Lecture #14: Oracle Reductions
Key Concepts

Reducibilities

Definition 1. A ***reducibility*** is any binary relation $\preceq_{\mathsf{Q}}$ between languages (possibly over different alphabets) such that the following properties are satisfied.

(a) $L \preceq_{\mathsf{Q}} L$ for every language $L \subseteq \Sigma^\star$ (and for every alphabet Σ).

(b) For all languages $L_1 \subseteq \Sigma_1^\star$, $L_2 \subseteq \Sigma_2^\star$ and $L_3 \subseteq \Sigma_3^\star$ (and alphabets Σ_1, Σ_2 and Σ_3) if $L_1 \preceq_{\mathsf{Q}} L_2$ and $L_2 \preceq_{\mathsf{Q}} L_3$ then $L_1 \preceq_{\mathsf{Q}} L_3$.

Oracle Reductions

Definitions

Definition 2. An ***oracle for a language $L \subseteq \Sigma_L^\star$*** is a device that is capable of reporting whether any string $\omega \in \Sigma_L^\star$ is a member of L.

Definition 3. Let $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$. Then L_1 is ***oracle reducible*** to L_2,

$$L_1 \preceq_{\mathsf{O}} L_2,$$

if there exists an algorithm that decides membership in L_1 that uses an oracle (that is, a subroutine) deciding membership in L_2.

Describing an Oracle Reduction Between Languages

To describe an oracle reduction from a language $L_1 \subseteq \Sigma_1^\star$ to a language $L_2 \subseteq \Sigma_2^\star$:

1. Give pseudocode for an algorithm that decides membership in L_1 using a separate ***Boolean method*** that decides membership in L_1 *You do not need to write a method deciding L_2 — just assume that one exists.*

2. If this is not obvious, *prove the correctness* of the algorithm that decides membership in L_1, assuming the correctness of an (unknown) algorithm that decides membership in L_2 and that is used whenever it is needed.
3. Adding more detail (only) as needed, show that *if there exists* a multi-tape Turing machine M_2 that decided membership in L_2 then this can be used as a component in a multi-tape Turing machine M_1 that would decide membership in L_1.

Useful Properties

The set of oracle reductions forms a ***reducibility*** — as this is defined above.

Let $L_1 \subseteq \Sigma_1^\star$ and let $L_2 \subseteq \Sigma_2^\star$, for alphabets Σ_1 and Σ_2. If

- $L_1 \preceq_{\mathsf{O}} L_2$, and
- L_2 is decidable,

then L_1 is decidable as well. That is, the set of decidable languages is ***closed*** under oracle reductions.

Note: It follows that if

- $L_1 \preceq_{\mathsf{O}} L_2$, and
- L_1 is ***undecidable***,

then L_2 is ***undecidable***.

However, the set of recognizable language is ***not*** closed under oracle languages: There exist languages $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ such that

- $L_1 \preceq_{\mathsf{O}} L_2$,
- L_2 is recognizable, but
- L_1 is unrecognizable.

Thus oracle reductions ***cannot*** be used to prove that languages are unrecognizable.

Another Undecidable Language

Let $\mathit{NA}_{\mathsf{TM}} \subseteq \Sigma_{\mathsf{TM}}^\star$ be the set of encodings of Turing machines M, and input strings ω for M, such that M ***does not*** accept μ. It was proved (in the preparatory material for this lecture), using an oracle reduction, that the language $\mathit{NA}_{\mathsf{TM}}$ is undecidable.