

A problem that can not be solved on a computer?

CS 317

August 26, 2008

1. A C program that outputs itself. ¹

```
char*s="char*s=%c%c%c;main(){printf(s,34,s,34);}";main(){printf(s,34,s,34);}
```

2. Question: Is it possible to implement the following function in C?

```
int halt(char *program) {if program halts return 1; else return 0;}
```

In other words, can we write a function that determines if a program will ever terminate in a finite amount of time? In this case, we assume the program is described by a string containing its C source code. ² How might you construct such a function?

3. Let's assume we can write `halt()` and thus we can use it in the following program. ³

```
char*s="char*s=%c%c%c;main(){char p[139];sprintf(p,s,34,s,34);while(halt(p));}";  
main(){char p[139];sprintf(p,s,34,s,34);while(halt(p));}
```

Hopefully you can see that the argument `p` passed to the `halt()` function references a string containing the source code for the program. Note that if `halt(p)` returns 1, we have an infinite while loop; i.e., the program never terminates! Our conclusion is

If our program halts, then it never halts.
If our program never halts, then it halts.

This is a logical contradiction, so our assumption that `halt()` is computable must be false.

- Note that the above construction omits the source code for the `sprintf()` function, but this could easily be inserted into the program.
- Similarly, if we could provide the source code for the `halt()` function, then we could insert it into the program as well.

¹34 is the ASCII value of the double quote character.

²If the string is not a valid C program we will say that this is the same as halting.

³This is a single line program, but we broke it into two lines so we could easily fit it on the page.

4. Hmmm...

- Is this a deficiency of the C programming language or is it impossible to implement `halt()` in *any* (reasonable) programming language?
- Obviously a program either will halt or it will run forever. Does this indicate that there are inherent limitations to computers?

5. Is there always a prime number between n^2 and $(n + 1)^2$?

- Mathematicians believe so, but its *never* been proven.
- If the halting problem were solvable, I could prove (or disprove) this conjecture using the following program:

```
main() {
    bigint n, a, b, i;
    for (n = 1; ; n++) {
        a = n*n;
        b = (n+1)*(n+1);
        for (i = a+1; i < b && !prime(i); i++)
            ;
        if (i == b) break; /* no prime found, halt program */
    }
}
```

- The type `bigint` is an unlimited size integer. This could be implemented by allocating more memory for a variable as more bits are needed. If the computer runs out of memory then we save the process's state and resume on a computer with more memory.
- Inserting the source code for `prime()` would be trivial (i.e., `prime()` is computable).
- If the conjecture is true, future descendants would keep our program running with computers containing more memory. Perhaps our program will be interrupted when the Sun becomes a red giant in about 7 billion years.

6. More problems than solutions.

- George Cantor (1873) demonstrated that infinity comes in different sizes. Infinite sets like the set of natural numbers or the set of rational numbers are *countably infinite*. Other infinite sets, like the set of real numbers are *uncountable infinite* – loosely speaking, the set of real numbers is “denser” than the set of natural numbers.
- The proof technique Cantor used to demonstrate this is called *diagonalization* (if we have time, I'll show you the trick).
- Cantor went on to show that the set of problems we can formally describe are uncountable infinite, whereas the set of programs we can write is only countably infinite. The halting problem is one the unsolvable problems.

7. Determining if a program will halt is possible with more restrictive programming languages (or with trivial programs).

- Finite Automata
- Deterministic Pushdown Automata (more on FA's and DPDA's later)

8. Bertrand Russell's *barber paradox*

- The barber in a certain town has a sign on the wall of his barber shop that says “I shave those men, and only those men, who do not shave themselves.” Let x denote any arbitrary man in town. More formally we have

$$(\text{barber shaves } x) \Leftrightarrow \neg(x \text{ shaves } x)$$

- What if $x = \text{barber}$?