

Summary

Termination and Productivity

In this thesis we are concerned with two key aspects of program correctness: termination and productivity.

A program is called terminating if it finishes the computation after a finite number of steps. In other words, the computation halts within finite time. Thus, we usually expect correct programs to be terminating. Even non-terminating control programs, such as Windows and Word, consist to a large part of terminating function calls, and termination of these functions is essential since they must return to the main control loop; otherwise the program hangs.

On the contrary, productivity is concerned with non-terminating processes. Productivity captures the intuitive notion of unlimited progress, of 'working' programs producing defined values indefinitely. For example, control programs (mentioned above) have to be productive; they should keep producing output, that is, keep responding to user inputs. Moreover, productivity is important for terminating programs that involve specifications of infinite structures, such as infinite lists or trees. For correctness of these programs it has to be guaranteed that every finite part of the infinite structure can be evaluated, that is, the specification of the structure must be productive.

With the increasing complexity of computer systems, program verification has become central in computer science. To keep the increasing complexity of software and hardware systems manageable, we need automated tools that support the verification of (certain aspects of) program correctness. In this thesis we develop methods for the automatic analysis of termination and productivity.