B-007

条件分岐と同時代入間の相互依存に配慮した形式仕様細分化アルゴリズム

Formal Specification Slicing Algorithm Considering Conditionals and Mutual Dependence between Simultaneous Substitutions

檜垣 廉 **Ren Higaki**　　**織田 健** **Takeshi Oda**

1 はじめに

我々は、形式手法 B Method 信頼性保証の枠組みを用いた部品再利用による高信頼ソフトウェア合成手法である MSSS 手法を提案している [1]。この手法では、既存のソフトウェアからの部品生成や部品の検索キーの作成のために、形式仕様を細分化する。先行研究では、形式仕様中の IF 文の分割と相互に依存する代入文群を残した同時代入文の分割による細分化が提案されていたが、これらを同時に行う具体的なアルゴリズムが示されていなかった [2]。本稿では、これらに同時に対応した代入文の分割による形式仕様の細分化アルゴリズムを提案する。

2 背景と目的

2.1 形式手法 B Method

B Method は集合論と一階述語論理に基づいた形式手法の一種である。抽象的な仕様である「モデル」から具体的なアルゴリズムやデータ構造を記述し、一般的なプログラミング言語のコード生成が可能な「実装」までの段階的詳細化を、各段階での無矛盾性と詳細化の整合性を定理証明器を用いて証明しながら行うことで、ソフトウェアの信頼性を保証することができる [3]。モデルや実装は、代入文や条件式などの構造を持った「B 言語」で記述され、状態を変数群として持ち、代入文を含む「操作」群によって状態遷移を表現する。モデルでは 1 つの操作内の状態遷移は同時に起こるため「同時代入文」の構文を、実装では「逐次代入文」の構文を用いる。

2.2 MSSS 手法

MSSS 手法は、B Method の信頼性保証の枠組みを利用し要求を記述したモデルからのソフトウェアの合成と高信頼部品の生成を行う手法である [1]。既存のソフトウェア（証明済みのモデルと対応する実装の組）に対し、モデルを細分化し対応する実装の各部分を抽出して組にすることで部品を生成する。新規ソフトウェアの開発時は、同様に細分化された要求のモデルをキーとして必要な部品を検索し、得られた部品の細分化実装を結合することによって要求モデルに整合する実装を得る。

2.3 モデル細分化と操作分割

モデル細分化では、原則 1 代入文単位に操作を分割したのち、その代入文と関連するモデル全体の各要素を抽出して 1 つの細分化モデルとする [4]。本手法では B 言語の多様な代入文のうち、内部に代入文を持つ IF 文と同時代入文を分割の対象とする。ここで、同時代入文内に 2 つの代入文 S_i, S_j があり、S_i が参照している変数が S_j で変更されているとき、「S_i は S_j に依存する」ということとする。さらに、依存関係を S_i から辿ったとき再び S_i に戻ってくる場合は、辿った軌跡上の代入文は全て S_i と「相互に依存している」とする。高橋は相互に依存する同時代入文内の代入文は分割しないことを提案している [2]。しかし、これら IF 文の分割と、同時代入文の分割を同時に行うアルゴリズムは提案されていない。

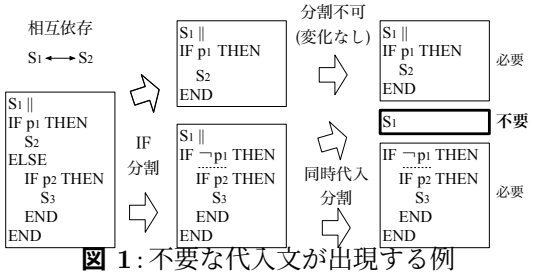

図 1: 不要な代入文が出現する例

2.4 研究の目的

MSSS 手法には多くの改善が提案されているが、手法や大まかなアルゴリズムの提案にとどまり、実装が進んでいない。本研究では、条件分岐と同時代入間の相互依存に配慮したモデル細分化の詳細なアルゴリズムの提案・決定と実装を行い、その妥当性を検討する。

3 手法統合の問題点

3.1 不要な代入文の出現

愚直に IF 文と同時代入文を再帰的に分割した例を図 1 に示す。記号 || は両側の代入文が同時に行われることを表し、1 つの IF 文は IF から END までである。ここでは、代入文 S_1 と S_2 が互いに依存しており同じ細分化モデルに入るべきであるが、S_1 が独立した代入文として分割されてしまう。最初の IF 分割で下側の代入文群に S_1 を含めたのはこの階層では S_1 が切り離せるかが不明であるためであり、これを防ぐには S_1 以外の部分の依存関係や構造を完全に把握しておく必要がある。

3.2 同一の変数への同時代入の出現

上記の課題を解決するため、最初に依存関係を考慮せず全ての IF 文と同時代入文を分割し、後から相互に依存するもののみを同時代入文として結合する方法が考えられる。しかし、この方法では B 言語で禁止されている同一の変数に対して同時代入文中の複数の代入文で変更を加えるような代入文が出現することがある。

4 条件分岐と同時代入間の相互依存に配慮した代入文分割アルゴリズム

4.1 全体の流れ

代入文分割アルゴリズムの概要を示す。

1. 各代入文について、代入文本体と実行されるための条件を持ったデータ構造「ブランチ」を生成する。

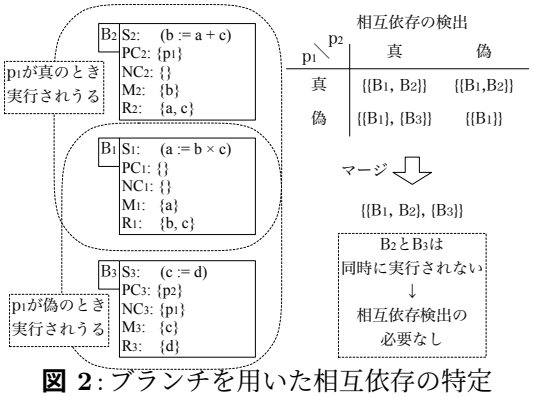

図 2: ブランチを用いた相互依存の特定

2. IF 文に登場する各条件に対する真偽値の割り当てのパターンを全て生成する。
3. 各条件パターン下で実行される代入文を含むブランチを特定し、変更する変数と参照する変数からブランチの相互依存を検出する。
4. 全ての条件パターン下での相互依存関係を推移的にマージし、全体でのブランチの相互依存を特定する。
5. IF 文の入れ子と skip 代入文を用いて相互依存ブランチを結合して代入文にする。
6. 規則を用いて代入文の構造を簡単化する。

4.2 ブランチの定義

代入文 S_i に対応するブランチを B_i とすると、$B_i = (S_i, PC_i, NC_i, M_i, R_i)$ と表せる。ただし、PC_i は IF 文の条件のうち S_i が実行されるための必要条件の集合、NC_i は ELSE 等により否定されている条件の集合、M_i は S_i で変更される変数の集合、R_i は S_i, PC_i, NC_i で参照される変数の集合である。

4.3 ブランチの生成

ブランチの生成は、そこに制御が到達するための各条件式に対する真偽の割り当て方を計算しながら IF 文内を深さ優先で下降し、分割不可能な代入文に行き当たる度に上記のデータ構造を生成することで行う。ここでは同時代入文の相互依存は考慮せず全て分割する。

4.4 条件式に対する真偽割り当てパターンを用いた相互依存代入文の特定

図 2 に、ブランチを用いた相互依存の特定の例を示す。IF 文中に出現した条件式に対し真偽値の割り当ての全てのパターンを生成した後、まずそれぞれのパターン下で実行される代入文を含むブランチを特定する。ある条件 p が真のとき S_i が実行されるための必要条件は、$p \notin NC_i$ であり、同様に p が偽のときは $p \notin PC_i$ である。図 2 では、例えば p_1, p_2 がともに真のとき実行されうるのは S_1, S_2 である。したがって右上の表のように、この割り当ての下では B_1, B_2 が相互依存するかを判定する。その後、各割り当てのパターン下での相互依存関係をもとに、全体でのブランチの相互依存を特定する。例えば、代入文 S_i と S_j が真偽パターン TF_1 下で相互依存し、S_j と S_k が真偽パターン TF_2 下で相互依存するとき、B_i, B_j, B_k は相互依存するグループとして扱う。

4.5 ブランチの結合と簡単化

B_i が独立しているとき、PC_i の要素と NC_i の各要素の否定の論理積をとって IF 文の条件とすると B_i を代入文に変換できる。B_i が B_j と相互に依存している場合は、PC_i, NC_i, PC_j, NC_j に含まれる条件に対する全ての真偽パターンを網羅する入れ子の IF 文を生成し、入れ子の最下層に S_i と S_j をそれぞれ実行されるべき条件の箇所に挿入する。代入文が挿入されない場所には、「なにもしない」ことを表す「skip」を挿入する。その後 8 個の書き換え規則を用いて IF 文の入れ子と skip を削除する。規則の一部を抜粋し、以下に示す。

IF c THEN $s1$ || $s2$ ELSE $s1$ || $s3$ END

$\longrightarrow$ $s1$ || IF c THEN $s2$ ELSE $s3$ END

IF c THEN s ELSE skip END

$\longrightarrow$ IF c THEN s END

skip || $s \longrightarrow s$

4.6 実装

提案したアルゴリズムを約 3100 行の Standard ML で実装した。4 階層までの IF 文と同時代入文についてテストし条件を考慮した分割ができていることを確認した。

5 考察

本手法によって、相互に依存する同時代入を分離せずに条件分岐を分割することが可能になったと考えられる。各真偽パターン下における相互依存を特定した後、全体での相互依存を特定せずに真偽パターンを条件式に反映した IF 文に分割する方法も考えられたが、もともと IF 文の外にあった代入文が特定の条件下で実行されるように変換されてしまい、部品生成時に実装の対応する部分を抽出することが難しくなる可能性があるため、このような手法となった。真偽パターンの列挙や IF 文の入れ子による代入文の結合では、条件数に対して計算量が指数オーダーとなるため、IF 文の条件数が 10 個程度の複雑なモデルが入力された場合には実用的な時間での実行ができない場合があると考えられ、対策が必要である。

6 終わりに

本研究では、条件分岐と同時代入間の相互依存に配慮した形式仕様細分化アルゴリズムの提案と実装を行った。今後は計算量の改善と MSSS 手法について過去に提案された別の改善案との統合を行い、全体の実装を進めたい。

参考文献

[1] 中村丈洋. B Method における部品再利用によるソフトウェア合成と高信頼ソフトウェア部品の整備. 電気通信大学 電気通信学研究科 博士 (工学) 学位論文. 2013.

[2] 高橋宏夢, 織田健. 形式手法 B Method の細粒度部品の結合による高信頼ソフトウェアの合成. 第 17 回情報科学フォーラム論文集 vol.1 pp.137-138. 2018.

[3] 来間啓伸. B メソッドによる形式仕様記述. 近代科学社, 2007.

[4] 三鍋孝介, 織田健. 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. 第 12 回情報科学技術フォーラム論文集 vol.1 pp.271-272. 2013.