Computability Theory **Fall 2020**

Lecture 10: Reducibility

Lecturer: Renjie Yang

10.1 Decision Problem

Definition 10.1 *The decision problem for a predicate $P(x_1, \ldots, x_n)$ is called recursively solvable if P is recursive; otherwise it is called recursively unsolvable.*

Definition 10.2 *The decision problem for a set S is called recursively solvable or unsolvable according as S is or is not recursive.*

Definition 10.3 *Let A be a set, and let Σ be a alphabet. An encoding of the elements of A, using Σ, is an injective function $Enc : A \to \Sigma^*$. We denote the encoding of $a \in A$ by $\langle a \rangle$. If $w \in \Sigma^*$ is such that there is some $a \in A$ with $w = \langle a \rangle$, then we say w is a valid encoding of an element in A. A set that can be encoded is called encodable.*

Example

- Problem: Given a DFA and a string, will the DFA accept?
- The same problem in terms of languages: Given a DFA B and a string w, is $\langle B, w\rangle$ a member of the language $A_{DFA} = \{\langle B, w\rangle | B \text{ is a DFA that accepts input string } w\}$?
- Decision problem: Is the language A_{DFA} decidable?
- The answer to this decision problem is yes. Here is a Turing machine that decides A_{DFA}:

  M_A = "On input $\langle B, w\rangle$,

  1. Check that $\langle B, w\rangle$ has length 2, $\langle B\rangle$ is an encoding of DFA. If not, reject;
  2. Simulate B on input w
  3. If the simulation ends in an accept state, accept . If it ends in a nonaccepting state, reject."

Example

- Problem: Given a DFA, will the DFA accept any string?
- The same problem in terms of languages: Given a DFA B, is B a member of the language $E_{DFA} = \{\langle B, w\rangle | B \text{ is a DFA and } L(B) = \phi\}$?
- Decision problem: Is the language E_{DFA} decidable?
- The answer to this decision problem is yes. Here is a Turing machine that decides E_{DFA}:

  M_E = "On input $\langle B, w\rangle$,

  1. Check that $\langle B\rangle$ is an encoding of DFA. If not, reject;
  2. Mark the start state of B;

3. Repeat until no new states get marked:
   Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, accept; otherwise, reject."

Example

- Problem: Given two DFAs, do they recognize the same language?
- The same problem in terms of languages: Given two DFAs A and B, is $\langle A, B\rangle$ a member of the language $EQ_{DFA} = \{\langle A, B\rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$?
- Decision Problem: Is the language EQ_{DFA} decidable?
- The answer to this decision problem is yes. Here is a Turing machine that decides EQ_{DFA}:

  $M_{EQ} =$ "On input $\langle A, B\rangle$,

  1. Check that $\langle A, B\rangle$ has length 2, $\langle B\rangle$ and $\langle A\rangle$ are encodings of DFA. If not, reject;
  2. Construct a DFA C such that $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$;
  3. Run TM M_E on input $\langle C\rangle$;
  4. If M_E accepts, accept. If M_E rejects, reject."

Example

- Problem: Given a Turing machine and a string, will the Turing machine accept?
- The same problem in terms of languages: Given a TM M and a string w, is $\langle M, w\rangle$ a member of the language $A_{TM} = \{\langle M, w\rangle | M \text{ is a TM that accepts input string } w\}$?
- Decision problem: Is the language A_{TM} decidable?
- The answer to this decision problem is NO.

Theorem 10.4 *A_{TM} is undecidable.*

Proof: Assume that A_{TM} is decidable. Then there exists a Turing machine H which is a decider for A_{TM}:

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept}, & \text{if } M \text{ accepts } w, \\ \textit{reject}, & \text{if } M \text{ does not accept } w. \end{cases}$$

Construct a new Turing machine D as follows:

$$D(\langle M\rangle) = \begin{cases} \textit{accept}, & \text{if } M \text{ does not accept } \langle M\rangle, \\ \textit{reject}, & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

Now apply D on $\langle D\rangle$:

$$D(\langle D\rangle) = \begin{cases} \textit{accept}, & \text{if } D \text{ does not accept } \langle D\rangle, \\ \textit{reject}, & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

Contradiction. Therefore the assumption that A_{TM} is decidable is false. ∎

Example

- Given a Turing machine and a string, will the Turing machine halt?
- The same problem in terms of languages: Given a Turing machine M and a string w, is $\langle M, w\rangle$ a member of the language $HALT_{TM} = \{\langle M, w\rangle | M$ is a TM that halts on string $w\}$?
- Decision problem: Is the language $HALT_{TM}$ decidable?
- The same decision problem in terms of predicates: Is the predicate $P_Z(x) \leftrightarrow$ "x is the Gödel number of an instantaneous description α of Z and there exists a computation of Z that begins with α'' computable or recursively solvable?
- The answer to this decision problem is NO.

Theorem 10.5 *$HALT_{TM}$ is undecidable.*

Proof: Assume a Turing machine R decides $HALT_{TM}$. Construct another Turing machine S to decide A_{TM}:
$S =$ "On input $\langle M, w\rangle$,

1. Check that $\langle M, w\rangle$ has length 2, $\langle M\rangle$ is an encoding of a Turing machine. If not, reject;
2. Run Turing machine R on input $\langle M, w\rangle$;
3. If R rejects, reject;
4. If R accepts, simulate M on w until it halts.
5. If M has accepted, accept. If M has rejected, reject."

Here is another proof. Let Z_0 be such that $\Psi_{Z_0}(x) = min_y T(x, x, y)$. Then x belongs to the domain of $\Psi_{Z_0}(x)$ if and only if $\exists y T(x, x, y)$. But x belongs to the domain of $\Psi_{Z_0}(x)$ if and only if $P_{Z_0}(gn(q_1\overline{x}))$. Hence, if $P_{Z_0}(x)$ were computable, so would be the domain of $\Psi_{Z_0}(x)$, and hence also the predicate $\exists y T(x, x, y)$. But $\exists y T(x, x, y)$ is not computable, contradiction. ■

Example

- Problem: Does a Turing machine accept any string?
- The same problem in terms of languages: Given a Turing machine M, is $\langle M\rangle$ a member of the language $E_{TM} = \{\langle M\rangle | M$ is a Turing machine and $L(M) = \phi\}$?
- Decision problem: Is the language E_{TM} decidable?
- The answer to this decision problem is NO.

Theorem 10.6 *E_{TM} is undecidable.*

Proof: We will use a modification of M constructed as follows:
$M_1 =$ "On input $\langle x\rangle$,

1. If $x \neq w$, reject;
2. If $x = w$, run M on input w and accept if M does."

Assume that Turing machine R decides E_{TM}. We can construct a Turing machine S that decides A_{TM} as follows:
$S =$ "On input $\langle M, w\rangle$,

1. Check that $\langle M, w\rangle$ has length 2, $\langle M\rangle$ is an encoding of a Turing machine. If not, reject;
2. Use the description of M and w to construct the Turing machine M_1 described above;
3. Run R on input $\langle M_1\rangle$;
4. If R accepts, reject; if R rejects, accept."

If R were a decider for E_{TM}, S would be a decider for A_{TM}. A decider for A_{TM} does not exist, contradiction. Therefore E_{TM} is undecidable. ■

Example

- Problem: Given two Turing machines, do they accept the same language?
- The same problem in terms of languages: Given Turing machines M_1 and M_2, is $\langle M_1, M_2\rangle$ a member of the language $EQ_{TM} = \{\langle M_1, M_2\rangle | M_1$ and M_2 are Turing machines and $L(M_1) = L(M_2)\}$?
- Decision problem: Is the language EQ_{TM} decidable?
- The answer to this decision problem is NO.

Theorem 10.7 *EQ_{TM} is undecidable.*

Proof: Assume That Turing machine R decides EQ_{TM}. Construct a decider S of E_{TM} as follows:
$S =$ "On input $\langle M\rangle$,

1. Check that $\langle M\rangle$ is an encoding of a Turing machine. If not, reject;
2. Run R on input $\langle M, M_1\rangle$, where M_1 is a Turing machine that rejects all inputs.
3. If R accepts, accept; if R rejects, reject."

If R decides EQ_{TM}, S decides E_{EM}. But E_{TM} is undecidable, contradiction. Therefore EQ_{TM} is undecidable. ■

10.2 Reducibility

Definition 10.8 *Let A and B be sets. then A is said to be many-one reduction (or mapping reduction) to B, written $A \leqslant_m B$, if there is a computable function f such that for every natural number x,*

$$x \in A \text{ if and only if } f(x) \in B$$

Example Define $K = \{x | \varphi_x(x)\}$, $K_0 = \{\langle i, x\rangle | \varphi_i(x) \downarrow\}$. There is a computable function $f : x \to \langle x, x\rangle$ such that $x \in K$ if and only if $\langle x, x\rangle \in K_0$. Therefore $K \leqslant_m K_0$.

Theorem 10.9 *If $A \leqslant_m B$ and $B \leqslant_m C$, then $A \leqslant_m C$*

Proof: Let f be the reduction function of A to B, g be the reduction function of B to C. Then

$$x \in A \text{ if and only if } f(x) \in B \text{ if and only if } g \circ f(x) \in C.$$

$g \circ f$ is the reduction function of A to C. ■

Theorem 10.10 *Let A and B be any sets, $A \leqslant_m B$.*

- *If B is computably enumerable, so is A.*
- *If B is computable, so is A.*

Proof: If B is the domain of partial function g, then A is the domain of $g \circ f$:

$$x \in A \leftrightarrow f(x) \in B \leftrightarrow g(f(x)) \downarrow$$

Thus the first claim is true.

For the second claim, since $x \in A \leftrightarrow f(x) \in B$, $C_A(x) = C_B(f(x))$ for any x, $C_A = C_B \circ f$. If C_B is computable, then C_A is also computable. ■

Example Let $K_1 = \{e | \varphi_e(0)\}$. K_1 is computably enumerable but not computable.

Proof: It is suffices to show that K_0 is reducible to K_1. Since T predicate is primitive recursive, according to the normal form theorem for r.e. sets, K_1 is computably enumerable. To show that K_1 is not computable, let f be the 3-ary function defined by

$$f(x, y, z) \simeq \varphi_x(y).$$

Pick an index e such that $f = \varphi_e^3$, we have

$$\varphi_e^3(x, y, z) \simeq \varphi_x(y).$$

By the s-m-n theorem, there is a function $s(e, x, y)$ (more precisely, $s_1^2(e, x, y)$) such that, for every z,

$$\varphi_{s(e,x,y)}(z) \simeq \varphi_e^3(x, y, z) \simeq \varphi_x(y).$$

$s(e, x, y)$ is an index for the machine that, for any input z, ignores that input and computes $\varphi_x(y)$. In particular, we have

$$\varphi_{s(e,x,y)}(0) \downarrow \text{ if and only if } \phi_x(y) \downarrow .$$

Therefore, $\langle x, y \rangle \in K_0$ if and only if $s(e, x, y) \in K_1$. So the function g defined by

$$g(w) = s(e, K(w), L(w))$$

is a reduction of K_0 to K_1. ■

Example Let $Tot = \{x | \text{ for every } y, \varphi_x(y) \downarrow\}$. Then Tot is not computable.

Proof: It is suffices to show that K is reducible to Tot. Define $h(x, y)$ as

$$h(x, y) \simeq \begin{cases} 0, & \text{if } x \in K \\ \text{undefined}, & \text{otherwise} \end{cases}$$

$h(x, y)$ is just $N(U(min_s T(x, x, s)))$, so it is partially computable. By the s-m-n theorem, there is a primitive recursive function $s(x)$ such that for every x and y,

$$\varphi_{s(x)}(y) \simeq \begin{cases} 0, & \text{if } x \in K \\ \text{undefined}, & \text{otherwise} \end{cases}$$

So $\phi_{k(x)}$ is total if $x \in K$, and undefined otherwise. Thus, k is a reduction of K to Tot. ∎

Theorem 10.11 *(Rice's Theorem) Let C be any set of partial computable functions, and let $A = \{n | \varphi_n \in C\}$. If A is computable, then either C is ϕ or C is the set of all the partial computable functions.*

Proof: Let g be any function in C, and f is the function that is nowhere defined. Without loss of generality, assume $f \notin C$. Define $h(x, y) \simeq P_1^2(g(y), Un(x, x))$. That is to say,

$$h(x, y) \simeq \begin{cases} \text{undefined}, & \text{if } \varphi_x(x) \downarrow \\ g(y), & \text{otherwise} \end{cases}$$

Since $h(x, y)$ is a composition of partial computable functions, it is a partial computable function, therefore it is equal to function φ_k for some index k. By the s-m-n theorem there is a primitive recursive function s such that for each x,

$$\varphi_{s(k,x)}(y) = \varphi_k(x, y) = h(x, y).$$

Now for each x, if $\phi_x(x) \downarrow$, then $\phi_{s(k,x)}$ is the same function as g, and so $s(k, x)$ is in A. On the other hand, if $\phi_x(x) \uparrow$, then $\varphi_{s(k,x)}$ is the same function as f, so $s(k, x)$ is not in A. In other words, we have that for every x, $x \in K$ if and only if $s(k, x) \in A$. If A were computable, then K would also be computable, contradiction. So A is not computable. ∎