Temporal Logic: The Lesser of Three Evils

Leslie Lamport
Microsoft Research

5 April 2010

> The evil that men do lives after them.
> *Julius Caesar*, by William Shakespeare

In the Beginning

Amir Pnueli introduced temporal logic to computer science in 1977 in a paper presented at FOCS [10]. That paper inspired Susan Owicki to organize an informal seminar on the subject at Stanford during the 1977–78 academic year. Temporal logic sounded to me like yet another of the useless formalisms that computer scientists seemed fond of, but I decided to attend anyway. That was one of the best decisions I ever made.

Susan (together with David Gries) and I had independently developed what is now known as the Owicki-Gries method for proving invariance properties of concurrent programs [4, 8]. I had also devised a method for proving liveness properties of the form $P \leadsto Q$, read P *leads to* Q, which asserts that if P is true then Q will eventually become true. Written informally, my proofs were reasonable. However, their formalization was ugly and complicated.

Susan and I soon realized that Amir's temporal logic was ideal for formalizing liveness proofs. The logic was simple, based on the single operator $\Box$, read *always* or *henceforth*, where $\Box P$ asserts that P is true from now on. Its dual $\Diamond P$, defined to equal $\neg\Diamond\neg P$, asserts that P is eventually true. My $\leadsto$ operator could be defined by

$$P \leadsto Q \triangleq \Box(P \Rightarrow \Diamond Q)$$

Temporal logic added to my liveness proofs the ability to directly use invariance properties. The invariance of a formula I means that $\Box I$ is true. We

could use this fact to prove liveness properties by applying the proof rule

$$\frac{(I \wedge P) \leadsto Q}{\Box I \Rightarrow (P \leadsto Q)}$$

This rule was the key to the elegant formalization of liveness proofs. It was enshrined in the boxes of the proof lattices Susan and I introduced [9].

Inadequate and Evil

In the late 1970s and early 1980s, I and many of my colleagues started going beyond the realm of proving that programs satisfied particular properties to trying to write and verify complete specifications. Temporal logic seemed to be wonderful for the task of specifying a system. A specification would simply be the conjunction of temporal-logic formulas that asserted properties the system must satisfy. I believe that the first publication advocating this approach was by Richard Schwartz and Michael Melliar-Smith [11].

By the time of that paper's publication, I had realized that temporal logic was not all that wonderful. In fact, I was originally an author but had my name removed because I had become disillusioned with the method. Watching my Richard and Michael spend days trying to specify a FIFO queue (a very trivial example) convinced me that the method would never work on any real example.

Others also realized that there was a problem. Most thought that the source of the problem lay in the simplicity of Amir's temporal logic. So, they developed a multitude of new, more complicated logics based on more expressive and more complicated temporal operators. I was not immune to that temptation [5]. However, I eventually realized that the fundamental problem lay in trying to specify something by a list of properties.

Years of experience have taught me that human beings cannot understand the consequences of a conjunction of separate properties. As one of many pieces of evidence, consider multiprocessor memory models. Engineers have often specified them by a list of properties—for example, in the specifications of the DEC/Compaq Alpha [1] and the Intel Itanium [3] memories. Even the people who wrote the specifications did not understand them. Jim Saxe discovered that the published Alpha memory specification permitted causal loops, in which a write stores a completely arbitrary value, and that value is justified by a later read. Using a formal specification that we wrote, my colleagues and I discovered errors in the (very simple) examples in an early version of the Itanium memory specification.

This experience revealed the inadequacy of temporal logic for writing specifications. However, inadequacy is not evil. I discovered that temporal logic is evil in the late 1980s when it led my colleague Martín Abadi and me to believe a false result for several days. Those who know me will not be surprised that I made such an error, but those who know Martín will realize that, if he could be confused by temporal logic, then anyone can be.

Temporal logic is evil because it does not satisfy the deduction principle. In ordinary mathematics, we prove the implication $P \Rightarrow Q$ (P implies Q) by assuming P is true and proving Q is true. This reasoning is expressed by the following proof rule, which is called the deduction principle.

$$\frac{\dfrac{P}{Q}}{P \Rightarrow Q}$$

The deduction principle is not valid for temporal logic and other modal logics. For example, a basic axiom of temporal logic is $\frac{P}{\Box P}$, which asserts that, if P is true, then it is always true. The deduction principle would allow us to deduce from this the truth of $P \Rightarrow \Box P$, a formula asserting that, if P is true initially, then it is always true—which is not true in general.

The Greater Evils

The source of temporal logic's evil is that its formulas have an implicit variable representing time. The truth of a temporal formula asserts that the formula is true for all values of this variable. Calling the variable t, a proof rule $\frac{P}{Q}$ asserts that $\forall\, t : P$ implies $\forall\, t : Q$, while the truth of $P \Rightarrow Q$ means $\forall\, t : (P \Rightarrow Q)$. The deduction principle is invalid for temporal logic because we cannot deduce $\forall\, t : (P \Rightarrow Q)$ from $(\forall\, t : P) \Rightarrow (\forall\, t : Q)$.

One way to eliminate this problem is to make the time variable t explicit. Every atomic formula becomes an explicit function of t, so $P \leadsto Q$ is written $\forall\, t : (P(t) \Rightarrow \exists\, s \geq t : Q(s))$. This is exactly what Nissim Francez did in his thesis [2]. The messiness of representing even so simple a formula as $P \leadsto Q$ indicates why this is a bad idea. In fact, Nissim was Amir's student, and I believe it was his thesis that inspired Amir to use temporal logic. Temporal logic is a lesser evil than the complexity introduced by an explicit time variable.

Another way people have tried to avoid the evil of temporal logic is to use some form of program logic in its place. Logic is a branch of mathematics, and one of the most basic operations of mathematics is substituting an

expression for a variable. Substitution is fundamental to computing because it lies at the heart of refinement, which is also called implementation. In a volume commemorating a happier occasion—the retirement of Willem-Paul de Roever—I illustrated refinement as substitution by showing how to derive an important hardware protocol from a simple specification. The main step essentially consisted of substituting $(p + c) \bmod 2$ for the variable x in the assignment statement $x := x + 1$ [7].

Although evil, temporal logic is still mathematics. One can therefore derive a temporal-logic description of the protocol from its temporal-logic specification by substituting $(p + c) \bmod 2$ for x. However, literally substituting for x in the statement $x := x + 1$ makes no sense. One cannot substitute an expression for a variable in a program logic with assignment statements. Indeed, I know of no program logic in which such substitution is possible. A "logic" that does not permit substitution is a greater evil than temporal logic.

A Necessary and Useful Evil

Although evil, temporal logic is necessary. It is the best way we know to reason about liveness. Moreover, its ability to describe reactive systems, even if only in principle, helps us to understand them. The traditional first step in creating a science is to introduce mathematics. Temporal logic is the natural mathematics of reactive systems.

We cannot remove the evil from temporal logic, but we can overcome its inadequacy for writing specifications. This doesn't require new temporal operators; the $\Box$ operator that Amir introduced in 1977 is (approximately) enough. The trick is to extend the base formulas from state predicates to *actions*, which are predicates on pairs of states [6]. The result is a logic that confines the evil of temporal logic mainly to the domain for which it is both necessary and useful: liveness.

I began working on verification because I wanted to ensure that the algorithms I developed were correct. I have always sought formal methods that would help me do that. Today, temporal logic is a tool that I use every day in my work on distributed and concurrent algorithms. I am continually grateful that this evil that Amir did lives after him.

References

[1] Alpha Architecture Committee. *Alpha Architecture Reference Manual*. Digital Press, Boston, third edition, 1998.

[2] Nissm Francez. *The Specification and Verification of Cyclic (Sequential and Concurrent) Programs.* PhD thesis, Weizmann Institute of Science, Rehovot, Israel, June 1976.

[3] Intel. A formal specification of intel itanium processor family memory ordering. Application Note. http://download.intel.com/design/Itanium/Downloads/25142901.pdf, October 2002.

[4] Leslie Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, SE-3(2):125–143, March 1977.

[5] Leslie Lamport. Timesets—a new method for temporal reasoning about programs. In Dexter Kozen, editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 177–196, Berlin, Heidelberg, New York, May 1981. Springer-Verlag.

[6] Leslie Lamport. The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, May 1994.

[7] Leslie Lamport. Computer science and state machines. In Dennis Dams, Ulrich Hannemann, and Martin Steffen, editors, *Concurrency, Compositionality, and Correctness (Essays in Honor of Willem-Paul de Roever)*, volume 5930 of *Lecture Notes in Computer Science*, pages 60–65. Springer, 2010.

[8] Susan Owicki and David Gries. Verifying properties of parallel programs: An axiomatic approach. *Communications of the ACM*, 19(5):279–284, May 1976.

[9] Susan Owicki and Leslie Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems*, 4(3):455–495, July 1982.

[10] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on the Foundations of Computer Science*, pages 46–57. IEEE, November 1977.

[11] Richard L. Schwartz and P. M. Melliar-Smith. Temporal logic specification of distributed systems. In *Proceedings of the 2nd International Conference on Distributed Computing Systems*, pages 446–454. IEEE Computer Society Press, April 1981.