Halting Problem

Discussion. Up to now, we analyzed what *can* be computed and on what computational device. However, there are some things that *cannot* be computed. One of these things is checking whether a program will halt on given data.

Of course, if a program halts, we know it halted, but what if it goes on and on? Maybe it will eventually halt – or maybe it will never halt? It turns out that in the general case, it is not possible to tell whether the program will eventually halt or not.

We do not teach this result to incoming students, since we want to preserve the right to lower their grades if their programs do not halt – just kidding. Seriously, for simple programs, it *is* possible to check whether a program halts, but, as we will show, not in the general case.

Let us now show the proof.

Theorem. *No algorithm is possible that, given a program p and data d, checks whether p halts on d.*

Proof. We will prove this result by contradiction. Let us assume that there exists an algorithm (i.e., a Java program) "halt-checker" that:

- *given* two strings: a program p and data d,
- *returns* true if p halts on d, and false otherwise.

This program is supposed to provide the correct answer for *any* Java program p and for *any* string d. In particular, this string d may be a string

```
public static ...
```

that describes a Java program: its first character is the letter p, its second character is the letter u, etc.

Such program-describing inputs are not unusual: e.g., a compiler can be viewed as a method that:

- takes, as an input, an ASCII string representing the program that we want to compile, and
- produces, e.g., the string describing the resulting executable code.

In the proof, we will use the possibility of such input strings.

Let us now build the following auxiliary program:

```
public static int aux(String x){
  if (halt-checker(x,x))
    {while(True) x = x;}
  else {return 0;}}
```

Will this program `aux` halt if, as the input `x`, we give it the string `aux` that forms the text of this program `aux`, i.e., the string that starts with

```
public static int aux(String x)
```

- If the program `aux` *halts* on this string `aux`, then, by definition of the halt-checker, the value

  halt-checker(aux,aux)

  is `true`. If you trace the above program `aux`, you will see that in this case, this program will go into an infinite loop – and thus, it will *not* halt.

- On the other hand, if the program `aux` *does not halt* on the string `aux`, then, by definition of the halt-checker, `halt-checker(aux,aux)` returns `false`. If you trace the above program `aux`, you will see that in this case, this program *will halt* – namely, it will return 0.

In both case, we get a contradiction:

- if we assume that it halts, we conclude that it does not halt, and
- if we assume that it does not halt, then we conclude that it halts.

This contradiction shows that halt-checkers are indeed not possible. The theorem is proven.

Comment. Not only we have a theoretical proof, but also:

- if someone presents us with a Boolean-valued program `halt-checker` that supposedly solves the halting program,
- we can immediately produce an example on which this presented program will not work correctly – namely, the above-constructed program `aux`.

Indeed, by its design, the program `aux`, when applied to the string `aux`, either returns 0 or never halts.

- If it returns 0, this means, by the design of this program, that the value `halt-checker(aux,aux)` is `false`. However, since `aux(aux)` returns 0, this means the program `aux` halts on the string `aux`. So, for this pair

  (`aux`, `aux`),

  the presented program `halt-checker` gives the wrong answer.

- If the program `aux`, when applied to the string `aux`, never halts, this means, by the design of this program, that the value `halt-checker(aux,aux)` is `true`. However, in this case, the program `aux` does not halt on the string `aux`. So, for this pair (`aux`, `aux`), the presented program `halt-checker` gives the wrong answer.

In both cases, the value `halt-checker(aux,aux)` differs from what a correct halt-checker should provide.