Lecture #17: Many-One Reductions
A First Example of a Reduction between Decision Problems

Overview

In this document, additional details are given for a second example that was (more briefly) described in the preparatory material for Lecture #17, specifically, an example of a "reduction between decision problems". Once this again, this document is for interest only — but may be helpful to students wishing to see another example of a reduction.

Consider, now, the following decision problems.

Acceptance Problem

Instance: A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and an input string $\omega \in \Sigma^\star$

Question: Does M accept ω?

Halting Problem

Instance: A Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and an input string $\omega \in \Sigma^\star$

Question: Does M halt when executed on input ω?

Instances of these problems are encoded using the alphabet Σ_{TM} and the encoding scheme, for Turing machines and their input strings, described in the preparatory material for Lecture #15. Using these, the following languages are defined:

- $L_{\text{TM+I}} \subseteq \Sigma_{\text{TM}}^\star$ is the language of encodings of Turing machines M (with some input alphabet Σ) and input strings $\omega \in \Sigma^\star$ for M. This is the language of Yes-instances for both the "Acceptance Problem" and the "Halting Problem" and, as discussed in the preparatory material and supplemental documents for Lecture #15, this language is decidable.

- $A_{\text{TM}} \subseteq L_{\text{TM+I}}$ is the language of encodings of Turing machines M (with some input alphabet $\Sigma^\star$) and input strings $\omega \in \Sigma^\star$ for M such that M accepts M — the language of Yes-instances for the "Acceptance Problem". As described in the preparatory material for Lecture #15, this language is both recognizable and *undecidable*.

- $NA_{\text{TM}} \subseteq L_{\text{TM+I}}$ is the language of encodings of Turing machines M (with some input alphabet Σ) and input strings $\omega \in \Sigma^\star$ for M such that M *does not* accept ω — the language of No-instances for the "Acceptance Problem". As discussed in the preparatory material for Lecture #16, this language is *undecidable*.

- $Halt_{\text{TM}} \subseteq L_{\text{TM+I}}$ is the language of encodings of Turing machines M (with some input alphabet Σ) and input strings $\omega \in \Sigma^\star$ for M such that M halts when executed on input M — the language of Yes-instances of the "Halting Problem".

- $Loop_{\text{TM}} \subseteq L_{\text{TM+I}}$ is the language of encodings of Turing machines M (with some input alphabet Σ) and input strings $\omega \in \Sigma^\star$ for M such that M loops on ω — the language of No-instances of the "Halting Problem"

This document provides more details of a proof that the "Halting Problem is reducible to the Acceptance Problem" that was briefly described in the preparatory material for Lecture #17 — that is, a proof that the language of Yes-instances for the "Halting Problem" is many-one reducible to the language of Yes-instances of the "Acceptance Problem",

$$A_{\text{TM}} \preceq_{\text{M}} Halt_{\text{TM}}.$$

A High-Level Mapping

As stated in the preparatory material for Lecture #17, this kind of many-one reduction can be obtained by starting with a mapping, φ, from instances of the "Acceptance Problem", to instances of the "Halting Problem", such that, for all Turing machines M (with some input alphabet Σ) and input strings $\omega \in \Sigma^\star$, φ maps (the ordered pair including) M and ω to (an ordered pair including) $\widehat{M}$ and $\widehat{\omega}$ — where $\widehat{M}$ is the same as M except that M's transitions to its rejecting state are redirected to its accepting state, and where $\widehat{\omega} = \omega$.

That is: If

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

and $\omega \in \Sigma^\star$ then φ maps (M, ω) to an ordered pair $(\widehat{M}, \omega)$ where

$$\widehat{M} = (Q, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

such that, for all $q \in Q$ such that $q \notin \{q_{\text{accept}}, q_{\text{reject}}\}$ and for all $\sigma \in \Gamma$,

$$\widehat{\delta}(q,\omega) = \begin{cases} (r,\tau,D) & \text{if } \delta(q,\omega) = (r,\tau,D), \text{ where } r \in Q \setminus \{q_{\text{reject}}\}, \\ & \qquad \tau \in \Gamma, \text{ and } D \in \{\mathtt{L},\mathtt{R}\}, \\ (q_{\text{accept}},\tau,D) & \text{if } \delta(q,\omega) = (q_{\text{reject}},\tau,D), \text{ where } \tau \in \Gamma \text{ and } D \in \{\mathtt{L},\mathtt{R}\}. \end{cases} \quad (1)$$

Claim 1. *Let M be a Turing machine (with some input alphabet Σ) and let $\omega \in \Sigma^\star$ (so that ω is an input string for M), and let $\widehat{M}$ be the Turing machine (which has the same input alphabet Σ) and let $\widehat{\omega} \in \Sigma^\star$ be the string such that φ maps (M,ω) to $(\widehat{M},\widehat{\omega})$ (so that $\widehat{\omega} = \omega$).*

Then M halts, when executed on input ω, if and only if $\widehat{M}$ accepts $\widehat{\omega}$.

In other words, (M,ω) is a Yes-instance of the "Halting Problem" if and only if $(\widehat{M},\widehat{\omega})$ is a Yes-instance of the "Acceptance Problem".

The following lemma will be used in the proof of the above claim.

Lemma 2. *Let M, ω, $\widehat{M}$ and $\widehat{\omega}$ be as in the statement of Claim 1) and let k be a non-negative integer such that M takes at least $k+1$ steps when executed on input ω. Let $\mathcal{C}$ be the configuration of M such that M is in configuration $\mathcal{C}$ after the first k steps of its execution on ω.*

Then $\widehat{M}$ also takes at least $k+1$ steps when executed on input ω and, furthermore, $\widehat{M}$ is also in configuration $\mathcal{C}$ after the first k steps of its execution on $\widehat{\omega}$.

How To Prove This. This can be proved by induction on k.

The claim in the basis follows from the facts that M and $\widehat{M}$ have the same start state (so they have the same start configuration for ω) and the start state is not the accepting state or the rejecting state, so that both Turing machines take at least one step when executed on the input string ω.

The claim, that must be established to complete the inductive step, can be established by an examination of $\widehat{M}$'s transition function, $\widehat{\delta}$, at line (1): The same transition will be applied by both Turing machines if M's computation is not about to end.

The proof is left as an exercise (for students who want to have more practice in writing proofs about computation). □

Proof of Claim 1. Let M, ω, $\widehat{M}$ and $\widehat{\omega}$ be as in the statement of the claim. Either M accepts ω, M rejects ω or M loops on ω. These cases are considered below.

- If M accepts ω then M halts on ω — so (since $\widehat{\omega} = \omega$) it is necessary (and sufficient) to show that $\widehat{M}$ accepts ω in order to establish the claim for this case.

Since M accepts ω in this case, there exists a non-negative integer k such that M accepts ω after taking exactly $k+1$ steps. It follows, by Lemma 2, that M and $\widehat{M}$ are in the same configuration after they have each taken k steps. Now, since M accepts ω at this point, it must use use the fact that a transition

$$\delta(q,\sigma) = (q_{\text{accept}}, \tau, D)$$

is now applied — where $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ is the current state and $\sigma \in \Gamma$ is the symbol that is currently visible on the tape, where $\tau \in \Gamma$, and where $D \in \{\mathtt{L}, \mathtt{R}\}$. One can see by an examination of the definition of $\widehat{\delta}$ at line (1) that (essentially) the same transition

$$\widehat{\delta}(q,\sigma) = \delta(q,\sigma) = (q_{\text{accept}}, \tau, D)$$

is now applied by $\widehat{M}$ as well — so that $\widehat{M}$ accepts ω after taking $k+1$ steps — as needed to establish the claim in this case.

- If M rejects ω then M halts on ω — so (since $\widehat{\omega} = \omega$) it is necessary (and sufficient) to show that $\widehat{M}$ accepts ω in order to establish the claim for this case.

  Since M rejects ω in this case, there exists a non-negative integer k such that M accepts ω after taking exactly $k+1$ steps. It follows, by Lemma 2, that M and $\widehat{M}$ are in the same configuration after they have each taken k steps. Now, since M rejects ω at this point, it must use use the fact that a transition

  $$\delta(q,\sigma) = (q_{\text{reject}}, \tau, D)$$

  is now applied — where $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ is the current state and $\sigma \in \Gamma$ is the symbol that is currently visible on the tape, where $\tau \in \Gamma$, and where $D \in \{\mathtt{L}, \mathtt{R}\}$. One see by an examination of the definition of $\widehat{\delta}$ at line (1) that a transition

  $$\widehat{\delta}(q,\sigma) = (q_{\text{accept}}, \tau, D)$$

  is now applied by $\widehat{M}$ — so that $\widehat{M}$ accepts ω after taking $k+1$ steps — as needed to establish the claim in this case.

- Finally, if M loops on ω then M does not halt on ω — so (since $\widehat{\omega} = \omega$) it is necessary (and sufficient) to show that $\widehat{M}$ *does not* accept ω in order to establish the claim for this case. One way to do that is to show that $\widehat{M}$ also loops on ω.

  Let k be any non-negative integer. Then, since M loops on ω, M takes at least $k+1$ steps when executed on input ω. It follows by Lemma 2 that $\widehat{M}$ takes $k+1$ steps as well. Since k is a an arbitrarily chosen integer it follows that $\widehat{M}$ also loops on ω — as needed to establish the claim in this case too.

Since the claim has been established in all possible cases, this completes its proof. □

Introducing Encodings

As noted above, instances of both problems are encoded using the alphabet Σ_{TM} and the encoding scheme, for Turing machines and their input strings, described in the preparatory material for Lecture #15. The languages of instances, the languages of Yes-instances, and the languages of No-instances for these decision problems (using this encoding scheme) are as described at the beginning of this document. In particular, the language of instances for both of these problems is the language $L_{\text{TM+I}}$ that was introduced in the preparatory material for Lecture #15.

- As noted above, this language is decidable.
- This language is not equal to $\Sigma^{\star}_{\text{TM}}$. For example, the empty string, λ, does not belong to $L_{\text{TM+I}}$.

Thus the "usual situation" that is described in the preparatory material for Lecture #17 is applicable, here — and the empty string, λ, can be set to be the string "x_{Junk}" when applying the process described in the preparatory material for Lecture #17.

Completing the Definition of f

The total function f, that will be proved to be a many-one reduction from A_{TM} to *Halt*$_{\text{TM}}$, can be now defined as follows. Let $\mu \in \Sigma^{\star}_{\text{TM}}$.

- If $\mu \in L_{\text{TM+I}}$ then μ is the encoding of a Turing machine M (with some input alphabet Σ) and an input string $\omega \in \Sigma^{\star}$, for M. In this case, $f(\mu)$ should be set to be the encoding of the Turing machine $\widehat{M}$ (with the same input Σ) and string $\widehat{\omega} = \omega$ that (M, ω) are mapped to, using the above mapping φ.
- If $\mu \in L_{\text{TM+I}}$ then $f(\mu)$ should be set to be the string $x_{\text{Junk}} = \lambda$ that was chosen above.

Proving Correctness

Claim 3. *Let $\mu \in \Sigma^{\star}_{\text{TM}}$. If $\mu \in$ Halt$_{TM}$ then $f(\mu) \in A_{TM}$.*

Proof. Let $\mu \in \Sigma^{\star}_{\text{TM}}$ such that $\mu \in$ *Halt*$_{\text{TM}}$. Then $\mu \in L_{\text{TM+I}}$, since *Halt*$_{\text{TM}} \subseteq L_{\text{TM+I}}$, so that μ is an encoding of a Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

and an input string $\omega \in \Sigma^\star$ for M. It follows by the definition of f, given above, that $f(\mu)$ is encoding of the Turing machine

$$\widehat{M} = (Q, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

that is described above and the same input string ω — that is, $f(\mu)$ is the encoding of the Turing machine and input string obtained from M and ω by applying the above mapping φ.

Now, since $\mu \in \textit{Halt}_{\text{TM}}$ and μ encodes M and ω, M halts when executed on input ω — and (since $\widehat{\omega} = \omega$) it follows, by Claim 1, above, that $\widehat{M}$ accepts ω. Since $f(\mu)$ is the encoding of $\widehat{M}$ and ω, $f(\mu) \in A_{\text{TM}}$.

Since μ was arbitrarily chosen from $\Sigma^\star_{\text{TM}}$ such that $\mu \in \textit{Halt}_{\text{TM}}$ this establishes the claim. □

Claim 4. *Let $\mu \in \Sigma^\star_{\text{TM}}$. If $\mu \notin \textit{Halt}_{TM}$ then $f(\mu) \notin A_{TM}$.*

Proof. Let $\mu \in \Sigma^\star_{\text{TM}}$ such that $\mu \notin \textit{Halt}_{\text{TM}}$. Then either $\mu \in L_{\text{TM+I}}$ but $\mu \notin \textit{Halt}_{\text{TM}}$, or $\mu \notin L_{\text{TM+I}}$. These cases are considered separately below.

- If $\mu \in L_{\text{TM+I}}$ but $\mu \notin \textit{Halt}_{\text{TM}}$ then μ is an encoding of a Turing machine

  $$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

  and an input $\omega \in \Sigma^\star$ for M. It follows by the definition of f, given above, that $f(\mu)$ is the encoding of the Turing machine

  $$\widehat{M} = (Q, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

  described above and the same input string ω — that is, $f(\mu)$ is the encoding of the Turing machine and input string obtained from M and ω by applying the above mapping φ.

  Now, since $\mu \notin \textit{Halt}_{\text{TM}}$, M's execution on ω does not halt and (since $\widehat{\omega} = \omega$) it follows, by Claim 1, above, that $\widehat{M}$ does not accept ω. Thus $f(\mu) \notin A_{\text{TM}}$ in this case.

- On the other hand, if $\mu \notin L_{\text{TM+I}}$ then it follows by the definition of f, above, that $f(\mu) = \mu_{\text{Junk}} = \lambda$, so that $f(\mu) \notin A_{\text{TM}}$ in this case, as well.

Since $f(\mu) \notin A_{\text{TM}}$ in every possible case, and μ was arbitrarily chosen from $\Sigma^\star_{\text{TM}}$ such that $\mu \notin \textit{Halt}_{\text{TM}}$, this establishes the claim. □

It follows by Claims #3 and #4, above, that $\mu \in \textit{Halt}_{\text{TM}}$ if and only if $f(\mu) \in A_{\text{TM}}$ for all $\mu \in \Sigma^\star_{\text{TM}}$.

On input $\mu \in \Sigma^{\star}_{\mathsf{TM}}$ {

1. `if` $(\mu \in L_{\mathsf{TM+I}})$ {
2. return $g(\mu)$ where the function g is as in Lemma 5

} `else` {

3. return λ

}

}

Figure 1: A High-Level Algorithm to Compute the Function f

Establishing that f is Computable

It will be helpful to examine the encoding scheme, for Turing machines and input strings for them, that is described in the Lecture #15.

Consider the symbols $\mathtt{Y}, \mathtt{N} \in \Sigma_{\mathsf{TM}}$.

- The symbol "$\mathtt{Y}$" is used as part of the encoding, "$\mathtt{qY}$", of the accepting state of a Turing machine. This symbol is not used in any other part of the encoding of either a Turing machine, or an input string for it.
- The symbol "$\mathtt{N}$" is used as part of the encoding, "$\mathtt{qN}$", of the rejecting state of a Turing machine. This symbol is not used in any other part of the encoding of either a Turing machine, or an input string for it.

These claims are easily checked by reviewing the encoding scheme mentioned above. They imply the following.

Lemma 5. *Let $g : \Sigma^{\star}_{\mathsf{TM}} \rightarrow\!\!\!\!\rightarrow \Sigma^{\star}_{\mathsf{TM}}$ such that, for all $\mu \in \Sigma^{\star}_{\mathsf{TM}}$, $g(\mu)$ is obtained from μ by replacing all copies of "$\mathtt{N}$" by copies of "$\mathtt{Y}$". That is, if*

$$\mu = \alpha_1 \alpha_2 \ldots \alpha_n$$

for a non-negative integer n, then

$$g(\mu) = \widehat{\alpha}_1 \widehat{\alpha}_2 \ldots \widehat{\alpha}_n$$

where, for $1 \leq i \leq n$,

$$\widehat{\alpha}_i = \begin{cases} \mathtt{Y} & \text{if } \alpha_i = \mathtt{N}, \\ \alpha_i & \text{if } \alpha_i \neq \mathtt{N}. \end{cases}$$

If $\mu \in L_{\mathsf{TM+I}}$ then $f(\mu) = g(\mu)$.

On input $\mu \in \Sigma^{\star}_{\mathsf{TM}}$ {

1. Sweeping right and back to the left (marking the leftmost cell of the left tape at the beginning and unmarking it again at the end) write a copy of μ onto the second tape — so that μ is stored on both tapes (with copies of "$\sqcup$") to the right and both tape heads are at their leftmost cells when this step ends.
2. Use $M_{\mathsf{TM+I}}$ to decide whether $\mu \in L_{\mathsf{TM+I}}$ — moving to step 3 (instead of accepting) if $\mu \in L_{\mathsf{TM+I}}$, and moving to step 4 (instead of accepting) if $\mu \notin L_{\mathsf{TM+I}}$.
3. Sweeping right and back to the left (marking the leftmost cell of the first tape and unmarking it again at the end) write a copy of $g(\mu)$ on to the second tape. That is, when a non-blank symbol σ is visible on the first tape, write the symbol

$$\widehat{\sigma} = \begin{cases} \mathtt{Y} & \text{if } \sigma = \mathtt{N}, \\ \sigma & \text{if } \sigma \neq \mathtt{N} \end{cases}$$

   — moving right on both tapes (and leaving the copy of σ on the first tape unchanged) when sweeping to the right — so that μ is stored on the first tape (with copies of "$\sqcup$" to the right), $g(\mu)$ is stored on the second tape (with copies of "$\sqcup$" to the right), and both tape heads are at their leftmost cells.
4. Sweeping right and then left on the first tape, erase all the non-blank symbols on the first tape without changing the contents of the second tape or moving its tape head.

Figure 2: Implementation-Level Description of a Turing Machine to Compute f

How to Prove This. As noted above, the lemma is a straightforward consequence of the details of the encoding scheme that are given before it — and its proof is left as an exercise. □

Now consider the algorithm shown in Figure 1 on page 7. It is easily proved using the definition of f, and Lemma 5, that this algorithm (correctly) computes the function f.

Next consider the "implementation-level" description shown in Figure 2, above — which concerns a 2-tape (deterministic) Turing machine. This uses a component, $M_{\mathsf{TM+I}}$, that uses the second tape to decide whether the string, written on this tape, belongs to $L_{\mathsf{TM+I}}$ and — which cleans up after itself, so that the location of the tape head and contents of the first tape are not

changed when $M_{\mathsf{TM+I}}$ is executed, but the second tape head is filled with copies of "$\sqcup$" and the tape head for the second tape is resting at the leftmost cell when the execution of $M_{\mathsf{TM+I}}$ ends. This "implementation-level" description implements (and adds more detail for) the algorithm in Figure 1: Steps #1 and #2 of the algorithm in Figure 2 implement the test at one #1 in the algorithm in Figure 1. If the input string, μ, is in $L_{\mathsf{TM+I}}$ (so that the test is passed) then steps $\#3$ and $\#4$ of the algorithm in Figure 2 are executed; since these end with the first tape filled with copies of "$\sqcup$", the second tape storing $g(\mu)$, and with both tape heads at their leftmost cells, these steps implement the step at line #2 of the algorithm in Figure 1. On the other hand, if $\mu \notin L_{\mathsf{TM+I}}$, then only the step at line #4 is executed; since this ends with both tapes filled with copies of "$\sqcup$" and the tape heads at their leftmost cells, the execution of this step provides an implementation of the step at line #3 of the algorithm in Figure 1. Thus the algorithm in Figure 2 also correctly computes the function f.

Exercise: Give a formal description of a 2-tape Turing machine that implements the algorithm in Figure 2 and sketch a (reasonably brief) proof that your Turing machine is correct.

If you have computed this exercise then you will have completed a (somewhat informal) proof of the following.

Claim 6. *The function f is computable.*

Finishing Up

It follows by Claims 3, 4, and 6 that f is a many-one reduction from $\mathit{Halt}_{\mathsf{TM}}$ to A_{TM} — establishing that

$$\mathit{Halt}_{\mathsf{TM}} \preceq_{\mathsf{M}} A_{\mathsf{TM}}.$$

Thus the "Halting Problem" is reducible to the "Acceptance" problem, as claimed.

Note: Unfortunately, this reduction is not very useful; in order to show that the "Halting Problem" is undecidable using the fact that the "Acceptance Problem" is undecidable (using many-one reduction) we would need to prove that $A_{\mathsf{TM}} \preceq_{\mathsf{M}} \mathit{Halt}_{\mathsf{TM}}$ instead. This will be considered in upcoming lecture presentations.