The efficiently implimentation of a programming language based on type theory

Atsushi Hikichi (010092)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 2002

Keywords: type directed compiler, A-normal form, Logical Abstract Machine.

1 Introduction

A compiler has been an ad hoc construction up to this time. Though it has compiled in the low-grade language from the high-class language, the error may have been included at the time of execution. It has turned out a relation between a logic and a compiler in recent years. That is the relation between a proposition and a type, and the relation between a proof and a program. If P is the proof of Proposition A in a logic, we can say that P is a program which calculates Type A in a programming language. The relation of this logic and programming language is applied to construction of a compiler. It became possible to analyse systematically. In this paper, We perform analysis and implimentation of a compiler to use that approach. The existing research, its problem, and the purpose of this research are described below.

2 Existing research and its problem

A compiler means the program changed into a target language from a sauce language through an intermediate language.

Existing research of some compiler are the compiler using CPS as an intermediate language [Appel et al : 89] and the compiler using A-normal form as an intermediate language [Flanagan et al.: 93]. Appel defined the compiler using CPS. CPS is one of the intermediate language to have been contained continuation and closure. The fault of CPS is that the amount of codes increases by taking continuation to an argument. The other side, Flanagan defined the compiler using A-normal form. A-normal form is one of the intermediate language to have a feature that an intermediate value is named. The advantage of A-normal form is that the amount of codes don't increases. Therefore, A-normal form can obtain good conversion rather than CPS. However, these compilers target the language without a type, they cannot apply a relation with logic.

Research of a compiler with type has been made to it recently. One of the research is type directed compiler [Morrisett et al.: 98]. That research use CPS with type. However,

we know that the direction of the compiler using A-normal form with type gives better conversion. So we make a design and an implementation of compiler based on the A-normal form with type.

It can be regarded as the same thing.

We know the following relations between a logic and a programming language. We use Typed Lambda Calculus that used for the sauce language of a compiler and use Natural Deduction that is a kind of logic. Typed Lambda Calculus in a programming language and Natural Neduction in a logic can be regarded as the same thing. Typed Lambda Calculus and Natural Deduction have same character between a type and a proposition and same character between a program and a proof. If P is the proof of Proposition A in Natural Deduction, P is the program that calculates type in Typed Lambda Calculus. It is said that Curry-Howard Isomorphism [Curry:80][Howard:80][Gallier:93].

Analysis which caught program conversion with proof conversion has been performed by using Curry-Howard Isomorphism in recent years. A kind of genzen style sequent calculus is called GK [Kleene : 52]. Subsystem of GK is called GKA [ohori:99]. The program conversion from Typed Lambda Calculus to Typed A-normal form is expressed by composition of the proof conversion NG that is from Natural Deduction to GK , and the proof conversion S that is from GK to GKA. Typed A-normal form is in agreement with GKA which is A-normal regularity of GK.

Finally, a compiler performs program conversion to Stack-based Logical Abstract Machine(SLAM) which has the same character as Sequential Sequent Calculus (SSC)[ohori:99]. The program conversion to SLAM from Typed A-normal from give same conversion as the proof conversion to SSC from GKA. However, that proof conversion is not defined.

We express to the following figure about the correspondence relation between a programming language and a logic.

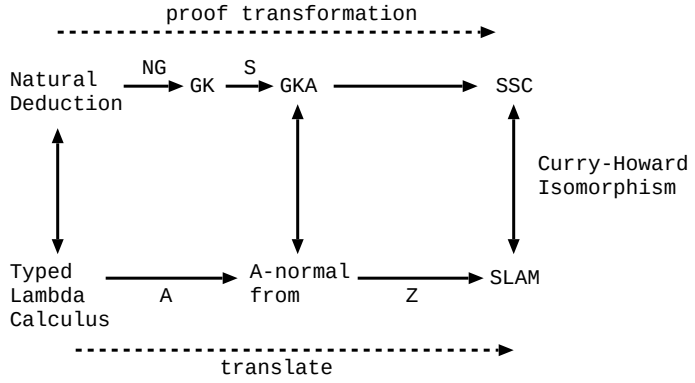

Figure 1: Curry-Howard Isomorphism between a programming language and a logic

3 Target

The target of this report is that we make a design and an inprementation of type directed compiler based on the logical interpretation of the intermidiate language "A-

normal form". Actually, we do three followings.

We examines the application possibility to the compiler of research of the existing type theory first. We prove a theorem that conserve type from A-normal form to SLAM second. Finally, we make a compiler based on that theory and that theorem.