Decision Problems and Recursively Enumerable Languages

- Examples of Decision Problems and Corresponding Languages:

  1. Decision problem:
     Halting: Given a TM $T$ and a string $w$, does $T$ accept $w$?
     Corresponding language:
     $H = \{ e(T)e(w) | w \in L(T) \}$
     $e(T)$ is the encoding of Turing machine $T$, $e(w)$ is the encoding of input $w$ to Turing machine $T$.

  2. Decision problem:
     Self-accepting: Given a TM $T$, does $T$ accept $e(T)$ (i.e., its own encoding)?
     Corresponding language:
     $SA = \{ w | w = e(T) \text{ and } T \text{ accepts } w \}$

  3. Decision problem:
     $Accepts(\Lambda)$: Given a TM $T$, is $\Lambda \in L(T)$?
     Corresponding language:
     $L_\Lambda = \{ e(T) | T \text{ is TM accepting } \Lambda \}$
Instances of Decision Problems

• An instance of Halting: a particular TM and a particular string: $T, w$

• A yes-instance of Halting: $T, w$ and $T$ accepts $w$
  A no-instance of Halting: $T, w$ and $T$ does not accept $w$

• An instance can be represented by a string:
  $T, w$ by $<e(T), e(w)>$

• A yes-instance of a problem has a corresponding string in the language.
Reduction of one Decision Problem to Another

Definition of reduction:

- **Definition 11.3:** If P1 and P2 are decision problems, we say P1 is reducible to P2 ($P1 \leq P2$) if there is an algorithmic procedure that allows us, given arbitrary instances $I_1$ of P1, to find an instance $F(I_1)$ of P2 so that for every $I_1$, $I_1$ is a yes-instance of P1 iff $F(I_1)$ is a yes-instance of P2.

Corresponding language definition:

- **Definition 11.3a:** If $L_1$ and $L_2$ are languages over alphabets $\Sigma_1$ and $\Sigma_2$, respectively, then we say that $L_1$ is reducible to $L_2$ (denoted by $L_1 \leq L_2$) if there is a Turing-computable function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ so that for any $x \in \Sigma_1^*$

  $x \in L_1$ iff $f(x) \in L_2$

Meaning of reduction:

- **Theorem 11.4a:**

  If $L_1$ and $L_2$ are languages over alphabets $\Sigma_1$ and $\Sigma_2$, respectively, and $L_1 \leq L_2$, then if $L_2$ is recursive, $L_1$ is also recursive. (Equivalently, if $L_1$ is not recursive, neither is $L_2$.)
Theorem 11.4b:
If P1 and P2 are decision problems and $P_1 \leq P_2$, then if P2 can be solved algorithmically, so can P1. (Equivalently, if P1 is unsolvable, so is P2.)

Theorems 11.4a and 11.4b correspond to above two definitions of reduction.

We proved H is a recursively enumerable language and not a recursive language (corresponds to theorem 11.4a). Correspondingly, we can show Halting problem is an unsolvable decision problem (corresponds to theorem 11.4b).
Schematic Representation of Reduction of Decision Problems

- Define Min function in terms of Max function.
- Conceptual representation of reduction of decision problems:
  \[ P_1 \leq P_2: \]
  \[ \begin{array}{c}
  \text{w} \rightarrow | \ F(\ ) \rightarrow \text{AP2} | \rightarrow \text{yes/no} \\
  \hline
  \text{AP1}
  \end{array} \]

  \[ F(\) is an algorithm and \]
  \[ P_1’s \ algorithm (\text{AP1}) \ written \ in \ terms \ of \ F() \ and \]
  \[ P_2’s \ algorithm (\text{AP2}). \]

- If P2 is solvable then P1 is
  Or if P1 is unsolvable then then P2 is unsolvable.

- contra positive: If P1 then P2 \( \iff \) if NOT P2 then NOT P1

- \( SA \leq \text{Halting} : \)
  \[ \begin{array}{c}
  \text{w} \rightarrow | \ F(\ ) -- w, e(w) \rightarrow \text{HaltingAlg} | \rightarrow \text{yes/no} \\
  \hline
  \text{Self-acceptingAlg}
  \end{array} \]
Since Self-accepting is unsolvable, Halting is unsolvable.
Other Unsolvable Problems

- **AcceptsSomething**: Given a TM T, does T accept some string (i.e., L(T) nonempty)?
  
  Reduction: \(\text{Accepts}(\Lambda) \leq \text{AcceptsSomething}\)

  \[
  \begin{align*}
  T &\rightarrow |F() - \text{EraseTape} T\rightarrow \text{AcceptsSomething}| \rightarrow \text{yes/no} \\
  \text{Accept}(\text{Lambda})
  \end{align*}
  \]

- If T accepts \(\Lambda\) then Erase T accepts everything, that is also accepting something.

- Examples of a few other unsolvable problems:
  1. **AcceptsEverything**: Given a TM T, does T accept every string?
  2. **AcceptsRegular**: Given a TM T, is the language accepted by T is regular?
  3. And just about any with this type of property. Any question about what a TM does?
Rices’s Theorem

• Property:
  for $Accepts(\Lambda)$ property is ”containing the null string”.
  for $AcceptsEverything$: property is ”containing all strings”

• If the property is satisfied by some but not all r.e. languages (correspondingly, not all TM) then we call it a non-trivial property. Above two properties are non-trivial properties.

  Problem: ”Given a TM T, is the language accepted by T recursively enumerable?” is (trivially) solvable because it corresponds to a trivial property.

• Rice’s theorem:
  If P is a property of languages that is satisfied by some but not all recursively enumerable languages, then the decision problem
  $D_P$: Given a TM T, does L(T) have property P?
  is unsolvable.