Interference and dependence are closely related concepts, the first being the observable phenomenon connected to the second. Interference essentially means that behavior of some parts of a dynamic system may influence the behavior of other parts of the same system. Dependence instead specifies the relation between the semantics of sub-components of a dynamic system. Discovering, measuring and controlling interference is essential in many aspects of modern computer science, in particular in security, program analysis and verification, debugging, systems specification, model checking, program manipulation, program slicing, reverse engineering, data mining, distributed databases, and systems biology. In all these fields, dependency and interference play a key role in designing suitable abstractions or in partitioning complex systems into simpler ones. Reasoning about dependency and interference requires theories, models, and semantics, as well as algorithms and tools for their analysis. The series of Programming Language Interference and Dependence (PLID) workshops has been devoted, since 2004, to promote and spreading cutting-edge research in this field, with a particular emphasis on unpublished results with great impact on a theoretical basis. PLID2007 was particularly successful, and it constituted the ideal forum where publishing a call for papers for a special issue on programming language interference and dependence, not necessarily restricted to PLID2007 contributions. We selected six contributions by leading researchers in the field, from many expressions of interest, some of them presented at the PLID2007 workshop, which took place at the The Technical University of Denmark, 21 August, 2007. The selected papers focus on foundational aspects of dependency and interference with applications in language based security, data-base management systems, and program slicing.

The paper of Ana Almeida Matos and Jan Cederquist studies information flows that occur in distributed programs in presence of code mobility. This is a key problem in language based security. The authors introduce interesting new forms of security leaks that are introduced by code mobility which can be checked by means of a type and effect system for enforcing security, which is validated into an expressive calculus of mobile code.

The contribution of Geoffrey Smith and Rafael Alpízar attacks the problem termination in security checking in presence of random assignments. This is a difficult and important problem in language based security which requires advanced mathematical models for being solved. The au-
David Clark, Roberto Giacobazzi and Chunyan Mu

Authors introduce an innovative notion of fast probabilistic simulation on Markov chains and show that this implies a key reachability property that is essential in checking information flows, proving that well-typed probabilistic programs are guaranteed to satisfy an approximate probabilistic noninterference property, provided that their probability of nontermination is relatively small.

The paper of Gilles Barthe, Pedro R. D’Argenio and Tamara Rezk investigates logical formulations of secure information flow based on the idea of self-composition. This allows a practical and innovative reduction of the problem of secure information flow of a program $P$ to a safety property for a modified program, derived from $P$, by composing $P$ with a renaming of itself. This technique makes it easy the verification of information flow policies by model checking and standard program verification methods, e.g. in a à la Hoare logic.

The contribution of Isabella Mastroeni and Anindya Banerjee explores the dimensions of a declassification-based noninterference policy, providing a new perspective in this field. Two of the dimensions consist in specifying the power of the attacker, that is, what public information an attacker can observe of a program, and what secret information of a program needs to be protected. The third dimension, which regulates the above two, specifies the choice of the program semantics. This choice can be formally specified as an abstract interpretation of a more concrete semantics.

The paper of James Cheney, Amal Ahmed, and Umut A. Acar studies provenance, i.e., the information recording the source or history of some computed data. The tracking of this information is particularly relevant in database management systems and security. The authors introduce an innovative formal semantics for modeling data provenance. Their idea is to use the notion of dependence, and corresponding models for dependence analysis and program slicing, for specifying provenance, providing one of the very first formal account of provenance in programming languages.

In the field of program slicing, Sebastian Danicic, Robert M. Hierons and Michael R. Laurence explored the computational complexity of dynamic program slicing for schemas. The authors formalize the Korel and Laski’s definition of a dynamic slice as applied to linear schemas, and formulate an innovative and less restrictive definition in which the path through the original program need not be preserved by the slice. The complexity analysis of the problem of determining whether a given slice from linear schema is a dynamic slice and whether a linear schema has a non-trivial dynamic slice is proved to be NP-hard.