ML as a Basis for Distributed Object Management

Jeannette Wing, Linda Leibengood, Greg Morrisett, and Scott Nettles
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The Venari Project at CMU is interested in performing queries over objects residing in a distributed set of persistent repositories. These queries are "content-addressable" in the sense that they are based on the objects' semantics. A sample application would be one where a person at a workstation writes a first-order predicate as a query to retrieve all procedures whose pre- and post-condition specification "satisfy" the query, where those procedures are in a library that lives at a site remote from the workstation. Here, the objects are the procedures; the repository is a program module library; the specifications represent the procedures' semantics.

In the context of ML, we are looking at three language- and systems-related issues: distribution (to perform remote queries), concurrency (to expedite search at a single site), and persistence (to prevent the loss of data if hardware failures occur). For each of these issues we are as keen in understanding their semantics, especially their impact on current ML semantics, as well as implementing language extensions to support them. We would like to design, implement and package these extensions so that the entire ML community would find them useful and indeed, use them. We have already made initial progress in all three areas; the most, for concurrency.

ML + Distribution

Remote evaluation refers to the ability of a client process to send code to a server process for evaluation or execution, thereby allowing a client to extend the basic interface exported by a server and providing the client with fine-grained control over the distribution of a computation. Our initial implementation of remote evaluation for ML will restrict client code executed by the server to the functional part of ML. Support for transmitting values of all basic types (except refs) and a mechanism for incorporating user-defined encode and decode functions will be provided.

ML + Concurrency

We are currently designing C-threads like extensions to ML. These extensions provide the ability to fork new processes, to manipulate mutexes, to synchronize on condition variables. We will implement these extensions initially on top of Mach.

ML + Persistence

We are designing extensions to ML's run-time system to provide persistent heap-allocated garbage-collected storage for ML objects. The implementation will build on top of a generic "recoverable virtual memory" interface currently being built at CMU. We plan to extend ML syntax to let users distinguish between persistent objects and regular (volatile) ones.