Chapter 5 Lecture Notes
(Start of Section 5.1: Undecidable Problems from Language Theory with some material from Section 5.3: Mapping Reducibility)

David Martin
dm@cs.uml.edu
With thanks to Giam Pecelli and modifications by Karen Daniels
Each point is a language in this Venn diagram.
Recap: accept, not accept, reject

- \( L(M) = \{ x \mid M \text{ accepts } x \} \)
- \( x \in L(M) \iff M \text{ accepts } x \)
- \( x \notin L(M) \iff M \text{ does not accept } x \)

- There are two ways to not accept:
  - M rejects \( x \Rightarrow x \notin L(M) \)
    - Easy to detect when this happens during runtime
  - M loops on \( x \Rightarrow x \notin L(M) \)
    - Hard to detect when this happens during runtime

- If M is a decider, then M never loops on any \( x \)
- If M is not a decider, then it may be very hard to determine what \( L(M) \) is.
Chapter 5: Reducibility

- We’ve learned that $A_{TM} \in \Sigma_1 - \Sigma_0$ and that $NA_{TM} \in \text{ALL} - \Sigma_1$

- There are infinitely many other problems that are undecidable or unrecognizable
  - Many of these problems are similar in spirit; we’ll first see some of these.
  - There are also such problems that seem to have nothing to do with TMs at all, such as Post Correspondence Problem PCP; we’ll see this later.
  - *Reducibility* is an operation and closure property that can be used for proving how complex a language is by relating it to a known language.
The Halting Problem $\text{HALT}^\text{TM}$

- $\text{HALT}^\text{TM} = \{ <M, w> \mid M \text{ is a TM and } M \text{ halts on the input } w \}$

  - Here, “halts on the input w” means that the TM either accepts w or rejects w; it does *not* go into an infinite loop on w.

  - Conceptually, this language should be undecidable, just like $A^\text{TM}$.

    - In $A^\text{TM}$, the question is “does the machine reach the accepting state”?

    - In $\text{HALT}^\text{TM}$, the question is “does the machine reach the accepting state or the rejecting state”?
HALT_{TM} is undecidable

**Theorem 5.1**  \( \text{HALT}_{\text{TM}} \notin \Sigma_0 \)

**Proof** by contradiction. Suppose that \( \text{HALT}_{\text{TM}} \) were decidable via a decider named \( M_1 \). We could then design a TM \( M_2 \) that efficiently transforms questions about \( A_{\text{TM}} \) (undecidable) into questions about \( \text{HALT}_{\text{TM}} \) as follows (different from textbook p. 189):

1. \( M_2 \): input \(<M,w>\) (reject if not of this form)
2. **Modify** \( M \) to \( M' \) so that transitions pointing to its reject state instead make it go into an infinite loop (why? how?)
3. Simulate the machine \( M_1 \) starting with input \(<M',w>\) until it halts:
   - If \( M_1 \) accepts \(<M',w>\), then \( M_2 \) accepts also
   - If \( M_1 \) rejects \(<M',w>\), then \( M_2 \) rejects also
   - (Note that \( M_1 \) never loops on any input (why?))
HALT$_{TM}$ is undecidable

1. $M_2$: input $<M,w>$ (reject if not of this form)
2. **Modify** $M$ to $M'$ so that transitions pointing to its reject state instead make it go into an infinite loop
3. Simulate the machine $M_1$ starting with input $<M',w>$ until it halts:
   - If $M_1$ accepts $<M',w>$, then $M_2$ accepts also
   - If $M_1$ rejects $<M',w>$, then $M_2$ rejects also
   - (Note that $M_1$ never loops on any input)

Then
- $<M,w> \in L(M_2) \Rightarrow <M',w> \in L(M_1) \Rightarrow <M,w> \in A_{TM}$
  (because $M'$ halts on $w \iff M$ accepts $w$)
- $<M,w> \notin L(M_2) \Rightarrow <M',w> \notin L(M_1) \Rightarrow <M,w> \notin A_{TM}$
  (same reason)
HALT_{\text{TM}} is undecidable

- So \( L(M_2) = A_{\text{TM}} \)
- And since \( M_1 \) is a decider, \( M_2 \) is a decider
  - But that contradicts the undecidability of \( A_{\text{TM}} \)
  - QED

- This shows that HALT_{\text{TM}} somehow contains the “hard part” of \( A_{\text{TM}} \); the other differences are not critical from a decidability point of view.

- We now describe a way to more concisely express the relationship between two languages: mapping reductions.
Section 5.3: 
Computable functions

- **Definition 5.17** A function \( f: \Sigma^* \rightarrow \Sigma^* \) is **computable** if there exists some TM \( M \) such that, for every input \( w \in \Sigma^* \),

  1. When started with input \( w \), \( M \) halts eventually
  2. At the time that \( M \) halts, its tape contains \( f(w) \) (followed by blanks, as usual)

- Note that \( f \) is defined *mathematically* and \( M \) is a *TM* that implements it
  - \( f \) must be defined for *all* strings in \( \Sigma^* \)
Section 5.3: Mapping reductions

- **Definition 5.20** Let $A$ and $B$ be languages over $\Sigma$. Then $A$ is *mapping reducible* to $B$, written “$A \leq_m B$ via $f$” if

1. There exists a computable function $f: \Sigma^* \rightarrow \Sigma^*$ and
2. For every $w \in \Sigma^*$, $w \in A \iff f(w) \in B$

- Equivalently: $w \in A \Rightarrow f(w) \in B$
  and $w \notin A \Rightarrow f(w) \notin B$

- These conditions say that it is possible to *efficiently convert* questions about $A$ into questions about $B$

- $A \leq_m B$ means that $B$ is “at least as hard as” $A$, with respect to computability (a very coarse measure)

- Fact: $\leq_m$ is reflexive and transitive (but not symmetric or antisymmetric)
Section 5.3: A partial picture of $A \leq_m B$ via $f$

Points are *strings* in this Venn diagram (not languages)

Each association is *consistent* with the statement “$A \leq_m B$ via $f$”

$f$ has to send every string somewhere; this diagram only shows 4 mappings

$f$ does *not* have to be 1-1 or onto
Section 5.3: Closure under $\leq_m$

☐ **Theorem 5.22**  If $A \leq_m B$ and $B \in \Sigma_0$ then $A \in \Sigma_0$

☐ **Theorem 5.28**  If $A \leq_m B$ and $B \in \Sigma_1$ then $A \in \Sigma_1$

☐ **Corollary 5.23**  If $A \leq_m B$ and $A \notin \Sigma_0$ then $B \notin \Sigma_0$

☐ **Corollary 5.29**  If $A \leq_m B$ and $A \notin \Sigma_1$ then $B \notin \Sigma_1$
Section 5.3: Sample proof

**Theorem 5.22** If $A \leq_m B$ and $B \in \Sigma_0$ then $A \in \Sigma_0$.

**Proof** Assume $A \leq_m B$ via $f$ and $B \in \Sigma_0$. Let $M_1$ be the TM for $f$ and $M_2$ be the decider TM for $B$. We define $M_3$ to decide $A$:

1. $M_3$: input $x$
2. simulate $M_1$ on input $x$
3. let $y$ be the contents of the tape when $M_1$ halts
   1. (Note $y = f(x)$ because $M_1$ implements the reduction $A \leq_m B$)
4. simulate $M_2$ on input $y$
   1. If $M_2$ accepts then accept
   2. If $M_2$ rejects then reject

Then $M_3$ is clearly a decider, because $M_1$ and $M_2$ always halt. And $L(M_3) = A$ because we know that $x \in A \iff f(x) \in B$  

QED
Sample Application

We carefully rephrase Theorem 5.1 using $\leq_m$:

**Theorem 5.1** $\text{HALT}_{\text{TM}} \notin \Sigma_0$

**Proof** By Corollary 5.23, it suffices to show that $A_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}$.

*(different from textbook Example 5.24, p. 208)*

First, let $M_{\text{loop}}$ be some fixed TM that goes into an infinite loop on every input it is given.
Define $M_2$ as a TM to compute a function $f: \Sigma^* \rightarrow \Sigma^*$ by:

1. $M_2$: input $x$
2. if $x$ is of the form $<M,w>$ where $M$ is a TM and $w$ is a string, then
   1. Modify $M$ to $M'$ so that transitions pointing to its reject state instead make it go into an infinite loop
   2. Halt with $<M',w>$ on the tape
3. else halt with $<M_{\text{loop}}, \varepsilon>$ on the tape

Then for strings $x$ of the form $<M,w>$:

$<M,w> \in A_{\text{TM}} \iff <M',w> \in \text{HALT}_{\text{TM}} \iff f(<M,w>) \in \text{HALT}_{\text{TM}}$

and for strings $x$ not of the form $<M,w>$:

$x \notin A_{\text{TM}}$, and $f(x) = <M_{\text{loop}}, \varepsilon> \notin \text{HALT}_{\text{TM}}$

Taken together, for all $x \in \Sigma^*$,

$x \in A_{\text{TM}} \iff f(x) \in \text{HALT}_{\text{TM}}$. So $A_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}$

QED
Comment

- The machine $M_{\text{loop}}$ was not strictly necessary in the previous example; $M_2$ could have just left $x$ on the tape if it were not of the form $<M,w>$ and things still work out OK (WHY?)
- But for some reductions you need to do something like $M_{\text{loop}}$
Review: Undecidability and reducibility

- An undecidable language is one that is not in $\Sigma_0$
- Standard example:
  $A_{TM} = L(U) = \{ <M,w> \mid M \text{ is a TM and } w \in L(M) \}$

- An unrecognizable language is one that is not in $\Sigma_1$
- Standard example:
  $NA_{TM} = \{ <M,w> \mid M \text{ is a TM and } w \notin L(M) \}$
Review: Undecidability and reducibility

- We also saw that \( \text{HALT}_{TM} = \{ <M,w> \mid M \text{ is a TM and } M \text{ halts on the input } w \} \) is undecidable.

- A mathematical function \( f: \Sigma^* \rightarrow \Sigma^* \) is \textit{computable} if there exists some TM \( M \) that transforms the TM's input into the function's output on the TM's tape.

- A is \textit{mapping reducible} to B, written \( A \leq_m B \) via \( f \) if
  1. There exists a \textit{computable} function \( f: \Sigma^* \rightarrow \Sigma^* \) and
  2. For every \( w \in \Sigma^* \), \( w \in A \iff f(w) \in B \)
     - Equivalently: \( w \in A \Rightarrow f(w) \in B \) \textbf{and} \( w \notin A \Rightarrow f(w) \notin B \)

- If A is undecidable and \( A \leq_m B \), then B is undecidable ("at least as hard as A")
Another application

Let $A_{10} = \{ <M> | M \text{ is a TM and } 10 \in L(M) \}$

(board examples)

Claim $A_{10} \in \Sigma_1 - \Sigma_0$

Proof that $A_{10}$ is undecidable: it suffices to show that $A_{TM} \leq_m A_{10}$. We want this to be true:

$\forall x \ x = <M, w> \in A_{TM} \Rightarrow f(<M, w>) \in A_{10}$

$\neg x \notin A_{TM} \Rightarrow f(x) \notin A_{10}$

So $f(x)$ should be a program $<M'>$ that accepts 10 if and only if $[ x = <M, w> \text{ and } M \text{ accepts } w ]$. 
Proof continued

Let \( M_\emptyset \) be some TM such that \( L(M_\emptyset) = \emptyset \). Then we define the TM computing \( f \):

1. \( f \): input \( x \)
2. if \( x \) is not of the form \( <M,w> \) then \( f \) prints \( <M_\emptyset> \) and halts, else
3. \( x = <M,w> \)
4. let \( M' \) be a TM defined as follows:
   1. \( M' \): input \( z \)
   2. simulate \( M \) on input \( w \) until it halts (if ever)
   3. if \( M \) accepted \( w \) then \( M' \) accepts \( z \)
   4. else if \( M \) rejected \( w \) then \( M' \) rejects \( z \)
   5. else \( M \) is looping on \( w \) so \( M' \) loops on \( z \) also
5. \( f \) prints \( <M'> \) and halts
Proof continued

- First, f is computable: the given TM just looks at its input as a string, formulates a different string, and prints it out (in step 2 or 5)
  - In particular, M' is not run yet, it is merely constructed

- <M,w> ∈ A_{TM} ⇒ L(M')=Σ* ⇒ 10 ∈ L(M') ⇒ f(<M,w>) ∈ A_{10}

- [ x not of the form <M,w> ] ⇒ x ∉ A_{TM} ⇒ f(x)=<M_∅> ∉ A_{10}
Proof continued

- Finally, $x = <M, w> \notin A_{TM} \Rightarrow L(M') = \emptyset \Rightarrow f(<M, w>) \notin A_{10}$

If $M$ loops forever on $w$, then $M'$ loops forever on $z$ as observed in step 4.5

Else if $M$ rejects $w$, then $M'$ rejects $z$ in step 4.4

So if $M$ doesn’t accept $w$, then $M'$ doesn’t accept $z$ either, no matter what $z$ is

- So $A_{TM} \leq_m A_{10}$ via $f$. 
  
  QED

Note that this proof doesn’t really depend on the string 10 in particular.
Tricky Case: $E_{TM} \notin \Sigma_0$ but $A_{TM} \leq_m E_{TM}$

- Let: $E_{TM} = \{<M>|<M>$ is a TM and $L(M) = \emptyset\}$
  
  - Approach: Set up proof by contradiction to show $E_{TM}$ is undecidable.
  
  - By way of contradiction, assume existence of decider $R$ for $E_{TM}$.
  
  - We’ll use this decider $R$ to create TM decider $S$ for $A_{TM}$, contradicting the fact that we know $A_{TM}$ is undecidable.
  
  - Subtlety: Run $R$ on modification $M_1$ of $<M>$ so that $M_1$ rejects all strings except $w$.

  - $S =$ “On input $<M,w>$:
    1. Use description of $M$ and $w$ to construct TM $M_1$.
    2. Run $R$ on input $<M_1>$
    3. If $R$ accepts, reject; if $R$ rejects, accept.

Why does this not provide mapping reducibility?
Example: $E_{TM} \leq_m EQ_{TM}$

- Let: $EQ_{TM} = \{< M_1, M_2 > | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$
  - Approach: Set up proof by contradiction to show $EQ_{TM}$ is undecidable.
  - By way of contradiction, assume existence of decider $R$ for $EQ_{TM}$.
  - We’ll use this decider $R$ to create TM decider $S$ for $E_{TM}$, contradicting the fact that we know $E_{TM}$ is undecidable.
  - Subtlety: Which 2 languages do we run $EQ_{TM}$ on?

  $S = \text{“On input } <M>:\$
  1. Construct TM $M_1$, which rejects all inputs.
  2. Run $R$ on input $< M, M_1 >$.
  3. If $R$ accepts, accepts; if $R$ rejects, reject.

Where is the mapping reducibility here?
Yet Another Tricky Case?

- **REG\textsubscript{TM} = \{<M> | M is a TM and L(M) \in \text{REG}\}**
- **Theorem 5.3** REG\textsubscript{TM} \not\in \Sigma_0 (not decidable)
- **Proof:**
  - Assume REG\textsubscript{TM} is decidable by TM R.
  - Using R, construct TM S that decides A\textsubscript{TM}.
  - S’s input is <M,w>. Create M\textsubscript{2} from M so M\textsubscript{2} recognizes a regular language iff M accepts w.
  - Key points:
    - M\textsubscript{2} recognizes nonregular language \{0^n1^n | n \geq 0\} if M does not accept w.
    - M\textsubscript{2} recognizes regular language \Sigma^* if M does accept w.
  - S = “On input <M,w>:
    - Construct M\textsubscript{2} (see next slide for further details)
    - Run R on input < M\textsubscript{2} > (Note R can decide whether or not L(M\textsubscript{2}) is regular.)
    - If R accepts, accept; if R rejects, reject.”

Sources: Sipser textbook and http://echochamber.me
Yet Another Tricky Case? (continued)

- Create $M_2$ from $M$ so $M_2$ recognizes a regular language iff $M$ accepts $w$.
  - $M_2$ recognizes nonregular language $\{0^n1^n \mid n \geq 0\}$ if $M$ does not accept $w$.
  - $M_2$ recognizes regular language $\Sigma^*$ if $M$ does accept $w$.
- **How?** We want this behavior for $M_2$:
  - On arbitrary input $x$:
    - If $M$ accepts $w$, then $M_2$ accepts
      - thus recognizing regular language $\Sigma^*$ since all strings are accepted by $M_2$ when $M$ accepts $w$
    - If $M$ does not accept $w$, then:
      - If $x \in \{0^n1^n \mid n \geq 0\}$, then $M_2$ accepts.
      - If $x \not\in \{0^n1^n \mid n \geq 0\}$, then $M_2$ rejects or loops.

---

Sources: Sipser textbook and http://echochamber.me
Yet Another Tricky Case? (continued)

☐ To fix problem, observe:
  - when $x \in \{0^n1^n \mid n \geq 0\}$ $M_2$ accepts regardless of whether $M$ accepts $w$
  - when $x \not\in \{0^n1^n \mid n \geq 0\}$ $M_2$ accepts iff $M$ accepts $w$

☐ Based on this, pull $x \in \{0^n1^n \mid n \geq 0\}$ test outside to obtain this $M_2$ behavior:

  ☐ On arbitrary input $x$:
    - If $x \in \{0^n1^n \mid n \geq 0\}$ then accept.
    - If $x \not\in \{0^n1^n \mid n \geq 0\}$ then run $M$ on input $w$ and accept if $M$ accepts $w$.

☐ $\text{REG}_{\text{TM}} \not\subseteq \Sigma_0$ QED

☐ Can we also show $\text{ATM} \leq_m \text{REG}_{\text{TM}}$?

Sources: Sipser textbook and http://echochamber.me
Each point is a language in this Venn diagram.