# UCT

# Oracles and Reductions

Klaus Sutner

Carnegie Mellon University

Spring 2024

# Separation



Note that the arithmetical hierarchy is cumulative, $\Sigma_k \subseteq \Sigma_{k+1}$.

How would we go about proving $\Sigma_k \subsetneq \Sigma_{k+1}$?

> **General Problem:** We have two complexity classes $\mathcal{C}_1 \subseteq \mathcal{C}_2$. We want to separate the two classes: show $\mathcal{C}_1 \neq \mathcal{C}_2$.

OK, but how? A natural approach is to find a particularly difficult element in $\mathcal{C}_2$, and hope it won't fit into $\mathcal{C}_1$.

> **Next Question:** How does one compare the difficulty of two arbitrary problems?

# Cop Out



Comparing the intrinsic difficulty (logical depth) of computational problems in general can be quite tricky, it often looks like the old apples-and-oranges situation.

Fortunately, for decision problems we can weasel around this issue: we can always compare Yes/No answers, even if the questions are totally unrelated.

Restricting one's attention to decision problems is less of a loss than one might think, one can often rephrase more general questions in terms of decision problems, without losing much information.

# Reductions

Suppose $\mathcal{C} \subseteq \mathfrak{P}(\mathbb{N})$ is some collection of decision problems.

## Definition

A reduction is a pre-order $\preceq$ (reflexive and transitive) on $\mathfrak{P}(\mathbb{N})$.

$B$ is $\mathcal{C}$-hard (for $\preceq$) if $A \preceq B$ for all $A \in \mathcal{C}$.

$B$ is $\mathcal{C}$-complete (for $\preceq$) if $B$ is $\mathcal{C}$-hard and $B \in \mathcal{C}$.

Very often we also want $\preceq$ to be compatible with $\mathcal{C}$:

$$A \preceq B, B \in \mathcal{C} \qquad \text{implies} \qquad A \in \mathcal{C}$$

# Keeping It Simple



$A \preceq B$ is supposed to mean that $B$ contains enough information to determine membership in $A$, modulo a little auxiliary computation expressed by $\preceq$.

For this to be interesting, $\preceq$ needs to be fairly weak computationally. The heavy lifting should happen in $B$, not in the auxiliary computation. Otherwise we could get weird and not particularly helpful situations like $K \preceq \emptyset$.

Compatibility tries to address this issue, the reduction has to be civilized (with respect to class $\mathcal{C}$).

# Hardness is Easy



It is not difficult to fabricate a hard set $B$ for $\mathcal{C}$: just take the disjoint union over the whole class. As an example, consider the class $\mathcal{E}$ of all semidecidable sets and let

$$H = \{ \langle e, x \rangle \mid x \in W_e \} \subseteq \mathbb{N}$$

By definition, $H$ contains information about all of $\mathcal{E}$, so $H$ is certainly $\mathcal{E}$-hard (we'll talk about the right reduction in a moment). In this case, $H$ is itself a member of $\mathcal{E}$, so we have a complete set.

Alas, achieving completeness in general can be difficult: to obtain hardness we want to pack a lot of information into the set, but to ensure membership in $\mathcal{C}$ we have to keep the set simple–a classical case of clashing requirements.

# Tackling AH



So to prove Post's theorem that the arithmetical hierarchy does not collapse, the hope is that we can

- come up with a reasonable notion of reduction for the classes $\Sigma_n$,
- and find a $\Sigma_n$-complete set for each $n$,
- and show that these complete sets are all different.

And, it would be nice to have some natural examples of $\Sigma_n$-complete sets, at least for the first few levels.

# The Mother of All Reductions



We still need to explain what we mean concretely by a reduction. Because of compatibility, we will wind up with different reductions for different complexity classes.

What is the most general way we could translate some decision problem $A$ into another decision problem $B$?

The idea is to **pretend** that we can solve $B$, and use this (quite possibly fictitious) ability to answer questions about $A$. Note that we may ask repeated questions about $B$, and initially we will allow arbitrarily complicated computations surrounding these queries..

# Turing's Wild Idea



> Let us suppose we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle as it were . . . this oracle cannot be a machine. With the help of the oracle we could form a new kind of machine (call them $o$-machines), having as one of its fundamental processes that of solving a given number-theoretic problem.

# Classical Oracles



# Quoi?



- Oracle machines were introduced by Turing in a 1939 paper under the name of $o$-machines (as opposed to the original $a$-machines). Curiously, he never really exploited this idea (Post and Kleene did).

- As we will see, OTMs provide a powerful way to classify problems according to their complexity.

- Think of the oracle as a data base, created by an alien super-civilization a trillion years ago. The oracle Turing machine has access to all their wisdom.

# Less Informally



Fix some set $A \subseteq \mathbb{N}$. We want to add knowledge about $A$ to a Turing machine: the machine writes $x \in \mathbb{N}$ on a special tape (alternatively, a special area on the main tape) and then enters a magical query state $q_{\text{query}}$.

At this point, a genie takes over and checks whether $x \in A$. If so, the machine state is changed to $q_{\text{yes}}$, otherwise it is changed to $q_{\text{no}}$.

Then the normal computation resumes.

## Definition

A Turing machine with this added facility is an oracle Turing machine (OTM) with oracle $A$.

# For Real

The last slide is rather vague. But one can make the notion of an oracle Turing machine precise, the same way as an ordinary Turing machine can be defined precisely (in the usual pseudo-set-theory setting).

## Exercise

*Give a precise definition of oracle Turing machines by redefining the next-step relation.*

We write $\mathcal{M}_e^A$ for the $e$th OTM with oracle $A$ and $\{e\}^A$ for the corresponding partial function.

The Turing $o$-machine is the single most important concept in computability, theoretical or practical.

# Comments

- The whole point behind Turing machines (meaning $a$-machines) is that they are physically realizable, at least in principle. They capture everything that is physically possible, and more.
- But $o$-machines are **NOT**, though some esteemed members of the hyper-computation community fail to realize that.
- The one exception is when the oracle $A$ is decidable: in this case we can replace magic by another Turing machine.

In other words, decidable oracles are no better than none, we get no additional power (but note that the running time might improve).

# The Key Generalization



We can now generalize all the basic notions we have relative to an arbitrary oracle $A$: we copy our old definitions, replacing TM by OTM everywhere.

- $A$-computable
- $A$-decidable
- $A$-semidecidable

So computable is $\emptyset$-computable and so forth.

# More Precisely . . .

## Definition

Let $A, B \subseteq \mathbb{N}$. A partial function $f : \mathbb{N}^k \nrightarrow \mathbb{N}$ is computable in $A$ or $A$-computable if there is an oracle Turing machine that computes $f$ using $A$ as an oracle.

A set $B$ is Turing-reducible to $A$ or $A$-decidable if its characteristic function is $A$-computable.

A set $B$ is semidecidable in $A$ or $A$-semidecidable if it is the support of an $A$-computable function.

Turing reducibility is usually written

$$B \leq_T A$$

# A Preorder

## Proposition

*Turing reducibility is a preorder (reflexive and transitive):*

- $A \leq_T A$.
- $A \leq_T B$ *and* $B \leq_T C$ *implies* $A \leq_T C$.

*Proof.* Every call to oracle $B$ in the algorithm for $A$ can be replaced by a computation which uses calls to oracle $C$.

The actual computation can be absorbed by the algorithm, so that all that remains are the calls to oracle $C$.

□

Of course, $\leq_T$ is not symmetric. There even are incomparable semidecidable sets, but that's more complicated.

# Dire Warning



Note that by definition Turing reducibility can handle complements:

$$\overline{A} \leq_T A$$

As a consequence, Turing reducibility is compatible with the class of all arithmetical sets, but not with the class $\mathcal{E}$ of semidecidable sets, or $\Sigma_n$ in general; it is simply too brutish for these classes.

We will need to fix this later, but, for the time being, let's just explore Turing reducibility.

# A Meta-Theorem

As it turns out, the machinery we developed for computable functions carries over to $A$-computable functions, just about verbatim.

In particular we still have a Kleene style enumeration

$$\{e\}^A$$

of all $A$-computable functions.

Then $B \leq_T A$ means

$$B(x) \simeq \{e\}^A(x)$$

for some index $e$ (we abuse notation slightly by writing $B$ for the characteristic function of $B$).

**Claim**

*Every semidecidable set is $K$-decidable.*

*Proof.* Let $W$ be semidecidable. We will translate a membership query for $W$ into a membership query for $K$.

Here is an overly careful argument. Define the following function:

$$g(x,z) \simeq \begin{cases} 0 & \text{if } x \in W, \\ \uparrow & \text{otherwise.} \end{cases}$$

Clearly, $g$ is computable and has some index $\widehat{g}$. Then

$$g(x,z) \simeq \{\widehat{g}\}(x,z) \simeq \{S_1^1(\widehat{g},x)\}(z)$$

by the $S$-$m$-$n$ theorem.

The map $f(x) := S^1_1(\widehat{g}, x)$ is easily computable (primitive recursive).

But then $x \in W \iff f(x) \in K$, done. $\square$

This is the anal-retentive version of the proof, usually this would be abbreviated to something a bit more cryptic like so: let

$$\{f(x)\}(z) \simeq \begin{cases} 0 & \text{if } x \in W, \\ \uparrow & \text{otherwise.} \end{cases}$$

Then $f$ is primitive recursive, blahblahblah, done.

Note that the critical point here is that an index for the last function is easily computable. One can prove this formally by applying the $S$-$m$-$n$ theorem, but usually no one bothers.

Since an oracle TM can compute $f$, it can check membership in $W$ by making just one call to the oracle and returning the same answer true or false (a bit like tail recursion).

This is a very special case, in general multiple calls are needed and the OTM has to do more work than just computing a primitive recursive function.

# Better Reductions



The fact that Turing reducibility automatically clobbers complements makes is it awkward to use in connection with semidecidability. Essentially, Turing reducibility is just too coarse for semidecidable sets.

We are looking for a weaker reduction that is compatible:

$$A \preceq B, B \text{ semidecidable} \quad \text{implies} \quad A \text{ semidecidable}$$

**Forward Pointer:** Finding the right notion of reducibility is absolutely critical for lower complexity classes like $\mathbb{NP}$.

# Many-One

Here is a type of reduction that is suggested by some of the examples above.

## Definition

Let $A, B \subseteq \mathbb{N}$. $A$ is many-one reducible to $B$ if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$x \in A \iff f(x) \in B.$$

If function $f$ is in addition injective, then $A$ is one-one reducible to $B$.

In symbols: $A \leq_m B$ and $A \leq_1 B$.

# Downward Closure

**Proposition**

$$A \leq_m B \quad \textit{implies} \quad A \leq_T B$$

Incidentally, the opposite implication is false.

**Proposition**

- *$A \leq_m B$ and $B$ decidable implies that $A$ is decidable.*
- *$A \leq_m B$ and $B$ semidecidable implies that $A$ is semidecidable.*

# One-One



Recall our proof that all semidecidable sets are many-one reducible to $K$: there is an easily computable function $f$ that returns a program that either computes the constant-zero function, or the totally divergent function. We have $x \in W \iff f(x) \in K$, so $W \leq_m K$.

As it turns out, we can even force $f$ to be one-one, so in fact $W \leq_1 K$ [†].

Start with a many-one function $f$, and construct an injective transformation $g$ in a way that sets $g(x) = f(x)$ provided that $f(x) \neq g(z)$ for all $z < x$.

Otherwise modify the program $e = f(x)$ to do something useless, like counting to $x$ before the actual program runs. The new program $e'$ has the same input/output behavior as $e$, but can be chosen different from all the programs already encountered.

---

[†] This is a bit of a hack and does not mean much, but on-one reducibility comes in handy in some theorem.

# Convention



In conjunction with the arithmetical hierarchy, the default reduction is many-one reducibility.

E.g., when we say that $A$ is $\Sigma_n$-complete this means

$$A \in \Sigma_n \qquad \text{and} \qquad B \leq_m A, \text{ all } B \in \Sigma_n$$

In particular $\mathcal{E}$-complete always is defined this way.

## Exercise

*Show that $\leq_m$ is compatible with $\Sigma_n$:*

$$B \leq_m A, A \in \Sigma_n \qquad \textit{implies} \qquad B \in \Sigma_n$$

# Some Index Sets



Here is a small collection of natural index sets.

$$\begin{aligned}
\mathsf{FIN} &= \{\, e \in \mathbb{N} \mid W_e \text{ is finite} \,\} \\
\mathsf{INF} &= \{\, e \in \mathbb{N} \mid W_e \text{ is infinite} \,\} \\
\mathsf{TOT} &= \{\, e \in \mathbb{N} \mid W_e = \mathbb{N} \,\} \\
\mathsf{REC} &= \{\, e \in \mathbb{N} \mid W_e \text{ is decidable } \,\} \\
\mathsf{CMP} &= \{\, e \in \mathbb{N} \mid W_e \text{ is complete } \,\}
\end{aligned}$$

By Rice's theorem, they are all undecidable. But we would like to understand their relative complexity more precisely.

# Relative Complexity

**Theorem**

$$K <_T \text{FIN} <_T \text{REC}$$

$$\text{FIN} \equiv_T \text{INF} \equiv_T \text{TOT}$$

FIN $\equiv_T$ INF is clear, but the others require work.

Also, we would like to have many-one reductions whenever possible.

We'll just hint at what needs to be done for a proof.

# How Bad Can It Be?



Intuitively, FIN is not semidecidable since by definition

$$e \in \mathsf{FIN} \iff \exists\, b \forall\, x, \sigma\, (x \in W_{e,\sigma} \Rightarrow x < b)$$

The matrix of the formula is easily decidable, but we are not just doing an infinite search (that would be the $\exists\, b$ part). Because of the universal quantifier, the only obvious bound we get from this is $\Sigma_2$.

But note: this is just an upper bound, not a lower bound. Experience shows, though, that unless we are obviously wasting quantifiers these upper bounds are tight.

# Lower Bound

**Claim:** FIN is $\Sigma_2$-complete.

To see this, suppose $A \in \Sigma_2$ arbitrary. We will show that $A \leq_m$ FIN.

First note that by the definition of $\Sigma_2$ we have

$$x \in A \Leftrightarrow \exists\, s \forall\, t\ R(x, s, t)$$

for some decidable relation $R$.

Define $f(x) = e$ where $e$ is an index such that

$$W_e = \{\, z \mid \forall\, s < z\, \exists\, t\, \neg R(x, s, t) \,\}$$

One can check that the set on the right is indeed semidecidable.

Then $f$ is the desired many-one reduction.

# Even Worse



Unsurprisingly, REC is even worse. Quantifier counting produces an upper bound of $\Sigma_3$

$$\begin{aligned} e \in \mathsf{REC} &\iff \exists\, e' \,(W_e = \mathbb{N} - W_{e'}) \\ &\iff \exists\, e' \,\forall\, x \,(\forall\, \sigma \,(W_{e,\sigma} \cap W_{e',\sigma} = \emptyset) \wedge \exists\, \tau \,(x \in W_{e,\tau} \cup W_{e',\tau})) \end{aligned}$$

As before, this upper bound is already tight, one can show that $A \leq_m \mathsf{REC}$ for any $\Sigma_3$ set A.

Alas, this requires more work ...

# Another Proof

We will show that

$$\text{INF} \leq_m \text{TOT}$$

As a warmup exercise, consider the downward closure operation

$$\text{dc}(A) = \{\, x \in \mathbb{N} \mid \exists\, a \in A\, (a \geq x) \,\}$$

So $\text{dc}(A)$ fills in the holes in $A$.

In general

$$\text{dc}(A) = \begin{cases} \{\, z \mid z \leq \max A \,\} & \text{if } A \text{ is finite,} \\ \mathbb{N} & \text{otherwise.} \end{cases}$$

# Effectivizing the Argument

So dc$(A)$ translates from infinite to total, fairly close to what we want.

But note that dc$(W)$ is r.e. whenever $W$ is r.e. Moreover, we can compute an index for dc$(W)$ from an index for $W$. In fact, there is a primitive recursive function $f$ that does it:

$$\{f(e)\}(z) \simeq \begin{cases} 0 & \text{if } \exists\, u\,(u \geq z \wedge u \in W_e), \\ \uparrow & \text{otherwise.} \end{cases}$$

Hence $\{f(e)\}$ is either constant 0, or undefined for all sufficiently large $z$. More precisely, $W_e$ is infinite iff $W_{f(e)} = \mathbb{N}$.

Hence we have the desired reduction: $e \in \mathsf{INF} \iff f(e) \in \mathsf{TOT}$, done.

# Motivation

So far, we only have one basic separation result: decidable is different from semidecidable because of the Halting set.

It might be a good idea to try to generalize this step from $\emptyset$ to $K$, so we can produce a whole chain of increasingly complicated sets (hopefully in the arithmetical hierarchy).

# The Jump

Here is nice way to explain (and generalize) the increasing levels of complexity in the arithmetical hierarchy.

## Definition

The jump of $A \subseteq \mathbb{N}$ is the Halting Set for OTMs with oracle $A$:

$$A' = \{\, e \mid \{e\}^A(e) \downarrow \}$$

So $\emptyset' = K$ is just the ordinary Halting Set.

But $\emptyset'' = K'$ is a new beast of higher complexity.

And $\emptyset'''$ is even worse, things are getting more and more undecidable.

**Lemma**

*$A'$ is $A$-semidecidable but not $A$-decidable.*

*Proof.*

The proof is verbatim the same as for plain Halting[†].

□

So we get an infinite hierarchy of more and more complicated problems:

$$\emptyset,\ \emptyset',\ \emptyset'',\ \emptyset''',\ \ldots,\ \emptyset^{(n)},\ldots$$

In reality, though, nothing much beyond $\emptyset^{(4)}$ seems to play a major role (except for problems entirely outside of this hierarchy like arithmetic truth).

[†]There is such a thing as a free lunch, on rare occasions.

# Jumping the Shark

- Even worse, we can collect all these sets into a single one, essentially by taking a disjoint union:

$$\emptyset^{(\omega)} = \{ \langle e, n \rangle \mid e \in \emptyset^{(n)} \}$$

  This happens to be exactly arithmetic truth.

- And nothing stops us from forming $(\emptyset^\omega)'$, $(\emptyset^\omega)''$ and so on. We can iterate the jump transfinitely often: $\omega^2$ times, $\omega^\omega$ times,

$$\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$$

  times. This is a subject for generalized recursion theory.

- Aspirin, anyone?

# Reductions vs. Jump

**Lemma**

*$A$ is semidecidable in $B$ if, and only if, $A \leq_1 B'$.*

*Proof.* Suppose $A$ is semidecidable in $B$. Hence there is a primitive recursive function $f$ such that

$$\{f(x)\}^B(z) \simeq \begin{cases} 0 & \text{if } x \in A, \\ \uparrow & \text{otherwise.} \end{cases}$$

# Proof, contd.



By assumption, this function $f(x)$ is $B$-computable.

But then $x \in A \iff f(x) \in B'$.

As usual, we can force $f$ to be injective by choosing a different index for each $x$.

Now suppose $A \leq_1 B'$, say, $x \in A \iff f(x) \in B'$.

To enumerate $A$ given $B$ as oracle, proceed in stages.

At stage $\sigma$, compute $f(x)$ for all $x < \sigma$.

Enumerate $B'_\sigma$ using oracle $B$.

If any of the $f(x)$ appear in $B'_s$, enumerate the corresponding $x$s into $A$. □

# Jump in the AH

**Theorem (Jump Theorem)**

- $\emptyset^{(n)}$ *is* $\Sigma_n$*-complete.*
- $A \in \Sigma_{n+1}$ *iff* $A$ *is semidecidable in* $\emptyset^{(n)}$*.*
- $A \in \Delta_{n+1}$ *iff* $A \leq_T \emptyset^{(n)}$*.*

# Concrete Examples



So $\emptyset' = K$ is just the ordinary Halting Set.

But $\emptyset'' = K'$ captures totality: TOT $\leq_T \emptyset''$.

Similarly $\emptyset'''$ captures decidability: REC $\leq_T \emptyset'''$.

And $\emptyset''''$ takes care of Universality: UTM $\leq_T \emptyset''''$.

For higher levels, things become a bit more obscure, there aren't really natural examples.

# Turing versus One-One



**Theorem**

*$A \leq_T B$ if, and only if, $A' \leq_1 B'$.*

*Proof.*

By definition chasing.

□

# Levels of AH

**Theorem**

*$A$ is in $\Sigma_{n+1}$ if, and only if, $A$ is semidecidable in some $\Pi_n$ set or some $\Sigma_n$ set.*

*Proof.*

$\Rightarrow$: Use the description of semidecidability in terms of projections.

$\Leftarrow$: Suppose for some $\Pi_n$ set $B$ we have

$$x \in A \Leftrightarrow x \in W_e^B$$

$$\Leftrightarrow \exists\, \sigma, \alpha\, (\alpha = B \upharpoonright \sigma \wedge x \in W_{e,\sigma}^{\alpha})$$

Here $\alpha$ is supposed to be the finite approximation of $B$ of length $\sigma$, just a bit-vector that replaces the oracle, and only works for queries less than $\sigma$.

□

# Proof Jump Theorem



We use induction on $n$; note that $n = 1$ is under control.

$$
\begin{aligned}
A \in \Sigma_{n+1} &\Leftrightarrow A \text{ is semidecidable in some } B \in \Pi_n \\
&\Leftrightarrow A \text{ is semidecidable in some } C \in \Sigma_n \\
&\Leftrightarrow A \text{ is semidecidable in } \emptyset^{(n)} \\
&\Leftrightarrow A \leq_1 \emptyset^{(n+1)}
\end{aligned}
$$

□

# Turing Degrees

We can lump together problems that are mutually Turing reducible, in some sense these problems are "equivalent."

## Definition

Two sets $A$ and $B$ are Turing equivalent if

$$A \leq_T B \qquad \text{and} \qquad B \leq_T A$$

This defines an equivalence relation $\equiv_T$ whose equivalence classes are called Turing degrees.

Notation:

$$\deg_T(A) = \{\, B \mid A \equiv_T B \,\}$$

# Degrees of Unsolvability



Historically, Turing degrees were called degrees of unsolvability since they measure the distance between a problem and solvability.

The only degree that is easy to describe is

$$\deg_T(\emptyset) = \text{all decidable sets.}$$

How about the degree of the Halting Problem, $\deg_T(K)$?

Or an index set like TOT?

# Just Two Results

**Theorem (Friedberg, Muchnik 1955/57)**

*There exist two semidecidable sets that are not comparable wrto Turing reducibility.*

**Theorem (Sacks 1964)**

*Let $A$ and $B$ semidecidable such that $A <_T B$. Then there exists a semidecidable $C$ such that $A <_T C <_T B$.*

# Quoi?



These results are somewhat counterintuitive (at least to me they are, and I'm quite familiar with the proofs).

Write $\mathcal{E}$ for the class of semidecidable sets. It is a fact of historical experience that for any concrete problem $A \in \mathcal{E}$ the following happens:

- Someone proves that $A$ is decidable, or
- someone proves that $A$ is $\mathcal{E}$-complete.

After a century of computability theory, not a single natural example of an intermediate degree is known. No one has any idea why.

The proof to the Friedberg-Muchnik theorem uses a new technique, called a finite injury priority argument. CRT lost its innocence in those days: priority arguments are significantly more complicated than anything that has come before.

And, annoyingly, they always seem to produce artificial results.

A lot of people find that rather off-putting.

*The study of degrees [of unsolvability] seems to be appealing only to some special kind of temperament since the results seem to go into many different directions. Methods of proof are emphasized to the extent that the main interest in this area is said to be not so much the conclusions proved as the elaborate methods of proof.*

*Hao Wang, 1977*

Hao Wang was a student W.V.O. Quine and Stephen Cook as his student.