What is Automated Reasoning?

Michael Beeson

July 31, 2006

Introduction

Automated reasoning is the art and science of getting computers to apply logical reasoning to solve problems, for example, to prove theorems, solve puzzles, design circuits, verify or synthesize computer programs. The purpose of this article is to introduce this art and science to the layperson or student.[1] The phrase "automated deduction" is also used; if there is a difference between "automated reasoning" and "automated deduction", it is that automated deduction might be construed more narrowly, emphasizing deduction in mathematics, while "automated reasoning" might include puzzle-solving or reasoning about electronic circuits, or legal reasoning. These are not so different, since in all cases the task boils down to proving a desired theorem from some axioms. To make the subject stand out clearly, I will begin by describing some neighboring subjects that it is not.

Since reasoning usually requires intelligence (when humans do it), this subject can be thought of as part of artificial intelligence (AI), but it has many specialized methods and (so far at least) does not use many of the techniques of AI, so it not usually thought of as a branch of AI. In any case, there is no restriction that in automated reasoning a program should "work like a human would". Usually, programs try to take advantage of the computer's superior ability to search through thousands or millions of possibilities, and hope in that way to make up for the computer's lack of "insight", whatever that is. This is usually the case in AI research, too, but the phrase "artificial intelligence" still has connotations of computers "working like the human brain" for much of the general public, and that is emphatically not the case in automated reasoning.

Automated reasoning can also be thought of as a branch of the enterprise "computerizing mathematics". That is a larger enterprise, in that there is more to mathematics than proving theorems. For example, making complicated calculations is mathematics, but it is not the same activity as proving theorems, although a proof may include a calculation. In fact, a proof can consist *entirely* of a calculation, but that is usually not a very interesting or beautiful proof, though it can be if there is something unexpected about the calculation. Normally though, a calculation proceeds "mechanically" according to a certain method that is usually used for that kind of problem, and it isn't really very surprising that such things can be done by computer. We distinguish *numerical* calculations and *symbolic* calculations. The latter include algebra and calculus, as well as some more specialized kinds of calculations, for example in number theory. Today, in 2006, there are several good general-purpose programs for symbolic and numerical calculations, and many special-purpose ones as well. The subject of symbolic computation has its own journals and conferences. However, this is not automated deduction.

To understand automated reasoning, you must understand that proving theorems is not the same as calculating, although there are intimate relationships between the two. When you are proving a

[1] A more technical introduction, for the non-specialist scientist, can be found in [1].

theorem, you are connecting various claims or "assertions" by chains of logical reasoning. You may believe that your desired theorem does indeed follow from the axioms you are assuming (possibly by using some previously proved theorems as well); but it will generally require some insight, inspiration, or at least experience, to find the right chains of reasoning.

Once a proof is found, it can usually be made more detailed. This is almost like looking at a physical specimen under the microscope, except that when the power is turned up, you don't see the proof at a greater level of detail–instead, you have to *supply* or *construct* that level of detail. Eventually, you reach a level of *complete detail*. This is called a *formal* proof. Such a proof is actually an *object*–you can write it down (including the justifications of the steps) according to fixed rules, and it can be manipulated by a computer programmed to work with those rules. One interesting thing you can have such a computer program do is *check* or *verify* the proof. That is, you have the computer check that the proof really *is* a proof, that it follows all the rules to the letter without any mistakes or omissions. This subject is called *proof-checking*–and it also is *not* automated deduction. Automated deduction is concerned with the *finding* of proofs by computer, not with checking proofs that have been found already.

So far we only touched on what the proofs are *about*. There are people who try to give proofs showing that certain computer programs do what they are supposed to do, or don't do things they are not supposed to do. This is called "program verification". If the computer is involved in producing such proofs, it's called computerized program verification. Usually it's considered too difficult to verify such proofs about ordinary computer programs, so usually these people develop programming systems in which a person constructs at the same time, a computer program and a proof that the computer program functions as intended. It takes a lot more effort to do this than just to write the program, but if lives or lots of money will depend on the program working correctly, it may be worth it. (Lives could depend on programs used in air traffic control, missile guidance, medical machinery, for example.) Some computer programs are put in hardware on chips in cars or factory machinery and many thousands of copies are sold, so lots of money can depend on the correctness of such things even if lives do not, or if lots of money doesn't depend on one single copy running correctly. For example, Intel has a group trying to use computerized program verification to ensure that they don't repeat their mistake of a few years ago–namely selling computers that can make incorrect numerical computations. But program verification, computerized or not, is *not* automated deduction. The reason is that the proofs are produced by humans, not by computers.

The setting for automated deduction involves some *assumptions* (mathematical formulas or statements) and a *goal*, a would-be theorem. The reason we use the word *goal* rather than *conjecture* is that the word "conjecture" is used when we don't know whether it is or is not a theorem. But often, in automated deduction, the goal is known to be a theorem (because humans have proved it), but we want to see if our computer program can find a proof, or perhaps a better proof than one we already have.

It is the human who decides on the particular goal. True, there have been a few research projects about getting a computer program to generate conjectures, but that is a separate enterprise, as the program is only asked to *generate* conjectures, not prove them.

It is also the human who decides on the particular axioms to be used. That is very important, because in the present state of the art, you can't just tell the program "Use any known mathematics that you think is relevant." That is much too complicated. Present-day programs can use only a single, simple set of axioms at a time. This is a serious limitation in the subject, since it means that big and important parts of mathematics (such as number theory) are not within reach. Number theory notoriously uses every other branch of mathematics.

Now, we have finally arrived at the point where we can say what automated deduction *is*, as opposed to what it *is not*:

Given a simple set of assumptions and a goal, automated deduction programs search for a proof of that goal from those assumptions.

This is like saying that a chess program searches for the best move given a certain configuration of pieces on the chessboard. The devil is in the details: *How* does the program search? You must understand that, in technical jargon, "the search space is exponentially large." That means that a complete, systematic search is impossible. You can't just start enumerating all the consequences of the assumptions until you reach the goal–you will usually wait forever. (That would be "forward search".) You also can't just reason backwards from the goal, looking for things that would imply it, and then things that would imply those things, and so on. (That would be "backwards search.") There are too many of those things, and you will be swamped and get nowhere. Similarly, a combination of forwards and backwards search, hoping to meet in the middle, also doesn't work.

An Example

Here is a puzzle (taken from [2], page 1.)

> There are twelve billiard balls, eleven of which are identical in weight. The remaining ball—the odd one— has a different weight. You are not told whether it is heavier or lighter. You have a balance scale for weighing the balls. Can you find which ball is the odd ball in three weighings, and can you also find out whether it is lighter or heavier than the others?

When this problem is suitably expressed in a logical language, an automated reasoning program can solve it. For other examples, see the other links on the Otter-lambda website (which was probably how you got here).

References

[1] Beeson, M., The mechanization of mathematics, in Teuscher, C. (ed.) Alan Turing: Life and Legacy of a Great Thinker, pp. 77-134. Springer-Verlag, Berlin Heidelberg New York, 2003.

[2] Wos, L. *A Fascinating Country in the World of Computing*, World Scientific, Singapore (1999).