I am told that, in the world of *objets d'art*, anything over twenty-five years old is considered an antique. It seems appropriate to celebrate Paul's entry into antiquity with a suitably aged *objet*. More *objet trouvé* than *objet d'art*, this is a litte note I wrote not quite twenty years ago—just a year or two before I met Paul. That was also five years before Paul started to drive. Had I known he was going to embark on that radical change of lifestyle, I would have sent it to him then, perhaps with the title reversed. I hope it can still be of some service.

This note was written when I was at Massachusetts Computer Associates. It was greeted with outrage at the time, so I did not distribute it further. But, Massachusetts Computer Associates was not as long-lived as Paul, and it passed away a couple of years ago. So the time has come to brush the dust off the manuscript and expose it to the subdued light of a fall day in Amsterdam.

Leslie Lamport
Palo Alto
September, 1996

How to Tell a Program from an Automobile

Leslie Lamport

28 January 1977

Computer programs and automobiles are both important in modern life, and people should learn to distinguish one from the other. Let us consider how "the man in the street" might try to find out the difference between them. This is not as easy as it sounds. We might think that he need only be told that automobiles are made mostly of metal parts, whereas programs are made of things like subroutines and assignment statements. Unfortunately, if he has never seen a subroutine or an assignment statement, he might think that they are just some kinds of metal parts. Being told what programs and automobiles are used for might not help either. What something is used for is an expression of intention, and intentions are often hard to discover by looking at what something does. (Seeing an automobile in a traffic jam, it might be hard to guess that it is used to move people from one place to another.)

Our man in the street could try asking an auto mechanic to tell him about automobiles, then asking a programmer to tell him about programs, and comparing the results. The mechanic might reply as follows. "An automobile is constructed by people. After they have finished constructing it, it *runs*. However, it may have some *bugs*. [Most mechanics would use the term "problems", but everyone understands this one when he speaks of 'bugs'.] These are usually minor [such as non-functioning windshield wipers], but occasionally they are serious enough to make the automobile stop running. The bugs are found by *testing*, and are fixed. The automobile then runs properly, and is sold to the user. The user then operates the automobile. However, as he uses it, new bugs may appear. Therefore, the automobile must be *maintained*. Maintenance is also required if the user wants to modify the automobile—for example, to improve its performance [perhaps by 'souping up' the engine]. When the automobile becomes to difficult to maintain, the user has to buy a new one."

Unfortunately, the programmer might say almost exactly the same things about programs! Our poor man in the street would be no further towards his goal. The reader has undoubtedly guessed by now that I am not interested in explaining the difference between a program and an automobile to the man in the street. He was just a pedagogical device, and will now be returned to his street. My purpose is to explain the difference between a program and an automobile to programmers.

An automobile runs, a program does not. (Computers run, but I'm not discussing them.) An automobile requires maintenance, a program does not. A program does not need to have its stack cleaned every 10,000 miles. Its *if* statements do not wear out through use. (Previously undetected *errors* may need to be corrected, or it might be necessary to write a new but similar program, but those are different matters.) An automobile is a piece of machinery, a program is some kind of mathematical expression.

Programmers may be disheartened to learn that their programs are not like automobiles, but are mathematical expressions. Automobiles can be loveable—one can relate to them almost as if they were alive. Mathematical expressions are not loveable—they are hard to relate to. Many programmers may feel like clinging to their belief that programs are like automobiles. However, further thought reveals that mathematical expressions do have certain advantages over automobiles.

Unlike an automobile, a mathematical expression can have a meaning. We can therefore ask whether it has the *correct* meaning. One cannot talk about the correctness of an automobile—it may run properly, but it makes no sense to say that it is correct. Since a program is a mathematical expression, one can (at least in principle) decide if it is correct. The user and the programmer can agree in advance what the meaning of the program should be, and the programmer can prove mathematically that his program has that meaning. An automobile salesman can never prove that he sold a properly running automobile, he can only give enough evidence to satisfy a jury.

I have tried briefly to indicate the difference between an automobile and a program. The programmer should now apply his new knowledge in the field, and see if he can tell the difference by himself. He should first examine his automobile, and ask whether it is running properly, or if it has some bugs and requires maintenance. If he cannot answer this question, he may have to ask an auto mechanic to help him. He should then examine a program, and ask what its meaning should be, and whether he can prove that it has the correct meaning. If he can't answer these questions, it is probably because the program was written by someone who thought he was building an automobile. In that case, I suggest that the programmer ask these questions about the next program he writes—while he is writing it! In this way, any programmer can learn to tell a program from an automobile.