

# Reducibility: a methodology for proving unsolvability

Teodor Rus

`rus@cs.uiowa.edu`

The University of Iowa, Department of Computer Science

# Reduction

- A reduction is a method of converting one problem into another problem in such a way that a solution to the second problem can be used to solve the first problem;
- Such methods are common in everyday life;
- Reducibility involves always two problems,  $A$  and  $B$ . If  $A$  reduces to  $B$  we can use a solution to  $B$  to solve  $A$ .

# Facts

1. Reducibility says nothing about solving  $A$  or  $B$ .
2. Reducibility is only about solvability of  $A$  using a solution to  $B$ .

# Everyday life examples

1. **P1:** find your way in a city;  
**P2:** This is easy if you have a map.  
**Note:** problem P1 is reduced to the problem P2.
2. **P1:** traveling from Iowa City to Paris reduces to  
**P2:** buy a plain ticket between two cities.
3. **P1:** measure the area of a circle, reduces to  
**P2:** measure the circle radius  $r$  and evaluate the expression  $\pi r^2$ .
4. **P1:** solve a system of linear equations, reduces to  
**P2:** inverting a matrix.

# Observations

- When problem  $A$  is reducible to problem  $B$ , solving  $A$  cannot be harder than solving  $B$ , because a solution to  $B$  gives a solution to  $A$ .
- If  $A$  is reducible to  $B$  and  $B$  is decidable, then  $A$  is decidable (because a solution to  $B$  solves  $A$ ).
- If  $A$  is undecidable and reducible to  $B$  then  $B$  is undecidable (because if  $B$  would be decidable then  $A$  would be decidable, what is a contradiction).

# Methodology

To prove that a problem  $P$  is unsolvable by reduction method proceeds as follows:

1. Find a problem  $Q$  known to be unsolvable;
2. Assume that  $P$  is solvable by a TM  $M_P$ ;
3. Use the TM  $M_P$  to solve  $Q$  thus reducing  $Q$  to  $P$  by:
  - (a) Encode the problem  $Q$  as an instance  $Q_P$  of problem  $P$ ;
  - (b) Construct a TM  $M_Q$  that solve  $Q_P$  using  $M_P$ ;
  - (c) Since  $M_P$  solves  $P$  and  $Q_P$  is an instance of  $Q$ ,  $M_P$  solves  $Q$ .
4. Since it is known that  $Q$  is unsolvable  $M_Q$  cannot exist. Hence,  $M_P$  cannot exist either.

# Application

- **Problem  $P$ :** the halting problem,  $HALT_{TM}$ :  
determine whether a Turing machine  $M$  halts on an input  $w$ .

**Language:**

$$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM} \wedge M \text{ halts on } w\}$$

- **Unsolvable problem  $Q$ :** we have established the unsolvability of a TM computation, i.e., the problem of determining whether a TM  $M$  accepts a string  $w$ .

- **Undecidable language  $A_{TM}$ :** we used the diagonalization methods to show that the language

$$A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM} \wedge M \text{ accepts } w\}$$

is undecidable.

# Example 1

We use the undecidability of  $A_{TM}$  to show that  $HALT_{TM}$  is undecidable by reducing  $A_{TM}$  to  $HALT_{TM}$

# Theorem 5.1

$HALT_{TM}$  is undecidable

**Proof idea:** by construction:

- Assume that  $HALT_{TM}$  is decidable and use this assumption to show that  $A_{TM}$  is decidable
- This contradicts undecidability of  $A_{TM}$ , previously established (Theorem 4.11 (4.9))

**Note:** the key idea is to show how is  $A_{TM}$  reducible to  $HALT_{TM}$

# Proof, continuation

**Reducibility:**  $A_{TM}$  reduces to  $HALT_{TM}$  if a TM  $R$  that solves  $HALT_{TM}$  can be used to construct a TM  $S$  that solves  $A_{TM}$ .

**Assumption:** suppose that the TM  $R$  decides  $HALT_{TM}$ . How can we use  $R$  to construct  $S$ , the TM that decides  $A_{TM}$  ?

# Construction

Pretend that we are  $S$ . How should we proceed?

1. For an input  $\langle M, w \rangle$  we must accept if  $M$  accepts  $w$ , and we must reject if  $M$  loops or rejects  $w$ .
2. If we try to simulate  $M$  on  $w$  we may not be able to determine whether  $M$  loops, and in this case simulation does not terminate. This is not acceptable because we are deciders!
3. Now we remember that we have the TM  $R$  that decides  $HALT_{TM}$ . We can use  $R$  to test whether  $M$  terminates on  $w$ .
4. If  $R$  tells us that  $M$  doesn't halt on  $w$  we can reject because  $M$  is looping, and thus  $\langle M, w \rangle \notin A_{TM}$ .
5. If  $R$  tells us that  $M$  halts on  $w$ , we can proceed with the simulation because  $M$  does not loop.

# The TM $S$

$S$  = "On input  $\langle M, w \rangle$ , an encoding of  $M$  and a string  $w$ :

1. Run TM  $R$  on input  $\langle M, w \rangle$ ;

2. If  $R$  rejects, *reject*;

**Note:**  $R$  rejects when  $M$  is looping, i.e.  $\langle M, w \rangle \notin A_{TM}$

3. If  $R$  accept, simulate  $M$  on  $w$  until it halts;

4. If  $M$  has accepted, *accept*; if  $M$  has rejected, *reject*."

**Note:** If  $R$  decides  $HALT_{TM}$ ,  $S$  decides  $A_{TM}$

This is a contradiction. Hence,  $R$  cannot decide  $HALT_{TM}$ .

# Language emptiness problem

## The problem:

For a TM  $M = (Q, \Sigma, \Gamma, \delta, q_0, a_a, a_s)$  is

$$L(M) = \{w \in \Sigma^* \mid q_0 \models^* w\} = \emptyset?$$

## The language:

$$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM} \wedge L(M) = \emptyset\}$$

## Theorem 5.2 $E_{TM}$ is undecidable

**Proof idea:** reduce the decidability of  $A_{TM}$  to the decidability of  $E_{TM}$ :

- Assume that  $E_{TM}$  is decidable and let TM  $R$  deciding  $E_{TM}$ .
- Show that  $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } w \text{ is accepted by } M\}$  is decidable by constructing TM  $S$  that uses  $R$  to decide  $A_{TM}$ .

# Constructing TM $S$

Assume again that we are the TM  $S$ .

- **A bad idea:** Run  $R$  on  $\langle M \rangle$ .
  1. If  $R$  accepts then we know that  $L(M) = \emptyset$  and therefore  $M$  cannot accept  $w$  because  $w \notin L(M)$ ; This is good!
  2. If  $R$  rejects,  $L(M) \neq \emptyset$ ; but  $R$  doesn't tell us whether  $M$  accepts  $w$ , rejects  $w$  or loop on  $w$ . Hence, we need another idea.

# Constructing TM $S$

- **A good idea:** run  $R$  on a modification  $\langle M_1 \rangle$  of  $\langle M \rangle$  where  $M_1$  rejects all strings except  $w$ . On  $w$  it works as  $M$ , i.e.,  $M_1(w)$  is the same as  $M(w)$ .
- Then use  $R$  to test whether  $M_1$  recognizes the empty language.
- The only string  $M_1$  can recognize is  $w$ , hence,  $L(M_1) \neq \emptyset$  iff  $M_1$  accepts  $w$ .
- If  $R$  accepts when it is fed with  $\langle M_1 \rangle$  it means that  $L(M_1) = \emptyset$ . Since  $M_1$  is identical with  $M$  on  $w$ , we know that  $M$  doesn't accept  $w$ . If  $R$  rejects then we know that  $L(M_1) = \{w\}$ . But  $M_1(w) = M(w)$  hence accept.

# Proof of Theorem 5.2

The modified machine,  $M_1$  is defined by:

$M_1 =$  "On input  $x$ :

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

**Note:**  $M_1$  has  $w$  as part of its description. It conducts the test  $x = w$  by scanning the input and comparing it character by character with  $w$ .

# Proof, continuation

Assume that  $E_{TM}$  is decidable by  $R$  and construct TM  $S$  that decides  $A_{TM}$  as follows:

$S =$  "On input  $\langle M, w \rangle$ , an encoding of a TM  $M$  and a string  $w$ :

1. Use the description of  $M$  and  $w$  to construct  $M_1$ :

$M_1 =$  "On input  $x$ :

(a) If  $x \neq w$ , *reject*;

(b) If  $x = w$ , run  $M$  on input  $w$  and accept if  $M$  does."

2. Run  $R$  on input  $\langle M_1 \rangle$ ;

3. If  $R$  accept, *rejects*; if  $R$  rejects, *accept*."

# Facts

1. If  $R$  accepts it means that language of  $M_1$  is empty by the definition of  $R$ .
2. Since only string  $M_1$  can accept is  $w$  it means that  $w \notin L(M_1)$ ; but  $L(M_1(w)) = L(M(w))$ , hence  $S$  should reject.
3. If  $R$  rejects, it means that language of  $M_1$  is not empty. Since only string  $M_1$  can accept is  $w$ , and  $L(M_1(w)) = L(M(w))$  it means that  $S$  should accept.

# Conclusions

- $S$  can actually compute a description of  $M_1$  from a description of  $M$  because it needs only add extra states to  $M$  to perform the  $x = w$  test;
- If  $R$  were a decider for  $E_{TM}$ ,  $S$  would be a decider for  $A_{TM}$ ;
- But a decider for  $A_{TM}$  cannot exist, hence  $R$  does not exist either and  $E_{TM}$  is undecidable.

# TM and regular languages

## Problem:

Can we decide whether a TM recognize a language recognized by a simpler computational model, such as a regular language?

**For example:**  $REGULAR_{TM}$ , is the problem of deciding whether a given TM has an equivalent finite automaton. This is the same as deciding whether a TM recognizes a regular language.

## Language:

$$REGULAR_{TM} = \{ \langle M \rangle \mid M \text{ is a TM} \wedge L(M) \text{ is regular} \}$$

# Theorem 5.3

$REGULAR_{TM}$  is undecidable

**Proof idea:** reduce  $A_{TM}$  to  $REGULAR_{TM}$ , i.e., assume that  $REGULAR_{TM}$  is decidable by a TM  $R$  and use this assumption to construct  $S$  that decides  $A_{TM}$ .

**Idea for  $S$ :** take input  $\langle M, w \rangle$  and modify  $M$  so that the resulting TM  $M_2$  recognizes a regular language iff  $M$  accepts  $w$ . Hence we have:

- $M_2$  recognizes the non-regular language  $\{0^n 1^n \mid n \geq 0\}$ , if  $M$  does not accept  $w$ ;
- $M_2$  recognizes the regular language  $\Sigma^*$  if  $M$  accepts  $w$ .

# Constructing $M_2$

$S$  constructs  $M_2$  from  $\langle M, w \rangle$  by the following procedure:

- $M_2$  recognizes the non-regular language  $\{0^n 1^n \mid n \geq 0\}$  if  $M$  does not accept  $w$ ;
- $M_2$  recognizes the regular language  $\Sigma^*$  if  $M$  accepts  $w$ .

# Proof

Let  $R$  be a TM that decides  $REGULAR_{TM}$ . We construct TM  $S$  that decides  $A_{TM}$  as follows:

$S =$  "On input  $\langle M, w \rangle$  where  $M$  is a TM and  $w$  is a string:

1. Construct TM  $M_2$  by the procedure:

$M_2 =$  "On input  $x$ :

- (a) If  $x = 0^n 1^n$  for some  $n \geq 0$ , *accept*;

- (b) If  $x \neq 0^n 1^n$ , run  $M$  on  $w$  and *accept* if  $M$  accepts  $w$ ;

2. Run  $R$  on  $\langle M_2 \rangle$ ;

3. If  $R$  accepts, *accept*; if  $R$  rejects, *reject*."

**Note:** If  $R$  accepts it means that  $M_2$  recognizes a regular language.

But this can happen only if  $M$  accepts  $w$ .

# Facts

1. If  $R$  accepts it means that the input  $x$  is regular, i.e.,  $x \neq 0^n 1^n$  for any  $n$ ;
2. But in this case  $R$  accepts only if  $M$  accepts  $w$ ;

**Note:** the idea for encoding  $A_{TM}$  into an instance of the problem solved by  $R$  is to map  $\langle M, w \rangle$  into a property that  $R$  can check.

# Conclusion

- If  $R$  decides  $REGULAR_{TM}$  then  $S$  decides  $A_{TM}$ ;
- But  $A_{TM}$  is undecidable, i.e.,  $S$  cannot exist;
- Hence,  $R$  cannot exist either, and  $REGULAR_{TM}$  is undecidable.

# Observations

- Determining whether the language of a TM is a context-free language is undecidable;
- Determining whether the language of a TM is decidable is undecidable;
- **General result:** determining whether any property of the language recognized by a Turing machine holds is undecidable (Rice's theorem)

# Rice's theorem (Problem 5.28)

Let  $P$  be any nontrivial property of a language of a Turing machine. Prove that the problem of determining whether a given TM's language has the property  $P$  is undecidable.

**Formally:** let  $P$  be a language consisting of TM descriptions, where  $P$  fulfills two conditions:

1.  $P$  is nontrivial, i.e.,  $P$  contains some, but not all, TM descriptions.
2.  $P$  is a property of TM languages, i.e., for any TMs  $M_1$  and  $M_2$  where  $L(M_1) = L(M_2)$  we have  $\langle M_1 \rangle \in P$  iff  $\langle M_2 \rangle \in P$ .

**Problem:** prove that  $P$  is undecidable.

(See proof in the book on page 215)

# Proof

## Proof idea:

- Assume for the sake of a contradiction that  $P$  is a decidable language satisfying the properties (1) and (2) and let  $R_P$  be a TM that decides  $P$ .
- We show how to decide  $A_{TM}$  using  $R_P$  by constructing an appropriate TM  $S$ .

# Observations

- Let  $T_\emptyset$  be a TM that always reject, i.e.,  $L(T_\emptyset) = \emptyset$ . We may assume that  $\langle T_\emptyset \rangle \notin P$ .

**Rationale:** if  $\langle T_\emptyset \rangle \in P$  we can proceed with  $\overline{P}$  and use the fact that complement of a decidable language is decidable.

- Because  $P$  is not trivial, there exists a TM  $T$  such that  $\langle T \rangle \in P$ .
- We can design the TM  $S$  that decides  $A_{TM}$  by using  $R_P$ 's ability to distinguish between  $\langle T_\emptyset \rangle \notin P$  and  $\langle T \rangle \in P$ .

# Constructing $S$

$S =$  “On input  $\langle M, w \rangle$

1. Use  $M$  and  $w$  to construct the following TM  $M_w$ :

$M_w =$  “On input  $x$ :

(a) Simulate  $M$  on  $w$ . If it halts and reject, *reject*.  
If it accepts, proceed to stage (b).

(b) Simulate  $T$  on  $x$ . If it accepts, *accept*.”

2. Use TM  $R_P$  to determine whether  $\langle M_w \rangle \in P$ .

If  $R_P$  accepts *accept*, if  $R_P$  rejects, *reject*.”

# Note

The TM  $M_w$  simulates  $T$  if  $M$  accepts  $w$ . Thus we have:

$$L(T) = \begin{cases} L(M), & \text{if } M \text{ accepts } w, \\ \emptyset, & \text{if } M \text{ doesn't accept } w. \end{cases} \quad (1)$$

Hence,  $M_w \in P$  iff  $M$  accepts  $w$ .

# Using other reductions

- Sometimes reducing other undecidable languages (different from  $A_{TM}$ ), such as  $E_{TM}$ , is more convenient
- **Example:** testing the equivalence of two TM:

$$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1, M_2 \text{ are TMs} \wedge L(M_1) = L(M_2)\}$$

can be proven undecidable by reduction from  $E_{TM}$ .

# Theorem 5.4

$EQ_{TM}$  is undecidable.

**Proof idea:** show that if  $EQ_{TM}$  were decidable then  $E_{TM}$  also would be decidable by giving a reduction from  $E_{TM}$  to  $EQ_{TM}$

- $E_{TM}$  is the problem of testing whether the language of a TM is empty
- $EQ_{TM}$  is the problem of testing whether the languages of two TMs are the same. If one of these languages is empty we end up with trying to solve  $E_{TM}$
- Hence,  $E_{TM}$  is a special case of  $EQ_{TM}$

# Proof

Let TM  $R$  decide  $EQ_{TM}$ . Construct TM  $S$  to decide  $E_{TM}$  as follows:

$S$  = "On input  $\langle M \rangle$ , where  $M$  is a TM:

1. Run  $R$  on input  $\langle M, M_1 \rangle$  where  $M_1$  is a TM that rejects all inputs;
2. If  $R$  accepts, *accept*, if  $R$  rejects, *reject*."

**Note:** If  $R$  decides  $EQ_{TM}$  then  $S$  decides  $E_{TM}$ ; but  $E_{TM}$  is undecidable, hence  $R$  cannot exist.

# Facts

1. Because  $M_1$  rejects all strings, if  $R$  accepts it means that language of  $M$  is the same with language of  $M_1$ , i.e., empty;
2. If  $R$  rejects, it means that language of  $M$  is not equal with the language of  $M_1$  which is empty. That is, language of  $M$  is not empty.