Første individuelle opgave

Nikolaj Dybdahl Rathcke
rfq695

December 4, 2014

1: Program Analysis

a

It is not possible to write such a program. However, it is possible in some cases. For example, if we had a program written in the following pseudocode

```
input x,y;
if true
  then x+=1;
  else y+=1;
end
```

the statement in the `else` branch is never used, and it is possible to remove the unused variable `y`. Some compilers have the ability to do this as well. However consider a program as below

```
input x,y;
if x < 5
  then x+1;
  else y+1;
end
```

We could never determine if `x` or `y` is used in an execution of the program since it relies on the input, so it is not always decidable. Maybe a requirement for the input is that $x < 5$ and `y` could be removed because it is unused or vice versa (this is assuming we know nothing about the input).

b

Since all Turing-complete programming languages suffers from the same problem as in a `WHILE` program, that is not possible.
A non Turing-complete language could be constructed so this is possible though. Consider a language which only supported variable declarations and variable alteration by simple arithmetic, so there were no jumps or branches. This means that you could run any program line for line. A program to detect unused variable could easily be created by checking which variables are declared and which variables are actually altered.

c

A property that could benefit a programmer is a program that can make small optimizations to your code, for example

```
if (2==2 and x)
  ..
```

could be reduced to

```
if x
  ..
```

obviously this is a very simple case, but it might even reduce time complexity. For example the following

```
input n;
k = 0;
for i in 1..n
  k+=i;
end
```

could be written as

```
input n;
k=n(n+1)/2;
```

which could save a lot of operations. This should be possible to make, but it needs to be very careful when it makes the changes and how efficient it can be made is also questionable.

Another property that would be of great value to the programmer, but that can not be computed by a program comes from the famous halting problem. The problem of determining whether a program will eventually terminate or if it will be stuck somewhere in the program and run forever.

2: Malware

a

It is not possible to create a malware detector that works once and for all since malware producers are constantly finding new ways to infect a system. Because there is no "key element" to identify malware by, it means that the producers of malware detection has to update their programs constantly to handle the new types of malware.
However, just because malware detectors is updated to detect new malware, it does not necessarily mean it is able to detect other malware that behaves similarly. It is "undecidable if two programs computes the same function"[1] which is a statement that follows from Rice's theorem.

b

Even though the "warfare" benefits both parties economically (producers of malware does it for economic reasons which makes way for the anti-malware business), it is also technical necessity. As long as there is money in making malware, we also need anti-malware to counteract it.
It might be seen as a real life problem as well: "As long as there are criminals, we need police to stop them". You can not imagine that anti-malware should not be developed, but you can not imagine that malware would suddenly stop being developed either, so we would have no need for anti-malware.

3: Church-Turing Thesis

a

As discussed in [1] there are several reasons to why we have multiple programming languages. There are factors that defines a good programming

[1] Fritz Henglein - datalogi som formel videnskab (slides), page 37 - translated

language, where common ones include size and speed of programs, and how easy it is to use the programming language. The reason we have so many different languages is because they each have their strengths and weaknesses. If a programming language was to be invented that was superior in all factors no matter what program you were writing, this would be the obvious choice, but since no such language exists yet, different programming languages are used. Either because they are more compact than others, because they are faster than others, or sometimes, because the programmer is adapted to or prefers a certain language. The general purpose languages today are very similar in most factors, but sometimes a language might perform better than all others when writing some specific programs.

b

These kind of languages, weak languages, can have advantages as also mentioned in [1]. These advantages include, among others, correctness proofs and upper boundaries for running times or memory use.
Another (important) way to use them is to make them solve specific tasks, so called specialized languages. These are often used in real life. If a machine is made to compute the same tasks over and over, a weak programming language can be used so that these tasks are efficiently computed.

References

[1] Torben Mogensen: Nogle programmeringssprog er mere lige end andre