Computable Analysis toolkit

All the definitions that follow are directly taken from the book **?**. We specify on which page the definition is given with brackets, for example we write {12} for page 12.

1 Questions

2 Basics in Discrete Complexity Theory

2.1 Deterministic computation

{13} In this book, Turing machines are described to have multiple states and tapes, with two distinguished states (input and halting state), and two distinguished tapes: a read-only input tape, and a write-only output tape. They also can only move right or left (they cannot stay on the same cell).

{14} A recognizer is a Turing machine that has two different halting states: an accepting state, and a rejecting state. For such a machine, we denote by $L(M)$ the set of strings for which the machine halts in the accepting state.

{14} $\mathrm{time}_M(s)$ is the number of steps that the machine M makes before halting on input $s \in \Gamma^*$. $\mathrm{space}_M(s)$ is the number of cells *that are not on the input and output tapes* that the machine M visits before halting on input $s \in \Gamma^*$. We define, for a machine M its time and space complexity functions as

$$t_M(n) := \max\{\mathrm{time}_M(s) : s \in \Gamma^n\}, \tag{1}$$

$$s_M(n) := \max\{\mathrm{space}_M(s) : s \in \Gamma^n\}. \tag{2}$$

For a computable set A, it is said to have time (space) complexity $\psi : \mathcal{N} \to \mathcal{N}$ if there exists a Turing machine M computing A such that $t_M(n) \leq \psi(n)$ ($s_M(n) \leq \psi(n)$). The complexity of a computable function ϕ is defined analogously. {15} We say that a computable set (function) A (ψ) is computable in polynomial time if ψ is a polynomial. $\boldsymbol{P}$ is the class of polynomial-time computable sets, and $\boldsymbol{FP}$ is the class of polynomial-time computable functions.

{15} **Conjecture: extended Church-Turing thesis.** For every two reasonable computation model, the time and space complexity of a given problem differ at most of a polynomial factor.

2.2 Nondeterministic computation

{15} In this book, nondeterministic Turing machines are described as Turing machines which transition function maps to a subset of all possible transitions.

{16} $A \subset \Gamma^*$ is recognized by a ND Turing machine M if there exists at least one computation statring at s path that halts in accepting state iff $s \in A$.

{16} $\mathrm{time}_M(s)$ is the number of steps that the machine M makes before at least one accepting computation path halts on input $s \in \Gamma^*$. $\mathrm{space}_M(s)$ is the number of cells that the machine M visits before at least one accepting computation path halts on input $s \in \Gamma^*$. We define, for a machine M its time and space complexity functions as

$$t_M(n) := \max\{\text{time}_M(s) : M \text{ accepts } s,\ s \in \Gamma^n\}, \tag{3}$$

$$s_M(n) := \max\{\text{space}_M(s) : M \text{ accepts } s,\ s \in \Gamma^n\}. \tag{4}$$

$\boldsymbol{NP}$ is the class of sets accepted by polynomial time nondeterministic Turing machines.

{17} More sets are introduced, with the obvious meaning: $\boldsymbol{LOGSPACE}$, $\boldsymbol{NLOGSPACE}$, $\boldsymbol{EXP}$, $\boldsymbol{PEXP}$, $\boldsymbol{PSPACE}$, $\boldsymbol{NPSPACE}$, $\boldsymbol{FLOGSPACE}$.

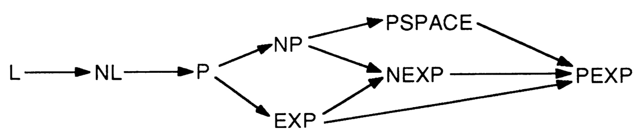

Figure 1.1. Inclusion relations on some complexity classes. In the above we use L for $LOGSPACE$ and NL for $NLOGSPACE$, and we write $A \to B$ for $A \subseteq B$.

2.3 Reducibilities

Definition *D1.1* - {18} *(Many-to-one reducibility).* A set A is polynomial-time many-to-one reducible to a set B, written $A \leq_m^P B$, if there exists $\phi \in \boldsymbol{FP}$ such that for all $s \in \Gamma^*$, $s \in A \iff \phi(s) \in B$.

Definition *D1.2* - {18} *(Hardness and completeness).* A set A is $\leq_m^P$-hard for the class $\mathcal{C}$ if for every $B \in \mathcal{C}$, $B \leq_m^P A$. A set A is $\leq_m^P$-complete for the class $\mathcal{C}$, or alternatively $\mathcal{C}$-complete if $A \in \mathcal{C}$ and A is $\leq_m^P B$-hard for the class $\mathcal{C}$.

Definition {19} *(Log-space many-to-one reducible).* We say that a set A is log-space many-to-one reducible to a set B, written $A \leq_m^{LS} B$, if there exists $\phi \in \boldsymbol{FLOGSPACE}$ such that for all $s \in \Gamma^*$, $s \in A \iff \phi(s) \in B$. We define $\leq_m^{LS}$-hardness and $\leq_m^{LS}$-completeness analogously.

Definition {20} *(Oracle Turing machine).* An oracle Turing machine is an ordinary Turing machine, with an extra tape, called the query tape, and two extra states, called the query state and answer state, and associated with a polynomially length-bounded function $\phi : \{0,1\}^* \to \{0,1\}^*$, i.e. $\ell(\phi(s)) \leq p(\ell(s))$ {22}. In query state, it reads the string $t \in \{0,1\}^*$ on the query tape, computes $\phi(t)$, and replaces t by $\phi(t)$ on query tape, and places the head of the Turing machine on the leftmost nonempty cell of the query tape.

{20} If M is a recognizer, we write $L(M, \phi)$ the set of string for which M^ϕ halts in accepting state.

{21} $\text{time}_{M^\phi}(s)$ counts all the regular steps of the Turing machine, but all the steps in the query state count only as one step. $\text{space}_{M^\phi}(s)$ counts the cell that the Turing machine visits, except the ones on the input, output and query tape. We define

$$t_M(\phi, n) := \max\{\mathrm{time}_{M^\phi}(s) : s \in \Gamma^n\}, \tag{5}$$
$$s_M(\phi, n) := \max\{\mathrm{space}_{M^\phi}(s) : s \in \Gamma^n\}. \tag{6}$$

An oracle TM is said to run in time (space) $\psi : \mathcal{N} \to \mathcal{N}$ if $t_M(\phi, \cdot) \le \psi(\cdot)$ for all ϕ.

Definition *D1.3* - {22} *(Turing reducibility).* A set A is polynomial-time Turing reducible to a set B, written $A \le_T^P B$, if there is a polynimial time oracle TM M which computes A when B is used as oracle.

Definition *D1.4* - {22} *(Hardness and completeness).* A set A is $\le_T^P$-hard for the class $\mathcal{C}$ if for every $B \in \mathcal{C}$, $B \le_T^P A$. A set A is $\le_T^P$-complete for the class $\mathcal{C}$, or alternatively $\mathcal{C}$-complete if $A \in \mathcal{C}$ and A is $\le_T^P B$-hard for the class $\mathcal{C}$.

Lemma *P1.5* - {22} *(Obvious statements).* (a) $A \le_m^P B \Rightarrow A \le_T^P B$.

(b) $\le_m^P$ −hardness implies $\le_T^P$ −hardness.

(c) If $A \le_T^P B$ and $B \in P$ then $A \in P$.

(d) If $A \le_m^P B$ and $B \in NP$ then $A \in NP$.

2.4 Polynomial hierarchy

{23} Definition of nondeterministic oracle Turing machines. We let $\boldsymbol{P}(\mathcal{C})$ to be the class of set that are accepted by polynomial-time deterministic oracle Turing machines using some $A \in \mathcal{C}$ as oracle. We define accordingly $\boldsymbol{NP}(\mathcal{C})$, $\boldsymbol{PSPACE}(\mathcal{C})$, etc. with the obvious meaning.

{23} Let $\mathcal{C}$ be a class of sets, we let co-C be the class of sets such that $\bar{A} \in \mathcal{C}$.

Definition *D1.6* - {23} *(Polynomial-time hierarchy).*

$$\Sigma_0^P = \Pi_0^P = \Delta_0^P = P \tag{7}$$
$$\Sigma_{k+1}^P = \boldsymbol{NP}(\Sigma_k^P) \tag{8}$$
$$\Pi_k^P = \text{co-}\Sigma_k^P \tag{9}$$
$$\Delta_{k+1}^P = \boldsymbol{P}(\Sigma_k^P) \tag{10}$$
$$\boldsymbol{PH} = \bigcup_{k=0}^{\infty} \Sigma_k^P. \tag{11}$$

Lemma *P1.7* - {23} *(First inclusions).*

$$\Sigma_k^P \cup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \cap \Pi_{k+1}^P \subseteq \boldsymbol{PSPACE}. \tag{12}$$

Theorem *T1.10* - {24} *(Formula for elements in Σ_k^P).* $A \in \Sigma_k^P \forall s \in A$ iff there exists a polynomial-time computable predicate R, and p a polynomial such that

$$s \in A \iff \exists t_1 \forall t_2 \exists t_3 \ldots \forall t_{k-1} \exists t_k R(s, t_1, t_2, \ldots, t_k), \tag{13}$$

with $\ell(t_1), \ell(t_2), \ldots, \ell(t_k) \le p(\ell(s))$.

2.5 Relativization

Definition {26} *(Relativized polynomial-time hierarchy).* Let A be a set. We define

$$\Sigma_0^P(A) = \Pi_0^P(A) = \Delta_0^P(A) = P(A) \tag{14}$$

$$\Sigma_{k+1}^P(A) = \boldsymbol{NP}(\Sigma_k^P(A)) \tag{15}$$

$$\Pi_k^P(A) = \text{co-}\Sigma_k^P(A) \tag{16}$$

$$\Delta_{k+1}^P(A) = \boldsymbol{P}(\Sigma_k^P(A)) \tag{17}$$

$$\boldsymbol{PH}(A) = \bigcup_{k=0}^{\infty} \Sigma_k^P(A). \tag{18}$$

See Theorem 1.12 for many solutions to P=NP and other similar problems, in relativized version.

{26} Let $\mathcal{S}$ be the set of all subsets of $\{0,1\}^*$, and $\Phi : \mathcal{S} \to \mathcal{S}$. We say that Φ is polynomial-time computable if there exists an polynomial-time oracle Turing machine M such that for each $A \in \mathcal{S}$, M^A accepts $\Phi(A)$. {27} We define analogously NP-computable, PSPACE-computable and Σ_k^P-computable operators. We denote each of these classes $\boldsymbol{P}_{op}$, $\boldsymbol{NP}_{op}$, $\boldsymbol{PSPACE}_{op}$ and $\Sigma_{k,op}^P$.

2.6 Probabilistic Complexity Classes

{28} A probabilistic TM is a TM that receives an input $s \in \{0,1\}^*$ and a sequence of coin flips $\alpha \in \{0,1\}^\infty$. The transfert function of the machine can sometimes have up to two choices available. In the case that the machine has a choice, it is made by reading the first bit of α. Then, this first bit is popped.

{28} We suppose that $\alpha \in \{0,1\}^\infty$ is generated according to the uniform distribution on $\{0,1\}^\infty$. We say that M accepts s if the probability of accepting s is greater than 1/2. $\boldsymbol{PP}$ is the set of sets accepted by polynomial-time probabilistic Turing machines. $\boldsymbol{BPP}$, $\boldsymbol{RP}$ and $\boldsymbol{ZPP}$ are on page {29}.

Many more results are on page {30}.

2.7 Complexity of counting

Definition *D1.18* - {31} *(Number of paths).* A function $\phi : \Gamma^* \to \mathcal{N}$ is in the class $\#P$ if there is a polynomial-time nondeterministic TM M such that for each $s \in \Gamma^*$, $\phi(s)$ is the number of accepting computations of $M(s)$.

Many results on that $\{31 - 32\}$.

2.8 One-way functions

Definition {32} *(One way function).* ϕ is polynomially honest if $\ell(s) \leq p(\ell(\phi(s)))$. A one way function is a polynomially honest functions that is polynomial time computable, one-to-one, and those all inverses is not polynomial-time computable.

Definition *D1.22*- {33} *(Unambiguous polynomial-time).* $A \in \boldsymbol{UP}$ if there is a polynomial-time nondeterministic Turing machine M computing A with exactly one accepting computation path. $\boldsymbol{P} \neq \boldsymbol{UP}$ is equivalent to the existence of one-way functions.

Relativized results on $\boldsymbol{UP}$ are discussed in page {34}.

2.9 Polynomial-size circuits and sparse sets

Definition *D1.28* - {36} *(Advice functions).* Let $\mathcal{C}$ be a class of sets and Φ be a class of functions $\Phi : \mathcal{N} \to \Gamma^*$. The class $\mathcal{C}/\Phi$ is the class of sets A for which there eixt a set $B \in \mathcal{C}$ and a function $\phi \in \Phi$ such that for all $s \in \Gamma^*$, $s \in A \iff \langle s, \phi(\ell(s)) \rangle \in B$.

Definition 1. {36}*[Sparse set, tally set] A set $S \subset \Gamma^*$ is sparse if there is a polynomial p such that $\#\{s \in S : \ell(s) = n\} \leq p(n)$. A tally set is a set $T \subset \{0\}^*$.*

Definition *P1.29* - {36} *(Characterisation of polynomial circuit sets).*

$$\boldsymbol{P}(SPARSE) = \boldsymbol{P}(TALLY) = \boldsymbol{P}/\,\text{poly} \tag{19}$$

Many final discussions on relativized P = NP, and sparse sets used as P = NP bridges are on pages $\{37 - 39\}$.

3 Computational Complexity of Real Functions

3.1 Computational complexity of real numbers

{41} We denote by $\mathcal{D}$ the dyadic numbers. $\ell(s)$ is the total length of a representation s of a dyadic number, and $\text{prec}(s)$ is the length after the dot.

Definition *D2.1* - {42} *(Representations of real numbers).*

(a) A function $\phi : \mathcal{N} \to \mathcal{D}$ is said to binary converge towards $x \in \mathbb{R}$ if $\prec (\phi(n)) = n$ and $|\phi(n) - x| \leq 2^{-n}$. CF_x is the set of all functions that binary converge towards x.

(b) $LC_x = \{d \in D | d < x\}$ is called the Dedekind left-cut of x.

(c) $BE_x : \mathcal{N} \to \{0, 1\}$ is such that

$$x = \sum_{i=1}^{\infty} BE_x(i) 2^{-i}. \tag{20}$$

{43} A real number x is computable $\iff$ there is a computable funcion in $CF_x \iff LC_x$ is computable $\iff BE_x$ is computable.

{44} Computable real numbers form a closed field. Equality testing is incomputable.

{47} We defined $\boldsymbol{P}_{CF}$, $\boldsymbol{P}_{BE}$, $\boldsymbol{P}_{LC}$ to be the polynomial-time complexity classes associated with the CF, BE and LC representations. It is shown that $P_{LC} = P_{BE} \underset{\neq}{\subseteq} P_{CF}$. {49} Moreover, P_{CF} is a closed field, whicle $P_{LC} = P_{BE}$ is not even closed under addition. So CF is the best representation to study real numbers complexity.

3.2 Computable real functions

Definition *D2.11* - {51} *(Computable real function).* Let $f : \mathbb{R} \to \mathbb{R}$. f is computable if there exists an oracle Turing machine M such that for all $x \in \mathbb{R}$ and all $\phi \in CF_x$, $n \mapsto M^{\phi}(0^n) \in CF_{f(x)}$. We can retrict the notion of computability on interval.

Definition *D2.12* - {52} *(Modulus functions).* Let $f : [a, b] \to \mathbb{R}$ be a continuous function. A function $m : \mathcal{N} \to \mathcal{N}$ is said to be a modulus function of f if for all $n \in \mathcal{N}$, and all $x, y \in [a, b]$, we have

$$|x - y| \leq 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}. \tag{21}$$

Theorem *T2.13* - {52} *(Uniform modulus theorem).* If $f : [a, b] \to \mathbb{R}$ is computable, then f is continuous and has a recursive modulus functions.

Theorem *C2.14* - {53} *(Characterization of Computable functions).* A real function $f : [a, b] \to \mathbb{R}$ is computable iff there exists two recursive functions $m : \mathcal{N} \to \mathcal{N}$ and $\psi : (\mathcal{D} \cap [a, b]) \times \mathcal{N} \to \mathcal{D}$ such that

(i) m is a modulus for f,

(ii) for all $d \in D \cap [a, b]$ and all $n \in \mathcal{N}$, $|\psi(d, n) - f(d)| \leq 2^{-n}$.

3.3 Complexity of real functions

Definition *D2.18* - {57} *(Time Complexity of Real functions).* Let G be a closed bounded interval or $\mathbb{R}$. Let M be an oracle Turing machine, and let

$$\text{time}'_M(x, n) := \max\{\text{time}_M(\phi, n) : \phi \in CF_x\}, \quad \forall x \in G, \forall n \in \mathcal{N}. \tag{22}$$

Let $f : G \to \mathbb{R}$ be a computable function. We say that the time complexity of f on G is bounded by a function $t : G \times \mathcal{N} \to \mathcal{N}$ if there exists an oracle TM M which computes f, such that for all $x \in G$ and all $n > 0$,

$$\text{time}'_M(x, n) \leq t(x, n). \tag{23}$$

We say that f has uniform time complexity $t'(n)$ if the time complexity $t(x, n)$ of f satisfies $t(x, n) \leq t'(n)$, for all $x \in G$. We say that f has nonuniform time complexity $t'(n)$ if the time complexity $t(x, n)$ of f satisfies $t(x, n) \leq t'(n)$, for all $x \in [-2^n, 2^n] \cap G$.

Definition *D2.18* - {58} *(Space Complexity of Real functions).* Let $f : G \to \mathbb{R}$ be a computable function. We say that the space complexity of f on G is bounded by a function $s : G \times \mathcal{N} \to \mathcal{N}$ if there exists an oracle TM M which computes f, such that for all $x \in G$ and all $n > 0$,

$$\text{space}_M(\phi, n) \leq s(x, n), \ \forall \phi \in CF_x. \tag{24}$$

We say that f has uniform space complexity $s'(n)$ if the time complexity $s(x, n)$ of f satisfies $s(x, n) \leq s'(n)$, for all $x \in G$.

We denote by $\boldsymbol{P}_{C[a,b]}$ the set of polynomial-time computable real functions.

Theorem *T2.19* - {58} *(Time complexity and modulus).* If $f : [a, b] \to \mathbb{R}$ has uniform time complexity $t(n)$, then $t(n + 2)$ is a modulus function for f.

Generalizations to multiple variables appear on pages $\{58 - 62\}$. Especially, the generalization implies that there is one oracle for each input variable. However, the characterization is NOT in terms of linear functions: it is a polynomial of degree 1.

3.4 Recursively Open sets

Definition *D2.29* - {63} *(Partial computable real functions).* Let $S \subseteq \mathbb{R}$, and $f : S \to \mathbb{R}$. f is said to be partial computable if there exist an oracle TM M such that

(i) for all $x \in S$, and all $\phi \in CF_x$, $M^\phi(n)$ halts and $|M^\phi(n) - f(x)| \leq 2^{-n}$, and

(ii) for all $x \notin S$ and all $\phi \in CF_x$, $M^\phi(n)$ does not halt for all n.

Definition *D2.30* - {63} *(Recursively open set).* A set $S \in \mathbb{R}$ is recursively open if $S = \varnothing$ or if there exists a recursive function $\phi : \mathcal{N} \to D$ such that

(i) for each $n \in \mathcal{N}$, $\phi(2n) < \phi(2n + 1)$, and

(ii) $S = \bigcup_{n=0}^{\infty} (\phi(2n), \phi(2n + 1))$,

A set S is recursively closed if $\mathbb{R} - S$ is open.

Theorem *T2.31* - {63} *(Characterization of recursively opne sets).* A set $S \subseteq \mathbb{R}$ is recursively open iff there is a partial computable function f whose domain is S

Lemma *C2.32* - {64} *(Continuity of partial computable functions).* Partial computable functions are continuous on their domain.

Definition *D2.33* - {64} *(r.e. numbers).* A real number x is left computable enumerable (left r.e.) if $(-\infty, x) \cap D$ is r.e., and a real number is right recursively enumerable (right r.e.) if $(x, \infty) \cap D$ is r.e.

Theorem *T2.34* - {65} *(Characterization of r.e.o. sets).* (a) If $x < y$, x is right r.e. and y is left r.e., then (x, y) is r.e.o.

(b) If $S \subset \mathbb{R}$ is a recursively open set, then for each component (x, y) of S (i.e., $(x, y) \subseteq S$ but $x \notin S$ and $y \notin S$), x is right r,e, and y is left r.e.

Definition *D2.35* - {67} *(Computable functional).* A numerical function F on $D \subset C[0, 1]$ is computable if there exists a TM M such that for all $f \in D$, given two oracle m and ϕ representing f,

$$|M^{m,\phi}(n) - F(f)| \leq 2^{-n}. \tag{25}$$

Definition *D2.35* - {67} *(Computable operator).* A numerical operator F on $D \subset C[0, 1]$ is computable if there exists a TM M such that for all f, and all $x \in [0, 1]$, given two oracle m and ϕ representing f, and ψ representing x, then

$$|M^{m,\phi,\psi}(n) - F(f)(x)| \leq 2^{-n}. \tag{26}$$

Definition *D2.35* - {67} *(Polynomial-time computable functional).* A computable function F is polynomial time if there exists p, q polynomials such that for all $f \in D$ represented by m, ϕ, M_F computes in time $p(m(0, 1, q(n)))$. We define similarly the complexity of operators.

4 Maximization

Theorem *T3.1* - {72} *(Maximal points of computable functions).* Let S be a nonempty subset of $[0, 1]$. Then, the following are equivalent

(a) S is recursively closed,

(b) there exists a computable function $f : [0, 1] \to \mathbb{R}$ whose maximum points are exactly S,

(c) there exists a polynomial-time computable function $f : [0, 1] \to \mathbb{R}$ whose maximum points are exactly S.

Lemma *C3.2/3* - {75} *(Recursiveness of maximum points).* Let f be a polynomial-time computable function on $[0, 1]$.

(a) f has at least oone left r.e. and one right r.e. maximum point,

(b) If x is an isolated maximum point of f, then x is recursive,

(c) If f has finitely many maximum points, they all are recursive,

(d) If f has countably infinitely many maximum points, then infinitely many of are recursive.

However, there exists a polynomial-time computable function f with uncountably many maximum points, all being nonrecursive.

Definition {76} *(Max function).* For $\phi : \{0,1\}^* \to \{0,1\}^*$, we define

$$\max_\phi(s) := \max\{\phi(\langle t, s\rangle) : \ell(t) = \ell(s)\}, \tag{27}$$
$$\max'_\phi(n) := \max\{\phi(\langle t, s\rangle) : \ell(t) = n\}. \tag{28}$$

Theorem *P3.4/5* - {76} *(NP-hardness of maximization).* The problem of computaing $\max_\phi$ is $\boldsymbol{NP}$-complete, and $\max'_\phi$ is $\boldsymbol{NP}_1$-hard.

In pages $\{81-99\}$, the authors establish the notion of NP real numbers, and make links with maximizing polynomial-time computable functions. In pages $\{100 - 105\}$, they establish a hierarchy of function classes by passing to the maximum and minimum, witch establishes a relation with the usual polynomial hierarchy.

References

Ker-I Ko. *Complexity Theory of Real Functions.* Birkhäuser, 1991.