Computer Science 351

Oracle Reductions

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #16

Goals for Today

Goals for Today:

- Introduction to ***reducibilities*** and ***oracle reductions***.
- A discussion of ***closure*** of various sets of languages with respect to reducibilities, and a discussion of why this is important.

Note: In my opinion, *Introduction to the Theory of Computation* ***is not*** a good reference for this material.

Reducibilities

Definition #1: A ***reducibility*** is any binary relation $\preceq_Q$ between languages (possibly over different alphabets) such that the following properties are satisfied.

(a) $L \preceq_Q L$ 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 $L_1 \preceq_Q L_2$ and $L_2 \preceq_Q L_3$ then $L_1 \preceq_Q L_3$.

Oracle Reductions

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

- This is like a "black box" or ***method from a library that you can call, as a subroutine*** that can be relied on to decide membership in L.

  You do not need to write this black box (or library routine) or understand how it works — but you *do* get to use it.

Oracle Reductions

- This is also like a ***component*** in a larger system:

Each ***component*** carries out a well-defined job. You do not care — or need to know — *how* the component works, as long as it does the job it is supposed to.

Oracle Reductions

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,

$$L_1 \preceq_O L_2,$$

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

Oracle Reductions

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.

Oracle Reductions

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.

Oracle Reductions

Formal Definitions

- Oracles, and oracle reductions, can be more formally defined using ***oracle Turing machines***.
- These are not needed for the rest of CPSC 351.

Oracle Reductions

It might seem like this what you are describing is...

Magic!!!

Oracle Reductions

Well... ***in a way,*** it really is.

- It is possible to use arguments involving an oracle reduction with an oracle for L, for certain languages L, to prove that L is *undecidable* — so that there is no algorithm that can be used to decide membership in L, at all.

Oracle Reductions — An Example

Recall the alphabet Σ_{TM} that was introduced in the lecture material introducing ***universal Turing machines***, along with the following languages (all of which are subsets of $\Sigma_{\text{TM}}^{\star}$):

- $L_{\text{TM+I}} \subseteq \Sigma_{\text{TM}}^{\star}$ is the language of encodings of Turing machines M and input strings ω for M. As noted in the above lecture material, this language is ***decidable***.
- $A_{\text{TM}} \subseteq \Sigma_{\text{TM}}^{\star}$ is the language of encodings of Turing machines M and input strings ω for M such that M ***accepts*** ω. It was established in the above lecture material that A_{TM} is ***recognizable*** — but it was also established, in that lecture material, that A_{TM} is ***undecidable***.

Oracle Reductions — An Example

- Let $NA_{\mathsf{TM}} \subseteq \Sigma^{\star}_{\mathsf{TM}}$ be the language of encodings of Turing machines M and input strings ω for M such that M *does not* accept ω.
- Then $A_{\mathsf{TM}} \cup NA_{\mathsf{TM}} = L_{\mathsf{TM+I}}$ and $A_{\mathsf{TM}} \cap NA_{\mathsf{TM}} = \emptyset$.
- Consider the algorithm — which makes use of a hypothetical method to decide membership in NA_{TM}, at line 2 — which is shown on the following slide.

Oracle Reductions — An Example

```
boolean acceptInput (μ : Σ*_TM) {
1.  if (μ ∈ L_TM+I)
2.    if (μ ∈ NA_TM) {
3.      reject μ
      } else {
4.      accept μ
      }
    } else {
5.    reject μ
    }
}
```

Oracle Reductions — An Example

Claim #4: Let $\mu \in A_{\mathsf{TM}}$. Then, if the above algorithm is executed on input μ, then this input string is ***accepted***.

Proof: Let $\mu \in A_{\mathsf{TM}}$ and consider an execution of the above algorithm on input μ.

- Since $A_{\mathsf{TM}} \subseteq L_{\mathsf{TM+I}}$, $\mu \in L_{\mathsf{TM+I}}$ and the test at at line 1 is *passed*: The algorithm's execution continues with the step at line 2.
- Since $A_{\mathsf{TM}} \cap NA_{\mathsf{TM}} = \emptyset$, $\mu \notin NA_{\mathsf{TM}}$ and the test at line 2 is *failed*: The algorithm's execution continues with the step at line 4.
- The string μ is accepted when the step at line 4 is executed.

Since μ was arbitrarily chosen from A_{TM}, it follows that the algorithm accepts every string in A_{TM}, as claimed. □

Oracle Reductions — An Example

Claim #5: Let $\mu \in \Sigma^{\star}_{\mathsf{TM}}$ such that $\mu \notin A_{\mathsf{TM}}$. Then, if the above algorithm is executed on input μ, then this input string is ***rejected***.

Proof: Let $\mu \in \Sigma^{\star}_{\mathsf{TM}}$ such that $\mu \notin A_{\mathsf{TM}}$. Then one of the following cases holds.

(a) $\mu \in L_{\mathsf{TM+I}}$ but $\mu \notin A_{\mathsf{TM}}$.

(b) $\mu \notin L_{\mathsf{TM+I}}$.

The cases are considered separately on the following slides.

Oracle Reductions — An Example

(a) *Case:* $\mu \in L_{\text{TM+I}}$ but $\mu \notin A_{\text{TM}}$. Since $A_{\text{TM}} \cup NA_{\text{TM}} = L_{\text{TM+I}}$ it follows that $\mu \in NA_{\text{TM}}$. Consider an execution of the above algorithm on input μ:

- Since $\mu \in L_{\text{TM+I}}$ the test at line 1 is *passed* when checked, so that the execution of the algorithm continues with the test at line 2.
- Since $\mu \in NA_{\text{TM}}$ the test at line 2 is also *passed*, so that the execution of the algorithm continues with the step at line 3.
- The input string μ is *rejected* when the step at line 3 is executed.

Thus the string μ is rejected in this case.

Oracle Reductions — An Example

(b) *Case:* $\mu \notin L_{\mathsf{TM+I}}$. Consider an execution of the above algorithm on input μ.

- Since $\mu \notin L_{\mathsf{TM+I}}$ the test at line 1 is *failed* when checked, so that the execution of the algorithm continues with the step at line 5.
- The input string μ is *rejected* when the step at line 5 is executed.

Thus the string μ is rejected in this case.

Since μ is rejected in every case, and μ was an arbitrarily chosen string in $\Sigma^{\star}_{\mathsf{TM}}$ such that $\mu \notin A_{\mathsf{TM}}$, this establishes the claim. □

Oracle Reductions — An Example

- It follows by Claims #4 and #5, above, that the algorithm `acceptInput` would correctly decide membership in A_{TM} — if an algorithm to decide membership in NA_{TM} existed (and was applied, whenever the step at line 2 of the algorithm `acceptInput` was executed).

Oracle Reductions — An Example

It remains only to show that if there exists a multi-tape Turing machine

$$M_{\text{NATM}} = (Q_{\text{NATM}}, \Sigma_{\text{TM}}, \Gamma_{\text{NATM}}, \delta_{\text{NATM}}, q_{0,\text{NATM}}, q_{\text{accept,NATM}}, q_{\text{reject,NATM}})$$

that decides NA_{TM}, then there must also exist a multi-tape Turing machine

$$M_{\text{ATM}} = (Q_{\text{ATM}}, \Sigma_{\text{TM}}, \Gamma_{\text{ATM}}, \delta_{\text{ATM}}, q_{0,\text{ATM}}, q_{\text{accept,ATM}}, q_{\text{reject,ATM}})$$

that decides A_{TM}.

Oracle Reductions — An Example

- Since $L_{\text{TM+I}}$ is decidable there exists a multi-tape Turing machine

$$M_{\text{TM+I}} = (Q_{\text{TM+I}}, \Sigma_{\text{TM}}, \Gamma_{\text{TM+I}}, \delta_{\text{TM+I}}, q_{0,\text{TM+I}}, q_{\text{accept,TM+I}}, q_{\text{reject,TM+I}})$$

  that decides $L_{\text{TM+I}}$.
- The multi-tape M_{ATM} will have (slightly modified versions of) the multi-tape Turing machine $M_{\text{TM+I}}$ (which definitely exists) and the *hypothetical* multi-tape Turing machine M_{NATM} as ***components***.

Oracle Reductions — An Example

- An execution of M_{ATM} on an input $\mu \in \Sigma^{\star}_{\text{TM}}$ begins by copying μ onto $M_{\text{TM+I}}$'s input tape (repositioning tape heads to leftmost cells).
- M_{ATM} then enters $M_{\text{TM+I}}$'s start state, $q_{0,\text{TM+I}}$. In M_{ATM}, transitions to $M_{\text{TM+I}}$'s accept state are redirected to a state beginning the *next* step — and transitions to $M_{\text{TM+I}}$'s reject state are redirected to M_{ATM}'s reject state, so that the steps at lines 1 and 5 of the algorithm `acceptInput` have been implemented.

Oracle Reductions — An Example

- If $M_{\text{TM+I}}$ accepted μ then the execution of M_{ATM} continues with the input string being copied to M_{NATM}'s input tape (repositioning tape heads to leftmost cells).
- M_{ATM} then enters M_{NATM}'s start state, $q_{0,\text{NATM}}$. In M_{ATM}, transitions to M_{NATM}'s accept state are redirected to M_{ATM}'s reject state and transitions to M_{NATM}'s reject state are redirected to M_{ATM}'s accept state, so that the the steps at lines 2, 3 and 4 of the algorithm `acceptInput` have been implemented.
- It follows, by the correctness of the algorithm `acceptInput`, that M_{ATM} would be a multi-tape Turing machine that decides the language A_{TM}.

Oracle Reductions — An Example

We have proved the following.

Claim #6: $A_{\mathrm{TM}} \preceq_{\mathrm{O}} NA_{\mathrm{TM}}$.

"Oracle Reductions" Comprise a Reducibility

Note that the set of all oracle reductions forms a *binary relation* between the set of languages and itself.

Claim #7: The set of oracle reductions forms a ***reducibility***.

- This means that $L \preceq_O L$, for every language $L \subseteq \Sigma^\star$ (for every alphabet Σ) and that, for all languages $L_1 \subseteq \Sigma_1^\star$, $L_2 \subseteq \Sigma_2^\star$, and $L_3 \subseteq \Sigma_3^\star$ (for alphabets Σ_1, Σ_2 and Σ_3), if $L_1 \preceq_O L_2$ and $L_2 \preceq_O L_3$ then $L_1 \preceq_O L_3$.
- A proof of Claim #7 is included in a supplemental document for this lecture.

Closure of the Set of Decidable Languages

Claim #8: Let $L_1 \subseteq \Sigma_1^\star$ and let $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2). If $L_1 \preceq_O L_2$ and L_2 is decidable then L_1 is decidable too.

In other words, the set of decidable languages is ***closed*** under oracle reductions.

- A proof of Claim #8 is also included in a supplemental document for this lecture.

Closure of the Set of Decidable Languages

Corollary #9: Let $L_1 \subseteq \Sigma_1^\star$ and let $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2). If $L_1 \preceq_O L_2$ and L_1 is ***undecidable*** then L_2 is undecidable too.

- This result — and its consequences — is the main reason why ***oracle reductions*** are part of this course.

Closure of the Set of Decidable Languages

Example, Continued:

Claim #10: The language NA_{TM} is undecidable.

Proof:

- It follows by Claim #6, above, that $A_{\text{TM}} \preceq_{\text{O}} NA_{\text{TM}}$.
- It was shown, in the lecture material that introduced ***universal Turing machines***, that the language A_{TM} is undecidable.
- It now follows by Corollary #9, above, that the language NA_{TM} is undecidable too. □

Closure of the Set of Decidable Languages

Another process to prove that a language $L \subseteq \Sigma^\star$ is undecidable:

- Choose another language $\widehat{L} \subseteq \widehat{\Sigma}^\star$ (over some alphabet $\widehat{\Sigma}$) such that $\widehat{L}$ is undecidable.
- Prove that $\widehat{L} \preceq_{\mathrm{O}} L$.
- Conclude, by Corollary #9, above, that L must be undecidable too.

Non-Closure of the Set of Recognizable Languages

Claim #11: There exist languages $L \subseteq \Sigma^\star$ and $\widehat{L} \subseteq \widehat{\Sigma}^\star$ (over alphabets Σ and $\widehat{\Sigma}$) such that

- L is ***not*** recognizable,
- $\widehat{L}$ *is* recognizable, and
- $L \preceq_{\mathrm{O}} \widehat{L}$.

The the set of recognizable languages is ***not*** closed under oracle reductions.

- In particular, this is true for $L = A_{\mathrm{TM}}^{C}$ and $\widehat{L} = A_{\mathrm{TM}}$.

Non-Closure of the Set of Recognizable Languages

- This implies that oracle reductions ***cannot*** be used to prove that languages are *unrecognizable*.
- Another kind of a reducibility — “many-one reductions” — that *can* be used for this will be introduced in the next set of lecture material.

Oracle Reductions are Also Called "Turing Reductions"

Oracle reductions also called a ***Turing reductions*** and the notation $\preceq_T$ is sometimes used instead of $\preceq_O$.

- While he did apparently did not call the relationships between problems (that he was working with) "reductions", Turing's work included an early version of the technique introduced in this lecture.