최적화 컴파일러에 맞춘 프로그램 오류 검증의 변환*

양홍석 이광근°
서울대학교 컴퓨터공학부
{hyang, kwang°}@ropas.snu.ac.kr

Proof Transformation for Source-level Optimization

Hongseok Yang and Kwangkeun Yi°
School of Computer Science and Engineering, Seoul National University

요 약

고급 언어(high-level language)로 프로그램을 쓰고 그것이 맞다고 증명했을때, 프로그램과 증명을 동시에 "컴파일"해주는 방법을 찾으려고 한다. 이러한 방법은 "증명 보내기"(Proof Carrying Code)라는 제안을 실용화할 때 요긴하게 사용할 수 있는 기술이다. 지면관계상 "반복문에서 명령어 끄집어내기"(Loop Invariant Code Motion)라는 최적화 과정에 맞게 Hoare 증명을 변환하는 방법에 대해서만 알아보겠다.

1 서론

인터넷을 통해 프로그램을 얻어 사용하면서, 어떻게 하면 제 3자로부터 얻은 프로그램을 안심하고 실행할 수 있겠는가라는 문제가 생겼다. 현재 이 문제에 대한 해법으로 Necula가 제시한 "증명 보내기"(Proof Carrying Code)라는 방법이 주목을 받고 있다 [1]. 이 방법의 핵심은 프로그램을 만든 사람이 "프로그램이 문제없이 돌아간다"는 증명까지 동시에 제공하는데 있다. 이렇게 증명이 따라온다면, 프로그램을 안심하고 사용하기 위해서 단지 딸려오는 증명에 허점이 없는지만 검사하면 된다. 그 프로그램을 만든 사람이 누구인지도 알 필요도 없다.

하지만, "증명 보내기"라는 방법을 실제에서 이용하기에는 아직 미흡하다. 왜냐하면 "어떻게 증명을 할 것인가"라는 문제에 적절한 해답을 찾지 못했기 때문이다. 현재까지 알려진 가장 좋은 해결책이 타입을 이용하자는 것인데, 타입만 가지고 증명할 수 있는 프로그램의 성질이 제한되어 있어서, 이 해결책을 가지고 "증명 보내기"를 실용화하기에는 아직 부족하다 [2].

본 연구에서 다른 방식으로 증명을 만드는 법을 제안하겠다. 이 방법의 핵심은 프로그램에 대한 증명을 "컴파일"하는데 있다. 인터넷을 통해 돌아다니는 프로그램의 대부분이 고급 언어(high-level language)로 우선 작성되고, 컴파일러를 통해 저급 언어(low-level language)로 바뀐 후에, 사용자에게 제공된다. 따라서, 이러한 프로그램에 대한 증명은 고급언어로 짜여진 프로그램에 대한 증명과 컴파일러가 오류없이 작동했다는 증명, 이렇게 두 가지로 이루어져있다. 본 연구의 목표는 두번째 증명을 자동으로 생성하는 것이다. 즉, 프로그래머가 고급언어로 프로그램을 작성하고 이 프로그램이 올바르게 작동한다는 증명도 하면, 컴파일러가 프로그램과 증명을 동시에 변환한다. 이러한 과정을 거쳐 저급언어로 작성된 프로그램과 저급 언어 프로그램에 대한 증명을 얻을 수 있다. 여기에서, 저급 언어 프로그램에 대한 증명은 결국 프로그래머가 만든 증명에 컴파일 과정이 맞다는 증명이 첨가된 것으로 볼 수 있다.

이 논문에서는 지면관계상 컴파일 과정 중의 일부만 살펴보려고 한다. 즉, "반복문에서 명령어 끄집어내기"(Loop Invariant Code Motion)라는 최적화 과정에 맞게 증명을 변환하는 방법에 대해 알아보겠다.

*본 연구는 한국과학재단 목적 기초연구(R08-2003-000-10370-0)와 두뇌한국 21의 지원을 받아 수행되었다.

2 WHILE 언어와 Hoare의 프로그램 증명 기술

WHILE 언어는 가장 간단한 명령형 언어로서, 다음과 같은 명령어들로 이루어져있다.

$$C ::= x := e \mid \texttt{skip} \mid C; C \mid \texttt{repeat}\ C\ b \mid \texttt{if}\ b\ C\ C$$

여기에서, e는 정수값을 가지는 정수식(integer expression)이고 b는 참과 거짓 중 하나의 값을 가지는 분별식(boolean expression)이다. 또한, $\texttt{repeat}\,C\,b$는, C를 우선 실행하고 나서 b가 참이면 실행을 마치고, 그렇지 않으면 $\texttt{repeat}\,C\,b$를 반복해서 수행하라는 명령어이다. 흔히 쓰이는 while대신에, repeat을 사용한 이유는, 뒤에 설명할 컴파일 과정을 더욱 간단하게 설명할 수 있기 때문이다.

Hoare의 증명 기술에서, 프로그램에 대한 요구사항은, Hoare 삼항식(triple)이라고 불리는, $\{P\}C\{Q\}$로 주어진다. 여기에서 P와 Q는 1차 논리(first-order logic)의 서술식(formula)으로, P는 프로그램 C를 수행하기 전에 어떤 조건이 만족해야하는지를 서술하고 있고, Q는 프로그램이 끝나면 무엇이 만족하는가를 서술하고 있다. 즉, Hoare 삼항식 $\{P\}C\{Q\}$는 "P가 성립할 때, 프로그램 C를 수행하면 Q를 만족하는 상태에서 수행이 끝난다"를 의미한다.

Hoare 삼항식들을 증명하는 규칙들은 다음과 같다.

$$\text{CONSEQUENCE}\ \frac{P' \Rightarrow P \quad \{P\}C\{Q\} \quad Q \Rightarrow Q'}{\{P'\}C\{Q'\}} \qquad \text{ASSIGNMENT}\ \frac{}{\{P[^{e}/_{x}]\}\,x := e\,\{P\}}$$

$$\text{SKIP}\ \frac{P \Rightarrow Q}{\{P\}\,\texttt{skip}\,\{Q\}} \qquad \text{CONDITIONAL}\ \frac{\{P \wedge b\}C_1\{Q\} \quad \{P \wedge \neg b\}C_2\{Q\}}{\{P\}\,\texttt{if}\,b\,C_1\,C_2\,\{Q\}}$$

$$\text{SEQUENCING}\ \frac{\{P\}C_1\{Q\} \quad \{Q\}C_2\{R\}}{\{P\}C_1;C_2\{R\}} \qquad \text{LOOP}\ \frac{\{P \vee (Q \wedge \neg b)\}C\{Q\}}{\{P\}\,\texttt{repeat}\,C\,b\,\{Q \wedge b\}}$$

위의 Hoare 규칙들과 1차 논리(first-order logic)의 규칙들을 이용하면 프로그램을 증명할 수 있다. 이렇게 증명하는 과정에서 가지 구조(tree)가 생기는데, 이것을 Hoare 증명이라고 부르겠다.

3 Hoare 증명을 기본꼴로 변환하는 방법

Hoare 증명을 기본꼴로 바꾸는 것이 모든 변환 방법의 시작이다. 본 장에서는 이렇게 기본꼴로 바꾸는 방법에 대해서 알아보겠다.

$\{P\}x := e\{Q\}$에 대한 증명은 아래와 같은 기본꼴로 바꿀 수 있다.

$$\frac{P \overset{\tau}{\Rightarrow} Q[^e/_x] \quad \overline{\{Q[^e/_x]\}x := e\{Q\}}}{\{P\}x := e\{Q\}}$$

왜냐하면, 일반적으로 $\{P\}x := e\{Q\}$에 대한 증명 σ는 다음과 같은 형태를 지니는데,

$$\frac{P \overset{\tau_1'}{\Rightarrow} Q'[^e/_x] \quad \overline{\{Q'[^e/_x]\}x := e\{Q'\}} \quad Q' \overset{\tau_2'}{\Rightarrow} Q}{\{P\}x := e\{Q\}}$$

여기에서부터, 아래와 같이 기본꼴의 증명을 얻을 수 있기 때문이다.

$$\frac{\dfrac{P \overset{\tau_1'}{\Rightarrow} Q'[^e/_x] \quad \dfrac{Q' \overset{\tau_2'}{\Rightarrow} Q}{Q'[^e/_x] \Rightarrow Q[^e/_x]}}{P \Rightarrow Q[^e/_x]} \quad \overline{\{Q[^e/_x]\}x := e\{Q\}}}{\{P\}x := e\{Q\}}$$

본 논문에서는 $\{P\}x := e\{Q\}$에 대한 기본꼴 증명을 다음과 같이 간략하게 나타낼 것이다.

$$\frac{P \overset{\tau}{\Rightarrow} Q[^e/_x]}{\{P\}x := e\{Q\}}$$

다음으로, $\{P\}C_1;C_2\{Q\}$에 대한 증명의 기본꼴은 아래와 같다.

$$\frac{\dfrac{P \overset{\tau_1}{\Rightarrow} P' \quad \{P'\}C_1\{R\}^{\sigma_1}}{\{P\}C_1\{R\}} \quad \dfrac{R \overset{\tau_2}{\Rightarrow} R' \quad \{R'\}C_2\{Q'\}^{\sigma_2} \quad Q' \overset{\tau_3}{\Rightarrow} Q}{\{R\}C_2\{Q\}}}{\{P\}C_1;C_2\{Q\}}$$

$\{P\}C_1;C_2\{Q\}$에 대한 증명의 일반적인 형태는 다음과 같은데,

$$\delta_1 \overset{\text{def}}{=} \frac{P'' \overset{\tau_1'}{\Rightarrow} P' \quad \{P'\}C_1\{R\}^{\sigma_1'} \quad R \overset{\tau_2'}{\Rightarrow} R''}{\{P''\}C_1\{R''\}}$$

$$\delta_2 \overset{\text{def}}{=} \frac{R'' \overset{\tau_3'}{\Rightarrow} R' \quad \{R'\}C_2\{Q'\}^{\sigma_2'} \quad Q' \overset{\tau_4'}{\Rightarrow} Q''}{\{R''\}C_2\{Q''\}}$$

$$\frac{P \overset{\tau_5'}{\Rightarrow} P'' \quad \dfrac{\{P''\}C_1\{R''\}^{\delta_1} \quad \{R''\}C_2\{Q''\}^{\delta_2}}{\{P''\}C_1;C_2\{Q''\}} \quad Q'' \overset{\tau_6'}{\Rightarrow} Q}{\{P\}C_1;C_2\{Q\}}$$

여기에서, 다음과 같이 기본꼴을 지니는 증명을 얻을 수 있다.

$$\delta_1 \overset{\text{def}}{=} \frac{\dfrac{P \overset{\tau_5'}{\Rightarrow} P'' \quad P'' \overset{\tau_1'}{\Rightarrow} P'}{P \Rightarrow P'} \quad \{P'\}C_1\{R\}^{\sigma_1'}}{\{P\}C_1\{R\}}$$

$$\delta_1 \overset{\text{def}}{=} \frac{\dfrac{R \overset{\tau_2'}{\Rightarrow} R'' \quad R'' \overset{\tau_3'}{\Rightarrow} R'}{R \Rightarrow R'} \quad \sigma_2' \quad \dfrac{Q' \overset{\tau_4'}{\Rightarrow} Q'' \quad Q'' \overset{\tau_6'}{\Rightarrow} Q}{Q' \Rightarrow Q}}{\{R\}C_2\{Q\}}$$

$$\frac{\{P\}C_1\{R\}^{\delta_1} \quad \{R\}C_2\{Q\}^{\delta_2}}{\{P\}C_1;C_2\{Q\}}$$

4 프로그램 최적화할 때 Hoare 증명을 변환하는 방법

이장에서는, "반복문에서 명령어 끄집에 내기"(Loop Invariant Code Motion)라는 기법에 따라서 프로그램을 최적화할 때, 입력 프로그램에 대한 Hoare 증명을 어떻게 변환해야하는가라는 문제를 다루겠다.

"반복문에서 명령어 끄집에 내기"란, 반복문 "repeat $(C_1;C)\,b$"의 C_1이 항상 같은 일을 할 경우에, C_1을 밖으로 끄집에 내어서, "C_1;(repeat $C\,b$)"로 전체 프로그램을 바꾸는 기법이다. 이렇게 바뀐 프로그램은 C_1을 한번만 수행해도 되므로, 빠르게 수행한다. 이 최적화 기법을 정확하게 기술하면 다음과 같다.

$$(\texttt{repeat}\,(x := e; C)\,b) \;\rightsquigarrow\; (x := e; \texttt{repeat}\,C\,b)$$

단, $\mathsf{Free}(x := e) \cap \mathsf{Modifies}(C_1) = \emptyset$ 와 $x \notin \mathsf{Free}(e)$ 이 성립할때만, 이 기법을 적용할 수 있다.

"반복문에서 명령어 끄집에 내기"에 따라서 Hoare 증명을 변환할 때, 가장 중요한 부분은, (repeat $(x := e; C)\,b$)의 C가 $x = e$라는 사실을 보존한다고 보이는 것이다. 이와 같은 경우를 다루는 일반적인 증명 변환 방법을 다음 명제에서 알아보겠다.

명제 1 $\mathsf{Free}(R) \cap \mathsf{Modifies}(C) = \emptyset$*가 성립할때,* $\{P\}C\{Q\}$*에 대한 증명* σ*를* $\{P \wedge R\}C\{Q \wedge R\}$*에 대한 증명* σ'*으로 바꾸는 알고리즘이 존재한다.*

증명: $\{P\}C\{Q\}$에 대한 증명 σ를 만들때 사용한 Hoare의 방법들을 살펴보면서, 각각의 경우를 알고리즘 T가 어떻게 바꾸는지 설명하겠다.

CONSEQUENCE:

$$\frac{P' \overset{\tau_1}{\Rightarrow} P \quad \{P\}C_1\{Q\}^{\sigma_1} \quad Q \overset{\tau_2}{\Rightarrow} Q'}{\{P'\}C_1\{Q'\}}$$

$$\rightsquigarrow$$

$$\frac{\dfrac{P' \overset{\tau_1}{\Rightarrow} P}{P' \wedge R \Rightarrow P \wedge R} \quad T(\sigma_1) \quad \dfrac{Q \overset{\tau_2}{\Rightarrow} Q'}{Q \wedge R \Rightarrow Q' \wedge R}}{\{P' \wedge R\}C_1\{Q' \wedge R\}}$$

ASSIGNMENT: $R[^e/_x] = R$라는 사실을 이용하여 이 경우를 다룬다.

$$\overline{\{P[^e/_x]\}x := e\{P\}} \;\rightsquigarrow\; \overline{\{P[^e/_x] \wedge R\}x := e\{P \wedge R\}}$$

SKIP:

$$\frac{P \overset{\tau_1}{\Rightarrow} Q}{\{P\}\texttt{skip}\{Q\}} \;\rightsquigarrow\; \frac{P \overset{\tau_1}{\Rightarrow} Q}{\dfrac{P \wedge R \Rightarrow Q \wedge R}{\{P \wedge R\}\texttt{skip}\{Q \wedge R\}}}$$

CONDITIONAL:

$$\frac{\overset{\sigma_1}{\{P\wedge b\}C_1\{Q\}} \quad \overset{\sigma_2}{\{P\wedge\neg b\}C_2\{Q\}}}{\{P\}\texttt{if}\ b\ C_1\ C_2\{Q\}}$$

$$\rightsquigarrow$$

$$\frac{\overset{T(\sigma_1)}{\{P\wedge b\wedge R\}C_1\{Q\wedge R\}} \quad \overset{T(\sigma_2)}{\{P\wedge\neg b\wedge R\}C_2\{Q\wedge R\}}}{\{P\wedge R\}\texttt{if}\ b\ C_1\ C_2\{Q\wedge R\}}$$

SEQUENCING:

$$\frac{\overset{\sigma_1}{\{P\}C_1\{P_1\}} \quad \overset{\sigma_2}{\{P_1\}C_2\{Q\}}}{\{P\}C_1;C_2\{Q\}}$$

$$\rightsquigarrow$$

$$\frac{\overset{T(\sigma_1)}{\{P\wedge R\}C_1\{P_1\wedge R\}} \quad \overset{T(\sigma_2)}{\{P_1\wedge R\}C_2\{Q\wedge R\}}}{\{P\wedge R\}C_1;C_2\{Q\wedge R\}}$$

LOOP:

$$\frac{\overset{\sigma_1}{\{P\vee(Q\wedge\neg b)\}C_1\{Q\}}}{\{P\}\texttt{repeat}\ C_1\ b\{Q\wedge b\}}$$

$$\rightsquigarrow$$

$$\frac{\dfrac{}{(P\wedge R)\vee(Q\wedge R\wedge\neg b)\Rightarrow(P\vee Q\wedge\neg b)\wedge R} \quad \overset{T(\sigma_1)}{\{(P\vee Q\wedge\neg b)\wedge R\}C_1\{Q\wedge R\}}}{\dfrac{\{(P\wedge R)\vee(Q\wedge R\wedge\neg b)\}C_1\{Q\wedge R\}}{\{P\wedge R\}\texttt{repeat}\ C_1\ b\{Q\wedge b\wedge R\}}}$$

□

이제까지 알아낸 변환 기법을 종합하여, $\{P'\}\texttt{repeat}\,(x{:=}e;C)\,b\{Q'\}$에 대한 증명을 바꾸어보자. 일반적으로 $\{P'\}\texttt{repeat}\,(x{:=}e;C)\,b\{Q'\}$에 대한 Hoare 증명은 다음과 같은 꼴을 지닌다.

$$\frac{\overset{\tau}{P'\Rightarrow P} \quad \{P\}\texttt{repeat}\ (x:=e;C)\ b\{Q\wedge b\} \quad \overset{\tau'}{(Q\wedge b)\Rightarrow Q'}}{\{P'\}\texttt{repeat}\ (x:=e;C)\ b\{Q'\}}$$

따라서, σ를 변환하여 $\{P\}x{:=}e;(\texttt{repeat}\,C\,b)\{Q\wedge b\}$에 대한 증명을 만들어야 한다.

우선, 3장에서 살펴본 변환들을 사용하여, σ를 다음 꼴을 가지도록 바꾼다.

$$\frac{\dfrac{\overset{\tau_1}{(P\vee(Q\wedge\neg b))\Rightarrow S[^e/_x]} \quad \overset{\sigma_1}{\{S\}C\{Q\}}}{\{P\vee(Q\wedge\neg b)\}x:=e\{S\}}}{\{P\vee(Q\wedge\neg b)\}x:=e;C\{Q\}}$$

이제 본장의 앞에서 살펴본 변환을 적용하여, $\{x=e\wedge\exists x.P\}\texttt{repeat}\,C\,b\{Q\wedge x=e\wedge\neg b\}$의 증명을 만들자. 앞에서 살펴본 변환을 이용하면, C가 $x=e$를 보존한다 사실을 $\{S\}C\{Q\}$에 덧붙일 수 있다. 즉, σ_1을 변형하여,

$$\overset{\sigma_1'}{\{S\wedge x=e\}C\{Q\wedge x=e\}}$$

을 얻을 수 있다.

또한, 1차 논리의 규칙들과 $x\notin\mathsf{Free}(S[^e/_x])$을 사용하면, "$(P\vee(Q\wedge\neg b))\Rightarrow S[^e/_x]$"의 증명 τ_1에서부터, "$((x{=}e\wedge\exists x.P)\vee(Q\wedge x{=}e\wedge\neg b))\Rightarrow S\wedge x{=}e$" 의 증명 τ_1'을 아래와 같이 얻을 수 있다.

$$\delta_1 \overset{\text{def}}{=} \frac{\dfrac{\dfrac{\overset{\tau_1}{(P\vee(Q\wedge\neg b))\Rightarrow S[^e/_x]}}{P\Rightarrow S[^e/_x]}}{(\exists x.P)\Rightarrow S[^e/_x]}}{(x=e\wedge\exists x.P)\Rightarrow(S[^e/_x]\wedge x=e)}$$

$$\frac{}{(x=e\wedge\exists x.P)\Rightarrow(S\wedge x=e)}$$

$$\delta_2 \overset{\text{def}}{=} \frac{\dfrac{\dfrac{\overset{\tau_1}{(P\vee(Q\wedge\neg b))\Rightarrow S[^e/_x]}}{(Q\wedge\neg b)\Rightarrow S[^e/_x]}}{(Q\wedge x=e\wedge\neg b)\Rightarrow(S[^e/_x]\wedge x=e)}}{(Q\wedge x=e\wedge\neg b)\Rightarrow(S\wedge x=e)}$$

$$\frac{\overset{\delta_1}{(x{=}e\wedge\exists x.P)\Rightarrow(S\wedge x{=}e)} \quad \overset{\delta_2}{(Q\wedge x{=}e\wedge\neg b)\Rightarrow(S\wedge x{=}e)}}{((x{=}e\wedge\exists x.P)\vee(Q\wedge x{=}e\wedge\neg b))\Rightarrow(S\wedge x{=}e)}$$

위의 σ_1'과 τ_1'을 이용하여, $\{x=e\wedge\exists x.P\}\texttt{repeat}\,C\,b\{Q\wedge x=e\wedge\neg b\}$의 증명 σ_2'을 얻는다.

$$\frac{\dfrac{\overset{\tau_1'}{(x{=}e\wedge\exists x.P)\vee(Q\wedge x{=}e\wedge\neg b)\Rightarrow S\wedge x{=}e} \quad \overset{\sigma_1'}{\{S\wedge x{=}e\}C\{Q\wedge x{=}e\}}}{\{(x=e\wedge\exists x.P)\vee(Q\wedge x=e\wedge\neg b)\}C\{Q\wedge x=e\}}}{\{x=e\wedge\exists x.P\}\texttt{repeat}\,C\,b\{Q\wedge x=e\wedge b\}}$$

마지막으로, $\{P\}x{:=}e;(\texttt{repeat}\,C\,b)\{Q\wedge b\}$의 증명을 아래와 같이 만든다.

$$\frac{\dfrac{\dfrac{}{(\exists x.P)\Rightarrow e{=}e\wedge\exists x.P} \quad \overset{\sigma_2'}{\{\exists x.P\}x{:=}e\{x{=}e\wedge\exists x.P\}\ \{x{=}e\wedge\exists x.P\}\texttt{repeat}Cb\{Q\wedge x{=}e\wedge b\}}}{\{\exists x.P\}x{:=}e;(\texttt{repeat}\,C\,b)\{Q\wedge x{=}e\wedge b\}}}{\{P\}x{:=}e;(\texttt{repeat}\,C\,b)\{Q\wedge b\}}$$

5 결론

본 논문에서, "반복문에서 명령어 끄집어 내기"(Loop Invariant Code Motion)에 맞게 Hoare 증명을 변환하는 방법을 알아보았다. 현재, "더 이상 사용하지 않는 명령어 제거하기"(Faint Code Elimination), "쓰기 명령어의 순서 바꾸기"(assignment sinking)등의 최적화 과정에 맞는 Hoare 증명 변환 방법을 찾아낸 상태이고, 그 외의 컴파일 과정에 맞는 변환은 살펴보고 있는 중이다.

참고 자료

[1] G. Necula. Proof-Carrying Code. In *Proceedings of ACM Symposium on Principles of Programming Languages*. ACM, January 1997.

[2] G. Morrisett, D. Walker, K. Crary, and N. Glew. From System F to Typed Assembly Language. *ACM Transactions on Programming Languages and Systems*, 21(3):528–569, May 1999.