新

組み込みソフトへの数理的アプローチ

～形式仕様記述をどのように使うか～

第13回

検証しながらプログラムを作るPDD
――事前条件と事後条件による仕様記述

藤倉 俊幸

はじめに

前回（2010年6月号，pp.190-195）はプログラム検証ツールを使用してソース・コードのリバースをしようとしたがあまりよい結果は得られなかった．しかし，検証しながらプログラムを作るPDD（Proof Driven Development）というものを思いついた．今回はPDDの可能性について話を進めてみたい．

ここでプログラムといっているのは，単純な関数の呼び出し系列のことで，スレッド1本を対象としている．マルチスレッドの場合は，並列性を考慮したアプローチが必要になる．まずは，シングル・スレッドで問題がないことを前提として展開した方が話が簡単になると予想されるので，ここではシングル・スレッドから検討する．

1 プログラムの仕様とは

● 事前条件と事後条件を使ってプログラムを定義する

プログラムの仕様とは，プログラムが何をしなければならないかを書いたものである．パフォーマンスとかメモリ使用量なども仕様には必要だが，ここでは「何」の部分を正確に書くことに注目する．

「何」の部分の形式的な仕様は，ホーア（Hoare）によって構築された論理体系をベースに考えることができる．それは，事前条件Qと事後条件Rを論理式，Sをプログラムとするとき，｛Q｝S｛R｝の形の表明で表現できることと，これを定理として証明できればプログラムSが正しいことを示せるというものである．

…と言われても何のことかわからないけれども，要するにプログラムが何をするのかについてはQとRで表現できるということである．契約による設計（Design by Contract：DbC）も同様の考え方である．ただし，DbCではクラスの概念があるので事前条件と事後条件のほかに不変条件も利用できる．関数だけを考えるときもループを構成する場合は，ループを回っている間は変化しない条件を考えることはできるが，DbCはデータ構造が満たすべき条件として不変条件を導入する．シングル・スレッドを考えている場合やプログラムが終了した後の結果の正しさを考えるのであれば，事前条件と事後条件で十分に記述できる．なので，とりあえず不変条件のことは考えないことにする．

｛Q｝S｛R｝の解釈は，

①プログラムSはQを満たすどのような状態の下で実行しても有限時間のうちに実行が終わり，Rを満たす状態になる(1)

②プログラムSはQを満たすどのような状態の下で実行しても，Sの実行が終わればRを満たす状態になる(2)

の2通りがある．①の場合，プログラムSの実行がちゃんと終了することも解釈の中に含んでいるが，②の場合はプログラムSが終了することは仮定として扱っている点が異なる．プログラムSがちゃんと終了しない理由としてはゼロの割り算とかいろいろあるが，ここでは無限ループを想定しており，ゼロの割り算は異常終了なのでプログラムSが正しくないと判断される．②の解釈というか証明手法による場合は，ループを抜けることを別途証明する必要がある．

この二つの解釈のうち，②の方が先（1969年）で，当時はQ｛S｝Rと書き，これによって証明されるのはプログラムの部分正当性（partial correctness）と呼ばれる性質である．①の場合は｛Q｝S｛R｝と書き，実行が終わることも保証する．証明される性質は全正当性（total correctness）と呼ばれる．実行が終わらないプログラムに対してQ｛S｝Rは真になるが｛Q｝S｛R｝は偽になる．ダイクストラ①とフロイド-ホーア②の仕事は同じだという意見もあるが，不

KEYWORD —— Proof driven development，Design by Contract，CBMC，フロイド-ホーア