A BEHAVIORAL SEMANTICS FOR NONDETERMINISTIC OPERATOR NETS

Janice Glasgow
Department of Computing and Information Science, Queen's University

Prakash Panangaden
Computer Science Department, Cornell University

ABSTRACT

1. Introduction

An operator net [Ashcroft 1985] consists of a set of nodes, a set of arcs, and a set of equations that relate the output arcs of nodes to functions or operators applied to the input arcs. These equations themselves can be considered a language: the functional programming language Lucid [Wadge 1985].

While operator nets are syntactically similar to dataflow nets, they completely separate the operational semantics from the mathematical semantics. An operational semantics for Lucid and operator nets, that corresponds to communication in a distributed system, has previously been defined [Glasgow 1987a]. This model has been used to specify and verify properties of secure distributed systems [Glasgow 1987b].

A current deficiency of the use of operator nets for reasoning about distributed systems is the inability to express nondeterministic behavior. In this abstract we present an approach to extending the behavioral semantics for operator nets to permit nondeterministic operators. This is achieved by adding a special symbol to the algebra of the operator net which allows us to test for availability of data on an input history sequence. This is similar to the approach described in [Panangaden 1984], where a fixed-point theory for the semantics of nondeterministic operators in a static dataflow network is presented. A formal denotational semantics for nondeterministic operator nets will not be included in the abstract but is currently under development.

2. Behavioral Semantics for Operator Nets

In the computational model described by the behavioral semantics for operator nets, each arc corresponds to a communication channel and each node corresponds to a process. Properties of a distributed system are expressed in terms of the histories of an operator net. A history h is a set of ordered pairs (a, a^h) where a is an arc and a^h is the value of arc a in history h. Each operator net is associated with a continuous algebra A consisting of elementary data objects. The value of an arc in a history is in the domain of nonintermittent sequences of elements from the universe of algebra A.

The basic unit of computation in the model is an event. For any subset of nodes of an operator net, an event corresponds to processing occurring at these nodes. At most, one new value may be added to a history sequence of an output arc for such a node. An event is denoted as a triple (h, P, b') where h is the history in which the event occurs, P is the subset of processes where computing is occurring and b' is the history resulting from the event. A computation in an operator net is a sequence of events, where each event in the sequence occurs in the history
resulting from the previous event. In this model, concurrency is represented by nondeterministic interleaving.

3. Behavior of Nondeterministic Operator Nets

Our approach to incorporating nondeterminism into operator nets involves adding the ability to test for the presence of data on an input channel. This is captured by adding an abstract concept of "delay" to the model. We will interpret nodes of an operator net as processes with internal clocks, where ticks of the clock denote time periods during which local processing occurs. These local clocks are in no way synchronized with the clocks of any other processes. Most processes of the operator net remain determinate; nondeterminism is achieved by allowing arbitrary delays in the history sequences.

Given an operator net based on a continuous algebra $A$, we extend the domain of the algebra to contain a special symbol $\tau$, called a hiaton [Park 1982]. An hiaton is interpreted as "no data present" on a history sequence. We also extend the operator net model to include two special operators called $\text{poll}$ and $\Delta$. The poll operator is defined as follows:

$$\text{poll}(x) = \text{not } (x \text{ equal } \tau)$$

Thus poll translates a $\tau$-sequence into a predicate sequence which is true whenever data is present on the input sequence. For example:

$$\text{poll}(\langle 1,2,\tau,3,\tau \rangle) = \langle t,t,f,f,t,f \rangle$$

The $\Delta$, or finite delay operator, is used to insert hiatalons into a sequence. If data is available then the delay operator just makes a copy of it, otherwise it produces a hiaton.

We will assume that the delay operator will only occur in compositions with a poll operator. Thus, we will have [* mention that this restriction is imposed for simplicity, a complete theory would consider polls and delays everywhere *] subnets of the form:

$$\xrightarrow{x} \Delta \xrightarrow{y} \text{poll} \xrightarrow{z}$$

with equations:

$$x = \Delta(x);$$

$$y = \text{poll}(y);$$

The behavioral semantics for nondeterministic operator nets is similar to that for deterministic nets. The only modification is when considering an event that contains a delay in its set of processes. In such an event, if there is no data available as new input to the delay operator then a hiaton is produced. For example, if $h1$ is the history for the above nondeterministic operator net such that:

$$x^{h1} = \langle 1,2,3 \rangle \text{ and } y^{h1} = \langle 1,2 \rangle$$

then an event occurring in $h1$ with processing at the delay node would result in a new history $h2$ such that:

$$x^{h2} = \langle 1,2,3 \rangle \text{ and } y^{h2} = \langle 1,2,3 \rangle$$

If similar processing now occurs at the same process the result would be a history $h3$, where

$$x^{h3} = \langle 1,2,3 \rangle \text{ and } y^{h3} = \langle 1,2,3,\tau \rangle.$$
4. Fair Nondeterministic Merge

Given a poll operator with \( \tau \)-sequence inputs, we define a fair nondeterministic merge function in Lucid as:

\[
\text{merge}(x,y) = \begin{cases} 
\text{if poll first } x \text{ then} \\
(\text{first } x) \text{ ifby merge}(y, \text{next } x) \\
\text{else merge}(y, \text{next } x)
\end{cases}
\]

This function polls the \( \tau \)-sequence \( x \). If the first value is not a hiaton then it adds it to the output sequence. Then merge is called recursively, alternating the order of the input streams. For this merge to be fair, we make two assumptions on the input sequence to the poll operator. First, there may be no indefinite delay (subsequence of hiatons) between defined values on the sequence. Second, there can be no indefinite wait between hiatons on the input sequence. Given that we are modeling a process which is synchronized using the hiatons, these assumptions reduce to an assumption of liveness of the process. Note however that arbitrary delay in the propagation of data prevents interprocess synchronization.

5. Future Work

To formally define a theory for the behavioral semantics of nondeterministic operator nets, we must first define a semantics for nondeterministic Lucid. The fixed-point semantics for Lucid and operator nets is complicated by the fact that conventional domain techniques cannot be used. This is because powerdomain methods can handle only bounded nondeterminism whereas fairness or finite delay introduces unbounded nondeterminism. Again following the lead of earlier work [Panagaden 1985], we will define the semantics of nondeterministic operator nets by employing the concept of abstract interpretation. This idea involves mapping a semantic domain onto a simplified domain and reinterpreting language constructs in terms of the simplified domain.

References

[Ashcroft 1985]

[Glasgow 1987a]

[Glasgow 1987b]

[Glasgow 1988]

[Panangaden 1984]

[Park 1982]

[Wadge 1985]

89