Computational Complexity

Idea: measure "hardness" of a computational problem by quantifying resources that are required to compute a solution with Turing machine or (quantum!) generalisations of T.M.s.

→ classification of problems into complexity classes, e.g.

P, BPP, NP, PSPACE, ...
↖ „classical"

BQP, QMA
↖ „quantum"

here: brief intro to comp. complexity with definitions / relations of the above classes; for a more thorough intro

with many more comp. classes see

e.g. S. Arora, B. Barak: Computational Complexity (Cambridge).

For simplification, general comp. problems are reduced to decision problems that are solved by YES/NO answers.

Example: (a) general problem:

factorize integer n!

↳ (b) decision problem:

does integer n have a non-trivial factor less than ℓ?

Note:

if the decision problem (b) can be efficiently solved, then so the general problem (a)!

Further formalization with notion of a "formal language":

Def.

A (formal) Language L over alphabet Σ is a subset $L \subset \Sigma^*$.

(Σ^*: set of all strings over Σ)

Any such formal language is equivalently described by its characteristic (indicator) function

$$\chi_L : \Sigma^* \longrightarrow \{0,1\}$$

$$x \longmapsto \begin{cases} 1 : x \in L \\ 0 : x \notin L \end{cases}$$

Examples of formal languages:

EVEN = set of all even integers

(in binary representation)

$= \{ 0, 10, 100, 110, 1000, 1010, \dots \}$,

PRIMES = set of all primes

$FACTOR = \{ (n, \ell) \mid n, \ell \in \mathbb{N}, \ell \leq n \text{ s.t. } \exists\, 1 < r \leq \ell : n = r k \}$

$k \in \mathbb{N}$

$HALT = \{ w \in \mathbb{N} \mid h(w) = 1 \}$

↑ Halting-function

Def.

Language L **decidable**

$:\Leftrightarrow$ χ_L Turing-computable

→ for any decidable language L exists →

a Turing machine M_L with the property:

$x \in L$: M_L on input x halts on output 1
("accepted")

$x \notin L$: M_L on input x halts on output 0
("not accepted")

→ decidable languages (and thus decision problems) can be classified by looking e.g. at the runtime or memory a T.M. M_L needs to decide on membership to L

→ complexity classes P and PSPACE!

Def: T.M. M, $x \in \Sigma^*$

$time_M(x)$:= runtime of M on input x
= number of configurations from $s_0 x$ to $s_e x'$

$\lceil k_1 = s_0 x \vdash k_2 \vdash \cdots \vdash k_n = s_e x' \rfloor$

$\rightsquigarrow \quad time_M(x) = n$

$space_M(x) :=$ number of tape-positions visited by read-write-unit during

$s_0 x \vdash^* s_e x'$

$\rightsquigarrow$ <u>Complexity class P</u>

("polynomial time")

$L \in P \quad :\Leftrightarrow \quad L$ is decided by T.M. M_L in <u>polynomial time</u>, i.e.

$\exists\, \alpha \in \mathbb{N} \quad \forall\, x \in L :$

$time_{M_L}(x) \leq |x|^\alpha$

$|x| :=$ length of $x \in \Sigma^*$

clearly $EVEN \in P$;

Agrawal, Kayal, Saxena 2004:

$PRIMES \in P$!

most likely: $FACTOR \notin P$

Complexity class $PSPACE$:

("polynomial space")

$L \in PSPACE :\Leftrightarrow$ L decided by T.M. M_L

in polynomial space;

i.e. $\exists\, a \in \mathbb{N},\ \forall x \in \Sigma^*$

$space_M(x) \leq |x|^a$

clearly $P \subseteq PSPACE$!

[in polynomial time read/write-unit can visit only a polynomially large number of tape positions]

→ EVEN, PRIMES ∈ PSPACE,

but also FACTOR ∈ PSPACE

[

gcd(u,v) can be computed in

polynomial time and space

→ $(\gcd(u,v))_{v=2,\dots,\lfloor\sqrt{u}\rfloor}$ can be com-

puted in polynomial space!

]

Complexity class NP

("non-deterministic, polynomial time")

(N) (P)

Colloquially: "NP = class of problems

for which solutions can be verified

in polynomial time by use of

a witness."

→

$\rightarrow$ FACTOR $\in$ NP

because $(u, \ell) \in$ FACTOR can be verified by non-trivial factor $r \leq \ell$ in polynomial time

formal definition:

(a)

$L \in NP :\Leftrightarrow \exists\, \alpha \in \mathbb{N}$ and T.M. M:

$x \in L \overset{!}{\Leftrightarrow} \exists\, w \in \Sigma^*$ s.t.

M on input x, w holds in 1 and $time_M(x,w) \leq |x|^\alpha$

an alternative definition use the notion of a <u>non-deterministic Turing machine</u> (NDTM):

(b)

$L \in NP :\Leftrightarrow$ L is decidable by a non-deterministic T.M. M_L in polynomial time

Definition of NDTM:

$$M = (S, \Sigma, \Gamma, \delta, s_0, \square, E)$$

exactly as for T.M., except for transition rules, which here are relations:

$$\delta : S \setminus E \times \Gamma \longrightarrow \mathcal{P}(S \times \Gamma \times \{N, L, R\})$$

$$(a, s) \longmapsto \delta(a, s) =$$

$$\{ (a_1, s_1, m_1),$$

$$(a_2, s_2, m_2),$$

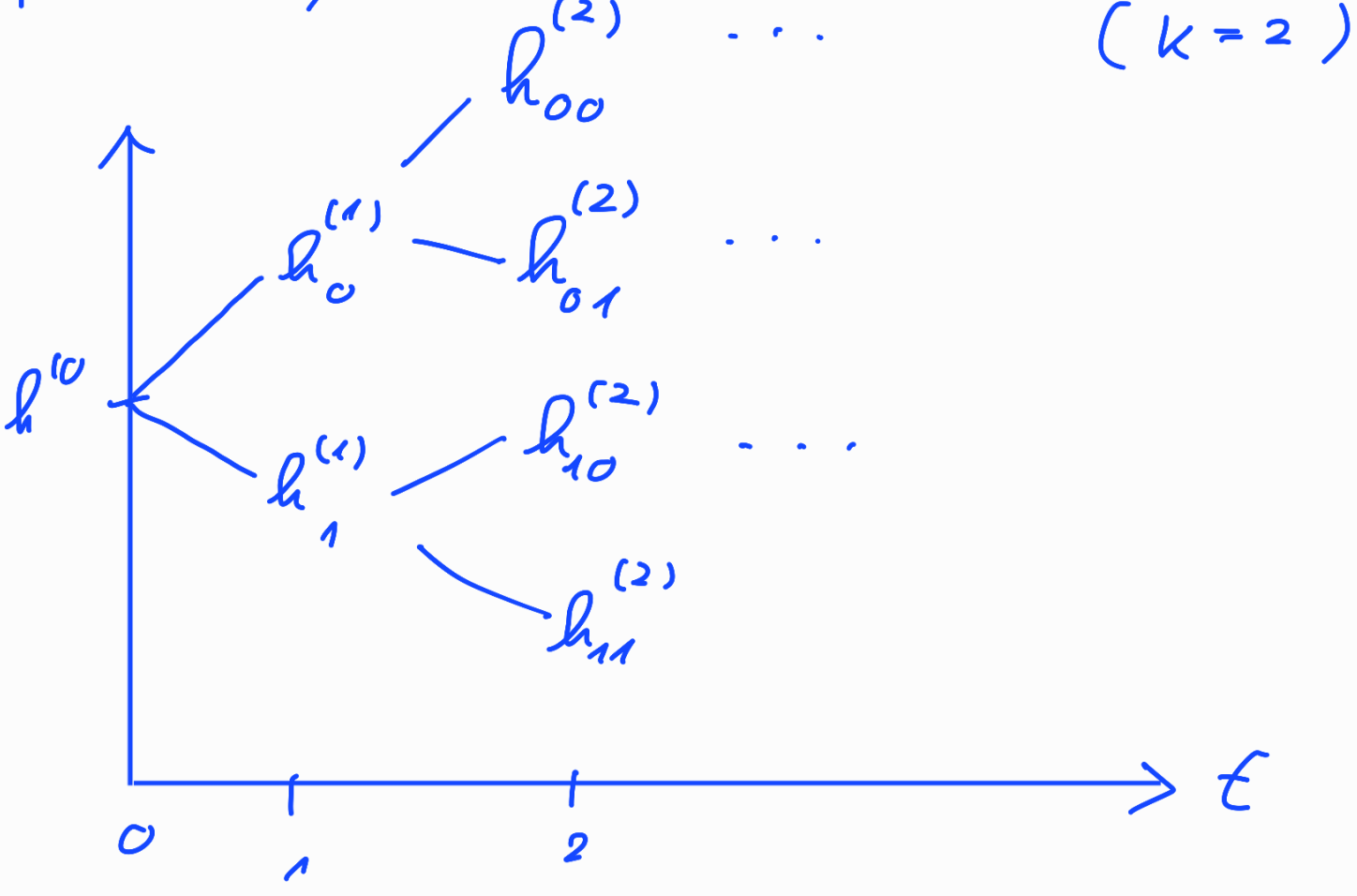
$$\vdots$$

$$(a_n, s_n, m_n)\}$$

→ configuration $h \in T^* S T^*$ implies $k > 1$ follow-up configurations in one step

→ k^u implied configurations in u steps;

pictorially:



Notations

- $asb \vdash_{ND} a's'b' :\Leftrightarrow \exists\, r \in \delta(s, b_0)$ s.t. $asb \vdash^{r} a's'b'$

- $h \overset{u}{\underset{ND}{\vdash}}{}^{*} h' \quad :\Leftrightarrow \quad \exists\, h_1, h_2, \ldots, h_u \quad \text{s.t.}$

  $h_1 = h, \quad h_u = h'$

  $h_i \underset{ND}{\vdash} h_{i+1}$

- NDTM M on input x halts on output $x' \quad :\Leftrightarrow \quad s_0 x \overset{*}{\underset{ND}{\vdash}} s_e x'$

- $\text{ntime}_M(x) :=$ runtime of NDTM M on input x
  ("non-deterministic time")

  $= \max \{ u \mid \exists x' \in \Gamma^* \text{ s.t. } s_0 x \overset{u}{\vdash}{}^{*} s_e x' \}$

2)

(f) $L \in NP \quad :\Leftrightarrow \quad L$ is decidable by NDTM M_L in polynomial time; i.e. $\exists\, d \in \mathbb{N}$

$\forall x \in L: \quad \text{ntime}_{M_L}(x) \leq |x|^d$

$\wedge \quad s_0 x \vdash^{*} s_e 1$.

equivalence of definitions (a) and (b)

shown e.g. in Arora & Barak (cf. Lecture).

clearly: $P \subseteq NP \subseteq PSPACE$!

Millenium problem: prove $P \neq NP$!