発表概要

計算状態操作機構を備えたS式ベースJava言語の変換に基づく実装に向けて

西田 知広[1,a)]　八杉 昌宏[2,b)]　平石 拓[3]　小出 洋[4]

2022年3月18日発表

SC 言語処理系は S 式ベースの拡張 C 言語を標準 C 言語への変換に基づき実装可能としたもので，C 言語の実行スタック中に眠る呼び出し元の変数の値への合法的アクセスを提供する計算状態操作機構などを実現できる．入れ子関数の形態をとる場合，クロージャを生成し，それへのポインタを用いて間接的に入れ子関数を呼び出すことで，クロージャ生成時の環境に含まれる変数へのアクセスが可能となる．クロージャ初期化や変数値維持を遅延した変換に基づく実装には遅延判定コストの問題が見つかったが，入れ子関数を持つ呼び出し元への一時的な制御/文脈移動については，例外処理を用いることで効率化できる可能性がある．本研究では，例外処理の利用や高信頼性を目的として，新たに S 式ベース Java 言語処理系を整備している．Java 言語では，ヒープを消費するものの，final でない変数を配列要素等に置き換えれば，計算状態操作機構としてラムダ式が利用できる．加えて，非局所脱出することなく例外ハンドラを呼び出せるとした持続型例外処理機構を新たな形態の計算状態操作機構として設計している．本発表では，同機構を備えた S 式ベースの拡張 Java 言語の Java 言語への変換に基づく実装手法を提案する．同機構により呼び出し元の変数の値へのアクセスを要する負荷分散などが実現可能となる．

Presentation Abstract

Towards Transformation-based Implementations of an S-expression-based Java Language That Features Legitimate Execution Stack Access

Tomohiro Nishida[1,a)]　Masahiro Yasugi[2,b)]　Tasuku Hiraishi[3]　Hiroshi Koide[4]

Presented: March 18, 2022

The SC language system supports transformation-based implementations of S-expression-based extended C languages by translating them into the standard C language. Mechanisms for legitimate execution stack access (LESA for short) are the most important applications of the SC language system; LESA mechanisms provide legitimate access to values of callers' variables slept deeply in C's execution stack. When a nested function is employed as an LESA mechanism, a closure is created, and a pointer to it is used to indirectly call the nested function, we can access variables in an environment closed in the closure at the creation time. The previous study reveals that the delay judgment costs are considerable in transformation-based implementations which employ lazy initialization of closures and delayed maintenance of variable values; among the delay judgment costs, costs for temporary control transfer (with context restoration) to a caller that owns a nested function may be reduced by using exception handling. In this study, we are newly developing an S-expression-based Java language system for the use of exception handling and high reliability. In Java, although the heap is consumed, we can employ lambda expressions as LESA mechanisms if non-final variables are replaced with array elements or fields. In addition, we are designing restartable exception handling mechanisms as new LESA mechanisms where exception handlers can be called without non-local exits. In this presentation, we propose techniques for a transformation-based implementation of an S-expression-based Java language that features LESA mechanisms by translating it into the standard Java language. LESA mechanisms enable dynamic load balancing which requires legitimate access to values of callers' variables.

[1] 九州工業大学大学院情報工学府
Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka, Fukuoka 820–8502, Japan
[2] 九州工業大学大学院情報工学研究院
Department of Computer Science and Networks, Kyushu Institute of Technology, Iizuka, Fukuoka 820–8502, Japan
[3] 京都橘大学工学部情報工学科
Department of Information and Computer Science, Faculty of Engineering, Kyoto Tachibana University, Kyoto 607–8175, Japan
[4] 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University, Fukuoka 819–0395, Japan
[a)] nishida@pl.ai.kyutech.ac.jp
[b)] yasugi@csn.kyutech.ac.jp