Annals of Pure and Applied Logic 165 (2014) 340–356

Contents lists available at SciVerse ScienceDirect

Annals of Pure and Applied Logic

www.elsevier.com/locate/apal

Proof internalization in generalized Frege systems for classical logic

Yury Savateev [1]

Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland

ARTICLE INFO

Article history:
Available online 13 August 2013

MSC:
03B45
03F07
03F45
03F05

Keywords:
Justification logic
Modal logic
Classical logic

ABSTRACT

We present a general method for inserting proofs in Frege systems for classical logic that produces systems that can internalize their own proofs.

1. Introduction

The Logic of Proofs was introduced by Artemov in [1] (a survey article can be found at [2]). It is based on a notion of proof polynomials, a sort of template for proofs which allows to talk about proofs and propositions in the same language. In a sense it is a refinement of modal logic — a formula $\Box A$, which is typically interpreted as *A is provable* or *A is known* is replaced by an expression of the form $t : A$, that can be read as *t is a formal proof of A* or *A is known because of the evidence provided by t*. This approach helps to avoid the problem of logical omniscience — when an agent knows all logical consequences of his assumptions (see [3]) and gives good ways to explore the semantics of provability. The central problem in justification logics is the realization theorem — how to replace all the boxes in a modal formula with corresponding justifications. A uniform way of doing it for different justification logics was presented in [6].

The structure of proof polynomials in justification logic is determined by the Hilbert-style system for classical logic. In this paper we offer a generalized approach to achieving the goals of justification logic that works on different styles of proof systems. We present a purely syntactical way of putting proofs themselves (not templates built from external constants) inside a general class of what we call generalized

E-mail address: yury.savateev@gmail.com.

[1] The research was partly supported by Hasler Foundation.

0168-0072/$ – see front matter
http://dx.doi.org/10.1016/j.apal.2013.07.017

Frege systems. This provides a uniform way to turn any system in this class into a system closed under the notion of proof internalization — any proof can be put inside a derivable object of the system and reasoned about using already existing rules. Our approach is different from the usual justification logics — instead of realizing modal logics we put proofs of a system for classical logic inside it. The main idea is that doing so we naturally arrive at a system that corresponds to a modal logic without adding anything else. The generalized Frege systems are a pretty wide class, and we only require them to hold some basic properties with respect to classical logic in order for the method to work.

While working with systems defined in such general way it is hard to provide a uniform realization procedure. Nevertheless, we were able to devise a uniform method of realization for some of the most commonly used systems for classical logic, like Hilbert-style systems and sequent systems with certain properties. Previous attempts of building justification system for sequent-based proofs (see [4]) were not entirely successful — the proof terms did not have enough information to recover the proof and it used external labels in proof terms.

2. Generalized Frege systems

In this section we will define what we will mean by generalized Frege system $\mathcal{F}$.

2.1. The language of the system $\mathcal{F}$

First we assume that we have a countable set of *propositional variables* $Prop = \{x_0, x_1, \ldots\}$. Then we have connectives to make formulas out of them and meta-connectives to build the derivable objects or DOs of our system (sequents).

We assume that the structure of formulas and DOs is given by a context-free grammar with the following restrictions:

1. There is a non-terminal symbol P that can be replaced by any propositional variable, and those rules are the only ones that mention propositional variables. This allows us to prohibit references to specific propositional variables in the axioms and rule descriptions.
2. There is a non-terminal symbol F rules for which can contain only terminal symbols for formula connectives and itself plus the additional rule $F \to P$.
3. There is a non-terminal symbol S rules for which can contain only terminal symbols for meta-connectives and non-terminals.

The set of all formulas (strings derivable from F) of our language we denote by $\mathrm{Fm}_{\mathcal{F}}$. We denote the set of all DOs (strings derivable from S) of our system by $\mathrm{DO}_{\mathcal{F}}$.

2.2. Axioms and rules of the system $\mathcal{F}$

Axiom schemes are strings in the language of our grammar that do not have propositional variables in them (they can have the symbol P) and that can have labels on non-terminal symbols.

Axiom instances are a subset of $\mathrm{DO}_{\mathcal{F}}$. Each axiom instance can be derived from an axiom scheme with the restriction that substrings derived from the same non-terminals with the same label are the same.

A *rule scheme* is a tuple of several such strings with shared labels. A rule can have parameters derivable from specified strings of our grammar and must have one or more premises and one result. An example of a rule with parameters is weakening in sequent systems — the formula being added to the sequent is a parameter. A *rule instance* is defined as a tuple of the elements derivable from corresponding strings with the same restrictions as the axioms. Premises and the result in the rule instance must be members of $\mathrm{DO}_{\mathcal{F}}$.