1B-5

ハイブリッドシステムモデリング言語HydLaの数式処理実行系

高田 賢士郎† 渋谷 俊† 細部 博史‡ 上田 和紀††

† 早稲田大学大学院 ‡ 国立情報学研究所 †† 早稲田大学理工学術院

1 はじめに

ハイブリッドシステムとは時間の経過に伴って状態が連続変化したり，状態や方程式が離散変化したりする系を指す．様々な分野のモデルを同じ枠組みによって統一的に扱えるため，その記述やシミュレーション，検証手法が研究されている [3]．ハイブリッドシステムの記述形式としてハイブリッドオートマトンやHybrid CCがあるが，複雑なモデルの制約を過不足なく記述するのは困難であったり，計算の精度が保証されていないために厳密なシミュレーションが行えない．我々はより簡潔な記述を目指したモデリング言語として，制約概念と制約階層 [1] に基づくモデリング言語 HydLa [5] を提案，開発している．現在開発中の HydLa 処理系は，高信頼なシミュレーションを行うことでシステム検証に役立てることを目標としている．計算手法の1つとして数式処理を採用しているが，簡単な例題しか扱うことができず，パルス関数のように，離散変化時刻と連続変化時刻とを厳密に扱う必要があるようなモデルは正しいシミュレーション実行ができなかった．

そこで，新たに定式化した HydLa の実行アルゴリズム [4] に基づき，数式処理によるシミュレーション実行系の改良を行った．本論文では HydLa の数式処理実行系において新しく導入した改良点について述べる．

2 HydLaとそのモデリング例

HydLa はハイブリッドシステムを記述するための，制約概念に基づいた宣言型言語である．HydLa プログラムの変数は時刻についての関数変数であり，システムが満たすべき条件（制約）を等式・不等式と微分方程式により記述することでモデリングを行う．また，制約間には優先度を設けることができ（制約階層），最低限の記述でシステムの挙動を記述できる．

例として物体が自由落下し地面で弾むモデルの HydLa プログラムを以下に示す．また，このモデルが描く軌道（解）を図 1 に示す．

```
INIT   <=> ht = 10 /\ v = 0.
FALL   <=> [](ht' = v /\ v' = -10).
BOUNCE <=> [](ht- = 0 => v = -(4/5)*v-).
INIT, (FALL << BOUNCE).
```

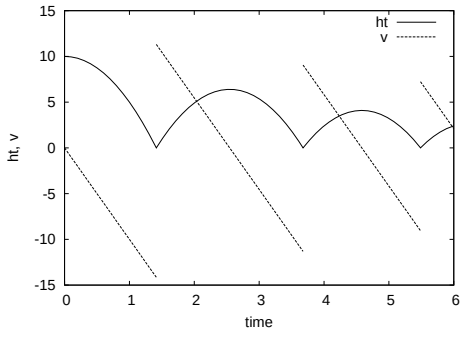

図 1: 物体が自由落下し地面で弾むモデルの実行結果

ht は物体の位置（高さ），v は速度を表す関数変数である．1–3 行目の INIT，FALL，BOUNCE は制約の定義で，INIT は時刻 0 における初期値を，FALL は連続変化（自由落下）を，BOUNCE は離散変化（床との衝突）を記述している．なお，[] は時刻 0 以降常に成り立つ制約であることを，=>は左辺を条件式（ガード条件）とする条件付き制約であることを，ht' は ht の時間微分を，ht- は ht の左極限値を表す．4 行目で制約の合成を行っている．FALL と BOUNCE について，両者が矛盾した場合は BOUNCE を優先的に採用するように合成し，さらに優先順序を付けずに INIT を合成している．

3 HydLa処理系

HydLa 処理系は，入力された HydLa プログラムをシステムが満たすべき仕様と見なし，それを満たすような解軌道を計算することでシミュレーションを行う．処理系はまず入力として受け取ったプログラムの展開を行い，次にプログラム中に与えられた制約階層を解き [2]，解候補モジュール集合の集合を求める．続いて，離散変化を扱うポイントフェーズ（PP）と，連続変化を扱うインターバルフェーズ（IP，2 つの PP を両端とする開区間）とを交互に行い，シミュレーションを進める．シミュレーション中に必要となる具体的な計算は，実際には外部のソルバを呼び出し，使用することによって行われる．HydLa 処理系内には仮想的な制約ソルバがあり，この機構により，さまざまなソルバによる計算を扱えるようになっている．

数式処理によってシミュレーションを行う数式処理実行系に対して行った改良点について述べる．ただし，扱う対象としては，[4] のアルゴリズムで扱えるもののうち，解軌道が一意に定まるようなモデルに限定している．

A Symbolic Simulator of Hybrid Systems Modeled in HydLa

†Kenshiro Takata, Shun Shibuya, Graduate School of Sci. and Engr., Waseda University

‡Hiroshi Hosobe, National Institute of Informatics

††Kazunori Ueda, Faculty of Sci. and Engr., Waseda University

4 数式処理実行系の改良

数式処理によるシミュレーション実行における主な計算は，各フェーズで成立すべき制約の連言である制約ストアを用いて行われる．制約ストアには，各フェーズで採用した制約に加え，前フェーズ終了時の各変数の値に関する制約が追加される．すなわち，PP では左極限値に関する制約，IP ではフェーズ開始時点で成立する制約（初期値制約）が含まれる．数式処理実行系に新たに改良を加えた点としては以下のようなものが挙げられる．

1. PP での左連続性制約追加 — PP において，ある変数の微分値に関する制約を採用している場合は，その変数の左連続性に関する制約を制約ストアに新たに追加する．これにより，連続な変数を正しく扱うことが可能になる．

2. IP でのガード条件判定 — 制約ストアの下でガード条件が満たされるかどうかを調べるガード条件判定 [4] については，IP ではフェーズ開始 “直後” に関しての判定を行う．これは，IP は開区間により表されるためである．

3. IP での微分方程式求解における初期値制約の選択 — IP において微分方程式を解く際は，制約ストア内にある初期値制約のうち適切なもののみを選択する必要がある．これは，PP から渡された初期値制約をすべて使って解こうとすると，モデルによっては過剰決定系扱いになってしまい，微分方程式が解けないことがあるためである．

1. と 3. については，採用している制約内に存在する各変数について最大微分回数を求め，それより小さい微分回数のものについて，制約の追加・選択を行う．

5 数式処理実行によるシミュレーション例

以上の改良を C++言語で実装し，正しいシミュレーションが行えることをパルス関数のモデルを元に確認した．パルス関数は，モデルにおいてあるイベントが一瞬だけ起きる際にそれをとらえる信号として役立つ．以下は時刻 3 で上がるパルス関数を表す HydLa プログラムであり，数式処理実行系によるシミュレーション結果を図 2 に示す．なお，数式処理のソルバとしては Mathematica を使用した．

```
INIT  <=> a = 0.
TIMER <=> [](a' = 1).
FLAT  <=> [](b = 1).
PULSE <=> [](a- = 3 => b = 2).
INIT, TIMER, (FLAT << PULSE).
```

実行結果から，時刻 3 での PP において b の値が 1 から 2 となり，続く IP において b の値は再び 1 となり，パルス関数の挙動をしていることが分かる．これは，IP でのガード条件判定において，時刻 3 の “直後” では PULSE のガード条件 a-==3 が成立しないことを正しく判定できているからである．なお，このモデルにおいて a は常に連続な変数であるので，このガード条件は左極限値を用いず a=3 と記述しても意味合いは変わらないはずであり，実際にそのように記述した場合でも正しくシミュレーションが行えることが確認できた．これは，PP において微分制約 a'=1 によって左連続性制約 a=a-が制約ストアに追加された結果，ガード条件判定が正しく行われるためである．また，IP において a は増加し続ける関数，b は 1 という定数関数とそれぞれ求まっており，適切な初期値制約を選択した上で微分方程式の求解が行われたことが分かる．

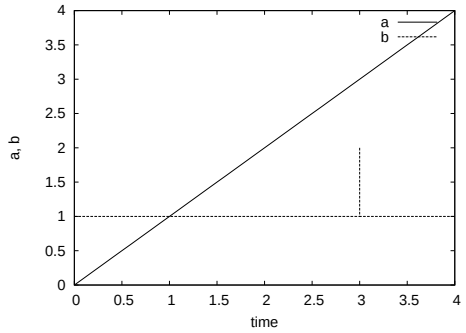

図 2: パルス関数のモデルの実行結果

6 まとめ

HydLa の数式処理実行系の実装を行った．これにより，解軌道が一意に定まるようなモデルについては数式処理に基づいた正しいシミュレーションを行うことが可能になった．今後は，複数の解軌道を持つモデルにも対応させることで，シミュレーションの幅を広げるとともに，検証分野への応用をめざす．

謝辞 本研究の一部は，科学研究費補助金（基盤研究（B）20300013）の補助を得ておこなった．

参考文献

[1] Borning, A., Freeman-Benson, B., Wilson, M. : Constraint Hierarchies. *Lisp and Symbolic Computation*, Vol. 5, No. 3, pp. 223–270, 1992.

[2] 廣瀬賢一, 大谷順司, 石井大輔, 細部博史, 上田和紀 : 制約階層によるハイブリッドシステムのモデリング手法, 日本ソフトウェア科学会第 26 回大会講演論文集, 2D-2, 2009.

[3] Lunze, J., Lamnabhi-Lagarrigue, F. : Handbook of Hybrid Systems Control: Theory, Tools, Applications. Cambridge University Press, 2009.

[4] 渋谷俊, 高田賢士郎, 細部博史, 上田和紀 : ハイブリッドシステムモデリング言語 HydLa 処理系における実行アルゴリズム, 日本ソフトウェア科学会第 27 回大会講演論文集, 1B-2, 2010.

[5] 上田和紀, 石井大輔, 細部博史 : ハイブリッド制約言語 HydLa の宣言的意味論, コンピュータソフトウェア, Vol. 28, No. 1, 2011 (掲載予定).