Computability

Three Equivalent Definitions

Function computation

- Function computation
- Language deciding
- Problem solving
    - intuitive definition
        - collection of Y/N questions
    - precise definition **3** → **2**
        - question = string
        - problem is a
            - set of codestrings for the questions (over Σ) q_i
            - map $\alpha : q_i \mapsto \{Y, N\}$
        - solvable if computable/decidable
    - problem is always countable
        - set of all questions of a problem is countable

Computing Functions (2→1)

- Given $L \subseteq \Sigma^*$:
    - encode Σ^* by numbers (e.g. the alphabetical enumeration)
        - possible because Σ^* is countable
        - $\gamma : w \mapsto n_w$ (bijection)
- Consider the function $f : \mathbb{N} \to \{0, 1\}$
    - $n \mapsto 0$ if $\gamma^{-1}(n) \notin L$
    - $n \mapsto 1$ else

Deciding Languages (1→2)

(we know how to decide languages)

- Given $f : \mathbb{N}^k \to \mathbb{N}$. (**a total function**)
    - consider input/output pairs $P = \{(x, y) | x \in \mathbb{N}^k, y = f(x)\}$
        - "graph set" of a function
    - cross product of a set remains the same cardinality
        - → this is countably infinite
    - visualization:

        - $\mathbb{N}^2$ as a graph
            - go from 0,0 to 0,1 to 1,0 to 1,1, ... (zig zag)
    - P can be seen as a language $P \subseteq \Sigma^*$ with $\Sigma = \{0, 1, 2, 3, .., 9, (,), ;\}$
    - Let A be an algorithm that decides P.
        - Use this A in alg for computing f as follows:
            - Input: x
            - Output: y
            - ```
              y = 0
              while not_finished:
                  if decision_algorithm((x,y)) == 'yes':
                      not_finished = false // finished
                      return y
                  else:
                      y += 1
              ```

Turing Computability

Is there an effective procedure for computing every finitely definable function on the natural numbers?

Turings approach:

- investigate & formalize what a human computer can and cannot do with brain, paper, pencil

Turing's Paper (1936)

On computable numbers with an application to the Entscheidungsproblem

Turing's view on Computing

- writing symbols on paper
    - finitely many symbols
        - otherwise abitrarily similar symbols
- behavior of a computer determined by
    - symbols **he** is observing
    - state of mind at that moment
        - assume a finite number of different states

Turing Machines

Definition

4-Tuple

- (K, Σ, δ, s)
              ```

- States K: is finite, non-empty
  - require halting states $h, yes, no \in K$
    - halting state
    - accepting state
    - rejecting state
  - (convenience items)
- Alphabet Σ: finite, non-empty (take alphabet)
  - require $\triangle, \_ \in \Sigma$
- Transition function $\delta : K \times \Sigma \to K \cup \{h, yes, no\} \times \Sigma \times \{\leftarrow, -, \rightarrow\}$
- Starting state $s \in K$