Reasoning About Parallelism

Using Operator Nets

J.I. Glasgow, G.H. MacEwen, D.B. Skillicorn
Queen's University, Kingston

Introduction

The graphical language, operator nets [Ashcroft85], provides a method for describing interprocess communication and parallelism in a distributed computing environment. An operator net consists of a set of nodes and a set of directed arcs corresponding to infinite sequences of data values from some underlying set. A program in the language consists of a set of equations that relate the output arc of a node to a function applied to the input arcs of the node. These equations can themselves be considered a language: the functional language Lucid [Wadge85].

A behavioral semantics for operator nets has been defined [Glasgow 1987a] in which properties of a distributed system are expressed in the operator net model in terms of the histories of an operator net and events that occur in such a net.

Operator nets can be used to express either fine or large grained parallelism. In the behavioral model for operator nets, a node and its associated equations are considered a process that consumes input sequences and produces an output sequence. These process nodes can either correspond to operators (fine grained) or Lucid functions of any complexity (large grained). Each arc of a net is interpreted as a communication channel that carries messages from one process to another.

Current research in using operator nets to specify parallelism in distributed systems is centred around four related projects:

1) Development of a formal theory of operator nets for reasoning about distributed systems
2) Specifying and verifying security properties of computer systems
3) Specifying and verifying real-time properties of computer systems
4) Incorporating representations of knowledge into specifications

In the remainder of this abstract, we summarize each of these projects.

Formal Theory of Operator Nets

One of the major problems with formal verification is that the languages used to reason about programs differ greatly from those in which systems are built. The underlying foundation of Lucid as a programming language was to provide a programming and proof technique that shared a single coherent structure. This was accomplished by defining the semantics of Lucid completely denotationally with mathematical properties such as referential transparency. Unfortunately, the program transformation rules provided by Lucid are sufficient for only a very limited
kind of formal reasoning. We are currently developing a proof system based on a behavioral semantics for operator nets. This theory will allow us to formally verify that Lucid specifications correspond to abstract specifications written in a logic language for operator nets.

The formal theory for operator nets is based on a behavioral semantics that intuitively models computations in a distributed system. This model has been extended to also allow for reasoning about knowledge, where knowledge is defined as a function of a process's initial knowledge, input history and reasoning capability.

We are currently extending the behavioral semantics to deal with the difficult problem of non-determinism. Many actual systems have non-deterministic components (servers in operating systems, transmission media in networks) and we wish to model these without giving up the functional framework.

**SNet Multilevel Secure System**

SNet is a prototype multilevel secure system providing a focus for a project investigating methods for specifying and verifying security properties of computer systems [Glasgow 1985,1987b, MacEwen 1987]. In particular, we are interested in methods that allow a natural decomposition of a security model into component models and subsequently functional components that can be verified and implemented independently from other models and components. Security properties of SNet have been specified and verified using operator nets. This approach has been particularly successful since it has allowed us to specify abstract constraints, using a behavioral semantics for operator nets, and concrete executable constraints using a Lucid specification.

The SNet design comprises host machines, secure terminal servers, and secure downgraders connected via an untrusted network. The current prototype contains three hosts, one downgrader, and one terminal server based on NS32000 processors connected via an Ethernet. The Lucid specification contains approximately fifty function nodes of varying functionality. The implementation is a network of Concurrent Euclid processes that mirror the structure of the operator net specifications.

**Real-Time Specification Using Operator Nets**

This project involves the development of a methodology for specifying real time systems using Lucid and operator nets [Skillcorn 1986]. Given any Lucid specification of a system, the approach can automatically construct an augmented specification. The new specification consists of the original operator net and two additional operator nets that describe the early and late time constraints of the system. When constraints on times at which inputs are available and outputs are required have been provided, the augmented specification captures the timed behavior of the system. Because the augmented specification is executable, potential hot-spots can be easily detected and the trade-offs between performance of components of the system as a whole investigated in fine detail. It seems likely, also, that some classes of
architectural constraints (degree of concurrency, number of processors) can be expressed within the formal model.

Reasoning About Knowledge

The use of knowledge to reason about about distributed systems has been the subject of much recent research (e.g. [Halpern 1985]). Most of this work has been focussed on the concept of common knowledge, corresponding to public information. The behavioral semantics for operator nets has been extended to allow for knowledge-based specifications. Knowledge in this model is defined as a function of a process's initial knowledge, input history and reasoning capability. We also combine the ability to reason about knowledge with the ability to express relationships about time, causality and dependency.

Knowledge-based specifications have proven to be useful for expressing security properties of a distributed system [Glasgow 1988]. At the abstract level, we can express high level constraints on what a process can "know" based on its security level. Using the dependency relation of the behavioral semantics for operator net, we can also define an equivalent operational definition for security.

References

[Ashcroft 1985]

[Glasgow 1985]

[Glasgow 1987a]

[Glasgow 1987b]

[Glasgow 1988]

[Halpern 1985]
[MacEwen 1987]

[Skillicorn 1986]

[Wadge 1985]