発表概要

手続き型動的型付け言語に対するリージョン推論の導入

齋地 崇大[1,a)]　前田 敦司[1,b)]

2018年8月1日発表

Tofte らによって提案されたリージョンによるメモリ管理では，ML プログラムの構造からリージョンを推論し，メモリを必要とするオブジェクトの生存範囲に応じて，オブジェクトのメモリの割付け・解放命令を適切な位置に挿入する．そのため，プログラマによるメモリの割付け・解放命令の記述を必要としないほか，ガベージコレクションによる管理をスタックによる管理に置換できるため，プログラムの性質によっては実行時のスループット向上につながる．しかしながら，既存の提案の多くは静的型付けを前提としており，動的型付け言語を対象としたパフォーマンスの議論や提案は少ない．そこで本発表では，手続き型動的型付け言語に対してリージョン推論の導入を行い，リージョンが推論可能なオブジェクトと推論不可能なオブジェクトに分類することで，ガベージコレクションなどの実行時メモリ管理機構によるコストの削減可能性について議論を行う．対象とする言語の構造は静的に型を決定することが困難であり，手続き的なプログラムの記述を行う Ruby や Python といったスクリプト言語とする．こうした言語処理系の多くは，静的な解析の困難さから，メモリを必要とするオブジェクトをすべてヒープ領域に格納し，管理を実行時に行わざるをえないため，ガベージコレクションによる処理を必要とし，スループットの低下や処理の停止時間が問題になりうる．

Presentation Abstract

Introduction of Region Inference for Dynamically-typed Procedural Programming Languages

Takamasa Saichi[1,a)]　Atusi Maeda[1,b)]

Presented: August 1, 2018

Region-based memory management system proposed by Tofte et al. infers regions from lexical structure of ML programs and inserts instructions to allocate/deallocate memory for objects at appropriate places in programs, according to extents of the objects. In this way, the system eliminates the need for programmers to manually specify memory allocation/deallocation. In addition, since memory management by garbage collection can be substituted to stack-like region-based management, possible improvement of throughput can be expected, depending on the characteristics of the programs. Most studies, however, assumes statically-typed languages and few are dealing with dynamically-typed languages and their performance with region-based memory management. In this presentation, we introduce region inference in dynamically-typed procedural languages by classifying objects into region-inferable and uninferable ones, and discuss the possibility of eliminating the cost related to dynamic memory management such as garbage collection. We treat scripting languages such as Ruby and Python as target language of the study. These language are procedural and in which types of data objects are hard to decide statically. Because of the difficulty of static analysis, most implementations of these languages have to store all memory-allocated objects into heap, relying solely on garbage collection to dynamically manage memory, which may lead to potentially lower throughput or undesirable pause of user programs.

This is the abstract of an unrefereed presentation, and it should not preclude subsequent publication.

[1] 筑波大学
University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[a)] saichi@ialab.cs.tsukuba.ac.jp
[b)] maeda@cs.tsukuba.ac.jp