

# 1 Undecidability

## Notation.

- It is convenient below to use the following notation

$$\begin{aligned}L_{\text{acc}} &= \{\langle w, x \rangle \mid x \in L(w)\} = \{\langle w, x \rangle \mid f_x(x) = 1\} \\L_{\text{self}} &= \{w \mid \langle w, w \rangle \in L_{\text{acc}}\} \\&= \{w \mid w \in L(w)\}\end{aligned}$$

- Below, when we speak of a *total* program we mean: a program that halts on all inputs.

### Problem 1.1. Sets and decision problems

For each of the following sets write the corresponding decision problem (in INSTANCE/QUESTION style).

1. The set of strings of even length.
2. The set of primes in binary.
3. The set  $\{a^n b^n \mid n \geq 0\}$ .
4. The universal set  $L_{\text{acc}}$ .
5. The set  $L_{\text{self}}$ .
6. The set  $\overline{L_{\text{self}}}$ .
7. The set of total programs.

### Solution:

1. The set: the set of strings of even length.

The decision problem:

INPUT: a string  $x$

QUESTION: is  $|x|$  even?

2. The set: the set of primes in binary.

The decision problem:

INPUT: a number  $n$  in binary notation

QUESTION: is  $n$  prime?

3. The set: the set  $\{a^n b^n \mid n \geq 0\}$ .

The decision problem:

INPUT: a string  $x \in \{a, b\}^*$

QUESTION: is  $x$  of the form  $a^n b^n$  for some  $n \geq 0$ ?

4. The set: the universal set  $L_{acc}$ .

The decision problem:

INPUT: a pair  $\langle P, x \rangle$

QUESTION: does  $P$  accept  $x$ ?

5. The set: the set  $L_{self}$ .

The decision problem:

INPUT: a program  $P$

QUESTION: does  $P$  accept  $P$ ?

6. The set: the set  $\overline{L_{self}}$ .

The decision problem:

INPUT: a program  $P$

QUESTION: does  $P$  fail to accept  $P$ ?

7. The set: the set of total programs.

The decision problem:

INPUT: a program  $P$

QUESTION: is  $P$  total?

**Problem 1.2.** *Decision problems and sets*

For each of the following problems identify the corresponding language.

1. INPUT: A program  $m$   
QUESTION: Does  $m$  have at least 999 lines?

**Solution:**

$\{m \mid m \text{ has at least 999 lines}\}$

2. INPUT: A program  $m$   
QUESTION: Does  $m$  accept  $\epsilon$ ?

**Solution:**

$\{m \mid \epsilon \in L(M)\}$

3. INPUT: A program  $m$   
QUESTION: Does  $m$  accept every string?

**Solution:**

$\{m \mid L(M) = \Sigma^*\}$

4. INPUT: A program  $m$

QUESTION: Is  $L(m)$  regular?

**Solution:**

$\{m \mid L(M) \text{ is regular} \}$

5. INPUT: A program  $m$   
QUESTION: Is  $L(m)$  decidable?

**Solution:**

$\{m \mid L(M) \text{ is decidable} \}$

6. INPUT: A program  $m$   
QUESTION: Is there some program  $n$  with fewer lines than  $m$  such that  $L(n) = L(m)$ ?

*Context.* This is essentially the problem of asking whether a given program can be shortened without changing its meaning.

**Solution:**

$\{m \mid \text{there exists } n, \text{ with } |n| < |m| \text{ and } L(n) = L(m)\}$

**Problem 1.3.** *Fundamental theorems*

Be able to state precisely *and prove* the theorems that

1. a language  $A$  is decidable if and only if  $A$  is RE and  $\bar{A}$  is RE.
2. the set  $L_{\text{self}}$  is RE but not decidable.
3. the universal set  $L_{\text{acc}}$  is RE but not decidable

**Problem 1.4.** *Decidable closure properties*

Suppose that  $A$  and  $B$  are decidable sets. Prove the following

1.  $\Sigma^* - A$  is decidable.
2.  $A \cap B$  is decidable.
3.  $A \cup B$  is decidable.
4.  $AB$  is decidable.
5.  $A^*$  is decidable.

**Solution:**

*Proof.* 1. Suppose  $A$  is a decidable set; we wish to show that  $\Sigma^* - A$  is decidable. Let  $P_A$  be a total program such that  $L(P_A)$  is  $A$ . Our goal is to show that there exists a total program  $Q$  such that  $L(Q)$  is  $\Sigma^* - A$ . We exhibit pseudocode for such a program as follows:

*on input  $w$ ;*  
*run  $P_A$  on  $w$ ;*  
*if this run is accepting, then EXIT\_REJECT else EXIT\_ACCEPT.*

To defend the claim that this program suffices we must argue that  $Q$  is total and that  $L(Q)$  is  $\Sigma^* - A$ . The fact that  $Q$  is total follows from the fact that  $P_A$  is total. The fact that  $L(Q)$  is  $\Sigma^* - A$  is obvious.

- Suppose  $A$  and  $B$  are decidable sets; we wish to show that  $A \cap B$  is decidable. Let  $P_A$  be a total program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a total program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a total program  $Q$  such that  $L(Q)$  is  $A \cap B$ . We exhibit pseudocode for such a program as follows:

*on input  $w$ ;*  
*run  $P_A$  on  $w$ ;*  
*run  $P_B$  on  $w$ ;*  
*if each of these runs are accepting, then EXIT\_ACCEPT else*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $Q$  is total and that  $L(Q)$  is  $A \cap B$ . The fact that  $Q$  is total follows from the facts that both  $P_A$  and  $P_B$  are total, and so when they are run on  $w$  they are guaranteed to return. The fact that  $L(Q)$  is  $A \cap B$  is immediate.

- Suppose  $A$  and  $B$  are decidable sets; we wish to show that  $A \cup B$  is decidable. Let  $P_A$  be a total program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a total program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a total program  $Q$  such that  $L(Q)$  is  $A \cup B$ . We exhibit pseudocode for such a program as follows:

*on input  $w$ ;*  
*run  $P_A$  on  $w$ ;*  
*run  $P_B$  on  $w$ ;*  
*if either of these runs are accepting, then EXIT\_ACCEPT else*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $Q$  is total and that  $L(Q)$  is  $A \cup B$ . The fact that  $Q$  is total follows from the facts that both  $P_A$  and  $P_B$  are total, and so when they are run on  $w$  they are guaranteed to return. The fact that  $L(Q)$  is  $A \cup B$  is immediate.

- Next is concatenation; this is a little more interesting. Suppose  $A$  and  $B$  are decidable sets; we wish to show that  $AB$  is decidable. Let  $P_A$  be a total program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a total program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a total program  $Q$  such that  $L(Q)$  is  $AB$ . We exhibit pseudocode for such a program as follows:

*on input  $w$ ;*  
*for each pair of words  $w_1, w_2$  such that  $w_1w_2 = w$ :*  
*run  $P_A$  on  $w_1$ ;*  
*run  $P_B$  on  $w_2$ ;*  
*if each of these runs are accepting, then EXIT\_ACCEPT*

*// If we get here we have failed...*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $Q$  is total and that  $L(Q)$  is  $AB$ . The fact that  $Q$  is total follows from the facts that both  $P_A$  and  $P_B$  are total, and that given  $w$  there are only finitely many pairs of words  $w_1, w_2$  such that  $w_1w_2 = w$ . (How many are there precisely, in terms of  $|w|$ ?). That is, the for-loop is guaranteed to have only finitely many iterations. The fact that  $L(Q)$  is  $AB$  is straight from the definition of concatenation: a word  $w$  is in  $AB$  iff there are words  $w_1 \in A$  and  $w_2 \in B$  such that  $w = w_1w_2$ .

5. Finally, for asterate. Suppose  $A$  is a decidable set; we wish to show that  $A^*$  is decidable. Let  $P_A$  be a total program such that  $L(P_A)$  is  $A$ . Our goal is to show that there exists a total program  $Q$  such that  $L(Q)$  is  $A^*$ . We exhibit pseudocode for such a program as follows:

*on input w;*  
*for each sequence of words  $w_1, w_2, \dots, w_n$  such that  $w_1w_2 \dots w_n = w$ :*  
     *run  $P_A$  on  $w_1$ ;*  
     *run  $P_A$  on  $w_2$ ;*  
     ...  
     *run  $P_A$  on  $w_n$ ;*  
     *if each of these runs are accepting, then EXIT\_ACCEPT*  
*// If we get here we have failed...*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $Q$  is total and that  $L(Q)$  is  $A^*$ . The fact that  $Q$  is total follows from the fact that  $P_A$  is total, and that given  $w$  there are only finitely many sequences of words  $w_1, w_2, \dots, w_n$  such that  $w_1w_2 \dots, w_n = w$ . That is, the for-loop is guaranteed to have only finitely many iterations, and each for-loop body has only finitely many statements. The fact that  $L(Q)$  is  $A^*$  is straight from the definition of asterate: a word  $w$  is in  $A^*$  iff there are words  $w_1, w_2, \dots, w_n$  such that each  $w_i$  is in  $A$  and  $w_1w_2 \dots w_n = w$ .

///

**Problem 1.5. RE closure properties**

Suppose that  $A$  and  $B$  are RE sets. Prove the following

1.  $A \cap B$  is RE.
2.  $A \cup B$  is RE.
3.  $AB$  is RE.
4.  $A^*$  is RE.

*Hint.* In each case you should proceed by outlining a not-necessarily-total program for the language in question (for instance,  $A \cup B$ ), based on the existence of such programs for the component languages (for instance  $A$  and  $B$ ).

Be careful! For example, here is a *wrong* answer to the problem of showing  $A \cup B$  RE given  $A$  RE and  $B$  RE:

We know there is a program  $M$  with  $L(M) = A$  and a program  $N$  with  $L(N) = B$ .  
Here is a program which recognizes  $A \cup B$ :

On input  $w$ , run  $M$  on  $w$ . If this accepts, print yes. If not, run  $N$  on  $w$ . If this accepts, print yes.

This is wrong wrong wrong because there is no reason to assume that  $M$  will necessarily halt and return control to the top level and allow you to go on to simulate  $N$ . You have to be more clever than that.

**Solution:**

1. Suppose  $A$  and  $B$  are RE sets; we wish to show that  $A \cap B$  is RE. Let  $P_A$  be a program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a program  $Q$  such that  $L(Q)$  is  $A \cap B$ . We exhibit pseudocode for such a program as follows:

```
on input  $w$ ;  
run  $P_A$  on  $w$ ;  
run  $P_B$  on  $w$ ;  
if each of these runs are accepting, then EXIT_ACCEPT else  
EXIT_REJECT.
```

To defend the claim that this program suffices we must argue that  $L(Q)$  is  $A \cap B$ . Note that we are making no claims that  $Q$  halts on all inputs! We only need to establish that if  $w$  is indeed in both  $A$  and  $B$  then program  $Q$  will halt with EXIT\_ACCEPT, and if  $w$  is not in both  $A$  and  $B$  then  $Q$  will not halt with EXIT\_ACCEPT (it may fail to halt or it may halt with a different exit condition). This is clear from inspection of the code. Note that we do not need the “run in parallel” trick to work with intersections ...

2. Suppose  $A$  and  $B$  are RE sets; we wish to show that  $A \cup B$  is RE. Let  $P_A$  be a program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a program  $Q$  such that  $L(Q)$  is  $A \cup B$ . *Here we have to be clever.* We cannot simply use the same pseudocode as we did for the decidable-set case. The problem is that if we run  $P_A$  first on an input  $w$  then this might never return, yet  $w$  might be in  $B$ , hence in  $A \cup B$ , but we never get a chance to verify this. The solution is easy though: we simply run  $P_A$  and  $P_B$  in parallel:

```
on input  $w$ ;  
in parallel:  
run  $P_A$  on  $w$ ;  
run  $P_B$  on  $w$ ;  
if and when either of these runs halts with acceptance, then  
EXIT_ACCEPT.
```

*// If we get here we have failed...*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $L(Q)$  is  $A \cup B$ . This merely amounts to observing that  $Q$  will halt accepting  $w$  precisely if at least one of  $P_A$  accepts  $w$  or  $P_B$  accepts  $w$ .

3. Next is concatenation. Suppose  $A$  and  $B$  are RE sets; we wish to show that  $AB$  is RE. Let  $P_A$  be a program such that  $L(P_A)$  is  $A$ , and let  $P_B$  be a program such that  $L(P_B)$  is  $B$ . Our goal is to show that there exists a program  $Q$  such that  $L(Q)$  is  $AB$ . We exhibit pseudocode for such a program as follows:

*on input w;*  
*In parallel, for each pair of words  $w_1, w_2$  such that  $w_1w_2 = w$ :*  
*run  $P_A$  on  $w_1$ ;*  
*run  $P_B$  on  $w_2$ ;*  
*if each of these runs are accepting, then EXIT\_ACCEPT*  
*// If we get here we have failed...*  
*EXIT\_REJECT.*

To defend the claim that this program suffices we must argue that  $L(Q)$  is  $AB$ . But as in the decidable-set case the fact that  $L(Q)$  is  $AB$  is straight from the definition of concatenation: a word  $w$  is in  $AB$  iff there are words  $w_1 \in A$  and  $w_2 \in B$  such that  $w = w_1w_2$ .

Note the need for parallelism needed here. For any *given* pair  $w_1, w_2$  such that  $w = w_1w_2$ , we can test whether this pair witnesses  $w$  to be in  $AB$  by virtue of having  $w_1 \in A$  and  $w_2 \in B$  without having to run these tests in parallel. Since: if the simulation of  $P_A$  on  $w_1$  fails to halt then that's fine,  $w_1 \notin A$  and we don't expect to "accept" this pair. But we need to be able to check *all* pairs  $w_1, w_2$  such that  $w = w_1w_2$ , without having some "bad" pair prevent us from getting to test a potentially "good" pair. So we test all the ways that  $w$  might be divide in two simultaneously.

4. Finally, for asterate. Suppose  $A$  is a RE set; we wish to show that  $A^*$  is RE. Let  $P_A$  be a program such that  $L(P_A)$  is  $A$ . Our goal is to show that there exists a program  $Q$  such that  $L(Q)$  is  $A^*$ . We exhibit pseudocode for such a program as follows:

*on input x;*  
*// try to find a way of dividing  $x$  into several pieces, each one from  $A$*   
*IN PARALLEL: for each sequence of strings  $u_1, u_2, \dots, u_k$  such that the*  
*concatenation  $u_1u_2 \dots u_k$  is*  
*test whether each  $u_i$  is accepted by  $P_A$ ;*  
*if so then EXIT\_ACCEPT*

The argument that this suffices should by now be familiar to you. You should note again the need for parallelism for essentially the same reason as for the case of concatenation. Note that although we do need to run the set of tests given by each pair  $\{w_1, w_2\}$  in parallel, for any given pair  $\{w_1, w_2\}$  the tests that each  $w_i$  is in  $A$  do not need to be run in parallel.

**Problem 1.6.** *RE subset*

1. Prove or disprove: If  $L$  is RE and  $K \subseteq L$  then  $K$  is RE.
2. Prove or disprove: If  $L$  is RE and  $L \subseteq K$  then  $K$  is RE.

**Solution:**

These are both false.

1. Counterexample: let  $K$  be any non-RE set; take  $L$  to be  $\Sigma^*$ .
2. Counterexample: let  $K$  be any non-RE set; take  $L$  to be  $\emptyset$ .

**Problem 1.7.** Prove or give a specific counterexample: If  $A \cup B$  is RE then  $A$  and  $B$  are RE.

What if we replace “RE” by decidable? What if we replace “ $\cup$ ” by “ $\cap$ ”?

**Solution:**

All are false. Let  $A$  be a set which is RE but not decidable, and take  $B$  to be the complement of  $A$ . Then  $A \cup B = \Sigma^*$ : this is RE, even decidable. And  $A \cap B = \emptyset$ : again RE, even decidable.

**Problem 1.8.** *RE partition (from HMU)*

Suppose  $L_1, L_2, \dots, L_k$  are recursively enumerable languages over the alphabet  $\Sigma$  such that

1.  $L_i \cap L_j = \emptyset$  when  $i \neq j$ , and
2.  $L_1 \cup L_2 \cup \dots \cup L_k = \Sigma^*$ .

Prove that  $L_1$  is decidable. Indicate clearly how each of the hypotheses (a) and (b) are used in your argument.

Is it really necessary to have *both* condition 1 and condition 2 in order to conclude 3? If so, give counterexamples showing that the result fails if one or the other condition fails. If not, explain why not.

**Solution:**

*Hint.* Think about the case  $k = 2$ , and the theorem that says that a set is decidable if and only if it and its complement are RE.

**Problem 1.9.** *RE splitting (from HMU)*

Suppose that  $L$  is RE but not decidable. (So  $\bar{L}$  is not RE...) Consider the language

$$L' \stackrel{\text{def}}{=} \{0x \mid x \text{ is in } L\} \cup \{1x \mid x \text{ is not in } L\}$$

Can you say for certain (without knowing more about  $L$ ) whether  $L'$  is decidable, RE, or non-RE? Justify your answer.

**Solution:**

The set  $L'$  is not RE (which of course implies that it is not decidable either). To prove this, assume for the sake of contradiction that  $L'$  were RE. We get our contradiction by showing that this implies that  $\bar{L}$  would be RE, contrary to what was given.

So suppose the  $P$  is a program with  $L(P) = L'$ . Here is a program  $Q$  with  $L(Q) = \bar{L}$ :

*on input  $x$ ;*  
*run program  $P$  on input  $1x$ ;*  
*if and when this accepts, EXIT\_ACCEPT.*

Program  $Q$  accepts  $x$  if and only if  $1x \in L'$ , but  $1x \in L'$  if and only if  $x \in \bar{L}$ . This establishes  $L(Q) = \bar{L}$  which is our contradiction.

**Problem 1.10.** Let  $X$  be decidable and let  $Y$  be semi-decidable but not decidable. Define

- $Z_1 = X - Y$
- $Z_2 = Y - X$

1. Exactly one of  $Z_1$  or  $Z_2$  is guaranteed to be semi-decidable. Which is it?
2. For the  $Z_i$  which is guaranteed to be semi-decidable, prove it. Here you may quote without proof any closure properties you know.
3. For the  $Z_i$  which is not guaranteed to be semi-decidable, give a recursive set  $X$  and an semi-decidable set  $Y$  which demonstrate this.

**Solution:**

$Z_2$  is guaranteed to be semi-decidable. Since:  $Z_2 = Y - X = Y \cap \bar{X}$ , and  $\bar{X}$  is decidable since  $X$  is, and so  $\bar{X}$  is semi-decidable, and the semi-decidable sets are closed under intersection.

On the other hand  $Z_1 = X \cap \bar{Y}$ . Take  $Y$  to be  $L_{\text{self}}$ , and take  $X$  to be  $\Sigma^*$ . Then  $X - Y$  is  $\overline{L_{\text{self}}}$ , which we know to be non-semi-decidable.

**Problem 1.11.** *Taxonomy*

For each of the following situations, either give an example of a language  $L$  fitting the description, or explain why the situation is impossible.

1.  $L$  is decidable and  $\bar{L}$  is decidable.
2.  $L$  is decidable and  $\bar{L}$  is RE but not decidable.
3.  $L$  is decidable and  $\bar{L}$  is not RE.
4.  $L$  is RE but not decidable and  $\bar{L}$  is decidable.
5.  $L$  is RE but not decidable and  $\bar{L}$  is RE but not decidable.

6.  $L$  is RE but not decidable and  $\bar{L}$  is not RE.
7.  $L$  is not RE and  $\bar{L}$  is decidable.
8.  $L$  is not RE and  $\bar{L}$  is RE but not decidable.
9.  $L$  is not RE and  $\bar{L}$  is not RE.

**Solution:**

1.  $L$  is decidable and  $\bar{L}$  is decidable.  
*Answer:* Any decidable  $L$  works here.
2.  $L$  is decidable and  $\bar{L}$  is RE but not decidable.  
*Answer:* Impossible: if  $L$  is decidable then so is  $\bar{L}$ .
3.  $L$  is decidable and  $\bar{L}$  is not RE.  
*Answer:* Impossible: if  $L$  is decidable then so is  $\bar{L}$ , so certainly  $\bar{L}$  is RE.
4.  $L$  is RE but not decidable and  $\bar{L}$  is decidable.  
*Answer:* Impossible: if  $\bar{L}$  is decidable then so is  $\bar{\bar{L}}$ . But this is just  $L$ , so  $L$  is decidable.
5.  $L$  is RE but not decidable and  $\bar{L}$  is RE but not decidable.  
*Answer:* Impossible: if  $L$  and  $\bar{L}$  were both RE they would both be decidable.
6.  $L$  is RE but not decidable and  $\bar{L}$  is not RE.  
*Answer:* The universal language  $L_{acc}$  fits this description.
7.  $L$  is not RE and  $\bar{L}$  is decidable.  
*Answer:* Impossible: if  $\bar{L}$  is decidable then so is  $L$ ; if  $L$  is decidable then it is certainly RE.
8.  $L$  is not RE and  $\bar{L}$  is RE but not decidable.  
*Answer:*  $L = \overline{L_{acc}}$ , the complement of the universal language, fits this description.
9.  $L$  is not RE and  $\bar{L}$  is not RE.  
*Answer:* In fact we have not proven that this possibility exists, but it does. An example of such a language is the set of all total programs.