Extraction of Proofs from the Clausal Normal Form Transformation

Hans de Nivelle
Max Planck Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
nivelle@mpi-sb.mpg.de

Abstract. This paper discusses the problem of how to transform a first-order formula into clausal normal form, and to simultaneously construct a proof that the clausal normal form is correct. This is relevant for applications of automated theorem proving where people want to be able to use theorem prover without having to trust it.

1 Introduction

Modern theorem provers are complicated pieces of software containing up to 100,000 lines of code. In order to make the prover sufficiently efficient, complicated datastructures are implemented for efficient maintenance of large sets of formulas ([16]) In addition, they are written in programming languages that do not directly support logical formulas, like C or C++. Because of this, theorem provers are subject to errors.

One of the main applications of automated reasoning is in verification, both of software and of hardware. Because of this, users must be able to trust proofs from theorem provers completely. There are two approaches to obtain this goal: The first is to formally verify the theorem prover (the internalization approach), the second is to make sure that the proofs of the theorem prover can be formally verified. We call this the external approach.

The first approach has been applied on simple versions of the CNF-transformation with success. In [10], a CNF-transformer has been implemented and verified in ACL2. In [5], a similar verification has been done in COQ.

The advantage of this approach is that once the check of the CNF-transformer is complete, there is no additional cost in using the CNF-transformer. It seems however difficult to implement and verify more sophisticated CNF-transformations, as those in [12], [1], or [8]. As a consequence, users have to accept that certain decision procedures are lost, or that less proofs will be found.

A principal problem seems to be the fact that in general, program verification can be done on only on small (inductive) types. For example in [5], it was necessary to inductively define a type prop mimicking the behaviour of Prop in COQ. In [10], it was necessary to limit the correctness proof to finite models. Because of this limitation, the internalization approach seems to be restricted to problems that are strictly first-order.
Another disadvantage of the internalization approach is the fact that proofs cannot be communicated. Suppose some party proved some theorem and wants to convince another party, who is skeptical. The other party is probably not willing to recheck correctness of the theorem prover and rerun it, because this might be very costly. It is much more likely that the other party is willing to recheck a proof.

In this paper, we explore the external approach. The main disadvantage of the external approach is the additional cost of proof checking. If one does the proof generation naively, the resulting proofs can have unacceptable size [6]. We present methods that bring down this cost considerably.

In this paper, we discuss the three main technical problems that appear when one wants to generate explicit type theory proofs from the CNF-transformation. The problems are the following: (1) Some of the transformations in the CNF-transformation are not equivalence preserving, but only satisfiability preserving. Because of this, it is in general not possible to prove $F \iff \text{CNF}(F)$. The problematic conversions are Skolemization, and subformula replacement. In order to simplify the handling of such transformations, we will define an intermediate proof representation language that has instructions that allow signature extension, and that make it possible to specify the condition that the new symbol must satisfy. When it is completed, the proof script can be transformed into a proof term.

(2) The second problem is that naive proof construction results in proofs of unacceptable size. This problem is caused by the fact that one has to build up the context of a replacement, which constructs proofs of quadratic size. Since for most transformations (for example the Negation Normal Form transformation), the total number of replacements is likely to be at least linear in the size of the formula, the resulting proof can easily have a size cubic in the size of the formula. Such a complexity would make the external approach impossible, because it is not uncommon for a formula to have 1000 or more symbols. We discuss this problem in Section 3. For many transformations, the complexity can be brought down to a linear complexity.

(3) The last technical problem that we discuss is caused by improved Skolemization methods, see [11], [13]. Soundness of Skolemization can be proven through choice axioms. There are many types of Skolemization around, and some of them are parametrized. We do not want have a choice axiom for each type of Skolemization, for each possible value of the parameter. That would result in far too many choice axioms. In Section 4 we show that all improved Skolemization methods (that the author knows of) can be reduced to standard Skolemization.

In the sequel, we will assume familiarity with type theory. (See [15], [3]) We make use only of standard polymorphic type theory. In particular, we don’t make use of inductive types.