Completeness of Resolution for Definite Answers with Case Analysis

Tanel Tammet

Department of Computing Science
Chalmers University of Technology and Göteborg University
S-41296 Göteborg, Sweden
e-mail: tammet@cs.chalmers.se

Abstract. We investigate the problem of finding a computable witness for the existential quantifier in a formula of the classical first-order predicate logic. The A-resolution calculus based on the program derivation algorithm A of C-L. Chang, R. C-T. Lee and R. Waldinger (a subsystem of the Manna-Waldinger calculus) is used for finding a definite substitution $t$ for an existentially bound variable $y$ in some formula $F$, such that $F\{t/y\}$ is provable. The term $t$ is built of the function and predicate symbols in $F$, plus Boolean functions and a case splitting function $if$, defined in the standard way: $if(True, x, y) = x$ and $if(False, x, y) = y$.

We prove that the A-resolution calculus is complete, i.e. if such a definite substitution exists, then the A-calculus derives a clause giving such a substitution. The result is strengthened by allowing the usage of liftable criterias $R$ of a certain type, prohibiting the derivation of the substitution terms $t$ for which $R(t)$ fails. This enables us to specify, for example, that the substitution $t$ must be in some special signature or must be type-correct, without losing completeness.

1 Introduction

The motivation for this work is to devise efficient automated theorem proving strategies for the first-order theorem proving tasks arising in the formal derivation of programs from specifications. The specific aim of the paper is to present completeness results for certain simple relatively well-known program synthesis algorithms.

One of the standard approaches to automated program construction is using intuitionistic logic with a suitable realizability interpretation to derive programs from proofs (see [5], [11], [8]). The programs derived in this way always enjoy an intuitionistic correctness proof.

Another approach (see [2], [6], [7], [1]) is to use classical logic instead, with the additional restrictions guaranteeing that the proof contains a single definite substitution $t$ into a certain existentially bound variable, and this $t$ is furthermore in a signature where all the function and predicate symbols are assumed to represent computable functions. The derived programs thus always have a classical correctness proof, although they may lack an intuitionistic one.
The following summarizes our motivation for using the second approach (classical logic) for program construction.

The known realizability interpretations for intuitionistic logic often give programs which contain computationally irrelevant parts. For example, the realization of the formula $\forall x \exists y (x = y \& y = x)$ is a term $\lambda x. (x, (id, id))$. The A-resolution gives a term $\lambda x. x$ as a program to compute $y$.

Some formulas which admit a proof by A-resolution (and hence give a program) are not provable by intuitionistic logic. For example, A-resolution gives a program $\lambda x. x$ for computing $y$ for the intuitionistically unprovable formula $\forall x \exists y ((A \lor \neg A) \& y = x)$.

The standard resolution method with Skolemization and/or conversion to a conjunctive normal form (CNF) cannot be used for intuitionistic logic, although there exist special resolution methods without Skolemization and CNF ([9], [10]) and a tableaux method with partial dynamic Skolemization ([15]) for intuitionistic logic. Also, there exists a sizeable amount of theory for the resolution method, which can be used for program derivation by A-resolution.

1.1 Basic Definitions

We consider closed formulas in the first-order predicate logic language with function symbols. When we speak about derivability (provability), we mean derivability in the classical first-order logic. We will restrict us to formulas which contain at least one positive occurrence of the existential quantifier or at least one negative occurrence of the universal quantifier. The polarity of subformula occurrence is defined in the standard way: subformulas under an odd number of negations and left argument positions of implications are negative, all the others are positive. We will refer to both the positive occurrences of existential quantifiers and negative occurrences of universal quantifiers as essentially existential quantifiers and all the others as essentially universal quantifiers.

As our goal is to derive programs by finding a certain definite substitution $t$ into one of the variables bound by the essentially existential quantifier, we assume that our formulas have an associated marker for this specific variable, which will be called the main variable of the formula. We require that the quantifier occurrence $Q$ binding the main variable must be outside the scope of other essentially existential quantifiers. The set of variables bound by the set $S$ of all occurrences of the essentially universal quantifiers such that $Q$ is in the scope of all the elements of $S$ is called the set of parametric variables of the formula.

Given a formula $F$ and its main variable $y$, we are looking for a proof of $F$ such that this proof gives a term $t$ for computing a value $r$ for $y$ for any set of values $t_1, t_2, \ldots, t_n$ assigned to parametric variables $x_1, x_2, \ldots, x_n$ so that a substitution instance $F\{t_1/x_1, t_2/x_2, \ldots t_n/x_n, r/y\}$ of the formula $F$ would be provable in the first-order classical logic. Here and elsewhere $\{t_1/x_1, \ldots, t_n/x_n\}$ represents the substitution of each $t_i$ ($1 \leq i \leq n$) for the variable $x_i$, respectively.

The terms $t$ (representing computable functions) we are looking for are assumed to contain only the function and predicate symbols and parametric variables of $F$, plus Boolean functions and a case-analysis function "if" defined in