Domain Theory in HOL

Sten Agerholm

Computer Science Department, Aarhus University, DK-8000 Aarhus C, Denmark

Abstract. In this paper we present a formalization of domain theory in HOL. The notions of complete partial order, continuous function and inclusive predicate are introduced as semantic constants in HOL and fixed point induction is a derived theorem, just as we can derive other techniques for recursion. We provide tools which prove certain terms are cpos, continuous functions or inclusive predicates, automatically.

1 Introduction

Domain theory is the study of complete partial orders (cpos) and continuous functions — the mathematical foundation of denotational semantics. It provides the concepts and techniques useful to reason about formal semantics, for instance the semantics of nontermination and recursive definitions (least fixed points). Its applicability is documented in a good many text books, e.g. [Wi93, Sc86, St77].

Scott’s Logic of Computable Functions was given a semantics using domain theory such that types in the logic were cpos and functions between types were continuous functions between cpos. Scott’s logic has been implemented in the theorem prover LCF [Pa87, GMW79]. LCF provides a first order logic and fixed point induction to prove recursive (continuous) functions satisfy inclusive predicates (called chain-complete or admissible predicates in [Pa87, GMW79]). Inclusiveness is a semantic notion and one of the disadvantages of LCF is that the test for inclusiveness in LCF is a syntactic one which is not complete. Thus, there are inclusive predicates which cannot be used for induction in LCF. Another problem in LCF is that fixed point induction is the only way to reason about recursive definitions (structural induction is derived from fixed point induction) but using other techniques (see e.g. [Wi93]) or reasoning directly about fixed points allow us to prove more theorems than with just fixed point induction. Finally, we can mention that LCF has very few built-in theories, tools and libraries which makes it tedious to use the system.

In this paper we present a formalization of domain theory in HOL that does not suffer from such problems. The notions of complete partial order, continuous function and inclusive predicate are introduced as semantic constants in HOL and one can therefore prove terms satisfy the semantic conditions in an ad hoc way in the HOL system. But we can also build proof tools, i.e. ML programs, which prove certain terms satisfy the cpo, continuity and inclusiveness conditions, automatically, based on the syntax of these terms. For the syntax a number of constructions on cpos and continuous functions have been introduced. Another point is that in our formalization fixed point induction is a derived theorem (requiring functions are continuous), just as we can derive other techniques.
for recursion and reason about fixed points directly. Finally, we are able to exploit the many built-in theories and libraries of HOL in applications of the formalization because the set part of cpos are subsets of HOL types. For instance, the set can be the booleans or the natural numbers.

A limitation of our approach is that the inverse limit construction cannot be directly formalized, and therefore recursive domain equations cannot be solved at the moment. If we switch to other models as information systems [Wi93] or Pω [Sc76, Ba84] we would be able to construct solutions to equations but we might not be able to use HOL types as directly as we do now (see below).

Other formalizations of domain theory exists. Petersen [Pe93] has formalized the Pω model (see the HOL contrib library) such that all recursive domain equations can be solved. However, domains live in Pω only and it is still not clear how to lift HOL types and functions to Pω and back. Therefore very few of HOL’s facilities can be exploited directly. Camilleri mechanized a theory of cpos and fixed points (see the HOL contrib library) which he used to define recursive operators in CSP trace theory [Ca90]. However, he did not consider constructions on cpos and continuous functions but proved continuity in an ad hoc way in the HOL system. A major problem with his approach is that it does not allow the continuous function space construction which is fundamental to our work. The problem is that the set of continuous functions between two cpos is a dependent subset of all HOL functions. We have solved this problem using dependent subtypes, also called term parameterized types. Though these are not provided by the HOL logic itself they can be simulated by predicates denoting subsets of types (the same approach is used in [JM93]).

The contribution of this work is a formalization of domain theoretical concepts in HOL and a number of proof tools to make the formalization more useful. The formalization is based on the book by Winskel [Wi93]. An interface and various definition tools are implemented on top of the basic tools for cpos, continuity and inclusiveness. In this way we obtain an integrated system where cpo and continuity facts are proved behind the scenes automatically (though the system is still a prototype and rather simple).

The goal is eventually to provide a system for reasoning about functional programs in HOL based on domain theory and its techniques. We hope that such a system can benefit from being embedded in as powerful and flexible a system as the HOL system (compared to a direct implementation of domain theory as in LCF), for instance by ensuring the HOL logic and tools are inherited and by allowing domain and set theoretical reasoning to be mixed.

The formalization is described in section 2, basic proof tools in section 3 and the interface in section 4. In section 5 a few examples and derived definition tools are provided to illustrate the use of the system.

2 Domain Theory

We give a brief overview of our formalization of domain theory in HOL.