Yet Another Process Logic

(Preliminary Version)

Moshe Y. Vardi
Stanford University

Pierre Wolper
Bell Laboratories

ABSTRACT

We present a process logic that differs from the one introduced by Harel, Kozen and Parikh in several ways. First, we use the extended temporal logic of Wolper for statements about paths. Second, we allow a "repeat" operator in the programs. This allows us to specify programs with infinite computations. However, we limit the interaction between programs and path statements by adopting semantics similar to the ones used by Nishimura. Also, we require atomic programs to be interpreted as binary relations. We argue that this gives us a more appropriate logic. We have obtained an elementary decision procedure for our logic. The time complexity of the decision procedure is four exponentials in the general case and two exponentials if the logic is restricted to finite paths.

1. Introduction

While dynamic logic [Pr76] has proven to be a very useful tool to reason about the input/output behavior of programs, it has become clear that it is not adequate for reasoning about the ongoing behavior of programs. In view of this, Pratt [Pr78] introduced a process logic, that extended dynamic logic with the connectives "during" and "throughout". Parikh [Pa78] chose to extend dynamic logic with quantification over computation paths. His logic, SOAPL, is strictly more expressive than Pratt's [Ha79].

At the same time, a different approach was taken by Pnueli, who developed a temporal logic, called TL [Pn77]. TL is oriented towards reasoning about the ongoing behavior of programs, but does not allow programs to be mentioned explicitly. In dynamic logic, on the other hand, the programs are an essential part of the formulas.
Nishimura [Ni80] suggested combining the two approaches. The essence of his logic is that computation paths are specified by referring to programs explicitly, as in dynamic logic, and temporal logic is used to specify temporal properties of these computation paths. He showed that his logic, while its syntax is much cleaner than that of SOAPL, is at least as expressive as the latter. This approach was continued by Harel et al. [HKP80]. They extended Nishimura's logic by removing his distinction between state formulas and path formulas. Moreover, their logic, called \( PL \), is defined in such a way that it is a direct extension of dynamic logic.

We contend that \( PL \) is not an adequate logic of processes, since it is at the same time too powerful and not powerful enough. Let us first see why \( PL \) is not powerful enough.

\( PL \) uses Pnueli's TL for its temporal part. TL, however, is equivalent [GPSS80] to the first-order theory of \( (N, \prec) \), the natural numbers with the less-than relation, and consequently cannot specify arbitrary regular properties. Thus, from that aspect, the temporal part of \( PL \) is weaker than its dynamic part (see also [Wo81,HP82]). Another weakness of \( PL \) is its limited ability to deal with non-terminating processes, e.g., operating systems. Such processes often run by repeatedly executing the same program. \( PL \), however, cannot specify the infinite repetition of programs, while reasoning about non-terminating processes was a primary motivation for introducing process logics.

Let us now see in what aspects \( PL \) is too powerful. The interpretation of an atomic program in \( PL \) is an arbitrary set of paths. But in practice the interpretation of an atomic program is never an arbitrary set of paths but rather a binary relation, i.e., a set of paths of length two, consisting of the initial state and the final state. Even if one wants to consider a higher-level program as atomic, the interpretation of such a program should not be an arbitrary set of paths.

Finally, we believe that the distinction between state formulas and path formulas is inherent to our thinking about processes. A computation path is characterized by the properties of its states, and a state is characterized by the properties of the paths that start from it. The results of removing this distinction are not very intuitive. Consider the \( PL \) formula \([\alpha]s\text{ome}\ P\), where \( \alpha \) is a program and \( P \) is an atomic proposition. While we want it to mean that all computations of \( \alpha \) eventually satisfy \( P \), it actually is true of all paths that either eventually satisfy \( P \) or can be extended by a computation of \( \alpha \) that eventually satisfies \( P \). The artificiality of the latter statement is self-evident. This comes as a result of the desire to have \( PL \) extend dynamic logic in a direct way. In our opinion, any attempt to have a logic for ongoing behavior that directly extends a logic