Abstract

The main objective of this thesis is to identify the benefits of adopting a higher-order functional language as the core of a mobile code language. We demonstrate that static analysis techniques traditionally associated with functional languages, in particular those which involve type systems, can be exploited further to reason about the dynamic behaviour of mobile code.

Chapter 1 introduces the characteristics of mobile computation and functional computation. It argues that integrating these two paradigms can offer solutions to the problems which have proved to be challenging in the context of mobile computation. The useful role which can be played by type and effect systems is discussed.

Chapter 2 gives an overview of the process calculi which provide formal models of distributed and mobile computation. This is followed by a closer look at the programming languages Concurrent ML, Facile and PLAN (Programming Language for Active Networks). These languages point to a consistent effort to benefit from the fundamental ideas behind ML in designing and implementing languages for concurrent and distributed computation.

Chapter 3 focuses on a language similar to Facile where values of all types, including functions and communication channels, can be transmitted between remote sites. The problem investigated in this chapter is the static estimation of functions and channels which may become mobile at run-time. A static analysis such as the one considered in this chapter would be a useful asset for compilers in dealing with the heterogeneity of the network nodes, detecting the locality of certain values and providing static profiling tools.

Chapter 4 focuses on the language PLAN. The form of support for code mobility in PLAN is different from that of Facile. It is based on a remote evaluation facility for functions. The design of PLAN has been influenced by the need to meet the strong safety and security requirements of active networks; especially by the need to protect against denial of service. The subject of this chapter is distributed call-tracking by means of a type and effect system in the framework of a PLAN-like language. It is argued that for an applicative language distributed call-tracking can provide the basis for static estimation of resource consumption.

Chapter 5 shifts the focus back to a language which resembles Facile. Some distributed systems are characterized by their heterogeneity in terms of the nature of the computing devices, security requirements of the information flowing in the system and the trust level of the users. Programmers who provide code for such systems would find it useful to have a language mechanism which enables them to confine the flow of certain values to a particular part of the sytem – a mobility region. This chapter discusses how a static type system can be used to enforce confinement in a specified mobility region.

Chapter 6 revisits the language of Chapter 2 and introduces a variant of it where the values of the language are classified with respect to their confidentiality level. As in Chapter 5, it is assumed that users which interact with the sytem may not be equally trustworthy. The sources of undesirable information flows are identified and a secure information flow property based on noninterference is introduced. Programs which are accepted by the proposed type and effect system for the language are shown to enjoy this property.

Chapter 7 includes a summary of the thesis which clarifies contributions made to the research areas of functional and mobile code languages, annotated type and effect systems and the language-based approach to security.