Functional Mobile Computation

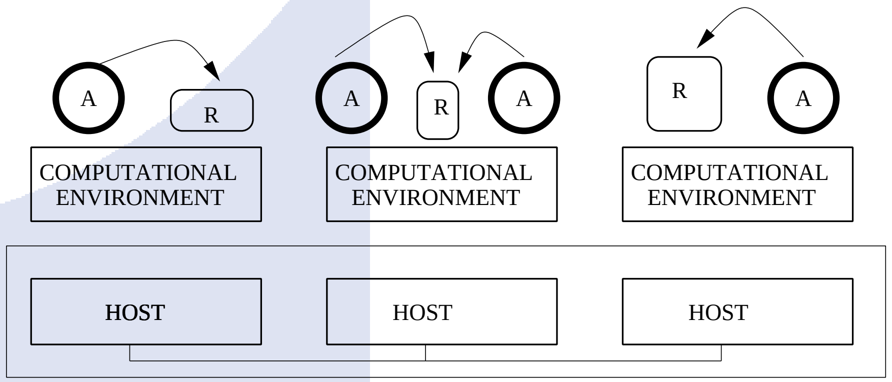
- *towards provably secure mobile programs* -

Dilsun Kırlı

<u>Abstract</u>: The mobile computation paradigm has urged researchers to revisit the traditional assumptions about distributed computation and to establish new principles for designing languages for mobile computation. We explore whether the benefits offered by sequential functional languages, particularly those related to security, can be carried over to the mobile computation context.

Mobile Computation

The advanced state of computer networking and telecommunications technology makes it possible to view computers as parts of a global computation platform, sharing their resources in terms of hardware, software and data. The possibility of exploiting the resources on a global scale has given rise to a new paradigm -the mobile computation paradigm- for computation in large scale distributed networks. Programming languages which support mobility of code or executing programs with their computation and communications context are becoming widely used [2]. Programmers are provided with facilities to create and deploy agents, which can move across the network using the resources of the localities they visit.



Formal Foundations and Security

The experience so far has shown that the mobile computation paradigm causes the interplay between language features to be very complex. A little gap in the understanding of a language feature may lead to undesirable consequences in the safety and security of a system. This fact suggests that it is perhaps simplicity and well-definedness of the computational model that should be sought by language designers. In the programming language community, functional languages are known for these characteristics. Moreover, in the context of sequential computation they have been able to offer significant security guarantees. This is due to the availability of sophisticated tools for semantic analysis of programs which exploit their close links with formal models of computation.

Mobile Functions

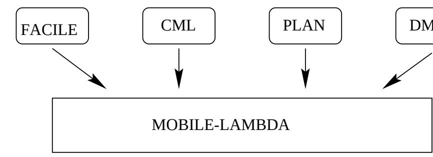
The higher-order features of functional languages allow one to draw an analogy between agents and functions. By this we mean that functions are first-class values; they can flow as arguments to other functions or they can be returned as results. Therefore, in a setting where their mobility between the nodes of a network is facilitated functions become a natural candidate for representing agents.

Static Analysis of Mobile Functions

It is beneficial to predict a mobile agent's behaviour from its code before it is actually executed. Static program analysis offers tools for making such predictions by exploiting theoretically well-founded methods. A type system is a typical static analysis tool which checks whether a program satisfies a predefined property expressed in terms of a particular logic. Strongly-typed functional languages such as Standard ML ensure that programs cannot corrupt the runtime system so that further execution of the program is not faithful to the language semantics. Extended with effects and other annotations type systems can be exploited to reason about computational aspects such as the flow of information through computation and resource usage of programs [5, 4]. These aspects of computation are particularly interesting from the security perspective in the context of mobile computation.

Mobile-λ and Type-based Analysis

We have examined a number of languages which integrate functional computation with some form of concurrency, distribution or support for mobile agents, ranging from general purpose languages such as Concurrent ML and Facile to domain specific languages such as the Packet Language for Active Networks (PLAN) [3, 1]. This led to the definition of Mobile-λ, a skeletal high-level mobile computation language, which can express the core features of the languages examined.



By using Mobile-λ as a meta-language, we have investigated the use of type systems for detecting potential mobility of functions, enforcing secure information flow, and performing distributed control flow analysis. The results show that, thanks to the possibility of developing well-founded tools for static analysis, functional mobile computation can prove useful in combining expressive power and security.

*

References

[1] A. Giacalone, P. Mishra, and S. Prasad. Facile: A symmetric integration of concurrent and functional programming. *International Journal of Parallel Programming*, 18(2):121–160, April 1989.

[2] J. Gosling, B.Joy, and G.Steele. *The Java Language Specification*. Addison-Wesley, 1996.

[3] F. Nielson, editor. *ML with Concurrency: Design, Analysis, Implementation, and Application*. Springer-Verlag New York, Inc., 1997.

[4] G. Smith and D. Volpano. A type-based approach to program security. In *Proceedings of TAPSOFT '97*, pages 607–621, Lille, France.

[5] J.P. Talpin and P. Jouvelot. The type and effect discipline. *Information and Computation*, 111:245–296, 1994.

Division of informatics