Theoretical Computer Science 192 (1998) 163–165

Theoretical Computer Science

Foreword

Theoretical aspects of coordination languages

Roberto Gorrieri[a,*], Chris Hankin[b]

[a] *Dipartimento di Scienze dell'Informazione, Università di Bologna, Mura Anteo Zamboni 7, I-40127 Bologna, Italy*
[b] *Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK*

About coordination

The emergence of high bandwidth network technology has fuelled the development of distributed computing and concurrent programming. Coordination languages are a new class of programming languages which offer a solution to the problem of managing the interaction among concurrent programs. Gelernter and Carriero coined the term *Coordination* in the following slogan:

Concurrent programming = Computation + Coordination.

They formulated this equation when introducing the coordination language Linda. The intent is that there should be a clear separation between the *components of the computation* and their *interaction* in the overall program or system. On the one hand, this separation facilitates the reuse of code; on the other hand, the same patterns of interaction occur in many different problems – so it might be possible to reuse the coordination component as well!

Coordination languages are not general purpose programming languages; rather, they are often defined as language extensions or scripting languages and they are exclusively concerned with coordination issues. In defining coordination languages there are a number of issues which must be addressed:

1. what is being coordinated?
2. what are the media for coordination?
3. what are the protocols and rules used for coordination?

Coordinated entities: The coordinated entities are usually active – agents or processes. Coordination of agents should not require reprogramming of the agents; the coordination mechanism is a wrapper around the existing, independent agents. The agents may have been programmed in a variety of different programming languages.

* Corresponding author.

Coordination media: In many coordination languages, coordination is accomplished via a shared data space. In such models, communication is *generative*: agents communicate by "generating" data in the shared space, this data is then available to any other agent that has access to the space – this contrasts with the message-passing paradigm of concurrency where communication is usually a private act between the participating agents. In a heterogeneous system, in which the agents are written in different languages, the data must be stored in a common format.

Coordination rules: The Linda proposal identifies a set of coordination primitives which may be used to access a shared data space – the primitives are normally implemented as library routines which are called from some host language such as C or Prolog. In contrast to Linda, many of the recent proposals have been for rule-based languages; one consequence of this shift to a more declarative view of coordination is increased reasoning power. In either case the coordination "rules" provide a level of abstraction which hides much of the complexity of coordination from the programmer.

This topic was the subject of the ESPRIT Long Term Research Project 9102. The project partners organised the first international conference on Coordination languages and models, which was held in Cesena in April 1996 [2]. The second conference will be held in Berlin in September 1997. The collection [1] contains papers which were presented at the various project workshops.

About this issue

Following the Coordination conference, we invited selected participants and the project partners to contribute to this special issue. Authors were asked to concentrate on theoretical issues relating to Coordination Languages. Theoretical Computer Science's sister journal, Science of Computer Programming, is publishing a special issue devoted to practical issues. The submissions were subjected to the usual reviewing process and six papers were selected for inclusion in the issue. We briefly review the papers below.

A Process Algebraic View of Linda Coordination Primitives (Busi, Gorrieri and Zavattaro) studies Linda primitives from a process algebraic point of view. The paper presents a lattice of languages; the languages are based on CCS extended with a tuple space and are differentiated by the tuple space operations that they include. The paper develops an observational semantics, based on an appropriate notion of barbed bisimulation, for each language. The top element of the lattice, LINPA, includes all of the Linda tuple operations (except for eval); the paper provides the first interleaving operational semantics for these operations.

A Lambda Calculus for Dynamic Binding (Dami) studies a formal calculus, λN, which models dynamic binding. Most interesting applications of coordination languages are to open systems. In such systems the correct modelling of dynamic binding is an important issue. The paper presents the basic theory of the calculus, including confluence and the relation to the classical λ-calculus, a simple type system, encoding of record structures and applications in functional programming.

Refining Multiset Transformers (Hankin, Le Métayer and Sands) studies the multiset transformation language Gamma. The paper presents an operational semantics for the language and develops a number of laws based on a notion of refinement derived from the semantics. The theory is used to develop a *pipelining* transformation that has fairly general applicability.

Entailment Based Actions for Coordination (Monteiro and Porto) studies an abstract model of coordination via a shared data space. The paper models the shared data space by a certain type of poset which is called the *situation space*. Updates to the situation space are defined via the notion of entailment; an update is the least extension of the space that preserves important properties (such as coherence, consistency, etc.). Actions generalise updates by introducing non-determinism and conditional operations.

A Process Calculus Based Abstraction for Coordinating Multi-Agent Groups (Mukherji and Kafura) studies a new calculus, Calculus of Coordinating Environments (CCE), for the analysis of coordination as the behavioural union of coordinated and coordinating agents. The paper starts by considering how CCS can be used to express coordination and highlights some inadequacies in CCS. This motivates the introduction of CCE; coordinated agents are still modelled as CCS processes but the coordinating agent is modelled by a different class of agent expressions that may be composed with the coordinated agents using a new composition operator.

Interactive Foundations of Computing (Wegner) presents a philosophical view of the computational power of interaction. The paper argues that interaction adds a new dimension to computability that overcomes the limits of Turing computatibility. The thesis is carefully argued and illustrated by a number of examples.

Four papers are from authors involved in the Coordination project, whereas two papers extend works published in the proceedings of the First Coordination conference. We thank the anonymous reviewers who helped us in the selection: their help was invaluable in putting together this special issue.

References

[1] J.-M. Andreoli, C. Hankin, D. Le Métayer (Eds.), Coordination Programming: Mechanisms, Models, Imperial College Press, 1996.

[2] P. Ciancarini, C. Hankin (Eds.), Coordination Languages and Models, vol. 1061, Lecture Notes in Computer Science, Springer, Berlin, 1996.