Complexity Theory — Part 3: Feasibility

Dusko Pavlovic

RHUL
Spring 2012

Outline

Introduction: Gödel's letter

Time and space complexity

Abstract complexity

Reductions, hardness, completeness

Projections, provers and verifiers

Gödel’s letter

Context: Entscheidungsproblem

Hilbert (1928): Construct a machine \( \text{Prov} : \Sigma^* \to \Sigma^* \) to decide valid formulas:
\[
\text{Prov}(F) \iff \exists \Pi. \Pi \vdash F
\]

where \( F \) is a formula, and \( \Pi \) a proof.

Church (1935), Turing (1936): Entscheidungsproblem is undecidable: a machine \( \text{Prove} \) may search forever.

Gödel’s letter

Gödel’s idea

**Bounded Entscheidungsproblem:** Construct a machine
\[
\text{Prov} : \Sigma^* \times \mathbb{N} \to \Sigma^*
\]
\[
\text{Prov}(F, n) \iff \exists \Pi. \Pi \vdash F \land |\Pi| \leq n
\]

where \( F \) is a formula, and \( \Pi \) a proof of length \( |\Pi| \leq n \).

**Decidable:** Construct a TM to try all proofs of length \( \leq n \).
Gödel's letter

Gödel's idea

**Complexity of provability:** Define

\[
\psi(F, n) = |\text{run of Prov}(F, n)|
\]

\[
\phi(n) = \max\{\psi(F, n) : F \text{ is provable}\}
\]

**NP:** If \(\phi(n) = O(n^2)\), then machines can feasibly find proofs.

---

SAT ∈ NP

**Task:** Determine whether a propositional formula as SATisfiable.

---

SAT ∈ NP

SAT ∈ NP

---

SAT ∈ NP
SAT ∈ NP

**Time complexity**

**Definition**

For a program \( p \)

The time complexity measurement is \( \text{time}_p : \Sigma^* \rightarrow \mathbb{N} \)

**Idea**

The time requirement of a machine \( M \) for the input \( x \) is the length of its run

\[
\langle \text{inp}(x) \rangle \xrightarrow{T} \langle q_1, \tau^1 \rangle \xrightarrow{T} \langle q_2, \tau^2 \rangle \xrightarrow{T} \cdots \xrightarrow{T} \langle q_k, \tau^k \rangle
\]

where \( \text{time} = k \)

For nondeterministic \( M \), the time requirement is maximized over the possible paths.

**Abstract complexity**

Reductions, hardness, completeness

Projections, provers and verifiers

**Outline**

Introduction: Gödel’s letter

Time and space complexity

Definitions

Classes

Abstract complexity

Reductions, hardness, completeness

Projections, provers and verifiers
Space complexity is a computable function

A Turing machine that measures space:

\[ \text{space}_p(n) = \text{do for } |x| = n \]
\[ \text{run } U(m, x); \]
\[ \text{count steps}; \]
\[ \text{if } U(p, x) \text{ halts then} \]
\[ \text{store maximum}; \]
\[ \text{od} \]

Space complexity

Idea

- The space requirement of a machine \( M \) for the input \( x \) is the greatest span that the head reaches

\[
\begin{array}{ccccccccccc}
m & 0 & & & & & & & & n \\
\hline
| & | & | & | & | & | & | & | & | \\
\hline
0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
\]

space = m + n

- For nondeterministic \( M \), the space requirement is maximized over the possible paths.

Definition

For a program \( p \)

- the space complexity measure is \( \text{space}_p : \{x\} \rightarrow \mathbb{N} \)

\[ \text{space}_p(n) = \bigvee_{x} \{ r \mid \exists y. T(p, x, y, r) \} \]

where \( r \) is the span of the head in the run \( r \).

Space complexity is a computable function

A Turing machine that measures space:

\[ \text{space}_p(n) = \text{do for } |x| = n \]
\[ \text{run } U(m, x); \]
\[ \text{record the offsets}; \]
\[ \text{if } U(p, x) \text{ halts then} \]
\[ \text{add up the offsets}; \]
\[ \text{store maximum}; \]
\[ \text{od} \]
**Complexity classes**

**Definition**

For any $f : \mathbb{N} \to \mathbb{N}$ define

- $\text{DTIME}(f) = \{ L(M) \subseteq \Sigma^* | M \in \text{DTM} \cdot \text{time}M^1 \leq f \}$
- $\text{NTIME}(f) = \{ L(M) \subseteq \Sigma^* | M \in \text{NTM} \cdot \text{time}M^1 \leq f \}$
- $\text{DSPACE}(f) = \{ L(M) \subseteq \Sigma^* | M \in \text{DTM} \cdot \text{space}M^1 \leq f \}$
- $\text{NSPACE}(f) = \{ L(M) \subseteq \Sigma^* | M \in \text{NTM} \cdot \text{space}M^1 \leq f \}$

**Preorder of functions**

**Notation**

$f \leq g \iff \forall x : f(x) \leq g(x)$

**Complexity classes**

**Two views of time-bounded machines**

1. TM that normally reach a decision in $\leq f(n)$ steps
   - approach: try to construct a machine that solves the problem in time
2. TM with a counter that clocks out after $f(n)$ steps
   - approach: consider all machines that solve the problem, find out which ones reach the decision in time

**Facts.**

- obvious:
  - $\text{DTIME}(f) \subseteq \text{NTIME}(f)$
  - $\text{DTIME}(f) \subseteq \text{NTIME}(f)$

  "you cannot use more space than time"

- $\text{DTIME}(f) \subseteq \text{DSPACE}(f)$
- $\text{NTIME}(f) \subseteq \text{NSPACE}(f)$
Proposition 3

For $f \geq \log$

$\text{NSPACE}(f) \subseteq \text{DTIME}(2^{O(f)})$

Proposition 4 (Savich)

For $f \geq \log$

$\text{NSPACE}(f) \subseteq \text{DSPACE}(f^2)$

Proposition 5 (Immerman-Szelepcsényi)

For $f \geq \log$

$\text{NSPACE}(f) = \text{co-NSPACE}(f)$

Computational complexity

- Compute computational complexity
  - Machines that measure complexity of machines
**Complexity measure**

**Definition**

An abstract complexity requirement is a partial function \( c : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \) such that

- \( \overline{c}(\overline{M^n}, x) \leq M(x) \)
- \( c(m, x) \leq k \) is decidable
- \( \overline{c}(\overline{M^n}, x) \leq \overline{c}(\overline{M^n}, M(x)) + \overline{c}(\overline{M^n}, x) \)
- \( \overline{c}(\overline{M^n}, (x; y)) \leq \overline{c}(\overline{M^n}, x) + \overline{c}(\overline{M^n}, y) \)
- \( \overline{c}(\text{read}^n, x) = |x| \)

**Complexity measure**

**Definition (continued)**

The abstract complexity measure associated with \( c \) is a function \( c_m : \mathbb{N} \rightarrow \mathbb{P} \) which maps the program \( m \in \mathbb{N} \) to the induced partial function

\[
c_m : n \mapsto \bigvee_{|x| = n} \overline{c}(m, x)
\]

**Nonexamples**

- \( \overline{c}(m, x) = \overline{u}(m, x) \) is not a complexity measure because the predicate \( U(m, x) = k \) is not decidable.
- \( \overline{c}(m, x) = |m| \cdot |x| \) is not a complexity measure because it is defined even when \( |m| \cdot |x| \) is not.

**The additivity induces the polynomial bounds**

- If \( p[q] \) denotes a program \( p \) that calls a procedure \( q \) then
  \[
c_{p[q]}(n) \leq c_p(n) \cdot c_q(n)
\]
  follows from the additivity of \( c \), because \( p \) can call \( q \) at most \( c_p(n) \) times.
- The exponential growth of complexity occurs when each procedure call makes further procedure calls, like in the exhaustive search algorithms.
Abstract complexity classes

Definition

For any complexity measure $c$ define

$$DCLASS_c(f) = \{ L(M) \subseteq \Sigma^* | \text{DTM } M \text{ s.t. } cr_{m^i} \leq f \}$$

$$NCLASS_c(f) = \{ L(M) \subseteq \Sigma^* | \text{NTM } M \text{ s.t. } cr_{m^i} \leq f \}$$

$$P_c = \bigcup_{k \geq 0} DCLASS_c(n^k)$$

$$NP_c = \bigcup_{k \geq 0} NCLASS_c(n^k)$$

Concrete complexity classes

Definition

$$P = P_{\text{time}} = \bigcup_{k \geq 0} \text{DTIME}(n^k)$$

$$NP = NP_{\text{time}} = \bigcup_{k \geq 0} \text{NTIME}(n^k)$$

$$PSPACE = P_{\text{space}} = \bigcup_{k \geq 0} \text{DSPACE}(n^k)$$

$$NPSPACE = NP_{\text{space}} = \bigcup_{k \geq 0} \text{NSPACE}(n^k)$$

Remarks

- Props. 4 and 5 imply
  $$PSPACE = NPSPACE = \text{co-NPSPACE}$$

- The quest for $$P \neq NP$$ is ongoing.

The hierarchy of complexity classes

Recall for $RE$ and $REC$

For

$$K(x) = \begin{cases} 1 & \text{if } U(x) \downarrow \\ 0 & \text{otherwise} \end{cases}$$

we proved $$K \in RE \setminus REC$$

Proof of $K_f \notin \text{CLASS}_C(f)$

Define

$$\bar{K}_f(x) = \begin{cases} 1 & \text{if } K_f(x) = 0 \\ \uparrow & \text{if } K_f(x) = 1 \end{cases}$$

and note

$$K_f(\bar{K}_f) \iff c(\bar{K}_f) \leq f(\bar{K}_f)$$

$$\Rightarrow c(\bar{K}_f) \leq f(\bar{K}_f)$$

$$\Rightarrow \bar{K}_f(\bar{K}_f) \downarrow$$

$$\Rightarrow \bar{K}_f(\bar{K}_f) \iff \neg K_f(\bar{K}_f)$$
**Proof of \( \mathcal{K}_r \notin \text{CLASS}_C(f) \)**

Hence

\[
\mathcal{K}_r^c(\mathcal{K}_r^c) \Rightarrow -\mathcal{K}_r^c(\mathcal{K}_r^c)
\]

\[
-\mathcal{K}_r^c(\mathcal{K}_r^c)
\]

\[
c(\mathcal{K}_r^c, \mathcal{K}_r^c) > f(\mathcal{K}_r^c)
\]

\[
\mathcal{K}_r \notin \text{CLASS}_C(f)
\]

---

**The Hierarchy Theorem**

**Theorem**

For every function \( f : \mathbb{N} \rightarrow \mathbb{N} \) and every complexity measure \( c \) there is \( k \geq 1 \) and a language

\[
\mathcal{K}_r \in \text{CLASS}_C(O(f \cdot \log^k f)) \setminus \text{CLASS}_C(O(f))
\]

---

**The Hierarchy Theorem**

**Corollary**

For every \( f \) and \( c \) there is some \( k \geq 1 \)

- \( \text{CLASS}_C(O(f)) \subsetneq \text{CLASS}_C(O(f^{k+1})) \)
- \( \text{CLASS}_C(O(\log^k)) \subsetneq \text{P}_C \subseteq \text{CLASS}_C(O(\exp)) \)

---

**Complexity bounds**

**Question**

Which other functions besides polynomials, logarithms and exponentials are meaningful as complexity bounds?

**Answer**

- Pathology: Gap theorem
- Health: \( c \)-costructible functions = functions for which \( c \) is the universal machine.

---

**Complexity bounds**

**Gap Theorem**

For every complexity measure \( c \) there is a recursive function \( r : \mathbb{N} \rightarrow \mathbb{N} \) such that

\[
c_M(r(x)) > r(\lceil x \rceil) \Rightarrow c_M^r(x) > 2^{\lceil x \rceil}
\]

for all \( M \in \text{TM} \) and all but finitely many \( x \).

Hence

\[
\text{CLASS}_C(r(\lceil x \rceil)) \supsetneq \text{CLASS}_C(2^{\lceil x \rceil})
\]
A function $f : \mathbb{N} \to \mathbb{N}$ is $c$-constructible if there is a function $F \in \text{DTM}$ such that $f(|x|) = c(|F|^x, x)$, for all $x \in \Sigma^*$.

1 can always assume multitape.
### Reductions

**Definition**

Let $A, B, R \in \mathcal{T}M$. We say that $A$ is reducible to $B$ by $R$ and write

$$A \xrightarrow{R} B$$

if for all inputs $x \in \Sigma^*$ holds

1. $A(x) \downarrow \iff B(R(x)) \downarrow$
2. $A(x) \downarrow \implies A(x) = B(R(x))$

### Properties of $\xrightarrow{\_}$

**Proposition 6**

(a) preordering

$$A \xrightarrow{C} B \quad \text{and} \quad A \xrightarrow{C} B \land B \xrightarrow{C} D \implies A \xrightarrow{C} D$$

(b) lower closed in $C$

$$A \xrightarrow{C} B \land B \in C \implies A \in C$$

---

**Examples**

Let $D \subseteq \mathcal{T}M$ be any class containing constants and identities and $C$ a complexity measure. Then

- $M \xrightarrow{D} U$ for all $M \in \mathcal{T}M$
- $F \xrightarrow{D} C$ iff $F$ is a $C$-constructible function.
Equivalence

Proposition 7
Any two complexity measures $C$ and $D$ are recursively equivalent:

$$ C \overset{\text{REC}}{\longleftrightarrow} D $$

Hardness and completeness

Definition
Let $B \in \wp(\Sigma^*)$ and $\mathcal{L}, C \subseteq \wp(\Sigma^*)$. We say that $B$ is $\mathcal{L}$-hard over $C$ if

$$ X \in \mathcal{L} \implies X \rightarrow_C B $$

We write $\mathcal{L} \rightarrow_C B$.

Examples
Let $\mathcal{D}$ be any class of languages containing constants and identities (i.e. the diagonal languages $\{\langle x, x \rangle\}$) and $C$ a complexity class.

- The halting language $H = \{\langle p, x \rangle \mid U(p, x) \downarrow\}$ is $RE$-complete over $\mathcal{D}$.
- The language consisting of programs in some $C$-bounded class is complete over $\mathcal{D}$ for the family of $C$-constructible functions.

Proposition 8
If $C \subseteq \wp(\Sigma^*)$ is closed under composition, i.e.

$$ M, N \in C \implies M \circ N \in C $$

then

$$ \exists B \in \mathcal{C}, \mathcal{L} \rightarrow_C B \implies \mathcal{L} \subseteq C $$

Corollary
$$ \exists B \in P, NP \rightarrow_P B \implies NP = P $$
**Hardness and completeness**

**Trouble**

If $C \subseteq \text{REC}$ is closed under composition and conditionals, i.e.

$$M, N \in C \implies M \circ N \in C$$

and

$$(\text{if } M \text{ then } N) \in C$$

then every nonempty $B \subseteq \Sigma^*$ is $C$-hard over $C$.

$$C \rightarrow B$$

(All main classes are closed under composition and conditionals.)

---

**Moral**

- Complexity classes cannot be meaningfully studied in terms of reducibility over themselves, but
- over a smaller class, say $\mathcal{L} = \text{DSPACE}(\log)$

---

**Convention**

By default

$$A \rightarrow B \text{ means } A \rightarrow B$$

---

**NP-completeness**

**Cook-Levin Theorem**

$$\text{NP} \rightarrow \text{SAT}$$
Complexity 3: Feasibility
Dusko Pavlovic
Introduction
Time and space
Abstract complexity
Reducibility
Summary
P vs NP

NP-completeness

Recall: Projections

Definition

For any family of languages $\mathcal{FAM} \subseteq \mathcal{P}(\Sigma^* \times \Sigma^*)$ the projection family is defined

$$\exists \mathcal{FAM} = \{ \mathcal{L}_\exists \subseteq \Sigma^* | \mathcal{L} \in \mathcal{FAM} \}$$

where $\mathcal{L}_\exists = \{ y \in \Sigma^* | \exists x. (x, y) \in \mathcal{L} \}$.

Recall: $RE = \exists REC$

Proof of $\exists REC \subseteq RE$

$$M(y) = \mu x. 1 - M(x, y) = 0$$

Recall: $RE = \exists REC$

Proof of $RE \subseteq \exists REC$

Kleene's recursive predicate $T : \Sigma^* \rightarrow \{0, 1\}$

$$U(m, x) = y \iff \exists r. T(m, x, y, r)$$

Recall: $RE = \exists REC$

Proof of $\exists REC \subseteq RE$

$$M(y) = \mu x. 1 - M(x, y) = 0$$

Complexity 3: Feasibility
Dusko Pavlovic
Introduction
Time and space
Abstract complexity
Reducibility
Summary
P vs NP

Recall: Projections

Outline

Introduction: Gödel's letter

Time and space complexity

Abstract complexity

Reductions, hardness, completeness

Projections, provers and verifiers

Projections

coNP

Summary
Polynomial projections

Definition

For any family of languages $FAM \subseteq \mathcal{P}(\Sigma^* \times \Sigma^*)$ the projection family is defined

$$\exists_q FAM = \{ L \exists \subseteq \Sigma^* | L \in FAM \land q \in \mathbb{Z}[n] \}$$

where $L \exists = \{ y \in \Sigma^* | \exists x, |x| \leq q(|y|) \land \langle x, y \rangle \in L \}$. 

NP = $\exists_q P$

Proof of $\exists_q P \subseteq NP$

Interpretation

Why is $P \subseteq NP$ harder than $REC \subseteq RE$?

$$RE = REC$$

$$NP = P$$

$\exists X \in 3REC \setminus REC$ $\exists X \in 3P \setminus P$
Why is $P \subseteq NP$ harder than $REC \subseteq RE$?

- $K \in \exists REC \setminus REC$
  - does not show that finding witnesses is hard
  - shows the decider of halting may not halt
    - $\exists$ says that there is a halting run

- $X \in \exists P \setminus P$
  - must show that finding witnesses is hard
  - all NP deciders halt eventually
    - $\exists$ says that they halt in poly time

---

$RE = \exists REC$

$B \in RE$

$\exists$ there is $A \in REC. (y \in B \iff \exists x \in \Sigma^*. \langle x, y \rangle \in A)$

---

$coRE = \forall REC$

$B \in coRE$

$\forall$ there is $A \in REC. (y \in B \iff \forall x \in \Sigma^*. \langle x, y \rangle \in A)$

---

$NP = \exists_p P$

$B \in coNP$

$\exists$ there are $A, q \in P. (y \in B \iff \exists x \in \Sigma^*. |x| \leq q(|y|) \Downarrow \langle x, y \rangle \in A)$

---

$coNP = \forall_p P$

$B \in coNP$

$\forall$ there are $A, q \in P. (y \in B \iff \forall x \in \Sigma^*. |x| \leq q(|y|) \Downarrow \langle x, y \rangle \in A)$
**NP ∩ coNP**

- although $RE \cap coRE = REC$
  - if $y \in L \in RE$ then $M_L(y)$ will halt and prove $y \in L$
  - if $y \notin L \in coRE$ then $M_L(y)$ will halt and prove $y \notin L$

- it is believed that $NP \cap coNP \neq P$, because having
  - a family of programs $A$ where $y \in B \Longleftrightarrow$ some $x$ satisfies it with $(x, y) \in A$, and
  - a family of programs $C$ where $y \in B \Longleftrightarrow$ every $z$ satisfies it with $(z, y) \in C$

  does not yield feasible way to eliminate search for building $x$ and eliminating $z$.

**Interpretation**

- $NP \ni B = \text{set of satisfiable properties}$
  - if $(x, y) \in A$ means "program $x$ satisfies $y$" then for every property $y \in B$ there is some program $x$ in $A$ that satisfies $y$
    - $NP \neq P$ because it is hard to find it.

- $coNP \ni B = \text{set of satisfied properties}$
  - if $(x, y) \in A$ means "program $x$ satisfies $y"$
    - then for every property $y \in B$ all programs $x$ in $A$ satisfy $y$
    - $coNP \neq P$ because it is hard to prove that

**The Inner Eggs**