1L-4

# モデルと実装間の整合性に配慮した形式的ソフトウェア分割手法

三鍋 孝介†
電気通信大学電気通信学部情報通信工学科

織田 健‡
電気通信大学情報理工学研究科総合情報学専攻

## 1 はじめに

近年ではライフラインの多くにソフトウェアが組み込まれており, 誤りの少ないソフトウェアの開発手法が求められている. ソフトウェア部品再利用は信頼性向上に有効だが大きな部品は汎用性が低く再利用に適さず, 細かな部品は数が多くなり検索や選択に手間がかかるという問題点がある. そこで我々は B Method の実装と仕様間の整合性保証に注目し, 細かな部品に形式的に記述された仕様を付加することで部品を検索し, ソフトウェアを合成する MSSS を提案した [1]. 本稿では MSSS の一部であるモデル細分化のアルゴリズムを提案する.

## 2 背景

### 2.1 形式手法 B Method

形式手法とは数学を基盤とした正確な仕様の記述や検証の技術である. その一つである B Method は抽象的なモデル (仕様) を段階的に詳細化してアルゴリズムが把握できるインプリメンテーション (実装) を記述することで, 実行可能なプログラムを生成することができる. また, 各段階の詳細化が正しいかどうかを数学的に証明することができる.

### 2.2 モデル充足ソフトウェア合成手法

我々が提案したモデル充足ソフトウェア合成手法 (以下 MSSS) は B Method で記述された矛盾のない要求仕様から実装を合成する. MSSS では要求仕様を小さな仕様に分解し, その仕様をキーにしてリポジトリ内の部品から同じ仕様を持つ部品を検索する. 最後に元の仕様に沿って部品を合成し実装を得る. また, リポジトリは B Method の記述で書かれたソフトウェアを分解することで仕様と実装をもった部品を大量に用意して整備する. この手法の仕様と実装は数学的な記述なので, 部品の検索や生成, ソフトウェアの合成は計算機で容易に扱える. よって MSSS は以下の性質を持つ.

- 部品のリポジトリが充実していれば仕様から実装を得ることが可能である.
- 人が行う作業は仕様の記述, 部品の選択や不足部品の追加のみで人的コストが少ない.
- B Method のソフトウェアプロダクトから部品を自動的に生成でき, リポジトリの整備が容易である.
- B Method のツールを用いることで部品自体の信頼性の保証が容易である.

本稿ではモデル (仕様) を分解した細分化モデルを得る具体的なアルゴリズムを提案する.

A Formal software slicing method keep consistency between a model and an implementation

†Kousuke Minabe, The University of Electro-Communications Dept. of Infomation and Communication Engineering

‡Takeshi Oda, The University of Electro-Communications Graduate School of Informatics

### 2.3 細分化モデル

モデルは代入文で書かれた操作と, 定数や変数の宣言や変数の制約を宣言する制約条件で記述される. 細分化モデルはこれを分解し, 一代入文の操作とその操作に関わる制約条件で成り立つ. 細分化モデルは MSSS において, 部品の検索やソフトウェアを分解して部品を作る際に利用する.

## 3 モデル細分化アルゴリズム

モデル細分化アルゴリズムはモデルを細分化モデルに分解する. MSSS では制約条件展開, 操作分割, 制約条件抽出の三つを行いモデルを分割する. 操作分割ではモデルの操作を一操作毎に分割する. 分割した操作に必要な制約条件を抽出して細分化モデルを生成する. 制約条件には推論にて成り立つ暗黙的な関係 (暗黙の条件) が存在するため, 操作の意味を変えないためにこれらも抽出する必要がある. そこであらかじめ制約条件展開でモデルの暗黙の条件を書き出す. 本稿では制約条件展開をさらに制約条件部分を式の定義に従って基本的な表現に書き換えるプリミティブ化と, 推移律から暗黙な条件を書き出す推移律の書き出しの二つに分ける. この他に重複する項の削除などを行う簡約化を適宜行う. 本アルゴリズムでは式の書き換えをしやすくするために, 式は全て演算子を節に, 定数や変数を葉とする構文木に変換している. 文章では式をポーランド記法で表す. それぞれのアルゴリズムは全て関数になっており, ある作業 X を式 A に対して行う場合は X(A) と表す.

### 3.1 プリミティブ化

プリミティブ化ではモデルの制約条件を B Method の式定義に従ってプリミティブな表現に変換する. 従来の制約条件展開では推論ルールを用意しそれを複数回適用する事で暗黙の条件を書き出すとした. しかし, 展開した式も暗黙の条件を持っており, いつ全ての条件を展開し切れるかわからなかった. そこで本稿では暗黙の条件を加えた条件式も元の式も数学的には同じ意味であることに注目し, 条件式をプリミティブな表現に変換することで一つの式に存在する暗黙の条件を明示する. 以下の手順を式が変形しなくなるまで適用する.

- 式が変数や定数であればそのまま出力する.
- 演算子に注目し定義によって書き換えが可能であれば書き換えて出力する.
- 演算子に注目し書き換えが不可能であれば (演算子プリミティブ化 (左辺) プリミティブ化 (右辺) ) を出力する.

例として集合 X は集合 a から集合 b への全域単射の集合 '$X : a \rightarrowtail b$' をプリミティブ化する. 表 1 の式定義に $X \in a \rightarrowtail b$ の演算子 $\rightarrowtail$ は存在するので定義にしたがって変換し $X \in (a \nrightarrow b \bigcap a \rightarrow b)$ となる. 繰り返し変換を行うと最終的に以下のようになる.

$X \in (\ \{f \mid f \in \{r \mid r \in a \leftrightarrow b \wedge dom(r) \subseteq a \wedge$

| 式 | 定義 |
|---|---|
| $s \nrightarrow t$ | $\{r \mid r \in s \leftrightarrow t \wedge dom(r) \subseteq s \wedge ran(r) \subseteq t\}$ |
| $s \rightarrow t$ | $\{f \mid f \in s \nrightarrow t \wedge dom(f) = s\}$ |
| $s \rightarrowtail t$ | $\{f \mid f \in s \nrightarrow t \wedge f^{-1} \in t \nrightarrow s\}$ |
| $s \rightarrowtail t$ | $s \rightarrowtail t \bigcap s \rightarrow t$ |
| $ran(r)$ | $if\ r \in s \leftrightarrow t\ then\ dom(r^{-1})$ |

**表 1**: 式定義 (抜粋)

| 簡約化前 | 簡約化後 |
|---|---|
| $A \wedge A$ | $A$ |
| $\{x \mid x \in \{y \mid y \text{ の式 }\} \cdots\}$ | $\{x \mid x(\text{元 } y) \text{ の式 } \wedge \cdots\}$ |
| $\{x \mid \cdots\} \bigcap \{y \mid y \text{ の式 }\}$ | $\{x \mid \cdots \wedge x(\text{元 } y) \text{ の式 }\}$ |
| $A \subseteq B \wedge A = B$ | $A = B$ |
| $X \in \{A\}\ (\wedge\ or \bigcap)\ X \in \{B\}$ | $X \in \{A \wedge B\}$ |

**表 2**: 簡約化ルール (抜粋)

$$
\begin{aligned}
&dom(r^{-1}) \subseteq b\} \wedge \\
f^{-1} \in \{r \mid r \in b \leftrightarrow a \wedge\ &dom(r) \subseteq b \wedge \\
&dom(r^{-1}) \subseteq a\}\} \bigcap \\
\{f \mid f \in \{r \mid r \in a \leftrightarrow b \wedge\ &dom(r) \subseteq a \wedge \\
&dom(r^{-1}) \subseteq b\} \wedge dom(f) = a\})
\end{aligned}
$$

### 3.2 簡約化

条件式中の重複する項を削除する. 簡約化は表 1 の代わりに表 2 を参照し 3.1 節と同じ手順を式が変化しなくなるまで適用する. 例として 3.1 節でプリミティブ化した式を簡約化すると以下のようになる.

$$
\begin{aligned}
X \in \{f \mid & f \in a \leftrightarrow b \wedge \\
& f^{-1} \in \{r \mid r \in b \leftrightarrow a \wedge \\
& \qquad dom(r) \subseteq b \wedge dom(r^{-1}) \subseteq a\} \wedge \\
& dom(f^{-1}) \subseteq b \wedge a = dom(f)\}
\end{aligned}
$$

### 3.3 推移律の書き出し

3.1 節で一式の暗黙の条件は明示できるが二式の推移律によって成り立つ条件は明示できない. 例えば $a < b$ と $b < c$ という二式が存在したとき, $a < c$ の暗黙の条件が成り立つ. 推移律の書き出しは全ての条件式を一つのリストにして入力し以下の作業を行う.

- リストの先頭から順に根の演算子が推移律を持つ演算子にマッチしないか調べる.
- マッチした場合, リストの他の全ての式とその項に対して変換対象を書き換えた式のリストを作る.
- 書き換えた式と元の式のリストを連結して重複した式を取り除く.
- 式が増えればその式のリストを入力として推移律の書き出しを行う, 変わらなければ終了する.

例としてこのアルゴリズムによって $X : \{x \mid x \in a \wedge x \in b\}, a \subseteq A, b = B$ の三式間の暗黙の条件を書き出すと以下のようになる.

$X : \{x \mid x \in a \wedge x \in b\}$, $X : \{x \mid x \in A \wedge x \in b\}$,
$X : \{x \mid x \in a \wedge x \in B\}$, $X : \{x \mid x \in A \wedge x \in B\}$,
$a \subseteq A$, $b = B$

一部の項が重複していることから, リストの条件式全てを $\wedge$ でつないで簡約化することで以下の式が得られる.

$X : \{x \mid x \in a \wedge x \in b \wedge x \in A \wedge x \in B\}, a \subseteq A, b = B$

| 演算子 | 変換対象 | 変換後 |
|---|---|---|
| $A = B$ | $A\ or\ B$ | $B\ or\ A$ |
| $A \leq B$ | $C \leq A\ or\ B \leq D$ | $C \leq B\ or\ A \leq D$ |
| $A \subseteq B$ | $Set\ A$ | $Set\ B$ |

**表 3**: 推移律の書き出し (抜粋)

### 3.4 操作分割

操作分割ではモデルの操作を一代入文毎に分割する. ただし if 文などで排他的な操作が存在する場合, その分岐毎に操作を一纏めにして一操作とする.

### 3.5 条件抽出

3.4 節で分割した操作に必要な制約条件を元のモデルから抽出する. 3.1 節と 3.3 節でモデルの制約条件は暗黙の条件が全て書き出されている. そのため, 操作で使われている変数に注目して抽出を行うと細分化モデルが生成される. 例として,$X := X \bigcap a$ の操作に必要な制約条件を 3.3 節で得られた制約条件から抽出すると以下の細分化モデルを得る. なお A と B は基本集合である.

制約条件

$$X : \{x \mid x \in a \wedge x \in A \wedge x \in B\}$$

操作

$$X := X \bigcap a$$

## 4 考察

従来の制約条件展開は停止性の保証がなく暗黙の条件の明示は困難であった. 本稿ではこれを二つにわけて問題の解決を図った.

### 4.1 暗黙の条件の明示

暗黙の条件は式を最も基本的な状態からの変換の違いによって生じるものと, 推移律によって推論できるものがある. プリミティブ化した式は演算子の種類が非常に少なくなり基本的な表現になっているため変換のしかたによって生じる暗黙の条件は無い. 基本的な表現の二式を推論しても異なる演算子に変換されることはない. よって, 推移律により式中の変数を変えた式を書き出すことにより暗黙の条件を全て明示できる.

### 4.2 アルゴリズムの停止性

従来は条件式が爆発的に増えるため, 停止性を保証できなかった. しかし制約条件展開をプリミティブ化と推移律の書き出しの二つにすることでプリミティブ化は一方向の変換ルールしか持たない式の書き換えになり, ループになる変換存在しないので必ず停止する. 一方, 推移律の書き出しでは条件式が増えるが, 推論ルールが少なく必ず有限回の書き出しで止まる.

## 5 まとめ

本稿ではプリミティブ化を用いることで従来の MSSS で問題となっていた全暗黙の条件の明示と停止性の二点を解決したモデル細分化アルゴリズムを提案した.

## 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, Vol. SE170, No. 18, pp. 1–8, Nov 2010.