Computer Science 313

Many-One Reductions

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #15

Many-One Reductions

Let Σ_1 and Σ_2 be two alphabets (possibly the same) and let $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ be two languages over these alphabets.

Definition: A ***many-one reduction*** from L_1 to L_2 is a ***total*** function

$$f : \Sigma_1^\star \to \Sigma_2^\star$$

such that the following properties are satisfied.

(a) For every string $\omega \in \Sigma_1^\star$, $\omega \in L_1$ *if and only if* $f(\omega) \in L_2$.

(b) The function f is computable.

We will say that L_1 is ***many-one reducible*** to L_2, and write

$$L_1 \preceq_{\mathrm{M}} L_2$$

if a many-one reduction from L_1 to L_2 exists.

Many-One Reductions

- It might help to think of a many-one reduction as being like a ***signal converter***:

It is, effectively, converting an instance of *one* problem into an instance of *another* problem that has the same solution as the instance it was given.

An Example of a Many-One Reduction

Recall that

- $\text{TM} = \{\zeta \in \Sigma_{\text{TM}}^{\star} \mid$
  ζ is a valid encoding of a Turing machine $M\}$

- $\text{TM+I} = \{\zeta \in \Sigma_{\text{TM}}^{\star} \mid \zeta$ is a valid encoding of a
  Turing machine M and input string ω for $M\}$

- A_{TM}, the subset of TM+I including valid encodings of Turing machines M and input strings ω for M such that M accepts ω.

It has already been argued that TM and TM+I are both ***decidable***. On the other hand, A_{TM} is ***recognizable*** but also ***undecidable***.

An Example of a Many-One Reduction

Now consider another language:

- $\mathrm{HALT}_{\mathrm{TM}}$, the subset of TM+I including valid encodings of Turing machines *M* and input strings ω for *M* such that *M* ***halts*** when executed on input ω.

An Example of a Many-One Reduction

Consider a function

$$f_1 : \Sigma_{\mathrm{TM}}^{\star} \to \Sigma_{\mathrm{TM}}^{\star}$$

that is defined as follows, for an input $\zeta \in \Sigma_{\mathrm{TM}}^{\star}$.

- If $\zeta \in \Sigma_{\mathrm{TM}}^{\star}$ and $\zeta \notin$ TM+I then $f_1(\zeta) = \zeta$.
- Suppose, instead, that $\zeta \in$ TM+I — so that ζ encodes some Turing machine

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

  with an input alphabet Σ and some string $\omega \in \Sigma^{\star}$.

Example of a Many-One Reduction

- Let

$$M_1 = (Q, \Sigma, \Gamma, \widehat{\delta}, q_0, q_{\text{accept}}, q_{\text{reject}})$$

with the same set of states, input alphabet, tape alphabet, start state, accepting state and halting state, but where, for $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and $\sigma \in \Gamma$,

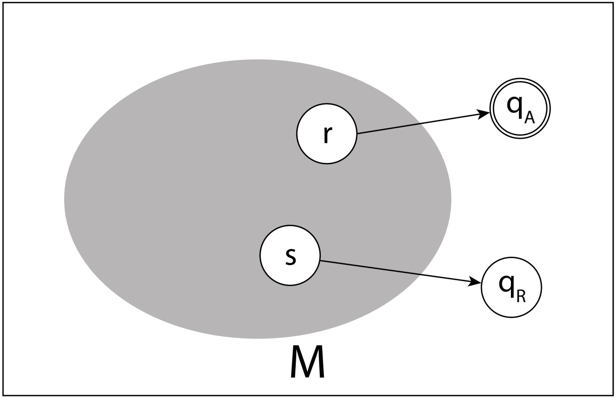
$$\widehat{\delta}(q, \sigma) = \begin{cases} \delta(q, \sigma) & \text{if } \delta(q, \sigma) = (r, \tau, m) \text{ where } r \neq q_{\text{reject}}, \\ (q_{\text{accept}}, \tau, m) & \text{if } \delta(q, \sigma) = (q_{\text{reject}}, \tau, m) \end{cases}$$

where $r \in Q$, $\tau \in \Gamma$, and $m \in \{\mathtt{L}, \mathtt{R}\}$ in the above definition.

Thus transitions to the ***rejecting*** state are replaced with similar transitions to the ***accepting*** state in M_1, and everything else is the same.

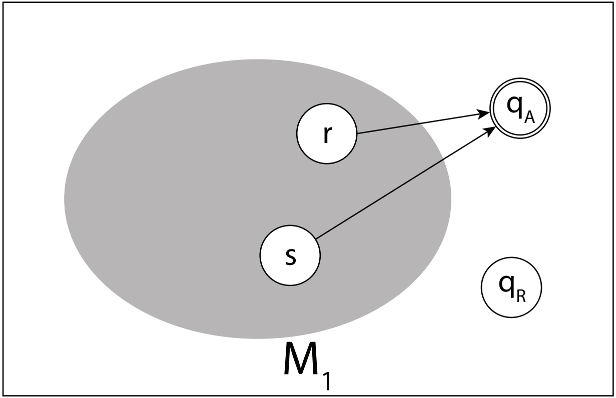
Example of a Many-One Reduction

That is, if *M* looks like this...



Example of a Many-One Reduction

Then M_1 looks like this, instead...



- Now, if ζ encodes M and ω, let $f_1(\zeta)$ be a string in $\Sigma_{TM}^{\star}$ that encodes M_1 and ω, instead.

Example of a Many-One Reduction

Claim #1: If $\zeta \in \Sigma_{\text{TM}}^{\star}$ and $\zeta \in \text{HALT}_{\text{TM}}$ then $f_1(\zeta) \in \text{A}_{\text{TM}}$.

Proof: Suppose that $\zeta \in \text{HALT}_{\text{TM}}$. Then $\zeta \in$ TM+I and ζ encodes some Turing machine

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

and string $\omega \in \Sigma^{\star}$ such that M halts when it is executed on input ω.

Let $f_1(\zeta)$ be as described, so that $f_1(\zeta)$ encodes the above Turing machine M_1 and the input string ω.

Since M halts when executed on the input ω either M accepts ω or M rejects ω. These cases are considered separately.

Example of a Many-One Reduction

Case: M accepts ω.

- In this case M_1 accepts ω too, because M_1 follows exactly the same sequence of configurations as M does.
- Since $f_1(\zeta)$ encodes M_1 and ω it now follows that $f_1(\zeta) \in \mathrm{A_{TM}}$ as claimed.

Example of a Many-One Reduction

Case: M rejects ω.

- Consider the penultimate (second-to-last) configuration that M reaches when executed on input ω. M_1 reaches this configuration too.
- However, if M is in state $q \in Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ at this point and a symbol $\sigma \in \Gamma$ is visible on M's tape then — since M rejects ω in its next move — M continues by applying a transition

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

for some symbol $\tau \in \Gamma$ and for $m \in \{\mathtt{L}, \mathtt{R}\}$.

Example of a Many-One Reduction

- M_1 must continue, instead, by applying a transition

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

  so that M_1 ***accepts*** ω in its next step, instead.

- Once again, since $f_1(\zeta)$ encodes M_1 and ω, it follows that $f_1(\zeta) \in \mathrm{A}_{\mathrm{TM}}$ in this case too — as needed to complete the proof of the claim. $\square$

Example of a Many-One Reduction

Claim #2: If $\zeta \in \Sigma_{\mathrm{TM}}^{\star}$ and $\zeta \notin \mathrm{HALT_{TM}}$ then $f_1(\zeta) \notin \mathrm{A_{TM}}$.

Proof:

- Suppose that $\zeta \in \Sigma_{\mathrm{TM}}^{\star}$ and $\zeta \notin \mathrm{HALT_{TM}}$. Then either $\zeta \notin$ TM+I or $\zeta \in$ TM+I but $\zeta \notin \mathrm{HALT_{TM}}$; these cases are considered separately.

Case: $\zeta \notin$ TM+I.

- In this case $f_1(\zeta) = \zeta \notin$ TM+I, so that $f_1(\zeta) \notin \mathrm{A_{TM}}$, as required.

Example of a Many-One Reduction

Case: $\zeta \in$ TM+I but $\zeta \notin \text{HALT}_{\text{TM}}$.

- In this case ζ encodes the Turing machine M and input string ω as described above.
- In this case M loops on ω.
- However, M_1 loops on ω too. Indeed, M_1 follows the same infinite sequence of transitions on the input ω as M does.
- Since $f_1(\zeta)$ encodes M_1 and ω it follows that $f_1(\zeta) \notin \text{A}_{\text{TM}}$ in this case too — as needed to complete the proof of this claim. $\square$

Example of a Many-One Reduction

Claim #3: The total function $f_1 : \Sigma_{\text{TM}}^{\star} \to \Sigma_{\text{TM}}^{\star}$ is a computable total function.

Proof: It follows from its definition that f_1 is a total function from $\Sigma_{\text{TM}}^{\star}$ to $\Sigma_{\text{TM}}^{\star}$. It remains to prove that f_1 is also a ***computable*** function.

- Recall that the language TM+I is decidable, so that it is possible to include a test

  $$\texttt{if}\ (\zeta \in \text{TM+I})$$

  as part of an algorithm that computes f_1.

- Now, if $\zeta \notin \text{TM+I}$ then $f_1(\zeta) = \zeta$. The identity function is *certainly* computable - so it remains only to prove that $f_1(\zeta)$ is also computable when $\zeta \in \text{TM+I}$.

Example of a Many-One Reduction

- Suppose, now, that $\zeta \in$ TM+I. Then — as described in Lecture #12 — ζ has the form

$$(\nu\,,\rho) \tag{1}$$

  where ν encodes a Turing machine and ρ encodes an input string for this Turing machine.

  The encoding, ρ, does not include any commas, so that the comma between ν and ρ, shown above, is the *rightmost* comma in ζ — making the substrings ν and ρ easy to find.

Example of a Many-One Reduction

- As described in Lecture #12, if ν encodes a Turing machine

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

then ν has the form

$$(\alpha\,,\beta\,,\gamma\,,\varphi) \tag{2}$$

where

  - α encodes the set Q of states in M;
  - β encodes the input alphabet Σ;
  - γ encodes the tape alphabet Γ; and
  - φ encodes the transition function δ.

The substrings α, β and γ do not include any commas — so that the commas separating the substrings, above, are the first three commas in ν. This makes the substrings α, β, γ and δ easy to find.

Example of a Many-One Reduction

- A consideration of the description of encodings of transition functions, previously supplied, should confirm that it is easy to produce a string $\widehat{\varphi}$ encoding the transition function for M_1 from the string φ encoding the transition function for M: All that you need to do is replace occurrences of $\mathtt{N}$ in φ with occurrences of $\mathtt{Y}$ in $\widehat{\varphi}$ — leaving all other symbols unchanged.

Example of a Many-One Reduction

- A string $\widehat{\nu}$ encoding M_1 can be computed from ν that encodes M as well — for $\widehat{\nu}$ has the form

$$(\alpha, \beta, \gamma, \widehat{\varphi})$$

  where α, β and γ are as shown at line (2), above.

- It now follows that $f_1(\zeta)$ is computable from ζ: $f_1(\zeta)$ has the form

$$(\widehat{\nu}, \rho)$$

  where ρ is as shown at line (1), above.

This completes the proof of Claim #3. $\square$

Example of a Many-One Reduction

Since all properties of a “many-one reduction” have now been established it follows that the above function

$$f_1 : \Sigma_{\mathrm{TM}}^{\star} \rightarrow \Sigma_{\mathrm{TM}}^{\star}$$

is a many-one reduction from $\mathrm{HALT}_{\mathrm{TM}}$ to A_{TM}.

Thus

$$\mathrm{HALT}_{\mathrm{TM}} \preceq_{\mathrm{M}} \mathrm{A}_{\mathrm{TM}}.$$

Process Followed To Provide a Mapping Reduction

To prove that a language $L_1 \subseteq \Sigma_1^\star$ is many-one reducible to a language $L_2 \subseteq \Sigma_2^\star$,

1. Clearly and precisely describe a ***total*** function $f : \Sigma_1^\star \to \Sigma_2^\star$.
2. ***Prove*** that if $x \in L_1$ then $f(x) \in L_2$ for every string $x \in \Sigma^\star$.
3. ***Prove*** that if $x \notin L_1$ then $f(x) \notin L_2$ for every string $x \in \Sigma^\star$.
4. ***Sketch a Proof*** that f is computable — including enough detail for it to be reasonably clear that you really *could* write a Python or Java program that computes this function from strings to strings.

This process has been followed in the above example.

Mistakes To Watch For and Avoid

- Giving a definition of f that is vague, ambiguous, or just-plain-unreadable.
- Defining a ***partial*** function from $\Sigma_1^\star$ to $\Sigma_2^\star$ (that is not defined for every string $x \in \Sigma_1^\star$) instead of a ***total function***.
- Forgetting about step 3, above — It is *not* sufficient just to show that if $x \in L_1$ then $f(x) \in L_2$.
- Failing to include enough detail at the end for it to be clear that your function f really ***is*** computable — sometimes because f has not been clearly defined and sometimes because it has, but f is not actually computable at all!

The Set of Many-One Reductions Forms a Reducibility

- Recall that a ***reducibility*** is any binary relation $\preceq_{\mathrm{Q}}$ between languages (possibly over different alphabets) such the following properties are satisfied.
  (a) $L \preceq_{\mathrm{Q}} L$ 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 $L_1 \preceq_{\mathrm{Q}} L_2$ and $L_2 \preceq_{\mathrm{Q}} L_3$ then $L_1 \preceq_{\mathrm{Q}} L_3$.
- One kind of reducibility — the set of all ***oracle reductions*** between languages was introduced in the previous lecture.

The Set of Many-One Reductions Forms a Reducibility

Claim #4: The set of many-one reductions forms a reducibility.

- This means that $L \preceq_{\mathrm{M}} L$ for every language $L \subseteq \Sigma^\star$ (and 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 $L_1 \preceq_{\mathrm{M}} L_2$ and $L_2 \preceq_{\mathrm{M}} L_3$ then $L_1 \preceq_{\mathrm{M}} L_3$.
- A proof of Claim #4 is given in a supplemental document for this lecture.

A Relationship Between Reducibilities

Claim #5: Let $L_1 \subseteq \Sigma_1^\star$ and let $L_2 \subseteq \Sigma_2^\star$. If $L_1 \preceq_{\mathrm{M}} L_2$ then $L_1 \preceq_{\mathrm{O}} L_2$.

Proof: Let $L_1 \subseteq \Sigma_1^\star$ and let $L_2 \subseteq \Sigma_2^\star$ such that $L_1 \preceq_{\mathrm{M}} L_2$.

- Then there exists a total function $f : \Sigma_1^\star \to \Sigma_2^\star$ such that $\omega \in L_1$ if and only if $f(\omega) \in L_2$ for all $\omega \in \Sigma_1^\star$ such that f is computable.
- Consider an oracle Turing machine with an oracle for L_2 that does the following when given an input string $\omega \in \Sigma_1^\star$: Compute $f(\omega)$, writing this onto the query tape and enter the query state. If the oracle Turing machine is in its "Yes" state immediately after that then ***accept*** ω. Otherwise ***reject*** ω.
- Comparisons of definitions confirms that this gives an oracle reduction from L_1 to L_2 — as needed to establish the claim. $\square$

Closure Properties

Claim: #6 Suppose that $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2) are languages such that $L_1 \preceq_{\mathrm{M}} L_2$. If L_2 is decidable then L_1 is decidable too.

Claim #7: Suppose that $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2) are languages such that $L_1 \preceq_{\mathrm{M}} L_2$. If L_2 is recognizable then L_1 is recognizable too.

- Proofs of Claim #5 and #6 are given in a supplemental document for this lecture.

Closure Properties

The following are “corollaries” of Claim #6 and of Claim #7, respectively.

Corollary #8: Suppose that $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2) are languages such that $L_1 \preceq_{\mathrm{M}} L_2$. If L_1 is undecidable then L_2 is undecidable too.

Corollary #9: Suppose that $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2) are languages such that $L_1 \preceq_{\mathrm{M}} L_2$. If L_1 is unrecognizable then L_2 is unrecognizable too.

Another Way to Prove Undecidability

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 $\widehat{L} \preceq_{\mathrm{M}} L$.
- Conclude, by Corollary #8, above, that L must be undecidable too.

A Way to Prove Unrecognizability

A process to prove that a language $L \subseteq \Sigma^\star$ is unrecognizable:

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

A Relationship Between Reducibilities

Claim #10: There exist languages $L_1 \subseteq \Sigma_1^\star$ and $L_2 \subseteq \Sigma_2^\star$ (for alphabets Σ_1 and Σ_2) such that $L_1 \preceq_{\mathrm{O}} L_2$ but $L_1 \not\preceq_{\mathrm{M}} L_2$.

Proof: Recall, by Claim #11 from the previous lecture, that there exist languages $L \subseteq \Sigma^\star$ and $\widehat{L} \subseteq \widehat{\Sigma}^\star$ (for alphabets Σ and $\widehat{\Sigma}$) such that L is not recognizable, $\widehat{L}$ is recognizable, and $L \preceq_{\mathrm{O}} \widehat{L}$. Let $L_1 = L$ and let $L_2 = \widehat{L}$ (so that $\Sigma_1 = \Sigma$ and $\Sigma_2 = \widehat{\Sigma}$).

- It follows by the choice of L_1 and L_2 that $L_1 \preceq_{\mathrm{O}} L_2$, as claimed.
- Suppose that $L_1 \preceq_{\mathrm{M}} L_2$. Then, since L_2 is recognizable it follows by Claim #7 that L_1 must be recognizable. However, since $L_1 = L$, L_1 is not recognizable — and it now follows by this *contradiction* that our assumption must be false. That is, $L_1 \not\preceq_{\mathrm{M}} L_2$, as needed to establish the claim. $\square$

Who Invented These?

- ***Emil Post*** was a Polish-American logician and mathematician who made significant contributions to the theory of computation.
- Many-one reductions were first used in a paper published by Post in 1944.