Reductions

Recall that a decision problem **P** is reducible to a decision problem **Q**, if there is a total Turing-computable function r, such that r converts instances of **P** into instances of **Q**. In other words, for any string w, $\mathbf{P}(w)$ *(the answer to w is yes)* iff $\mathbf{Q}(w)$ *(the answer to $r(w)$ is yes)*. We can frame this more formally in terms of languages.

Definition 1.1. Let X, Y be languages over an alphabet Σ, *i.e.,* $X, Y \subseteq \Sigma^*$. A Turing-computable function r is a *reduction* from X to Y if

$$\forall w \in \Sigma^*.\, w \in X \iff r(w) \in Y$$

Such reductions are also called *many-one reductions* in the literature.

Example 1.2. Let $X = \{a^i b^i c^j \mid i \geq 0, j \geq 0\}$ and $Y = \{a^i b^i \mid i \geq 0\}$. There is a reduction from X to Y that takes, given a string, removes all the c's from the end of the string.

Example 1.3. The halting problem **H** reduces to the blank tape halting problem **B**. Given an encoding of a machine M and an input string w, one can compute an encoding of the machine M' that runs $M(w)$. In particular, $M(w)\downarrow \iff M'(\lambda)\downarrow$.

Exercise 1.4. Suppose that r is a reduction from X to Y. Then verify:

a. if Y is decidable/recursive, then so is X;

b. if Y is recursively enumerable, then so is X;

c. if X is undecidable/non-recursive, then so is Y.

Exercise 1.5. Convince yourself that the notion of reducibility is reflexive and transitive, *i.e.,*

a. For any language X there is a reduction from X to itself;

b. If r_1 is a reduction from X to Y, and r_2 is a reduction from Y to Z, then you can construct a reduction from X to Z.

Exercise 1.6. Verify that X is reducible to Y iff $\overline{X}$ is reducible to $\overline{Y}$.

Exercise 1.7. Suppose that X is a recursively enumerable language[1], *i.e.,* there is a Turing machine M such that $L(M) = X$. Show that you can reduce the problem associated with X to the halting problem. More specifically, you need to construct a reduction from X to the set

$$\{R(M)w \mid M \text{ terminates on } w\}.$$

Exercise 1.8. Show the converse of Exercise 1.8: a language X is recursively enumerable if it reduces to the halting problem.

The two exercise above states that the halting problem is *complete*: in a sense it is the hardest decision problem. In the next section we will use the reduction technique to show that a large class of decision problems in computability theory are undecidable.

Properties of r.e. languages and Rice's theorem

Let $S \subseteq \mathcal{P}(\Sigma^*)$ be a set of languages (over the alphabet Σ) such that

[1]Also known as *recursief opsombaar*, and written "r.e." for short.

a. $\exists M_1 .\, L(M_1) \in S$;

b. $\exists M_2 .\, L(M_2) \notin S$.

That is, S is *nontrivial.* There is at least one r.e. language in S, and at least one r.e. language outside of S.

We can view S as a *predicate* on r.e. languages, *i.e.,* $L \in S$ if L has a specific (nontrivial) property.

Example 1.9. Some examples of nontrivial properties:

a. $L \in S$ iff L is a regular language;

b. $L \in S$ iff $\sigma \in L$ for some constant string σ;

c. $L \in S$ iff L is finite.

Exercise 1.10. Verify that each predicate in Example 1.9 is nontrivial.

Definition 1.11. For such a nontrivial predicate S we can associate a decision problem $\mathbf{D}_S$: given a Turing machine M, does the language recognised by M has the property S?

$$\mathbf{D}_S(R(M)) \triangleq L(M) \in S?$$

Exercise 1.12. Verify that if S is a nontrivial property of recursively enumerable languages, then so is its complement $\overline{S}$. Show that $\mathbf{D}_S$ is decidable iff $\mathbf{D}_{\overline{S}}$ is decidable.

Exercise 1.13. Come up with an S, such that $\mathbf{D}_S$ is the blank tape halting problem.

Rice's theorem states that $\mathbf{D}_S$ is undecidable for a nontrivial S. We prove it by constructing a reduction from the blank tape halting problem to $\mathbf{D}_S$. We reason by contradiction: suppose $\mathbf{D}_S$ is decidable and the machine P decided it, *i.e.,*

$$P(R(M)) = 1 \iff L(M) \in S.$$

Given a Turing machine N, we want to construct another Turing machine N' such that

$$P(R(N')) = 1 \iff N(\lambda) \downarrow$$

thus reducing the blank tape halting problem to D_S.

Assume that $\emptyset \notin S$, and let M_1 be a Turing machine such that $L(M_1) \in S$. The behaviour of the machine N' on the input string x is as follows:

(i) Run $N(\lambda)$;

(ii) Run $M_1(x)$ and return the result.

Exercise 1.14. Verify that, under assumption that $\emptyset \notin S$,

a. $L(N') = L(M_1)$, if $N(\lambda) \downarrow$;

b. $L(N') = \emptyset$ otherwise.

Conclude that $P(R(N')) = 1 \iff N(\lambda) \downarrow$.

Exercise 1.15 (Rice's theorem)**.** Apply Exercise 1.12 to get rid of the assumption $\emptyset \notin S$ in Exercise 1.14. (Reason whether $\emptyset \in S$ or $\emptyset \notin S$.) Conclude that you have a reduction from the blank tape halting problem to D_S.