The Virus Problem

CSE 5211 Analysis of Algorithms

Spring 2017 (April 26, 2017)

Dave Evans presents a nice description of the *virus problem* (VP), see On the Impossibility of Virus Detection [*]

[*] Evans, D. (2017). On the impossibility of virus detection

Definition 1: Virus

A virus is a computer program that when executed will copy its own code into another program.

Problem 1: Virus Problem

Input: *A description of a program* `P` *and its input* `x`.

Output: *If* `P(x)` *behaves like a virus (running it can infect other files) output* `True`. *Otherwise, output* `False`.

Assume there a program, virusDetect, that *decides* the Virus Problem (VP).

Listing 1: Virus Detection

1 ⟨ *Virus Detection* 1 ⟩≡

```
virusDetect(program P, input x) {
  if (P(x) acts like a virus) then true;
  else false;
}
```

A Diagonalization Argument

Assume there is a virus `V`. (Imagine how such a program could be written). Mimic the argument given in class and the notes to show there cannot be a program that decides the virus problem (VP).

That is, write a program, call it `D(program P)` if you like, that uses `virusDetect`, to create a contradiction.

Answer: The idea behind `D` is to use `virusDetect` to check if the input `P` is a virus, and then do the opposite. That is,

If `P` is a virus, `D(P)` simply halts without executing a virus.

If `P` is not a virus, `D(P)` executes virus `V`

Listing 2: Diagonal Counter Virus

2a ⟨*Diagonal Program* 2a⟩≡

```
D(program P) { if (virusDetect(P, P) then halt; else V; }
```

Now consider the execution of `D` on itself.

If `D(D)` acts like a virus, (That is, if `virusDetect(D, D))=true`), then `D(D)` halts, That is, `D(D)` never executes a virus `V`.

(If `D(D)` is a virus, then `D(D)` does not act like a virus.)

On, the other hand, if `D(D)` is a not a virus, then `D(D)` executes virus `V` acting like a virus.

(If `D(D)` is not a virus, then `D(D)` does acts like a virus.)

There are a contradictions in both cases.

Reduction of HP to VP

Spend a few minutes going over Evans' note On the Impossibility of Virus Detection.

Another way to show the virus problem is undecidable is to show HP *reduces* to VP. That is, if VP were decidable, then HP would be decidable. And, since HP is undecidable, VP must be also.

The main idea is to construct a program `makeVirus` that first executes any input program `P`, and then serially, if `P` halts, executes a virus `V`.

Listing 3: Make a Virus

2b ⟨*Make Virus* 2b⟩≡

```
makeVirus(program P)  {
  P;
  V;
}
```

Argue that program `halt` below decides the halting problem. Use this to conclude there cannot be a virus detection decider.

Listing 4: Halting Decider

2c ⟨*Halt Decider* 2c⟩≡

```
halt(program P) {
  if (virusDetect(makeVirus, P)) then true;
  else false;
}
```

Answer:

If `P` halts, `makeVirus` will execute virus `V` and `virusDetect(makeVirus, P)` will return `true`.

On, the other hand, if `P` does not halt, then `makeVirus` will never execute `V` and `virusDetect(makeVirus, P)` will return `false`.

In both cases the halting problem is correctly decided. Since this is not possible, then cannot be a virus detection program. There are a few wrinkles to iron out to fully nail down the argument. See [†].

[†] Evans, D. (2017). On the impossibility of virus detection

References

Evans, D. (2017). On the impossibility of virus detection.