Almost There…

- Team Challenge
  - Focused on
    - Turing Machines
    - Variations of Turing Machines
    - Recursive & recursively enumerable languages
    - Vocabulary
    - Decision Problems
    - Decidability and Turing Machines
    - Turing-Decidable vs Turing Recognizable
    - Halting Problem

Types of Problems Related to a TM

- Problems about the **behavior** of a TM:
  - whether M runs for more than 10 steps on the empty string - decidable
  - whether M writes symbol a on the tape at some point - undecidable
- Problems about the **structure** of a TM:
  - whether M has more than 10 states - decidable
  - whether M has at least 5 transitions - decidable
- Problems about the **language** of a TM:
  - whether L(M) contains the empty string – nontrivial, therefore undecidable
  - whether L(M) is regular – nontrivial, therefore undecidable

Verify case by case

Create a decider

Non-trivial property
=> Undecidable

Common problems...

- Does M halts on all inputs?
- Does M not accept <M>?
- Does M accept w?
- Is L(M) regular?
- M has more than 5 states

UNDECIDABLE!

UNDECIDABLE!

UNDECIDABLE!

UNDECIDABLE!

DECIDABLE!

Identifying Decidable/Undecidable is the 1st **step**...
Chances are you'll need to use reductions/construction of a decider to **prove** your answers ;)

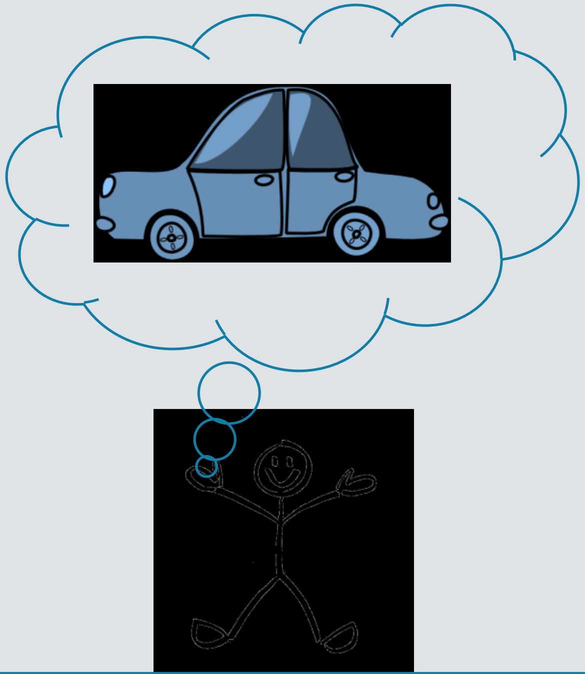
CHAPTER 5

Reducibility

Reduction

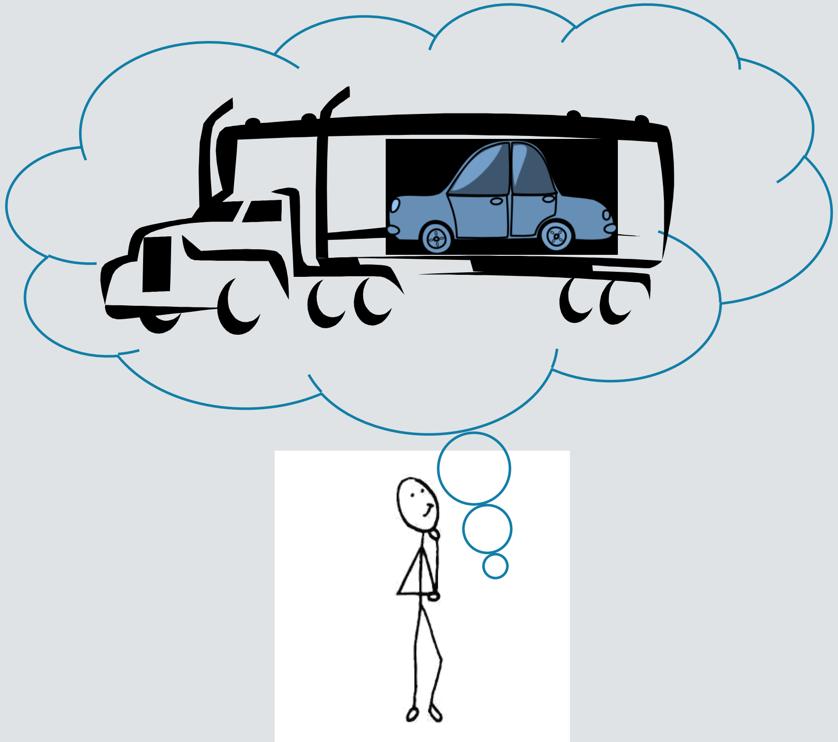
- What is reduction?
  - Is a **powerful method** which can be used to prove that a language A has some property based on the fact that another language B has the same property.
- Reduction method idea
  - Consider two problems A and B, if we can use the solution to B to construct a solution to A, we say that A can be reduced to B (or *A is reducible to B*)

A Simple Reduction



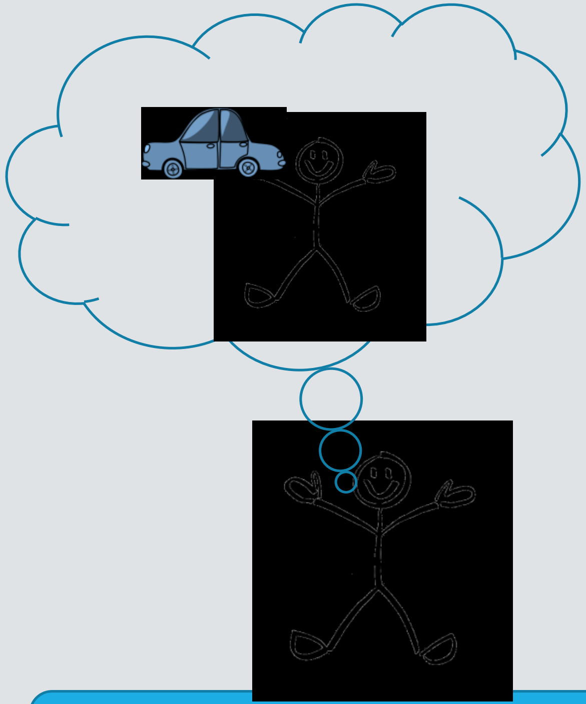
I can't lift a car

Given



Can I lift a truck?

New Question



No!... Because that would mean I can lift the car

Answer

Simple Reductions

- Examples
  - The problem of traveling from Boise to Scotland

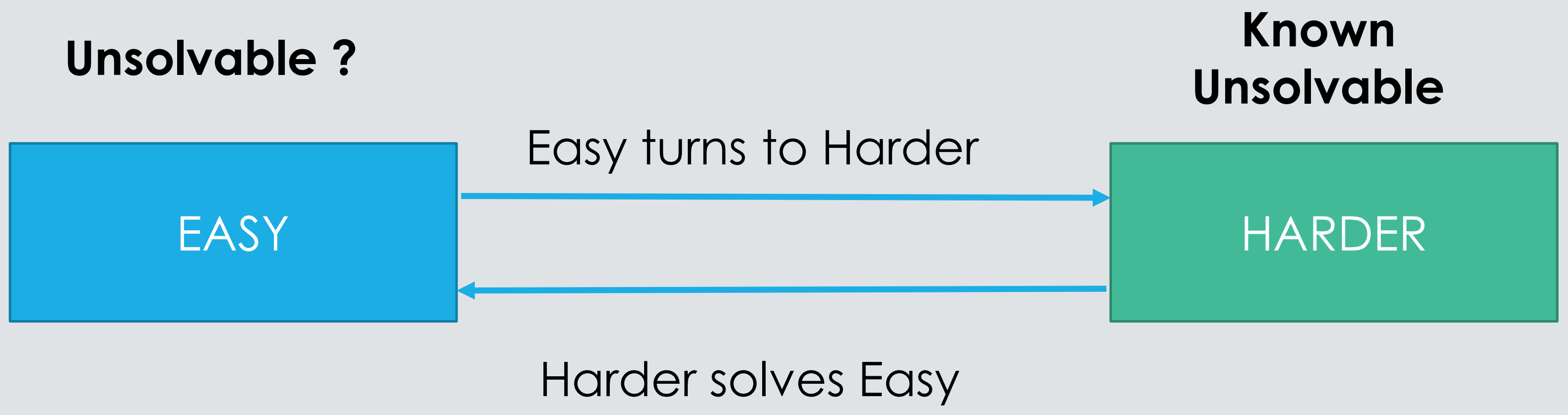
Can be reduced to the problem of buying a plane ticket between two cites

  - The problem of measuring the area of a rectangle

Can be reduced to the problem of measuring its length and width

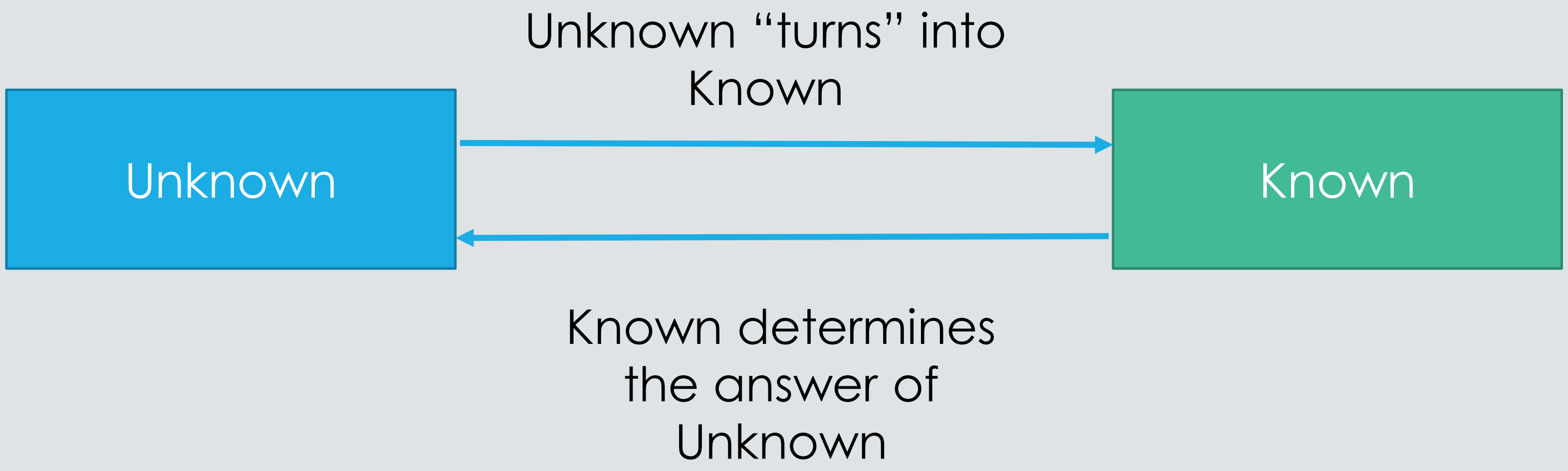
Relating Hard/Easy Problems

- A reduction is a way of solving a problem EASY with a problem HARDER
  - Basically, turning a instance of EASY into an instance of HARDER
- If we can't solve Easy, and we can reduce Easy to Harder, then we can't solve Harder either



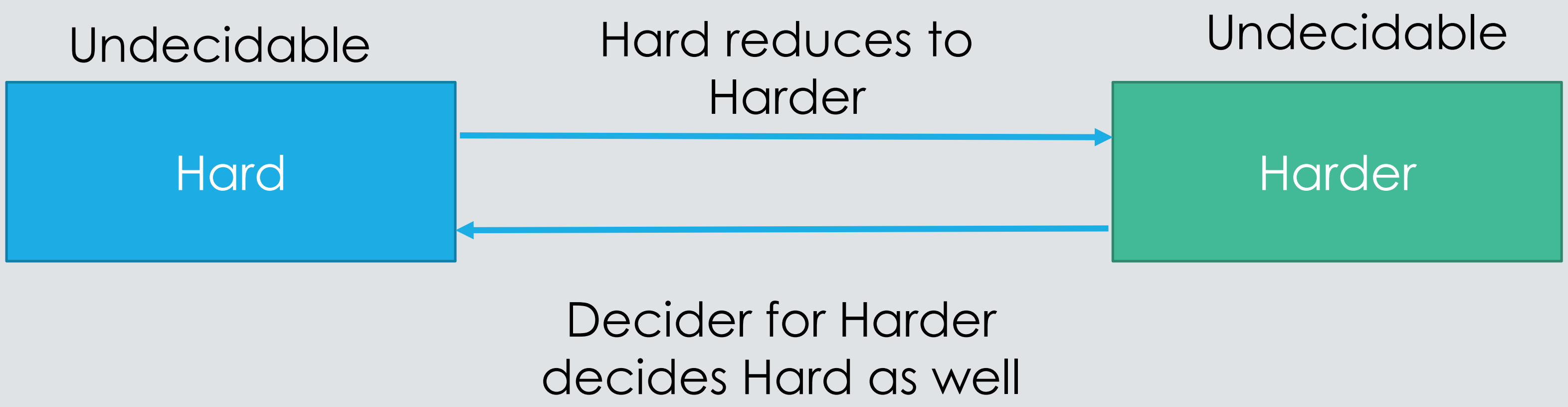
Relating Known/Unknown Answers

- A reduction is a way of providing an answer for a new (unknown) problem, based on the answer given by a known



Relating Reduction and Decidability

- If we can "reduce" Hard into a **decidable** problem Harder, then Harder can decide Hard
- If we can reduce an **undecidable** problem HARD to HARDER, then we know that we cannot decide HARDER



More Reduction Examples

- The problem of determining if a language is decidable. Consider A and B are two languages and a solution is a decider. In this case if A can be reduced to B:
  - If B is decidable then A is decidable
  - If A is undecidable then B is undecidable

- The problem of determining if a language is Turing-recognizable. Consider A and B are two languages and a solution is a TM. In this case if A can be reduced to B
  - If B is Turing-recognizable then A is Turing-recognizable
  - If A is Turing-unrecognizable then B is Turing-unrecognizable

Solving Problems Using Reduction

- Recall:
  - A_{TM} is the problem of determining whether a TM accepts a given input
  - A_{TM} is undecidable, based on the proof covered in Chapter 4 (Theorem 4.11)
- Is the problem of determining whether a TM halts on a given input decidable?
  - $HALT_{TM}$={ <M,w> | M is a TM and M halts on input w}
  - Theorem 5.1: $HALT_{TM}$ is undecidable
  - Proof

Lets assume that a TM R decides $HALT_{TM}$ and lets construct a TM S to decide A_{TM} :
S= on input <M,w>, the encoding of a TM M and a string w

1. Run TM R on <M,w>
2. If R rejects, *reject*
3. If R accepts, simulate M on w until it halts
4. If M has accepted, *accept*, if M has rejected, *reject*.

Clearly, if R decides $HALT_{TM}$ then S decides A_{TM} , which is a contradiction, since A_{TM} is undecidable . Consequently, $HALT_{TM}$ is undecidable.

Solving Problems Using Reduction

- Let TOTAL ={<M>| M is a TM that halts on all inputs}
  - How to prove this is undecidable?
    - Reduce HALT to TOTAL
    - Show how a decider for TOTAL could be used to build a decider for HALT
    - Conclude such a decider can't exist

This is a key step in most reductions:
Build a TM that has a property of the *new problem* (e.g., total) based on whether other TM has a property of the *old problem* (e.g., HALT).
Deciding whether this TM has the new property decides whether some other TM has the old property

Assume TOTAL is decidable.
Let T be a decider for TOTAL
H = "On input <M, w> :
    Construct the TM M' = "On input x:
        Ignore x.
        Run M on w.
        If M accepts w, accept.
        If M rejects w, reject."
    Run T on <M'> .
    If T accepts, *accept*.
    If T rejects, *reject*."

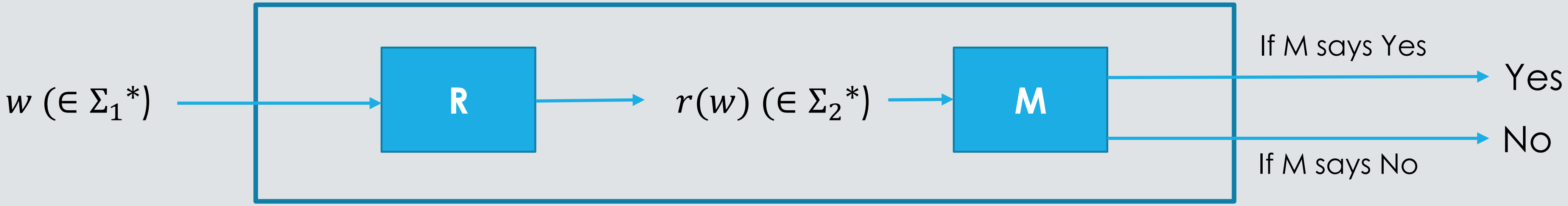
1. Assume TOTAL is decidable
2. Build a TM H that accepts <M,w> and constructs a TM M' that is a decider if M halts on w
3. Using the decider for TOTAL, H checks whether M' is a decider:
   1. If M' is a decider, then M halts on w
   2. If M' is not a decider, then M does not halt on w
4. Conclude that if TOTAL is decidable, then HALT is decidable
5. But HALT is undecidable, so the initial assumption was wrong and TOTAL is undecidable.

Mapping Reducibility

- Mapping Reducibility:
  - Transform the instances of the new problem into those of a problem that has been solved
- Definition: Language A is **mapping reducible** to a language B, written **$A \leq_m B$**, if there is a function f: $\Sigma^* \rightarrow \Sigma^*$, where for every w: $w \in A \Leftrightarrow f(w) \in B$. The function *f* is called **reduction** from A to B.
- Intuitive idea:
  - Let L be a language over alphabet Σ_1 and Q be a language over Σ_2. L is mapping reducible to Q if there exists a Turing computable function $r : \Sigma_1 \rightarrow \Sigma_2$ such that $w \in L$ if, and only if, $r(w) \in Q$.

Mapping Reducibility

- Let R be the TM that computes the reduction, i.e., maps inputs from L to inputs from Q, and M the TM that accepts language Q. The sequential execution of R and M on strings from Σ_1 accepts language L (by accepting inputs to Q) is



  - R, the reduction TM, which **does not determine membership** in either L or Q, transforms strings from $\Sigma_1{}^*$ to $\Sigma_2{}^*$.
  - Strings in Q are accepted/rejected by M, and strings in L are accepted/rejected based on the combination of R and M.

Using Mapping Reducibility

- Theorem 5.22: Let A and B be any two languages. If A $\leq_m$ B and B is decidable, then A is decidable
  - Basically, if a language A is reducible to a **decidable** language B by a function r, then A is also **decidable**.
  - Proof:

Let M be a decider for B and f be a reduction from A to B.
N= On input w
1. Compute f(w)
2. Run M on input f(w) and output whatever M outputs.

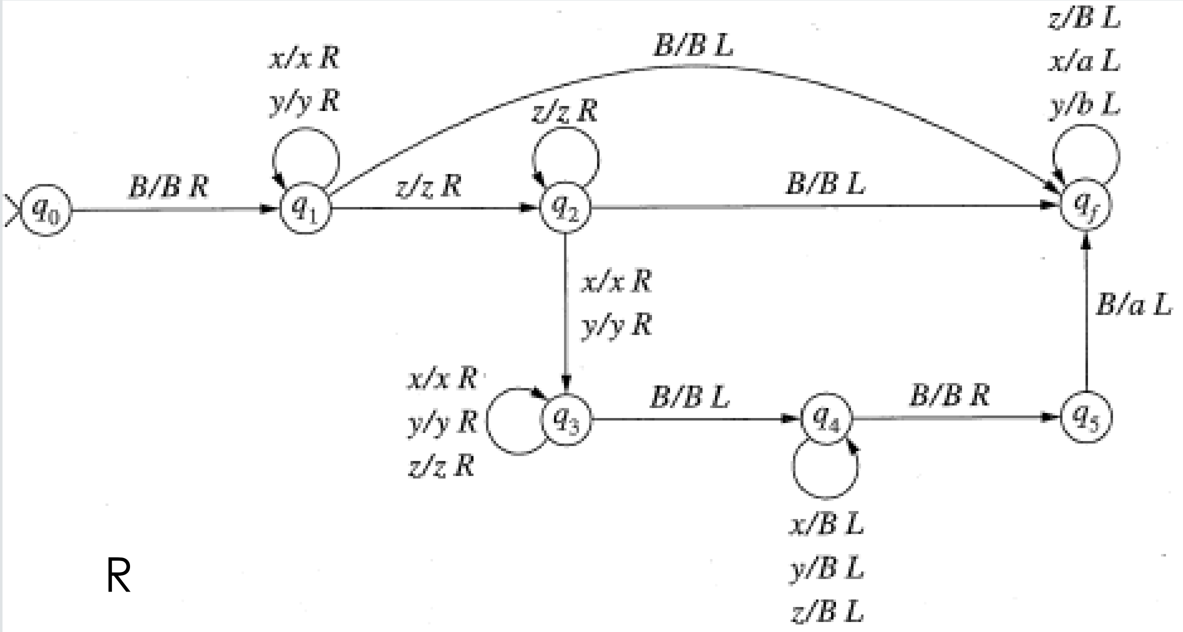
Clearly, If w ∈ A, then f(w) ∈ B because f is a reduction from A to B. Thus, M accepts f(w) whenever w ∈ A. Therefore, N works as expected.

- Corollary 5.24: Let A and B be any two languages. If A $\leq_m$ B and A is undecidable, then B is undecidable

Using Mapping Reducibility

- Decide if a string in L = {$x^i y^i z^k$ | i, k ≥ 0} is accepted/rejected based on a TM M that decides language Q = {$a^i b^i$ | i ≥ 0}
  - Basically, transform w ∈ {x, y, z}* to r(w) ∈ {a, b}*
    - If w ∈ x*y*z*, replace each 'x' by 'a' and 'y' by 'b', and erase the z's
    - Otherwise, replace w by a single 'a'

Reduction	Input	Condition
L	$w \in \{x, y, z\}^*$	$w \in L$
↓	$\downarrow r$	if and only if
Q	$r(w) \in \{a, b\}^*$	$r(w) \in Q$



Using Mapping Reducibility

- Remember that R only "reduces" the string from the alphabet of L to the alphabet of Q, you still need to determine acceptance/rejection of a string using M