Computational Complexity Theory

Block 3: Abstract measures of computational complexity

David Orellana Martín
Mario de J. Pérez Jiménez

Research Group on Natural Computing
Dpt. of Computer Science and Artificial Intelligence
University of Seville

dorellana@us.es (http://www.cs.us.es/∼dorellana/)
marper@us.es (http://www.cs.us.es/∼marper/)

MSc in Mathematics

Acad. year 2021-2022

Index

* Complexity measures.
* **Blum** axioms.
* Generation of dynamic complexity measures.
  - **Relative recursion** theorem.
* Computational complexity abstract classes.
  - **Borodin gap** theorem.
* Existence of **intractable problems**.
* **Blum's speedup** theorem.

It is recommended to read Chapter 14 of "*M.D. Davis, R. Sigal, E.J. Weyuker.* **Computability, Complexity and Languages. Fundamentals of Theoretical Computer Science**. Academic Press, Inc., Second Edition, 1994, XIX + 609 pages".

Dynamic complexity measures

Computational resources needed to execute a mechanical procedure:

- Complexity measures.
  - Static.
  - Dynamic.

Effective enumeration of DTMs: $\{M_e \mid \ e \in \mathbf{N}\}$.

- $\varphi_e^{(n)}$: function of arity $n \geqslant 1$ computed by M_e.

$\{\varphi_e^{(n)} \mid \ e \in \mathbf{N}\}$: set of the n–ary **computable** functions by DTMs.

Relevant results in computability theory

(a) A **computable predicate** is a total computable Boolean function.

(b) A **set** is **computable iff** its characteristic function is computable.

(c) A **set** is **resusively enumerable** (r.e.) **iff** is the domain of a computable function.

(d) Every computable set is a r.e. set.

(e) There exist r.e. sets that are **not** computable; for example, the set $\mathcal{K} = \{x \in \mathbf{N} \mid \varphi_x^{(1)}(x) \downarrow\}$ (the set of the **halting problem**).

(f) A partial function is computable **iff** its graphic is a r.e. set.

Dynamic complexity measures

Definition: A **dynamic computational complexity measure**[1] is a set of 1–ary functions $\{f_e \mid e \in \mathbf{N}\}$ from $\mathbf{N}$ onto $\mathbf{N}$, such that:

(a) For each $e \in \mathbf{N}$ the following holds: $\mathrm{dom}(f_e) = \mathrm{dom}(\varphi_e^{(1)})$.

(b) The predicate $\theta(e,x,y) = \left\{ \begin{array}{ll} 1 & \text{si } f_e(x) = y \\ 0 & \text{e.c.o.c.} \end{array} \right\}$ is **computable**.

We will note, briefly, $\theta(e,x,y) \equiv (f_e(x) = y)$.

Conditions (a) and (b) are denominated **Blum axioms**.

[1] M. Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. **Journal of the ACM**, **14**, 2 (1967), pages 322–336.

Manuel Blum

* Born in Caracas in April 26th 1938.
* Studied at MIT (Master in Computer Science and Engineering).
* PhD in Math (supervisor: Marvin Minsky).
* Among his PhD students, excelled L.M. Adleman.
* Turing Prize in 1995.

Medidas dinámicas de complejidad

First, let us see that 1–ary functions that form a complexity measure are, all of them, computable.

Proposition: *If* $\{f_e \mid e \in \mathbf{N}\}$ *is a complexity measure, then for each* $e \in \mathbf{N}$ f_e *is a computable function.*

> It is enough to take into account that the graphic of f_e, $G(f_e) = \{(x, y) \in \mathbf{N} \times \mathbf{N} \mid f_e(x) = y\}$, is a computable set (thus, r.e.) because for each $(x, y) \in \mathbf{N} \times \mathbf{N}$ the following holds: $(x, y) \in G(f_e) \Leftrightarrow \theta(e, x, y) = 1$.

Dynamic complexity measures

Theorem: *Let* $\{f_e \mid e \in \mathbf{N}\}$ *a set of 1–ary functions over* $\mathbf{N}$.
The following statements are equivalent:

(a) *The predicate* $\theta(e,x,y) \equiv (f_e(x)=y)$ *is computable.*

(b) *The predicate* $\theta'(e,x,y) \equiv (f_e(x)<y)$ *is computable.*

(c) *The predicate* $\theta''(e,x,y) \equiv (f_e(x)\leqslant y)$ *is computable.*

(a) ⇒ (b): It is enough to take into account that

$\theta'(e,x,y) \equiv f_e(x) < y \Leftrightarrow \exists z < y\ (f_e(x) = z) \Leftrightarrow \exists z < y\ \theta(e,x,z)$

(b) ⇒ (c): It is enough to take into account that

$\theta''(e,x,y) \equiv f_e(x) \leq y \Leftrightarrow f_e(x) < y+1 \Leftrightarrow\ \theta'(e,x,y+1)$

(c) ⇒ (a): It is enough to take into account that

$\theta(e,x,y) \equiv f_e(x) = y \Leftrightarrow (y = 0 \wedge f_e(x) \leq 0) \vee (y \neq 0 \wedge f_e(x) \leq y \wedge \neg(f_e(x) \leq y-1))$

$\theta(e,x,y) \equiv (y = 0 \wedge \theta''(e,x,0)) \vee (y \neq 0 \wedge \theta''(e,x,y) \wedge \neg\theta''(e,x,y-1))$

TIME as a complexity measure (I)

The **complexity measure TIME** is the set of 1–ary functions over $\mathbf{N}$, $\{t_e \mid e \in \mathbf{N}\}$, defined as follows:

$$t_e(x) = \begin{cases} \text{number of steps in the computation } M_e(x), & \text{if } M_e(x)\downarrow \\ \uparrow & \text{, if } M_e(x)\uparrow \end{cases}$$

This set of functions verifies, obviously, the first Blum axiom.

Let us see that the set $\{t_e \mid e \in \mathbf{N}\}$ satisfies the second Blum axiom.

For that, it is enough to prove that the predicate $\theta''(e, x, y) \equiv (t_e(x) \leq y)$ is computable.

TIME as a complexity measure (II)

We present an algorithmic scheme that stablishes the computability of the predicate $\theta''(e, x, y) \equiv t_e(x) \leq y$.

Input: $(e, x, y) \in \mathbf{N} \times \mathbf{N} \times \mathbf{N}$

```
if e = 0 then return 1
otherwise
  if y = 0 then return 0
  otherwise
     t ← 1
     while t ≤ y do
       if configuration C_t of M_e(x) is halting then return 1
       otherwise
         t ← t + 1
     return 0
```

SPACE as a complexity measure (I)

The **complexity measure SPACE** is the set of 1–ary functions over **N**, $\{s_e \mid e \in \mathbf{N}\}$, defined as follows:

$$s_e(x) = \begin{cases} \text{higher value of any variable in the computation } M_e(x), & \text{si } M_e(x)\downarrow \\ \uparrow & \text{, si } M_e(x)\uparrow \end{cases}$$

This set of function verifies, obviously, the first Blum axiom.

Let us see that the set $\{s_e \mid e \in \mathbf{N}\}$ satisfies the second Blum axiom.

For that, it is enough to prove that the predicate $\theta''(e, x, y) \equiv (s_e(x) \leq y)$ is computable.

Notation: If $M_e(x)\downarrow$ y $t \in \mathbf{N}\}$, we will note $s_{e,t}(x)$ the higher value of any variable in configuration C_t of $M_e(x)$.

SPACE as a complexity measure (II)

Let $(e, x, y) \in \mathbf{N} \times \mathbf{N} \times \mathbf{N}$ such that $M_e(x)\downarrow$. Then the set of configurations C_t of $M_e(x)$ is finite such that $s_{e,t}(x) \leq y$. Let $n(e, x, y)$ the cardinal of that set.

Entrada: $(e, x, y) \in \mathbf{N} \times \mathbf{N} \times \mathbf{N}$

$t \leftarrow 0$

`while` $t \leq n(e, x, y)$ `do`

  `if configuration` C_t `of` $M_e(x)$ `verifies` $s_{e,t}(x) > y$ `then return 0`

  `otherwise`

    `if` C_t `is halting then return 1`

    `otherwise`

      $t \leftarrow t + 1$

`return 0`

Note: take into account that if in computation $M_e(x)$ there exist two configurations $C_t, C_{t'}$ (with $t \neq t'$) such that $C_t = C_{t'}$, then $M_e(x)\uparrow$

A class of functions that is NOT a complexity measure

The set of **all** the 1–ary **computable functions** $\{\varphi_e^{(1)} \mid e \in \mathbf{N}\}$ **is not** a complexity measure.

It is enough to prove that the second Blum axiom is **not** verified.
Otherwise, the predicate $\theta(e, x, y) \equiv (\varphi_e^{(1)}(x) = y)$ would be computable.

In that situation, the function g from $\mathbf{N}$ onto $\mathbf{N}$ defined by

$$g(x) = \begin{cases} 0 & \text{if } \varphi_x^{(1)}(x) \downarrow \\ \uparrow & \text{if } \varphi_x^{(1)}(x) \uparrow \end{cases}$$

would be computable because its graphic
$\{(x, 0) \mid \varphi_x^{(1)}(x) \downarrow\} = \{(x, 0) \mid x \in \mathcal{K}\}$ would be a r.e. set.

Let $e' \in \mathbf{N}$ such that $g = \varphi_{e'}$. Then, the following would verify:

$$\theta(e', x, 0) \equiv (\varphi_{e'}^{(1)}(x) = 0) \Leftrightarrow g(x) = 0 \Leftrightarrow x \in \mathcal{K}$$

That contradicts the assumption that the predicate θ is computable.

Generation of complexity measures

Let f a partial function from $\mathbf{N}$ onto $\mathbf{N}$. If $f(x) \uparrow$ then we will note $f(x) = +\infty$.

Definition: *A* **scale factor** *is a total computable 1–ary function, h, such that*

* *h is growing: for each $x \in \mathbf{N}$ it holds that $h(x) \leqslant h(x+1)$.*
* $\lim_{x \to +\infty} h(x) = +\infty$.

Theorem: *If $\{f_e \mid e \in \mathbf{N}\}$ is a complexity measure and h is a scale factor, then $\{h \circ f_e \mid e \in \mathbf{N}\}$ is other complexity measure.*

* A proof of this result can be found in the recommended text (**theorem 1.1, pages 421-422**).

Scale factors provide a **mechanism** to **generate** new complexity measures from other measures.

Definition: *A* **1–ary predicate**, $\theta(x)$, *over* $\mathbf{N}$ **it is verified asymptotically** *iff there exists* $n \in \mathbf{N}$ *such that* $\forall x \geqslant n \ (\theta(x) = 1)$.

Theorem: (**Relative recursion**) *Let* $\{f_e \mid \ e \in \mathbf{N}\}$ *and* $\{g_e \mid \ e \in \mathbf{N}\}$ *complexity measures. There exists a total computable function,* $h : \mathbf{N}^2 \to \mathbf{N}$, *verifying:*

(a) $\forall x \ \forall y \ (h(x, y) < h(x, y + 1))$.

(b) $\forall e \ \exists n \ \forall x \geqslant n \ (f_e(x) \leqslant h(x, g_e(x)))$.

(c) $\forall e \ \exists n \ \forall x \geqslant n \ (g_e(x) \leqslant h(x, f_e(x)))$.

(*Notation:* $h(x, +\infty) = +\infty$, *for each* $x \in \mathbf{N}$)

- A proof of this result can be found in the recommended text (**theorem 1.2, pages 422-423**).

Two arbitrary complexity measures are always related between them; specifically, through a "kind of scale factor" (asymptotically).

Abstract complexity classes

Definition: *Let* $\mathcal{M} = \{f_e \mid e \in \mathbf{N}\}$ *a complexity measure and* g *a 1–ary total computable function. The* **complexity class** *determined by* g*, with respect to* $\mathcal{M}$ *is:*

$$\mathcal{C}_g^{\mathcal{M}} = \{\varphi_e \mid \varphi_e \text{ total} \wedge \exists n_0 \forall x \geqslant n_0 \ (f_e(x) \leqslant g(x))\}$$

Question: If $g < h$ (asymptotically) can $\mathcal{C}_g^{\mathcal{M}} \subsetneq \mathcal{C}_h^{\mathcal{M}}$ be ensured?

Theorem (**gap, Borodin**): *Let* $\{f_e \mid e \in \mathbf{N}\}$ *a complexity measure. Let* $g(x, y)$ *a total computable function such that* $\forall x, y \ (g(x, y) > y)$*. There exists a total computable function* h *from* $\mathbf{N}$ *onto* $\mathbf{N}$ *verifying:*

$$\forall e \ \forall x > e \ (f_e(x) \downarrow \wedge f_e(x) < g(x, h(x)) \Longrightarrow f_e(x) \leqslant h(x))$$

* A proof of this result can be found in the recommended text (**theorem 2.1, pages 426-427**).

Corolary: *Let $\mathcal{M}$ a complexity measure and $r(x)$ a total computable function such that $\forall x\ (r(x) \geqslant x)$. There exists a total computable function, $h : \mathbf{N} \to \mathbf{N}$, verifying that $\mathcal{C}^{\mathcal{M}}_{r \circ h} = \mathcal{C}^{\mathcal{M}}_{h}$.*

Interesting consequence:

- **There exists a total computable function**, $h'(x)$, *verifying the following*:
  - Every DTM whose execution time over a "fast" machine is bounded by $h'(x)$, asymptotically, will take a time bounded by $h'(x)$, asymptotically, when it is executed in a **much slower machine**.

Existence of intractable problems

Theorem: *Let* $\{f_e \mid e \in \mathbf{N}\}$ *a complexity measure. For each 1–ary total computable function,* f*, there exists a 1–ary computable function,* g*, verifying that if* $e \in \mathbf{N}$ *satisfies* $g = \varphi_e^{(1)}$*, then*

$$\exists n \, \forall x \geqslant n \, (f(x) < f_e(x))$$

* A proof of this result can be found in the section **Material de trabajo** of the subject website.

Computational interpretation: With respect to any complexity measure, there exists a problem X that is **intractable** with respect to that measure

(any mechanical solution of X takes an "arbitrarily big" amount of resources, for instances of "*big* size").

Existence of problems that lack of optimal solutions

Theorem: (**speedup**, **Blum**) *Let* $\{f_e \mid e \in \mathbf{N}\}$ *a complexity measure and* g *a 1–ary total computable function and growing. There exists a 1–ary total computable function,* h*, verifying:*

- **For each DTM,** M_e**, that computes** h**, there exists other DTM,** $M_{e'}$**, that also computes** h **and, besides, verifies:** $\exists n\ \forall x \geqslant n\ (g(f_{e'}(x)) < f_e(x))$**.**

* A proof of this result can be found in the recommended text (**theorem 4.3, pages 435-438**).

Corolary: *Let* $\{f_e \mid e \in \mathbf{N}\}$ *a complexity measure. There exists a 1–ary total computable function that* **lacks** *of an* **optimal DTM** *that computes it.*

Computational interpretation of the speedup theorem

With respect of any abstract complexity measure, there exist problems that lack of a *optimal DTM* that solves it.

Interesting consequence: Let A (**extremely fast**) and B (**extremely slow**) two machines that implement DTMs. There exists a problem such that for each DTM, M_e, that solves it, there exists another DTM, $M_{e'}$, that also solves it and, besides,

* **$M_{e'}$ executed on the slow machine B is strictly faster than M_e executed on the fast machine A (asymptotically).**

Let A and B two machines such that the first one is 10^6 faster than the second one

- If for each DTM M_e, we denote $t_e^A(x)$ and $t_e^B(x)$ the necessary times to execute $M_e(x)$ on the machines A and B, respectively, then $t_e^A(x) = \frac{t_e^B(x)}{10^6}$.
- Blum's speedup theorem is applied to the complexity measure $\{t_e^B \mid e \in \mathbf{N}\}$ and the total computable function $g(x) = 10^6 x$.
- Then there exists a total computable function h from $\mathbf{N}$ onto $\mathbf{N}$ such that
  - For each DTM M_e that computes h there exists another DTM $M_{e'}$ that also computes h and, besides, verifies that there exists $n_1 \in \mathbf{N}$ verifying $g(t_{e'}^B(x)) < t_e^B(x)$, for each $x \geqslant n_1$. Therefore

$$t_{e'}^B(x) < t_e^A(x), \text{ for each } x \geqslant n_1$$

There exists a problem (represented by h) that for each mechanical solution M_e of it, there exists another mechanical solution $M_{e'}$ that, executed on the extremely slow machine B, it takes less time than the solution M_e executed on the extremely fast machine A.