The proof proceeds by reductio ad absurdum. We will assume that there is an algorithm halts(a*,* i) that decides if the algorithm encoded by the string a will halt when given as input the string i, and then show that this leads to a contradiction.

We start with assuming that there is an algorithm halts(a*,* i) that returns the boolean true if the algorithm represented by the string a halts when given as input the string i, and returns the boolean false otherwise. Given this algorithm we can construct another algorithm trouble(s) as follows:

**public boolean** trouble (**String** s) {

**if** (!halts(s, s)) {

**return** **true**;

} **else {**

**while** (**true**) { // loop forever

**int** x = 0; // anything would work here

}

}

**}**

This algorithm takes a string s as its argument and runs the algorithm halts, giving it s both as the description of the algorithm to check and as the initial data to feed to that algorithm. If halts outputs false, then trouble outputs true, otherwise trouble goes into an infinite loop. Since all Java program segments are sequences of characters, they can be represented by strings. So the following string T represents the algorithm trouble.

String T = “**public String** trouble(**String** s){**if**(!halts(s, s)){**return**” +

“**true**;}**else{while**(**true**){**int** x=0;}}}”;

We can now ask the following question: Does trouble(T) halt?

Let us consider both possible cases:

1. Assume that trouble(T) halts. The only way this can happen is that halts(T*,* T) returned false, but that in turn indicates that trouble(T) does not halt. *Contradiction*.
2. Assume that trouble(T) does not halt. Since halts always halts, this can only happen when trouble goes into its infinite loop. This means that halts(T*,* T) outputs true, but that in turn would mean that trouble(T) does halt. *Contradiction*.

Since both cases lead to a contradiction, the initial assumption that the algorithm Halt exists must be false.

SOURCE: Adapted from *Wikipedia* -- http://en.wikipedia.org/wiki/Halting\_problem