ALTERNATIVE SEMANTICS FOR TEMPORAL LOGICS

E. Allen Emerson

TR-182 October 1981

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

ALTERNATIVE SEMANTICS FOR TEMPORAL LOGICS

E. Allen Emerson
Computer Sciences Department
University of Texas
Austin, Texas 78712

1. Introduction

A number of temporal logics have been proposed in which the underlying semantics of a concurrent program is expressed in terms of a set of computation paths. Various constraints on the allowable sets of computation paths can be built into a logic in an effort to ensure that the abstract computation path semantics accurately reflects essential properties of "real" concurrent programs. Three common constraints are:

1. Suffix closure - every suffix of a path is itself a path. (See [6].)

2. Fusion closure - a computation may follow a path π_1 until a state s is reached, and then follow some suffix of a path π_2 starting at an occurrence in π_2 of s. (See [8].)

3. Koenig closure - if a path can be followed for an arbitrarily long but finite length of time, it can be followed for an infinite length of time. (See [1].)

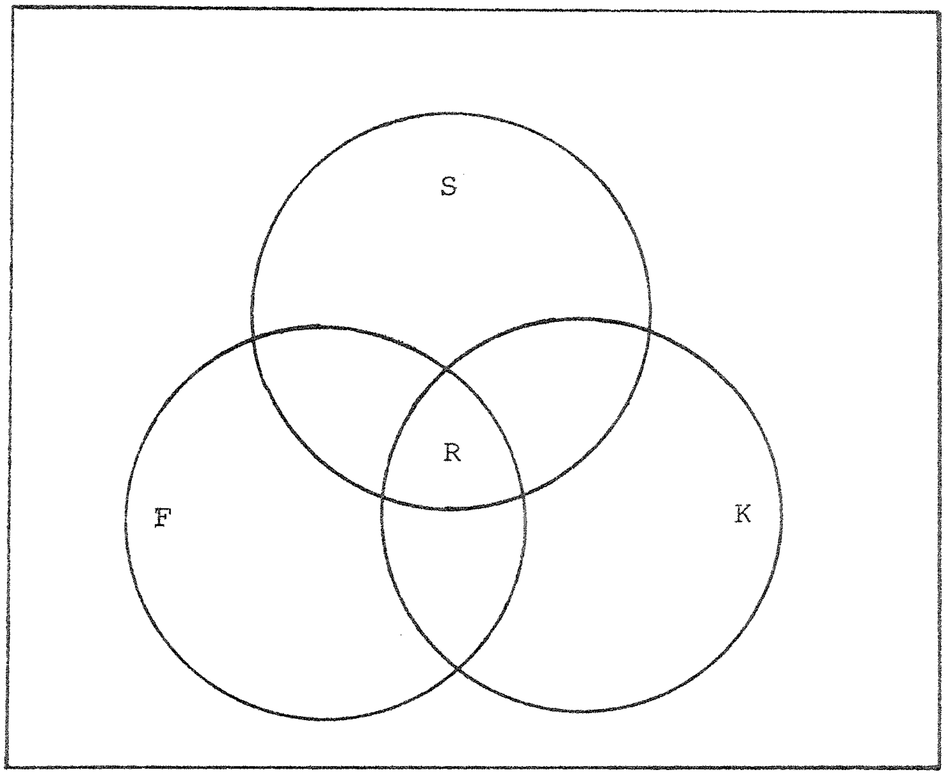
The first two constraints attempt to capture the idea that how a computation proceeds in the future depends only on its current state. The third constraint specifies a sort of continuity property: the existence of all finite prefixes of a path ensures that the whole "limit" path is itself a legitimate computation. An additional constraint is

4. R-generable - the set of paths can be generated by some binary relation R. (See [7].)

A set of paths satisfying this constraint is naturally representable as a computation tree and corresponds to computations of parallel programs executed under pure nondeterministic scheduling.

In this paper, we investigate the relationship between these four constraints. One key finding is that Constraint 4 is precisely equivalent to the conjunction Constraint 1 and Constraint 2 and Constraint 3. Furthermore, each of Constraints 1-3 is independent of the others. Our results may be conveniently summarized in the Venn Diagram of Figure 1.1 where all regions shown are nonempty.

This paper is organized as follows: Section 2 gives preliminary definitions and terminology. The technical results are proved in Section 3. Their implications and significance are discussed in the concluding Section 4.



R = the class of all R-generable path sets

S = the class of all suffix closed path sets

F = the class of all fusion closed path sets

K = the class of all Koenig closed path sets

All Regions Shown Are Non-Empty

Figure 1.1

2. Preliminary Terminology

We shall be concerned with temporal logics interpreted over structures of the form (S,Π,L) where

S is a finite or countably infinite set of states,

Π is a set of computation paths intended to provide the meaning of a program, and

L is a labelling of each state with the atomic propositions true in the state.

For simplicity, we assume that all programs are nonterminating. Thus, a path is an infinite sequence of nodes, each labelled with a state. (We will not be concerned with the arcs between nodes which might, for example, indicate which process performs which transition.) We need the following additional terminology:

$S^\omega = \{(s_0,s_1,s_2,\ldots):\forall i\geq 0 \;\; s_i \in S\}$ denotes the set of infinite sequences over S. A member $(s_0,s_1,s_2,\ldots)$ of S^ω is called a path and S^ω is the set of all possible paths over S. We use π_1, π_2, π_3, ..., etc. to denote individual paths. S^* denotes the set of finite sequences over S. $S^* = S^+ \cup \{\lambda\}$ where λ is the empty sequence and $S^+ = \{(s_0,s_1,\ldots,s_k):\exists k\geq 0 \; \forall i\leq k \;\; s_i \in S\}$ is the set of all finite nonempty sequences over S. The members of S^* are prefixes of paths. We use ρ_1, ρ_2, ρ_3, ..., etc. to denote prefixes. In general, if $\pi = (s_0,s_1,s_2,\ldots)$ is a path, then for any $k \geq 0$ $(s_0,\ldots,s_k)$ is a prefix of π and $(s_k,s_{k+1},s_{k+2},\ldots)$ is a suffix of π. (λ is a prefix of any path π.) If $\rho = (s_0,s_1,\ldots,s_k) \in S^*$ and $\pi = (s'_0,s'_1,s'_2,\ldots) \in S^\omega$, we write $\rho\pi$ to indicate the path $(s_0,s_1,\ldots,s_k,s'_0,s'_1,s'_2,\ldots)$.

We let $\Pi \subseteq S^\omega$ denote an arbitrary nonempty set of paths. We also let $R \subseteq S \times S$ denote an arbitrary nonempty binary relation on S.

As explained in Section 1, there are various constraints one might wish to impose upon Π so that the program it represents has a sensible "meaning". We formally define these constraints below:

(1) Π is <u>suffix closed</u> provided $(s_0,s_1,s_2,\ldots) \in \Pi$ implies $(s_1,s_2,s_3,\ldots) \in \Pi$.

(2) Π is <u>fusion closed</u> provided

$$(\rho_1 s\pi_1 \in \Pi \text{ and } \rho_2 s\pi_2 \in \Pi) \text{ imply } \rho_1 s\pi_2 \in \Pi.$$

(3) Π is <u>Koenig closed</u> provided

$$\left\{\begin{array}{l} \rho_1\pi_1 \in \Pi \\ \rho_1\rho_2\pi_2 \in \Pi \\ \rho_1\rho_2\rho_3\pi_3 \in \Pi \\ \quad\vdots \end{array}\right\} \text{ implies } \rho_1\rho_2\rho_3\ldots \in \Pi$$

where each $\rho_i \in S^+$ and each $\pi_i \in S$.

(4) Π is <u>R-generable</u> provided

there exists a relation $R \subseteq S \times S$ such that

$(s_0,s_1,s_2,\ldots) \in \Pi$ iff $\forall i \geq 0 \;\; (s_i,s_{i+1}) \in R$.

We define two operators $\text{PATHS}: 2^{S\times S} \to 2^{S^\omega}$ and $\text{RELN}: 2^{S^\omega} \to 2^{S\times S}$ as follows:

(a) Given a binary relation R,

$$\text{PATHS}[R] = \{(s_0,s_1,s_2,\ldots) \in S^\omega\colon \forall i \geqslant 0\ (s_i,s_{i+1}) \in R\}.$$

(b) Given a set of paths Π,

$$\text{RELN}[\Pi] = \{(s,t) \in S{\times}S\colon \exists\pi \in \Pi,\ \rho \in S^*,\ \pi' \in S^\omega$$
$$\pi = \rho s t \pi'\}.$$

Note: if Π is R-generable then Π = PATHS[R].

Remark 2.1

(a) If Π is suffix closed and $(s_0,s_1,s_2,\ldots) \in \Pi$ then for all $k \geqslant 0$ the suffix $(s_k,s_{k+1},s_{k+2},\ldots) \in \Pi$.

(b) If Π is fusion closed and $\rho_1 s\pi_1, \rho_2 s\pi_2 \in \Pi$ then (by symmetry) it follows that $\rho_2 s\pi_1 \in \Pi$ as well as $\rho_1 s\pi_2 \in \Pi$. ■

Remark 2.2

If R is <u>total</u> (i.e., $\forall s \in S\ \ \exists t \in s\ \ (s,t) \in R$) then PATHS[R] $\neq \emptyset$. If R is not total, then it is possible that PATHS[R] $= \emptyset$. For example, if $R = \{(a,b)\}$ then PATHS[R] $= \emptyset$. Also, if $\Pi \neq \emptyset$, then RELN[Π] is nonempty and total. Since we are interested only in nonempty sets of paths, throughout the remainder of the paper we assume that R is nonempty and total. ■

3. Technical Results

Theorem 3.1

If the set of paths Π is R-generable then

(i) Π is suffix closed,

(ii) Π is fusion closed, and

(iii) Π is Koenig closed.

Proof. By hypothesis,

$\Pi = \text{PATHS}[R] = \{(s_0, s_1, s_2, \ldots) \in S^\omega \colon \forall i \geq 0\ (s_i, s_{i+1}) \in R\}$.

(i): Choose an arbitrary $(s_0, s_1, s_2, \ldots) \in \Pi$. By definition of PATHS[R], $(s_0, s_1) \in R$, $(s_1, s_2) \in R$, $(s_2, s_3) \in R$, ... This trivially implies $(s_1, s_2) \in R$, $(s_2, s_3) \in R$, $(s_3, s_4) \in R$, ... and that $(s_1, s_2, s_3, \ldots) \in \text{PATHS}[R] = \Pi$.

(ii): Choose arbitrary $\rho_1 s \pi_1, \rho_2 s \pi_2 \in \Pi = \text{PATHS}[R]$. Now, we can write $\rho_1 = (s_0, s_1, \ldots, s_{k-1})$ and $s = s_k$. By definition of PATHS[R], $(s_0, s_1) \in R$, $(s_1, s_2) \in R, \ldots (s_{k-1}, s_k) \in R$. By repeated application of (i), $s\pi_2 \in \text{PATHS}[R]$. If we write $\pi_2 = (s_{k+1}, s_{k+2}, s_{k+3}, \ldots)$, it follows that $(s_k, s_{k+1}) \in R$, $(s_{k+1}, s_{k+2}) \in R$, $(s_{k+2}, s_{k+3}) \in R$, ... Hence, $(s_0, s_1, s_2, \ldots) = \rho_1 s \pi_2 \in \text{PATHS}[R]$. Similarly, $\rho_2 s \pi_1 \in \text{PATHS}[R]$.

(iii): Choose arbitrary $\pi_1, \pi_2, \pi_3, \ldots \in \Pi$ such that $\pi_1 = \rho_1 \pi_1'$, $\pi_2 = \rho_1 \rho_2 \pi_2'$, $\pi_3 = \rho_1 \rho_2 \rho_3 \pi_3'$, ... where each $\rho_i \in S^*$ and each $\pi_i' \in S^\omega$. Let $\pi_\omega = \rho_1 \rho_2 \rho_3 \ldots = (s_0, s_1, s_2, \ldots)$. We wish to show that $\pi_\omega \in \Pi = \text{PATHS}[R]$. It suffices to show that each consecutive pair of states $(s_i, s_{i+1}) \in R$. We observe that for each $i > 0$, there exists a $j > 0$ such that π_j is of the form $(\ldots, s_i, s_{i+1}, \ldots)$. Since $\pi_j \in \Pi =$ PATHS[R], it follows that $(s_i, s_{i+1}) \in R$. ■

<u>Lemma 3.2</u>

For all sets of paths Π, $\Pi \subseteq$ PATHS[RELN[Π]].

<u>Proof</u>. Choose an arbitrary $(s_0, s_1, s_2, \ldots) \in \Pi$. It suffices to show that for all $i \geq 0$, $(s_i, s_{i+1}) \in$ RELN[Π]. But this is immediate by definition of RELN[Π]. ■

<u>Theorem 3.3</u>

Suppose the set of paths Π satisfies the following three closure conditions:

(i) suffix closure

(ii) fusion closure

(iii) Koenig closure

Then, there exists a relation R such that $\Pi =$ PATHS[R].

<u>Proof</u>. Let R = RELN[Π]. We wish to show $\Pi =$ PATHS[R] = PATHS[RELN[Π]]. $\Pi \subseteq$ PATHS[RELN[Π]]: This follows immediately from Lemma 3.2.

PATHS[RELN[Π]] $\subseteq \Pi$: Choose an arbitrary $(s_0, s_1, s_2, \ldots) \in$ PATHS[RELN[Π]]. For each $i \geq 0$, by definition of PATHS, $(s_i, s_{i+1}) \in$ RELN[Π]. By definition of RELN, there is a path in Π of the form $\rho_i s_i s_{i+1} \pi_i$ where $\rho_i \in S^*$ and $\pi_i \in S^\omega$. Since Π is suffix closed, $s_i s_{i+1} \pi_i \in \Pi$. In other words, $s_0 s_1 \pi_1 \in \Pi$, $s_1 s_2 \pi_2 \in \Pi$, $s_2 s_3 \pi_3 \in \Pi$, ... We now use the fact that Π is fusion closed. Fuse $s_0 s_1 \pi_1$ and $s_1 s_2 \pi_2$ on s_1 to conclude that $s_0 s_1 s_2 \pi_2 \in \Pi$. Fuse $s_0 s_1 s_2 \pi_2$ and $s_2 s_3 \pi_3$ on s_2 to conclude $s_0 s_1 s_2 s_3 \pi_3 \in \Pi$. In general, we can fuse on s_k to conclude that $s_0 \ldots s_k \pi_k \in \Pi$. This process can be repeated indefinitely and it follows that $s_0 s_1 \pi_1 \in \Pi$, $s_0 s_1 s_2 \pi_2 \in \Pi$, $s_0 s_1 s_2 s_3 \pi_3 \in \Pi$, ... Since Π is Koenig closed, $s_0 s_1 s_2 \ldots = (s_0, s_1, s_2, \ldots) \in \Pi$.

We conclude that Π = PATHS[RELN[Π]] = PATHS[R] as desired. ■

Together, Theorem 3.1 and Theorem 3.3 establish a key result:

<u>Theorem 3.4</u>

A set of paths Π is R-generable iff it is

(i) suffix closed,

(ii) fusion closed, and

(iii) Koenig closed.

The Propositions in the remainder of this Section establish that each region of the Venn Diagram in Figure 1.1 is nonempty. The letters a, b, ..., g denote distinct states. For each region, we exhibit a path set Π that belongs in the region. To simplify the notation, we use extended regular expressions to represent sets of paths; e.g., a^+b^ω denotes the set of paths {(a,b,b,b,...), (a,a,b,b,b,...), (a,a,a,b,b,b,...), ...}, and a^*bc^ω denotes the set of paths {(b,c,c,c,...), (a,b,c,c,c,...), (a,a,b,c,c,c,...), ...}.

Finally, recall that F denotes the class of fusion closed path sets, K denotes the class of Koenig closed path sets, and S denotes the class of suffix closed path sets.

<u>Proposition 3.5</u>

$\bar{S} \cap \bar{F} \cap \bar{K} \neq \emptyset$.

<u>Proof</u>. Let $\Pi = \{a^+b^\omega, cde^\omega, fdg^\omega\}$.

Π is not suffix closed: While $ab^\omega \in \Pi$ by virtue of a^+b^ω, the suffix $b^\omega \notin \Pi$.

Π is not fusion closed: While $cde^\omega \in \Pi$ and $fdg^\omega \in \Pi$, $cdg^\omega \notin \Pi$.

Π is not Koenig closed: Each of ab^ω, a^2b^ω, a^3b^ω, ... $\in \Pi$ by virtue of a^+b^ω. Yet, the "limit" $a^\omega \notin \Pi$. ∎

<u>Proposition 3.6</u>

$\bar{S} \cap \bar{F} \cap K \neq \emptyset$.

<u>Proof</u>. Let $\Pi = \{abc^\omega, dbe^\omega\}$.

Π is not suffix closed: While $abc^\omega \in \Pi$, the suffix $bc^\omega \notin \Pi$.

Π is not fusion closed: While $abc^\omega \in \Pi$ and $dbe^\omega \in \Pi$, $abe^\omega \notin \Pi$.

Π is Koenig closed: Assume that for each i, $\rho_1 \ldots \rho_i \pi_i \in \Pi$. It must be each $\rho_1 \ldots \rho_i \pi_i$ is the same string, either abc^ω or dbe^ω (because abc^ω and dbe^ω have no nonempty common prefix). That $\rho_1 \rho_2 \rho_3 \ldots \in \Pi$ follows immediately. ∎

<u>Proposition 3.7</u>

$\bar{S} \cap F \cap \bar{K} \neq \emptyset$.

<u>Proof</u>. Let $\Pi = \{a^+b^\omega\}$.

Π is not suffix closed: While $ab^\omega \in \Pi$ by virtue of a^+b^ω, the suffix $b^\omega \notin \Pi$.

Π is fusion closed: Fusion must be performed on either a or b. First, suppose we fuse on a. Let $\pi_1 = a^{k_1}aa^{\ell_1}b^\omega$ and $\pi_2 = a^{k_2}aa^{\ell_2}b^\omega$. Then the fusion $a^{k_1}aa^{\ell_2}b^\omega \in \Pi$ by virtue of a^+b^ω. Now suppose we fuse on b. Let $\pi_1 = a^{k_1}b^{\ell_1}bb^\omega$ and $\pi_2 = a^{k_2}b^{\ell_2}bb^\omega$. Then the fusion $a^{k_1}b^{\ell_1}b^\omega \in \Pi$ trivially.

Π is not Koenig closed: While each of ab^ω, a^2b^ω, a^3b^ω, ... $\in \Pi$ by virtue of a^+b^ω, $a^\omega \notin \Pi$. ∎

<u>Proposition 3.8</u>

$\bar{S} \cap F \cap K \neq \emptyset$.

<u>Proof</u>. Let $\Pi = \{ab^\omega\}$.

Π is not suffix closed: While $ab^\omega \in \Pi$, the suffix $b^\omega \notin \Pi$.

Π is fusion product closed: Since Π contains only one path, this follows immediately.

Π is Koenig closed: Since Π contains only one path, this follows immediately. ∎

<u>Proposition 3.9</u>

$S \cap \bar{F} \cap \bar{K} \neq \emptyset$.

<u>Proof</u>. Let $\Pi = \{abc^\omega, bc^\omega, c^\omega, dbe^\omega, be^\omega, e^\omega, f^*g^\omega\}$.

Π is suffix closed: This may be verified by inspection.

Π is not fusion closed: While $abc^{\omega} \in \Pi$ and $dbe^{\omega} \in \Pi$, $abe^{\omega} \notin \Pi$.

Π is not Koenig closed: While $fg^{\omega}, f^2g^{\omega}, f^3g^{\omega}, \ldots \in \Pi$ by virtue of f^*g^{ω}, $f^{\omega} \notin \Pi$. ■

Proposition 3.10

$S \cap \bar{F} \cap K \neq \emptyset$.

Proof. Let $\Pi = \{abc^{\omega}, bc^{\omega}, c^{\omega}, dbe^{\omega}, be^{\omega}, e^{\omega}\}$.

Π is suffix closed: This may be verified by inspection.

Π is not fusion closed: While $abc^{\omega} \in \Pi$ and $dbe^{\omega} \in \Pi$, $abe^{\omega} \notin \Pi$.

Π is Koenig closed: By inspection, if $\pi_1 \in \Pi$, $\pi_2 \in \Pi$, and $\pi_1 \neq \pi_2$ then π_1 and π_2 do not have a common nonempty prefix. Hence, if $\rho_1\rho_2\ldots\rho_i\pi_i' \in \Pi$ for all i, it must be that all $\rho_1\rho_2\ldots\rho_i\pi_i' = \pi_0$ for some $\pi_0 \in \Pi$. It follows that $\rho_1\rho_2\rho_3\ldots = \pi_0 \in \Pi$. ■

Proposition 3.11

$S \cap F \cap \bar{K} \neq \emptyset$.

Proof. Let $\Pi = \{a^*b^{\omega}\}$.

Π is suffix closed: If $\pi \in \Pi$ then π is of the form a^ib^{ω}. If $i = 0$ then $\pi = b^{\omega}$ and the suffix of π, b^{ω}, $\in \Pi$. If $i > 0$, then the suffix of π, $a^{i-1}b^{\omega}$, $\in \Pi$ by definition of a^*b^{ω}.

Π is fusion closed: We must fuse on either a or b.

If $\pi_1 = a^{k_1}aa^{\ell_1}b^{\omega}$ and $\pi_2 = a^{k_2}aa^{\ell_2}b^{\omega}$, then the fusion $a^{k_1}aa^{\ell_2}b^{\omega} \in \Pi$ as desired. If $\pi_1 = a^{k_1}b^{\ell_1}bb^{\omega}$ and $\pi_2 = a^{k_2}b^{\ell_2}bb^{\omega}$, then the fusion $a^{k_1}b^{\ell_1}bb^{\omega} \in \Pi$ immediately.

Π is not Koenig closed: While each of ab^{ω}, a^2b^{ω}, a^3b^{ω}, ... $\in \Pi$, the limit $a^{\omega} \notin \Pi$. ∎

Proposition 3.12

$S \cap F \cap K \neq \emptyset$.

Proof. Let $\Pi = \{a^{\omega}\}$. It follows immediately that Π is suffix closed, fusion closed, and Koenig closed. ∎

4. Discussion

Suffix closure is the only restriction placed on a set of paths by Lamport [6] with the intention that "future behavior depends only upon the current state, and not upon how that state was reached." However, the formal notion of suffix closure is not quite strong enough to guarantee that Lamport's informally stated requirement about future behavior is satisfied. To see this, consider the path set $\Pi = \{abc^{\omega}, bc^{\omega}, c^{\omega}, ebd^{\omega}, bd^{\omega}, d^{\omega}\} \in S \cap \bar{F} \cap K$. In state b, the next state is

(i) c if the previous state was a,

(ii) d if the previous state was b, and

(iii) either c or d if b is the initial state of the path.

Thus, both fusion closure and suffix closure are needed to meet Lamport's informal requirement. Fusion closure derives from the notion of the fusion product of two paths described in [8] and [5]. (The fusion product of $(s_0, \ldots, s_k)$ and $(s_k, s_{k+1}, \ldots)$ is $(s_0, s_1, s_2, \ldots)$.) Note that in [8] and [5] which are "exogeneous" logics, there are no constraints on the sets of paths that determine the semantics of atomic programs; fusion product is merely a device used in defining the semantics of composite programs. The concept of Koenig closure appears in [1] (where the terminology used is "a closed process"). The notion of R-generable occurs in a number of logics. See, e.g., [7], [2] and [3].

Which constraint or combination of constraints yields the most desirable underlying semantics for a temporal logic? The answer, of course, depends upon the intended application of the logic. Note that there are applications in which it would make sense to violate certain constraints. For example, if the states referenced by the logic contain information about the data values stored in a process but no control information about a process, then the future behavior of a program may very well depend on its past behavior and not just its current state. In this case, it would make sense to allow an underlying semantics in which a set of paths Π (which would really be a sequence of data values) violated the suffix closure and/or fusion closure constraints. It might also make sense to allow the Koenig closure constraint to be violated when discussing programs executing under fair scheduling: Let b denote a state which results from execution of process 1 and c denote a state which results from execution of process 2. Then each path represented by $(b^+c^+)^\omega$ is a fair path along which both processes execute infinitely often. If $\{(b^+c^+)^\omega\} \subseteq \Pi$ and Π is Koenig closed, then the unfair path $b^\omega \in \Pi$.

Nonethless, for most applications it seems desirable to require that Π be R-generable. One might think that the fewer the restrictions on Π built into the underlying semantics of the logic, the better, because the logic would be more general. But the resulting generality may be of little use in practice because it allows programs with "pathological" behavior that correspond to no real world model of concurrency. In contrast, the requirement that Π be R-generable corresponds naturally to execution under the weakest scheduling criterion which guarantees that some process make some progress: pure nondeterministic scheduling. The next state relation R is defined in terms of arbitrarily choosing an enabled process and executing one step of that process. If we wish to talk about processes executing under fair scheduling so that the set of paths corresponding to fairly scheduled executions is not Koenig closed, we should specify this wish through a specification formula of the logic and not in the underlying semantics. This allows substantially greater flexibility than building in the fairness requirement into the underlying semantics because we can still talk about processes executing under pure nondeterminism if desired. In general, if we wish to restrict our attention to a subset $\Pi' \subseteq \Pi$ that is not R-generable, then the members of Π' should be specifiable by a formula of the logic.

REFERENCES

[1] Abrahamson, K., Expressiveness and Decidability of Logics of Processes. Ph.D. Thesis, Univ. of Washington, Seattle, 1980.

[2] Ben-Ari, M., Manna, Z., and Pneuli, A., The Temporal Logic of Branching Time. 8th Annual ACM Symp. on Principles of Programming Languages, 1981.

[3] Emerson, E. A. and Clarke, E.M., Characterizing Correctness Properties of Parallel Programs as Fixpoints. Proc. 7th Int. Colloquium on Automata, Languages, and Programming. Lecture Notes in Computer Science #85, Springer-Verlag, 1980.

[4] Hughes, G. and Cresswell, M., An Introduction to Modal Logic. Methuen, London, 1968.

[5] Harel, D., Kozen, D. and Parikh, R., Process Logic: Expressiveness, Decidability, Completeness. 21st Annual Symposium of Foundations of Computer Science, 1980.

[6] Lamport, L., "Sometime" Is Sometimes "Not Never." 7th Annual ACM Symp. on Principles of Programming Languages, 1980.

[7] Manna, Z. and Pnueli, A., The Modal Logic of Programs. Proc. 6th Int. Colloquium on Automata, Languages, and Programming, Lecture Notes in Computer Science #71, Springer-Verlag, 1979.

[8] Pratt, V.R., Process Logic. Proc. 6th Annual ACM Symposium on Principles of Programming Languages, 1979.