発表概要

インタフェースに基づく並行論理プログラム最適化コンパイラの構成法

加藤紀夫† 上田和紀††

本発表では，並行論理プログラムの中から逐次的に実行できる部分を自動的に抽出するためのボトムアップな解析法を紹介する．さらに，この方法に基づいて生成される逐次の中間コードを用いて，並行論理プログラムの最適化コンパイラを構成する方式を提案する．細粒度並行言語のコンパイルではソースレベルではなく処理系レベルの最適化が重要であり，それを並行論理プログラムに対して行うには，プロセスに対して特定の実行方法に特化された目的コードを生成することが必要となる．ところがそのような特化は並列実行および局所的選択に関する非決定性を解消することを含むため，目的コードの正当性をソースプログラムのレベルで保証することは直接的でない．そのような特化を定式化するために，本発表ではプロセスのインタフェースという定式化を導入する．インタフェースは，ある入力条件を仮定したとき，プロセスが特定の出力をした後，特定のプロセスとして振る舞うことができるというプロセスの性質を表現する．インタフェースを使ったボトムアップな解析により，プログラム中の逐次的な部分を抽出すると同時に逐次の中間コードを効率良く生成できることを紹介する．そして，中間コードに対して最適化を行うことにより，従来アドホックに正当化してきたupdate-in-placeやタグ除去など様々な最適化の安全性を，中間コードのレベルで体系的に保証できることを明らかにする．

Interface-based Optimizing Compilation for Concurrent Logic Programs

Norio Kato† and Kazunori Ueda††

We present a bottom-up method of extracting those fragments of concurrent logic programs that can be executed sequentially, and we also propose a framework of optimizing compilation of concurrent logic programs that uses sequential intermediate code generated with the method. Implementation-level optimization, as opposed to source-level optimization, is important for the compilation of fine-grained concurrent languages. To perform this, a compiler of concurrent logic programs needs to generate object code specialized for a particular way of execution of processes. As code specialization includes elimination of nondeterminacy on parallel execution and local choices, it is not straightforward to justify resulting object code by source-level analysis. To formalize such specialization, we introduce the notion of *interfaces* of a concurrent process. An interface represents the property of a process that, assuming a particular input, it can produce a particular output and then behave as another process. We show how bottom-up analysis of sequentialization using interfaces allows us to extract sequential fragments of programs and simultaneously to generate sequential intermediate code efficiently. We also demonstrate that optimization on the intermediate code can give us systematic, intermediate-code-level justification of various optimization techniques including update-in-place and tag elimination.

（平成13年10月22日発表）

† 早稲田大学大学院理工学研究科
Graduate School of Science and Engineering, Waseda University
†† 早稲田大学理工学部情報学科
Department of Information Processing, Waseda University