B-024

B-Method におけるリファインメントの集約化手法

Merging Method of Refinement in B-Method

福泉 真隆　　織田 健
Masataka Fukuizumi　　Takeshi Oda

1　はじめに

B-Method による高信頼ソフトウェアを分解、部品としての整備、要求に応じた合成によりソフトウェアを生成する MSSS(モデル充足ソフトウェア合成) 手法を我々は提案してきた。B-Method の開発では抽象度の高いモデルから、より具体的なモデルへのリファインメントを繰り返す。この段階的詳細化を考慮してソフトウェア部品を不都合なく生成するためには、部品の仕様として妥当なモデルの抽象度の決定と再利用元のソフトウェア毎に異なるリファインメントの段数の固定を行う必要がある。本研究では MSSS 手法において不十分であったリファインメントへの対応を進めるため、部品の仕様としてモデルの抽象度を定めた上で詳細化の段数を集約する手法を提案する。

2　研究背景

2.1　形式手法 B-Method における段階的詳細化

形式手法の 1 つである B-Method の開発では、まずソフトウェアの仕様となる抽象機械 (要求モデル) を記述する。この段階では基本的にアルゴリズムのような実装に関わることは記述しない。モデルに対してアルゴリズムやデータ構造について詳細化したモデル (リファインメント) を段階的に繰り返し記述していく。これを段階的詳細化と呼ぶ。モデルとリファインメントの関係は、モデル変数とリファインメント変数間の関係をリファインメント内に記述することで表現する。この関係をリンク不変条件という。リンク不変条件を用いてモデルとリファインメント間の整合性は証明される。

2.2　MSSS(モデル充足ソフトウェア合成) 手法

2.2.1　概要

MSSS 手法 (図 1) は B-Method によって記述されたモデルからソフトウェアを自動合成する手法である。MSSS(モデル充足ソフトウェア合成) と MSFC(モデル充足細粒度部品) 生成の 2 つによって成る。MSSS は要求が記述された B-Method のモデルを入力として与え、ソフトウェア部品リポジトリからそれに合致した部品を取り出し合成することでソフトウェアを生成、出力する。MSFC 生成は既存の B-Method のソフトウェアを分解して、ソフトウェア部品である MSFC を生成し部品リポジトリに登録する。

2.3　段階的詳細化を考慮した部品生成

MSSS 手法において段階的詳細化を考慮した上で不都合なく部品を生成するためには以下の課題が存在する。

2.3.1　部品のリファインメント段数の差異

MSSS 手法の再利用は、要求モデルを小さな粒度で分解して得られた細分化モデル単位で行う。部品リポジトリから細分化モデルを検索キーとして部品検索を行い、マッチしたソフトウェア部品群を合成することで要求に応じたソフトウェアを生成する。このソフトウェア部品の出自となる既存のソフトウェアは部品毎に違う可能性があり、またリファインメントの段数がソフトウェア毎に異なることがある。そのためリファインメントの段数をそのままにして部品生成を行うと、部品毎にリファインメントの段数が差異が生じることがあり得るため、部品結合時に不都合が起こる。

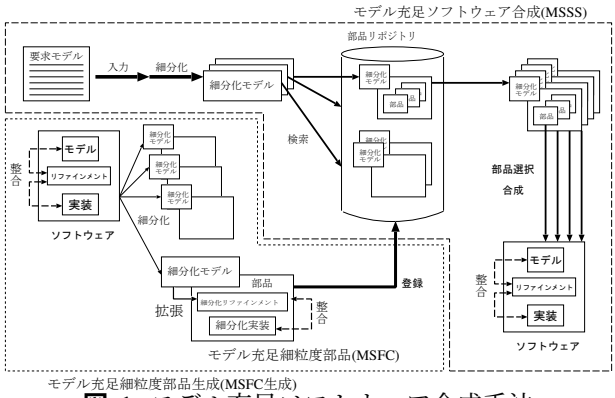

図 1: モデル充足ソフトウェア合成手法

2.3.2　部品仕様の抽象度

B-Method の開発では段階的詳細化により、高抽象度の仕様であるモデルと実装に向けた具体的な記述を持つ仕様であるリファインメントを繰り返し記述して、最終的に実装を記述する。モデルの段階では非決定的な条件分岐といった、振る舞いに関して曖昧な記述が記述できる。このようなモデルをソフトウェア部品の仕様として扱うのは問題があると考えられる。

3　リファインメントの集約化手法

3.1　部品仕様の抽象度の決定

段階的詳細化により、部品の仕様と成りうる記述はモデル (R_0) とこれを段階的に詳細化しているリファインメント群 ($R_1..R_n$) が存在する。R_0 における非決定的条件分岐の記述は、段階的詳細化を繰り返すことで最終的にある時点のリファインメント (R_X) で決定的条件分岐の記述に変化する。よって $R_0..R_X$ を仕様、$R_{X+1}..R_n$ をリファインメントとして扱えば良いと考えられる。部品検索時には細分化モデルという 1 段の仕様を検索キーとして使用するため、$R_0..R_X$ も 1 段の仕様であるモデルにすべきだと考えられる。またリファインメントも 2.3.1 節より段数を固定する必要があるため 1 段にしなければならない。そのため多段のリファインメントを 1 段にするためのリファインメントの集約化が必要となる。

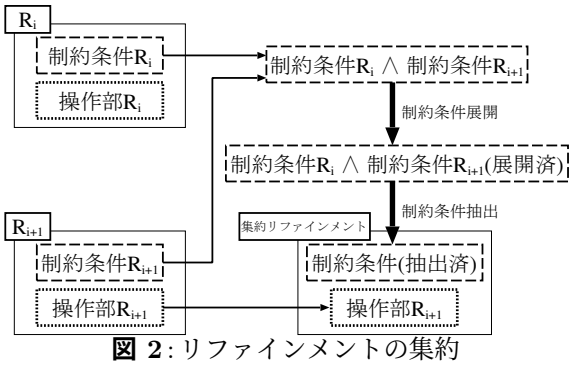

図 2: リファインメントの集約

3.2 集約化の概要

リファインメントの集約はあるリファインメント (R_i) とそれより 1 段具体的に記述されたリファインメント (R_{i+1}) の集約を繰り返すことで、複数段を 1 段に集約する。R_{i+1} には自身が持つ変数と R_i 内の変数間の関係を定義する制約条件であるリンク不変条件を持つ。このリンク不変条件を使うことで、R_i の変数の制約条件を R_{i+1} の変数を使う表現に変換し R_{i+1} 内に追加することで R_i と R_{i+1} の集約が可能になる。

R_i の制約条件の変換では、三鍋 [3] が提案した、制約条件の表現を全て数学的に等価な一定の形に書き換える処理である制約条件展開を利用する。この制約条件展開を R_i の制約条件、リンク不変条件を含んだ R_{i+1} の制約条件を論理積で繋いだ制約条件に対して行い、展開した制約条件から R_{i+1} の操作部にとって過不足無い制約条件を抽出する。この抽出した制約条件と R_{i+i} の操作部を組み合わせた物を R_i と R_{i+1} の集約リファインメントとして出力する。これから 2 つのリファインメントの集約手順 (図 2) について説明する。

3.3 制約条件展開

まず R_i の制約条件を R_{i+1} の変数を用いた形に変換するために、R_i、R_{i+1} の制約条件とリンク不変条件を論理積で繋いだ制約条件に対して制約条件展開 (図 3) を行う。制約条件展開は、制約条件の表現をプリミティブな演算子のみを使用する形に変換するプリミティブ化を行い、それらに対して推論を行い制約条件間に存在する暗黙の条件を導き出す。

プリミティブ化は制約条件の式中の全ての演算子を低機能なプリミティブな演算子に変換する動作である。例えば、$P \Leftrightarrow Q$ は $P \Rightarrow Q \wedge Q \Rightarrow P$ のように変換される。推論では、プリミティブな演算子のみで構成された制約条件間の暗黙の条件を導き出す。例えば、$a \leq b \wedge b \leq c$ から $a \leq c$ が導出される。

3.4 制約条件抽出

制約条件展開された制約条件には、R_i の変数が存在する制約条件等の集約リファインメントにとって必要のない制約条件が存在するため、必要なものを過不足無く抽出する (図 4) 抽出ルールは以下の 2 つ場合で異なる。

$R_0..R_X$ の集約: R_{i+1} の変数のみで構成された制約条件

$R_{X+1}..R_n$ の集約: R_X の変数 (=$R_0..R_X$ の集約モデルの変数) と R_{i+1} の変数のみで構成された制約条件

リファインメント段数の集約時にはモデルと集約リファインメントの関係性の保持が必要となるため、リンク不変条件を残す必要がある。そのためモデル変数である R_X の変数が使用された制約条件も抽出しなければならない。

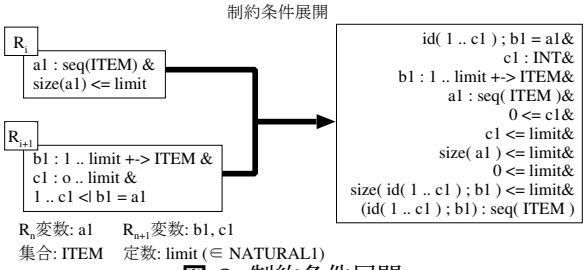

図 3: 制約条件展開

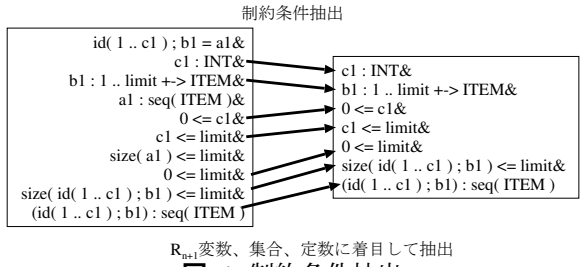

図 4: 制約条件抽出

3.5 入出力

集約の入出力は $R_0..R_X$ の集約と $R_{X+1}..R_n$ の集約で幾らか異なる。入力は前者はモデルと複数段のリファインメント、後者は複数段のリファインメントとなる。出力は前者はモデル、後者は単一のリファインメントが出力される。

4 考察

小規模なモデルを用いてリファインメントの集約を行った。B-Method の開発ツールである Atelier B の証明器を用いて検証したところ解決できない証明責務が残ってしまった。これは制約条件展開の過程で 1 以上の自然数の集合を意味する NATURAL1 が、実装可能な整数の集合を示す INT かつ 0 より大きい集合という表現に変わり、意味合いの変化が起きてしまったため誤りが生じたことが原因だと考えられる。正しく式を展開するために、制約条件展開のルールを見直すことが必要となる。

5 おわりに

本稿では、MSSS 手法において不都合のない部品生成を目指したリファインメントの集約手法について提案した。今回の実験で生じた問題の解決とより実践的なモデルを用いた検証、部品整備手順の決定を行うことが課題である。

参考文献

[1] 中島震, 来間啓伸. B Method による形式仕様記述, 近代科学社, 2007.

[2] 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 大学院 電気通信学研究科 博士 (工学) 学位論文, 2014.

[3] 三鍋孝介. 文字列一致による数学的等価性判定可能なモデル細分化アルゴリズム, 第 12 回情報科学技術フォーラム, 2013.