P vs NP
for the rigorous 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

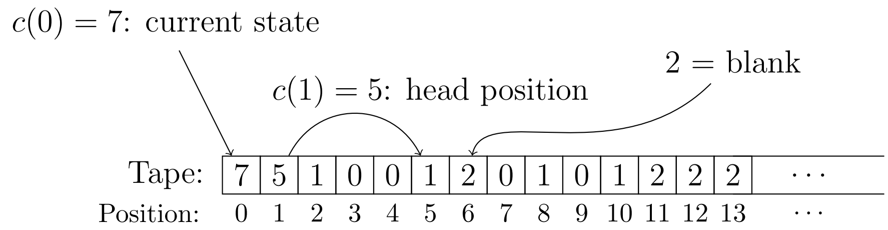
Let ω denote the first infinite ordinal (which can be thought of as $\mathbb{N}$, the set of non-negative integers). Recall that each $n \in \omega$ is an *ordinal*, that is, $0 := \emptyset$ and for $n > 0$, its members are the numbers $0, \dots, n-1$, so $n = \{0, \dots, n-1\}$. We write ω^ω for the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

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.

Definition 2.1. Let $n \geq 2$ be an integer. Then an *n-configuration* is a function (also called an *integer sequence*) $c \in \omega^\omega$ with the following properties:

(1) $c(0) \in n = \{0, \dots, n-1\}$,
(2) $c(1) \geq 2$,
(3) $c(k) \in \{0, 1, 2\}$ for all $k \in \omega \setminus \{0, 1\}$, and
(4) c 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 $k \geq N$.)

Next, we illustrate and explain the meanings of the entries of c in detail:



- $c(0) \in n$ represents the *state* of the possible n states $\{0, \dots, n-1\}$ of the configuration. States 0 and 1 (stored in the first cell, $c(0)$) have a special meaning: $0 =$ *reject*, $1 =$ *accept*.
- $c(1) \geq 2$ represents the position of the *read/write head* of the configuration.

- 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. Every configuration is eventually blank (=2).

We let $\mathrm{Config}(n)$ be the collection of n-configurations. Note that whenever $n \leq n' \in \omega$ and $n \geq 2$, we have $\mathrm{Config}(n) \subseteq \mathrm{Config}(n')$.

3. Turing machines

Definition 3.1. A *Turing machine* is a tuple $M = (n, \delta)$ where $n \in \omega, n \geq 2$ and $\delta : n \times \{0,1,2\} \to n \times \{0,1,2\} \times \{-1,1\}$ is a function with following property:

if $q \in \{0,1\}$[1] and $b \in \{0,1,2\}$, then $\delta(q,b) = (q,b,-1)$.

The interpretation of this is that δ gets constant whenever an *accept* or *reject* state has been reached.

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

- $\texttt{nextstate}(q,b) = q'$,
- $\texttt{output}(q,b) = b'$, and
- $\texttt{step}(q,b) = s \in \{-1,1\}$.

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

4. Synthesis: combining configurations and Turing machines

Definition 4.1. If $M = (n, \delta)$ be a Turing machine, then M induces a *configuration map*

$$\mathfrak{C}_M : \mathrm{Config}(n) \to \mathrm{Config}(n)$$

in the following way. If $c \in \mathrm{Config}(n)$, then we define $\mathfrak{C}_M(c) : \omega \to \omega$ by

- $\mathfrak{C}_M(c)(0) = \texttt{nextstate}(c(0), c(c(1)))$ [2]
- $\mathfrak{C}_M(c)(1) = \max\{2, c(1) + \texttt{step}(c(0), c(c(1)))\}$, [3]
- $\mathfrak{C}_M(c)(c(1)) = \texttt{output}(c(0), c(c(1)))$, and [4]
- $\mathfrak{C}_M(c)(x) = c(x)$ for all $x \in \omega \setminus \{0, 1, c(1)\}$.

So we have $\mathfrak{C}_M(c) \in \omega^\omega$, and it easy to check that $\mathfrak{C}_M(c) \in \mathrm{Config}(n)$.

5. Run time and worst-case run time

If $X \neq \emptyset$ is a set and $f : X \to X$, we define inductively for any $x \in X$:

- $f^{(0)}(x) = x$, and
- $f^{(n+1)}(x) = f\big(f^{(n)}(x)\big)$.

For the remainder of this section, fix $n \geq 2$.

[1] recall that 0,1 are special states with the meanings {*reject, accept*}, respectively.

[2] $c(0)$ is the current state in the set of possible states $\{0, \dots, n-1\}$, and $c(1)$ is the position of the read/write head, and finally $c(c(1))$ is the **value** of the cell at the head.

[3] $\texttt{step}(c(0), c(c(1))) \in \{-1, 1\}$.

[4] so the output created by the transition function δ gets inserted at the head position $c(1)$.

Definition 5.1. If M is an n-Turing machine and $c \in \mathrm{Config}(n)$, then we consider the set

$$\mathcal{T}_M(c) = \big\{n \in \omega : \mathfrak{C}_M^{(n)}(0) \in \{0,1\}\big\}.^5$$

(1) If $\mathcal{T}_M(c) \neq \emptyset$, we say that M *terminates* on constellation c.
(2) The *run time* of M on c defined by $t_M(c) = \min \mathcal{T}_M(c)$ if M terminates on c, and we set $t_M(c) = \infty$ otherwise.
(3) If there is $n \in \omega$ with $\mathfrak{C}_M^{(n)}(0) = 0$, then M is said to *reject* c.
(4) If there is $n \in \omega$ with $\mathfrak{C}_M^{(n)}(0) = 1$, then M is said to *accept* c.
(5) We define the the *language accepted by* M by

$$L(M) = \{c \in \mathrm{Config}(n) : M \text{ accepts } c\}.$$

For the *worst case run time* we need the notion of the length of a configuration. Note that every configuration is eventually constant 2.

Definition 5.2. If $c \in \mathrm{Config}(n)$, the the *length* of c is defined by

$$\mathrm{len}(c) = \min\{N \in \omega \setminus \{0,1\} : c(x) = 2 \text{ for all } x \geq N\} - 2.$$

(It is a bit aesthetically displeasing that one has to do this $\omega \setminus \{0,1\}[\ldots] - 2$ trick. This is because the first two cells $c(0), c(1)$ of each configuration c have special meanings, and the input starts at $c(2)$.)

Definition 5.3. If M is an n-Turing machine and $\ell \in \omega$ then we define the *worst case run time* to be

$$T_M(\ell) = \sup\{t_M(c) : c \in \mathrm{Config}(n) \text{ and } \mathrm{len}(c) = \ell\} \in \omega \cup \{\infty\}.$$

We say that M **runs in polynomial time** if there are positive integers j, k such that for all $\ell \in \omega$ we have $T_M(\ell) \leq \ell^j + k$.

6. The class P

$$\mathbf{P} = \big\{L \subseteq \omega^\omega : \text{there is an integer } n \geq 2 \text{ and an } n\text{-Turing machine } M \text{ such that } L = L(M) \text{ and } M \text{ runs in polynomial time}\big\}.$$

[5] This is the set of iterations n such that the state stored in the first cell is 0 (reject) or 1 (accept).