Reducibility

This, in general, means that you look for the solution to a problem by "restating" it in a different domain, solve the problem in the new domain and interpret the result in terms of the original problem and domain. The technique is quite common in language and NP-completeness proofs. (TMs let us conclude that every question we can ask is one about decidability - or partial decidability - of a language.)

**Def.** Let $A$ and $B$ be sets of strings over an alphabet $\Sigma$. $A$ is many-to-one reducible to $B$ ($A \preceq_m B$) if there exists a recursive function $f : \Sigma^* \rightarrow \Sigma^*$ such that for every string $x \in \Sigma^*$

$$x \in A \iff f(x) \in B.$$ 

$f$ is called the reduction function.

**Proposition 5.24.** Many-to-one reducibility is a reflexive and transitive relation on the subsets of $\Sigma^*$.

Proof. **Reflexive.** $A \preceq_m A$ - the identity function provides a reduction function. **Transitive.** Let $A \preceq_m B$ and $B \preceq_m C$. Let $f$ be a reduction function associated with $A \preceq_m B$ and let $g$ be a reduction function associated with $B \preceq_m C$. Then $h(x) = g(f(x))$ is a reduction function for $A \preceq_m C$.

**Proposition 5.25.** Let $A$, $B$ be nonempty proper subsets of $\Sigma^*$.

a. If $A$ is recursive, then $A \preceq_m B$.

b. If $A \subseteq B$ and $B \subseteq A$ are recursive, then $A \preceq_m B$ (and $B \preceq_m A$).

Proof. a. Let $x_0 \in B$, $x_1 \in B$. Define

$$f(x) = \begin{cases} x_1 & \text{if } x \in A \\ x_0 & \text{if } x \not\in A \end{cases}$$

Since both $A \cap B$ and $B \setminus A$ are recursive, $g$ is recursive. We still need to prove $x \in A \Rightarrow g(x) \in B$. Assume $x \in A$. Then either $g(x) = x_1 \in B$ or $g(x) = x$ and $x \not\in A \cup B$. But $x \not\in A \cup B \Rightarrow x \in B$. In both cases $g(x) \in B$. If $x \not\in A$, either $g(x) = x_0 \in B$ or $g(x) = x$ and $x \not\in B \setminus A$. In both cases $g(x) \in B$. Thus $g$ is a reduction function and $A \preceq_m B$.

A similar argument gives the opposite reduction.

Thus $x \in A \Rightarrow f(x) \in B$. Since $A$ is recursive, $\chi_A$ is recursive, which now implies that $f$ is recursive. Thus $f$ is the desired reduction function.

b. Choose two strings $x_0 \in B$, $x_1 \in B$. Define

$$\begin{array}{lcl}
\sigma(x) = x_1 & \text{if } x \in A \cup B \\
\sigma(x) = x_0 & \text{otherwise}
\end{array}$$

Since both $A \cap B$ and $B \setminus A$ are recursive, $g$ is recursive. We still need to prove $x \in A \Rightarrow g(x) \in B$. Assume $x \in A$. Then either $g(x) = x_1 \in B$ or $g(x) = x$ and $x \not\in A \cup B$. But $x \not\in A \cup B \Rightarrow x \in B$. In both cases $g(x) \in B$. If $x \not\in A$, either $g(x) = x_0 \in B$ or $g(x) = x$ and $x \not\in B \setminus A$. In both cases $g(x) \in B$. Thus $g$ is a reduction function and $A \preceq_m B$. A similar argument gives the opposite reduction.

We now look at a fairly deep technical Lemma that will allow us to attack other problems through reducibility arguments.
Before we get involved in the details of the proof, let us look at some applications. Recall that \( W_e = \text{LM}(M_e) \), and \( (W_e \mid e \geq 0) \) is the class of r.e. sets over \( \{0, 1\}^* \). We proved that \( \{n \mid W_e = \emptyset\} \) is r.e. We now prove:

**Theorem.** The set \( \text{EMP} = \{n \mid W_e = \emptyset\} \) is not recursive.

**Proof.** We will reduce the halting problem \( K = \{n \mid n \in W_e\} \) to \( \text{co} \text{EMP} \); \( K \ni \text{co} \text{EMP} \). Note that \( \text{EMP} \) is recursive iff \( \text{co} \text{EMP} \) is recursive. Since \( K \) is known to be not recursive (we proved this a little while ago - but it is r.e.), the reduction would let us conclude as desired. We define a function \( g \) as follows:

\[
g(y, x) = \begin{cases} 1 & \text{if } x \in K \\ \uparrow & \text{otherwise.} \end{cases}
\]

\( g \) is partial recursive because \( g(y, x) = \phi_e(x) \).

---

The **Enumeration Theorem** (the set \( (\phi_e^n(x_1, \ldots, x_m) \mid n \geq 0, m \geq 1) \) is countable and includes all partial recursive functions) implies that for all \( y \in \{0, 1\}^* \), the function \( g(y, x) = \phi_e^n(y, x) \) is recursive. For this fixed \( e \), define \( f(x) = s_e(e, x) \), where \( s_e(e, x) \) is the function guaranteed by the \( s-m-n \) Theorem, and thus \( f \) is primitive recursive - hence recursive. We now prove that \( f \) is a reduction function \( f : K = \{n \mid n \in W_e\} \ni \text{co} \text{EMP} \). We will need to show that for all \( x \in K \), \( f(x) = \text{co} \text{EMP} \).

If \( x \in K \), then \( s_e(e, y) = 1 \) for all \( y \in \{0, 1\}^* \) and \( W_{s_e(e, y)} = \{0, 1\}^* \). Thus \( f(x) = \text{co} \text{EMP} \).

We have just proven that \( f \) is a reduction function for the problem.
Reducibility

**Theorem 5.28** (The s-m-n theorem). For each pair of integers $m, n > 0$, there is a primitive recursive function $s_m^n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that

$$\phi_{s_m^n}^{\alpha_1, \ldots, \alpha_m}(x_1, \ldots, x_m) = \phi_{s_m^n}(x_1, \ldots, x_m).$$

**Proof.** The proof proceeds by induction on $n$ (and arbitrary $m$), with the base case ($n = 1$) being the hard part.

$n = 1$. The function $s_1^1$ (that we must construct) takes as input a DTM code $e$ and a string $y \in \{0, 1\}^*$, and outputs a code $z$ s.t. $M_z$ on inputs $x_1, \ldots, x_m$ outputs the same value as $M_e$ on inputs $x_1, \ldots, x_m$. What must $M_z$ do? It must start from a configuration $(q_1, Bx_1B \ldots x_mB)$ and end in a configuration $(q_1, Bx_1B \ldots x_mB)$. Let $M_z$ output this string after which machine $M_z$ takes over and executes.

Reducibility

**Theorem 5.29.** The halting problem $K = \{ \langle \sigma, n \rangle \mid \sigma \in \Sigma^* \}$ is complete.

**Proof.** Let $B$ be r.e. We must construct a reduction $B \leq_m K$. Let $g(x, y) = 1$ if $x \in B$ and $g(x, y) = 0$ otherwise.

If $x \in B$, $g(x, y) = 1$. By the Enumeration Theorem, $g(y, x) = \phi_{s_m^n}^n(y, x)$ for some $e \geq 0$. For this fixed $e$, define $f(x) = s_e^1(c, x)$. So $f$ is (primitive) recursive. We must show $x \in B$ if $f(x) \in K$.

If $x \in B$, $g(x, y) = 1 \forall y \in \{0, 1\}^*$. Letting $y = f(x)$, we have $\phi_{s_m^n}(f(x)) = 1$ which implies that $f(x) \in K$. The halting problem $K = \{ \langle \sigma, n \rangle \mid \sigma \in \Sigma^* \}$ is (primitive) recursive. We must show $\forall y \in \{0, 1\}^*$.

Thus $\phi_{s_m^n}(f(x)) = 1$.

**Corollary.** col(Empr) is complete.

**Proof.** The two results just obtained give that, for every r.e. set $B$, $\langle B \leq_m \text{col}(\text{Empr}) \rangle$ is complete. Transitivity of reducibility gives the result.

We are now ready to look at the details of the proof of the s-m-n theorem.

Reducibility

If $x \not\in B$, then $g(x, y) = 0$ for all $y$. Thus $\phi_{s_m^n}(y) = 0$ for all $y$. By the Enumeration Theorem, $\phi_{s_m^n}(y) = \phi_{\text{Empr}}^n(y)$ for some $e \geq 0$. For this fixed $e$, define $f(x) = s_e^1(c, x)$. So $f$ is (primitive) recursive. We must show $x \in B$ if $f(x) \in K$.

If $x \in B$, $g(x, y) = 1 \forall y \in \{0, 1\}^*$. Letting $y = f(x)$, we have $\phi_{s_m^n}(f(x)) = 1$ which implies that $f(x) \in K$. The halting problem $K = \{ \langle \sigma, n \rangle \mid \sigma \in \Sigma^* \}$ is (primitive) recursive. We must show $\forall y \in \{0, 1\}^*$.

Thus $\phi_{s_m^n}(f(x)) = 1$.

**Corollary.** col(Empr) is complete.

**Proof.** The two results just obtained give that, for every r.e. set $B$, $\langle B \leq_m \text{col}(\text{Empr}) \rangle$ is complete. Transitivity of reducibility gives the result.

We are now ready to look at the details of the proof of the s-m-n theorem.

Reducibility

If $x \not\in B$, then $g(x, y) = 0$ for all $y$. Thus $\phi_{s_m^n}(y) = 0$ for all $y$. By the Enumeration Theorem, $\phi_{s_m^n}(y) = \phi_{\text{Empr}}^n(y)$ for some $e \geq 0$. For this fixed $e$, define $f(x) = s_e^1(c, x)$. So $f$ is (primitive) recursive. We must show $x \in B$ if $f(x) \in K$.

If $x \in B$, $g(x, y) = 1 \forall y \in \{0, 1\}^*$. Letting $y = f(x)$, we have $\phi_{s_m^n}(f(x)) = 1$ which implies that $f(x) \in K$. The halting problem $K = \{ \langle \sigma, n \rangle \mid \sigma \in \Sigma^* \}$ is (primitive) recursive. We must show $\forall y \in \{0, 1\}^*$.

Thus $\phi_{s_m^n}(f(x)) = 1$.
Reductibility

This will have the desired effect on the initial configuration. How do we capture this in a primitive recursive function from code to code? Define \( g : ((0, 1)^*) \rightarrow (0, 1)^* \):

a. If \( i = 0 \), \( g(e, y, i) \) is the code of the instructions of \( M_i \) in part i).

b. If \( 1 \leq i \leq k + 1 \), then \( g(e, y, i) \) is the code of the \( i \)th instruction in parts ii).

c. If \( i > k + 2 \), \( g(e, y, i) = 0 \).

Claim: \( g \) is primitive recursive.

a. \( g(e, y, 0) \) can be obtained from \( e \) by replacing every substring \( 0^p1 \) with \( 0^{p+1}1 \) and every substring \( 01 \) with \( 011 \). Ex. 5.1.1(b) (hint: recall that all strings are encoded as numeric values) shows that \( g(e, y, 0) \) is primitive recursive.

b. can be split into three parts: \( i = 1 \) returns the new instruction \( \delta(q_i, b) = (q_{i+1}, \#b, R) \) - recall that \( q_i \) has become \( q_{i+1} \). \( i = k + 1 \) returns the instruction \( \delta(q_{k+1}, b) = (q_{k+2}, y, R) \).

Reducibility

For the general case, define

\[
x_{e-y} = (x_{we}(x_{w-e}(x_{w-01}) + 1), \ldots, y_b).
\]

We can verify:

\[
\phi^{x_{e-y}}(x_1, \ldots, x_n, y_1, \ldots, y_m) = \phi^{x_{w-e}}(x_1, \ldots, x_n, y_1, \ldots, y_m).
\]

Thus \( x_{e-y} \) satisfies the desired properties.

Reducibility

2 \( \leq i \leq k + 1 \) (i.e., iii) we can express \( g(e, y, i) \) as

if \( \text{substr}(y, \text{minus}(i, 1), 0) = 0 \) then \( 10^{m-i}10^{m-i}10^{m-i} \)

else \( 10^{m-i}10^{m-i}10^{m-i} \).

These are the instructions that copy \( y \) to the right of \( x_1, \ldots, x_n, 0^i \) is the blank at the right end of the current tape, \( 0 \) is a 0, \( 0^i \) is a 1, and the last \( 0^i \) is a move-right).

So \( g(e, y, i) \) is primitive recursive. Now define a new function \( s' \) that concatenates all the instructions generated by \( g \):

\[
s'(e, y, i + 1) = \text{concat}(s'(e, y, i), g(e, y, i + 1)).
\]

Define \( s'(e, y) = \text{concat}(s'(e, y, 1), 11) \), which is, clearly, primitive recursive.

We are done with the base step of the induction.

Reducibility

Index Sets. We want some way to ask the question does a partial recursive function / have property \( P \)? For each such function there may be many (supposedly equivalent) Turing machines that compute it, yet there is a single function \( q \), associated with it, as well as a single set \( W_q = L(M) \). Does our capacity to determine whether the property of \( f \) is decidable depend just on \( q \), or \( W_q \), as one would hope, or does it depend on the specific Turing machine \( M \) used to compute it?

- Def.: function-index set. \( A \subseteq \{0, 1\}^* \) is a function-index set if, for any \( x, y \in \{0, 1\}^* \), \( q^x = q^y \) implies \( \chi_A(x) = \chi_A(y) \).

- Def.: set-index set. \( A \subseteq \{0, 1\}^* \) is a set-index set (or index set) if, for any \( x, y \in \{0, 1\}^* \), \( W_x = W_y \) implies \( \chi_A(x) = \chi_A(y) \).

Note: a set-index set is a function-index set.
Reducibility

Def.: The problem of determining whether a partial recursive function \( f \) has property \( P \) is said to be \textit{undecidable} (from the code of a DTM that computes \( f = \phi_j \)) if the corresponding function-index set \( A_f = \{ x : f(x) \in P(\phi_j) \} \) is not recursive.

Def.: The problem of determining whether an r.e. set \( A \) has property \( P \) is said to be \textit{undecidable} if the corresponding set-index set \( A_P = \{ x : f(x) \in P(W_x) \} \) is not recursive.

Examples.
1. \( \text{Exp} = \{ x : W_x = \emptyset \} \) is a set-index set, since the question: is \( x \in \text{Exp} \) depends only on whether \( W_x = \emptyset \) or not.
2. \( K = \{ x : x \in W_x \} \) is not a set-index set, since the question: is \( x \in K \) depends not only on the set \( W_x \) but also on the membership of \( x \) in \( W_x \).
3. \( B_1 = \{ x : M_{0,0} \text{ halts in at most 200 moves} \} \) is not a set-index set, since, by permuting the instructions, one can design a second TM computing the same function, but in a different \# of moves.

\( g(y, x) \) is partial recursive, since \( g(y, x) = \alpha_k(x) \psi_k(y) \). The enumeration theorem implies \( \exists x \geq 0 \) s.t. \( g(y, x) = \phi_k^* (x, y) \). Define \( f(x) = \psi^+_k (x, y) \), which is primitive recursive by the \( \psi \)-\( \alpha \)-\( \nu \)-Theorem.

Claim: \( f \) is a reduction function - \( f : K \rightarrow A \). \( (x : f(x) \in A) \).

Pf. of Claim: Start with \( x \in K \). Then \( \psi_{f(x)} (y) = \psi_k^* (x, y) \forall y \in \{ 0, 1 \}^* \), and \( \psi_k = \psi_k^* \). Since \( A \) is a function-index set and \( x_k \in A, f(x) \in A \). Now, assume \( x \in K \). Then \( \psi_{f(x)} (y) = \psi_k^* (x, y) \forall y \in \{ 0, 1 \}^* \). Thus \( W_{f(x)} = \emptyset \), and \( f(x) \in A \).

2. Assume all indices \( x \) s.t. \( W_x = \emptyset \) are in \( A \). The same argument, with \( x_k \in A \), gives that \( K \subseteq A \).

Since at least one of \( A \) and \( \complement A \) is not r.e. (if both were, \( A \) would be recursive and thus \( A \) would be recursive), \( \complement A \) cannot be recursive.
Reducibility

We still need to show that $A_1$ is a function-index set: for any $x, y \in \{0, 1\}^*$, $\phi_x = \phi_y$ implies $\chi_{A_1}(x) = \chi_{A_1}(y)$. Certainly $\phi_x = \phi_y$ implies that $\phi_x(0) = \phi_y(0)$, and so, for both, the desired inequality holds or fails simultaneously. So $\chi_{A_1}(x) = \chi_{A_1}(y)$.

b. The identity function (TM that returns its input) gives that $\text{TOT} \neq \emptyset$. Furthermore, $x \in \text{TOT}$ implies that $\text{TOT} \neq \{0, 1\}^*$. Other examples are easy to construct. To show that $\text{TOT}$ is a set-index set: let $x$ and $y$ be two strings in $\{0, 1\}^*$.

f. We are asking for the Turing machines that accept palindrome languages. That both $\text{REV}$ and its complement are non-empty should be trivial to see. $W_x = W_y$ means (at least) that either both machines define recursive functions, or both do not. So $\chi_{\text{REV}}(x) = \chi_{\text{REV}}(y)$.

Reducibility

Lemma. If $A$ is a nontrivial function-index set and $\text{EMP} \subseteq A$, then $A$ is not r.e.

Proof. Recall that, by the $s-m-n$ theorem, we were able to show that $K = \{x | x \in W_x \} \leq_m \text{EMP}$, where $\text{EMP} = \{y | W_y = \emptyset\}$. Since, by our current assumption, all indices $y$ s.t. $W_y = \emptyset$ are in $A$, the second part of the proof of Rice’s Theorem gives that $K \leq_m A$. By the definition of reducibility, we have the existence of a recursive function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ s.t. $x \in K \iff f(x) \in A$. But this is equivalent to $x \in K \iff f(x) \in A$. Thus, the same $f$ provides the reducibility $K \leq_m A$. If $A$ were recursively enumerable, then $K$ would also be so (Prop. 5.26). Since $K$ is recursively enumerable but not recursive, $K$ cannot be r.e.

Corollary. The sets $\text{FIN}$, $\text{REC}$, $\text{REG}$ and $\text{REV}$ are not r.e.

Proof. The sets are all non-trivial index sets containing $\text{EMP}$.

Another proof for $\text{FIN}$.

$x \in \text{FIN} \iff (\exists y \exists w)[\text{print}(0, w, x, t) \text{ and } w > 5]$

$\iff (\exists z)[\text{print}(0, l(z), x, r(z)) \text{ and } l(z) > 5]$. By the Projection Theorem, $\text{FIN}$ is r.e. since we have a recursive predicate $R$ s.t. $\text{FIN} = \{x | (\exists y)R(x, y)\}$. Since, by the Corollary of Rice’s Theorem, $\text{FIN}$ is not recursive, $\text{FIN}$ cannot be r.e.