B-022

文字列一致による数学的等価性判定可能なモデル分割アルゴリズム

A Model Slicing Algorithm for Mathematical Equivalence Check by String Matching

三鍋孝介 Kousuke Minabe　織田 健 Takeshi Oda

1 はじめに

ソフトウェア部品を用いた開発手法は信頼性の確保の手段として以前から提案されているが、部品の検索や整備のコストに問題がある。我々は形式手法を用いて部品に仕様を付加することで健全性の高い検索を可能とし、既存ソフトウェアの分解で整備の手間を減らす手法を提案した [1]。本稿では部品の仕様をソフトウェアモデルの分解により得るアルゴリズムについて述べる。

2 背景・目的

2.1 B Method を応用したソフトウェア部品

形式手法 B Method は要求を抽象的なモデルとして記述してその無矛盾性を証明し、そのモデルと整合性のとれた実装を作ることで高い信頼性を保証している。我々はこれを応用し、ソフトウェア部品を B Method のモデルと実装の組とした。このような部品は整備にコストがかかるが、モデルと実装のどちらも形式的に書かれているため、既存のモデルと実装の組を計算機に分割させて部品を生成できる。このソフトウェア部品を使用することにより、必要な機能を同様にモデルすれば、定理証明を用いて健全性の高い検索が可能である。また、部品検索用のモデルは要求のモデルを、部品分割と同様の手段で分割すればよく、部品検索の大部分を自動化できる。そのため部品の粒度を非常に細くし、汎用性の高い部品にすることができる。ただし、大量の部品に対して定理証明を行う場合その計算時間は膨大なものとなってしまう。そこで、仕様や要求に対して構文書き換えを行い、おなじ機能はすべて同じ字面になるようにした上で、テキストの全文一致検索を行う。これにより健全性を保ったまま効率的な検索が可能となる。

2.2 モデル分割の課題

モデルは実装の動作を表す操作と操作に制約を与える制約条件からなる。モデルの分割は操作を一定の粒度に分解し、その操作に関わる制約条件を抽出すれば良い。このとき、抽出する制約条件は信頼性の保証のために操作で使われている変数のみが使われている式だけにする。しかし、制約条件には明示されていないが推論によって成り立つ条件（暗黙の条件）が存在するため、これらの条件を抽出しなければ、元のモデルで意図した操作とは異なった振る舞いをすることになる。そのため、予めこれら暗黙の条件を推論して書き出しておく必要がある。

2.3 目標

本稿ではソフトウェアのモデルを分解して以下の条件を満す部品モデルを得るアルゴリズムを目標とする。

1. 細分化したモデルは無矛盾である
2. 数学的に等価なモデルは可能な限り同じ字面である

3 モデル分割アルゴリズム

3.1 モデル分割の流れ

2.2 節で述べたようにモデル分割は暗黙の条件の書き出し・操作の分割・制約条件の抽出の流れで行われる。一方、字面の統一は項書換えを用いて演算子を基本的な演算子の集合（プリミティブな演算子）だけの式に書き換えるのと、変数名のつけ替えで行う。このプリミティブな演算子だけの式に書き換える手順をプリミティブ化と呼ぶ。暗黙の条件を推論する前にプリミティブ化を行えばその他の演算子を使った式を推論しなくてよい分推論に必要なルールを減らすことができる。以上より、モデル分割の流れはプリミティブ化、推論、操作分割、制約条件抽出、項の並び替え、変数名の付替えの順に行う。

3.2 プリミティブ化

プリミティブ化では予め定めたプリミティブな演算子で式が構成されるように項を書き換える。演算子を表現するのに必要な最低限の演算子をプリミティブな演算子とする。具体的には演算子の定義に使われている基本的な演算子をプリミティブな演算子とみなす。また、$\geq$ や $\leq$ のように相互に表現することが可能な演算子は一方をプリミティブな演算子とする。プリミティブ化の項書き換えルールは演算子の定義をそのまま用いる。また、プリミティブ化を行うと式が二重否定や括弧の入れ子になるため、これらを解消するルールも加える。プリミティブ化は条件式群にプリミティブ化ルールを適用し、変化しなくなるまで繰り返し行う。プリミティブ化のルールは演算子の定義を用いているため、1 つの演算子に対する項書き換えルールは高々1 つなので必ず停止する。また、二重否定や入れ子構造の解消のルールも式が短くなる方向の書換であり、演算子は増えないため必ず停止する。

3.3 推論

推論では 2 つ以上の式から推論される式を推論元となった式とともに並記していく。推論元の式の集合を R, 推論した式の集合を E として以下の手順で推論を行う。

1. R から全ての 2 式の組み合わせに対して推論ルールを適用した式を E に加える
2. R と E からそれぞれ 1 式ずつ全ての組み合わせに対して推論ルールを適用した式の集合を E' とする
3. E と E' が等しくなるまで、R に E を加え新しい R にし E' を新しい E として 2 を繰り返し行う

推論ルールによっては上記の手順は停止しないため、操作に含まれる変数の数 $\times 100$ より制約条件が多くなった段階で上記手順 3 の繰り返しを行わないようにする。

3.4 操作分割・制約条件抽出

操作分割の粒度は 1 代入文を基本とする。条件分岐等が含まれる場合は、その分岐条件を事前条件に加えて、

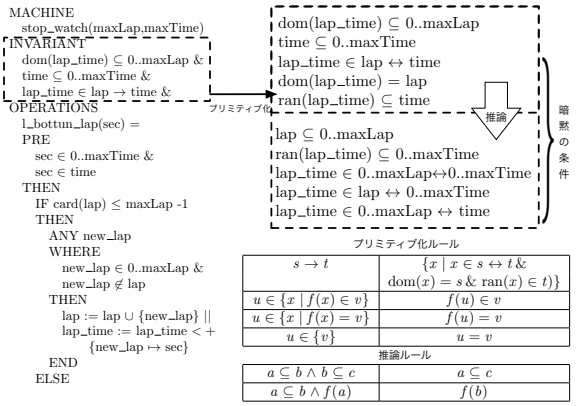

図 1: ストップウォッチモデルの制約条件書き出し

その条件毎に部品を生成する。また、排他的でない条件分岐に関しては分割を行わずに 1 操作とする。そののち分割した操作に含まれている変数に注目し、制約条件の中からこの変数だけで構成されている式を抽出する。

3.5 項の並び替え・変数名の付替え

項の並び替えは式を構文木に直した時に浅い方が左辺になるように並べ替える。各演算子の重さは予め定めておくが、変数の重みは相対的に登場順の早いものから軽い重さをつける。手順は以下のようになる。

1. 変数の重さを全て同じ重さとして並び替えを行う
2. 登場順に変数に重さをつける。どちらが先か決定できないときは同じ重さとしておく
3. 変数の登場順が一意に定まるまで手順 2 繰り返す

変数の登場順が一意に定められないまま変化しなくなることがあるが、この時はそれらの変数はどちらを前後にしても部品の字面には影響がないため、適当に決めることが可能である。項の並び替えが終わったら、このときに決まった変数の登場順を使って規則的に変数名を書き換える。またこの時、定数名も登場順に規則的に書き換え、モデルの名前や操作名等を適当な名前に書き換える。

4 実験

3 章の手順を実際のモデルに適用して無矛盾なモデルができることと、同等の操作が同じ部品に細分化されることを確認する。図 1 はプリミティブ化と推論で暗黙の条件を得る例である。図 1 左のモデルはストップウォッチのモデルの抜粋である。このストップウォッチモデルはラップタイムを図る機能を示している。図 1 右下の表はそれぞれプリミティブ化と推論のルールの抜粋である。B Method で定義されている写像の定義や単一の要素の集合の性質等を抜粋した [2]。図中の不変条件 (INVARIANT) lap_time∈lap→time に複数回プリミティブ化のルールを適用すると 3 つの式になり、増えた式の推移律や項の代入により 5 つの式が推論できる。結果 3 つの式から 10 の制約条件が得られた。このモデルの操作 l_bottun_lap はストップウォッチ左のボタンを押した時のラップタイムの記録の操作である。このうち lap := lap∪{new_lap} に注目して操作を分割し変数名を付替えると図 2 のモデルが得られる。注目した操作に使われている変数は lap, new_lap の 2 つだったためこれらだけが使われている制約条件を探した時、先ほど

```
MACHINE
    P(I001)
INVARIANT
    v001 ⊆ 0..I001 &
OPERATIONS
    addPerson(v002) =
    PRE
        card(v001) ≤ I001 - 1 &
        v002 ∈ 0..I001 &
        v002 ∉ v001
    THEN
        v001 := v001 ∪ {v002}
```

図 2: 細分化モデル (抜粋)

```
MACHINE
    AddressBook(maxPersons)
INVARIANT
    persons ⊆ 0..maxPersons &
    names : persons → seq(0..255) &
OPERATIONS
    nid ← addPerson(name) =
    PRE
        name ∈ seq(0..255) &
        card(persons) leq maxPersons - 1 &
    THEN
        ANY newId
        WHERE
            newId ∈ 0..maxPersons &
            newId ∉ persons
        THEN
            persons := persons ∪ {newId} ||
            names := names < + {newId ↦ name} ||
            nid := newId
        END
    END;
```

図 3: アドレス帳のモデル (抜粋)

の展開した制約条件の lap⊆0. .maxLap が抽出されている。変数名が付替えられているが等価な制約条件 v001 ⊆0. .I001 を図 2 に見ることができる。図 2 のモデルは定理証明器で無矛盾であることが証明できた。

次に数学的に同じ操作が同じ字面になるかを調べる。図 3 はアドレス帳のモデルである。図 1 のラップタイム更新の操作とはアドレス帳が一杯のときには更新作業が行えなかったり、保存しているデータがシーケンスであったりと異なる。しかし新規 ID の追加登録の操作 (persons := persons ∪ {newId}) と、ラップタイムが最大数に達していないときのラップ番号追加 (lap := lap∪{new_lap}) の操作が共通している。この操作に本アルゴリズムを適用すると図 2 と同じモデルが得られる。

5 まとめ

本稿では形式手法を用いたソフトウェア合成を目的としたモデル分割のアルゴリズムを提案した。本アルゴリズムにより異なる 2 つのモデルから同じ操作を同じ字面で得ることを確認した。しかし、同じ字面にできるかどうかは推論のルール次第であり、同じ字面にできる表現の範囲も明示できていない。今後は推論ルールの整備や同じ字面にできる表現の範囲の明示が課題である。

参考文献

[1] 中村丈洋, 織田健. B method における自動コード合成フレームワークの提案. 情報処理学会研究報告. ソフトウェア工学研究会報告, Vol. 2010, No. 18, pp. 1–8, 2010.

[2] Jean-Raymond Abrial, Jean-Raymond Abrial, and A Hoare. *The B-book: assigning programs to meanings*. Cambridge University Press, 2005.