕様記述する.

最後に, 開発したハイブリッドシステムがHYTECH [45] により, 形式的に検証できることを確認した.

本研究は, 計算機科学と制御理論の統合化アプローチから, ハイブリッドシステムを形式的に開発する方法論の研究である. 従来のハイブリッドシステムの開発方法論では, 制御理論のアプローチから, $Matrix_x$ [37] や $MATLAB$ [38] などのCADツールにより, ブロック線図を設計して, その動作をシミュレーションにより確認するものであった. しかし, この方法論では, 設計の正当性の確認が困難であったり, デジタルな動作がうまく表現できなかった. 本研究では, アナログ動作とデジタル動作の両方を合理的に表現して, 形式的に設計と正当性検証ができた.

今後の研究課題しては以下が考えられる：

1. 提案手法を多数の事例に適用して, その有効性を確認すること
2. ハイブリッドシステムのモジュール記述及びモジュール検証を実現する理論を開発すること
3. 段階的に詳細化しながら開発した仕様間の詳細化の正当性を検証する理論を開発すること

6 確率線形ハイブリッドオートマトンの記号的到達可能解析手法

6.1 まえがき

昨今，コンピュータの遍在 (*ubiquitous*) 化が進み，あらゆるものにシステムが内蔵される時代となった．その中で，組込みシステムの重要性が増し，システムに対する信頼性・安全性の保証 (*dependable computing*) が要求されている．

組込みシステムは，自動車や航空機などの制御コントローラとしても，広く用いられており，このような制御システムが，万一，エラーやバグによる障害などを引き起こしたならば，その人的及び金銭的な被害・損失は莫大なものとなってしまう．ゆえに，上の *safety-critical* システムに対しては，意図しない動作が起こらないなどの，安全性が保証されていることが，きわめて重要となる．

システムの *dependability* を保証する，つまり検証を行なうには，かねてより，テストやシミュレーションなどの方法が用いられてきたが，限られたテストパターンに対してのみチェックが行なわれ，経験や知識に無いような事例に関しての，検査もれがあることは否めない．

ゆえに，本論文では，形式的・数理的技法 (*formal method*) に基づく検証手法を提案する．

形式的技法では，システムを数学的なモデルを用いて表現し，そのモデルに対する解析により，システムが要求を満たすか否かの数学的証明を行なう．計算モデルとしては，総じて，状態遷移グラフが用いられる．検証は，そのグラフの探索問題 (モデル検査 [52]) に帰着される．

状態遷移グラフのとり得る全ての状態，つまり，システム上で起こり得る全ての動作を，網羅的（しらみつぶし）に全数探索し，要求に反する状態が存在しないことを示すことで，システムが必ず要求を満たすこと，*dependability* を保証する．

形式的検証に関する，従来の代表的な研究としては，以下が挙げられる．

1. 1996 年，Alur らは，線形ハイブリッドオートマトンの記号モデル検査手法を考案し，検証器 HyTech に実装した [26].
2. 2001 年，Sproston は，確率 *rectangular* オートマトンに対する種々の解析手法を考案した [48].
3. 2003 年，Kwiatkowska らは，確率時間オートマトンの記号モデル検査手法を考案し [47]，検証器 Prism に実装した．

本章では，従来研究の流れを汲みながらも，新たに，システム記述言語としての，確率線形ハイブリッドオートマトン，及び，その記号的な到達可能性解析手法を提案する．

確率線形ハイブリッドオートマトンによって，システムの不確かな動作や，統計的に評価せざるを得ないような事象（ネットワークにおける信号の遅延，雑音の影響など）をも確率的に定量化し，それらを考慮した柔軟な検証が可能となる．

6.2 確率線形ハイブリッドオートマトン

本章では，システム記述言語として，確率線形ハイブリッドオートマトンを定義する．

確率線形ハイブリッドオートマトンとは，簡単に言うと，実変数と微分方程式を備えたオートマトンであるハイブリッドオートマトン[26]に，線形制約を課し，オートマトンの各エッジに確率を付加することで拡張したものである．事例は，P.157 に示す．

6.2.1 線形の概念

$\vec{x}$ を実変数のベクトルとする．$\vec{x}$ 上の線形項とは，変数 $x_i \in \vec{x}$ の線形結合 $\sum_i c_i \cdot x_i + c$ である．ただし，$c_i, c \in \mathbb{Z}$（整数）とする．

$\vec{x}$ 上の線形不等式とは，$\vec{x}$ 上の線形項間の不等式である．

$\vec{x}$ 上の線形式は，線形不等式の有限なブール結合で表現される．特に，論理積の場合，凸状の線形式（*convex linear formula*）と言い，凸多面体（*convex polyhedron*）を表わす．ここで，凸状線形式の集合を `clf` と定義する．また，全ての線形式は，凸状線形式の論理和（選言標準形）に変換できる．

次に，ハイブリッドオートマトンに対する線形制約を以下のように定める．

1. ハイブリッドオートマトンが持つ変数の第1次導関数は，凸状線形式に従う．すなわち，定数 $(\dot{x}_i = 2)$ か，もしくは，定数による区間 $(1 \leq \dot{x}_i \leq 3)$ の形をとる．後者は，システムの動作に幅を持たせたることを許し，また，非線形な動作を近似する際にも有用である．
2. ハイブリッドオートマトンに現れる条件式は，凸状線形式で与えられる．

[凸多面体]

集合 S を n 次元実数空間の部分集合 $S \subseteq \mathbb{R}^n$ とする．任意の，2点 $\vec{s}, \vec{s}' \in S$ と，実数 $a \in [0,1]$ に対して，$a \cdot \vec{s} + (1-a) \cdot \vec{s}' \in S$ が成り立つ場合，S は凸多面体であるという．

6.2.2 確率分布 p, μ

有限集合 Q 上の離散確率分布 p とは，要素 $q \in Q$ に，確率 $0 \leq p(q) \leq 1$ を割り当てる関数，$p : Q \to [0,1]$ である．なお，$\sum_{q \in Q} p(q) = 1$．集合 Q 上の分布の集合を $\mathtt{Dist}(Q)$ と表記する．

また，Q の部分集合 $\mathtt{support}(p) = \{q \mid p(q) > 0\}$ と定義する．

6.3 構文

確率線形ハイブリッドオートマトン (*probabilistic linear hybrid automaton*) は，

$\mathtt{PLHA} = \langle \vec{x}, L, inv, dif, prob, (grd_l)_{l \in L}, E \rangle$ の7組から定義される．

6.3.1 データ変数 $\vec{x}$

データ変数 $\vec{x}$ とは，時間に従って変化する実変数のベクトル $\vec{x} = \{x_1, x_2, \ldots, x_n\}$ である．また，$\vec{x}$ の要素数 n は，オートマトン PLHA の次元と呼ばれる．

変数の評価値とは，実数値 $\vec{s} = \{s_1, \ldots, s_n\} \in \mathbb{R}^n$ (n 次元実数空間内の1点) を指す．

先に定義した (凸状) 線形式 f は，(凸) 多面体 $[\![f]\!] \subseteq \mathbb{R}^n$ を表わす．すなわち，式 f を満たす評価値 $\vec{s}$ の集合であり，$\vec{s} \in [\![f]\!]$ iff $f[\vec{x} := \vec{s}] = \mathsf{T}$，もしくは，$[\![f]\!] = \{\vec{s} \mid f[\vec{x} := \vec{s}] = \mathsf{T}\}$ と書ける．このように，集合を線形式によって定義することで，無限個の要素から成る集合に対して，

1つの記号的表現を与えることができる.

各データ変数 x_i に対して，その第1次導関数を $\dot{x}_i$ と表記する.

6.3.2 制御ロケーション L

L は制御ロケーションと呼ばれるノードの有限集合である.

オートマトンPLHAの状態 $(l, \vec{s})$ は，制御ロケーション $l \in L$ と，評価値 $\vec{s} \in \mathbb{R}^n$ の組で構成される. リージョン (*region*) $R = \bigcup_{l \in L}(l, S_l)$ とは状態の集合を表わし，多面体 (凸多面体の和集合) $S_l \subseteq \mathbb{R}^n$ をロケーションに関して集めたものである. また，リージョンは線形式によって記号的に表現できる. リージョンを示す記号表現 $\pi = \bigcup_{l \in L}(l, f_l)$ は述語 (*predicate*) と呼ばれ，$[\![\pi]\!] = \bigcup_{l \in L}(l, [\![f_l]\!])$ を意味する.

6.3.3 ロケーション不変条件 *inv*

関数 $inv : L \to \texttt{clf}$ は，各制御ロケーション $l \in L$ に，凸状線形式 $inv(l) \in \texttt{clf}$ を割り付ける. 凸状線形式 $inv(l)$ はロケーション l の不変条件 (*invariant*) であり，オートマトンのロケーション遷移を確実に行なうために用いられる. なぜならば，オートマトンPLHAのコントロールは，変数の評価値 $\vec{s}$ が不変条件 $inv(l)$ を満たす間 ($\vec{s} \in [\![inv(l)]\!]$) に限って，ロケーション l にとどまることが許されるためである. すなわち，不変条件が偽になる前にロケーション遷移が起こらなければならない.

6.3.4 フロー *dif*

関数 $dif : L \to \mathtt{clf}$ は，各制御ロケーション $l \in L$ に，$\vec{x}$ 上の凸状線形式 $dif(l) \in \mathtt{clf}$ を割り当てる．凸状線形式 $dif(l)$ はロケーション l のフロー (*flow*) と呼ばれ，データ変数 $\vec{x}$ の変化を規定する．オートマトンPLHAのコントロールがロケーション l にある間，変数の第1次導関数 $\dot{\vec{x}}$ はフロー $dif(l)$ に従う．要するに，$\dot{\vec{s}} \in [\![dif(l)]\!]$ (*differential inclusion*) である．

6.3.5 確率分布 *prob*

関数 $prob : L \to 2^{\mathtt{Dist}(2^{\vec{x}} \times \mathtt{clf} \times L)}_{fn}$ は，各制御ロケーション $l \in L$ に，$\vec{x}$ のべき集合, clf, 及び L 上の離散確率分布の，空でない有限集合 $prob(l) \subseteq \mathtt{Dist}(2^{\vec{x}} \times \mathtt{clf} \times L)$ を割り当てる．つまり，ロケーション l には，確率分布の集合 $prob(l) = \{p_l^1, \ldots, p_l^{|prob(l)|}\}$ が割り当てられる．

また，前述の定義より，確率分布 $p_l^i \in prob(l)$ は，関数 $p_l^i : 2^{\vec{x}} \times \mathtt{clf} \times L \to [0,1]$ である．

6.3.6 事前条件 $(grd_l)_{l \in L}$

$(grd_l)_{l \in L}$ は関数族であり，各制御ロケーション $l \in L$ に対して，関数 $grd_l : prob(l) \to \mathtt{clf}$ は，ロケーション l における各離散確率分布 $p_l^i \in prob(l)$ に，凸状線形式 $grd_l(p_l^i) \in \mathtt{clf}$ を割り付ける．凸状線形式 $grd_l(p_l^i)$ は確率分布 p_l^i の事前条件 (*guard*) であり，確率分布 p_l^i が実行可能か否かを決める．変数の評価値 $\vec{s}$ が事前条件 $grd_l(p_l^i)$ を満足するとき ($\vec{s} \in [\![grd_l(p_l^i)]\!]$)，p_l^i は実行可能であり，オートマトンPLHAのロケーション l において，確率分布 p_l^i による確率的な遷移が起こり得る．

6.3.7 確率的遷移関係 E

上記の $prob$, $(grd_l)_{l\in L}$ により，オートマトン PLHA の確率的な遷移関係（エッジ）が定められる．現在のロケーションを $l \in L$，確率分布を $p_l^i = p \in prob(l)$，事前条件を $grd_l(p) = g \in \mathtt{clf}$ とすると，エッジ $e \in E$ は，$e = (l, g, p, X, updt, l')$ となる．ただし，$X \subseteq \vec{x}$, $updt \in \mathtt{clf}$, $l' \in L$, $p(X, updt, l') > 0$ である．また，全ての確率的遷移関係 e の集合を E と表記する．

X は遷移の際に更新されるデータ変数である．遷移後の更新された新しい変数を，$\vec{x}'$ のように，プライム記号 $'$ をつけて書くものとする．

$updt$ は，更新式 (*update*) と呼ばれる $\vec{x} \uplus X'$ 上の凸状線形式であり，現在の評価値 $\vec{s}$ に対して，更新後の評価値 $\vec{s}'$ がとり得る範囲を示す．

また，$act = updt \wedge \bigwedge_{x_i \in \vec{x}\setminus X}(x_i' = x_i)$，及び $\vec{s}' \in [\![act(\vec{s})]\!]$ *iff* $act[\vec{x}, \vec{x}' := \vec{s}, \vec{s}'] = \top$ と定義する．つまり，アクション (*action*) と呼ばれる凸状線形式 $act \in \mathtt{clf}$ によって，各評価値 $\vec{s} \in \mathbb{R}^n$ に，凸多面体 $[\![act(\vec{s})]\!] \subseteq \mathbb{R}^n$ を割り当てる関数 $[\![act()]\!] : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ が特徴づけられる．

l' は遷移先ロケーションを表わす．

結局，エッジ e の意味するところは次のとおりである．

- 変数の評価値 $\vec{s}$ が事前条件 g を満たす時，オートマトン PLHA のコントロールは，遷移元ロケーション l から，確率 $p(X, updt, l')$ で，遷移先ロケーション l' へ遷移し，かつ同時に，$x_i \in X$ が，$updt$ の示す範囲内で非決定的に，いずれかの新しい評価値 $s_i' \in [\![act(\vec{s})]\!]_i$ に更新される．$x_i \in \vec{x}\setminus X$ は更新されず，$s_i' = s_i$ となる．

- これは，2状態 $(l,\vec{s}),(l',\vec{s}')$ 間の離散的な (時間の経過を伴なわない) 遷移を表わしている．

$$(l,\vec{s}) \xrightarrow{e} (l',\vec{s}') \quad iff \quad e=(l,g,p,X,updt,l'),\ \vec{s}\in[\![g]\!] \ and \ \vec{s}'\in[\![act(\vec{s})]\!] .$$

[例]

次元 $n=4$，$X=\{x_1,x_2,x_3\}$，$updt=3\le x_1'\le 5\ \wedge\ x_2'=x_1+1$ の場合．

1. 遷移後の x_1' は，区間 $[3,5]$ 内の実数であり，
2. 更新された x_2' は，遷移前の x_1 の値より1大きく，
3. 新しい x_3' は，あらゆる任意の実数 となり得る．なぜなら，x_3' に関しては無条件 であるため．
4. x_4 は更新されない．$(x_4\notin X,\ x_4'=x_4)$

6.4 意味論

本節では，確率線形ハイブリッドオートマトン PLHA の意味を，状態遷移システムとして定義する．確率線形ハイブリッドオートマトンの意味とは，その動作的側面を指し，並行確率システム [48] に相当する．

6.4.1 並行確率システム

並行確率システム (*concurrent probabilistic system*) は，マルコフ決定過程に基礎を成しており，非決定性および確率，両方の側面を持つ．形式的には，以下のように定義される．

並行確率システム $\mathtt{CPS} = \langle Q, \Sigma, \mathit{Steps} \rangle$

- Q は状態集合.
- Σ はイベントの集合.
- 関数 $\mathit{Steps} : Q \to 2^{\Sigma \times \mathtt{Dist}(Q)}$ は,

  各状態 $q \in Q$ に，イベント $\sigma \in \Sigma$ と，Q 上の離散確率分布 $\mu \in \mathtt{Dist}(Q)$ から成る，

  組 (σ, μ) の非空な集合 $\mathit{Steps}(q) \subseteq \Sigma \times \mathtt{Dist}(Q)$ を割り当てる.

並行確率システム CPS における，状態 q からの確率的遷移を $q \xrightarrow{\sigma,\mu} q'$ と書く．これは，

1. イベントと確率分布の組 $(\sigma, \mu) \in \mathit{Steps}(q)$ の 非決定的選択$(\star)$ および，それに続く，
2. 確率 $\mu(q') > 0$ である次状態 q' の，確率的選択 によって決定される.

並行確率システムの実行過程 (ふるまい) を意味するパス (*path*) は，非決定的選択および確率的選択，両方を 解決することで得られ，上記の状態遷移の系列として以下のように表わされる.

$$\omega = q_0 \xrightarrow{\sigma_0,\mu_0} q_1 \xrightarrow{\sigma_1,\mu_1} q_2 \xrightarrow{\sigma_2,\mu_2} \cdots .$$

パス ω について，最初の状態を $\mathit{first}(\omega)$，(今のところ) 最後の状態を $\mathit{last}(\omega)$，長さを $|\omega|$，k 番目の状態を $\omega(k)$，k 番目の遷移を $\mathit{step}(\omega, k)$，0〜k 番目までの *prefix* (長さ k) を $\omega^{(k)} = q_0 \xrightarrow{\sigma_0,\mu_0} q_1 \xrightarrow{\sigma_1,\mu_1} \cdots \xrightarrow{\sigma_{k-1},\mu_{k-1}} q_k$ と書く.

また，並行確率システムCPSにおける全ての，有限長のパスの集合を $Path_{fin}$，$first(\omega) = q$ であるパスの集合を $Path_{fin}(q)$ と表記する．

[Adversary]

ここでは，前述の非決定的選択を解決する手段として，関数 *Adversary* [?] を導入する．*Adversary* は，スケジューラ，もしくは，ポリシーとも呼ばれ，直観的には，非決定性・確率双方の特性を示すシステムを，単に確率的な性質のみを持つシステムとして扱うための概念であると言える．本論文では，種々さまざまある *Adversary* の中から，決定性 *Adversary* を採用する．

並行確率システム $\mathtt{CPS} = \langle Q, \Sigma, Steps \rangle$ の決定性 *Adversary* $\mathcal{A}$

- *Adversary* は，関数 $\mathcal{A} : Path_{fin} \rightarrow \Sigma \times \mathtt{Dist}(Q)$ であり，各有限長のパス $\omega \in Path_{fin}$ に，$\mathcal{A}(\omega) \in Steps(last(\omega))$ であるイベントと確率分布の組 $\mathcal{A}(\omega) = (\sigma, \mu)$ を写像する．

先に定義した関数 $Steps : Q \rightarrow 2^{\Sigma \times \mathtt{Dist}(Q)}$ と比較すると，現在の状態に対して，*Steps* が組 (σ, μ) の 集合（非決定性が残っている） を割り当てる一方で，$\mathcal{A}$ は組 (σ, μ) を 一意的（決定的） に写像することが解かる．つまり，*Adversary* $\mathcal{A}$ によって，並行確率システムCPSの非決定性 $(\star)$ が除去される．また一般に，1つの並行確率システムCPSに対して，複数の *Adversary* が存在し得る．

並行確率システムの *Adversary* $\mathcal{A}$ について，$Path_{fin}^{\mathcal{A}}$ を定義する．

$Path_{fin}^{\mathcal{A}}$ は，任意の $0 \leq i < |\omega|$ に対して，$step(\omega, i) = \mathcal{A}(\omega^{(i)})$ である有限長のパスの集合とする．

[マルコフ連鎖]

決定性 *Adversary* $\mathcal{A}$ によって，それに対応するマルコフ連鎖が定められる．

マルコフ連鎖とは，記憶の無い確率過程であり，次状態（への遷移確率）を割り当てる確率分布が，現在の状態にのみ依存して決まる．

並行確率システム CPS の決定性 *Adversary* $\mathcal{A}$ に対応する，マルコフ連鎖 $\mathrm{MC}^{\mathcal{A}}$ を以下に示す．

マルコフ連鎖 $\mathrm{MC}^{\mathcal{A}} = \langle Path^{\mathcal{A}}_{fin}, \mathbf{P}^{\mathcal{A}} \rangle$

- $\mathrm{MC}^{\mathcal{A}}$ における状態は，*Adversary* $\mathcal{A}$ が作る有限長のパス $\omega \in Path^{\mathcal{A}}_{fin}$ に相当し，
- 推移確率行列は，関数 $\mathbf{P}^{\mathcal{A}} : Path^{\mathcal{A}}_{fin} \times Path^{\mathcal{A}}_{fin} \to [0,1]$ であり，

  現在のパス $\omega \in Path^{\mathcal{A}}_{fin}$ から，CPS の状態が 1 つ進んだ（$|\omega'| = |\omega| + 1$ である）パス $\omega' \in Path^{\mathcal{A}}_{fin}$ への遷移の生起確率を与える．

$$\mathbf{P}^{\mathcal{A}}(\omega, \omega') = \begin{cases} \mu(q) & \text{if } \mathcal{A}(\omega) = (\sigma, \mu) \text{ and } \omega' = \omega \xrightarrow{\sigma,\mu} q \\ 0 & \text{otherwise.} \end{cases}$$

前述の定義より，$\mathcal{A}(\omega) \in Steps(last(\omega))$ である．つまり，CPS の現在の状態 $last(\omega)$ だけで，次状態 q への遷移確率が決定されており，マルコフ連鎖であることが確かめられる．

[パス上の確率 $\mathcal{P}^{\mathcal{A}}$]

パス上の確率は，写像 $\mathcal{P}^{\mathcal{A}} : Path^{\mathcal{A}}_{fin} \to [0,1]$ として，以下のように，パスの長さについて帰納的に定義される．

1. もし $|\omega| = 0$ ならば，$\mathcal{P}^{\mathcal{A}}(\omega) = 1$,

2. $|\omega| > 0$ の場合，*if* $\omega' = \omega \xrightarrow{\sigma,\mu} q$ *for some* $\omega \in Path^{\mathcal{A}}_{fin}$, *then*

$$\mathcal{P}^{\mathcal{A}}(\omega') = \mathcal{P}^{\mathcal{A}}(\omega) \cdot \mathrm{P}^{\mathcal{A}}(\omega, \omega') \ .$$

以上より，並行確率システム CPS における，パス $\omega \in Path^{\mathcal{A}}_{fin}$ が実行される確率は，マルコフ連鎖 $\mathrm{MC}^{\mathcal{A}}$ に帰着して求められる，と結論づけることができる．

ゆえに，CPS の確率的なふるまいを，定量的に評価することが可能となった．

6.4.2 確率線形ハイブリッドオートマトンの意味

確率線形ハイブリッドオートマトン PLHA の意味は，先に定義した並行確率システムとして捉えることができる．また，その動作は，時間経過および離散（即時）的な変化から成る 2 種類の状態遷移によって表現される．

まず，記法の説明を行なう．現在の状態を $(l, \vec{s})$ とする．

確率分布 $p \in prob(l)$ の $\mathtt{support}(p) = \{(X^1, updt^1, l'^1), (X^2, updt^2, l'^2), \ldots, (X^m, updt^m, l'^m)\}$ に対応する，アクション（更新式）の組を $\mathtt{extract}(p) = (act^1, act^2, \ldots, act^m)$ で表わす．

加えて，$\langle v \rangle = (\vec{v}^1, \vec{v}^2, \ldots, \vec{v}^m)$ とする．ただし，$i \in \{1, \ldots, m\}$ について $\vec{v}^i \in [\![act(\vec{s})^i]\!]$．現在の状態と分布から導かれる全ての（非決定性）$\langle v \rangle$ を，集合 $\mathtt{Combinations}(\vec{s}, p)$ によって定義する．

$\mathrm{PLHA} = \langle \vec{x}, L, inv, dif, prob, (grd_l)_{l \in L}, E \rangle$ の意味に対応する並行確率システム $\mathrm{CPS}_{\mathrm{PLHA}}$ は，形式的に以下のように定義される．

PLHA の意味としての並行確率システム $\mathrm{CPS}_{\mathrm{PLHA}} = \langle Q_{\mathrm{PLHA}}, \Sigma_{\mathrm{PLHA}}, Steps_{\mathrm{PLHA}} \rangle$

- $Q_{\mathrm{PLHA}} \subseteq L \times \mathbb{R}^n$ は状態 $(l, \vec{s})$ の集合である．ただし，$\vec{s} \in [\![inv(l)]\!]$ を満たすものに限る．

- $\Sigma_{\mathrm{PLHA}} = \mathbb{R}_{\geq 0}$ であり，時間経過を意味する．

- 関数 $Steps_{\mathrm{PLHA}} : Q_{\mathrm{PLHA}} \to 2^{\Sigma_{\mathrm{PLHA}} \times \mathrm{Dist}(Q_{\mathrm{PLHA}})}$ ．

  各状態 $(l, \vec{s}) \in Q_{\mathrm{PLHA}}$ について，集合 $Steps_{\mathrm{PLHA}}(l, \vec{s}) = Cont(l, \vec{s}) \cup Disc(l, \vec{s})$ であり，

  遅延と分布の組 $(\delta, \mu) \in Steps_{\mathrm{PLHA}}(l, \vec{s})$ は，以下のように，PLHA から導出される．

  – 時間遷移 $Cont$ $\qquad (l, \vec{s}) \xrightarrow{\delta, \mu_{(l,\vec{s}')}} (l, \vec{s}')$

  各時間経過 $\delta \in \mathbb{R}_{\geq 0}$ に関して，$(\delta, \mu_{(l,\vec{s}')}) \in Cont(l, \vec{s})$ が存在し，その必要十分条件は，

  1. 全ての実数 $t \in [0, \delta]$ に対して，$fun(t) \in [\![inv(l)]\!]$ であり，かつ，
  2. 全ての実数 $t \in (0, \delta)$ に対して，$\frac{dfun(t)}{dt} \in [\![dif(l)]\!]$，及び
  3. $fun(0) = \vec{s}$，$fun(\delta) = \vec{s}'$ となる，微分可能な関数 $fun : [0, \delta] \to \mathbb{R}^n$ が存在する．

  つまり，時間遷移は，同一ロケーション内における，変数の評価値の連続的変化を表わす．

  また，時間経過は，ディラック分布 $\mu_{(l,\vec{s}')}$ により，確率 1 で確定的に行なわれる．

  – 離散遷移 $Disc$ $\qquad (l, \vec{s}) \xrightarrow{0, \mu_{(v)}} (l', \vec{s}')$

  $\delta = 0$ であり，非決定的に選択した PLHA の各離散確率分布 $p \in prob(l)$ に関して，

  もし $\vec{s} \in [\![grd_l(p)]\!]$ ならば，PLHA の分布 p から，以下のように構成される $\mathrm{CPS}_{\mathrm{PLHA}}$ の確率分

布 $\mu_{\langle v \rangle}$ が存在し，$(0, \mu_{\langle v \rangle}) \in Disc(l, \vec{s})$ である．

$$\mu_{\langle v \rangle}(l', \vec{s}') = \sum_{i \in \{1,\ldots,m=|\mathrm{support}(p)|\} \& l'=l'^i \& \vec{s}'=\vec{v}^i} p(X^i, updt^i, l'^i).$$

つまり，離散遷移は，非決定的に選択された実行可能な確率分布 $\mu_{\langle v \rangle}$ に従う生起確率 $\mu_{\langle v \rangle}(l', \vec{s}')$ の状態遷移を表わす．また，離散遷移は，遅延 0 で即時的に行なわれる．

ここで，PLHA の同一な確率分布 p に従う異なるエッジ e^i, e^j では，等しい状態に離散遷移することが無いとする仮定を設ける．すなわち，*semantics* レベルでは，

$$\mu_{\langle v \rangle}(l'^i, \vec{v}^i) = p(X^i, updt^i, l'^i).$$

となり，*syntax* レベルにおいて，PLHA の各確率分布 p に対し，次の制約を課すことに相当する．

$$\forall i, j \in \{1, \ldots, m = |\mathrm{support}(p)|\}\ (i \neq j), \quad if\ l'^i = l'^j\ then\ [\![act(\vec{s})^i]\!] \cap [\![act(\vec{s})^j]\!] = \emptyset.$$

6.5 記号的検証手法

本章では，確率線形ハイブリッドオートマトン PLHA に対する，記号的検証手法を提案する．

PLHA においては，変数のとり得る値が実数空間全体に及ぶため，状態 $(l, \vec{s})$ を 1 つ 1 つ区別したのでは，全状態数が非可算無限個となり，このままでは，直接，計算機上で扱うことはできない．このような，無限の状態空間に由来する複雑性を回避するために，本論文では，記号的な手法を取り入れ

る.

記号的であるとは，状態を陽に数え上げるのではなく，リージョンと呼ばれる状態集合単位で扱うことを指す．つまり，先に示したように，線形式により，評価値を範囲としてまとめて表現することに対応する．また，記号表現した状態集合に対しての演算（数式処理）を定義することにより，状態空間を有限の構造とみなして，検証を行なうことが可能となる．

対象とする検証問題は到達可能性である．到達可能性問題とは，システムが，その初期状態からターゲットである状態へ，遷移し得るか否かを判定する問題であり，システム検証の分野において最も基本的かつ中心的な問題のひとつである．

例えば，システムの安全性は，到達可能性によって記述可能であり，システムが危険な状態に陥ることが無い，つまりは到達しない，と示すことで検証できる．

6.6 逆像計算

到達可能性解析は，その方向性から，大きく以下の２つに分類される．

- 前向き解析 (*forward analysis*)

  初期状態から到達可能な全ての状態を求め，その中にターゲットが含まれているか調べる．

- 後向き解析 (*backward analysis*)

  ターゲットへ到達可能な全ての状態を，ターゲットから逆向きに，さかのぼることで求め，そ

の中に初期状態が含まれているか調べる.

また，一般に，後者のほうが，効率が良いとされている．ゆえに，本論文では，後向き解析の方針を採用し，線形式に対する記号処理によってそれを実現する．つまり，並行確率システム $\mathrm{CPS}_{\mathrm{PLHA}}$ の遷移関係 *Cont*, *Disc* に対する，後向き演算を定義することに他ならない.

それらは，逆像計算 (*inverse image computation*) と呼ばれ，以降で示す操作から構成される．特長は，現在の状態へ，1 度の状態遷移 (*Cont* もしくは *Disc*) で到達できるような，つまり，現在から遷移 1 回分過去に戻った全状態集合 (*predecessor*) を，一気呵成にまとめて求める点にある.

6.6.1 評価値の時間逆像 tpre

演算子 tpre は，単一ロケーション $l \in L$ における，現在の評価値の集合 $S_l \subseteq \mathbb{R}^n$ に対して，1 度の時間経過のみで，S_l へ遷移可能な，過去の評価値集合を記号的に求める.

つまり，現在の評価値集合を線形式 f_l で記号表現して ($[\![f_l]\!] = S_l$)，線形式 f_l に関する数式 (*symbolic*) 処理を行ない，結果である過去の評価値集合も，また，線形式 $\mathtt{tpre}(f_l)$ で返す.

$$\mathtt{tpre}(f_l) \;=\; inv(l) \wedge \left(\exists \delta \geq 0. \exists \vec{d}. \left(dif(l)[\vec{x} := \vec{d}] \;\wedge\; (f_l \wedge inv(l))[\vec{x} := \vec{x} + \delta \cdot \vec{d}] \right)\right).$$

上式においては，第 2 項 (括弧内) が，評価値の時間逆像計算を意味し，第 1 項 ($inv(l)$) によって，時間逆像をロケーション l の不変条件の範囲に限定する.

以下では，tpre により，単一ロケーション l における，評価値集合 S_l の全ての時間逆像のみが求められることを示す.

正当性の証明

変数の評価値 $\vec{s}$ が，時間逆像 $\mathtt{tpre}(f_l)$ に含まれるための，必要十分条件は，

$\vec{s} \in [\![inv(l)]\!]$，かつ，$\vec{s}' \in S_l$ について，時間遷移 $(l, \vec{s}) \xrightarrow{\delta, \mu(l,\vec{s}')} (l, \vec{s}')$ が存在することである．

まず，凸多面体の定義より，凸多面体は，内部の任意の 2 点を結ぶ直線について，直線上の全ての点が，同一の凸多面体に含まれるような点対の集合であると言える．

次に，不変条件 $inv(l) \in \mathtt{clf}$ は，凸状線形式であったので，$[\![inv(l)]\!]$ は凸多面体となる．また，フロー $dif(l) \in \mathtt{clf}$ は定数の形で与えられているため，各変数の傾きは一定であり，2 点 $\vec{s}, \vec{s}'$ を結ぶ線分は，直線となる．

つまり，$\vec{s}, \vec{s}' \in [\![inv(l)]\!]$ ならば，直線 $\vec{s}\vec{s}'$ 上の全ての点も，不変条件 $inv(l)$ を満たす．

以上をふまえて，時間遷移の定義より，評価値 $\vec{s}$ が，時間逆像 $\mathtt{tpre}(f_l)$ に含まれるか否かの判定は，$\vec{s}$ に関して，初期値 $\vec{s} \in [\![inv(l)]\!]$，時間遷移後の値 $\vec{s} + \delta \cdot \vec{d} \in [\![f_l \wedge inv(l)]\!]$，$\vec{d} \in [\![dif(l)]\!]$ となるような，時間遷移の存在，つまり遅延 δ，傾き $\vec{d}$ の存在を調べれば充分であることは明らか． ■

なお，実際の処理では，便宜上，先に示した式と同値な，以下の式を用いる．

$$\mathtt{tpre}(f_l) \;=\; inv(l) \wedge \Big(\exists \delta \geq 0. \exists \vec{c}. \Big((\delta \cdot dif(l))[\vec{x} := \vec{c}] \;\wedge\; (f_l \wedge inv(l))[\vec{x} := \vec{x} + \vec{c}]\Big)\Big).$$

6.6.2 リージョンの時間逆像 cpre

リージョンを $R = \bigcup_{l \in L}(l, S_l)$，その記号表現である述語を $\pi = \bigcup_{l \in L}(l, f_l)$ とする．

現在のリージョン π に対する，時間逆像のリージョンを求める演算子 cpre は，評価値の時間逆像 tpre から，以下のように構成される．

$$\mathrm{cpre}(\pi) \;=\; \bigcup_{l \in L}(l, \mathrm{tpre}(f_l)) \,.$$

つまり，ロケーション毎に，評価値の時間逆像 $\mathrm{tpre}(f_l)$ を求め，現在のリージョン π に対する，時間逆像を表わすリージョン $\mathrm{cpre}(\pi)$ を導出している．

6.6.3 記号的状態の離散逆像 epre

オートマトン PLHA の状態集合である，記号的状態 (l, S_l) は，ロケーション $l \in L$ と，l における評価値の集合 $S_l \subseteq \mathbb{R}^n$ の組で構成されるものとする．

演算子 epre は，現在の，単一ロケーション $l' \in L$ と，評価値集合 $S_{l'} \subseteq \mathbb{R}^n$ の組である記号的状態 $(l', S_{l'})$ に対して，1 回のロケーション (離散) 遷移のみで，$(l', S_{l'})$ へ遷移可能な，過去のリージョンを記号的に求める．

$$\mathrm{epre}(l', f_{l'}) = \bigcup_{e=(l,g,p,X,updt,l') \in E} \Big(l,\;\; inv(l) \wedge\; \exists \vec{x}'.(g \wedge act \wedge\; (f_{l'} \wedge inv(l'))[\vec{x} := \vec{x}'] \;) \Big) \,.$$

上式においては，第 2 項 ($\exists \vec{x}'$ 以降) が，各エッジ e に対する，評価値の離散逆像計算を意味し，第 1 項 ($inv(l)$) によって，離散逆像を遷移元ロケーション l の不変条件の範囲に限定する．

以下では，epre により，記号的状態 $(l', S_{l'})$ の全ての離散逆像（リージョン）のみが求められることを示す．

証明

各ロケーション l に対して，状態 $(l, \vec{s})$ が，離散逆像のリージョン $\mathrm{epre}(l', f_{l'})$ に属するのは，$\vec{s} \in [\![inv(l)]\!]$，かつ，$\vec{s}' \in S_{l'}$ について，（エッジ e による）離散遷移 $(l, \vec{s}) \xrightarrow{0,\mu(e)} (l', \vec{s}')$ が存在する場合である．

離散遷移の定義より，状態 $(l, \vec{s})$ が，離散逆像 $\mathrm{epre}(l', f_{l'})$ に属するか否かの判定は，$\vec{s}$ に関して，初期値 $\vec{s} \in [\![inv(l)]\!]$，$\vec{s} \in [\![g]\!]$，離散遷移後の値 $\vec{s}' \in [\![f_{l'} \wedge inv(l')]\!]$，$\vec{s}' \in [\![act(\vec{s})]\!]$ となるような，離散遷移の存在，つまり，評価値 $\vec{s}'$ の存在を調べれば充分であることは明らか． ■

6.6.4 リージョンの離散逆像 dpre

現在のリージョン π に対する，離散逆像のリージョンを求める演算子 dpre は，記号的状態の離散逆像 epre から，以下のように構成される．

$$\mathrm{dpre}(\pi) = \bigcup_{l' \in L} \mathrm{epre}(l', f_{l'}) .$$

つまり，ロケーション毎に，記号的状態の離散逆像 $\mathrm{epre}(l', f_{l'})$ を求め，現在のリージョン π に対する，離散逆像を表わすリージョン $\mathrm{dpre}(\pi)$ を導出している．

6.6.5 遷移確率

上に示した，各逆像計算は，確率が 0 でない (存在する) 遷移関係に対して，非確率的に，つまり，遷移する確率がいくつであるかは考慮せずに，逆像を求める．

ゆえに，ここでは，PLHA における，確率の情報を取り扱うために，遷移確率を保持する拡張表現と，この拡張表現に対して遷移確率を求める演算を定義する．

[確率付き多面体]

多面体の拡張表現である，確率付き多面体 (S, P_S) は，評価値の集合 (多面体) $S \subseteq \mathbb{R}^n$ と，S に割り当てられている確率 $P_S \in [0,1]$ の組で構成されるものとする．また，ロケーション $l \in L$ における，確率付き多面体を表わす，線形式と確率から成る組の集合を $\mathtt{poly}_l$ と定義する．

以後，リージョンは，$\pi = \bigcup_{l \in L}(l,\ \mathtt{poly}_l)$ と表記される．

[多面体の遷移確率] 記号的状態 $(l, (S, P_S))$ が，確率 ν で，$(l', (S', P_{S'}))$ へ遷移するとき，遷移後の多面体 S' に割り当てられる確率 $P_{S'}$ は，$P_{S'} = P_S \cdot \nu$ である．

これら拡張表現の意味するところは，多面体の和集合を，割り当てられている確率で区別することに他ならない．

ゆえに，逆像計算は，各多面体に対する，その遷移確率を求めることができるよう，以下のように改良される．ただし，逆像計算の一貫性を保つため，同じ確率分布 p に従う離散逆像計算によって，結果得られる多面体は，同一ロケーション上において共通部分を持たない と仮定する．(*cf.* P.145)

$$\mathtt{tpre}(\mathtt{poly}_l)$$

$$= \bigcup_{f_l \in \mathtt{poly}_l} \left(inv(l) \wedge \left(\exists \delta \geq 0. \exists \vec{c}. \left((\delta \cdot dif(l))[\vec{x} := \vec{c}] \ \wedge \ (f_l \wedge inv(l))[\vec{x} := \vec{x} + \vec{c}] \right) \right), \ P_{f_l} \cdot 1 \right) .$$

$$\mathtt{cpre}(\pi) \ = \ \bigcup_{l \in L} (l, \mathtt{tpre}(\mathtt{poly}_l)) .$$

$$\mathtt{epre}(l', \mathtt{poly}_{l'})$$

$$= \bigcup_{e=(l,g,p,X,updt,l') \in E} \left(l, \ \bigcup_{f_{l'} \in \mathtt{poly}_{l'}} \left(inv(l) \wedge \ \exists \vec{x}'. (g \wedge act \wedge \ (f_{l'} \wedge inv(l'))[\vec{x} := \vec{x}'] \ ), \ P_{f_{l'}} \cdot p(X, updt, l') \right) \right) .$$

$$\mathtt{dpre}(\pi) \ = \ \bigcup_{l' \in L} \mathtt{epre}(l', \mathtt{poly}_{l'}) .$$

以下のように定義し，次式は，リージョン π へ1度の遷移で到達可能な全状態（逆像）を意味する．

$$\mathtt{pre}(\pi) \ = \ \mathtt{cpre}(\pi) \cup \mathtt{dpre}(\pi) .$$

□

6.6.6 Quantifier Elimination

先に示した各逆像計算は，限定記号ヨを含む式で，逆像を表現している．とはいえ，直観的には，理解しにくく，逆像計算の結果得られる式は，不等式による区間の形であれば理想的である．

ゆえに，実際に解析を行なう際には，*QuantifierElimination*（限定記号消去,QE）[49] と呼ばれる数式処理手法を用い，逆像を表わす線形式に対し，その式と等価で，かつ，限定記号を含まない式を導出する．

6.7 到達可能性解析

到達可能性問題とは，オートマトンPLHAが，その初期状態*init*から，ターゲットリージョンTへ，遷移し得るか否かを判定する問題である．本節では，ターゲットへの到達可能性を，先に定義した逆像計算preを用いて解析する．それは，以下の2ステップから構成される．

1. ターゲットへ到達可能な全状態を，逆像計算を繰り返し適用することによって求め，
2. 初期状態からの確率的到達可能性を，論理式および確率的要求の充足関係から判定する．

6.7.1 確率的到達可能性

確率ハイブリッドシステムにおける到達可能性問題は，一般に $(\mathrm{T}, \sqsupseteq, \lambda)$ と表わすことができる．なお，$\sqsupseteq \in \{\geq, >\}$，$\lambda \in [0,1]$ である．つまり，ターゲットTへ到達するパスが存在し，なおかつ，到達確率 P_{Reach} が，要求 $P_{\mathrm{Reach}} \sqsupseteq \lambda$ を満たすか否か，例えば，$P_{\mathrm{Reach}} \geq 0.35$ であるかを判定する．

6.7.2 到達可能状態の反復計算

ターゲットTへの遷移が存在する全状態Qは，次式の最小不動点として求められる．なお，$Q_0 = \mathrm{T}$，$i \geq 1$は反復回数，ターゲットリージョンTの各多面体に割り当てられている初期確率は1である．

$$Q_i \;=\; \mathrm{T} \cup \mathrm{pre}(Q_{i-1})\,.$$

pre(T)は，ターゲットTへ1度の遷移で到達可能な全状態であった．このことから，Q_iは，多くともi回の遷移で，ターゲットへ到達可能な状態の集合を表わしていると解かる．つまり，上記の反復計算は，ターゲットへ到達可能な状態についての，後向きの幅優先探索であると言える．

6.7.3 到達可能性判定

PLHAの初期状態を$init = (l_0, f_{l_0})$，ターゲットであるリージョンをTと，述語によって表現する．PLHAが，ターゲットへ到達可能であるとは，以下の論理式Reachが真であることと同値である．

$$\mathrm{Reach} \;=\; (\; init \stackrel{*}{\Rightarrow} Q\;)$$

Tへ到達可能なリージョンQは，ロケーションと確率付き多面体の組の集合として表わされる．

$$Q \;=\; \bigcup_{l \in \mathcal{L}} (l,\ \mathrm{poly}_l) \;=\; \bigcup_{l \in \mathcal{L}} \Big(l,\ \bigcup_{f_l \in \mathrm{poly}_l} (f_l, P_{f_l})\Big)$$

論理式Reachの解釈は，

1. もし，$l = l_0$となる，ロケーションlがQ内に存在し，かつ，
2. f_{l_0}と共通部分を持つような，多面体f_lがl上に存在する場合，Reachは真であるとする．

また，Tへの到達確率 $P_{\text{Reach}} = P_{f_l}$ であるから，確率的要求 $P_{f_l} \sqsupseteq \lambda$ も判定可能であり，充足するならば，PLHAに対する確率的到達可能性問題 $(\mathrm{T}, \sqsupseteq, \lambda)$ は，YESであると結論づけられる． □

6.7.4 安全性

システムの安全性は，PLHAが危険な状態へ到達しないと示すことで検証できる．つまり，危険性を示すリージョンをターゲット T_d とした到達可能性解析を行ない，結果が偽であれば，危険状態へ到達せず，システムは安全といえる．

安全性の例を挙げると，

- デッドロックフリー (*deadlock-free*)

  システムが，デッドロックに陥らない．

- 排他制御 (*mutual exclusion*)

  複数のプロセスが，同時に共有リソースを獲得しない．

- 原子炉内の温度が，規定水準を超えない．などがある．

6.7.5 決定不能問題

線形ハイブリッドオートマトンに対する，到達可能性や，モデル検査などの検証問題は，一般に，決定不能 (*undecidable*) であることが知られている [4]．同様に，確率線形ハイブリッドオートマトンの到達可能性問題も，決定不能である．つまり，到達可能性解析における一連の手続きに対する停止

性は保証されていない.

ただ，もし止まれば，その結果は正しいものである．(部分決定可能，*semi-decision procedure*)

このような，ハイブリッドシステムが，生来有する複雑性の縮減を目的に，様々な近似・抽象化の手法が考案されている [4][50][51].

- *widening* [4]

  多面体に対する *widening* 演算を用いた近似により，不動点計算の収束性を保証する.

- 凸包 (*convex hull*) [26]

  凸多面体の選言 (和集合) を，凸包で近似表現.

- 述語抽象化 (*predicate abstraction*) [50]

  述語抽象化により，状態空間 (アナログ領域) の有限化及び単純化を図る.

6.8 検証事例

本章では，事例研究により，確率線形ハイブリッドオートマトン PLHA の記号的到達可能性解析について考察する.

[確率線形ハイブリッドオートマトンの事例]

まず，確率線形ハイブリッドオートマトン PLHA の例を示す．形式的には，以下のように記述できる.

PLHA $= \langle \vec{x}, L, inv, dif, prob, (grd_l)_{l \in L}, E \rangle$

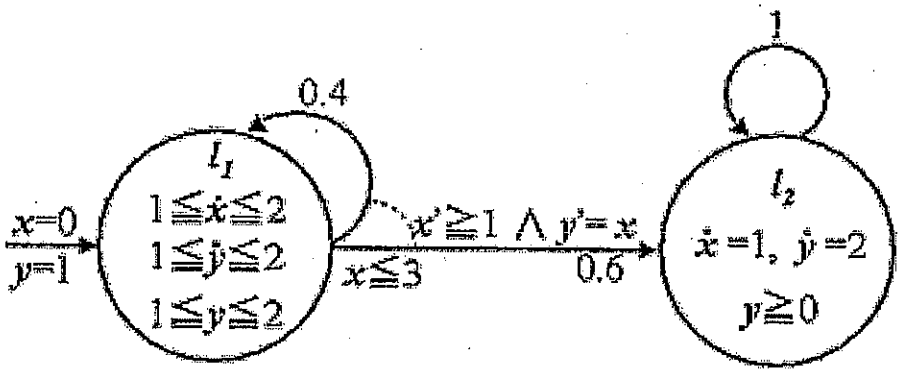
$$\vec{x} = \{x, y\}, \quad L = \{l_1, l_2\}, \quad inv(l_1) = 1 \leq y \leq 2, \quad inv(l_2) = y \geq 0,$$

$$dif(l_1) = 1 \leq \dot{x} \leq 2 \wedge 1 \leq \dot{y} \leq 2, \quad dif(l_2) = (\dot{x} = 1 \wedge \dot{y} = 2),$$

$$prob(l_1) = \{p_{l_1}\}, \quad prob(l_2) = \{p_{l_2}\}, \quad grd_l(p_{l_1}) = x \leq 3,$$

$$p_{l_1}(\{x, y\}, x' \geq 1 \wedge y' = x, l_2) = 0.6, \quad p_{l_1}(\emptyset, -, l_1) = 0.4, \quad p_{l_2}(\emptyset, -, l_2) = 1$$

図的表現では，以下のようになる．



このオートマトンPLHAは，l_1 と l_2，2つのロケーションより構成される．

l_2 において，変数 x は1，y は2の速度で，それぞれ増加する．l_1 では，変化にゆらぎがあるため，速度が1〜2となっている．このような，動作に幅を持つシステムも記述可能であることが解かる．また，l_1 においては，ランダム性による確率的な状態遷移がみてとれる．

6.8.1 記号的到達可能性解析例

次に，上記のPLHAに対する，到達可能性解析例を示す．

ここで，制御変数l及びPを導入する．lはロケーション，Pは確率に対する制御変数であり，結果として，リージョンを，ロケーション，線形式(多面体)，確率から成る組ではなく，全体が記号化された線形論理式(述語)によって表現可能となる．記法については，以降の事例を参照されたし．

到達可能性問題 $(\mathrm{T}, \sqsupseteq, \lambda)$

PLHAの初期状態$init$は，$init = (l = l_1 \wedge x = 0 \wedge y = 1)$である．このPLHAに対して，

到達可能性 $(\mathrm{T}, \geq, 0.35)$ を問う．なお，ターゲットリージョンTを，

$$\mathrm{T} = (l = l_2 \wedge 1 \leq x \leq 2 \wedge 2 \leq y \leq 3 \wedge P = 1)\,, \quad \text{図.26}$$

とし，システムが，ターゲットに，確率0.35以上で到達し得るか否か判定する．

一般に，$\mathrm{pre}(A \vee B) = \mathrm{pre}(A) \vee \mathrm{pre}(B)$であるから，到達可能状態の反復計算は，以下のように変形できる．なお，pre^iは，pre演算子のi回の適用を意味し，$Q_0 = \mathrm{T}$，$i \geq 1$である．

$$\begin{aligned} Q_i &= \mathrm{T} \vee \mathrm{pre}(Q_{i-1}) \\ &= \mathrm{T} \vee \mathrm{pre}(\mathrm{T} \vee \mathrm{pre}(\mathrm{T} \vee \mathrm{pre}(\mathrm{T} \vee \mathrm{pre}(\cdots)))) \\ &= \mathrm{T} \vee \mathrm{pre}(\mathrm{T}) \vee \mathrm{pre}(\mathrm{pre}(\mathrm{T})) \vee \mathrm{pre}(\mathrm{pre}(\mathrm{pre}(\mathrm{T}))) \vee \cdots \vee \mathrm{pre}^i(\mathrm{T}) \\ &= \mathrm{T} \vee Q_1 \vee Q_2 \vee \cdots \vee Q_{i-1} \vee \mathrm{pre}^i(\mathrm{T}) \end{aligned}$$

$$= \bigvee_{j=0}^{i-1} Q_j \vee \mathrm{pre}^i(\mathrm{T})$$
$$= Q_{i-1} \vee \mathrm{pre}^i(\mathrm{T}) .$$

つまるところ，Q_i を求める際には，その時点で，Q_j $(j = 0, 1, \cdots, i-1)$ および $\mathrm{pre}^{i-1}(\mathrm{T})$ が，既に得られているため，$\mathrm{pre}^i(\mathrm{T}) = \mathrm{pre}(\mathrm{pre}^{i-1}(\mathrm{T}))$ のみ計算すればよい．$(\star_2)$

また，$Q_{i-1} \subseteq Q_i$ (単調増加) より，$\mathrm{pre}^i(\mathrm{T}) = \emptyset$ となる Q_i が，最小不動点 Q であると解かる． □

実際に，逆像の反復計算を行なうと，以下のようになる．

まず，$\mathrm{pre}(\mathrm{T}) = \mathrm{cpre}(\mathrm{T}) \vee \mathrm{dpre}(\mathrm{T})$ を求める．(*cf.* P.152)

Quantifier Elimination を用い，限定記号∃を消去することで，式を簡約する．

ロケーション l_2 における，確率1の自己ループでは，離散逆像が変化しない $(= \mathrm{T})$ ため，この遷移に対する，離散逆像を求める必要は無い．

$\mathrm{cpre}(1 \le x \le 2 \wedge 2 \le y \le 3 \wedge P = 1)$

$= (y \ge 0 \wedge (\exists \delta \ge 0. \exists c_x, c_y. (\ c_x = \delta \wedge c_y = 2 \cdot \delta \ \wedge \ 1 \le x + c_x \le 2 \wedge 2 \le y + c_y \le 3 \wedge y + c_y \ge 0\ )) \wedge P = 1 \cdot 1)$

$= (y \ge 0 \wedge (\exists \delta \ge 0. (\ 1 \le x + \delta \le 2 \wedge 2 \le y + 2\delta \le 3\ )) \wedge P = 1)$

$= (x \le 2 \wedge 0 \le y \le 3 \wedge -2 \le y - 2x \le 1 \wedge P = 1)$. 図.27

$\mathrm{epre}(l = l_2 \wedge 1 \le x \le 2 \wedge 2 \le y \le 3 \wedge P = 1)$

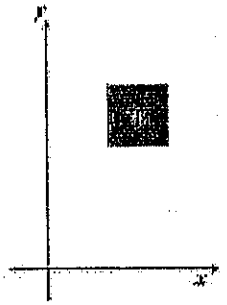


図 26: ターゲット T

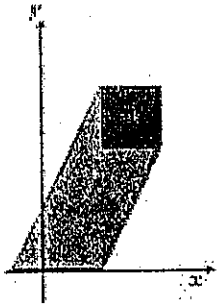


図 27: 時間逆像 cpre(T)

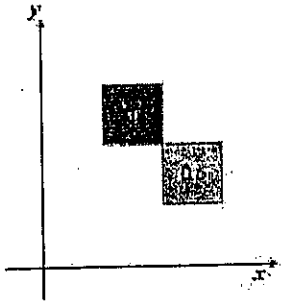


図 28: 離散逆像 dpre(T)

$= (l = l_1 \wedge (1 \le y \le 2 \wedge \exists x'.\exists y'.(\, x \le 3 \ \wedge \ x' \ge 1 \wedge y' = x \ \wedge \ 1 \le x' \le 2 \wedge 2 \le y' \le 3 \wedge y' \ge 0\,)) \wedge P = 1 \cdot 0.6)$

$= (l = l_1 \wedge (1 \le y \le 2 \wedge \exists x'.(\, x \le 3 \wedge 1 \le x' \le 2 \wedge 2 \le x \le 3\,)) \wedge P = 0.6)$

$= (l = l_1 \wedge 2 \le x \le 3 \wedge 1 \le y \le 2 \wedge P = 0.6)$. 図.28

pre(T)

$= (l = l_2 \wedge x \le 2 \wedge 0 \le y \le 3 \wedge -2 \le y - 2x \le 1 \wedge P = 1) \ \vee \ (l = l_1 \wedge 2 \le x \le 3 \wedge 1 \le y \le 2 \wedge P = 0.6)$

$= \pi_1 \vee \pi_2$.

また，以下のように，逐次的，つまり反復計算毎に，論理式の充足性を調べることで，最小不動点まで求めなくとも，到達可能性が判定できる．先の $(\star_2)$ と併せると，一般に，i 回目の反復計算では，新たに求められた $\mathrm{pre}^i(\mathrm{T})$ に対してのみ，到達可能性判定を行なえば充分であると言える．

$$\mathrm{Reach}_1 \ = \ (\ init \stackrel{*}{\Rightarrow} \mathrm{pre(T)}\ )$$

$$= \ \bot \ .$$

Tへ1度の遷移で到達可能な全状態 pre(T) に，*init* は含まれておらず，現段階では，到達不能.

続いて，$\mathrm{pre}(\mathrm{pre}(\mathrm{T})) = \mathrm{pre}(\pi_1 \vee \pi_2)$ を求める.

π_1 については，l_2 における時間逆像が，既に求められており $(= \pi_1)$，確率1の自己ループでは，離散逆像が変化しない $(= \pi_1)$ ため，この2つの遷移に対する逆像計算は不要.

$\mathrm{tpre}(2 \le x \le 3 \wedge 1 \le y \le 2 \wedge P = 0.6)$

$= (1 \le y \le 2 \wedge (\exists \delta \ge 0. \exists c_x, c_y. (\ \delta \le c_x \le 2\delta \wedge \delta \le c_y \le 2\delta \ \wedge\ 2 \le x + c_x \le 3 \wedge 1 \le y + c_y \le 2\ )) \wedge P = 0.6 \cdot 1)$

$= (1 \le y \le 2 \wedge (\exists \delta \ge 0. (\ 2 - x \le 2\delta \wedge 1 - y \le 2\delta \wedge \delta \le 3 - x \wedge \delta \le 2 - y\ )) \wedge P = 0.6)$

$= (x \le 3 \wedge 1 \le y \le 2 \wedge 2x - y \le 5 \wedge 2y - x \le 2 \wedge P = 0.6)$.

$\mathrm{epre}(\pi_1)$ は，簡単のため略.

$\mathrm{epre}(\pi_2)$

$= (l = l_1 \wedge (1 \le y \le 2 \wedge\ \exists x'. \exists y'. (\ x \le 3\ \wedge\ x' = x \wedge y' = y\ \wedge\ 2 \le x' \le 3 \wedge 1 \le y' \le 2\ )) \wedge P = 0.6 \cdot 0.4)$

$= (l = l_1 \wedge (1 \le y \le 2 \wedge\ x \le 3 \wedge 2 \le x \le 3 \wedge 1 \le y \le 2) \wedge P = 0.24)$

$= (l = l_1 \wedge 2 \le x \le 3 \wedge 1 \le y \le 2 \wedge P = 0.24).$

$\mathrm{pre}(\mathrm{pre}(\mathrm{T})) = \mathrm{epre}(\pi_1) \ \vee\ (l = l_1 \wedge x \le 3 \wedge 1 \le y \le 2 \wedge 2x - y \le 5 \wedge 2y - x \le 2 \wedge P = 0.6)$

$\vee\ (l = l_1 \wedge 2 \le x \le 3 \wedge 1 \le y \le 2 \wedge P = 0.24)$.

$$\mathrm{Reach}_2 \;=\; (\, \mathit{init} \stackrel{*}{\Rightarrow} \mathrm{pre}^2(\mathrm{T}) \,)$$

$$= \; \top .$$

$\mathrm{pre}^2(\mathrm{T})$ における，$(l = l_1 \wedge x \leq 3 \wedge 1 \leq y \leq 2 \wedge 2x - y \leq 5 \wedge 2y - x \leq 2 \wedge P = 0.6)$ と，初期状態 $\mathit{init} \;=\; (l = l_1 \wedge x = 0 \wedge y = 1)$ が，共通部分を持ち，なおかつ，到達確率 $P = 0.6$ が，確率的要求 $P = 0.6 \geq 0.35$ を充足する．ゆえに，オートマトン PLHA に対する，確率的到達可能性問題 $(\mathrm{T}, \geq, 0.35)$ は，YES であると判定される． □

6.8.2 効率化

先の到達可能性解析において，計算 $\mathrm{pre}^2(\mathrm{T})$ の際に，状態集合 $(l = l_1 \wedge 2 \leq x \leq 3 \wedge 1 \leq y \leq 2 \wedge P = 0.24) = \pi_3$ が求められた．しかし，この状態集合 π_3 は，現段階で既に，確率的要求に反し $P = 0.24 \not\geq 0.35$，また，一般に，遷移確率は単調減少するため，これ以降，π_3 の逆像を求めたとしても，結果得られる逆像 $\mathrm{pre}^i(\pi_3)$ が，確率的要求を充足することは無い．

ゆえに，引き続き，到達可能性解析を行なうならば，上記の状態集合 π_3 に対する逆像計算は，不要であり，このような，確率的要求を満たすことが無い状態集合の間引きによる，計算対象の削減ができると解かる．

また，演算子 pre を，以下のように定義しても，最終的に得られる最小不動点，つまり，ターゲッ

トTへ到達可能な全状態Qは，変わらない．

$$\mathtt{pre}(\pi) \;=\; \mathtt{cpre}(\pi) \;\vee\; \mathtt{dpre}(\mathtt{cpre}(\pi))\,.$$

時間遷移は反射的な関係であるので，その時間逆像 $\mathtt{cpre}(\pi)$ は，必ず現状態 π を含む．つまり，$\mathtt{dpre}(\pi)$ を，あえて別に求めなくとも，まず，π について，$\mathtt{cpre}(\pi)$ を求め，続いて，時間が戻りきっている $\mathtt{cpre}(\pi)$ に対して，離散逆像 $\mathtt{dpre}(\mathtt{cpre}(\pi))$ を計算しさえすれば，その結果に $\mathtt{dpre}(\pi)$ は含まれており，このように，逆像計算をまとめることが可能だと言える．

経験的には，上の方法により，検証コストを縮小できると予想される．

Procedure SRA:

Input: a probabilistic linear hybrid automaton PLHA;

an initial state $init = (l_0, f_{l_0})$;

a target region and a probabilistic requirement $(\mathrm{T}, \sqsupseteq, \lambda)$.

Output: YES, *reachable* / NO;

(a region Q which can reach T.)

/* computation of a region Q which can reach T. */

Y := T

Z := Y

repeat

Z := pre(Z)

Y := Y ∪ Z

until Z = ∅

/* judgment of reachability, Y is the form of $\bigcup_{l\in L}(l, \mathrm{poly}_l) = \bigcup_{l\in L}(l, \bigcup_{f_l\in \mathrm{poly}_l}(f_l, P_{f_l}))$. */

for each y ∈ Y wrt $l \in L$

if $l = l_0$

for each $f_l \in \mathrm{poly}_l$

if $f_{l_0} \wedge f_l \neq \emptyset$

if $P_{f_l} \sqsupseteq \lambda$

return YES, *reachable.*

end if

end if

end for each

end if

end for each

return NO.

6.9 むすび

本論文では，形式的検証理論に基づく，システムの信頼性保証を目的に，システム記述言語としての，確率線形ハイブリッドオートマトンを提案し，その記号的な到達可能性解析手法を開発した．

組込みシステムや，分散システムをはじめとして，実社会における多くのリアルタイム・ハイブリッドシステムが，確率線形ハイブリッドオートマトンにより，モデル化されることから，提案手法の適用対象となり得る領域は，決して狭いものではないと考える．

提案手法では，処理対象であるリージョンを，述語論理式として表現することで，その論理式に対しての，機械的な記号操作だけによる，検証プロセスを実現している．

具体的には，*Quantifier Elimination* [?] を導入し，数値計算ではなく，数式処理（計算機代数）的な算法を用いており，また，到達可能性も，論理式の充足性に帰着し判定することで，解析の際にグラフを構築しない，といった特徴を持つ．

到達可能性という，システム検証の分野における基本的な問題に対する，解法が得られたことで，今後，より高度な検証手法，例えば，モデル検査 [?] などへの応用が期待できる．

ただ，当座のところ，提案手法における，一連の到達可能性解析手続きについて，その具体的な計算量は考慮されておらず，依然検討の余地あり，と言える．

今後の課題および方針としては，解析の効率化，近似・抽象化手法の導入，モデル検査手法への理論的拡張，提案手法のプログラムとしての実装による有効性の評価が挙げられる．

7 まとめと今後の展望

本研究では, ハイブリッドシステムの観点から, 組込み型システムの信頼性が保証できる設計方法論を構築することであった. 本質的には, 組込み型システムはアナログ動作とデジタル動作が混在している. 従来の研究では, アナログ動作を抽象化してデジタル動作のみを対象とする手法がほとんどであった. なぜならば, 従来の計算機科学には, アナログ動作を対象とする考え方は存在しなかった. 本研究では, 計算機科学の立場から, ハイブリッドモデルを用いて, 組込み型システムの新たな検証理論, 設計方法論, 計算モデルなどを開発した. 具体的には以下である.

1. M. Abadi らの詳細化検証理論 [12] を拡張することにより, ハイブリッドシステムの演繹的な詳細化検証理論を開発した.
2. O. Maler らのフェーズ遷移システム [3] を拡張することにより, ハイブリッドシステムのモジュール毎の仕様記述と検証の手法を開発した.
3. ハイブリッドモデルの実問題への適用事例として, リアルタイムソフトウェアをハイブリッドシステムとしてモデル化して, そのスケジューラビリティ検証を行った.
4. 制御理論と計算機科学の統合化の一つの試みとして, 制御理論に基づく制御仕様からハイブリッドモデルへの統合的な手法を開発した.
5. J. Sportston [48] らの手法を拡張することにより, 確率線形ハイブリッドオートマトンの検証理論を開発した.

本研究で開発した理論や手法などにより, ハイブリッドモデルに基づく組込み型システムの段階的詳細化開発及びその検証, 既存の制御理論との統合化が実現できる. さらに, 確率の導入により, ソフ

トリアルタイム

今度の組込み型システムの開発手法に関する研究の展望としては, 以下が考えられる.

1. 最近, インターネットや携帯などにより, 組込み型システムも複雑化しており, ソフトリアルタイムシステムの重要性が増している. そこで, 確率ハイブリッドオートマトンや確率時間オートマトンを拡張することにより, ソフトリアルタイムシステムの仕様記述, 検証や性能評価などの手法を開発する必要がある.
2. 最近, オブジェクト指向分析設計手法が実用化されており, UML により, 開発するシステムも増加すると考えられる. そこで, UML によりモデル化されたシステムの検証手法の開発及びそのモデルからのリアルタイムタスクの構成手法などを開発する必要がある.

参考文献

[1] O. Maler. Hybrid Systems and Real-World Computations. *manuscript*, 1992.

[2] F.W. Vaandrager. Hybrid systems. *Images of SMC Research 1996*, pp.305-316, Stichting Mathematisch Centrum, 1996.

[3] O. Maler, Z. Manna, A. Pnueli. From timed to hybrid systems. *LNCS 600*, pp.447-484, 1992.

[4] R. Alur, C. Courcoubetis, T.A. Henzinger, P.-H. Ho. Hybrid Automata: An algorithmic approach to the specification and verification of hybrid systems. *LNCS 736*, pp. 209-229, 1993.

[5] N. Lynch, R. Segala, F. Vaandrager, H.B. Weinberg. Hybrid I/O Automata. *LNCS 1066*, pp. 496-510, 1996.

[6] T.A. Henzinger, Z. Manna, A. Pnueli. Towards refining temporal specifications into hybrid systems. *LNCS 736*, pp. 60-76, 1993.

[7] T.A. Henzinger. Hybrid automata with finite bisimulations. *LNCS 944*, pp. 324-335, 1995.

[8] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specifications of hybrid systems in CHARON. *LNCS 1790*, pp. 6-19, 2000

[9] T. A. Henzinger. Masaccio: a formal model for embedded components. *LNCS 1872*, pp. 549-563, 2000.

[10] T. A. Henzinger, B. Horowitz, C. M. Kirsch. Giotto: a time-triggered language for embedded programming. *LNCS 2211*, pp. 166-184, 2001.

[11] R. Alur, T.A. Henzinger, P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3), pp.181-201, 1996.

[12] M. Abadi, L. Lamport. The existence of refinement mappings. *Theoretical Computer Science*, 82(2):253-284, 1991.

[13] N. Bjorner, Z. Manna, H. Sipma, T. Uribe. Deductive Verification of Real-time Systems using STeP. *LNCS 1231*, pp 22-43, 1997.

[14] R. Alur, T.A. Henzinger. *Modularity for timed and hybrid systems.* LNCS 1243, pp. 74-88, 1997.

[15] R. Milner. Communication and Concurrency. *Prentice Hall*, P.260, 1989.

[16] N. Bjorner, Z. Manna, H. Sipma, T. Uribe. Deductive Verification of Real-time Systems using STeP. *Theoretical Computer Science*, Vol.253, No.1, pp.27-60, 2001.

[17] D. L. DIll. Trace Theory for Automatic Hierachical Verification of Speed-Independent Circuits. *MIT Press*, 1989.

[18] R. Gawlick, R. Segala, J.F. Sogaard-Andersen, N. Lynch. *Liveness in Timed and Untimed Systems.* Information and Computation, Vol.141, pp.119-171, 1998.

[19] Z. Manna, A. Pnueli. *Temporal Verification of Reactive Systems.* Springer-Verlag, P. 512, 1995.

[20] Y. Kesten, Z. Manna, A. Pnueli. Verification of Clocked and Hybrid Systems. *Acta Informatica*, Vol. 36, No. 11, pp. 836 - 912, 2000.

[21] Z. Manna, A. Pnueli. Completing the Temporal Picture. *Theoretical Computer Science*, Vol. 83, No. 1, pp. 97-130, 1991.

[22] M.R. Garey, D.S. Johnson. Computers and Intractability. *W.H. FREEMAN AND COMPANY*, P.340, 1979.

[23] A. Puri, P. Varaiya. *Decidability of Hybrid Systems with Rectangular Differential Inclusions.* LNCS 818, pp. 95-104, 1994.

[24] G. Pappas, G. Lafferriere, S. Yovine. *A new class of decidable hybrid systems.* LNCS 1569, pp 137-151, 1999

[25] A.M. Tilborg, G.M. Koob. Foundations of Real-time Computing:Formal Specifications and Methods. *Kluwer Academic Pub.*, P.316, 1991.

[26] R. Alur, T.A. Henzinger, P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. on Software Engineering*, 22(3), pp.181-201, 1996.

[27] M. Joseph. Real-Time Systems: Specification, Verification and Analysis. *Prentice Hall Intl.*, 1996.

[28] S. Vestal. Modeling and verification of real-time software using extended linear hybrid automata. *Fifth NASA Langley Formal Methods Workshop*, pp.95-106, 2000.

[29] V. Braberman, M. Felder. Verication of real-time designs: Combining scheduling theory with automatic formal verication. *Lecture Notes in Computer Science 1687*, pp.494-510, 1999.

[30] K. Altisen, G. Goessler, J. Sifakis. Scheduler modeling based on the controller synthesis paradigm. *Journal of Real-Time Systems*, No.23, pp.55-84, 2002.

[31] Z. Liu, M. Joseph. Specification and verification of fault-tolerance, timing and scheduling, *ACM Transactions on Languages and Systems*, Vol.21, No.1, pp.46-89, 1999.

[32] O. Sokolsky, I. Lee, and H. Ben-Abdallah. Specification and Analysis of Real-Time Systems with PARAGON, *Annals of Software Engineering*, Vol.7, pp.211-234, 1999.

[33] J. Lehoczky, L. Sha, Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior, *Proc. of the 10th IEEE Symposium on Real-Time Systems*, pp.166-171, IEEE, 1989.

[34] S. Yamane. Refinement Theory of Embedded systems based on Hybrid models. *The 2002 International Conference on Information and Knowledge Engineering*, pp.455-461, CSREA Press, 2002

[35] M.J.C. Gordon, T.F. Melham. Introduction to HOL. *Cambridge University Press*, P.471, 1991.

[36] S. Owre, J.M. Rushby, N. Shankar. PVS: A prototype verification system *LNAI 607*, pp. 748-752, 1992

[37] Integrated Systems Inc. Integrated Systems:Products families-MATRIXx. *http://customer.isi.com/matrixx/knowbase/*, 2001.

[38] 野波健蔵. MATLABによる制御系設計. 東京電機大学出版局, 1998.

[39] T. Villa, H. Wong-Toi, A. Balluchi, J. Preussig, A.L. Sangiovanni-Vincentelli, Y. Watanabe. Formal Verification of an Automotive Engine Controller in Cutoff Mode. *Proceedings of IEEE Conference on Decision and Control*, pp.4271-4276, 1998.

[40] T. Stauner, O. Muller, M. Fuchs. Using HyTech to verify an automotive control system. *LNCS 1201*, pp.139-153, 1997.

[41] K. Warwick. Control systems - An Introduction. *Prentice-Hall*, 1989.

[42] R. Alur, C. Coucoubetis, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science 138*, pp.3-34, 1995.

[43] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. *LNCS 939*, pp. 225-238, 1995.

[44] T.A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. *LNCS 1066*, pp. 377-388, 1996.

[45] T.A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HyTech: a model checker for hybrid systems. *Software Tools for Technology Transfer 1*, pp.110-122, 1997.

[46] M. Fuchs. Elektronische Hohenstandskontrolle EHC. *BMW Technical Report*, 1995.

[47] M. Kwiatkowska, G. Norman, and J. Sproston. Symbolic model checking for probabilistic timed automata. *Technical Report CSR-03-10, School of Computer Science, University of Birmingham*, 2003.

[48] J. Sproston. Model checking for probabilistic timed and hybrid systems. *PhD thesis, Technical Report CSR-01-04, School of Computer Science, University of Birmingham*, 2001.

[49] A. Tarski. A decision method for elementary algebra and geometry. *University of California Press, Berkeley and Los Angeles, California*, 2nd ed., 1951.

[50] R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems using counter-example guided predicate abstraction. *Technical Report MS-CIS-02-34, University of Pennsylvania*, 2002.

[51] E.M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Technical Report CMU-CS-03-104, School of Computer Science, Carnegie Mellon University*, 2003.

[52] E.M. Clarke, O. Grumberg, D.A. Peled. Model checking. *MIT Press*, 1999.