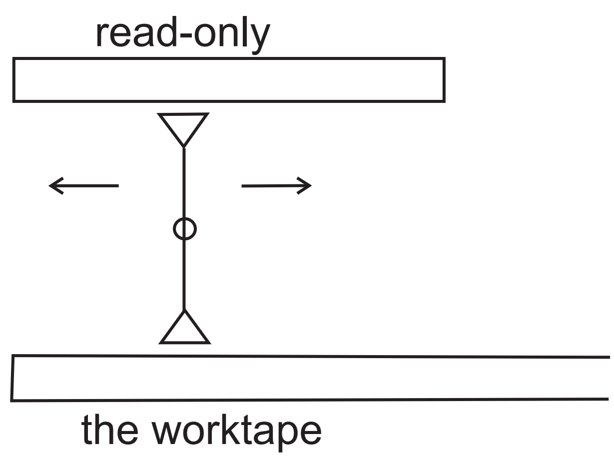
A Turing Machine Distance Hierarchy

Stanislav Žák and Jiří Šíma

Institute of Computer Science
Academy of Sciences
Prague
Czech Republic

Deterministic or nondeterministic Turing machines

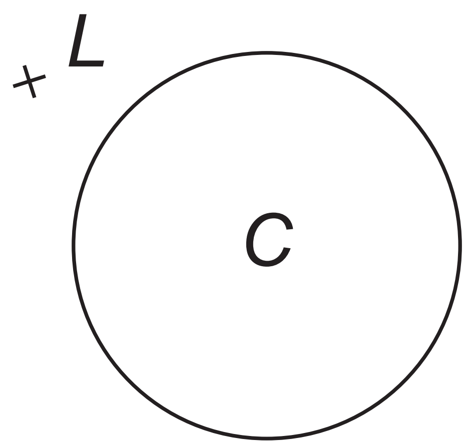


Complexity measures

Complexity bounds

Complexity classes

given a complexity measure



a complexity class C
given by a bound c

L within a complexity bound c_1

$$c_1 > c$$

Problem: to find L such that c_1 is very near to c

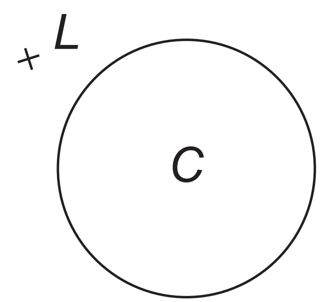
The complexity of a computation

$$\equiv$$

The amount of the source exhausted during its simulation on a fixed universal machine

---

An example:



$C = PSPACE \qquad L \notin C$

a) $c_1 >$ each polynomial

b) p – a fixed polynomial

$c_1 > p \cdot k$ – for each constant k

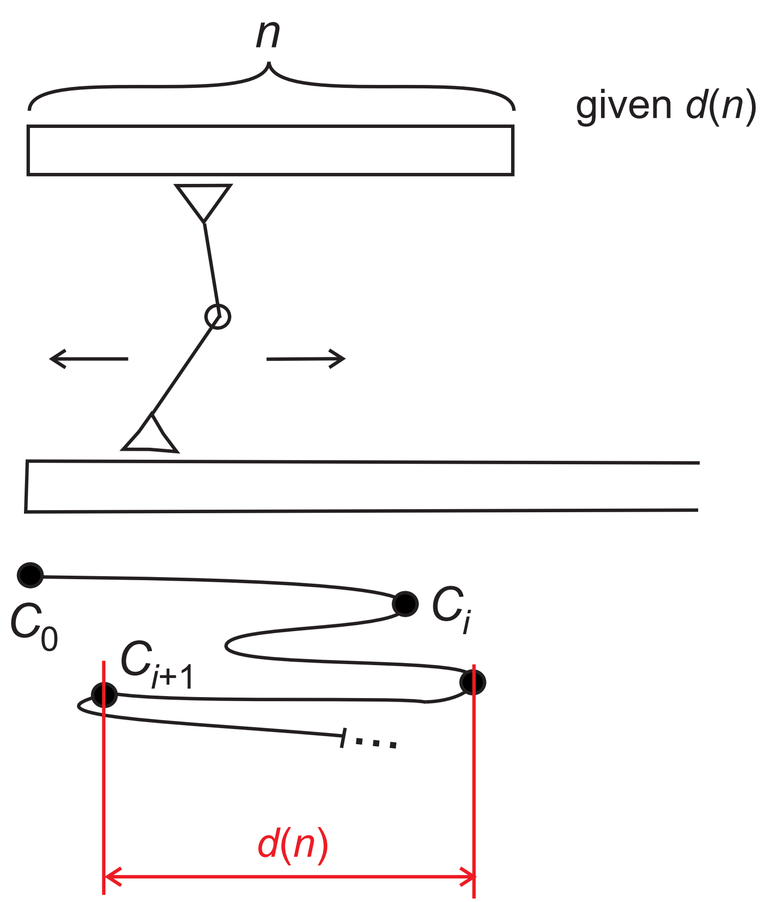
it depends on the number of the worktape symbols for different machines

c) a fixed polynomial – say n – + a fixed worktape alphabet

$c_1 > n + k$ for each constant k

d) a fixed universal machine

$c_1(n) = n + k \Rightarrow \exists k_0 SPACE_U(n + k_0 + 1) \supsetneqq SPACE_U(n + k_0)$



$c_0, c_1, c_2, \ldots, c_k$ a subsequence of configurations

$k \equiv$ the distance complexity of the computation

Theorem

Let U be a fixed universal machine

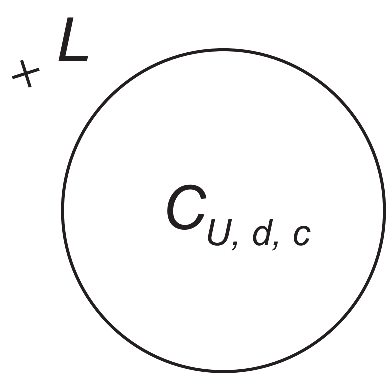
$c : N \to N$, a recursive function (complexity bound)

$d : N \to N$, $d(n) \geq \log_2 n$

$\Rightarrow$

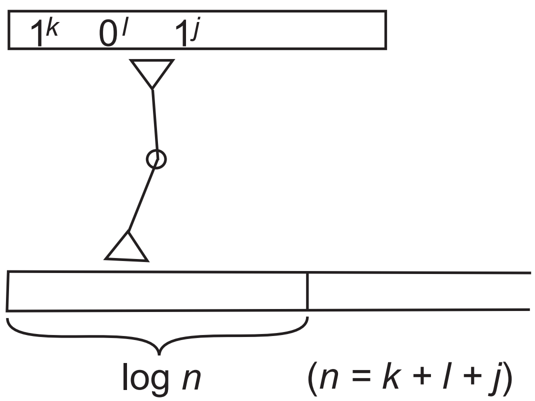
$\exists L \qquad L \in C_{U,d(n+1)+K,c(n+1)}$

$\qquad\quad L \notin C_{U,d,c}$



$C_{U,d,c} = \{L | \exists p \forall u \; u \in L \longleftrightarrow$ using the distance $d(|u|)$ during the simulation on U according to p u needs only $c(|u|)$ nodes of d-subsequence to be accepted$\}$

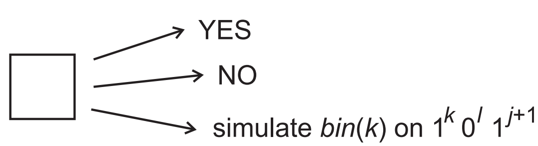
S – a recursive set of programs
$R =_{df} \{1^k 0^l | k, l > 0, bin(k) \in S\}$
$F : R \to S \qquad F(1^k 0^l) = bin(k)$
$\forall p \in S \ L_p$ be a uniformly recursive part of $L(M_p)$
$C =_{df} \{L_p | p \in S\}$
M :



1. action: to check $1^k 0^l 1^j$, to construct $\log n$, to construct $bin(k)$ and verify $bin(k) \in S$.

2. action: to decide whether $1^k 0^l \in L_{bin(k)}$ or not

3. action:



$z(r) =_{df}$ the first j s.t. computing on $1^k 0^l 1^j$ N decides whether $1^k 0^l \in L_{bin(k)}$

a) $r1^{z(r)} \in L(M) \longleftrightarrow r \notin L_{F(r)}$
b) $\forall j < z(r)$
$r1^j \in L(M) \longleftrightarrow r1^{j+1} \in L_{F(r)}$

$L(M) \in C \Rightarrow \exists r(L = L(M) = L_{F(r)})$
according to a) $r1^{z(r)} \in L \longleftrightarrow r \notin L$
according to b) $r \in L \longleftrightarrow r1^{z(r)} \in L$

A contradiction !

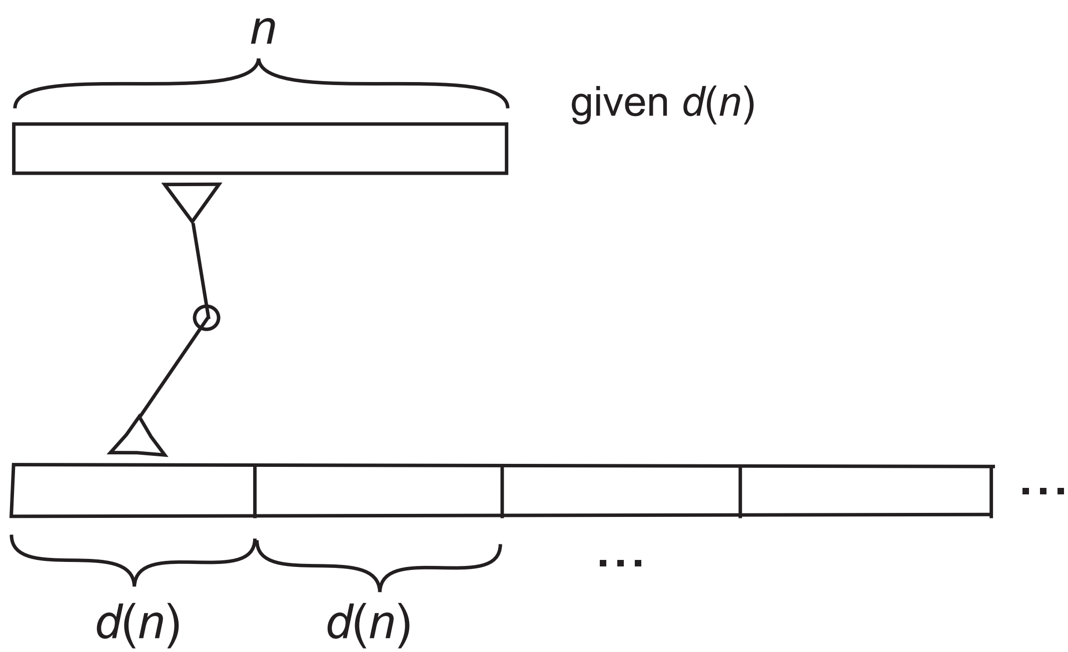
The contradiction in Cantor's diagonalization:

$$r \in L \longleftrightarrow r \notin L$$

Our contradiction:

$$r \in L \longleftrightarrow r1^{z(r)} \in L \longleftrightarrow r \notin L$$

A similar measure – s.c. *buffer measure*



the buffer complexity of a computation

=

the number of crosses of frontiers between
the blocks of length $d(n)$ (during its simulation on U)

the computational action inside the blocks are not detected,
they are gratis

Theorem

$C : N \to N$ – a recursive complexity bound
$d : N \to N$ – a recursive function, $d(n) \geq \log_2 n$
U – a fixed universal machine
$\Rightarrow$
$\exists K$ constant $\exists L$ – language

$L \in BUFFER_{U,d_{(n+1)+K},c_{(n+1)}}$

$L \notin BUFFER_{U,d,c}.$