LOGICALC: an environment for interactive proof development*

D. Duchier and D. McDermott
Department of Computer Science
Yale University

Abstract

LOGICALC is a system for interactive proof development. It lives in a graph editor environment based on DUCK's "walk mode" [10]. The user grows a tree of goals and plans by invoking plan generators. Goals are sequents of the form $A \Rightarrow p$, where $A$ is an assumption set. Each plan has a validation that indicates how to conclude a proof of its parent goal from the proofs of its steps. When every step of a plan has been solved, their proofs are combined according to the validation and the resulting proof is handed to the parent goal. The logic is based on skolemization and unification. The system addresses subsumption within the goal graph, and automatic generalization of proofs.

Introduction

We present a system for interactive proof development. The project started out of the theorem proving demands of our spatial reasoning research. It was further justified by our interest in circumscription and the sad fact that most arguments of circumscription theory are quite tedious to work one's way through.

The requirements for the system were that it should find an acceptable compromise between being formal and being helpful, that it should lend itself, not only to proof checking, but to proof discovery as well, and, eventually, to alternative forms of automation, and that it should produce, not simply yes/no answers, but complete and self contained proof objects.

Our system is called LOGICALC, and its essential features are as follows:

*this research was supported by grant DAAA-15-87-K-0001 from BRL/DARPA
• It offers a goal/plan based approach to incremental proof refinement. Each goal may have more than one plan attached to it. Thus, the user can keep track of several alternative proof attempts at the same time.

• Goals are sequents of the restricted form $A \Rightarrow p$; where $p$ is the conclusion to be derived, and $A$ is the assumption set (represented as a data pool [9,10], for efficient access).

• Assumptions and goals are skolemized, so that quantifier-manipulation rules are unnecessary and the logic can be based on unification. A conclusion containing skolem terms and such that its proof does not depend on assumptions about them is appropriately generalized.

• The system avoids duplicating goals and maintains links between stronger and weaker goals. Thus, answers can be shared and appropriately propagated. As a consequence, what we think of as the proof tree is really a graph.

• The logic is first-order, but lambda-expressions can appear as terms. Appropriate support is provided in terms of inference rules, and the unifier can cope with them to a limited extent. [not discussed in this paper]

• The user can move from node to node in the graph; where a node may be a goal, a plan, an answer, a proof, or an assumption. Depending on the type of a node, the information displayed and the available commands will vary.

1 Overview

LOGICALC generalizes backward chaining. The generalization is just as powerful as resolution, and, indeed, resembles it in some ways, such as the use of skolemization for handling quantifiers. The details are quite different however, and the system is mostly based on natural deduction, although some of its techniques are related to McAllester's [8].

In a backward chaining approach, a prover works on goals, finding implications whose consequents unify with the current goal, and producing subgoals from their antecedents. This process bottoms out when a goal unifies with a fact. We generalize this paradigm in the following ways:

1. We provide several other methods of generating subgoals, such as reasoning by enumeration of cases, equality substitution, and proof by contradiction. A general method of generating subgoals is called a plan generator (cf. tactics in LCF [12]).

2. Goals and conclusions are represented by sequents [5] of the form $A \Rightarrow p$, where $A$ is a set of assumptions. We say that the sequent $A \Rightarrow p$ is true iff $p$ logically follows from $A$. Typically, $A$ is a large data base, including an entire