CafeOBJ から Java への変換と同期記述

兼　英樹

北陸先端科学技術大学院大学 情報科学研究科

1999 年 2 月 15 日

キーワード： CafeOBJ, Java, 書き換え規則, オブジェクト合成, 同期記述.

1　研究の背景と目的

要求仕様からコーディングまでのソフトフェア開発のさまざまな局面において形式手法に基づいた活動が盛んに行われている。形式手法の仕様作成段階で用いられる言語は形式仕様言語と呼ばれ、厳密な数学モデルや論理体系に基づいて仕様を記述することができる。これは、より信頼性の高い仕様を作成することができ、仕様の機械的検証も可能であるためソフトウェア開発の効率面でも多くの利点を持っている。

形式仕様言語 CafeOBJ において、射影演算を用いたオブジェクト合成はシステムの振る舞いを非常にうまく記述することが出来、その分析も簡単に行うことができる。

このスタイルの仕様は以下の特性を持っている。

1. 単純なプリミティブモジュールから合成されている。
2. 仕様の分析は簡単に行え、CafeOBJ インタープリタのような自動ツールで支援することができる。

また、近年 Java 言語はその言語としてのすぐれた特性、プラットフォーム非依存性により、急速に開発言語として導入されるケースが増えている。しかし、Java 言語は数学的意味の裏付けを持たないため、設計段階で検証を行うことが難しい。

形式言語から java 言語への変換システムが作成されれば、設計段階で検証を行い、その仕様を忠実に反映した Java コードを得ることができるため、システムの信頼性を向上させることができる。

本研究では、形式仕様言語 CafeOBJ の仕様から Java プログラムへ変換法を開発することを目的としている．ただし，一般的な CafeOBJ 仕様を対象とするのではなく，射影

演算を用いて階層的に合成された振舞い仕様を対象とし，仕様の構造を保存する効率的なJavaプログラムへの変換法を目的とする．

変換方法に関しては以下の特性をふまえて提案を行う。

1. CafeOBJ 仕様の構造を保存する
2. 作成されたJavaプログラムは合理的な時間と良いスペース効率で実行できる

2 オブジェクト合成

合成対象の各オブジェクトの状態を得るために射影演算 (projection) という演算を定義する。この射影演算は合成対象となっている各オブジェクトに対して定義され、合成されたオブジェクトに対する (操作、属性) 演算が合成対象のオブジェクトに対する (操作、属性) 演算になるように置き換える演算である。

射影演算を用いたオブジェクト合成では、仕様の合成対象のオブジェクトの仕様コードと振る舞い等価の証明が再利用できるため、検証における手間を大きく省くことができる。

3 研究のアプローチ

本研究では代数仕様言語CafeOBJを用いて記述された仕様を効率的なJavaプログラムへ変換する規則の提案を行う。仕様の例としてATM(現金自動預金システム) の例をとりあげ、オブジェクト合成された仕様がその構造を保ったまま変換できることを確かめる。

形式仕様を使う利点の一つに検証可能であることが挙げられるが、システムが大きくなればなるほど検証は困難になる。射影演算を用いて階層的に合成された振舞い仕様では、合成対象のオブジェクトにおいて証明されたことが合成後のオブジェクトにおいて再利用可能であるので、検証における手間が大きく省かれている。

研究の流れとしては、最初に簡単な例題 (基本データ構造) についてCafeOBJで仕様を記述し、Javaへの変換手順、変換したJavaが仕様を満たしていることについての考察を行った。次にオブジェクト合成された仕様記述に対しても同様な変換手順で変換できるようhidden sort, 射影演算についての変換規則の提案を行った。

また、CafeOBJで記述された仕様に並行性がある場合について、Javaのマルチスレッドを使って実際に並行動作するようなコードに変換する方法を考え、排他制御に関しても考察を行った。これによりCafeOBJで並列分散システムの記述が容易に行える。

最後に、関連研究として、CafeOBJの仕様の変更がどの範囲まで影響するかを調査し、低コストで仕様を変更できるように考慮した。

一般に、ソフトウェアの要求が変化した場合にはその仕様を変更し、変更された仕様において要求が満たされていることを証明しなおす必要があるが、もし要求の変化がどのコンポーネントに影響し、そのコンポーネントを他のコンポーネントと置き換えた際にどん

な性質を検証すれば全体として全ての要求を満たすことができるのかが分かれば大幅なコスト削減が期待できる。

この影響範囲と変換した Java プログラムに対しての影響範囲について比較し、低コストで変更できるスタイルについて考察を行った。

4 結論

本稿では、射影演算を用いて階層的に合成された振舞い仕様を対象とし、

CafeOBJ 仕様から仕様の構造を保存しつつ、効率的な Java プログラムへ変換する規則を ATM の例を用いて示した。変換した Java プログラムは CafeOBJ の仕様で表現されている複数のモデルの中で具体的な 1 つのモデルを表現している。

また、Java に変換した際に実際に並行動作できるモデルに変換するための同期記述について述べ、CafeOBJ で記述した ATM が実際に Java で並行動作することを確かめた。

また、最上位のモジュールと最下位のモジュールを変更した際に変更が必要な範囲を、仕様に対して既存の機能の組み合わせ＋インタフェースの追加を行う例で示した。この結果から、任意のモジュールが変更された場合にその上位のモジュールと下位のモジュールの変更すべき場所が特定できることが分かった。