論文の内容の要旨

Verification of Concurrent Programs using Proof Assistants
(定理証明器に基づく並行プログラムの検証)

アフェルト レナルド

近年、これまで人間が行っていた重大な仕事の多くがコンピュータシステムに任されるようになってきている。そのような重要なシステムの多くは並行・分散システムであり、そのようなシステムの誤りにより悲惨な事態が起こる可能性を防ぐため、並行プログラムの形式的な検証の重要性が増している。並行プログラムは逐次プログラムよりも本質的に複雑であるので、形式的な検証は容易ではなく、逐次プログラムの検証よりも一般的な技術が必要となる。この点で、定理証明器は並行プログラムの検証のための道具として特に適していると思われる。しかしながら、現在の定理証明器を並行プログラムの検証に直接適用することは困難である。なぜなら、並列性および非決定性のため、システムのとりうる状態数が爆発してしまうからである。また、定理証明器の多くには、λ計算に関して推論するための機構が備わっており、逐次プログラムの検証に利用することができるが、並行プログラムの検証のためのそのようなサポートはない。並行プログラムの検証をサポートするため、定理証明器上にプロセス計算を形式化した研究はあるが、それらは純粋なプロセス計算を対象としており、実用的なプログラムの検証を行う上では不十分である。

本論文では、定理証明器を用いた現実的な並行プログラムの検証を可能にするための２つのアプローチを提案し、定理証明器 Coq を用いて評価を行う。

１番目のアプローチとして、並行プログラムを関数プログラムとしてモデル化、検証する手法を提案する。このアプローチでは、検証の対象とする並行プログラムおよびその環境をシステマティックに関数としてモデル化する。例えば、並行プログラムの非決定性は、関数の引数として試行オラクルを与えることによって表現される。このアプローチに基づき、現実的なクライアント／サーバアプリケーションの例として、SMTP サーバの受信部分（700 行の Java プログラム）が満たすべき４つの性質を形式的に検証した。この検証により、プログラムのバグを発見するとともに、証明の複雑さが適切な範囲内におさまることを確認した。このアプローチは、上記の利点にも関わらず、様々な問題を抱えている。例えば、モデルが複雑になり、補題の再利用性も落ちる。

第２のアプローチでは、上記に述べた第１のアプローチの問題点を解決するために、並行プログラムを直接定理証明器上にモデル化し、検証するためのライブラリを設計、実装する。このライブラリは、(1)モデル記述言語、(2)仕様記述言語、(3)検証に利用する補題群からなる。モデル記述言語は、Coq のデータ型と関数を用いて拡張された pi 計算として構築される。仕様記述言語は、公平性の概念を導入して拡張された空間論理として構築される。これは、種々の現実的な並行プログラムをモデル化・定式化するのに役立つ。補題群の設計において、並行プログラムが多くの状態をもつ場合に状態空間が爆発するとい問題がある。我々は、partial order reduction ---検証のために探索される状態空間を削減する技術--- を我々の仕様記述言語の述語に適応することにより、この問題を解決した。このライブラリ（Coq のスクリプトでおよそ 1 万 5 千行）を用いて、第１のアプローチで用いたメールサーバプログラムモデル化を行い、大事な性質を形式的に検証した。この実験により、複雑なモデル化の技術を用いることなく、現実的な並行プログラムを定理証明器上で検証できることを示した。