Convergent Data Types for Autonomous Operation

Carlos Baquero Francisco Moura

{cbm,fsm}@.di.uminho.pt
http://gama.di.uminho.pt/

Universidade do Minho, Portugal

Despite continuous advances in ubiquitous connectivity, autonomous operation is still needed when availability is at issue. As a result, applications may generate or use local copies of data, and thus replica consistency should be addressed. However, mobility precludes the use of most of the well-known techniques for ensuring strong consistency, so much so that some applications even choose to ignore the existence of replicas. For example, personal information managers, Mail and News readers and Web browsers, often store persistent state in local files, but tacitly assume a single copy.

This work addresses replica reconciliation taking advantage of pairwise encounters of replica holders, in the line of Bayou and Ficus. It aims at creating a generic framework for reconciliation of data that is concurrently updated by placing some restrictions on the data types that abstract replicated data. In fact data types are restricted in order to completely avoid conflicts upon reconciliation. Operations are allowed, at all times, over any replica. New replicas can be forked at any time from a given replica. Any pair of replicas can be reconciled at any time, as long as both are available for computing the reconciliation procedure, so that all replicas can eventually converge.

Although these rules strongly constrain data types, they have the clear advantage of placing no restrictions on the availability of data. For many cases this enables a degree of sharing for environments that previously had no sharing at all. Sharing can be added to existing applications by creating application-specific reconciliators.

We have identified several basic convergent data types, namely grow only sets and sets where removals have priority over insertions. In addition, there are two structuring data types, simple aggregation by cartesian products and grow only partial functions, that act as heterogeneous containers and recursively apply the appropriate fork and merging procedures. This constrasts with ad-hoc construction and enables the derivation of well-founded reconciliators.

A formal description of the underlying environment and data types can be obtained on the web at `http://www.di.uminho.pt/~cbm/ps/scadt3.ps`. The present pool of data types holds enough expressive power to construct reconciliators for Netscape folded bookmark files and for replicated mail folders. We expect to apply this technique to the pairwise file reconciliators identified on the Ficus and Coda optimistic mobile file systems. This will help to identify and introduce new convergent data types.

The initial implementation test in C++ relied heavily on the STL, but the lack of adequate support of runtime types for the generic containers suggested a more appropriate language. Java was selected and the pool of data types was implemented on a Java class hierarchy. All instances of these classes share the ability to fork new ones and to compute a join with another instance of the same type.

In order to extend the support for duplication and reconciliation of aggregate convergent data types to other languages, a simple intermediate language based on S-expression representations was devised. The translation back and forth into this language follows the guidelines for Java serialization and externalization but creates a language independent representation.

A package with the Java framework and an example reconciliator for Netscape bookmarks will be made publicly available by the end of summer 1997.