코드에 자유변수가 있는 다단계 프로그램에서 자유변수 없애기*

어현준○ 이광근
서울대학교 전기컴퓨터공학부
poisson@ropas.snu.ac.kr kwang@ropas.snu.ac.kr

Transforming Open Code Multi-staged Programs into Closed One

Hyunjun Eo○ Kwangkeun Yi
School of Electrical Engineering and Computer Scienc, Seoul National University

다단계 프로그래밍은 계산을 여러 단계로 나누어 현재 단계에서는 다음 단계의 계산에 필요한 코드를 생성하고 마지막 단계에 전체적인 계산을 수행하도록 하는 프로그래밍 방법이다. 다단계 프로그래밍은 매크로 프로그래밍[10,3], 부분 계산(partial evaluation)[4,1], 실행시간 코드 생성 기법(run-time code generation)[6,8]등을 포섭하는 일반적인 방법론으로 주로 코드의 재사용, 부분 입력에 대한 전문화(specialization)등에 사용된다.

다단계 프로그래밍에서 계산은 크게 두 가지로 구분이 된다. 그 중 하나는 일반적인 계산을 수행하는 부분이고, 나머지 하나는 다음 단계의 계산에 필요한 코드를 생성하는 부분이다. 즉 3단계 프로그래밍을 통해 계산을 한다면 0단계는 1단계 코드를, 1단계는 2단계 코드를 생성하고, 마지막 2단계 코드를 실행함으로써 모든 계산이 완료된다. 이 과정에서 각 단계에 부분적으로 주어지는 입력들을 이용하여 다음 단계의 계산을 전문화(specialize)하여 단계를 나누지 않은 프로그램보다 더 효율적인 계산을 할 수 있도록 해 준다.

다단계 프로그래밍을 안전하게 하기 위해서, 다단계 프로그래밍 언어에 대한 타입 이론이 활발하게 연구되어져 왔다. 그 중 대표적인 것이 양상 논리(modal logic)를 이용한 타입 시스템이다. 가장 먼저, 1996년에 Davies와 Pfenning이 코드에 자유변수를 허용하지 않는 다단계 언어에 대한 타입 시스템을 제안하였다[2]. 그 후 코드에 자유변수를 허용하도록 다단계 언어의 타입 시스템을 만드는 연구가 10여년 동안 진행되어 오다가 2006년 Kim, Yi와 Calcagno에 의해 리습(Lisp)의 매크로를 완벽하게 지원하면서 자유변수를 허용하는 다단계 언어의 타입 시스템이 제안 되었다[5].

코드에 자유변수를 허용하는 타입 시스템을 이용한 프로그램은 자유변수를 허용하지 않는 프로그램에 비해 더 효율적인 코드를 생성할 수 있다. 코드에 자유변수를 허용하지 않으면 모든 코드 부품들이 닫힌(closed) 형태가 되어야 하므로 모든 자유변수를 입력으로 받는 함수의 형태로 만들어 주어야 하며, 이를 실행하는 데 있어서 자유변수를 허용하는 프로그램보다 비효율적일 수 밖에 없다.

Kim, Yi와 Calcagno에 의해 자유변수를 허용하는 다단계 언어와 그 타입 시스템이 제안되었지만[5], 이를 어떻게 구현할 수 있을 지에 대한 연구는 아직까지 진행되어진 바가 없다. 다단계 언어를 구현하는 방법으로는 다단계 언어를 실행시킬 수 있는 가상 기계를 고안하는 방법과 다단계 프로그램을 단계가 없는 보통의 프로그램으로 변환한 다음 그 프로그램을 실행시키는 방법 등이 있다.

본 논문은 자유변수를 허용하는 다단계 언어를 단계가 없는 보통의 프로그램을 변환하여 실행시키는 연구의 일 부분으로 시작되었다. 자유변수를 허용하지 않는 다단계 언어를 단계가 없는 보통의 프로그램으로 변환하는 방법은 Davies와 Pfenning에 의해 간략하게 제시된 바가 있다. 따라서, Davies와

* 본 연구는 교육인적자원부 두뇌한국21사업의 지원으로 수행되었음.

Pfenning의 방법과 본 논문에서 제안하는 변환 과정을 결합하여 자유변수를 허용하는 다단계 프로그램을 단계가 없는 보통의 프로그램으로 변환할 수 있다.

본 논문에서 제시하는 방법은 자유변수가 있는 함수를 자유변수가 없는 함수로 바꾸어 주는 방법인 클로져 변환(closure conversion)[7,9]과 매우 유사하다. 클로져(closure)란 함수와 그 함수가 정의될 때의 환경(environment)를 쌍(pair)으로 묶은 것을 말한다. 클로져 변환을 통해 함수는 환경을 레코드(record)의 형태로 입력 받도록 변환되고, 모든 자유변수는 입력으로 받은 환경의 필드(field)가 되어 함수 내에 자유변수가 더 이상 존재하지 않게 된다.

자유변수가 있는 다단계 프로그램을 자유변수가 없도록 변환할 때 가장 큰 문제점은 자유변수가 동적으로 바인딩(binding)되는것이다. 즉, 코드에 있는 자유변수는 코드가 정의될 때의 환경에 영향을 받는 것이 아니라 코드가 사용될 때의 환경에 영향을 받는 것이다. 따라서, 본 논문에서 제안하는 방법은 클로져 변환에서처럼 코드가 생성될 때의 환경을 코드에 넘겨주는 것이 아니라 코드가 사용될 때의 환경을 코드에 넘겨주도록 변환한다.

제안된 변환을 거친 프로그램은 코드에 자유변수가 없고, 변환되기 전과 동일한 결과를 계산한다. 제안된 변환을 구현하여 실제로 사용할 수 있도록 하는 것이 차후 과제이다.

참고문헌

[1] Oliver Danvy. Type-directed partial evaluation. In *Proceedings of the Symposium on Principles of Programming Languages*, pages 242-257, ACM, Jan 1996.

[2] Rowan Davies and Frank Pfenning. A modal analysis of staged computation. In *Proceedings of the Symposium on Principles of Programming Languages*, pages 258-270, ACM, Jan 1996.

[3] Paul Graham. *On Lisp: an advanced technique for Common Lisp*. Prentice Hall, 1994.

[4] Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. *Partial evaluation and automatic generation.* Prentice Hall, 1993.

[5] Ik-Soon Kim, Kwangkeun Yi, and Cristiano Calcagno. A polymorphic modal type system for Lisp-like multi-staged languages. In *Proceedings of the Symposium on Principles of Programming Languages*, pages 257-268, ACM, Jan 2006.

[6] M. Leone and Peter Lee. Optimizing ML with run-time code generation. In *Proceedings of the Conference on Programming Language Design and Implementation*, pages 137-148. ACM, Jun 1996.

[7] Yasuhiko Minamide, Greg Morrisett, and Robert Harper. Typed closure conversion. In *Proceedings of the Symposium on Principles of Programming Languages*, pages 271-283, ACM, Jan 1996.

[8] Massimilian Poletto, Wilson C. Hsieh, Dawson R. Engler, and M. Frans Kasshoek. C and tcc: a language and compiler for dynamic code generation. *ACM Transactions on Programming Languages and Systems*, 21:324-369, Mar 1999.

[9] Paul A. Steckler and Mitchell Wand. Lightweight closure conversion. *ACM Transactions on Programming Languages and Systems*, 19(1):48-86, Jan 1997.

[10] Guy L. Steele. *Common Lisp the Language, 2nd edition.* Digital Press, 1990.