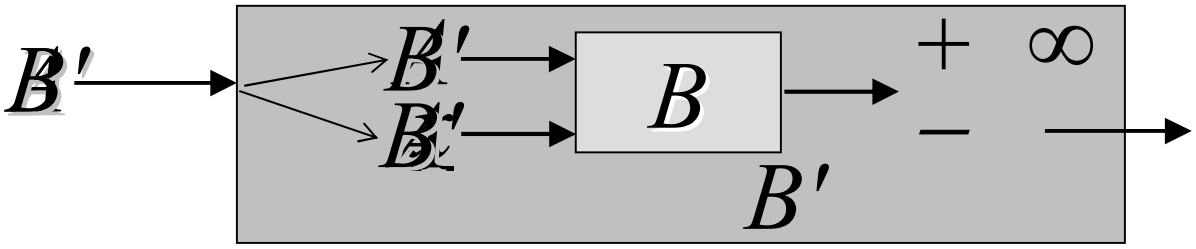
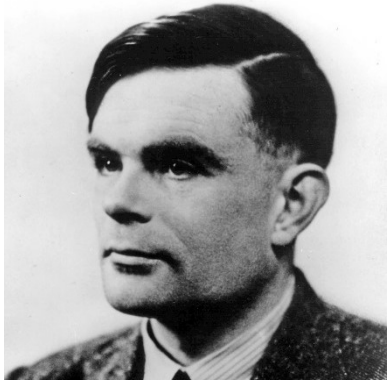
Alan M. Turing 1936

KAIST
CS422 M. Ziegler

- first scientific calculations on digital computers
- *What are its fundamental limitations?*





- Undecidable Halting Problem *H*: **No** algorithm *B* can always correctly ans ~~simulator/interpreter *B* ?~~
  Given ⟨A,x⟩, does algorithm A terminate on input x?

Proof by contradiction: Consider algorithm *B*' that, on input *A*, executes *B* on ⟨*A*,*A*⟩ and, upon a positive answer, loops infinitely. How does *B*' behave on *B*′ ?

Theory of Computation

KAIST
CS422 M. Ziegler

References

- Papadimitriou: Computational Complexity, Addison Wesley (1993)
- Moore, Mertens: The Nature of Computation (2011)
- Lewis, Papadimitrou: Elements of the Theory of Computation, Prentice-Hall (1997).
- Arora, Barak: Computational Complexity - A Modern Approach, Cambridge University Press (2009).
- Sipser: Introduction to the Theory of Computation, PWS Publishing (1997).
- Enderton: Computability Theory (2011).

Course page **`http://kaist.theoryofcomputation.asia/21cs422`**

E-Learning `http://klms.kaist.ac.kr/`

No email requests, please!

Homework: weekly assignments, individually, hand-written; random grading

Pledge for *integrity and academic honesty* to be signed and submitted!

Overview

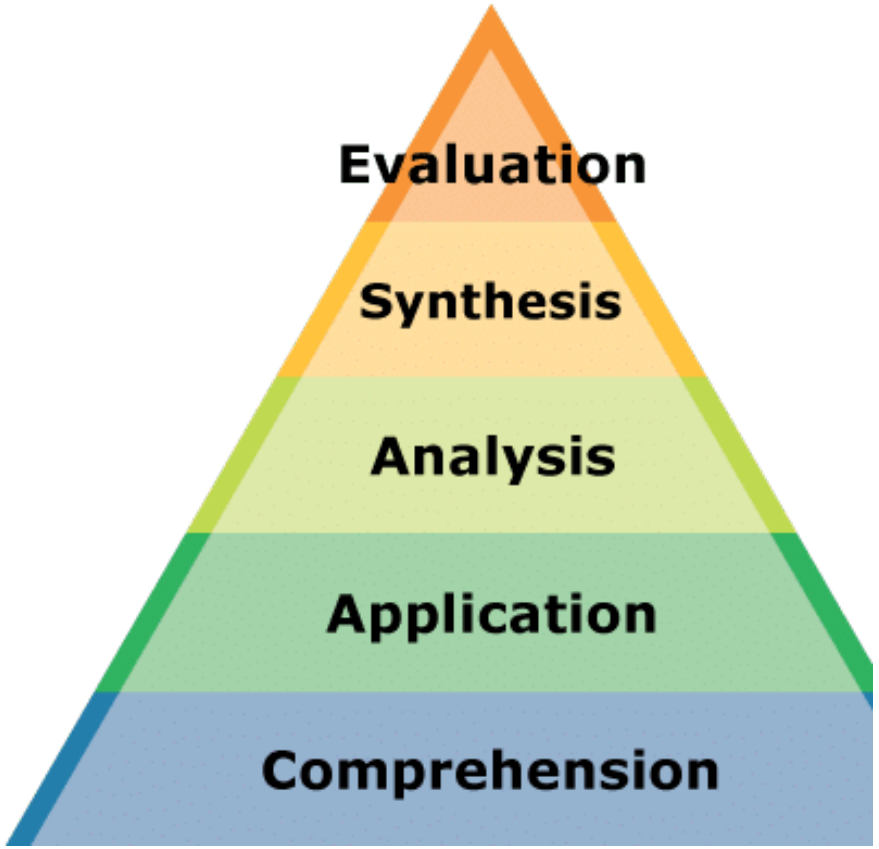
Motivation

- Basic Computability
- Advanced Computability
- Computational Complexity
- Structural Complexity: $\mathcal{NP}$c
- PSPACE and Polynomial Hierarchy
- Advanced Complexity

Conclusion

Pedagogical Concept

This course focuses
on *Comprehension*,
Application, and *Analysis*
and takes for granted
your ability to amass
(or look up) *Knowledge*!



Bloom's Taxonomy

§1 Basic Computability Theory

- Computability, semi-/decidability, enumerability
- Examples of undecidable problems
- Reduction: comparing problems
- Busy Beaver/Rado function
- Ackermann function
- LOOP programs

Un-/Semi-/Decidability I

Definition: a) An 'algorithm' $\mathcal{A}$ computes a partial function $f :\subseteq \mathbb{N} \to \mathbb{N}$ if it

- on inputs $x \in \mathrm{dom}(f)$ prints $f(x)$ and terminates,
- on inputs $x \notin \mathrm{dom}(f)$ does not terminate.

Injective pairing function ("*Hilbert Hotel*")

$$\langle x,y \rangle := x + (x+y)\cdot(x+y+1)/2$$

b) $\mathcal{A}$ decides set $L \subseteq \mathbb{N}$ if it computes its total char. function: $\mathrm{cf}_L(x) := 1$ for $x \in L$, $\mathrm{cf}_L(x) := 0$ for $x \notin L$.

c) $\mathcal{A}$ semi-decides L if terminates precisely on $x \in L$

d) $\mathcal{A}$ enumerates L if it computes some total injective $f:\mathbb{N}\to\mathbb{N}$ with $L = \mathrm{range}(f)$.

Un-/Semi-/Decidability II

Example: The Halting problem H, considered as subset of $\mathbb{N}$, is semi-decidable, not decidable.

Theorem: a) Every finite L is decidable.

b) L is decidable iff its complement $\overline{L}$ is.

c) L is decidable iff both $L, \overline{L}$ are semi-decidable.

d) L is enumerable iff infinite and semi-decidable

b) $\mathcal{A}$ decides set $L \subseteq \mathbb{N}$ if it computes its total char. function: $\mathrm{cf}_L(x) := 1$ for $x \in L$, $\mathrm{cf}_L(x) := 0$ for $x \notin L$.

c) $\mathcal{A}$ semi-decides L if terminates precisely on $x \in L$

d) $\mathcal{A}$ enumerates L if it computes some total injective $f : \mathbb{N} \to \mathbb{N}$ with $L = \mathrm{range}(f)$.

Proof: „enumerability = semi-decidability“

Theorem:

d) L is enumerable iff infinite and semi-decidable

"$\Rightarrow$": Infinite $\sqrt{}$. Given x,
compute and test "$f(0)=x$?", "$f(1)=x$?", "$f(2)=x$?" ...

"$\Leftarrow$": Let $\mathcal{A}$ semi-decide L. Fix $w \in L$.

On input $\langle x,m \rangle$ let $\mathcal{B}$ simulate and see whether $\mathcal{A}$ accepts x within m steps.

If so let $\mathcal{B}$ output x, otherwise output w.

not injective!

Def: d) $\mathcal{A}$ enumerates L if it computes some total injective $f:\mathbb{N}\to\mathbb{N}$ with $L=\mathrm{range}(f)$.

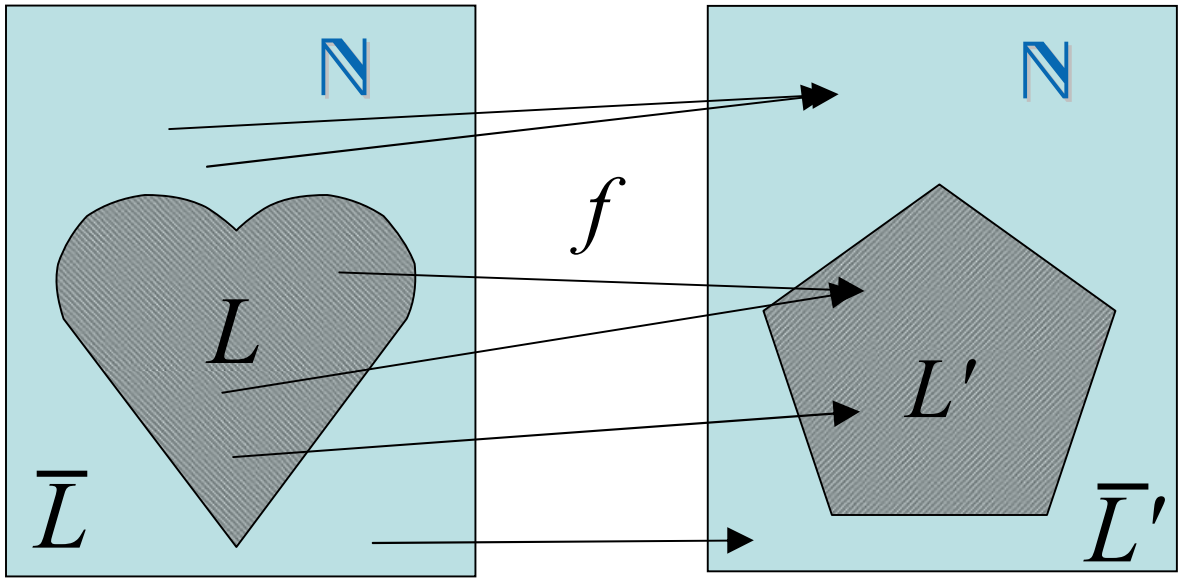
c) $\mathcal{A}$ semi-decides L if terminates precisely on $x \in L$

Comparing Decision Problems

Halting problem $H = \{ \langle \mathcal{A}, x \rangle : \mathcal{A}(x) \text{ terminates} \}$

Nontriviality $N = \{ \langle \mathcal{A} \rangle : \exists y\ \mathcal{A}(y) \text{ terminates} \}$

Totality problem $T = \{ \langle \mathcal{A} \rangle : \forall z\ \mathcal{A}(z) \text{ terminates} \}$



- $H \leqslant N$ undecidable
- $H \leqslant T$ undecidable
- $N \leqslant H \leqslant \overline{H}$
- $\overline{H} \leqslant T, \Rightarrow T \leqslant H$

For $L, L' \subseteq \mathbb{N}$ write $L \leqslant L'$ if there is a computable $f: \mathbb{N} \to \mathbb{N}$ such that $\forall x: x \in L \Leftrightarrow f(x) \in L'$.

a) $\overline{L'}$ semi-/decidable $\Rightarrow$ so $\overline{L}$. b) $L \leqslant L' \leqslant L'' \Rightarrow L \leqslant L''$

Comparing Decision Problems

KAIST
CS422 M. Ziegler

Halting problem $H = \{ \langle \mathcal{A}, x \rangle : \mathcal{A}(x) \text{ terminates} \}$

Nontriviality $N = \{ \langle \mathcal{A} \rangle : \exists y\ \mathcal{A}(y) \text{ terminates} \}$

Totality problem $T = \{ \langle \mathcal{A} \rangle : \forall z\ \mathcal{A}(z) \text{ terminates} \}$

Proof $\overline{H} \leqslant T$**:** Given $\langle \mathcal{A}, x \rangle$, convert to $\mathcal{B}$ which on input m simulates $\mathcal{A}(x)$ for m steps and terminates if $\mathcal{A}(x)$ does not terminate within m steps, otherwise deliberately `FREEZE`s.

- $H \leqslant N$ undecidable
- $H \leqslant T$ undecidable
- $N \leqslant H \not\leqslant \overline{H}$
- $\overline{H} \leqslant T, \Rightarrow T \not\leqslant H$

For $L, L' \subseteq \mathbb{N}$ write $L \leqslant L'$ if there is a computable $f: \mathbb{N} \to \mathbb{N}$ such that $\forall x: x \in L \Leftrightarrow f(x) \in L'$.

a) $\overline{L'}$ semi-/decidable $\Rightarrow$ so $\overline{L}$. b) $L \leqslant L' \leqslant L'' \Rightarrow L \leqslant L''$

Approximating the Halting Problem

Algorithm to answer "$\langle \mathcal{A} \rangle \in \hat{H}$?" always and

i) Correct *finitely* often → *Trivially* possible!

ii) Correct *in*finitely often → *Trivially* possible!

*iii) In*correct only *finitely* often → Impossible!

iv) Never *in*correct → Impossible!

Not every integer encodes an algorithm:

$\mathcal{A} \to \langle \mathcal{A} \rangle \in \mathbb{N}$ is not surjective.

But: $\mathrm{range}(\langle\rangle) \subseteq \mathbb{N}$ is decidable!

"Missing" syntactically *in*correct source codes...

$\hat{H} := \{ \langle \mathcal{A} \rangle : \mathcal{A}$ terminates on input $0 \} \equiv H$

Undecidable!

Busy Beaver (aka Rado Function)

For algorithm $\mathcal{A}$ let $t(\langle\mathcal{A}\rangle) := \#\text{steps } \mathcal{A} \text{ makes}$ (on input 0) before terminating, $:=\infty$ if doesn't.

$$\beta(n) := \max\{\, t(m) : m \leq n,\ t(m) < \infty \,\}$$ Busy Beaver

Theorem: Every computable $f:\mathbb{N}\to\mathbb{N}$ satisfies $f(m) < \beta(m)$ for some (infinitely many) $m \in \mathbb{N}$.

Proof: The following algorithm answers "$\langle\mathcal{A}\rangle \in \hat{H}$ **?**" whenever $f(\langle\mathcal{A}\rangle) \geq \beta(\langle\mathcal{A}\rangle)$:

Compute $N := f(\langle\mathcal{A}\rangle)$. Simulate $\mathcal{A}$ (on input 0) for N steps.

If not yet terminated, answer ***no***.

$\hat{H} := \{\, \langle\mathcal{A}\rangle : \mathcal{A}$ terminates on input $0 \,\} \equiv H$

Ackermann's Function

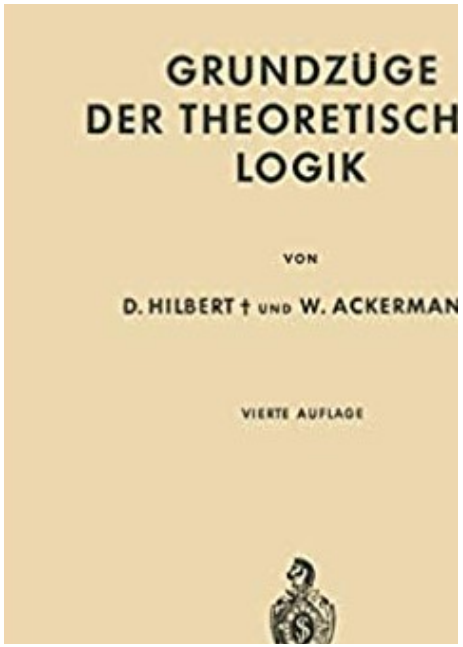
$A(0,n) = n+2$ $\qquad$ $A(\ell+1,0)=A(\ell,1)$

$A(\ell+1,n+1) = A(\ell,A(\ell+1,n))$

- $A(1,n) = 2n+3$
- $A(2,n) = \text{expon.}$
- $A(3,n) = \text{tetration}$
- $A(4,n) = ?$

computable!

Diagonal Ackermann: $n \to A(n,n)$



LOOP Programs

Syntax in Backus—Naur Form:

$$P := (\ x_j := 0 \mid x_j := x_k \mid x_k\,{+}{+}\ \mid\ P\,;P \mid \textbf{FREEZE}\ \mid$$
$$\text{LOOP } x_j \text{ DO } P \text{ END }) \qquad \in\mathbb{N}$$

Semantics:

- $x_1,\ldots x_k$ contain input $\in\mathbb{N}^k$
- LOOP executed x_j times
- Body must not change x_j

Example: simulate

"$x_j := \max(0, x_i - 1)$" :

$x_j := 0\ ;\ x_k := 0\ ;$
LOOP x_i DO
  $x_j := x_k\ ;\ x_k\,{+}{+}$
END

Example: simulate

"**IF** $x_j \neq 0$ **THEN** P **ELSE** Q"

$x_k := 0$; LOOP x_j DO $x_k := 1$ END ; $x_\ell := 1$;
LOOP x_k DO P ; $x_\ell := 0$ END ; LOOP x_ℓ DO Q END

Capabilities of LOOP Programs

Examples: simulate addition "$x_k := x_k + x_j$"

LOOP x_j DO x_k++ END

Example: simulate "$x_j := \max(0, x_i - 1)$" :

$x_j := 0$; $x_k := 0$;
LOOP x_i DO
  $x_j := x_k$; $x_k := x_k + 1$
END

Simulate multiplication "$x_k := x_j \times x_i$"

$x_k := 0$; LOOP x_i DO $x_k := x_k + x_j$ END

Power of LOOP Programs

Def: Recall bijective $\mathbb{N}^2 \ni (x,y) \to \langle x,y \rangle \in \mathbb{N}$

and write $\langle x,y,z \rangle := \langle \langle x,y \rangle, z \rangle$, $\langle x,y,z,w \rangle := \langle \langle x,y,z \rangle, w \rangle$ etc.

Lemma: a) There exists a LOOP program that, given $x,y \in \mathbb{N}$, returns $\langle x,y \rangle \in \mathbb{N}$.

b) There exists a LOOP program that, given $\langle x,y \rangle \in \mathbb{N}$, returns x; another one returns $y \in \mathbb{N}$.

c) There exists a LOOP program that, given integers $n \leq N$ and $\langle x_1, x_2, \ldots, x_n, \ldots, x_N \rangle$, returns x_n.

d) There exists a LOOP program that, given $n \leq N$ and y and $\langle x_1, x_2, \ldots, x_n \ldots, x_N \rangle$, returns $\langle x_1, x_2, \ldots, y, \ldots, x_N \rangle$.

Array of integers with indirect addressing

Limitations of LOOP Programs

Theorem: • To every **LOOP** program $P=P(x_1,\ldots x_k)$ there exists some $\ell=\ell(P)\in\mathbb{N}$ s.t. P on input $\underline{x}\in\mathbb{N}^k$ makes $\leq A(\ell,|\underline{x}|+\ell)<\infty$ steps, where $|\underline{x}| := \max_j x_j$

• $A(n,n)$ is not computable by any **LOOP** program!

Now recall **Ackerman's function:** $A(0,n)=2+n$

$$A(\ell+1,0) = A(\ell,1) \qquad A(\ell+1,n+1) = A(\ell,A(\ell+1,n))$$

$$\Rightarrow \quad A(1,n) = 2n+3 \qquad A(2,n) = 2^{n+3}-3$$

Recall: **LOOP** can realize addition "$x_k := x_j + x_i$"

LOOP can realize multiplication "$x_k := x_j \times x_i$"

Limitations of LOOP Programs

Theorem: • To every **LOOP** program $P=P(x_1,\ldots x_k)$ there exists some $\ell=\ell(P)\in\mathbb{N}$ s.t. P on input $\underline{x}\in\mathbb{N}^k$ makes $\leq A(\ell,|\underline{x}|+\ell)<\infty$ steps, where $|\underline{x}| := \max_j x_j$

• $A(n,n)$ is not computable by any **LOOP** program!

Corollary: The Halting Problem for LOOP Programs

a) is decidable!

b) but *not* decidable by LOOP programs.

LOOP programs *in*sufficient to formalize algorithms.

Instead: WHILE programs in §2.

§1 Basic Computability Theory

- Computability, semi-/decidability, enumerability
- Examples of undecidable problems
- Reduction: comparing problems
- Busy Beaver/Rado function
- Ackermann function
- LOOP programs

§1 Quiz: Ackermann Function

$$A(0,n) = n+2 \qquad A(\ell+1,0)=A(\ell,1)$$

$$A(\ell+1,n+1) = A(\ell,A(\ell+1,n))$$

Prove by induction:

- $A(1,n) = 2n+3$
- $A(2,n) = 2^{n+3}-3$
- $A(3,n) = ?$
- $A(4,n) = ?$