

Reduction: Examples

- Suppose that one wants to find his/her way in a city, A . This would be easy if one had a city map. Hence, problem A is reduced to the problem of finding a city map, problem B .
- Problem A : measuring the area of a circle, reduces to problem B : measuring r , the circle radius, which reduces to problem C : performing πr^2 .
- Problem A : solving a system of linear equations, reduces to problem B : triangulating a matrix.
- Problem A : proving a set is uncountable, reduces to problem B : establishing a correspondence between this set and the set of reals.

22c:131 Limits of Computation

Hantao Zhang

<http://www.cs.uiowa.edu/~hzhang/c131>

The University of Iowa
Department of Computer Science

Reducibility

Methodology

Convention: “solvable” means either “decidable” or “Turing-recognizable”.

To prove that a problem Q is solvable by reduction method proceeds as follows:

1. Find a problem P known to be solvable.
2. Assume that P is solvable by a TM M_P
3. Use the TM M_P to construct a TM M_Q that solve Q :
 - (a) Encode every instance q of the problem Q as an instance q_P of problem P
 - (b) Use M_P to solve q_P and return the result.

Observations

- Reduction is a terminating process.
- When problem A is reduced to problem B , solving A cannot be harder than the sum of reduction and solving B , because a solution to B gives a solution to A .
- If A is reduced to B and B is decidable, then A is decidable (because a solution to B solves A) in finite number of steps.
- If A is undecidable and reducible to B then B is undecidable (because if B would be decidable then A would, which is a contradiction).

Application

- **Problem P :** the halting problem, $HALT_{TM}$, is the problem of determining whether a Turing machine M halts on an input w
 $HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM} \wedge M \text{ halts on } w\}$
- **Unsolvability problem Q :** we have established the undecidability of
 $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM} \wedge M \text{ accepts } w\}$
the problem of determining whether a TM M accepts a string w .
- We use the undecidability of A_{TM} to show that $HALT_{TM}$ is undecidable by reducing A_{TM} to $HALT_{TM}$

Methodology

Convention: “unsolvable” means either “undecidable” or “not Turing-recognizable”.

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 construct a TM M_Q that solve Q :
 - (a) Encode every instance q of the problem Q as an instance q_P of problem P
 - (b) Use M_P to solve q_P
4. Since it is known that Q is unsolvable M_Q cannot exist. Hence, M_P cannot exist either and P is unsolvable.

Emptiness Problem for TM

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

Theorem 5.2 E_{TM} is undecidable

Proof idea:

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

Theorem 5.1

$HALT_{TM}$ is undecidable

Proof idea: by contradiction:

Assume that the TM R decides $HALT_{TM}$ and use R to construct S , a TM that decides A_{TM} .

$S =$ "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, i.e., M loops on w , *reject*.
3. If R accepts, i.e., M halts on w , simulate M on w until it halts;
4. If M has accepted, *accept*; if M has rejected, *reject*"

Proof

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 if M accepts, *accept*."

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

Constructing TM S

- **Bad idea:** Run R on $\langle M \rangle$. If it accepts then we know that $L(M) = \emptyset$ and therefore M does not accept w ; If R rejects, $L(M) \neq \emptyset$; but we don't know if M accept or not w , hence we need another idea.
- **Good idea:** run R on a modification $\langle M_1 \rangle$ of $\langle M \rangle$ that guarantees that M_1 rejects all strings except w . That is, $L(M_1) = \{w\}$ if $w \in L(M)$ and $L(M_1) = \emptyset$ if $w \notin L(M)$. So $L(M_1) = \emptyset$ iff $w \notin L(M)$. R can test if $L(M_1) = \emptyset$.

TM and Regular Languages

- Can a TM recognize a language recognized by a simpler computational model, such as a regular language?
- For example, $REGULAR_{TM}$, is the problem of testing whether a given TM has an equivalent finite automaton
- This is the same as testing whether a TM recognizes a regular language.

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

Proof, continuation

Assume that E_{TM} is decidable by R , we 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
2. Run R on input $\langle M_1 \rangle$
3. If R accept, *rejects*; if R rejects, *accept*."

Conclusion:

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 and E_{TM} is undecidable.

Note: E_{TM} is co-Turing-recognizable.

Constructing M_2

S constructs M_2 by the following procedure:

- M_2 accepts automatically all strings in $\{0^n 1^n \mid n \geq 0\}$
- In addition, if M accepts w then M_2 accepts all other strings

Theorem 5.3

$REGULAR_{TM}$ is undecidable

Proof idea: reduce to A_{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 :

- 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 Σ^* if M accepts w

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.

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 if M accepts w , then *accept*.
2. Run R on $\langle M_2 \rangle$.
3. If R accepts, *accept*; if R rejects, *reject*."

Proof

- Suppose P is a decidable language satisfying properties (1) and (2).
- Let T_P be a TM that decides P . Without loss of generality assume that T_\emptyset with $L(T_\emptyset) = \emptyset$, $\langle T_\emptyset \rangle \notin P$; if $\langle T_\emptyset \rangle \in P$ we can consider \bar{P} .
- Since P is not trivial, there exists TM M_1 with $\langle M_1 \rangle \in P$. Then the following TM X could decide A_{TM} :
 $X =$ "On input $\langle M, w \rangle$:
 1. Construct a TM M_w that accepts input x iff both M accepts w and M_1 accepts x
 2. Run T_P on $\langle M_w \rangle$. If T_P accepts, *accept*; otherwise *reject*."

Rice's Theorem (Exercise 5.28)

Let P be any property about Turing machines. As usual we express P as a language, i.e., P is the language of TMs having property P . Assume that P satisfies the following two properties:

1. For any TMs M_1 and M_2 , where $L(M_1) = L(M_2)$, we have $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$. I.e., membership of a TM M in P depends only on the language of M .
2. There exist TMs M_1 and M_2 , where $\langle M_1 \rangle \in P$ and $\langle M_2 \rangle \notin P$. I.e., P is not trivial, it holds for some TMs, but not for all.

Show that P is undecidable.

Fact

Both conditions in Rice's theorem are necessary for proving that P is undecidable

Proof:

1. Let $P = \{\langle M \rangle \mid M \text{ is a TM with 5 states}\}$. P is nontrivial, so it satisfies the condition (2) of Rice's theorem. But in this case P could be decided by checking the number of states of the input TM M .
2. Let P be the empty set. Then it does not contain any TM and so it satisfies the condition (1) of Rice's theorem. But in this case P can be decided by a TM that always rejects.

Therefore both properties in Rice's theorem are necessary for proving that P is undecidable.

Proof, continuation

- If $w \in A_{TM}$, $L(M_w) = L(M_1)$ and $\langle M_w \rangle$ should be accepted by T_P since $\langle M_1 \rangle \in P$
- If $w \notin A_{TM}$, $L(M_w) = \emptyset = L(T_\emptyset)$ and $\langle M_w \rangle$ should be rejected by T_P by the assumption that $\langle T_\emptyset \rangle \notin P$.

Conclusion: $w \in A_{TM}$ iff $\langle M_1 \rangle \in P$. However, since A_{TM} is undecidable, hence, P is not decidable either

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}

Using Other Reductions

- Sometimes reducing from other (different from A_{TM}) undecidable languages, 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} .

Note

- 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.
- 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.
- EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

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