2B-04

制約階層に基づくハイブリッドシステムモデリング言語HydLaの静的誤り検出手法

小山峻平 †　上田和紀 ‡

早稲田大学大学院基幹理工学研究科情報理工・情報通信専攻 †
早稲田大学理工学術院 ‡

1. 背景と目的

時間経過に伴い連続的な変化と離散的な変化を繰り返すシステムのことをハイブリッド（動的）システム [2] と呼ぶ．床を跳ねるボールや，車のブレーキ制御など様々な現象をハイブリッドシステムとして表現することが可能である．宣言型言語 HydLa[1] はこのハイブリッドシステムをシミュレーションするためのモデリング言語である．

しかし，HydLa のようなモデリング言語を用いて検証したいシステムをモデル化する際，記述されたモデルが記述者の意図を正確に反映していないという問題がしばしば起きる．本研究ではこれをモデリングエラーと呼ぶ．

HydLa は制約プログラミングパラダイムに基づくプログラミング言語であり，その制約間に優先度を設定することができる制約階層の考え方も導入している．それにより，制約の条件を記述することによる記述量の増大を抑えることができるなどのメリットがある反面，記述量が少ないことでプログラムから一見して得られる情報は多くない．そのためモデリングエラーが発生しても検出や修正は難しい．

本研究では，モデリングエラーの修正を容易にすることを目的として，静的にモデリングエラーを検出する手法を提案する．

2. 制約プログラミングとモデリングエラー

2.1 制約プログラミング

制約プログラミングとは，制約でモデルを表現するプログラミングパラダイムである．ここで述べている制約とは，解の特性を表す方程式や不等式からなる原子論理式のことであり，それらの制約を満たす解の組み合わせを計算する．手続きを記述する必要がないため，プログラムの記述が容易であることがメリットとして挙げられる．一方，手続きを記述しないためプログラムの細かい挙動がわかりにくいことがデメリットとして挙げられる．

2.2 モデリングエラー

モデリングエラーとは，システムをモデル化する際，記述されたモデルが記述者の意図を正確に反映していないという問題のことである．特に，制約プログラミングのように処理の手続きが外から見えにくいモデルでは，原因の発見が困難になるためプログラマの経験に頼る部分が多くなることや，長時間の修正作業が必要になるという問題がある．

Static error detection algorithm for hybrid system modeling language HydLa based on constraint hierarchies

†Shunpei KOYAMA, Dept. of Graduate School of Fundamental Science and Engineering, Waseda University
‡Kazunori UEDA, Dept. of Faculty of Science and Engineering, Waseda University

3. HydLa におけるモデリングエラー

3.1 HydLa

HydLa[1] とは Hybrid dynamical Language の略であり，ハイブリッドシステムをモデリングするための宣言型言語のことである．

図 1 に HydLa によるプログラムの例を示す．示すプログラムは，床でバウンドする質点のモデルである．

```
INIT <=> y=10 & y'=0 .
MOVE <=> [](y''=-1).
BOUNCE <=> [](y-=0 => y' = -y'- ).

INIT, MOVE << BOUNCE.
```

図 1: 床で跳ねる質点のモデル

このプログラムは，(INIT) y の初期値 10，初速度 0，(MOVE) y の加速度 -1，(BOUNCE) y が接地した時 ($y- = 0$)，y の速度が反転する ($y' = -y'-$) という意味を持つ制約モジュールの組み合わせから成る．この意味に従い，時間を進めることでシミュレーションを行い解軌道を得ることを目的としている．

HydLa は対象ユーザを，プログラミングを専門としない技術者としている [3]．なので，数学や論理学を理解していれば記述にあたって新たに習得すべき必要のある記法や概念を大幅に減らすことができ，簡潔なモデリングが可能であるように設計されている．

3.2 モデリングエラー

HydLa プログラムは，前件 (ask 制約) と後件 (tell 制約) の組み合わせである制約モジュールの間に優先度による階層構造 (制約階層) を与えることで構成される．図 1 の例では，制約モジュール BOUNCE の前件が ($y- = 0$)，後件が ($y' = -y'-$) である．前件が満たされた場合のみ，後件の制約が有効になる．それぞれ有効になっている後件が矛盾する場合，制約階層に従い，極大無矛盾になるように制約モジュールを無視することでシミュレーションを行っている．このシミュレーション実行時に，シミュレーションが進まないことや，値が定まらないなど，プログラマの意図と異なる解軌道が得られることがある．その中でも，制約プログラミングでシステムをモデル化する際に生ずるもののことをモデリングエラーと呼ぶ．

本研究では HydLa のモデリングエラーを，制約過多と制約不足に分けて扱う．制約式が矛盾しているのにもかかわらず，それらの制約間に優先度が付いていない場合を制約過多，各変数に対して適当な制約式が存在しない場合を制約不足とする．

3.3 モデリングエラーの例

次にモデリングエラーの例を具体例を元に示す．

図 2 に図 1 にて示した床を跳ねるボールのモデルを変更して，氷の張った湖面で跳ねる質点のモデルを示す．図 1 のモデルに，x 軸の概念を追加，さらに，$4 < x < 6$ の範囲は氷が溶けており水面が見えているとして制約 WATER を追加した．

```
INIT <=> y = 10 & y' = 0 & x = 0 & x' = 1.
FALL <=> [](y'' = -10).
MOVE <=> [](x'' = 0).
FLOOR <=> [](y-=0 => y' = -4/5 * y'-).
WATER <=> [](y-=0 & 4<x<6
         => y' = 4/5 * y'- & x' = 1/2 * x'-).

INIT, (FALL, MOVE) << FLOOR << WATER.
```

図 2: 氷の張った湖面で跳ねる質点のモデル

HydLa では制約として使用する論理式の組み合わせ方や制約階層の与え方について自由度が高いため，同じモデルに対して多様な記述が可能である．そのため図 2 のプログラムにはモデリングエラーが存在しないが，モデリングエラーを引き起こしてしまうモデルも作成される可能性がある．例えば，質点のデフォルトの挙動を示す制約モジュール FALL と MOVE を一つの制約モジュールとして記述してしまうと制約不足のモデリングエラーを含む図 3 のプログラムが作成される．

```
INIT <=> y = 10 & y' = 0 & x = 0 & x' = 1.
FALL_MOVE <=> [](y'' = -10 & x'' = 0).
FLOOR <=> [](y-=0 => y' = -4/5 * y'-).
WATER <=> [](y-=0 & 4<x<6
         => y' = 4/5 * y'- & x' = 1/2 * x'-).

INIT, FALL_MOVE << FLOOR << WATER.
```

図 3: エラーを含む，氷の張った湖面で跳ねる質点のモデル

図 3 のモデルでは，$x \leq 4$ の時と $6 \leq x$ の時に制約モジュール FLOOR のみが有効になり制約モジュール FALL_MOVE が無視されることで変数 x に対しての制約式が存在しない．それにより，制約不足のモデリングエラーが発生する．

このようにプログラム中におけるエラーの詳細な原因に関して，手続き型言語では研究があるが HydLa のような宣言型言語では研究が進んでいない．

4. 検出手法と検出例

4.1 検出手法

これまでに示したモデリングエラーを検出するための手法を考案し，HydLa 言語処理系 HyLaGI 上に実装した．提案手法は大きく 3 つの手順から構成される．制約階層を考慮した上で HydLa プログラムが取る状態を過不足なく検証し，プログラム内で発生しないモデリングエラーを誤検出しないために以下の手法をとる．

実装アルゴリズムは図 4 のようになる．

MakeCandidateConstraintSets
同時に前件を満たす制約モジュールの組み合わせを求める．この制約モジュールの組み合わせのことを候補制約集合とする．ここで，前件が同時に満たされない (例，$x = 1, x = 2$) 制約モジュールの集合は検出対象から外すことができる．

SolveCandidateConstraintSets
求めた候補制約集合ごとに後件も含めて解を求め，それを候補制約集合解とする．ここで，特殊な解 (例，$x = -x$ の制約において解が $x = 0$ の場合) を含む場合とそれ以外に分割することができる．

GetModelingErrors
得られた候補制約集合と候補制約集合解より，候補制約集合解に解がない場合を制約過多，解が不定となる場合を制約不足とする．その場合の制約集合，変数の値を求め，提示する．

Input: HydLa プログラム HydLaProgram
Output: モデリングエラーの内容 ME
1: $CCS := MakeCandidateConstraintSets(HydLaProgram)$
2: $CCS_{tell} := SolveCandidateConstraintSets(CCS)$
3: $ME := GetModelingErrors(CCS, CCS_{tell})$

図 4: モデリングエラー検出アルゴリズム

4.2 検出例

図 3 のモデルに対して，提案手法を実行すると検出できるモデリングエラーは図 5 のようになる．

```
CCS → FLOOR.
prevVAL → y=0, x>=6 or 4>=x.
ME → Under Constrint.
x is undefined.
```

図 5: 図 3 に対するモデリングエラー検出例

この検出結果より，モデリングエラーが発生する時の各変数の値と，制約モジュール集合が得られるため，容易にプログラムの修正を行うことができるようになる．

5. まとめ

本論文では，HydLa のモデリングエラーを HydLa プログラムから静的に検出する手法を提案した．本論文に提案した手法は，シミュレーション実行を介さずにエラーの原因を探すことが可能であり，これによりモデル化の手間を大きく削減することが期待される．実行時にエラーが出力されてもエラーの原因がよくわからなかったプログラムに対して，エラーの原因を探る一つの方策になると思われる．

参考文献

[1] 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, pp. 306–311.

[2] J. Lunze : 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, SSV2008(第 5 回システム検証の科学技術シンポジウム), 2008.