The Halting Problem

1 Introducing the halting problem

Is there any decision problem, any set of words, that is undecidable? The answer is clearly yes. There are only countably many Java programs (or Turing machines), but uncountably many sets of words. But we'd like to see a specific example of an undecidable problem. And so far, I've given you some examples, such as ambiguity of context free grammars. But I haven't proved that these properties are undecidable.

Now I'm going to tell you the most famous example of an undecidable problem in computer science: the halting problem: to distinguish between those nullary programs that halt and those that hang. (A *nullary* program is one with no arguments.)

For example, the following program halts:

```
void programA () {
  nat sum = 0;
  nat count = 0;
  while (count <= 5) {
    sum = sum + count;
    count = count + 2 - 1;
  }
  if (count < 100) {
    return;
  } else {
    while (true) {}
  }
}
```

The following program hangs:

```
void programB () {
  nat sum = 0;
  nat count = 0;
  while (count <= 5) {
    sum = sum + count;
    count = count + 1 - 1;
  }
  if (count < 100) {
    return;
  } else {
```

```
    while (true) {}
  }
}
```

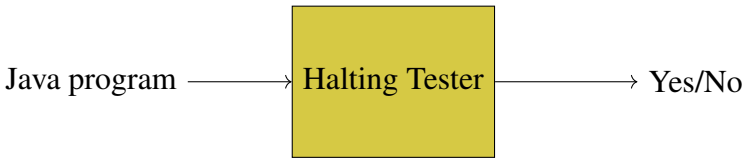
The next example is based on *Goldbach's conjecture*—the statement that every even number $\geqslant 4$ is a sum of two primes. It is currently unknown whether this is true or false.[1] Here's a program that hangs if Goldbach's conjecture is true, and halts if it's false:

```
boolean isprime(nat n) {
   for (nat i = 2, i < n, i++) {
      if (n mod i == 0) {return false;}
   }
   return true;
}

boolean isasumoftwoprimes(nat n) {
   for (nat i = 2, i < n, i++){
      if (isprime(i) and isprime(n-i)) {return true;}
   }
   return false;
}

void programC (){
   nat k = 4;
   while(isasumoftwoprimes(k)){
      k = k+2;
   }
}
```

Our question is whether the halting problem is decidable. Thus we seek a program



For example, if we feed in Program A, it returns True. If we feed in Program B, it returns False. If we feed in Program C, it return False if Goldbach's conjecture is true, and True otherwise.

Turing proved that the halting problem is undecidable, i.e. there is no such tester. This is called the *Halting Theorem*.

2 Proof of the Halting Theorem

The proof of the Halting Theorem is as follows. Suppose it is decidable.

[1] It has been checked mechanically that every even number from 4 to 4×10^{18} is a sum of two primes. But, for all we know, there might be a larger even number that isn't.

1. Now consider the *unary halting problem*: given a unary Java method

```
void f (String x){
   ...
}
```

   and a string y, does `f` terminate when called with argument y? We can reduce the unary halting problem to the nullary one: given a unary method `f` and a string x, obtain a nullary method `g` by taking the code of `f` and replacing `x` with y; then `g` terminates when called if `f` terminates when called with argument y. So since the nullary problem is (we're assuming) decidable, the unary one is too. So we have a program

```
boolean haltcheck (String somemethod, String y)
```

   where `somemethod` is the body of a unary method. When applied to M and y, this method returns true if M applied to y terminates, otherwise it returns false.

2. Next we turn this into a program

```
void hangcheck (String somemethod, String y){
  if haltcheck (somemethod, y) {
    while true {}
  } else {
    return;
  }
}
```

   This method, when applied to M and y, hangs if M applied to y terminates, otherwise it returns.

3. Next we turn this into a program

```
void doublehang (String y){
   hangcheck(y,y)
}
```

   This method, when applied to y (the body of a unary method), will hang if y applied to y terminates, otherwise it returns.

4. Finally let z be the body of `doublehang`. We see that `doublehang`, when applied to z, terminates iff it hangs. Contradiction.