修 士 論 文 概 要 書

2010 年 2 月提出

専攻名	情報理工学	氏名	廣瀬 賢一	指導教員	上田 和紀 教授 印
研究指導名	並列知識情報処理	学籍番号	5108B107 – 0 CD		
研究題目	ハイブリッドシステムモデリング言語 HydLa の実行アルゴリズムの提案と実装				

1 はじめに

時間の経過に伴って状態が連続変化したり，状態や方程式自体が離散変化したりする系をハイブリッドシステムと呼ぶ．たとえば，状態を表す連続関数が時間にしたがって切り替わるモデルはハイブリッドシステムの 1 つであり，そのようなモデルは物理学，生命工学，制御工学等をはじめとする様々な分野に存在するため，その適用分野は広い．そのため，ハイブリッドシステムを容易に扱うことができる枠組みの作成が期待されている [2]．ハイブリッドシステム記述のための既存手法としてハイブリッドオートマトンや Hybrid CC などを挙げることができるが，ユーザがモデルの取りうる状態をすべて列挙し，記述をおこなわなければならないため，記述が煩雑になりがちとなる問題が存在する．そこで，現在ハイブリッドシステムモデリング言語として HydLa[3] の提案をおこなっているが，現状では HydLa には宣言的意味論が与えられただけであり，その実行手法については議論がおこなわれていなかった．

本研究ではまず，HydLa モデルに対するシミュレーション実行の為の数式処理を用いた手法の提案をおこなう．定義の呼び出しの解決をはじめとする HydLa プログラムの変換手法，制約モジュール集合間の強弱関係を表す関係グラフを構築することによる制約階層の求解手法，制約の意味を解析するための型付け，実行時におけるポイントフェーズ・インターバルフェーズといったフェーズの分割，および各フェーズにおける処理の詳細を中心に述べていく．さらに，その手法を実際に取り入れ作成をおこなったシミュレーション実行処理系について述べる．処理系は各処理を部品化することで，それぞれを任意に入れ替え可能とし，複数の処理手法が利用可能となる統合処理系として作成した．各処理の抽象・部品化をおこない，任意に入れ替え可能とした設計について述べる．最後に，HydLa を用いてモデリングをおこなうにあたって基本となる記述手法や注意すべき点，制約階層等を用いた応用的な記述手法についても考察する．

2 HydLa

HydLa は宣言型，制約ベース，制約階層概念の導入といった特徴を持つ言語であり，既存のハイブリッドシステムモデリング手法と比べ，より容易なモデリングが可能となることを目標として設計されている．詳細は文献 [3, 4, 5] を参照されたい．

3 実行アルゴリズムの提案

HydLa をシミュレーション実行する為の手法について述べる．まず，プログラム内に存在する制約定義やプログラム定義の展開をおこなうことによって，呼び出しが存在しないプログラムに変換する．次に，そのプログラムを元に 3.1 節の手法を用い解候補モジュール集合の集合を導出する事で，制約階層が存在しない HydLa プログラムの集合を作成する．そして，解候補モジュール集合ごとにシミュレーション実行をおこない，無矛盾であったモジュール集合のうち，極大元であったもの（無矛盾極大モジュール集合）の解を実行結果の解とする．無矛盾極大モジュール集合の導出手法は 3.2 節で，解候補モジュール集合に対するシミュレーション手法は 3.3 節で述べる．

3.1 解候補モジュール集合の集合の導出

解候補モジュール集合は以下に定義する関数 ms を用いて導出する．なお M は制約モジュール，MS は制約モジュール集合，X および Y は解候補モジュール集合の集合である．

$$
\begin{aligned}
ms(M) &= \{\{M\}\} \\
ms((MS_1, MS_2)) &= parallel(ms(MS_1), ms(MS_2)) \\
ms((MS_1 \ll MS_2)) &= ordered(ms(MS_1), ms(MS_2)) \\
parallel(X, Y) &= X \cup Y \cup \{x \cup y \mid x \in X, y \in Y\} \\
ordered(X, Y) &= Y \cup \{x \cup y \mid x \in X, y \in Y\}
\end{aligned}
$$

3.2 無矛盾極大モジュール集合の導出手法

無矛盾極大モジュール集合は複数存在する可能性がある．HydLa の実行においては 1 つを採用すればよいが，全解探索をおこなう場合にはすべてを導出する必要がある．

無矛盾極大モジュール集合を 1 つ導出する場合，解候補モジュール集合の集合から要素数が多い順に矛盾が存在しないものを探索すればよい．要素数の多い順に探索をおこなうことで極大が求められることは，解候補モジュール集合の集合が以下の性質を持つことからわかる．ここで，$\mathcal{MS}$ は解候補モジュール集合からなる集合であり，$size(MS)$ はモジュール集合 MS に含まれるモジュール数を表す．

$$
\begin{aligned}
&\neg(\exists MS1 \in \mathcal{MS}\ \exists MS2 \in \mathcal{MS} \\
&\quad (size(MS1) \leq size(MS2) \Rightarrow MS1 \supset MS2))
\end{aligned}
$$

一方，すべての無矛盾極大モジュール集合を導出する場合には，次の手順をおこなう必要がある．まず，解候補モジュール集合の集合に対して，集合の包含関係による依存関係のグラフの構築をおこなう．次に，グラフの各ノードに対して無矛盾性を確認し，矛盾が存在した場合はそのノードおよびそれを包含する集合であるノードを削除する．この際，任意の順にノードを探索してよい．すると，残ったグラフのノードの内，極大元であるものが無矛盾極大モジュール集合となる．むろん，極大元は複数存在する可能性がある．

3.3 解候補モジュール集合の実行手法

解候補モジュール集合のシミュレーション実行は “離散フェーズ” および “連続フェーズ” の二つのフェーズに分け，それぞれを交互に実行する事で進める．離散フェーズは離散変化を扱うフェーズであり，このフェーズで導き出された値は連続フェーズにおける ODE(Ordinary Differential Equations) の初期値となる．連続フェーズは連続変化を扱うフェーズであり，このフェーズでは含意制約の導出状態が変化するか，制約階層をまたいだ制約間の関連性によって無矛盾極大モジュール集合が変化するまで計算がおこなわれ，最終的に導き出された値は離散フェーズにおける prev 変数の値となる．なお，各フェーズの開始時に毎回，極大元となる無矛盾なモジュール集合の導出がおこなわれ，そのモジュール集合が実行される．

各フェーズでは次の簡約規則を用いて制約式の解釈をおこなう．ここで，P はプログラム，c は制約式，g は含意条件，σ は制約ストア，Δ は次のフェーズで使用するプログラムである．

$$\text{tell}: \langle c, \sigma, \Delta\rangle \rightarrow \langle \epsilon, \sigma \cup \{c\}, \Delta\rangle$$

$$\text{ask}: \frac{\sigma \vdash g}{\langle (g \texttt{=>} P), \sigma, \Delta\rangle \rightarrow \langle P, \sigma, \Delta\rangle}$$

$$\text{always}: \langle (\texttt{[]}P), \sigma, \Delta\rangle \rightarrow \langle P, \sigma, (\Delta \texttt{/\textbackslash} (\texttt{[]}P))\rangle$$

$$\text{conjunction}: \frac{\langle P_1, \sigma, \Delta\rangle \xrightarrow{*} \langle P'_1, \sigma, \Delta\rangle}{\langle (P_1 \texttt{/\textbackslash} P_2), \sigma, \Delta\rangle \rightarrow \langle (P'_1 \texttt{/\textbackslash} P_2), \sigma, \Delta\rangle}$$

4 実行処理系の実装

提案したアルゴリズムを元に，HydLa のシミュレーション実行処理系を作成した．処理系の構成図は図 1 であり，Windows および Linux においてネイティブコードとしてコンパイル可能である．作成には C++ 言語を使用し，ライブラリとして Boost C++ Libraries[*1]を利用している．また，制約求解のために Mathematica[*2] および RealPaver[1] を使用している．

開発には 3 名が関わり，処理系の全体の規模は約 20000 行である．自身の記述量は約 15000 行であり，処理系全体の設計及び，図中の色つきの部分の作成をおこなった．

処理系における根幹は構文木生成処理 (Parse Tree genarator)，制約階層処理 (Constraint Hierarchy solver)，シミュレーション実行処理 (Forward Simulator)，制約求解処理 (Virtual Constraint Solver) であり，それぞれが独立し任意に入れ替え・拡張可能となっている．そのため，Symbolic Legacy simulator のように構文木生成処理のみを利用するシミュレータを結合したり，Mathematica constraint solver や RealPaver constraint solver のように任意に求解手法を追加可能である．

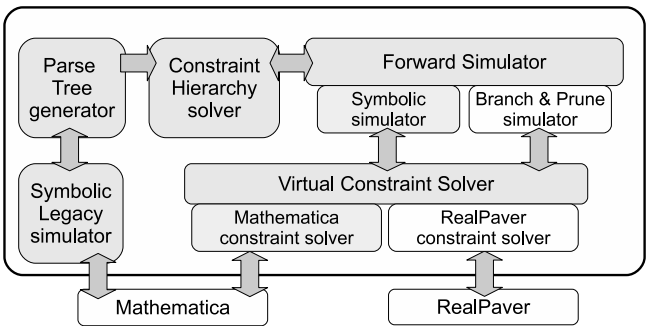

図 1 HydLa のシミュレーション実行処理系の構成図

5 まとめと今後の課題

本研究により，HydLa モデルに対するシミュレーション実行のアルゴリズムが具体化された．そして，様々な求解手法を統一的に扱うことができるシミュレーション実行処理系を作成することで，HydLa プログラムを複数の手法を使用して実際に動作させることが可能となった．

今後の課題として，まず，現状では連続変化制約と離散変化制約の矛盾性を認識できないという問題が存在する為，矛盾を検出するための手法の導入が挙げられる．また，処理系の並列化を挙げることができる．全解探索をおこなう実行の場合，分岐が多量に発生するために非常に時間がかかってしまう．分岐が起こった場合複数の状態が存在することになるが，それぞれは完全に独立したものであるため，高い並列効果を期待できる．

参考文献

[1] Granvilliers, L., Benhamou, F., Algorithm 852: RealPaver: an interval solver using constraint satisfaction techniques, *ACM Trans. Math. Software*, Vol. 32, No. 1, pp. 138–156, 2006.

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

[3] 上田和紀, 石井大輔, 細部博史, 制約概念に基づくハイブリッドシステムモデリング言語 HydLa, 第五回システム検証の科学技術シンポジウム予稿集, pp. 1–6, 2008.

[4] 大谷順司, 廣瀬賢一, 石井大輔, 上田和紀, 不確実値を持つハイブリッドシステムの高信頼なシミュレーション手法, 第 6 回ディペンダブルシステムシンポジウム論文集, pp. 145–153, 2009.

[5] 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 回大会論文集, 2D-2, 2009.

[*1] http://www.boost.org/

[*2] http://www.wolfram.co.jp/products/mathematica/index.html