情報処理学会論文誌　プログラミング　Vol.6　No.3　20–32 (Dec. 2013)

無限小定数と限量子除去法による ハイブリッドシステムの検証に向けて

岩塚 卓弥[1,a)]　寺内 多智弘[1,b)]　結縁 祥治[1,c)]

受付日 2013年1月29日, 採録日 2013年8月26日

概要：ハイブリッドシステムの形式的検証における課題の 1 つは，連続的な値の変化を離散的な値の変化と同一のフレームワーク上で扱う方法である．連続的変化を超準解析に由来する無限小定数を用いて表現し，ホーア論理でハイブリッドシステムを検証するフレームワークとして末永らによる HOARE^{dt} が提案されている．本研究では，限量子除去法を適用することで HOARE^{dt} によるハイブリッドシステムの検証を支援する手法を提案する．HOARE^{dt} による検証では，無限小定数を持つ WHILE^{dt} という言語でハイブリッドシステムをモデル化する．WHILE^{dt} は手続き型プログラミング言語であるため，既存のプログラム検証手法を有効に利用できると期待される．本研究では，制約解消によるプログラム検証手法を利用し，制約解消法として限量子除去法を用いる．限量子除去ツールとして `Reduce/Redlog` および `QEPCAD B`，`Slfq` を組み合わせて利用する．本論文では，WHILE^{dt} で記述されたモデルと入力された各条件から，証明に必要な限量子付き論理式を生成する手法を記す．限量子除去ツールを用いた事前条件，ループ不変条件の検証と反例条件の提示による各条件の修正支援法について記述する．実際にハイブリッドシステムをモデル化し検証した例について結果を示し，考察する．検証実験の結果から，HOARE^{dt} フレームワークに限量子除去を用いる手法がハイブリッドシステムの検証に有用であることを示す．

キーワード：プログラム検証，ハイブリッドシステム，超準解析，限量子除去

Toward Verification of Hybrid System with Infinitesimal and Quantifier Elimination

Takuya Iwatsuka[1,a)]　Tachio Terauchi[1,b)]　Shoji Yuen[1,c)]

Received: January 29, 2013, Accepted: August 26, 2013

Abstract: We propose a new approach to the formal verification of hybrid systems. Hybrid systems comprise both discrete and continuous behaviors, and a challenge here is handling the both kinds of behaviors in a unified verification framework. To address the issue, Suenaga and Hasuo [11], [23] have recently proposed the HOARE^{dt} verification framework that employs the notion of an "infinitesimal" from nonstandard analysis to express continuous changes in a discrete framework. Building on their work, we propose a verification method inspired by constraint-based program analysis, and uses quantifier elimination. In HOARE^{dt}, hybrid systems are modeled in a language called WHILE^{dt}, which is an imperative language containing an infinitesimal constant. HOARE^{dt} mimics the Hoare logic and is designed to "look like" a standard verification framework for a (discrete) imperative language, facilitating the application of the existing verification techniques for imperative programs. Following this scheme, we adopt and apply constraint based techniques, originally designed for imperative program verification, to the verification problem formalized in HOARE^{dt}. We use quantifier elimination tools as constraint solvers. We use three quantifier elimination tools: `Reduce/Redlog`, `QEPCAD B` and `Slfq`. This paper shows a method for generating quantified first order formula sufficient for verifying partial correctness expressed in HOARE^{dt}. Our method verifies the correctness of user provided pre-and-post-conditions and loop-invariants, and generates counter-examples when they are incorrect. We show the potential of the method through several non-trivial examples, including the verification of a simplified version of the European Train Control System.

Keywords: program verification, hybrid system, nonstandard analysis, quantifier elimination

1. はじめに

連続的な値の変化と離散的な値の変化の両方を含むシステムをハイブリッドシステムと呼ぶ．自動車はハイブリッドシステムの一例であり，速度や加速度は連続的に変化し，ギアは離散的に変化する．そのほかにも航空機など様々な機械，システムがハイブリッドシステムである．

ハイブリッドシステムの信頼性の向上に関して，テストやシミュレーションなどの検査手法以外に形式的な検証手法も広く研究されている．ハイブリッドシステムの形式的検証の課題の1つとして，連続的変化と離散的変化をどのようにして同じフレームワークで扱うかという問題があげられる．その1つの方法として，無限小定数を用いて連続的変化を離散的変化として表現し，すべてを離散的なフレームワーク上で扱う研究 [23] が行われている．文献 [23] ではホーア論理に無限小定数を導入して拡張した HOARE$^{\mathrm{dt}}$ という論理を提案している．HOARE$^{\mathrm{dt}}$ ではハイブリッドシステムをプログラムとしてモデル化するため，既存のプログラム解析，プログラム検証の手法を利用してハイブリッドシステムの検証を行うことが可能となる．本研究では，HOARE$^{\mathrm{dt}}$ と限量子除去法を用いたハイブリッドシステムの検証手法を提案する．

HOARE$^{\mathrm{dt}}$ による検証では，ハイブリッドシステムをモデル化し，WHILE$^{\mathrm{dt}}$ という言語で記述する．WHILE$^{\mathrm{dt}}$ は **if** 文による分岐や **while** 文によるループなど，手続き型プログラミング言語と同様の構文を持つ．WHILE$^{\mathrm{dt}}$ で記述したモデルに対して，事前条件と事後条件を ASSN$^{\mathrm{dt}}$ という言語で記述する．HOARE$^{\mathrm{dt}}$ の規則と公理を用いて事前条件，モデル，事後条件の三つ組を証明することによって部分正当性を証明する．

WHILE$^{\mathrm{dt}}$ や ASSN$^{\mathrm{dt}}$ は超準解析に由来する無限小定数 **dt** を持つ．$A, B \in \mathrm{ASSN}^{\mathrm{dt}}$，$c \in \mathrm{WHILE}^{\mathrm{dt}}$ とすると，HOARE$^{\mathrm{dt}}$ における $\{A\}c\{B\}$ の正しさは，A，B，c 中の **dt** を $\frac{1}{i+1}$ $(i \in \mathbb{N})$ に置換した $\{A|_i\}c|_i\{B|_i\}$ が “ほとんどすべての i について正しい” ことと同値である [23]．本研究の提案手法では，制約解消によるプログラム検証の手法 [9], [19] を利用してハイブリッドシステムの検証を行う．“ほとんどすべての i について正しい” を限量子付き論理式で表現し，限量子付きの制約を計算する．この限量子付き論理式に対して限量子除去法を適用し，等価な限量子のない論理式に変換する．

限量子除去法として Cylindrical Algebraic Decomposition (CAD) による方法と Virtual Substitution による方法の2つを利用する．CAD による限量子除去法を実装したツールとして `QEPCAD B` [5] を，Virtual Substitution による限量子除去法を実装したツールとして `Reduce/Redlog` [25] をそれぞれ用いる．また，`QEPCAD B` を用いて論理式の簡略化を行う `Slfq` を使用する．

提案手法では，検証する事後条件に対して適切な事前条件とループ不変条件を推測し，限量子除去ツールを用いて各条件が証明に必要な制約を満たしているか否かを判定する．制約が充足されていないとき，事前条件とループ不変条件を修正し，制約充足の判定を繰り返す．反例の条件を発見し修正の手がかりとする方法を示し，本手法がハイブリッドシステムの検証に有用であることを述べる．

本論文の構成は次のとおりである．2章では HOARE$^{\mathrm{dt}}$ とその背景にある超準解析の用語定義などの詳細を示す．3章では HOARE$^{\mathrm{dt}}$ と限量子除去法を用いてハイブリッドシステムの検証を行う手順を単純な例を使用して説明する．4章では提案手法によるハイブリッドシステムの検証の実験結果を示し，有用性を確認する．5章では本論文のまとめと今後の課題について記す．

1.1 関連研究

HOARE$^{\mathrm{dt}}$ は文献 [23] で提案され，文献 [11] では HOARE$^{\mathrm{dt}}$ を用いたハイブリッドシステムの検証手法が提案されている．文献 [11] では，WHILE$^{\mathrm{dt}}$ で記述したモデルをプログラム変換の手法 [21] を応用して単純化した後，微分と限量子除去法を用いて事前条件，ループ不変条件の自動発見を試みる．ループ中の代入の回数について全称限量子のついた式に対し限量子除去法を適用しており，本研究の提案手法における限量子除去法とは対象となる式が異なる．

文献 [11] の手法は，成功すれば証明が自動で行える点で本研究の提案手法に対して有利である．対照的に，本研究の提案手法では事前条件，ループ不変条件の候補を人間が推測するため，発見する各条件の形を限定することができる．そのため，人間にとって扱いやすい，都合のよい形の事前条件を得ることができるという点で優れている．

ハイブリッドシステムをオートマトンとしてモデル化して検証する手法として，ハイブリッドオートマトン [2], [12] がよく知られている．ハイブリッドオートマトンでは離散的変化を有限有向グラフのノード間の遷移で表現し，各ノードにおける連続的な値の変化をフロー条件として与える．また，離散的変化が起きる条件をジャンプ条件として与える．

ハイブリッドオートマトンで記述したモデルに対し，検証する性質を CTL，TCTL などの様相論理の式で与え，モデル検査によって自動検証する．ハイブリッドオートマトンのモデル検査ツールとして `HyTech` [3], [13] がある．

ハイブリッドオートマトンではフロー条件とジャンプ条

[1] 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University, Nagoya, Aichi 464–8601, Japan
[a] iwatsuka@sqlab.jp
[b] terauchi@is.nagoya-u.ac.jp
[c] yuen@is.nagoya-u.ac.jp

件で離散的変化と連続的変化を明確に分けて記述するため，WHILE^{dt} を用いたモデルの方が記述の柔軟性の点で有利である．

演繹的なアプローチでハイブリッドシステムを検証する手法として，Platzer らが提案している KeYmaera [15] が知られている．

KeYmaera ではハイブリッドプログラムと呼ばれる言語でハイブリッドシステムをモデル化する．ハイブリッドプログラムは離散的な変化を変数への値の代入で表し，連続的な値の変化は変数に対してそれぞれ一次導関数を与えることで表している．値を変化させる命令を非決定的な選択，連接，繰返しの制御構文で組み合わせてシステムを記述する．

ハイブリッドプログラムで記述したモデルに対して，Differential Dynamic Logic [14] という論理で正当性を検証する．KeYmaera は KeY [1] を基にした対話的な証明の支援を行い，自動証明もサポートしている．ハイブリッドオートマトンで記述されたハイブリッドシステムのモデルをハイブリッドプログラムで記述することが可能であり，KeYmaera はこの変換機能を実装している．

KeYmaera では HOARE^{dt} による証明同様，ハイブリッドシステムを手続き型プログラミング言語と同様の構文を持った言語でモデル化するため柔軟な記述が可能である．また，HOARE^{dt} と同様にシステム全体の正しさを部品ごとの正しさから証明できる合成性を持つ．KeYmaera は GUI を備えた洗練されたツールであるが，Differential Dynamic Logic に特有の証明手法を開発する必要がある．これに対して，WHILE^{dt} は連続的変化を離散的変化と同一の命令で扱っているため，離散的プログラミング言語ですでに実績のあるプログラム検証の手法を取り入れることができる．

プログラム検証の手法として制約解消を用いる方法 [9], [18], [19] が提案されている．事前条件，プログラム，事後条件の組みに対し，自由変数を含むループ不変条件のテンプレートを入力する．最弱事前条件を計算することにより，検証に必要な制約条件を導出し，その制約条件を満たす自由変数への値の割当てを探索する．

この手法は入力したテンプレートに対して健全かつ完全である．すなわち，入力したテンプレートの形のループ不変条件が存在するならばこの手法で発見することが可能であり，この手法で発見することができなければ入力したテンプレートの形のループ不変条件は存在しない．

本研究における提案手法では，制約解消によるプログラム検証の手法を HOARE^{dt} に応用し，ハイブリッドシステムの検証に利用する．

文献 [9] では，計算したすべての制約条件から Farkas の補題 [20] を用いて全称限量子のない線形な制約に変換し，SMT ソルバで制約を満たす解を探索する．この方法では，すべての制約式を同時に扱う必要がある．これに対し我々の提案手法では，隣接するラベル間の各パスに対してそれぞれ 1 つの制約式を独立して扱うことができる．そのため，どのパスで制約が充足されていないかを確認しながら事前条件，ループ不変条件の修正を行うことが可能である点で優れている．

このほか，ハイブリッドシステムの検証に必要な制約を限量子で量化された論理式として導き，制約解消によって検証を行う手法 [10], [17], [22] が提案されている．

システムの安全性は，以下の条件を満たす帰納的不変条件 Inv を発見することで検証される．

(1) システムの初期状態が Inv を満たしている．
(2) Inv が検証する安全性の条件を満たしている．
(3) システムの状態が Inv を満たしているとき，可能な連続的，離散的値の変化によって Inv が満たされない状態に変化することはない．

これらの条件から制約となる論理式を導出して，制約を満たす Inv を探索する．Inv の探索には前述の文献 [9], [18], [19] と同様に，テンプレートを用いる．テンプレートにはシステム中に現れないフレッシュな変数が含まれ，これらの変数に対して適切な割当てが存在するか否かを判定する．結果として，$\exists\forall$ で量化された制約が得られ，解消に成功すればシステムの検証に成功する．文献 [22] はこの制約の解消に限量子除去ツールを用いている．アルゴリズムの異なる複数の限量子ツールを利用する手法は，本研究の提案手法にも取り入れている．

ハイブリッドオートマトンと同様に，離散的変化と連続的変化を明確に区分しているため，記述の柔軟性という点で HOARE^{dt} の方が有利である．また，これらの手法に対して本研究での提案手法は制約条件とシステムの間の関係が直感的であり，正しい不変条件を推測しやすいという点で優れている．

2. 無限小定数の導入によるホーア論理の拡張

本章では，ホーア論理に無限小定数を導入して拡張した HOARE^{dt} について，文献 [23] を抜粋し用語などの定義を記す．

2.1 超準解析と無限小定数

定義 2.1（超フィルタ）．
フィルタ $\mathcal{F} \subseteq \mathcal{P}(\mathbb{N})$ は以下を満たす族である．
(1) $X \in \mathcal{F}$ かつ $X \subseteq U$ ならば $U \in \mathcal{F}$
(2) $X, Y \in \mathcal{F}$ ならば $X \cap Y \in \mathcal{F}$
空でないフィルタ $\mathcal{F}$ が空集合を含まない，すなわち $\mathcal{F} \neq \mathcal{P}(\mathbb{N})$ であるとき，$\mathcal{F}$ を固有フィルタと呼ぶ．超フィルタは最大の固有フィルタである．任意の $S \subseteq \mathbb{N}$ に対して S か $\mathbb{N}\backslash S$ のいずれか一方のみが必ず超フィルタ $\mathcal{F}$ に含まれる．

補題 2.1.
$\mathcal{F}_0 = \{S \subseteq \mathbb{N} \mid \mathbb{N}\backslash S$ が有限 $\}$ を含む超フィルタ $\mathcal{F}$ が存在する．

以下では $\mathcal{F}$ を補題 2.1 の超フィルタに固定する．$\mathcal{F}$ は次の条件を満たす．

(1) $\mathcal{F}$ は有限の積と無限の和で閉じている．
(2) $S \subseteq \mathbb{N}$ なる任意の S について，S または $\mathbb{N}\backslash S$ のいずれか一方のみが必ず $\mathcal{F}$ に含まれる．
(3) $\mathbb{N}\backslash S$ が有限であるならば，$S \in \mathcal{F}$．

$\{i \mid \varphi(i)$ が成り立つ$\} \in \mathcal{F}$ のとき，“ほとんどすべての i に対して $\varphi(i)$ が成り立つ” という．

定義 2.2（超実数，超自然数）．
集合 $\mathbb{D}$（$\mathbb{D}$ は $\mathbb{N}$ または $\mathbb{R}$）に対して，集合 ${}^*\mathbb{D}$ を ${}^*\mathbb{D} \equiv \mathbb{D}^{\mathbb{N}}/\sim_{\mathcal{F}}$ と定義する．ここで $\sim_{\mathcal{F}}$ は以下のように定義する．

$$\{i \in \mathbb{N} \mid d_i = d'_i\} \in \mathcal{F} \Longleftrightarrow (d_0, d_1, \dots) \sim_{\mathcal{F}} (d'_0, d'_1, \dots)$$

特に混乱を招かない場合，同値クラス $[(d_i)_{i\in\mathbb{N}}]_{\sim_{\mathcal{F}}} \in {}^*\mathbb{D}$ を $[(d_i)_{i\in\mathbb{N}}]$ または $(d_i)_{i\in\mathbb{N}}$ と表記する．${}^*\mathbb{D}$ の要素 $\boldsymbol{d} \in {}^*\mathbb{D}$ を，$\mathbb{D}$ が $\mathbb{N}$ の場合は超自然数，$\mathbb{R}$ の場合には超実数と呼ぶ．また，$d \in \mathbb{D}$ を標準の数と呼ぶ．

$\boldsymbol{d} = [(d_i)_{i\in\mathbb{N}}]$ ならば，$(d_i)_{i\in\mathbb{N}}$ を $\boldsymbol{d} \in {}^*\mathbb{D}$ のシーケンス表記と呼ぶ．$(d_i)_{i\in\mathbb{N}}$ に対して，シーケンス表現は一意に定まらない．標準の数から超数へのマッピング $\mathbb{D} \hookrightarrow {}^*\mathbb{D}$ では，標準の数 d は超数 $[(d, d, \dots)]$ にマップされる．

定義 2.3（${}^*\mathbb{D}$ 上の操作と関係）．
k 個の有限な引数をとる任意の操作 $f : \mathbb{D}^k \to \mathbb{D}$ に対して，$f : ({}^*\mathbb{D})^k \to {}^*\mathbb{D}$ となる以下に定義される拡張が存在する．

$$f([(d_i^{(0)})_{i\in\mathbb{N}}], \dots, [(d_i^{(k-1)})_{i\in\mathbb{N}}]) \equiv [(f(d_i^{(0)}, \dots, d_i^{(k-1)}))_{i\in\mathbb{N}}]$$

二項関係 $R \subseteq \mathbb{D}^2$ に対しても $R \subseteq ({}^*\mathbb{D})^2$ となる以下に定義される拡張が存在する．

$$[(d_i)_{i\in\mathbb{N}}] \;\; R \;\; [(d'_i)_{i\in\mathbb{N}}] \overset{\text{def}}{\Longleftrightarrow} \text{ほとんどすべての } i \text{ について } d_i \; R \; d'_i$$

定義 2.4（無限小，無限大）．超実数 $\boldsymbol{d} \in {}^*\mathbb{D}$ がいかなる実数 $d \in \mathbb{D}$ よりも真に 0 に近いとき，$\boldsymbol{d}$ は無限小であるという．また，$\boldsymbol{d} \in {}^*\mathbb{D}$ がいかなる実数 $d \in \mathbb{D}$ よりも真に大きいとき，$\boldsymbol{d}$ は無限大であるという．

定義 2.5（無限に近い）．$x, y \in {}^*\mathbb{D}$ とする．$x - y$ が無限小であるとき，x と y は互いに無限に近いといい，以下のように表記する．

$$x - y \text{ が無限小} \overset{\text{def}}{\Longleftrightarrow} x \approx y$$

例 2.1（ω と ω^{-1}）．
超実数 $[(1, 2, 3, \dots)] \in {}^*\mathbb{N}$ を ω とする．ω は任意の標準の自然数 $n = [(n, n, n, \dots)]$ に対して，$n < i$ となる任意の i において $\omega_i > n_i$ となるため，$\omega > n$ である．ω の存在により，$\mathbb{N} \subsetneq {}^*\mathbb{N}$，$\mathbb{R} \subsetneq {}^*\mathbb{R}$ であるといえる．逆数 $\omega^{-1} = [(1, \frac{1}{2}, \frac{1}{3}, \dots)]$ は正（$0 < \omega^{-1}$）であり，かつ任意の標準の正の実数 $r > 0$ よりも小さい．すなわち，ω は無限大超実数であり，ω^{-1} は無限小超実数である．

2.2 HOARE$^{\mathrm{dt}}$ フレームワーク

2.2.1 モデリング言語 WHILE$^{\mathrm{dt}}$

変数の有限集合を **Var** とする．

定義 2.6（WHILE$^{\mathrm{dt}}$ の構文）．

$$\mathbf{AExp} \ni a ::= x \mid c_r \mid a_1 \; \mathbf{aop} \; a_2 \mid \mathbf{dt} \mid \infty$$
$$\mathbf{BExp} \ni b ::= \mathbf{true} \mid \mathbf{false} \mid b_1 \wedge b_2 \mid \neg b \mid a_1 < a_2$$
$$\mathbf{Cmd} \ni c ::= \mathbf{skip} \mid x := a \mid c_1; c_2 \mid \mathbf{if}\ b\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2 \mid \mathbf{while}\ b\ \mathbf{do}\ c$$

ただし，$x \in \mathbf{Var}$ であり，c_r は $r \in \mathbb{R}$ の定数，$\mathbf{aop} \in \{+, -, \cdot, \hat{}\}$ である．$\hat{}$ はべき乗を表す演算子であり，$a\hat{}b$ は a^b を意味する．**dt** および ∞ はそれぞれ例 2.1 の ω^{-1} と ω である．

WHILE$^{\mathrm{dt}}$ から定数 **dt**，∞ を除いた言語を WHILE 言語と呼ぶ．

定義 2.7（WHILE$^{\mathrm{dt}}$ のセクション）．
e を WHILE$^{\mathrm{dt}}$ の式とし，$i \in \mathbb{N}$ とする．e の i 番目のセクションとは e 中の **dt**, ∞ をそれぞれ定数 $c_{1/(i+1)}$，$c_{(i+1)}$ で置換した式であり，$e|_i$ と表記する．明らかに，$e|_i$ は WHILE 言語の式である．

無限小定数 $\omega^{-1} = [(1, \frac{1}{2}, \frac{1}{3}, \dots)]$ などの超実数をメモリに格納するための状態を以下のように定義する．

定義 2.8（状態と超状態）．超状態 $\boldsymbol{\sigma}$ は $\boldsymbol{\sigma} = \perp$（未定義）または $\boldsymbol{\sigma} : \mathbf{V} \to {}^*\mathbb{R}$ のいずれかである．また，（標準の）状態 σ は $\sigma = \perp$（未定義）または $\sigma : \mathbf{V} \to \mathbb{R}$ のいずれかである．

超状態全体からなる集合を **HSt**，状態全体からなる集合を **St** と表記する．

WHILE$^{\mathrm{dt}}$ の式の表示的意味論は慣例に従って定義されている．$c \in \mathbf{Cmd}$ について，$[\![c]\!]$ は超状態へのマッピングであり，$\boldsymbol{\sigma}$ からマップされる超状態を $[\![c]\!]\boldsymbol{\sigma}$ と表記する．

定義 2.9（シーケンス表記）．$(\sigma_i)_{i\in\mathbb{N}}$ を状態のシーケンスとする．これは次の方法で超状態（$[(\sigma_i)_{i\in\mathbb{N}}]$ と表記する）に持ち上げることができる．

- ほとんどすべての i について $\sigma_i = \perp$ ならば $(\sigma_i)_{i\in\mathbb{N}} \equiv \perp$ とする．
- そうでなければ，$[(\sigma_i)_{i\in\mathbb{N}}] \neq \perp$ であり，$[(\sigma_i)_{i\in\mathbb{N}}](x) \equiv [(\sigma_i(x))_{i\in\mathbb{N}}]$ とする．

$\sigma_i = \perp$ となる任意の $i \in \mathbb{N}$ について，$\sigma_i(x)$ は定義されていない．この場合 $\sigma_i(x)$ は任意の実数（0 など）と定める．

$\boldsymbol{\sigma}$ を超状態，$(\sigma_i)_{i\in\mathbb{N}}$ を状態のシーケンスとする．$(\sigma_i)_{i\in\mathbb{N}}$ を持ち上げた超状態 $[(\sigma_i)_{i\in\mathbb{N}}]$ が $[(\sigma_i)_{i\in\mathbb{N}}] = \boldsymbol{\sigma}$ のとき，

$(\sigma_i)_{i\in\mathbb{N}}$ を $\boldsymbol{\sigma}$ のシーケンス表現と呼ぶ．以下では $\boldsymbol{\sigma}$ のシーケンス表現を $(\boldsymbol{\sigma}|_i)_{i\in\mathbb{N}}$ と表記する．

2.2.2 表明言語 ASSN$^{\mathrm{dt}}$

定義 2.10（ASSN$^{\mathrm{dt}}$ の文法）．

$$\mathrm{AExp} \ni a ::= x \mid c_r \mid a_1 \ \mathbf{aop}\ a_2 \mid \mathbf{dt} \mid \infty$$
$$\mathrm{Fml} \ni A ::= \mathbf{true} \mid \mathbf{false} \mid A_1 \wedge A_2 \mid \neg A \mid a_1 < a_2 \mid \forall x \in {}^*\mathbb{N}.A \mid \forall x \in {}^*\mathbb{R}.A$$

ただし，$x \in \mathbf{Var}$ である．

ASSN 言語とは ASSN$^{\mathrm{dt}}$ から定数 $\mathbf{dt}$ と ∞ を除き，$\forall x \in {}^*\mathbb{R}$ と $\forall x \in {}^*\mathbb{N}$ をそれぞれ $\forall x \in \mathbb{R}$ と $\forall x \in \mathbb{N}$ に置き換えた言語である．

定義 2.11（ASSN$^{\mathrm{dt}}$ のセクション）．

e を ASSN$^{\mathrm{dt}}$ の式とし，$i \in \mathbb{N}$ とする．e の i 番目のセクションとは e 中の $\mathbf{dt}$，∞ をそれぞれ定数 $c_{1/(i+1)}$，$c_{(i+1)}$ で置換し，$\forall x \in {}^*\mathbb{D}(\mathbb{D} \in \{\mathbb{N}, \mathbb{R}\})$ を $\forall x \in \mathbb{D}$ に置換した式であり，$e|_i$ と表記する．明らかに，$e|_i$ は ASSN 言語の式である．

$A \in \mathbf{Fml}$ の意味論は慣例に従って定義されている．任意の $\boldsymbol{\sigma} \in \mathbf{HSt}$ について $\boldsymbol{\sigma} \models A$ とき，ASSN$^{\mathrm{dt}}$ の式 $A \in \mathbf{Fml}$ が正しいといい，$\models A$ と表記する．

補題 2.2（Sectionwise Satisfaction）．A を ASSN$^{\mathrm{dt}}$ の式，$\boldsymbol{\sigma}$ を超状態，$(\boldsymbol{\sigma}|_i)_{i\in\mathbb{N}}$ を $\boldsymbol{\sigma}$ のシーケンス表現とする．このとき，

$$\boldsymbol{\sigma} \models A \iff \text{ほとんどすべての } i \text{ について } \boldsymbol{\sigma}|_i \models A|_i$$

状態 $\sigma|_i$ と ASSN の式 $A|_i$ の関係 $\models$ の定義は慣例に従う．

2.2.3 HOARE$^{\mathrm{dt}}$ 論理

HOARE$^{\mathrm{dt}}$ では，WHILE$^{\mathrm{dt}}$ で記述されたモデルに対し，以下に定義する $\{A\}c\{B\}$ を用いて正当性を証明する．

定義 2.12（HOARE$^{\mathrm{dt}}$ におけるホーアの三つ組）．HOARE$^{\mathrm{dt}}$ におけるホーアの三つ組 $\{A\}c\{B\}$ とは，ASSN$^{\mathrm{dt}}$ の式 A，B と WHILE$^{\mathrm{dt}}$ の命令 c からなる三つ組である．

任意の超状態 $\boldsymbol{\sigma} \in \mathbf{HSt}$ について $\boldsymbol{\sigma} \models A$ ならば $[\![c]\!]\boldsymbol{\sigma} \models B$ が成り立つとき，$\{A\}c\{B\}$ は正当であるといい，$\models \{A\}c\{B\}$ と表記する．

$\{A\}c\{B\}$ は部分正当性の表明である．$\boldsymbol{\sigma}$ の状態で c の実行を開始したとき，停止しないならば $[\![c]\!]\boldsymbol{\sigma} = \bot$ であり，すなわち $[\![c]\!]\boldsymbol{\sigma} \models B$ である．式 A を事前条件，B を事後条件と呼ぶ．

HOARE$^{\mathrm{dt}}$ におけるホーアの三つ組の正当性は，各セクションごとの正当性から証明することができる．次章で述べる本研究の提案手法では，この性質を用いてハイブリッドシステムの検証を行う．

補題 2.3（Sectionwise validity）．A，B を ASSN$^{\mathrm{dt}}$ の式，$c \in \mathbf{Cmd}$ を WHILE$^{\mathrm{dt}}$ の命令とする．

$$\models \{A\}\ c\ \{B\} \iff \text{ほとんどすべての } i \text{ について } \models \{A|_i\}c|_i\{B|_i\}$$

HOARE$^{\mathrm{dt}}$ の推論規則は，通常の Hoare 論理と同じ規則を用いる．$\{A\}c\{B\}$ が HOARE$^{\mathrm{dt}}$ の推論規則から導出できるとき，$\vdash \{A\}c\{B\}$ と表記する．

定理 2.1（HOARE$^{\mathrm{dt}}$ の健全性 [23]）．
$\vdash \{A\}c\{B\}$ ならば $\models \{A\}c\{B\}$ である．

定理 2.2（HOARE$^{\mathrm{dt}}$ の相対完全性 [23]）．
$\models \{A\}c\{B\}$ ならば $\vdash \{A\}c\{B\}$ である．

3. 限量子除去法を用いた WHILE$^{\mathrm{dt}}$ プログラムの検証

HOARE$^{\mathrm{dt}}$ による証明は表明とプログラムに超実数が含まれている点を除けばホーア論理と同様に行うことができる．補題 2.3 の Sectionwise validity より，HOARE$^{\mathrm{dt}}$ において正当性を証明するためには，ほとんどすべてのセクションで標準のホーア論理による正当性の証明が可能であればよいことが分かる．本章では，HOARE$^{\mathrm{dt}}$ での証明を限量子付き論理式を用いて標準のホーア論理での証明に帰着する方法を示す．

3.1 Sectionwise validity の限量子付き論理式による表現

定理 3.1. $\vec{x} \subseteq \mathbf{Var}$，$P(\vec{x}, i)$ を式中の変数が $\vec{x} \cup \{i\}$ に含まれる ASSN 言語の式とする．また，$y, i \in \mathbb{N}$ とする．以下の式が真であるとき，ほとんどすべての i について $P(\vec{x}, i)$ が成り立つ．

$$\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow P(\vec{x}, i)$$

ただし，y は $P(\vec{x}, i)$ に含まれない変数とする．

証明. “ほとんどすべての i について $P(\vec{x}, i)$ が成り立つ”とは $\{i \mid P(\vec{x}, i)\} \in \mathcal{F}$ であることを指す．また，$\mathbb{N}\backslash S$ が有限であるならば，$S \in \mathcal{F}$ である．

$\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow P(\vec{x}, i)$ が成り立つとき，$\{i \mid i \geq y\} \subseteq \{i \mid P(\vec{x}, i)\}$．$\mathbb{N}\backslash\{i \mid i \geq y\} = \{i \mid i < y\}$ は有限集合である．したがって $\mathbb{N}\backslash\{i \mid P(\vec{x}, i)\}$ も有限集合である．

以上より，$\{i \mid P(\vec{x}, i)\} \in \mathcal{F}$ である． □

$\mathcal{F}$ には補有限でない無限集合が含まれるため，定理 3.1 の逆は一般には成り立たない．$\mathcal{F}_0 = \{S \subseteq \mathbb{N} \mid \mathbb{N}\backslash S \text{ が有限}\}$ とすると，以下の定理が成り立つ．

定理 3.2. $\{i \mid P(\vec{x}, i)\} \in \mathcal{F}_0$ ならば $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow P(\vec{x}, i)$．

証明. $\{i \mid P(\vec{x}, i)\} \in \mathcal{F}_0$ なので，$\mathbb{N}\backslash\{i \mid P(\vec{x}, i)\}$ は有限集合である．よって $\mathbb{N}\backslash\{i \mid P(\vec{x}, i)\}$ には上限 u が存在する．したがって u より大きい任意の自然数 i について $P(\vec{x}, i)$ である． □

ハイブリッドシステムの検証の際に使用する式 $P(\vec{x}, i)$ については，ほとんどが $\{i \mid P(\vec{x}, i)\} \in \mathcal{F}_0$ である．

定理 3.1 より，以下の定理が導ける．

定理 3.3. A，B を ASSN$^{\mathrm{dt}}$ の式，c を WHILE$^{\mathrm{dt}}$ の命令と

するとき，以下が成り立つ．

$$\models \exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow \{A|_i\}c|_i\{B|_i\} \text{ ならば } \models \{A\}c\{B\}$$

ただし，A，B，c 中に現れる変数は $\vec{x}$ に含まれるとし，y は $\vec{x}$ に含まれない変数とする．

定理 3.1 同様，定理 3.3 の逆も一般的には成り立たない．ただし，$\models \{A\}c\{B\}$ のとき，$\{i \mid \{A|_i\}c|_i\{B|_i\}\} \subseteq \mathbb{N}$ が有限であれば $\models \exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow \{A|_i\}c|_i\{B|_i\}$ である．

3.2　限量子除去

入力された限量子付きの論理式と等価な限量子のない論理式を出力することを，限量子除去と呼ぶ．実閉体の理論に対する限量子除去は決定可能であることが Tarski によって証明されている [24]．Tarski の示した限量子除去法の非効率なアリゴリズムを改良した以下に示す Cylindrical Algebraic Decomposition や Virtual Substitution など効率の良い手法が提案され，実装されている．

3.2.1　Cylindrical Algebraic Decomposition を用いた限量子除去法

1975 年に Collins によって Cylindrical Algebraic Decomposition（以下 CAD とする）を用いた限量子除去法が提案された [6]．

CAD を用いた限量子除去法は，入力中の不等式に次数の制限はない．計算複雑性は入力中の変数の数に対して二重指数関数となる [7]．CAD を用いた限量子除去のアルゴリズムを実装しているツールとしては，`Mathematica` や `QEPCAD B` [5] がある．

3.2.2　Virtual Substitution

Virtual Substitution [26] は，二次式の解の存在条件を用いた限量子除去法である．

計算複雑性は束縛変数の数に対して二重指数関数的 [8] であり，CAD を用いた限量子除去法と比較すると自由変数が多い場合に有利である．解の存在条件を利用することから入力の次数が二次式までに制限される点が短所である．複数の変数について限量子除去を行うとき，1 つの変数を除去することで他の変数の次数が上がる可能性があり，連続して適用できない場合がある．項の置換を繰り返すため，出力の式が冗長になりやすい点も欠点である．

Virtual Substitution が実装されているツールとしては数式処理システムの `Reduce` 上のパッケージとして提供されている `Redlog` [25] などがある．

3.2.3　CAD を用いた論理式の簡略化

大きな限量子なし論理式に対して，CAD を用いて簡略化を行うことができる [4]．この手法を実装したツールとして `Slfq` がある．`Slfq` は入力された限量子なし論理式を連言標準形に変換し，各項を `QEPCAD B` を用いて簡略化する．

Virtual Substitution の出力結果は冗長となる場合が多く，直感的に扱いにくい．Virtual Substitution の結果を `Slfq` で簡略化することで扱いを容易にすることができる．限量子付き論理式に対しても，内部の限量子なし論理式を簡略化することで全体の簡略化が可能である．

(1) 検証対象のハイブリッドシステムをモデル化し，WHILE^{dt} で記述
(2) 検証する事後条件を記述
(3) 事前条件・ループ不変条件を推測し，記述
(4) 検証器に 1〜3 で記述したモデルおよび条件を入力
　(a) 入力から各条件が満たすべき制約として限量子付き論理式を計算
　(b) 限量子除去ツールを用いて等価な限量子なし論理式に変換する
　(c) 検証に失敗した場合，反例の条件を求めて出力する
(5) 検証器の出力から，正しいループ不変条件が得られれば検証成功．そうでなければ 1 または 3 に戻る

図 1　限量子除去法と WHILE^{dt} によるハイブリッドシステム検証手順
Fig. 1　Procedure of verification with quantifier elimination and WHILE^{dt}.

```
1  x:=0;
2  v:=0;
3  t:=0;
4  a:=a0;
5  while(t<tmax){
6     x:=x+v*dt;
7     v:=v+a*dt;
8     t:=t+dt
9  }
```

図 2　等加速度運動する物体の WHILE^{dt} モデル
Fig. 2　A WHILE^{dt} model of uniformly accelerated motion.

3.3　検証手順

限量子除去法と WHILE^{dt} によるハイブリッドシステム検証の手順を**図 1** に示す．

はじめに，検証の対象であるハイブリッドシステムをモデル化し，WHILE^{dt} で記述する．次に，WHILE^{dt} で記述したモデルに対し，検証する事後条件を ASSN^{dt} 言語で記述する．WHILE^{dt} 言語は **if** 文による分岐や **while** 文による繰返しなど，命令型プログラミング言語と同様の制御構文を使用しており，プログラム検証の手法を応用して検証できるとされている．提案手法では，制約解消によるプログラム検証 [9], [18], [19] を応用する．

以下に具体例を示して手順を解説する．等加速度運動する物体を WHILE^{dt} で記述したモデルを**図 2** に示す．この物体は x を変位，v を速度，a を加速度，t を時間とし，$t = 0$，$v = 0$ の状態で $x = 0$ から $a = a_0$ で $t < t_{max}$ の間等加速度運動する．

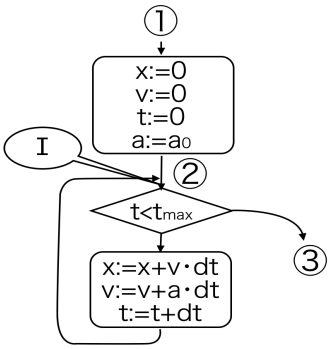


図 3　図 2 の CFG
Fig. 3　CFG for Fig. 2.

$$wp(\mathbf{skip}, I) = I \qquad wp(\text{assume } p, I) = p \Rightarrow I$$
$$wp(x := e, I) = I[e/x] \qquad wp(\text{assert } p, I) = p \wedge I$$
$$wp(S_1; S_2, I) = wp(S_1, wp(S_2, I))$$

図 4　最弱事前条件を求める関数
Fig. 4　Function yielding the weakest precondition.

上記の条件で等加速度運動する物体の $t = t'$ 時点での変位 x の値は $\frac{1}{2}at'^2$ である．ここで，$t < t_{max}$ の間等加速度運動した後の変位が $\frac{1}{2}a(t_{max} + \mathbf{dt})^2$ 以下であることを検証する．したがって，事後条件を $\frac{1}{2}a(t_{max} + \mathbf{dt})^2 \geq x$ とする．図 1 から作成したコントロールフローグラフ（以下 CFG とする）を**図 3** に示す．CFG の両端と **while** ループの先頭にあたる点にそれぞれラベルを付ける．図 3 中では①，②，③とラベル付けしている．

ラベル l_i について，それぞれ ASSN^{dt} の式 I_{l_i} と関係づける．CFG 上で他のラベルを通過しないパスを持つ 2 つのラベルを “隣接する” と定義する．図 3 において隣接するラベルは①と②，②と②，②と③である．隣接するラベル l_1，l_2 の間のパスの集合を $\text{Paths}(l_1, l_2)$ とすると，I_{l_1} と I_{l_2} の間で成り立つべき制約 $\text{VC}(l_1, l_2)$ は次のように定義される．

$$\text{VC}(l_1, l_2) = \bigwedge_{\pi \in \text{Paths}(l_1, l_2)} (\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow (I_{l_1}|_i \Rightarrow wp(\pi, I_{l_2}|_i))) \quad (1)$$

ただし，$\vec{x}$ は I_{l_1}，$wp(\pi, I_{l_2})$ に出現する変数の集合，y は $\vec{x}$ に含まれない変数である．また $wp(\pi, I)$ は**図 4** に定義する，パス π による I の最弱事前条件を求める関数である．

上記の方法で計算される制約を満たす事前条件およびループ不変条件の候補を推測する．

本手法の特徴として，文献 [9], [18], [19] の手法を応用しているため，ここで推測する各条件には自由変数を含めてもよい．

WHILE^{dt} モデルと検証する事後条件，推測した事前条件とループ不変条件を検証器に入力する．検証器は入力から上記の方法で各条件が満たすべき制約を計算する．計算した各制約条件は限量子付き論理式として表される．これらの式に対してそれぞれ限量子除去ツールを用いて等価な限量子のない論理式に変換する．

(1) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow (\text{Pre}|_i \Rightarrow I[a_0/a][0/t][0/v][0/x]|_i)$
(2) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge t < t_{max})|_i \Rightarrow I[(t + \mathbf{dt})/t][(v + a\mathbf{dt})/v][(x + v\mathbf{dt})/x]|_i)$
(3) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge \neg(t < t_{max}))|_i \Rightarrow \text{Post}|_i)$

図 5　図 3 から計算される制約
Fig. 5　Calculated constraints from Fig. 3.

変換を効率良く行うために，文献 [22] と同様に複数の限量子除去ツールを組み合わせて使用する．初めに，計算時間の点で最も有利な Virtual Substitution による限量子除去法を実装した `Reduce/Redlog` に限量子付き論理式を入力する．限量子除去に成功した場合は出力を `Slfq` に入力し，式を簡略化する．

`Reduce/Redlog` はすべての限量子が除去できなかった場合，可能な限り限量子を除去した限量子付き論理式を出力する．この場合，出力された限量子付き論理式の内側の限量子なし部分を `Slfq` で簡略化する．簡略化された限量子付き論理式を CAD による限量子除去法を実装した `QEPCAD B` に入力して等価な限量子のない論理式に変換する．

検証器に入力する事前条件，ループ不変条件の候補に自由変数が含まれない場合，限量子除去の結果は **true** または **false** である．結果が **true** である場合計算した制約を満たしており，**false** の場合は制約が満たされていない．入力に自由変数を含めた場合，限量子除去の結果は **true**，**false** のほかに自由変数のみからなる限量子のない論理式となる場合がある．自由変数に対して，限量子除去の結果の式が真となる任意の値を代入することで，証明に必要な事前条件，ループ不変条件が得られる．

それぞれの制約に対して限量子除去を行い，結果の連言が **false** でなければ検証成功である．検証に失敗した場合，再び別の事前条件，ループ不変条件を推測し，検証器に入力する．図 1 中 4c の反例出力については次節に詳細を記す．検証が成功するまで条件の推測と検証器による制約充足の確認を繰り返す．

図 3 の例において，$I_{①} = \text{Pre}$，$I_{②} = I$，$I_{③} = \text{Post}$ とすると，3 つのパスから**図 5** に示す 3 つの制約が得られる．ただし $\vec{x} = \{x, v, t, a, a_0, t_{max}\}$ である．ここで，Post は検証する事後条件，すなわち $\frac{1}{2}a(t_{max} + \mathbf{dt})^2 \geq x$ であり，Pre，I はそれぞれ推測する事前条件，ループ不変条件である．

式 (1) から，図 5 の各制約の連言が $\text{VC}(\text{Pre}, \text{Post})$ である．限量子除去の結果は等価な除去前の式と等価な式であるため，各制約について限量子除去法を適用した後にそれぞれの結果の連言をとればよい．

制約解消によるプログラム検証の手法 [9], [18], [19] では，不変条件に現れる単項式 $\{a_i \mid 0 \le i \le n\}$ を推測し，その係数を $\{c_{ijk} \mid 0 \le i \le n\}$ として不変条件のテンプレート $\bigvee_k \bigwedge_j (\sum_{i=0}^{n} c_{ijk} a_i \ge 0)$ を作り，適切な不変条件を求める．この手法を用いて，不変条件のテンプレートを作る．ここでは簡単のため，α，β を自由変数としたテンプレート I を以下のように作成し，図 2 と以下の事前条件，事後条件とともに検証器に入力する．

$$\mathrm{Pre} = (a_0 > 0 \wedge t_{max} > 0)$$
$$I = (t \ge 0 \wedge t_{max} > 0 \wedge a_0 > 0 \wedge \alpha a t^2 \ge x \wedge v = \beta a t \wedge t < t_{max} + \mathbf{dt} \wedge a = a_0)$$
$$\mathrm{Post} = \left(\frac{1}{2} a (t_{max} + \mathbf{dt})^2 \ge x\right)$$

図 5 の各制約について以下の結果が得られる．

(1) `Reduce/Redlog` による限量子除去に成功．**true** が出力される．
(2) `Reduce/Redlog` による限量子除去に成功．`Slfq` による式の簡略化の結果 $2\alpha - 1 \ge 0 \wedge \beta - 1 = 0$ が出力される．
(3) `Reduce/Redlog` による限量子除去に失敗．y，t_{max} について限量子が残った式が出力される．出力された限量子付き論理式の限量子のない部分を `Slfq` で簡略化した結果，$2\alpha - 1 \le 0$ が出力される．$(\exists y \forall t_{max}(2\alpha - 1 \le 0)) = 2\alpha - 1 \le 0$ なので結果として限量子のない論理式が得られる．

各制約を変換した限量子のない論理式の連言から，$\alpha = \frac{1}{2}$，$\beta = 1$ が得られる．したがって，$I = (t \ge 0 \wedge t_{max} > 0 \wedge a_0 > 0 \wedge \frac{1}{2} a t^2 \ge x \wedge v = at \wedge t < t_{max} + \mathbf{dt} \wedge a = a_0)$ とすることで，図 2 で表されるハイブリッドシステムに関して事前条件 Pre，事後条件 Post の部分正当性が証明できる．

3.4　反例の出力

システムに不具合があった場合や，推測した事前条件やループ不変条件が誤っていた場合，検証に失敗し，限量子消去の結果 **false** と出力される場合がある．このとき，システムの修正や再び適切な条件を推測する手がかりとして，反例を利用することができる．

反例を得るためには，制約の限量子付き論理式の否定をとった式を限量子除去ツールに入力すればよい．検証器が出力する制約は $\exists y \forall \vec{x} \forall i P$ という形の式となるため，その否定は $\forall y \exists \vec{x} \exists i \neg P$ の形となる．ここで，$\vec{x}$ 中の変数の一部を自由変数とし，限量子除去を行う．$\vec{z} \subseteq \vec{x}$ を自由変数として限量子消去するとき，明らかに以下の式 (2) が成り立つ．

$$\forall \vec{z}((\forall y \exists (\vec{x} \setminus \vec{z}) \exists i \neg P) \Rightarrow (\forall y \exists \vec{x} \exists i \neg P)) \qquad (2)$$

$\forall y \exists (\vec{x} \setminus \vec{z}) \exists i \neg P$ に対して限量子除去法を適用した結果を $\phi(\vec{z})$ とする．式 (2) より $\phi(\vec{z})$ が真であるとき，$\forall y \exists \vec{x} \exists i \neg P$ も真となる．したがって，$\phi(\vec{z})$ が真となる $\vec{z}$ から $\exists y \forall \vec{x} \forall i P$ の反例が得られる．

ここでは，推測した条件が誤っていた場合の反例出力の例を示す．図 3 の例で，ループ不変条件のテンプレート I から $a_0 \ge 0$ という条件が欠けていたとする．すなわち，図 5 中の `Pre`，I，`Post` を以下の式とする．

$$\mathrm{Pre} = t_{max} > 0$$
$$I = t \ge 0 \wedge t_{max} > 0 \wedge \frac{1}{2} a t^2 \ge x \wedge v = at \wedge t < t_{max} + \mathbf{dt} \wedge a = a_0$$
$$\mathrm{Post} = \frac{1}{2} a (t_{max} + \mathbf{dt})^2 \ge x$$

このとき各制約について限量子除去を行った結果はそれぞれ **true**，**false**，**false** となる．a_0 を自由変数とし，各制約の否定をとって限量子除去を行うと，結果として **false**，$a_0 < 0$，$a_0 < 0$ が得られる．この結果から，VC(②,②)，VC(②,③) において，$a_0 < 0$ が反例の条件であることが分かるため，I に $a_0 \ge 0$ を追加する．

I を修正して限量子除去を行うと，3 つの制約式に対して **false**，**true**，**true** という結果がそれぞれ得られる．同様に反例条件を計算すると，VC(①,②) の反例の条件が $a_0 < 0$ であることが分かる．`Pre` に $a \ge 0$ を追加して検証器に入力すると，すべての制約を満たすことが確認できる．

4.　実験

本章では，HOARE$^{\mathrm{dt}}$ と限量子除去法を用いたハイブリッドシステムの検証例を示す．

検証システムの全体を**図 6** に示す．HOARE$^{\mathrm{dt}}$ で記述したシステムと検証する各条件を，我々の実装した制約導出ツールに入力し，限量子付きの論理式を得る．得られた論理式を `Reduce/Redlog` で限量子除去し，`Slfq` で出力を簡略化する．`Slfq` は内部的に `QEPCAD B` を利用する．

検証器の実行に使用した環境は，仮想マシン上の Ubuntu で CPU は 4 コア，メモリは 3.6 GB を割り当てている．仮想マシンを実行している環境は Quad-Core Xeon 2.8 GHz プロセッサ搭載の Mac Pro である．実行時間として，time コマンドが出力する CPU 時間を記す．

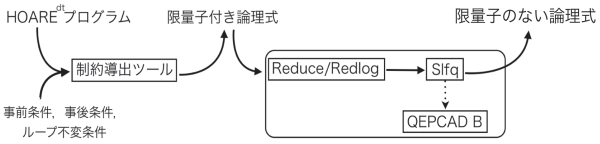


図 6　検証システム
Fig. 6　Verification system.

4.1　地面でバウンドする物体

ハイブリッドシステムの例として地面でバウンドする

```
1   g:=9.8;
2   t:=0;
3   h:=hmax;
4   while (t<tmax) {
5     if (h<=0 ∧ v<0) {
6       v:=−c∗v
7     } else {
8       skip
9     };
10    h:=h+v∗dt;
11    v:=v−g∗dt;
12    t:=t+dt
13  }
```

図 7　地面でバウンドする物体の WHILE$^{\mathrm{dt}}$ モデル
Fig. 7　A WHILE$^{\mathrm{dt}}$ model of bounce object.

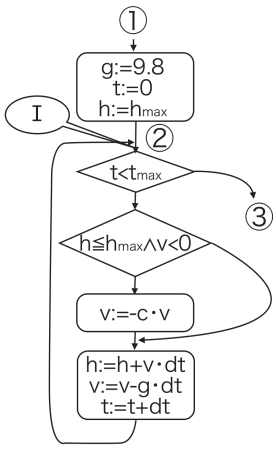


図 8　図 7 の CFG
Fig. 8　CFG for Fig. 7.

物体をモデル化する．物体は高さ h_{max} の位置から初速度 0 で自由落下する．物体はつねに重力加速度 $g = 9.8$ で地面に向かって加速し，接地すると跳ね返る．弾性係数を c（$0 \leq c < 1$）とする．これは有限時間内に無限回の離散的変化が起こる Zeno と呼ばれる振舞いの例である．Zeno はハイブリッドシステムの解析，検証を困難にする振舞いであることが広く知られている．この物体について，高さがつねに地面と h_{max} の間であることを検証する．

WHILE$^{\mathrm{dt}}$ で記述したモデルを**図 7** に示す．また，図 7 から作成した CFG を**図 8** に示す．図 7 の 5〜9 行目は，物体が接地した場合に速度を変化させるための分岐である．10〜12 行目の代入文で物体の微小時間における値の変化を表している．4 行目の **while** ループは $t \geq t_{max}$ のときに抜ける．1 回のループで t は無限小定数 **dt** ずつ加算されるので，任意の $t_{max} > 0$ について事後条件 Post が成り立てば，物体が運動中つねに Post が成り立っているといえる．

事前条件を Pre，ループ不変条件を I，事後条件を Post とすると，以下の 4 つの制約が計算される．

(1) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow (\mathrm{Pre}|_i \Rightarrow I[9.8/g][0/t][h_{max}/h]|_i)$
(2) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow$
$((I \wedge t < t_{max})|_i \Rightarrow ((h \leq h_{max} \wedge v < 0) \Rightarrow I[(t+\mathbf{dt})/t][(v - g\mathbf{dt})/v][(x + v\mathbf{dt})/x][-cv/v]|_i))$
(3) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow$
$((I \wedge t < t_{max})|_i \Rightarrow (\neg(h \leq h_{max} \wedge v < 0) \Rightarrow I[(t+\mathbf{dt})/t][(v - g\mathbf{dt})/v][(x + v\mathbf{dt})/x]|_i))$
(4) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge \neg(t < t_{max}))|_i \Rightarrow \mathrm{Post}|_i)$

(1) は図 8 における VC(①,②)，(2) と (3) は VC(②,②)，(4) は VC(②,③) を計算することで得られる制約である．

検証器に以下の各条件を入力した結果，検証に成功した．

$$\mathrm{Pre} = (0 \leq c \wedge c < 1 \wedge t_{max} > 0 \wedge g = 9.8 \wedge h_{max} \geq 0 \wedge v = 0)$$

$$I = (0 \leq c \wedge c < 1 \wedge t_{max} > 0 \wedge g = 9.8 \wedge h_{max} \geq 0 \wedge t < t_{max} + \mathbf{dt} \wedge v^2 \leq 2g(h_{max} - h))$$

$$\mathrm{Post} = ((h \leq 0 \Rightarrow h^2 \leq 2gh_{max}\mathbf{dt}^2) \wedge h < h_{max})$$

事後条件中の $(h \leq 0 \Rightarrow h^2 \leq 2gh_{max}\mathbf{dt}^2)$ は，$-\sqrt{2gh_{max}}\mathbf{dt} \leq h$ と等価な式である．ただし，後者を検証器に入力すると限量子除去に失敗するため，前者の形で入力する必要がある．**dt** は無限小定数であり，$\sqrt{2gh_{max}}$ は標準の実数であるので，$\sqrt{2gh_{max}}\mathbf{dt}$ は無限小の値である．したがって，前述の事後条件で $(0 \approx h \vee 0 < h) \wedge h < h_{max}$ を表しているといえる．

制約式の計算に要した時間は 0.096 秒であった．すべての制約式に対して `Reduce/Redlog` が限量子除去に成功し，**true** という結果を出力した．限量子除去に要した時間は 0.964 秒であった．

4.2　ETCS

European Train Control System のうちの一部（以下 ETCS）を切り出し，簡略したシステムを対象として，提案手法で検証する例を示す．具体的には，文献 [16] に記されたモデルを基に，文献 [23] で HOARE$^{\mathrm{dt}}$ に移植されたシステムを対象とする．

ETCS は電車と，電車の走行を無線で管理するコントローラの 2 つのコンポーネントからなるシステムである．ETCS ではあらかじめ電車の加速度の上限 A・下限 $-b$ と，目標地点までの安全距離 s，電車の速度と変位を検査する時間間隔 ϵ が定められている．また，電車の速度を v，加速度を a，変位を z と表す．

コントローラは，電車の目標地点 m と推奨速度，目標地点に到達したときの電車の速度の上限を定める．電車は，速度検査，距離検査，走行処理の 3 つの処理を順に繰り返す．それぞれの処理の内容は以下のとおりである．

- 速度検査：速度 v が推奨速度以下であれば，加速度 a

```
1   e:=dt;
2   while (v>0) {
3     t:=0;
4     if (m-z<s) {
5       a:=-b
6     } else {
7       a:=0
8     };
9     while (t<e) {
10      z:=z+v*dt;
11      v:=v+a*dt;
12      t:=t+dt;
13    }
14  }
```

図 9　ETCS の WHILE$^{\mathrm{dt}}$ モデル
Fig. 9　A WHILE$^{\mathrm{dt}}$ model of ETCS.

を $-b$ から A までの任意の値に変更する．そうでなければ加速度 a を $-b$ から 0 までの任意の値に変更する．

- 距離検査：変位 z と目標地点 m との差が安全距離 s 未満の場合に加速度を $-b$ に変更する．
- 走行処理：時間 t を 0 に変更し，電車を速度 v，加速度 a で $t < \epsilon$ の間走行させる．

ETCS は加速度の変化による非線形な振舞い，ϵ ごとの t のリセットを含むため，モデル化や検証が困難な例であるとされている [14]．

簡単のため，ETCS を以下のように簡略化する．

- コントローラは目標地点 m のみを定め，m の値を更新しない．
- 電車は距離検査と走行処理のみを行い，初速度を v_0，初期の加速度を 0 に固定する．

簡略化したシステムに対して，電車が停止したとき，目標地点を超えて走行していないことを検証する．

簡略化した ETCS を WHILE$^{\mathrm{dt}}$ で記述したモデルを**図 9** に示す．また，図 9 から作成した CFG を**図 10** に示す．

図 9 のモデルは **while** ループがネストした構造をしている．図 10 に示したように，外側のループ不変条件を I，内側のループ不変条件を I' とし，事前条件，事後条件をそれぞれ Pre，Post とすると，以下の 6 つの制約条件が計算される．

(1) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow (\mathrm{Pre}|_i \Rightarrow I[\mathbf{dt}/\epsilon]|_i)$
(2) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge \neg(v > 0))|_i \Rightarrow \mathrm{Post}|_i)$
(3) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge v > 0)|_i \Rightarrow (m - z < s \Rightarrow I'[-b/a])[0/t]|_i)$
(4) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I \wedge v > 0)|_i \Rightarrow (\neg(m - z < s) \Rightarrow I'[0/a])[0/t]|_i)$
(5) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I' \wedge \neg(t < \epsilon))|_i \Rightarrow I|_i)$
(6) $\exists y \forall \vec{x} \forall i (i \geq y) \Rightarrow ((I' \wedge t < \epsilon)|_i \Rightarrow (I'[(t+\mathbf{dt})/t][(v+a\mathbf{dt})/v][(z+v\mathbf{dt})/z])|_i)$

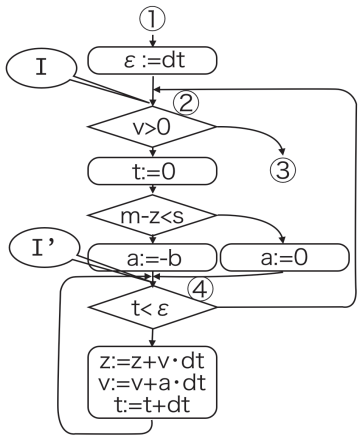


図 10　図 9 の CFG
Fig. 10　CFG for Fig. 9.

(1) は VC(①,②)，(2) は VC(②,③)，(3) と (4) は VC(②,④)，(5) は VC(④,②)，(6) は VC(④,④) を計算することで得られる制約である．

以下に示す事前条件，事後条件で検証に成功した．2 つのループ不変条件は大きな論理式であるため，直感的な意味の説明とともに付録 A.1 に示す．

$$\mathrm{Pre} = \left(v_0 \geq 0 \wedge v = v_0 \wedge z = 0 \wedge t = 0 \wedge \frac{v^2}{2b} \leq s \wedge m \geq 0 \wedge b > 0 \wedge s \geq 0 \wedge m - s \geq 0 \right)$$

$$\mathrm{Post} = \left(z \leq m + 2v_0\mathbf{dt} + \frac{b\mathbf{dt}^2}{2} \right)$$

事後条件が成り立つためには，適切な安全距離 s を設定する必要がある．$m - z < s$ のとき，電車は $v > 0$ の間等加速度 $-b$ で走行する．$v = 0$ となるまでに走行する距離は $\frac{v^2}{2b}$ であるため，事前条件は $\frac{v^2}{2b} \leq s$ という条件を含んでいる．

事後条件中の $2v_0\mathbf{dt} + \frac{b\mathbf{dt}^2}{2}$ はそれぞれの項が無限小の数と標準の数の積であるため，$2v_0\mathbf{dt} + \frac{b\mathbf{dt}^2}{2} \approx 0$ である．したがって，上記の事後条件は $(z < m \vee z \approx m)$ を表している．

制約式の計算に要した時間は 0.076 秒であった．すべての制約式に対して `Reduce/Redlog` が限量子除去に成功し，**true** という結果を出力した．限量子除去に要した時間は 1.080 秒であった．

さらに，内側のループを複数回実行される場合について検証する．図 9 の 1 行目を $e := n * \mathbf{dt}$ に変更する．検証に成功する n の値の範囲を得るため，n を自由変数として，以下に示す事前条件，事後条件を検証器に入力した．2 つのループ不変条件は付録 A.2 に示す．

$$\mathrm{Pre} = \Big(v_0 \geq 0 \wedge v = v_0 \wedge z = 0 \wedge t = 0 \wedge$$

$$\frac{v^2}{2b} + v_0 n\mathbf{dt} \leq s \wedge m \geq 0 \wedge b > 0 \wedge s \geq 0 \wedge m - s \geq 0\Big)$$
$$\mathrm{Post} = (z \leq m + 2v_0\mathbf{dt} + b\mathbf{dt})$$

$\epsilon = \mathbf{dt}$ の場合と比べて，走行処理の間に進む距離が長くなるため，安全距離 s を長くとらなければならない．具体的には，$v = v_0$，$a = 0$ で走行処理の間に進む距離は $v_0 n\mathbf{dt}$ であるため，事前条件中に $\frac{v^2}{2b} + v_0 n\mathbf{dt} \leq s$ という条件が含まれる．また，無限小による誤差が大きくなるため，事後条件に含まれる誤差も $\epsilon = \mathbf{dt}$ の場合より大きくなっている．

6 つの制約条件のうち VC(②,③)，VC(②,④) のうち $a := -b$ を通るパスから計算される制約について，Reduce/Redlog での限量子消去に失敗し，一部の限量子が残った論理式が消去された．いずれも，Reduce/Redlog が出力した限量子付き論理式の，限量子のない部分を Slfq で簡略化したところ，**true** が出力された．その他の条件は Reduce/Redlog で限量子消去に成功し，VC(①,②) から計算される制約から $n + 1 > 0$ が，それ以外からは **true** が出力された．したがって，検証に成功する n の範囲は $n > -1$ となる．$n <= 0$ は意味をなさないため，任意の n について上で与えた事前条件で実行すると事後条件が成り立つといえる．

制約式の計算に要した時間は 0.032 秒であった．限量子消去と出力の簡略化に要した時間は 6 分 32.228 秒であった．

4.3　考察

一般的に限量子除去は高価な計算であるが，本章での検証例で自由変数含まない式の限量子除去は数ミリ秒から数秒で成功する場合がほとんどであった．ただし，4.1 節で記したように，平方根を含む条件を入力すると限量子除去に失敗する場合もみられた．この場合，次数を上げるなどして等価な式を計算するなどの対処が必要となる．

モデルを記述する際に，**if** 文や **while** 文の条件として連続変化する変数についての等式が扱えないという問題がある．本章中の例では，図 7 の 5 行目の **if** 文の条件は，地面と接触したかどうかで分岐するための条件であるため，$h = 0$ となるべきである．しかし，h はループ中で無限小ずつ変化し，$h \approx 0$ となる場合はあるが厳密に $h = 0$ にならないため，**if** 文の条件を $h = 0$ とするとその分岐は決して起こらない．この問題に注意してモデル化を行う必要がある．

4.1 節，4.2 節の両例ともに，検証する事後条件として **dt** を含む式を入力した．これは連続的な値の変化をループと無限小定数を使って表現した場合に無限小の誤差が生じるためである．この誤差を含めて各条件を推測しなければならず，式が複雑になりやすい．大きなモデルを検証する場合，誤差が累積して書く条件はより複雑な式となる．この問題に対しては，無限小の値は 0 に無限に近い値であるため事後条件に含まれる無限小の値を 0 に近似することで対処できる．大きなモデルは HOARE$^{\mathrm{dt}}$ の合成性を利用して複数の小さなモデルに分割し，それぞれの事後条件中の無限小の値を 0 で近似することにより誤差の累積を防ぐことができる．

5.　おわりに

5.1　まとめ

本研究では，HOARE$^{\mathrm{dt}}$ フレームワークと限量子除去ツールを用いたハイブリッドシステムの検証手法を提案した．HOARE$^{\mathrm{dt}}$ における “ほとんどすべての i について成り立つ” を限量子付きの論理式で表現し，限量子除去法を用いて等価な限量子のない論理式に変換する．これにより HOARE$^{\mathrm{dt}}$ でのハイブリッドシステムの検証をホーア論理によるプログラム検証と同じ方法で扱うことができる．限量子除去には異なる限量子除去法を実装した複数のツールを組み合わせて使用することで速度と出力の簡潔さを両立させている．

制約解消によるプログラム検証手法を応用し，推測した事前条件，ループ不変条件が制約条件を充足しているか否かをコントロールフローグラフ上のパスごとに確認する．また，満たされていない制約に対して反例の条件を探索する方法を記した．これによってどの条件をどう修正すべきかという手がかりを得ることができ，効率の良い検証が行える．

本論文では，地面でバウンドする物体を WHILE$^{\mathrm{dt}}$ でモデル化し，その振舞いを検証した．HOARE$^{\mathrm{dt}}$ では連続的変化と離散的変化を同じ方法で扱うため，有限時間内に無限回の離散値の変化が起きる Zeno の場合であっても区別せずに正当性を検証できることを示した．**while** ループの内部に **if** 文による分岐と **while** ループを持つ複雑な WHILE$^{\mathrm{dt}}$ モデルの検証例として，ETCS を簡略化したシステムの振舞いを検証した．以上の例において，検証に使用した限量子除去ツールはおおむね短時間で限量子除去に成功し，失敗する場合でも表現を変えることで成功する場合もあることを示した．

5.2　今後の課題

限量子除去による反例の条件の探索の際に，現状では反例と関連のありそうな変数を推測し，その変数を自由変数として限量子除去を行っている．その結果発見した反例の条件から各条件の修正も手動で行っている．反例の条件を自動探索して，反例を利用した各条件の洗練を自動で行うように改良することでより効率の良い検証が可能となる．

提案手法では一度記述した WHILE$^{\mathrm{dt}}$ モデルに対してはいっさい変更を加えていない．文献 [11] で使用している文献 [21] などのプログラム変換の手法を前処理として用いて

モデルを単純化し，その後に事前条件やループ不変条件を推測するように手順を変更することにより，それぞれの条件がより単純な式となり，条件の推測が容易にする手法が考えられる．

謝辞　本研究は科学研究費補助金 23700026 および 25280023 の助成を受けた．

日頃より研究活動を支え，ときに貴重なご意見をいただきました結縁・寺内研究室の皆様に感謝いたします．

参考文献

[1] Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., Menzel, W., Mostowski, W., Roth, A., Schlager, S. and Schmitt, P.H.: The KeY tool, *Software and System Modeling*, Vol.4, No.1, pp.32–54 (2005).

[2] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J. and Yovine, S.: The Algorithmic Analysis of Hybrid Systems, Vol.138, No.1, pp.3–34 (1995).

[3] Alur, R., Henzinger, T.A. and Ho, P.-H.: Automatic Symbolic Verification of Embedded Systems, *IEEE Trans. Softw. Eng.*, Vol.22, No.3, pp.181–201 (1996).

[4] Brown, C.W.: Simple CAD Construction and its Applications, *J. Symb. Comput.*, Vol.31, No.5, pp.521–547 (2001).

[5] Brown, C.W.: QEPCAD B: A program for computing with semi-algebraic sets using CADs, *SIGSAM Bull.*, Vol.37, No.4, pp.97–108 (2003).

[6] Collins, G.E.: Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Automata Theory and Formal Languages*, pp.134–183 (1975).

[7] Davenport, J.H. and Heintz, J.: Real Quantifier Elimination is Doubly Exponential, *J. Symb. Comput.*, Vol.5, No.1/2, pp.29–35 (1988).

[8] Dolzmann, A. and Sturm, T.: REDLOG: Computer algebra meets computer logic, *SIGSAM Bull.*, Vol.31, No.2, pp.2–9 (1997).

[9] Gulwani, S., Srivastava, S. and Venkatesan, R.: Program analysis as constraint solving, *SIGPLAN Not.*, Vol.43, No.6, pp.281–292 (online), DOI: 10.1145/1379022.1375616 (2008).

[10] Gulwani, S. and Tiwari, A.: Constraint-Based Approach for Analysis of Hybrid Systems, *CAV*, pp.190–203 (2008).

[11] Hasuo, I. and Suenaga, K.: Exercises in Nonstandard Static Analysis of Hybrid Systems, *CAV*, Berlin, Heidelberg, pp.462–478 (2012).

[12] Henzinger, T.A.: The Theory of Hybrid Automata, *LICS*, pp.278–292 (1996).

[13] Henzinger, T.A., Ho, P.-H. and Wong-Toi, H.: HYTECH: A Model Checker for Hybrid Systems, *CAV*, pp.460–463 (1997).

[14] Platzer, A.: Differential Dynamic Logic for Hybrid Systems, *J. Autom. Reasoning*, Vol.41, No.2, pp.143–189 (2008).

[15] Platzer, A. and Quesel, J.-D.: KeYmaera: A Hybrid Theorem Prover for Hybrid Systems (System Description), *IJCAR*, pp.171–178 (2008).

[16] Platzer, A. and Quesel, J.-D.: European Train Control System: A Case Study in Formal Verification, *ICFEM*, pp.246–265 (2009).

[17] Sankaranarayanan, S., Sipma, H. and Manna, Z.: Constructing Invariants for Hybrid Systems, *HSCC*, pp.539–554 (2004).

[18] Sankaranarayanan, S., Sipma, H. and Manna, Z.: Nonlinear Loop Invariant Generation Using Gröbner Bases, pp.318–329 (2004).

[19] Sankaranarayanan, S., Sipma, H.B. and Manna, Z.: Constraint-Based Linear-Relations Analysis, *SAS*, pp.53–68 (2004).

[20] Schrijver, A.: *Theory of linear and integer programming*, Wiley-Interscience series in discrete mathematics and optimization, Wiley (1999).

[21] Sharma, R., Dillig, I., Dillig, T. and Aiken, A.: Simplifying Loop Invariant Generation Using Splitter Predicates, *CAV*, pp.703–719 (2011).

[22] Sturm, T. and Tiwari, A.: Verification and synthesis using real quantifier elimination, *ISSAC*, pp.329–336 (2011).

[23] Suenaga, K. and Hasuo, I.: Programming with Infinitesimals: A While-Language for Hybrid System Modeling, *ICALP* (*2*), pp.392–403 (2011).

[24] Tarski, A.: *A Decision Method for Elementary Algebra and Geometry*, Berkeley: University of California Press (1951).

[25] Weispfenning, V.: The Complexity of Linear Problems in Fields, *J. Symb. Comput.*, Vol.5, No.1/2, pp.3–27 (1988).

[26] Weispfenning, V.: Quantifier Elimination for Real Algebra — the Quadratic Case and Beyond, Vol.8, No.2, pp.85–101 (1997).

付　　録

A.1　ETCS のループ不変条件（$\epsilon = \mathbf{dt}$）

$\epsilon = \mathbf{dt}$ のとき，以下の 2 つのループ不変条件で検証が成功した．t が 0 または $\mathbf{dt}$ 以外の値をとらないことを利用している．

$$
\begin{aligned}
I \;=\; & \Bigg(\Bigg((t = 0 \wedge m - s \geq z) \\
& \vee \left(t = \epsilon \wedge m - s + v_0\mathbf{dt} \geq z \right. \\
& \quad \left. \wedge \left(m - z \leq s \Rightarrow m + v_0\mathbf{dt} \geq z + \frac{v^2}{2b}\right)\right) \\
& \vee \left(m + 2v_0\mathbf{dt} \geq z + \frac{v^2}{2b} + v\mathbf{dt}\right) \\
& \Bigg) \wedge 0 \leq t \wedge (t \leq \epsilon - \mathbf{dt} \vee t = e) \wedge v \geq -bt \wedge \\
& \frac{v_0^2}{2b} \leq s \wedge b > 0 \wedge s \geq 0 \wedge m - s \geq 0 \wedge \epsilon = \mathbf{dt} \wedge v \leq v_0\Bigg) \\
I' \;=\; & \Bigg(\Bigg((t = 0 \wedge m - s \geq z \wedge a = 0) \qquad \cdots (1) \\
& \vee \left(t = \epsilon \wedge m - s + v_0\mathbf{dt} \geq z \right. \\
& \quad \left. \wedge \left(m - z \leq s \Rightarrow m + v_0\mathbf{dt} \geq z + \frac{v^2}{2b} \wedge a = 0\right)\right) \cdots (2) \\
& \vee \left(m + 2v_0\mathbf{dt} \geq z + \frac{v^2}{2b} + v\mathbf{dt} \wedge a = -b\right) \qquad \cdots (3), (4)
\end{aligned}
$$

$$\Big)\wedge 0 \le t \wedge (t \le \epsilon - \mathbf{dt} \vee t = e) \wedge v \ge -bt \wedge$$
$$\frac{v_0^2}{2b} \le s \wedge b > 0 \wedge s \ge 0 \wedge m - s \ge 0 \wedge \epsilon = \mathbf{dt} \wedge v \le v_0\Big)$$

図 10 の④では以下の 4 つの状態をとりうる．それぞれ上記の I' の (1)〜(4) に対応する．

(1) 加速度 0 で走行処理が行われる前
(2) 加速度 0 で走行処理が行われた後
(3) 加速度 $-b$ で走行処理が行われた前
(4) 加速度 $-b$ で走行処理が行われた後

(2) のとき，走行処理の結果 $m - z < s$ となる場合がある．この場合，たかだか $v_0\mathbf{dt}$ だけ安全距離を超えて走行したこととなる．$m - z < s$ なので，次の走行処理以降は加速度が $-b$ となるため，残りの走行距離は $\frac{v^2}{2b}$ である．$m + v_0\mathbf{dt} \ge z + \frac{v^2}{2b}$ は，(目標地点 + 安全距離を超えて走行した距離) が (現在の変位 + 残りの走行距離) よりも大きいことを表している．

ただし，無限小の誤差のために実際の残りの走行距離は $\frac{v^2}{2b}$ とは完全には一致しない．そのため，(3)，(4) では $m + 2v_0\mathbf{dt} \ge z + \frac{v^2}{2b} + v\mathbf{dt}$ として目標地点を超える距離を増加させている．

A.2　ETCS のループ不変条件（$\epsilon = n\mathbf{dt}$）

$\epsilon = n\mathbf{dt}$ のとき，以下の 2 つのループ不変条件で検証が成功した．

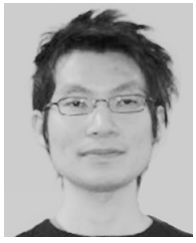
$$I = \Big((m - s \ge z - v_0 t) \vee \Big(m + 2v_0\mathbf{dt} \ge z + \frac{v^2}{2b} + v\mathbf{dt}\Big)$$
$$\wedge 0 \le t \wedge t < \epsilon + \mathbf{dt} \wedge t \ge 0 \wedge v \ge -bt \wedge \frac{v_0^2}{2b} + v_0 e \le s$$
$$\wedge b > 0 \wedge s \ge 0 \wedge m - s \ge 0 \wedge \epsilon = n\mathbf{dt} \wedge v \le v_0\Big)$$

$$I' = \Big((m - s \ge z - v_0 t \wedge a = 0) \qquad \cdots (1)'$$
$$\vee \Big(m + 2v_0\mathbf{dt} \ge z + \frac{v^2}{2b} + v\mathbf{dt} \wedge a = -b\Big)$$
$$\wedge 0 \le t \wedge t < \epsilon + \mathbf{dt} \wedge t \ge 0 \wedge v \ge -bt \wedge \frac{v_0^2}{2b} + v_0 e \le s$$
$$\wedge b > 0 \wedge s \ge 0 \wedge m - s \ge 0 \wedge \epsilon = n\mathbf{dt} \wedge v \le v_0\Big)$$

$\epsilon = \mathbf{dt}$ のときと異なり，付録 A.1 の (1) と (2) がここでは (1)′ に集約されている．$a = 0$ となるのは走行処理の開始前に $m - s \ge z$ であり，走行処理中に進んだ距離は $v_0 t$ であるため，$a = 0$ で走行中は (1)' の条件が成り立つ．$\frac{v_0^2}{2b} + v_0 e \le s$ の条件は，4.2 節に記述したとおり，安全距離が $\epsilon = \mathbf{dt}$ の場合よりも大きくなった結果の式である．

岩塚 卓弥

2013 年名古屋大学大学院博士課程（前期課程）修了．修士（情報科学）．並行システム，ハイブリッドシステムの検証に関する研究に従事．



寺内 多智弘

2006 年カリフォルニア大学バークレー校博士課程修了．2007 年東北大学大学院情報科学研究科助教．2011 年名古屋大学大学院情報科学研究科准教授，現在に至る．Ph.D. プログラミング言語，特にプログラム解析およびプログラム検証の研究に興味を持つ．

結縁 祥治　（正会員）

1990 年名古屋大学大学院博士課程満了．名古屋大学工学研究科助手，1998 年同情報メディア教育センター助教授，同大学院情報科学研究科准教授を経て，現在，同大学同研究科教授．博士（工学）．並行計算モデルに関する研究に従事．ACM，電子情報通信学会，日本ソフトウェア科学会各会員．