1 Reducibility

Definition Reduction: transforming one problem into another, such that the solution to the second problem yields the solution to the first one.

solve_A(...)
...
...
    solve_B(...)
...
...
return

- If B can be solved, then A can be solved.
- Or, negatively, if A is unsolvable, then B is unsolvable.

Notation: $A \leq B$:

- A can be reduced to B.
- A is no harder than B.

Example
solve_L_d(...)
...
...
    solve_A_{TM}(...)
...
...
return
$L_d \leq A_{TM}$

1.1 More undecidable problems / languages

- $HALT_{TM} = \{< M, w > \mid M \ halts \ on \ w\}$

  Theorem 1.1 *$HALT_{TM}$ is undecidable.*

  Proof by reduction of A_{TM}

  Assume $HALT_{TM}$ decidable by some decider R. Then we can construct a decider S for A_{TM} with Algorithm 1.

Algorithm 1 S on $< M, w >$

Run R on $< M, w >$
if R rejects (i.e. M loops on w) **then**
  reject
else
  Run M on $< w >$
  if M accepts w **then**
    accept
  else if M rejects w **then**
    reject
  end if
end if

But we know that A_{TM} is undecidable and no such S exists. Then, no such R exists, and $HALT_{TM}$ is undecidable. ∎

- $E_{TM} = \{< M > \mid L(M) = \emptyset\}$

  Theorem 1.2 *E_{TM} is undecidable.*

  Proof by reduction from A_{TM}.

  For a given A_{TM} instance $< M, w >$, construct an E_{TM} instance $< M_1 >$ such that $L(M_1)$ will be $\emptyset$ or not depending on whether M accepts w or not with Algorithm 2.

Algorithm 2 M_1 on x

```
if x ≠ w then
  reject
else
  Run M on < w >
  if M accepts w then
    accept
  else if M rejects w then
    reject
  end if
end if
```

$$L(M_1) = \begin{cases} \emptyset, \; if \; w \notin L(M) \\ \{w\}, \; if \; w \in L(M) \end{cases}$$

Now, assume E_{TM} is decidable by some TM R. We can construct a decider S for A_{TM} using Algorithm 3.

Algorithm 3 S on $< M, w >$

```
Construct M_1 from M and w
Run R on < M_1 >
if R accepts then
  reject
else if R rejects then
  accept
end if
```

We know S does not exist. Therefore E_{TM} is undecidable. ∎

- $REGULAR_{TM} = \{< M > \mid L(M) \; is \; a \; regular \; language\}$

  Similarly, we will reduce from A_{TM}.

  The idea is to construct a new machine M_2 that recognizes a regular language iff M accepts w. modified machine M_2 . Initially, M_2 will recognize the non-regular language $\{0^n 1^n | n \geq 0\}$ if M does not accept w, and will recognize the regular language Σ^* if M accepts w.

  M_2 = On input x:

  1. If x has the form $0^n 1^n$, *accept*.
  2. If x does not have this form, run M on input w and accept if M accepts w.

Let R be a decider TM that decides $REGULAR_{TM}$, and let S be a decider for A_{TM} .

Algorithm 4 S on $< M, w >$

Construct M_2 from M and w.
Run R on $< M_2 >$
if R accepts **then**
  accept
else if R rejects **then**
  reject
end if

We know S does not exist. Therefore $REGULAR_{TM}$ is undecidable. ∎

- $E_{TM} = \{< M_1, M_2 > \mid L(M) = L(M_2)\}$

Theorem 1.3 *EQ_{TM} is undecidable.*

Proof by reduction from E_{TM}.

Assume EQ_{TM} is decidable by R. Then we can have a decider for S as outlined in Algorithm 5.

Algorithm 5 S on $< M >$

Construct a TM M_3 that rejects everything. (i.e. $L(M_3) = \emptyset$)
Run R on $< M, M_3 >$
if R accepts **then**
  reject
else if R rejects **then**
  accept
end if

We know S does not exist. Therefore EQ_{TM} is undecidable. ∎

1.2 Rice's Theorem

For any non-trivial property $\mathcal{P}$ of the language of a TM:

$$\mathcal{P}_{TM} = \{< M > \mid L(M)\ has\ property\ \mathcal{P}\}$$

is undecidable.

$$\left.\begin{array}{l} L(M) \subseteq \Sigma^* \\ L(M) \supseteq \emptyset \end{array}\right\} Trivial\ properties$$

Example

- $REGULAR_{TM} = \{< M > \mid L(M)\ is\ a\ regular\ language\}$ is undecidable.
- $CF_{TM} = \{< M > \mid L(M)\ is\ a\ context\ free\ language\}$ is undecidable.