Formal Proof for Termination of Programs Using Ordinal Numbers

Andrés Felipe Tamayo-Arango and Andrés Sicard-Ramírez[†a]
aftamayo@eafit.edu.co, asr@eafit.edu.co

Mathematical Engineering, Universidad EAFIT
Mathematical Science Department, School of Sciences, Universidad EAFIT

[a]Tutor

Table of contents

Problem definition

Problem and definitions

- Formal proofs
- Formal verification
- Correctness
- Termination proofs
- Curry-Howard correspondence
- Ordinal numbers

State of Art

Fundamental ideas

- Halting problem (Davis, 1958)
- Termination, (Turing, 1949) ”How can one check a routine in the sense of making sure that it is right?”
- Basis for formal definitions of the meaning of programs (Floyd, 1967)
- Mapping the iterations of an algorithm to a corresponding set of ordinal numbers (Nachum Dershowitz, 1979)
- Processes that transform trees or terms can often be proved terminating by viewing the tree or the tree representation of the tree as an ordinal. (Dershowitz, 1993)

Definition
A tree is an ordered set $(T, \leq)$ which has a least element and is such that, for every $x \in T$, the set $\{y \in T \quad | \quad y < x\}$ is well-ordered by $\leq$. (Hrbacek and Jech, 1999)

Theorem
Every well-ordered set is isomorphic to a unique ordinal number. (Hrbacek and Jech, 1999)

Progress and challenges

A program for simplification of a formula

Consider the following system:

$$\neg\neg x \Rightarrow x$$
$$\neg(x \vee y) \Rightarrow \neg x \wedge \neg y$$
$$\neg(x \wedge y) \Rightarrow \neg x \vee \neg y$$
$$x \wedge (y \vee z) \Rightarrow (x \wedge y) \vee (x \wedge z)$$
$$(y \vee z) \wedge x \Rightarrow (y \wedge x) \vee (z \wedge x)$$

If we develop an algorithm for this system, how can we prove that it terminates?

The procedure of applying the previous rules to a logical formula produces a formula in **disjunctive normal form**.

Main challenge:
To find a function that maps the recursive calls of the algorithm that are decreasing to a set of ordinal numbers or to find the tree representation of the term as an ordinal.

Rewriting systems

- A **rewrite (term-rewriting)** system R over a set of terms T is a (finite) set of rewrite rules, each of the form l -> r, where l and r are terms containing variables ranging over T, and such that r only contains variables also in l.
- **Confluence** is a property of rewriting systems, describing which terms in such a system can be rewritten in more than one way.

Termination of rewriting systems

- Termination is a property of rewrite systems. No infinite derivations are possible.
- The difficulty that may be encountered when attempting to determine if a rewrite systems terminates is related to the non-deterministic choice of the rules of rewriting.

$$\neg(x \vee (y \wedge z)) \Rightarrow \ldots \Rightarrow \neg x \wedge \neg(y \wedge z)$$
$$\neg(x \vee (y \wedge z)) \Rightarrow \ldots \Rightarrow \neg((x \vee y) \wedge (x \vee z))$$

Question: How many recursive calls are necessary in order that the given formula is in disjunctive normal form?

References

Martin Davis. *Computability and Unsolvability*. McGraw-Hill, 1958.

Nachum Dershowitz. Trees, ordinals and termination. *Springer, Berlin, Heidelberg*, 1993.

Robert W. Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science*, 1967.

Karel Hrbacek and Thomas Jech. *Introduction to Set Theory. Third Edition, Revised and Expanded*. Marcel Dekker, 1999.

Zohar Manna Nachum Dershowitz. Proving termination with multiset orderings. *Communications of the ACM*, 1979.

Alan M. Turing. Checking a large routine. *Report of a Conference on High Speed Automatic Calculating*, 1949.