発表概要

スタックの直接操作を必要としない一級継続の実装

川田大蔵[†] 小宮常康[††]

本発表では，Scheme 言語のプログラムを Java やスクリプト言語へコンパイルする方式について述べる．Java やスクリプト言語では，制御スタックを直接扱うことができないため，Scheme の一級継続を実装するには，継続をターゲット言語レベルで扱える表現にする必要がある．その 1 つの手法として，Baker の CPS 実行方式がある．この手法では，環境を配列で実装し，継続は明示的に関数で表現する．そして，関数呼び出し時に明示的に継続を渡し，関数からのリターンを継続の呼び出しによって行う．そのため，制御スタックが溢れる可能性があるが，C 言語を用いる Baker 方式は制御スタックのごみ集めによってこれを回避する．一方，Java やスクリプト言語では制御スタックを直接操作できないためにこの方法は使えない．そこで，我々の手法では通常はリターンによる実行を行い，制御スタックを巻き戻す．また，環境を配列で表現せずに Scheme の変数は直接ターゲット言語の変数に対応付ける．このコードの性能は，とりわけスクリプト言語においては Baker 方式より高速である．そして，一級継続を捕えるときは，例外を使って環境の値を集め，Baker 方式の環境表現にし，その実行には Baker 方式のコードを用いる．ただし，Baker 方式のコードが実行されるのは捕まえた継続に含まれるリターンの処理のときだけであるため，一級継続のサポートによる速度低下を抑えることができ，スタック溢れの問題を回避できる．

Implementation of First-class Continuations without Direct Stack Manipulation

Taizo Kawada[†] and Tsuneyasu Komiya[††]

We describe a method of compiling Scheme language to Java or scripting languages. In order to implement first-class continuations of Scheme, continuations should be expressed at a target language level since Java and scripting languages cannot handle the control stack directly. Baker's CPS execution model is one of the methods for realizing it. In this method, an environment is expressed with arrays and a continuation is expressed with a C language's function. When a compiled Scheme's function is called, those are passed to the compiled function, and returning from the compiled function is expressed by calling the continuation. Therefore, there is possibility of overflowing the control stack, and Baker's method, which uses C language, avoids it by garbage collection of the control stack. Java and scripting languages, however, cannot do garbage collection of the control stack because it cannot handle the control stack directly. So, our method normally rewinds control stack by return operations, and the environment is not expressed explicitly. Instead, Scheme variables are directly associated with a target language's variables. In scripting languages, the performance of this method is faster than Baker's one. When capturing a first-class continuation, our method collects environment information by using exception and makes it into an environment for Baker's method. When executing the first-class continuation, a code based on Baker's method is used. Our method can suppress the speed decrease caused by supporting first-class continuations and avoid the stack overflow because the code is executed only when returns in the captured continuation are performed.

（平成 18 年 3 月 17 日発表）

† 豊橋技術科学大学情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology
†† 豊橋技術科学大学情報工学系/インテリジェントセンシングシステムリサーチセンター
Department of Information and Computer Sciences, Toyohashi University of Technology/Intelligent Sensing System Research Center