論文の内容の要旨

論文題目“Type Systems for Formal Verification of Concurrent Programs”
(並行プログラムの形式的検証のための型システム)

氏名　　末永 幸平

信頼性の高い並行プログラムの記述は、デッドロックや競合状態の可能性のために、逐次プログラムの記述よりも難しい。また、スレッドが非決定的に並行実行されることにより起こる状態爆発により、プログラムの状態を追跡することが難しいことも、高信頼な並行プログラムの記述の難しさの原因となっている。

ソフトウェアの信頼性を上げるためによく用いられる手法に、テスト実行によるデバッグがある。しかしながら、この手法では、正しくない振る舞いに至るプログラム中の実行経路を見落とす可能性があるため、高い信頼性を保証する目的には不十分である。また、特に並行プログラムの場合、すべての実行経路をテストすることは、実行の非決定性のためにそもそも困難である場合が多い。さらに、正しくない振る舞いが発見された場合においても、動作の非決定性のためにその振る舞いを再現することが難しい場合があり、原因を突き止めることが難しくなっている。

このような問題に対処するための有望な手法の一つに、静的検証が挙げられる。これは、プログラムを実行する前に、そのプログラムを正しさが数学的に証明された方法で検証するという手法である。しかしながら、これまでに逐次プログラムの静的検証手法は多く研究されているものの、並行プログラムに対する静的検証手法の研究は、割り込みや実行時の通信チャネルの生成といった、現実のソフトウェアによく現れる特徴が扱われていないという点で、未だ発展途上の段階である。テスト実行によるデバッグ手法は、並行プログラムに対しては逐次プログラムの場合よりも有効性が低いため、これらの特徴を扱うことのできる静的検証手法の研究は非常に重要である。

そのような現実的な並行プログラムを扱うことのできる静的検証手法の構築への第一歩として、我々はデッドロックと資源使用という並行プログラムの二つの重要な安全性に関わる問題について、型システムに基づく検証手法を提案する。本論文で提案するデッドロック検証手法では、(1) 入れ子構造をとらないロックプリミティブ (2) 破壊的代入が可能なロックへの参照 (3) 割り込みという三つの特徴を備えたプログラムを扱うことができる。これらの特徴は、現実のプログラムではよく用いられるにもかかわらず、既存のデッドロック解析では扱われてこなかった。

我々は、これらの特徴を備えた計算体系を構築し、その計算体系で記述されたプログラムのための型システムを提案する。我々の型システムは、プログラムがデッドロックに陥らないことを保証するために (1) ロック/アンロック操作の間に循環した依存性がないことと (2) 獲得されたロックがちょうど一度解放されること、の二つの性質を検証する。一つ目の性質を検証するために、型システムは各ロック型には**ロックレベル**と呼ばれる自然数を割り振り、プログラムがロックをロックレベルの昇順に獲得することを保証する。二つ目の性質の検証においては、エイリアシングやロックへの参照への競合状態といった問題に対応するために、**権利と義務**という線形型に基づく概念と、**所有権**という参照へのアクセスをコントロールするための概念を導入する。また、この型システムに基づいた手法により、ネットワークプロトコルの実装の一部に対して検証を行った予備実験の結果を報告する。

次に我々はπ計算のための型に基づく資源使用法解析を提案する。資源使用法解析は、ファイル、メモリ、ソケット等の様々な資源を、プログラムがあらかじめ定められた仕様に基づいて正しくアクセスしているかどうかを実行前に検証するという解析である。この解析手法は、逐次プログラムに対しては非常に多くの研究がなされてきたが、並行プログラムに対する研究は少なく、特に資源や通信チャネルを実行時に生成してアクセスするような、現実によく使われる並行プログラムに対する検証

手法は研究されていなかった。

そのような並行プログラムに対する資源使用法解析手法を構築するために、我々は資源の生成やアクセスを行うプリミティブでπ計算を拡張し、その計算体系のための型システムを、既存の五十嵐と小林による振る舞い型システムの拡張として定義する。提案する型システムは、禁じられているアクセスを行わないという通常の資源安全性に加えて、プログラムが正常終了すれば、必要なアクセスは全て行われているという部分活性を保証する。例えば、プログラムが正常終了すれば、ファイルに対して必ず close 操作が行われていることが、部分活性により保証される。また、プログラマが型に関する複雑な記述をせずとも済むように、健全な型推論アルゴリズムを設計した。さらに、このアルゴリズムに基づいて、我々はπ計算に対する資源使用法解析器のプロトタイプを実装した。我々の知る限りにおいて、本手法はπ計算のような表現力豊かな並行言語に対する初めての型に基づく資源使用法解析である。