B-021

形式的ソフトウェア合成手法における不足部品の要求提示

Presenting the Requirement of Shortage Components in the Formal Software Synthesis Method

熊谷 恒 **織田 健**
Wataru Kumagai **Takeshi Oda**

1 はじめに

我々は仕様と実装が対になった細粒度部品が登録されたリポジトリから、新規要求を細分化した数学的モデルをキーとして検索を行い、得られた部品群を結合することで高信頼ソフトウェアを合成する手法の確立を目指している [1]。合成に必要な部品が不足してしまう場合があり、利用者の手で実装の記述を補完する必要がある。部品の仕様は形式的に提示可能だが、細分化されたモデルは可読性が低い為そのまま提示するのは適切ではない。また実装の記述によっては他の部品と整合がとれなくなる場合がある。本稿では可読性の向上、及び他の部品との整合性に配慮した部品充足の支援方法を提案する。

2 形式手法 B Method

B Method[2] は形式手法の 1 つで、仕様記述からプログラムの導出までの一連の過程を支援する。集合論と述語論理に基づいた仕様記述言語を用いることで仕様の曖昧さを除き、仕様の無矛盾性と実装との整合性を機械的に検証できる。B Method のモデルと実装の組から C 言語等のコードを生成可能なため、3 節で述べる MSSS 手法ではモデルと実装の組をソフトウェアとして扱う。

3 MSSS 手法

MSSS 手法は B Method で記述されたモデルを要求モデルとして受取り、高信頼ソフトウェア部品を再利用することで、要求モデルを満たす B Method のソフトウェアを生成する手法である。MSSS 手法は、図 1 のように部品生成と自動コード合成 (MSSS) からなる。MSSS は要求モデルを入力とし、実装及びその実装に必要な拡張を施したモデルを出力する。高信頼ソフトウェア部品をモデル充足細粒度部品 (MSFC) と呼ぶ。MSFC は実装依存モデル、細分化実装の 2 つの記述で構成され、細分化モデルを部品の仕様として持つ。実装依存モデルは細分化モデルの拡張であり、細分化実装は実装依存モデルと整合性を保つことにより、MSFC の信頼性を保証する。

3.1 細分化モデル

細分化モデルとは、B Method のモデルに含まれる操作群を可能な限り細かく分割した一要素を唯一の操作として持つモデルである。不変条件や事前条件等の制約条件は操作を表すのに必要最低限な制約のみを持つ。モデル細分化の際、着目した操作に関する制約条件を漏らさず抽出するために、暗黙的に存在する条件の推論が行われる。また数学的に等価なモデルの字面を統一する事を目的として、モデル内の演算子の記述を低機能な物に統一するプリミティブ化、及び構文の並び替えと変数名の付け替えが行われる。上記操作により字面一致による部品検索が可能となるが、モデルの記述が書き換えられる

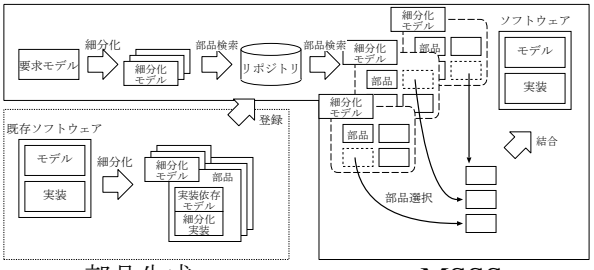

図 1: MSSS 手法

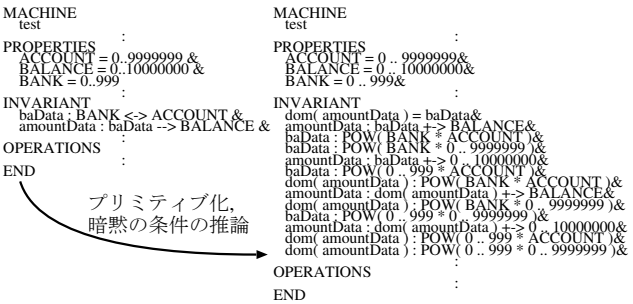

図 2: モデル細分化による式変形

ため、細分化モデルの可読性は低くなる。図 2 はプリミティブ化及び暗黙の条件の推論による式変形を表している。制約条件が 5 個から 16 個に増加しており、演算子が低機能な物に置き換えられている。

3.2 MSSS

ソフトウェア合成には、リポジトリから要求モデルを満たす部品を検索する必要がある。部品検索では要求モデルを細分化し、得られた細分化モデル群を検索キーとして、それぞれのモデルを満たす部品群を得る。B Method によるモデルは抽象度が高く、1 つの細分化モデルに対し実装方法の異なる複数の部品が存在し、非数学的要求の実装方法の差異等により結合不能な部品の組み合わせが存在する。リポジトリに登録されている部品の不足や結合不能な部品の組み合わせを原因として、ソフトウェア合成に必要な部品が不足する場合がある。

3.3 利用者による不足部品の補完

必要な部品が不足している場合には、利用者の手によって不足部品を追加する必要がある。この際不足部品の仕様として細分化モデルを利用者に提示する必要があるが、可読性が低下しているためそのまま提示するのは適切ではない。また利用者が追加した部品は他の選択済み部品と合成可能である必要があるが、その条件は細分化モデルを見ただけでは判らない。したがって本研究では他の部品と結合するための条件を明確化した上で、可読性を向上させた細分化モデルを提示する手法を提案する。

4 不足部品の要求提示

4.1 可読性の向上

可読性を定量的に評価することは難しいため本稿では、MSSS 手法の運用では要求モデルの作成者と不足部品を補完するものが同一人物であるという前提のもと、要求モデルの記述を可能な限り再現したモデルが、可読性の高いモデルであると定義する。細分化モデルの可読性を高めるために以下のアプローチをとる。

プリミティブ化の逆変換 モデル細分化でのプリミティブ化による式変形の対応関係を保存しておき、プリミティブ化前の記述に戻す事で要求モデルの記述を再現する。細分化モデルは必要最低限の制約しか含まず再現できるケースは稀であるため、再現出来ない制約条件に対してはプリミティブ化の変換ルールを逆方向に適用することで高機能な演算子を含む式に変換し、可読性を向上させる。

自明な制約条件の削除 細分化モデルは暗黙の条件の推論により、要求モデルに比べ多くの制約条件が含まれる。しかし推論により追加された制約条件は数学的に等価で冗長な物を含む。従って自明な制約条件を削除することで可読性を向上させる。要求モデルの記述に近づけるため、暗黙の条件の推論を行う際に何段階目の推論で導出された式かを記録しておき、後に導出された制約条件を優先的に削除する。

変数名の付け替え 変数名付け替えの対応関係を保存しておくことで、要求モデルと同じ変数名を使用して利用者に提示する。

モデル細分化は、プリミティブ化、暗黙の条件の推論、変数の付け替えの順に行われる。要求モデルの記述を再現するに当たっては、逆順に適用するのが適当であると考えられる。以下の手順で可読性の向上を行う。

1. 変数名を要求モデルの物に付け替える。
2. 細分化モデルの制約条件について、モデル細分化時と同様の推論を行う。
3. 推論により得られた式に一致する制約条件を、冗長な制約条件として削除する。
4. プリミティブ化による式変形の記録を参照し、要求モデルの記述に再現可能な制約条件を書き換える。
5. 4 で書き換えられなかった制約条件に対し、プリミティブ化ルールを逆方向に適用し式変形を行う。

4.2 他部品との整合性の確保

利用者に提示した細分化モデルに従って実装を記述しても、実装依存変数の型等が周囲の部品で定めたものと矛盾する場合、部品結合を行うことができない。その為、利用者に実装を記述させる際には周囲の部品の情報を実装依存モデル、及び細分化実装に輸入した上で提示する必要がある。利用者が実装する部品の記述のうち、周囲の部品の影響を受ける物は以下が挙げられる。

モジュール B Method のモデルでは I/O に関する記述を扱えないため、実装にモジュールを輸入する事によって機能を実現する。

リンク不変条件 B Method ではモデルの変数は抽象変数と呼ばれ実装で扱えないため、実装では実装変数を定義して実装変数と抽象変数の関係を不変条件に記述する。このような条件をリンク不変条件と呼ぶ。

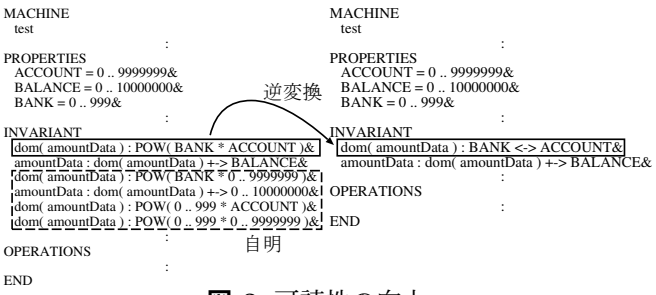

図 3: 可読性の向上

実装依存変数 モジュールの輸入に伴い実装に特化した記述をモデルに追加し、実装依存モデルを構成する。この際追加される変数を実装依存変数と呼ぶ。

これらの情報を他の部品から輸入しテンプレートとして利用者に提示することで、他の部品との整合性がとれた部品の記述が可能になる。

5 適用例

図 2 右側の記述から生成された細分化モデルに本稿のアプローチを適用した場合の記述の変化を検討する。細分化モデルには変数 baData 及び baData を含む制約条件は含まれないと仮定する (図 3 左側)。まず制約条件に対し、推論を行う。この結果 INVARIANT 節の 6 行目以降が削除される。またプリミティブ化ルールの逆変換を行う事で、INVARIANT 節の 1 行目の演算子が高機能な物に置き換えられる。結果として図 3 の右側の記述が得られ、可読性の向上が確認できた。

6 考察

本手法では自明な制約条件をモデル細分化で行った推論を再度行うことで発見しているが、既存のルールセットだけでは全ての自明な制約条件を推論するには不十分である。現在は 2 式以上から 1 式を推論するルールしか整備されていないが、1 式から 1 式を推論する新たなルールセットを定義する必要があると考えられる。

本稿では自明な式の削除を行った後にプリミティブ化の逆変換を行ったが、要求モデルの記述が再現できなくなる場合が考えられる。プリミティブ化の逆変換を先に行った場合自明な式の推論が難しくなるため、各操作を一度で完了させるのではなく。交互に複数回行うような工夫が必要になると考えられる。

7 おわりに

本稿では MSSS 手法の不足部品の要求提示において、演算子の書き換え、自明な制約条件の削除、変数名の付け替えによる可読性の向上、及びモジュール、リンク不変条件、実装依存変数の輸入により他部品との整合性に配慮した実装テンプレートを提示する手法を提案した。今後はより詳細なアルゴリズムの策定、追加の推論ルールの定義、評価実験を用いた有用性の評価を行う。

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. *SIGSE 170*, 2010.

[2] 中島震, 来間啓伸. B メソッドによる形式使用記述. 近代科学社, 2007.