発表概要

スタックベースのML処理系における効率的な一級継続の実装

皆川宜久[†] 鵜川始陽[†]
八杉昌宏[†] 湯淺太一[†]

継続とは，計算のある時点の残りの計算である．Schemeでは，これをファーストクラスのオブジェクトとして扱え，コルーチンやマルチスレッドなどの機能を実現するのに有用である．ML言語において一級継続の機能を搭載した処理系としては，Standard ML of New Jerseyがある．この処理系はスタックを用いず関数フレームをすべてヒープ領域に割り付けているため，定数オーダの時間で継続の生成を実現している．しかし，一般に関数フレーム生成時にスタックを用いた方が，高速であり，メモリ効率も良い．本研究では，スタックを用いているML処理系に対し，一級継続の実装を行った．この場合，継続の生成時にスタックの内容をヒープにコピーするスタック法が一般的である．実際，スタックを用いたScheme処理系で多く採用されている．この方法は動作が単純で実装しやすいなどの利点がある．一方，継続生成のたびにスタックのコピーが行われるので，生成時間，メモリ効率の点などで効率が悪いことがある．そこで本研究では，その効率化手法である遅延スタックコピー法を採用した．この手法は，継続を使用したプログラムの効率を高める一方，オブジェクトの生成時や代入時にオーバヘッドが存在するという欠点がある．そこで，コンパイル時にMLの型情報を利用して，そのオーバヘッドを軽減する手法を提案し，実装，評価を行った．

An Effective Implementation of First-class Continuations in a Stack-based ML System

Nobuhisa Minagawa,[†] Tomoharu Ugawa,[†] Masahiro Yasugi[†]
and Taiichi Yuasa[†]

A continuation is the remaining task at a point in a computation. In Scheme, a continuation can be captured as a first-class object. This is useful for implementing coroutines and multi-threading. Standard ML of New Jersey is an implemenation of ML with first-class continuations. This system allocates all activation records into the heap, and this feature allows capturing continuations in constant order time. In general, however, a stack is more efficent in terms of execution time and memory space to allocate activation records. We implemented first-class continuations in a stack-based ML system. In order to implement first-class continuations, we adapted the stack strategy which is used in many stack-based implementations of Scheme. With this strategy, the entire stack contents are copied into the heap whenever a continuation is captured. This strategy is simple and easy to implement, but capturing continuations requires a lot of time and memory space. Thus, we employed the lazy stack copying strategy, which is based on the stack strategy but more efficent. While this strategy is useful when creating continuations, there is an overhead when creating and modifying objects. We propose a technique to reduce this overhead by using the type information of ML at compile time, and we applied this techinique to the ML system and evaluated it.

（平成17年1月21日発表）

† 京都大学大学院情報学研究科通信情報システム専攻
Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University