Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 117 (2005) 375–392
www.elsevier.com/locate/entcs

A Run-Time Environment for Concurrent Objects With Asynchronous Method Calls

Einar Broch Johnsen, Olaf Owe, and Eyvind W. Axelsen

Department of Informatics, University of Oslo
PO Box 1080 Blindern, N-0316 Oslo, Norway
Email: {einarj,olaf,eyvindwa}@ifi.uio.no

Abstract

A distributed system may be modeled by objects that run concurrently, each with its own processor, and communicate by remote method calls. However objects may have to wait for response to external calls; which can lead to inefficient use of processor capacity or even to deadlock. This paper addresses this limitation by means of asynchronous method calls and conditional processor release points. Although at the cost of additional internal nondeterminism in the objects, this approach seems attractive in asynchronous or unreliable distributed environments. The concepts are illustrated by the small object-oriented language Creol and its operational semantics, which is defined using rewriting logic as a semantic framework. Thus, Creol specifications may be executed with Maude as a language interpreter, which allows an incremental development of the language constructs and their operational semantics supported by testing in Maude. However, for prototyping of highly nondeterministic systems, Maude's deterministic engine may be a limitation to practical testing. To overcome this problem, a rewrite strategy based on a pseudo-random number generator is proposed, providing Maude with nondeterministic behavior.

Keywords: Object orientation, asynchronous method calls, operational semantics, rewriting logic, nondeterministic rewrite strategies

1 Introduction

The importance of inter-process communication is rapidly increasing with the development of distributed computing, both over the Internet and over local networks. Object orientation appears as a promising framework for concurrent and distributed systems [20], but object interaction by means of method calls is usually synchronous and therefore less suitable in a distributed setting. Intuitive high-level programming constructs are needed to unite object orientation and distribution in a natural way. In this paper programming

1571-0661/$ – see front matter
doi:10.1016/j.entcs.2004.06.012

constructs for concurrent objects are proposed with an object-oriented design language Creol, based on *processor release points* and *asynchronous method calls*. Processor release points are used to influence the implicit internal control flow in concurrent objects. This reduces time spent waiting for replies to method calls in a distributed environment and allows objects to dynamically change between active and reactive behavior (client and server).

We consider how object-oriented method calls, returning output values in response to input values, can be adapted to the distributed setting. With the *remote procedure call* (RPC) model, an object is brought to life by a procedure call [6]. Control is transferred with the call so there is a master-slave relationship between the caller and the callee. Concurrency is achieved by multiple execution threads, e.g. Hybrid [26] and Java [19]. In Java the interference problem related to shared variables reemerges when threads operate concurrently in the same object, and reasoning about programs in this setting is a highly complex matter [1,11]. Reasoning considerations suggest that all methods should be serialized [9], which is the approach taken by Hybrid. But with serialized methods, the caller must *wait* for the return of a call, blocking for any other activity in the object. In a distributed setting this limitation is severe; delays and instabilities due to distribution may cause considerable waiting. In contrast, message passing is a communication form without transfer of control. For synchronous message passing, as in Ada's Rendezvous mechanism, both sender and receiver must be ready before communication can occur. Method calls may be modeled by pairs of messages, on which the two objects must synchronize [6]. For distributed systems, this synchronization still results in much waiting. In the asynchronous setting, messages may always be emitted regardless of when the receiver accepts the message. Communication by asynchronous message passing is well-known from e.g. the Actor model [2,3]. However, method calls imply an ordering on communication not easily captured in the Actor model.

In this paper, method calls are taken as the communication primitive for concurrent objects and given an operational semantics reflected by pairs of asynchronous messages, allowing message overtaking. The result resembles programming with so-called future variables [8,10,16,28,29]; computation may continue until the return value of the call is explicitly needed in the code. To avoid blocking the object at this point, we propose interleaved method evaluations in objects by defining potential processor release points in method bodies using inner guards. Hence, present activity may be suspended, allowing the object's invoked and enabled methods to compete for the free processor.

The operational semantics of Creol has been defined in rewriting logic [23], which is supported by the executable modeling and analysis tool Maude [13].