P vs NP
for the (formalist-leaning) mathematician

Dominic van der Zypen

1. Introduction

I have long had an iffy feeling about the formulation of the **P** vs **NP** problem. I sort-of know Turing machines, I have a vague concept of what is "a class of problems", and I have heard many times that **NP** is the class of problems for which a solution can be checked in polynomial time for correctness.

All my ifs-and-buts add to a quite shaky view - and I decided to get to the bottom of things and provide a rigorous and (hopefully) mathematically appealing definition of Turing machines, languages, and the like.

2. Configurations

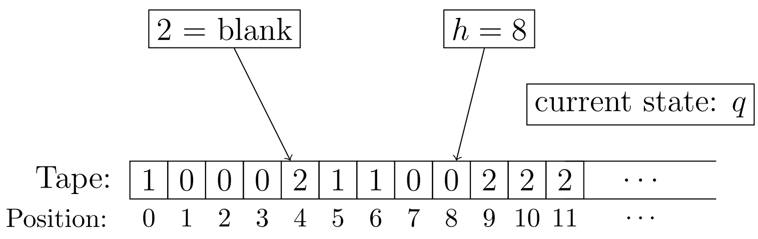
Let ω denote the first infinite ordinal (which can be thought of as $\mathbb{N}$, the set of non-negative integers). Recall that each $n \in \omega$ is an *ordinal*, that is, $0 := \emptyset$ and for $n > 0$, its members are the numbers $0, \ldots, n-1$, so $n = \{0, \ldots, n-1\}$. We write n^ω for the collection of all maps $f : \omega \to n$.

Our first central concept, the **configuration**, is the mathematical model of the notion of the **tape** of the Turing machine, together with the **writing head** and the **internal state** of the machine.

We fix a finite set Q such that $Q \cap \omega = 0$ and refer to Q as the *set of states*. Q has two special states, called *accept* and *reject*.

Definition 2.1. A *configuration* is a triple $C = (q, h, t) \in Q \times \omega \times 3^\omega$ (where $3 = \{0, 1, 2\}$) such that the sequence (function) $t : \omega \to 3$ is eventually constant with value 2. (This means that there is $N \in \omega$ such that $c(k) = 2$ for all $k \in \omega$ with $k \geq N$.) The collection of configurations over the set of states Q is denoted by $\mathrm{Config}(Q)$.

Remarks. The letter t stands for "tape". We interpret 2 as being the *blank* symbol. The *value* symbols are 0 and 1. The blank symbol 2 can be used to separate "input strings" consisting of 0,1. The value $h \in \omega$ can be thought of as the position of the *head* of the Turing machine, and $q \in Q$ is the current state.



3. Turing machines

We are ready for the central notion – maybe of computer science:

Definition 3.1. A *Turing machine* is a tuple (Q, δ), such that

- Q is a finite set with $Q \cap \omega = \emptyset$ containing two special elements *accept* and *reject*, and
- $\delta : Q \times \{0,1,2\} \to Q \times \{0,1,2\} \times \{-1,1\}$ is a function with the following property: if $q \in \{\textit{reject, accept}\}$ and $b \in \{0,1,2\}$, then $\delta(q,b) = (q,b,-1)$. If $\delta(q,b) = (q',b',s)$ for $q,q' \in Q, b,b' \in \{0,1,2\}$ and $s \in \{-1,1\}$ we write
  - $\mathrm{state}(\delta(q,b)) = q'$,
  - $\mathrm{output}(\delta(q,b)) = b'$, and
  - $\mathrm{step}(\delta(q,b)) = s \in \{-1,1\}$.

The function δ is called the *transition function* and can be looked at as the "clockwork" of the Turing machine.

Time for the synthesis: combining configurations and Turing machines!

We need the following *convention*: if $n \in \omega$, define the function $\mathrm{pred} : \omega \to \omega$ by $0 \mapsto 0$ and $n \mapsto n-1$ for $n \in \omega \setminus \{0\}$. By slight abuse of notation, we write $n-1$ instead of $\mathrm{pred}(n)$ for all $n \in \omega$.

Definition 3.2. Let $T = (Q, \delta)$ be a Turing machine, T induces a *configuration map*

$$\mathfrak{C}_T : \mathrm{Config}(Q) \to \mathrm{Config}(Q)$$

in the following way:

If $C = (q,h,c) \in \mathrm{Config}(Q)$ then $\mathfrak{C}_T\big((q,h,c)\big) = (q',h',c')$ where

- $q' = \mathrm{state}\big(\delta(q,c(h))\big)$,[1]
- $h' = h + \mathrm{step}\big(\delta(q,c(h))\big)$, and [2]
- $c' : \omega \to \{0,1,2\}$ is defined by $c'(h) = \mathrm{output}\big(\delta(q,c(h))\big)$ and $c'(x) = c(x)$ for all $x \in \omega \setminus \{h\}$. [3]

[1] recall that $c(h)$ is the value of the configuration at head position h.
[2] remember that $\mathrm{step}\big(\delta(q,c(h))\big) \in \{0,1\}$, and $0-1=0$ in our convention.
[3] that is c' agrees with c except possibly on the head position h.