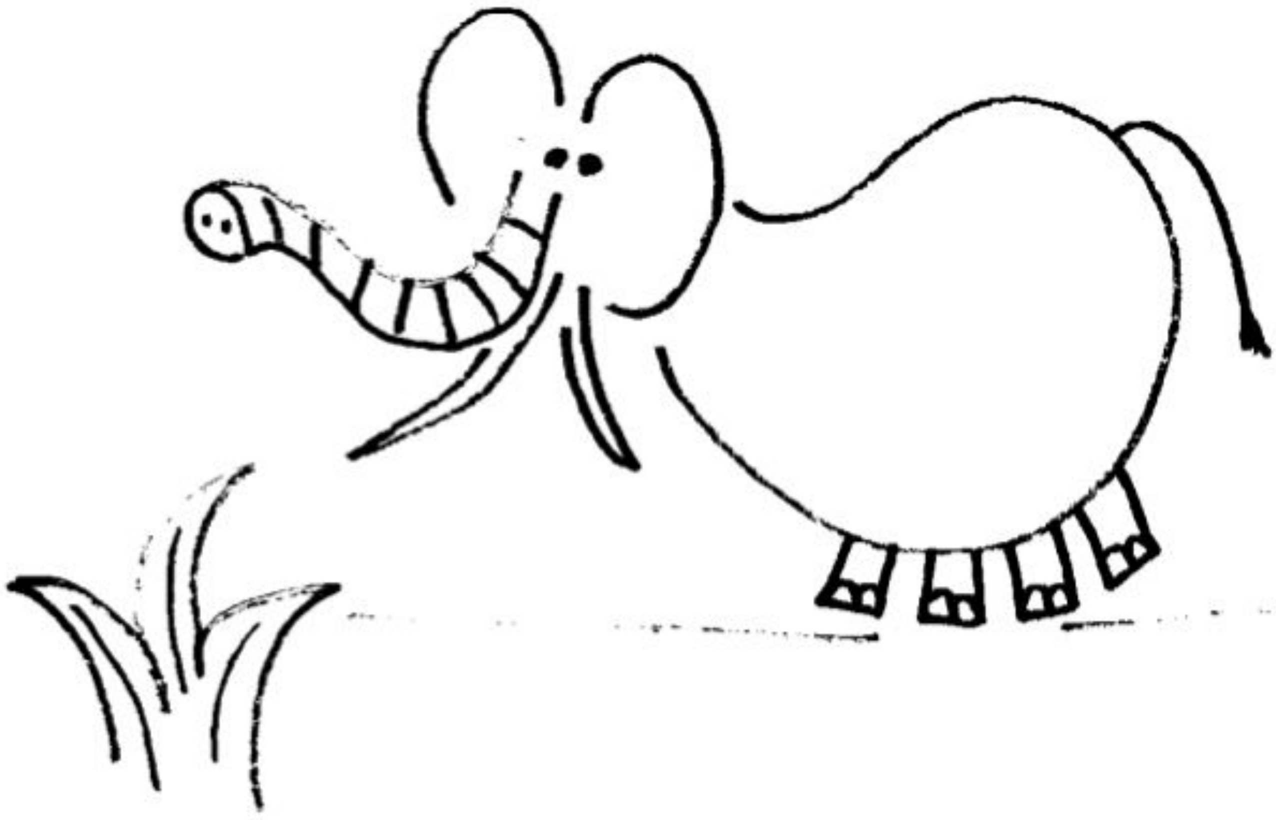
Reducibility

Sipser 5.1 (pages 187-198)

Reducibility



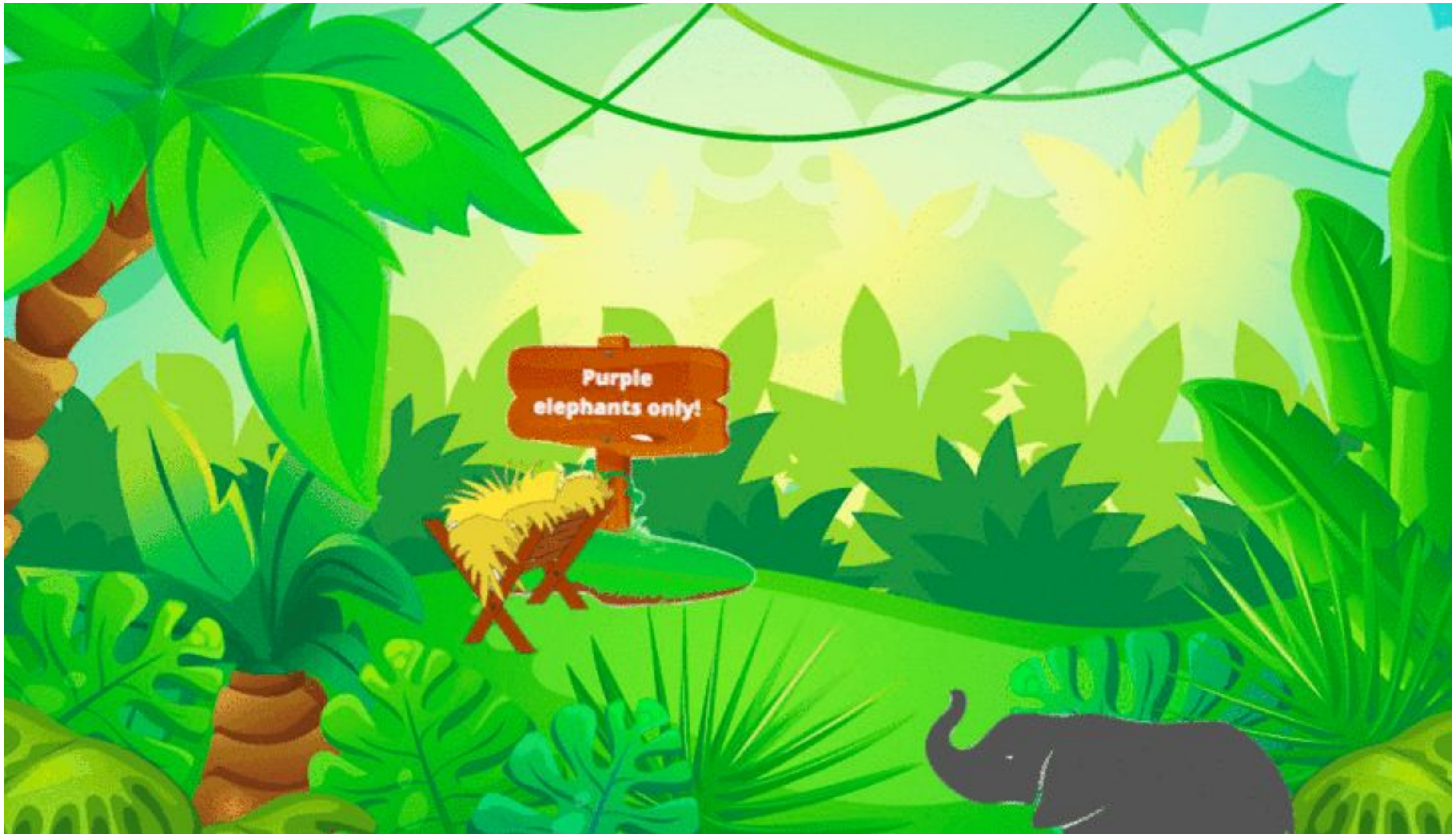


Image credit: Masha Lifshits (Fall 2019)

What it boils down to…

Driving directions

Cambridge

Western Mass

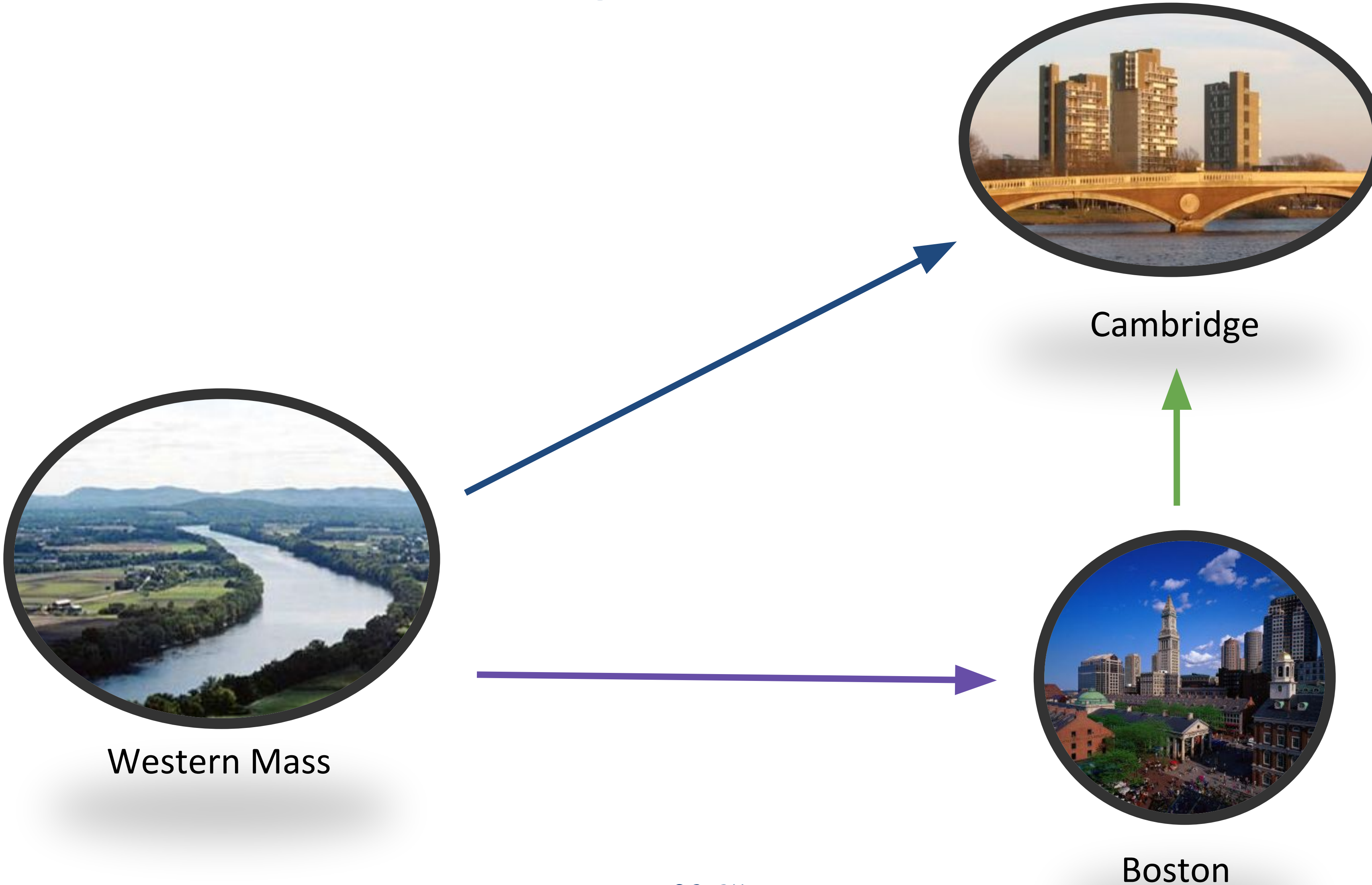
Driving directions

Cambridge

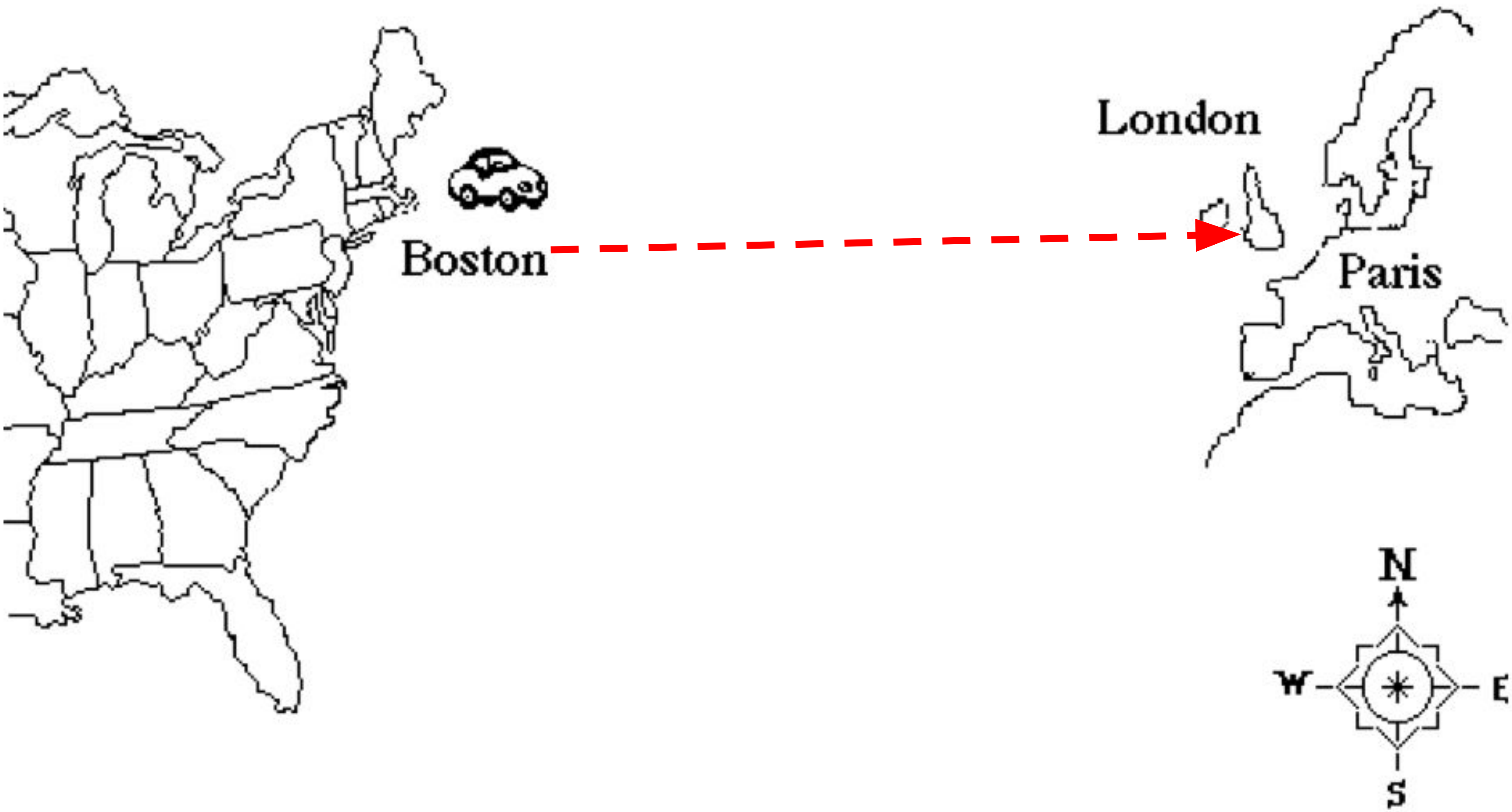
Western Mass

Boston

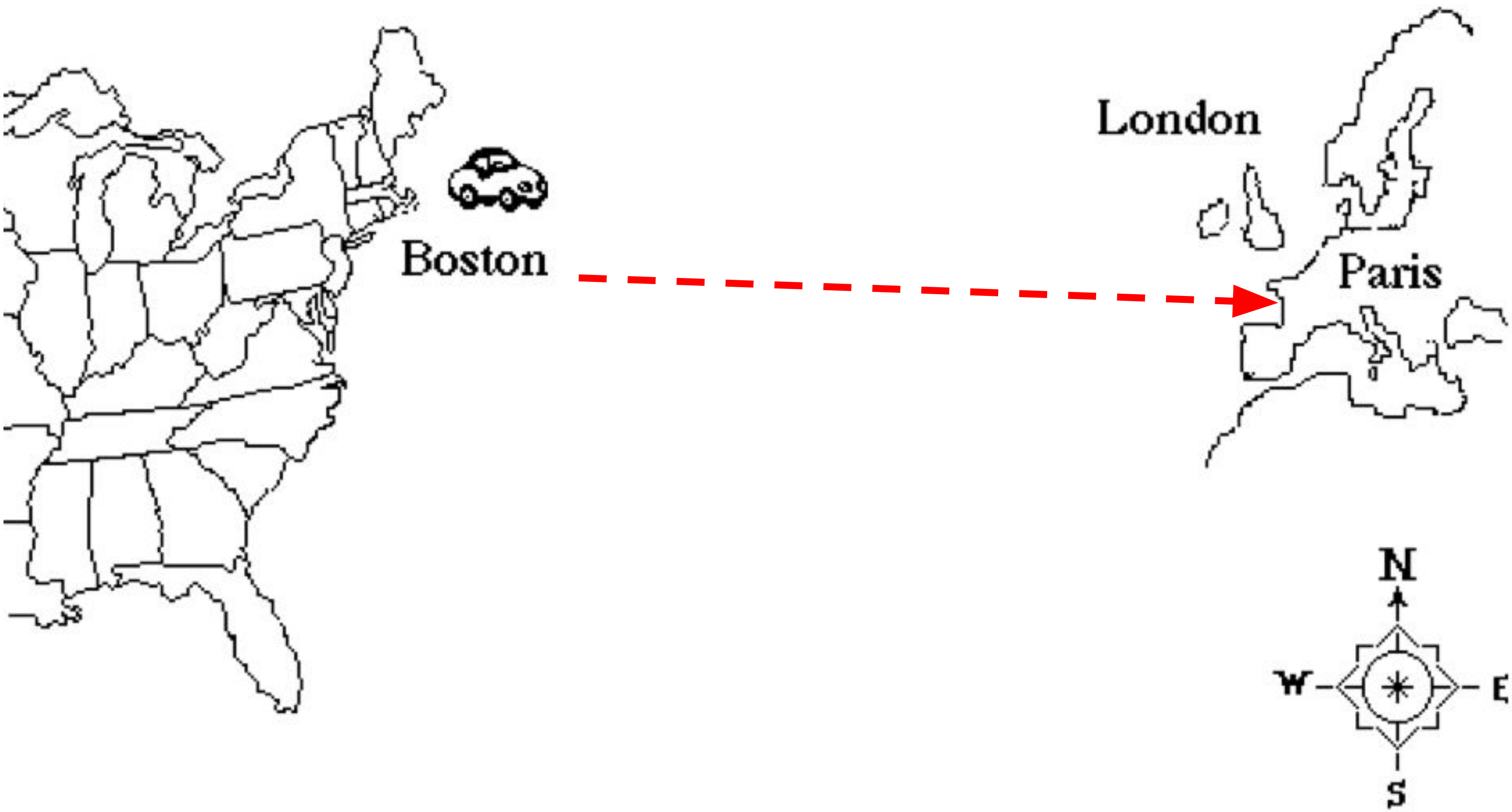
Driving directions



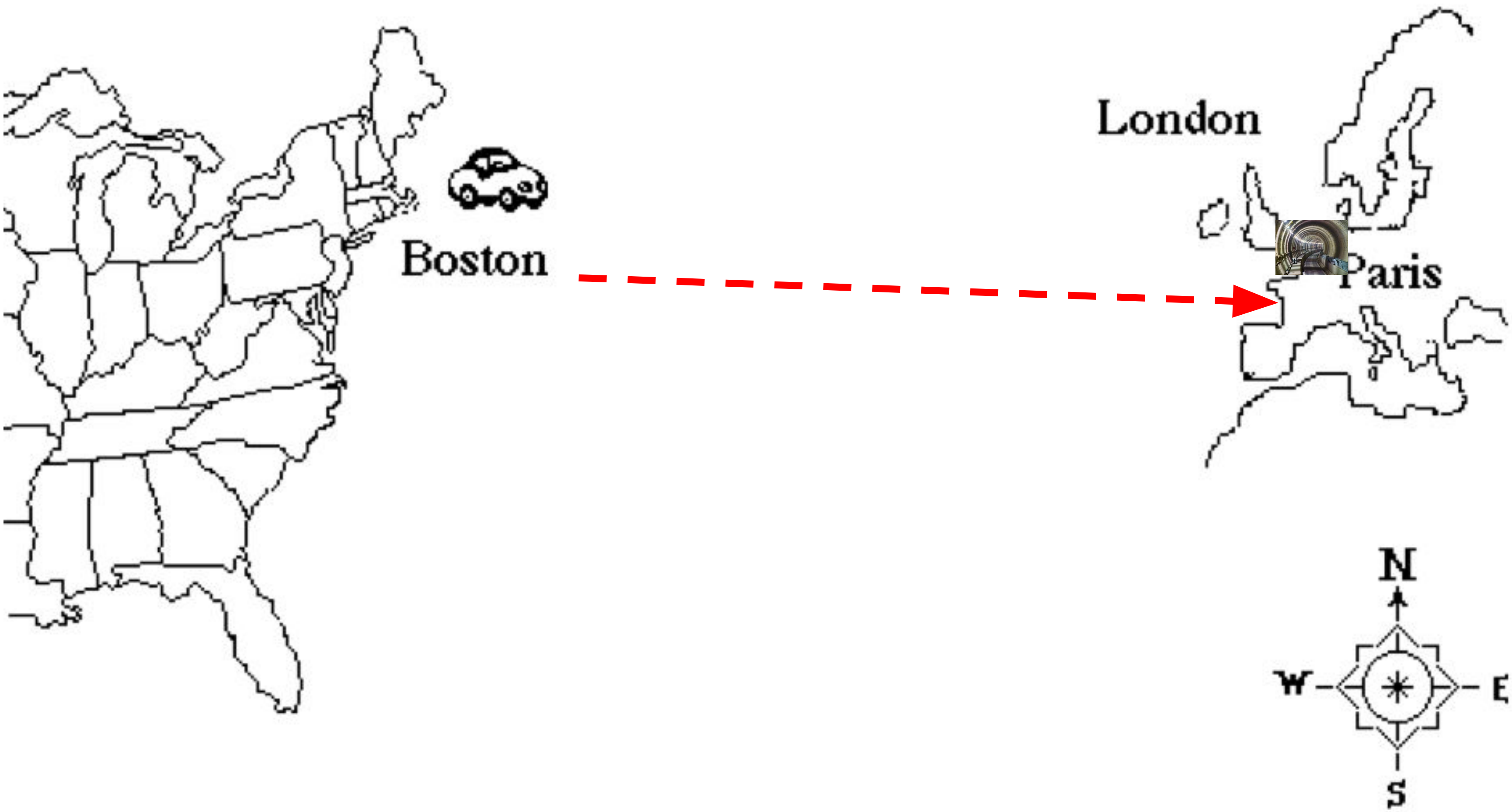
If you can't drive to London...



then you can't drive to Paris!



because… chunnel…



If something's impossible…

- Theorem 4.11:
  $A_{TM} = \{<M,w> \mid M$ is a TM and M accepts $w\}$ is undecidable.

If something's impossible…

- Theorem 4.11:
  $A_{TM} = \{<M,w> \mid M$ is a TM and M accepts $w\}$
  is undecidable.

- Define:
  $HALT_{TM} =$
  $\{<M,w> \mid M$ is a TM and M halts on input $w\}$
- Is $HALT_{TM}$ decidable?

The Halting Problem (again!)

- Theorem 5.1: $HALT_{TM}$ is undecidable.

- Proof Idea:
  - We know A_{TM} is undecidable.
  - We need to reduce one of $HALT_{TM}$ or A_{TM} to the other.
    - Which way to go?

$HALT_{TM}$ is undecidable.

- Proof:

  Suppose R decides $HALT_{TM}$. Define

  S = "On input $<M,w>$, where M is a TM and w a string:

  1. Run TM R on input $<M,w>$.
  2. If R rejects, then *reject*.
  3. If R accepts, simulate M on input w until it halts.
  4. If M enters its accept state, *accept*;

  if M enters its reject state, *reject*."

What about emptiness?

- $E_{TM} = \{<M> \mid M$ is a TM and $L(M) = \emptyset\}$
- Theorem 5.2: E_{TM} is undecidable.

A step along the way

- Given an input $<M,w>$, define a machine M_w as follows.
- M_w = "On input x:

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

$E_{TM} = \{<M> \mid M \text{ is a TM and } L(M) = \emptyset\}$

- Proof:

  Suppose TM R decides E_{TM}. Define a TM to decide A_{TM}

  S = "On input $<M,w>$:

  1. Use the description of M and w to construct M_w.
  2. Run R on input $<M_w>$.
  3. If R accepts, *reject*; if R rejects, *accept*."

With power comes uncertainty

	M accepts w	$L(M) = \emptyset$	$L(M_1) = L(M_2)$
Turing machines	✗	✗	✗
PDA	✓	✓	✗
Finite automata	✓	✓	✓

Is there anything that can be done?

- Rice's Theorem:

  Testing *any nontrivial property* of the languages recognized by Turing machines is undecidable!

We can't even tell when something's regular!

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

- Theorem 5.3: $REGULAR_{TM}$ is undecidable.

$REGULAR_{TM}$ is undecidable

- Proof:

  Assume R is a TM that decides $REGULAR_{TM}$.

  Define S = "On input $<M,w>$:

  1. Construct TM

     M_2 = "On input x:

     1. If x has the form 0^n1^n, *accept*.
     2. Otherwise, run M on input w and *accept* if M accepts w."

  2. Run R on input $<M_2>$.
  3. If R accepts, *accept*; if R rejects, *reject*."