

# AN OPERATIONAL MODEL FOR A LOGIC OF TIME AND KNOWLEDGE

Janice Glasgow

Department of Computing and Information Science  
Queen's University, Kingston

## 1 Introduction

An operator net is a graph consisting of nodes and directed arcs. A program in the language of operator nets is a set of equations that relates the output arc of a node to a Lucid function or operator applied to the input arcs of the node. Previously, an operational semantics for operator nets has been defined that provides a formal model for a distributed system. In this paper we demonstrate that the operational semantics can be interpreted as a modal logic for time and knowledge. As such, it provides an intermediate specification language for distributed systems.

Section 2 defines a modal logic for time and knowledge. This logic is constructed using a possible worlds semantics that combines standard knowledge and temporal logics. Such a logic provides an intuitive language for describing the behavior of a distributed system in terms of "what the processes of the system know". Adding time to this model of behavior also allows for the specification of causality and liveness properties for distributed systems.

Section 3 summarizes the operational semantics for operator nets and demonstrates how this semantics can be used to define a Kripke structure for the logic. This structure provides a basis for verifying that operator nets preserve safety and liveness properties that are expressed in the modal logic.

## 2 A Logic for Time and Knowledge

In this section we define a logic of knowledge and time. This logic is constructed using a *possible worlds model*, [Hin62], a well understood formalism for defining theories of knowledge. In this approach, one defines a set of “worlds”, the so called *possible worlds*, and a family of indistinguishability relations, one for each subject, that say which worlds an subject can tell apart. A subject is said to know a formula if the formula is true in all worlds considered possible by the subject. This view matches the computational situation in distributed systems very well [HMS4] as the possible worlds correspond to the global states and the possibility relations are determined by the local states of processes.

In developing a formalism for reasoning about distributed systems, we are concerned with the relationship between knowledge and time or, more precisely, between knowledge and the execution of a system. Thus our logic for reasoning about distributed computations will include a combined theory of knowledge and time. This theory is one that has been developed to reason about security in distributed systems [GM90]. When considering security applications, we also include deontic operators for permission and obligation in the theory.

Later in the paper we give concrete interpretations for the possible worlds semantics for distributed systems. In particular we relate the set of possible worlds to the set of histories of an operator net.

Recent research in the area of reasoning about knowledge in distributed systems has focused on studying agreement protocols. These protocols are formalized via *common knowledge* [HM84]. Such knowledge is important in situations in which every process in a network must know a fact before an action can be taken.

Temporal logic [Pri57] is a well-understood formalism for reasoning about systems [Pnu81]. In defining such a theory, there are two contrasting interpretations regarding the nature of time: *linear* and *branching* [Lam80, EH82]. In the linear interpretation, one talks about the truth of formulae with respect to a *given* computation sequence. The branching interpretation allows one to talk about a world and *all* of its future worlds. For the purpose of this paper we wish to reason about computations of an operator net; that is, properties that must hold in all possible future computation paths. Therefore we use the branching interpretation of time.



## 2.1 A Logic of Knowledge

In this section we review a standard theory for reasoning about knowledge [Hin62]. This theory is an S5 modal logic and has previously been used for reasoning about distributed system protocols [HM84]. The concept of knowledge that we define for our theory is very similar. The main difference is that our theory assumes that knowledge is monotonic over time; that is, a subject can only gain knowledge, not lose it.

The language for reasoning about knowledge consists of: a set of subjects labeled  $1 \dots n$ ; a set of primitive propositions  $\Sigma$ ; and for all subjects  $i$ , a modal operator  $K_i$ . A proposition of the form  $K_i\phi$  states that *subject  $i$  knows that proposition  $\phi$  is true*.

The notion of possible worlds can be formalized by using *Kripke structures* [Kri63]. In a system with  $n$  subjects the Kripke structure is defined as a tuple  $M = (S, \pi, \kappa_1, \dots, \kappa_n)$  where:

1.  $S$  is the set of possible worlds,
2. The relation  $\kappa_i$  for a subject  $i$  is an equivalence relation (reflexive, symmetric and transitive) on  $S$ . We say that two worlds are *indistinguishable* to subject  $i$  if they belong to the same equivalence class in  $\kappa_i$ .
3. A formula is defined to be true or false in a possible world. We write  $s \models \phi$  to denote that formula  $\phi$  is true in world  $s$ .

For each world  $s$  in  $S$  and each primitive proposition  $\phi \in \Sigma$ ,  $\pi$  assigns a truth value to  $\phi$  in  $s$  (i.e.  $\pi(s, \phi) \in \{true, false\}$ ). We define the notion of a formula being true in a world via the  $\models$  relation.

$$s \models \phi \text{ iff } \pi(s, \phi) = true \ (\forall \phi \in \Sigma).$$

$$s \models \phi \wedge \psi \text{ iff } s \models \phi \text{ and } s \models \psi$$

$$s \models \neg\phi \text{ iff not } s \models \phi$$

$$s \models K_i\phi \text{ iff } \forall s' \text{ such that } (s, s') \in \kappa_i, s' \models \phi$$

A formula  $\phi$  is said to be *valid* if it is true in all worlds  $s \in S$ . We denote this as  $\models \phi$ .

A sound and complete axiom scheme that characterizes the logic described above has previously been studied extensively by philosophers. These axioms for knowledge are:

Axiom K1: all propositional tautologies

Axiom K2:  $K_i\phi \rightarrow \phi$

Axiom K3:  $K_i(\phi \rightarrow \psi) \rightarrow K_i\phi \rightarrow K_i\psi$

Axiom K4:  $K_i\phi \rightarrow K_iK_i\phi$

Axiom K5:  $\neg K_i\phi \rightarrow K_i(\neg K_i\phi)$

All tautologies of propositional logic are axioms in the system. The second axiom implies that a subject cannot know anything that is false. This distinguishes knowledge from belief. Axiom 3 states that a subject knows all things that can be deduced from its knowledge. The final two axioms state that a subject has introspective ability.

The theory includes the proof rules of propositional logic as well as a rule that states that if a formula is valid (true in all possible worlds) then a subject knows that formula:

Rule 1: If  $\phi$  is valid then so is  $K_i\phi$

In the following subsection we expand the Kripke structure for knowledge to include a second relation that allows us to define a modal operators for time.

## 2.2 Knowledge and Time

The temporal logic for knowledge presented in this paper is based on a branching time structure. In particular, we use the logic *UB*: the *unified* system of *branching* time [BAMP81] to develop our theory. In the *UB* system the underlying model consists of a branching tree of all possible paths of reachable possible worlds. Temporal operators are defined that correspond to quantification over these paths.

The basis of the language for our theory is the propositional calculus. As well as the modal operators  $K_i$  for knowledge, we include two of the six



temporal operators of *UB*. These are the operators:  $\forall\Box$  (*always*) and  $\forall\Diamond$  (*eventually*). The first symbol in these operators denotes the quantification over paths, while  $\Box$  and  $\Diamond$  denote temporal quantification over the selected paths. Given a world  $s$ , a path defined as a sequence of states, and a formula  $\phi$  we interpret the temporal operators applied to  $\phi$  as follows:

- $s \models \forall\Box\phi$  iff  $\phi$  is true in all worlds along all paths initiated at  $s$ .
- $s \models \forall\Diamond\phi$  iff  $\phi$  is true for some world along all paths initiated at  $s$ .

We now give a more formal presentation of the language and semantics for the temporal logic for knowledge. The alphabet of the language consists of:

1. A denumerable set  $\Gamma$  of primitive proposition letters  $\phi, \psi, \dots$ ;
2. The logical symbols  $\top$  (truth) and  $\perp$  (falsehood), and the logical connectives  $\neg$  (negation),  $\wedge$  (conjunction),  $\vee$  (disjunction) and  $\rightarrow$  (implication);
3. The modal operators  $K_i$  (knowledge) for all subjects  $i$ ;
4. The temporal operators  $\forall\Box$  (always) and  $\forall\Diamond$  (eventually);
5. The parentheses  $( )$ .

The set of well formed formulae for the language is defined as the smallest set  $W$  such that:

1. Every proposition letter in  $\Gamma$ ,  $\top$  and  $\perp$  are in  $W$ ;
2. If  $\phi$  and  $\psi$  are in  $W$  then so are  $\neg\phi$ ,  $\phi \wedge \psi$ ,  $\phi \vee \psi$ ,  $\phi \rightarrow \psi$  and  $(\phi)$ ;
3. If  $\phi$  is in  $W$  then so are  $K_i\phi$  (for all subjects  $i$ ),  $\forall\Box\phi$  and  $\forall\Diamond\phi$ .

We now define a possible worlds model for the extended language. This model assumes a binary relation  $R$  on possible worlds. We say that for any two worlds  $s$  and  $s'$ ,  $(s, s') \in R$  if world  $s'$  is *directly reachable* from world  $s$ .

The model includes quantification over a set of paths defined by the relation  $R$ . A path  $p$  is a sequence  $(s_0, s_1, \dots)$  of worlds such that for all adjacent

worlds  $s_i, s_{i+1}$  in path  $p$ ,  $(s_i, s_{i+1}) \in R$ . We say that the path  $p$  is initiated in world  $s_0$  (or  $first(p) = s_0$ ). When considering distributed system applications, a path corresponds to a computation (sequence of state transformations) for the system.

The theory assumes the truth assignment rule for knowledge:

$$s \models K_i \phi \text{ iff } \forall s' \text{ such that } (s, s') \in \kappa_i, s' \models \phi$$

where  $\kappa_i$  is an equivalence relation over worlds in the model. In the following rules for the temporal operators we assume that the quantification of  $p$  is over paths in the model and the quantification of  $s, s'$  and  $s''$  is over possible worlds.

$$s \models \forall \Box \phi \text{ iff } \forall p \forall s' (first(p) = s \wedge s' \in p \rightarrow s' \models \phi)$$

$$s \models \forall \Diamond \phi \text{ iff } \forall p (first(p) = s \rightarrow \exists s' (s' \in p \wedge s' \models \phi))$$

We now present a deductive system of axioms and inference rules for proving validity of formulae in the theory. We include the axioms and rule of knowledge. We also include the following standard axioms for branching temporal logic:

$$\text{Axiom T1: } \forall \Box \phi \rightarrow \phi$$

$$\text{Axiom T2: } \forall \Box (\phi \rightarrow \psi) \rightarrow (\forall \Box \phi \rightarrow \forall \Box \psi)$$

$$\text{Axiom T3: } \forall \Box \phi \rightarrow \forall \Diamond \phi$$

$$\text{Axiom T4: } \forall \Box \phi \rightarrow \forall \Box \forall \Box \phi$$

When considering a deduction system for operator nets and the logic, we may wish to include a more complete temporal logic. For example the *UB* operator for *next* would allow us to express inductive over possible worlds in a path.

The rules of inference for the system include Rule 1 for knowledge as well as modes ponens and the following generalization rule:

$$\text{Rule 2: } \text{If } \phi \text{ is valid then so is } \forall \Box \phi$$



### 3 A Possible Worlds Interpretation

In this section we illustrate how the possible worlds model can be interpreted in terms of the operational semantics for operator nets. We begin by summarizing the operational semantics and then relating the Kripke structure for our logic of time and knowledge to the behavior of an operator net.

#### 3.1 Operational Semantics for Operator Nets

An operator net is a graph consisting of nodes and directed arcs [AJ85]. Each arc of a net has an associated infinite sequence of values (history sequence): each node corresponds to a function that accepts input history sequences from incoming arcs and produces an output sequence along its outgoing arc.

An operational semantics for operator nets has previously been defined that relates the nodes of a net to the processes of a distributed system and the arcs to the communication channels that carry message values from one process to another [GM89]. This semantics defines an operator net in terms of its possible *computations*, where a computation is a possibly infinite sequence of events  $e_1, e_2, e_3, \dots$ . An *event* denotes a transition from one history to another. We say that an event  $(h, h')$  occurs in history  $h$  and results in history  $h'$ . Similar to the concept of a state, a *history* in the model associates each arc of a net with a partially defined history sequence. Thus a computation can be considered as a sequence of events:  $(h_0, h_1), (h_1, h_2), \dots$  such that  $h_1$  is an initial history and for each event  $(h_i, h_{i+1})$  in the computation,  $h_i \leq h_{i+1}$ . A history  $h$  approximates ( $\leq$ ) a history  $h'$  if for each arc  $x$  in the net, the history sequence  $h(x)$  is a prefix of the sequence  $h'(x)$ . As well, for any history  $h$  and arc  $x$  it must be the case that  $h(x)$  approximates the meaning of  $x$  given by the denotational semantics for operator nets.

The set of possible computations defines an operator net by specifying all of its possible behaviors. An extension of this semantics that includes a nondeterministic merge operator has also been defined and proved compositional [Bak90]. We have demonstrated that the semantics can be used to specify the behavior of secure distributed systems [GM87] and the behavior of real-time distributed systems [SG89]. In this section we demonstrate how this semantics can also be considered as an operational model for a modal logic of time and knowledge. As such it will provide an intermediate specification language that is halfway between a high-level logic and a practical

implementation.

### 3.2 Kripke Structure

We can now view an operator net as a Kripke structure. We begin by considering an operator net to be defined as:

- A domain  $D$  for the algebra which the functions for the operator net are based on.
- A set  $N = \{1, \dots, n\}$  of node identifiers, where  $n$  is the total number of nodes in the net. We associate with each node  $i$ , a set of input arcs  $\{x_1, \dots, x_k\}$  and an output arc  $y$ .
- A set  $H$  of histories of the operator net.
- A set  $C$  of computations of the operator net.

For history  $h \in H$  with input arcs  $x_1, \dots, x_k$  and output arc  $y$ , we define the *local history* for node  $i \in N$  as the tuple  $h[i] = \langle h(x_1), \dots, h(x_k), h(y) \rangle$ .

To define a Kripke structure for time and knowledge we must define:  $S$ , the set of possible worlds;  $\pi$ , the truth assignment for primitive formulae;  $\kappa_i$ , the equivalence relations for the possible worlds; and  $R$ , the accessibility relation for the temporal logic. In the remainder of this subsection, we define each of these components of the structure in terms of operational semantics for operator nets.

The set of nodes  $N$  of an operator net defines the set of subjects for our model and the set of histories  $H$  defines the set of possible worlds. We consider two histories  $h$  and  $h'$  to be in the equivalence relation  $\kappa_i$  for a node  $i$  if  $i$  is unable to distinguish between the two histories; that is, the local histories are the same for node  $i$  and histories  $h$  and  $h'$ . More formally:

For all histories  $h$  and  $h' \in H$  and all nodes  $i \in N$ ,  

$$(h, h') \in \kappa_i \leftrightarrow h[i] = h'[i].$$

The direct accessibility relation  $R$ , that defines the structure of paths, is defined by the valid events and histories of an operator net. We say that history  $h'$  is directly reachable from history  $h$ ,  $(h, h') \in R$ , if  $(h, h')$  is a



valid event for the operator net. Thus a path  $p$ , such that  $first(p) = h$ , corresponds to a computation of an operator net initiated at history  $h$ .

Finally, we define the truth assignment operator  $\pi$  for primitive formulae. First, though, we must consider what we mean by a formula in an operator net. The simplest (but not the only) interpretation of a primitive formula is as an element of a history sequence. More realistically, we can consider a formula as a function of a message. Thus, we define the set  $\Sigma$  as a function  $F$  such that  $F : D \rightarrow \Sigma$ . Now, for any history  $h$  and arc  $x$  of the operator net, the history sequence  $h(x) = \langle x_0, x_1, \dots \rangle$  can be mapped onto a set of primitive formulae  $F(x_0), F(x_1), \dots$ . If this is the case then we say that for any history  $h$  the truth value assigned to primitive formula  $\phi$  is true if and only if  $\phi = F(x_i)$  and  $x_i$  is a message on the history sequence  $h(x)$  for some arc  $x$  of the net.

In dealing with certain applications, such as security, it may be desirable to also express such formulae such as *message  $d$  arrived at time  $t$* , where time  $t$  is a local time for a particular process. For such cases, the definition of set of primitive formulae would be more complex than a function from messages to formulae.

### 3.3 Specifying Properties of Distributed Systems

Modal logics have long been considered as languages for expressing abstract properties for distributed systems (e.g. [Hal87, MP84, NP86]). It has also been demonstrated that operator nets provide for an executable specification of distributed systems that describes the components and communication channels of a system [GM89]. In the remainder of this section, we illustrate how the operator net specification can be considered as an intermediate point between the implementation of the system and the modal logic description.

Lamport has demonstrated how properties of concurrency can be expressed in temporal logic [Lam80]. These properties can be categorized into two fundamental types of properties: *safety properties*, which assert that “something bad never happens”; and *liveness properties*, which assert that “something good must eventually happen”. A safety property  $\phi$  is expressed using the always operator ( $\forall \square \phi$ ) and a liveness property  $\phi$  is expressed using the eventually operator ( $\forall \Diamond \phi$ ).

To prove a safety property  $\phi$  for an operator net, we must show that the formula  $\forall \square \phi$  is valid. That is, for all histories  $h \in H$ ,  $h \models \forall \square \phi$ . This can be

reduced to proving that the formula  $\phi$  is valid or true in all histories. A general strategy for such a proof is by induction on the length of a computation sequence. Thus we must prove that the property holds in the initial history, and given any history  $h$  if  $h \models \phi$  and  $(h, h')$  is a valid event then  $h' \models \phi$ .

To verify a liveness property  $\phi$  for an operator net, we must prove that the formula  $\forall \Diamond \phi$  is valid. That is for every history  $h \in H$  and all computation paths initiated at  $h$ , there is some history  $h'$  on that computation path for which  $\phi$  is true. The theory does not guarantee that this computation path resulting in history  $h'$  is finite so it may be necessary to augment the theory to place such a restriction on the eventuality operator. Proof of a liveness property in an operator net also relies on a liveness assumption of the processors corresponding to the nodes of an operator net. This is to guarantee that if certain events can occur then they will occur in some finite time.

The ability to ascribe knowledge to processors has also proved useful in specifying protocols and properties of a distributed system. A node  $i$  has knowledge of a formula  $\phi$  in history  $h$  if  $h' \models \phi$  for all histories  $h'$  that are indistinguishable from  $h$  for node  $i$ . More formally:

$$h \models K_i \phi \text{ iff for all } (h, h') \in \kappa_i, h' \models \phi$$

Combining knowledge and time allows for additional properties of a distributed system to be specified. It has been shown that security properties can be expressed that involve such an integrated logic [GM90]. For example, the notion of integrity can be defined as the concept that a process must *eventually know* a formula  $\phi$ . The fact that node  $i$  eventually knows formula  $\phi$  is expressed in the logic as:  $\forall \Diamond K_i \phi$ .

## 4 Discussion

The behavioral semantics for operator nets provides a formal model for distributed systems that is an intermediate point between the actual system and a modal logic specification. It allows for a separation of the operational considerations from the possible worlds semantics that makes a distributed system's description comprehensive. Timing properties and causality are captured abstractly in the semantics by considering how histories change over time. The nodes and arcs of an operator net can be used to identify the processes and communication channels of a distributed system. Restrictions



and/or requirements of communication can be specified in terms of what a process “knows”.

The interpretation of an operator net in terms of a Kripke structure is the first step in the development of a formal verification system for operator nets. Abstract properties can be expressed in the logic of time and knowledge then implemented and proven using an operator net specification.

## References

- [AJ85] E.A. Ashcroft and R. Jagannathan. Operator nets. In *IFIP TC-10 Conference on Fifth Generation Computer Architectures*. North-Holland, 1985.
- [Bak90] L. Bakker. A compositional semantics for operator nets. Master’s thesis, Queen’s University, 1990.
- [BAMP81] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *8th Annual ACM Symp. on Principles of Programming Languages*, pages 164–176, 1981.
- [EH82] E.A. Emerson and J.Y. Halpern. “sometimes” and “not never” revisited. In *9th Annual ACM Symp. on Principles of Programming Languages*, pages 127–139, 1982.
- [GM87] J.I. Glasgow and G.H. MacEwen. The development and proof of a formal specification for a multi-level secure system. *ACM Transactions on Computer Systems*, 5(2):151–184, May 1987.
- [GM89] J.I. Glasgow and G.H. MacEwen. An operator net model for distributed systems. *Distributed Computing*, 3(4):196–209, 1989.
- [GM90] J.I. Glasgow and G.H. MacEwen. A logic for reasoning about security. In *Proceedings of the Computer Security Foundations Workshop III*, 1990.
- [Hal87] J. Halpern. Reasoning about knowledge. In *Annual Reviews of Computer Science*, pages 21–35. 1987.
- [Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University, 1962.

- [HM84] J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. In *Proceedings of the 3rd ACM Conference on Distributed Computing*, pages 50–61, 1984.
- [Kri63] S. Kripke. Semantical considerations of modal logic. *Acta Philosophica Fennica*, 16:83–94, 1963.
- [Lam80] L. Lamport. “sometimes” is sometimes “not never”. In *7th Annual ACM Symp. on Principles of Programming Languages*, pages 174–185, 1980.
- [MP84] Z. Manna and A. Pnueli. Adequate proof principles for invariance and liveness properties of concurrent programs. *Science of Computer Programming*, 4:257–289, 1984.
- [NP86] V. Nguyen and K. Perry. Knowledge, communication and time. In *IFIP TC2 Working Conference on Knowledge and Data*, 1986.
- [Pnu81] A. Pnueli. The temporal logic of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
- [Pri57] A. Prior. *Time and Modality*. Oxford Press, 1957.
- [SG89] D.B. Skillicorn and J.I. Glasgow. Real time specification using lucid. *IEEE Transactions on Software Engineering*, February 1989.