함수 언어 Haskell과 Clean의 상용화 동향

변 석우
한국전자통신연구원
컴퓨터.소프트웨어연구소

최 광훈
한국과학기술원
전산학과

제 1 절 서 론

함수 언어는 견고한 이론을 기반으로 설계 구현되고 있으므로, 다른 부류의 언어들이 갖지 못하는 많은 장점을 가지고 있다. 그러나 함수 언어는 실용적으로 아직 활발하게 사용되지 못하고 있는 실정이다. 함수 언어의 '이론적 장점'과 '비실용성' 사이의 괴리를 해소하기 위해서 극복해야 할 주요 난관은 함수 언어 '컴파일러의 저성능'과 'side-effect 프로그래밍의 부적합성'의 문제였다.

최근 하드웨어의 빠른 성장과 함수 언어 컴파일러 기법의 발전으로 함수 언어 컴파일러의 저성능의 문제는 많이 완화되었다고 본다. 예를 들어, 함수 언어 Clean의 성능은 C와 견줄 수 있을 정도로 빠르다. 함수 언어를 이용한 side-effect 프로그래밍 연구의 핵심은 함수 프로그래밍 고유의 수학적 순수성 (예를 들어, equational reasoning과 referential transparency 특성 등)을 유지하면서 입출력 및 인터페이스에 대한 이론적 기반을 정립하고 이것을 구현하는 데 있다. 최근 category 이론을 응용한 Monad와 선형 논리의 개념을 응용한 Clean의 Unique 타입이 각각 함수 언어 Haskell과 Clean에서 구현되었다. 이 기능들 덕택으로 함수 프로그래밍에서 side-effect가 쉽게 표현할 수 있게 되었다.

한편, 컴퓨터 및 관련 시스템 환경이 급격히 바뀌고 있다. 하드웨어 가격 대비 소프트웨어의 가격이 상승하고 있으며, 인터넷을 이용하는 응용 프로그램이 등장함에 따라 시스템의 안전성 및 보안 등이 중요한 이슈로서 등장하고 있다. 이러한 환경의 변화는 함수 프로그래밍 언어의 장점들을 더욱 부각시키고 있으며, 함수 언어 상용화에 대한 관심을 고조시키고 있다. 함수 언어의 상용화가 성공된다면 이것은 컴퓨터 분야에서 '학문'과 '실용' 사이의 거리를 단축시키는 중요한 공헌을 하는 셈이며, 산업과 연구 전반에 걸쳐 큰 영향을 미칠 것으로 예상한다.

함수 언어 상용화에 대한 여러 연구들 중에서, 본 고에서는 함수 언어 Haskell을 일반 대중이 사용할 수 있도록 상용화하기 위한 연구와, Clean의 상품화에 대하여 소개하려 한다. 좀 더 구체적으로 관심이 있는 독자들은 제시된 참고 문헌들을 참고하기 바라며, 본 고에서는 기본적인 사항들에 대해서만 언급하기로 한다.

제 2 절 Haskell의 상용화 연구 동향

Haskell 위원회는 순수 지연 함수 언어를 위한 표준화 제시와 최신 기술의 구현을 목적으로 1980년대 중반 조직되었다. 이 위원회에는 Simon Peyton-Jones (영국 Glasgow 대학), Paul Hudak (Yale 대학), Philip Wadler (Bell Lab.), John Hughes (스웨덴 Chalmers 대학)를 비롯한 많은 저명한 학자들이 참여하고 있다. 새로운 기술들이 꾸준히 구현되어, 현재 버전 1.4가 배포 중이다 (`http://haskell.org`).

1 스크립트 언어로서의 Haskell

Peyton-Jones (영국 Glasgow 대학), Mark Jones(영국 Nottingham 대학), Erik Meijer (네덜란드 Utrecht 대학)은 공동으로 영국 정부에서 지원하는 EPSRC 프로젝트 *First-class modules for component-based programming*을 수행하고 있다. 이 과제의 주요 이슈는 component-based programming에서의 타입과 모듈의 역할 및 이들을 이용하는 응용 프로그래밍 개발에 관한 것이다. 좀 더 구체적으로, 이들의 목적은 함수 언어의 추상화 기능과 타입의 특성을 이용하여 함수 언어를 스크립트 프로그래밍 언어로서 사용하기 위한 연구를 수행하고 있다.

Meijer는 Haskell의 타입, lazy evaluation, 고계 함수 (higher-order function)의 기능 덕택으로 Haskell은 Tcl과 Perl 등의 스크립트 언어보다도 훨씬 우수한 스크립트 언어의 역할을 할 수 있다고 주장한다 [6]. 특히, 그는 Haskell의 이러한 특성을 Mircrosoft의 COM/AxtiveX의 환경에서 구현함으로써 Haskell을 대중화시키고, Visual Basic을 능가하는 일반 대중용 프로그래밍 언어로서 발전시키려는 노력을 하고 있다. 스크립트 언어로서 갖추어야 할 특성, 함수 언어의 장점, 함수 언어의 상용화에 대한 걸림돌 등에 관한 그의 주장이 소개되어 있다. 향후 약 2-3 년에 걸쳐 이들의 아이디어가 실현될 수 있는 지 주목할 필요가 있다고 생각한다.

2 Haskell에서의 foreign language interface

어떤 한 프로그래밍 언어가 모든 응용 분야에서 가장 최적화된 언어의 역할을 할 수는 없다. 새로운 프로그램 언어가 널리 사용되기 위해서는 기존에 개발된 많은 프로그램을 이용할 수 있도록 하는 foreign language interface 기능이 필수적으로 요구되고 있다. 이 기능은 위에서 소개한 component-based programming을 가능케 하는 중요한 요소이다. 인터페이스 기능을 통해서 특정 프로그래밍 언어의 표현력이나 효율성을 극복할 수 있고 다른 프로그래밍 언어로 작성된 모듈을 쉽게 이용할 수 있다. 이러한 인터페이스를 위해서는 각 프로그래밍 언어의 상호 호출 방법, 주고 받을 값의 표현 방법, 메모리 할당 전략 및 변환 방법이 정의되어야 한다.

Glasgow 대학에서 foreign language interface를 위한 코드 생성을 위해 Haskell을 확장하고[1], IDL로 기술된 인터페이스로부터 Haskell로 기술된 인터페이스를 만들어내는 도구 H/Direct를 개발하였다[2][5].

현재 Visual Basic, Visual C++, Java등의 언어에서도 유사한 도구가 이미 개발되어 COM 객체를 다룰수 있다. Haskell에서는 다른 언어와 구분되는 특징을 통해 COM 객체를 보다 자유롭게 다룰수 있다. 고계 함수 기능과 lazy evaluation과 같은 확장된 제어구조를 이용하여 보다 세밀한 모듈을 구성하여 COM 객체를 다룰수 있고[4][5], COM 객체를 통한 입출력 기능을 정수나 실수 타입의 데이터와 같이 자유롭게 함수의 인자로 넘기고 결과 값으로 받을 수 있고 리스트(list)나 트리(tree)와 같은 임의의 데이터 구조에 저장하여 다룰수 있는 장점이 있다[3].

H/Direct를 이용해 Haskell과 잠재적으로 모든 프로그래밍 언어와 쉽게 연결할 수 있어 Haskell의 응용 분야를 한층 넓일 것이고, Haskell의 다양한 특징을 보다 폭넓게 시험할 수 있을 것이다.

제 3 절 Clean 언어의 상품화

Clean은 다른 함수 언어와는 달리 이론적 기반이 람다 계산법 (lambda calcululs)이 아닌 TGRS (term graph rewriting systems)에 근거를 두고 있다. 여기서 이론적 기반이라 함은 TGRS가 개념적인 모델링의 역할 뿐만아니라 실제 구현에 있어서 추상 머신 (abstact machine)의 역할까지도 수행함을 의미한다. 즉, Clean의 컴파일 과정에서 TGRS의 변환 기법이 적용되고 있다.

Graph rewriting 기술은 1972년 Wadsworth의 Oxford 대학 박사학위 논문에서 처음 소개되었다. 그는 람다 계산법을 구현할 때 중복되는 람다 텀을 그래프 형태로 표현함으로써 동일한 텀의 복사와 중복된 계산 과정을 방지할 수 있는 방법을 제안하였다. 그 후 Staples가 이 개념을 TRS에서 적용될 수 있도록 확장하였고 (1979년), East Anglia 대학의 Sleep교수 팀에서 처음 TGRS 기반의 함수 언어 DACTL을 구현함으로써 실용화되었다. East Anglia 대학과 Nijmegen 대학은 그 후 공동으로 Dactl의 후속 버전인 LEAN (Language of East Anlia and Nijmegen)을 개발하였고, Nijmegen 대학은 그 후 Lean을 발전시켜 Clean을 개발하였다.

Clean은 네덜란드의 Katholic University of Nijmegen 대학의 Plasmeijer 교수와 van Eekelen 교수 팀에 의해서 구현되었다.(이 대학에는 또한 람다 계산법 분야의 권위자인 Barendregt 교수가 재직하고 있다.) Nijmegen 대학은 지난 약 10년 동안 Clean의 상용화를 위한 꾸준한 연구를 수행한 결과 Clean을 상품화하게 되었다. 이제 Clean은 Hilt 사의 상품으로서 교육용으로는 바이너리 화일이 무상으로 제공되고 있지만, 일반 회사에게는 그렇지 않다. 함수 언어가 연구용 toy의 범위를 넘어, 실용적으로 여러 응용 분야에서 사용될 수 있음은 함수 언어의 발전을 위해서 매우 고무적인 일이다. 이미 약 10년 전에 Miranda (Miranda는 Research Software 사의 상품임)가 유사한 목적으로 상품화된 적이 있으나, 교육 분야를 제외하고 실제 응용된 분야는 많지 않았다. Clean은 Miranda와는 달리 인터프리터 뿐만 아니라 컴파일러를 제공하고 있으며, 컴파일러의 성능이 C 컴파일러의 성능에 견줄 수 있을 만큼 빠르고, Unique 타입을 이용하여 입출력 및 side-effect 프로그래밍을 효율적으로 지원할 수 있는 장점이 있다. 이미 side-effect를 이용하는 많은 데모용 프로그램들이 Clean 으로서 개발되었다.

자세한 사항은 `http://www.cs.kun.nl/~clean`를 참조하기 바라며, 기술적인 사항들은 [7]와 [8]을 참조하기 바란다.

참고 문헌

[1] Sigbjorn Finne et al. A primitive foreign function interface for haskell, March 1998.

[2] Sigbjorn Finne, Daan Leijen, Erik Meijer, and Simon Peyton Jones. H/direct: A binary foreign language interface for haskell. In *International Conference on Functional Programming*, Baltimore, USA, September 1998. `http://www.dcs.gla.ac.uk/fp/software/hdirect/`.

[3] Paul Hudak and John Peterson. A gentle introduction to haskell 1.4, March 1997.

[4] John Hughes. Why functional programming matters. *Computer Journal*, 32(2):98–107, 1989.

[5] Simon Peyton Jones and Erik Meijer. Scripting com components in haskell. In *Proceedings of the Fifth International Conference on Software Reuse*, Victoria, Birtish Columbia, June 1998.

[6] Erik Meijer. Functional programming: Higher level scripting, for the 21th centry. `http://www.cse.ogi/~erik/Personal/Papers/scripting.dvi`.

[7] Rinus Plasmeijer and Marcko van Eekelen. *Functional Programming and Parallel Graph Rewriting.* Addison Wesley, 1993.

[8] M.R. Sleep, M.J. Plasmeijer, and M. van Eekelen, editors. *Term Graph Rewriting: Theory and Practice.* John Wiley & Sons, 1993.