Coordination with Attributes

Alan Wood

Department of Computer Science
University of York
wood@cs.york.ac.uk

Abstract. This paper addresses the opportunities for, and effects of, incorporating object attributes in LINDA-like tuple-space systems. The focus is on exploring how the coordination attributes are affected by this enhancement, and investigating in what ways the classical LINDA model needs to be modified. Particular emphasis is placed on the potential practical gains to be expected in both performance and expressibility, with consideration being given to methods for implementing attribute frameworks in open distributed systems.

1 Introduction

The concept of coordination could not be claimed to be ‘new’ — even in the narrow domain of computing, and its sub-domain of concurrency, the need to coordinate things has been around a long time. However, it is only relatively recently that coordination has been identified as a topic for independent study, and which promises to unify several apparently disparate fields in computing.

In their 1992 paper [6], Carriero and Gelernter crystallised this idea by identifying the two functions of a “full” programming language as computation and coordination, and suggesting that these two aspects of a language are orthogonal. Consequently, the study of coordination may be undertaken independently of any host computation technology.

Once the idea of coordination being distinct from computation is accepted, the question arises as to what the coordination structures are. These would be analogous to the control and data structures in the computation half of a language. Some coordination structures have been in common use for many years: semaphores, monitors, signals (in the UNIX sense). Others have been emerging from the increased use of parallel and distributed architectures ... examples might be light-weight threads, message-passing, or barrier-synchronisation.

This paper addresses coordination problems in open distributed systems, which are to be contrasted with closed systems. The distinction lies in the fact that in an open system agents (processes) can ‘join’ or ‘leave’ during the lifetime of the system with no prior knowledge of their (potential) existence necessarily being available to the other agents. From the point of view of a programmer, or more importantly a compiler, full information about the behaviour of the system while his/her/its process is executing cannot be determined. In a closed system, however, such information is in principle available, and can be used by a
compiler, say, to optimise the process ensemble. The open system model is that of the internet; the closed system is that of a subset of processors in a box, with only a single parallel program running.

1.1 Object Attributes

Objects have a number of properties associated with them — indeed, it is the set of properties that define an object. In a computational context, perhaps the most fundamental properties are value and type. However, there are other properties which are normally associated with objects including their scope (or visibility), updatability, or other accessibility features. Although these are some obvious concrete examples, attributes should be seen as a collection of abstract properties associated with objects. This view admits the possibility of having attributes which may be user-defined (and, consequently, would need to be user-maintained), rather than fixed by the system.

1.2 LINDA

Since this paper addresses some general aspects of coordination, while suggesting practical ways of implementing them, trying to relate the ideas to the diversity of current coordination languages would almost certainly obscure the issues. Therefore, only one coordination system will be used as a focus for the ideas presented — LINDA. There are two main reasons for choosing LINDA. Firstly, it is (probably) the earliest explicitly ‘coordinative’ language, and (certainly) one of the best-known of the current coordination languages. Secondly, it is a very flexible model which subsumes most others (although, as shall be seen later, it still has some deficiencies in its coordinative features).

However, many of the issues raised here are not LINDA-specific — indeed, much of the point of the paper is in identifying ‘purely’ coordinative issues which exist independently of any specific coordination model.

The LINDA model considered here is the ‘classical’ one [5], with in, out, rd, and eval primitives extended to work with multiple tuple spaces.

1.3 Motivation

In LINDA-like systems, agents (processes) coordinate by means of generative communication [8]. This term is intended to suggest that the ‘carrier particles’ of coordination are created by agents, but then have existence independent of their creators. Using this medium, LINDA is able to provide (nearly) all the coordination mechanisms normally used in parallel processing situations. There are other ways of achieving coordination which do not fall neatly into the category of data communications, as will be seen later. However, why should they be needed? There are two reasons:

1. This is in contrast to conventional synchronous message-passing communication models in which the message only has a fleeting existence at the instant of synchronisation. It therefore cannot exist independently of the sender and receiver.