Is This Really Interesting?

We now know that there are languages that are not Turing recognizable, but we do not know what kind of languages are non-TM-recognizable.

Are there interesting languages for which we can prove that there is no Turing machine that recognizes it?

Proving Undecidability (1)

Recall the language

A_{TM} = { <M,w> | M is a TM that accepts w }.

Proof that A_{TM} is not TM-decidable (Thm. 4.11)

(Contradiction) Assume that TM G decides A_{TM}:

$$G\langle M, w\rangle = \begin{cases} \text{"accept" if M accepts w} \\ \text{"reject" if M does not accept w} \end{cases}$$

From G we construct a new TM D that will get us into trouble…

Proving Undecidability (2)

The TM D works as follows on input <M> (a TM):

1) Run G on <M,<M>>
2) Disagree with the answer of G

(The TM D always halts because G always halts.)

In short: $D\langle M\rangle = \begin{cases} \text{"accept" if G rejects } \langle M, \langle M\rangle\rangle \\ \text{"reject" if G accepts } \langle M, \langle M\rangle\rangle \end{cases}$

Hence: $D\langle M\rangle = \begin{cases} \text{"accept" if M does not accept } \langle M\rangle \\ \text{"reject" if M does accept } \langle M\rangle \end{cases}$

Now run D on <D> ("on itself")…

Proving Undecidability (3)

Result: $D\langle D\rangle = \begin{cases} \text{"accept" if D does not accept } \langle D\rangle \\ \text{"reject" if D does accept } \langle D\rangle \end{cases}$

This does not make sense: D only accepts if it rejects, and vice versa.
(Note again that D always halts.)

Contradiction: **A_{TM} is not TM-decidable**.

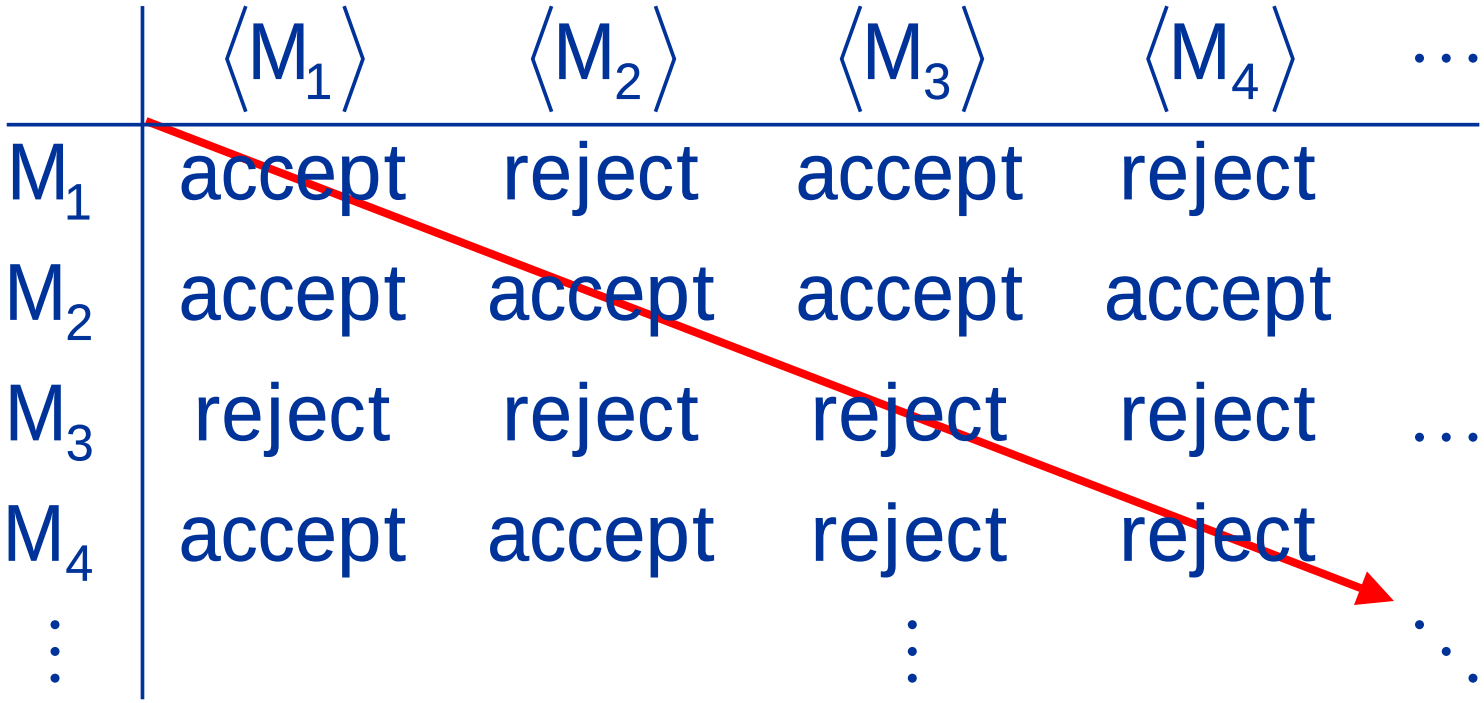
This proof used diagonalization implicitly…

Review of Proof (1)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	…
M_1	accept		accept		
M_2	accept	accept	accept	accept	
M_3					…
M_4	accept	accept			
⋮			⋮		⋱

‘Acceptance behavior’ of M_i on $\langle M_j \rangle$

Review of Proof (2)

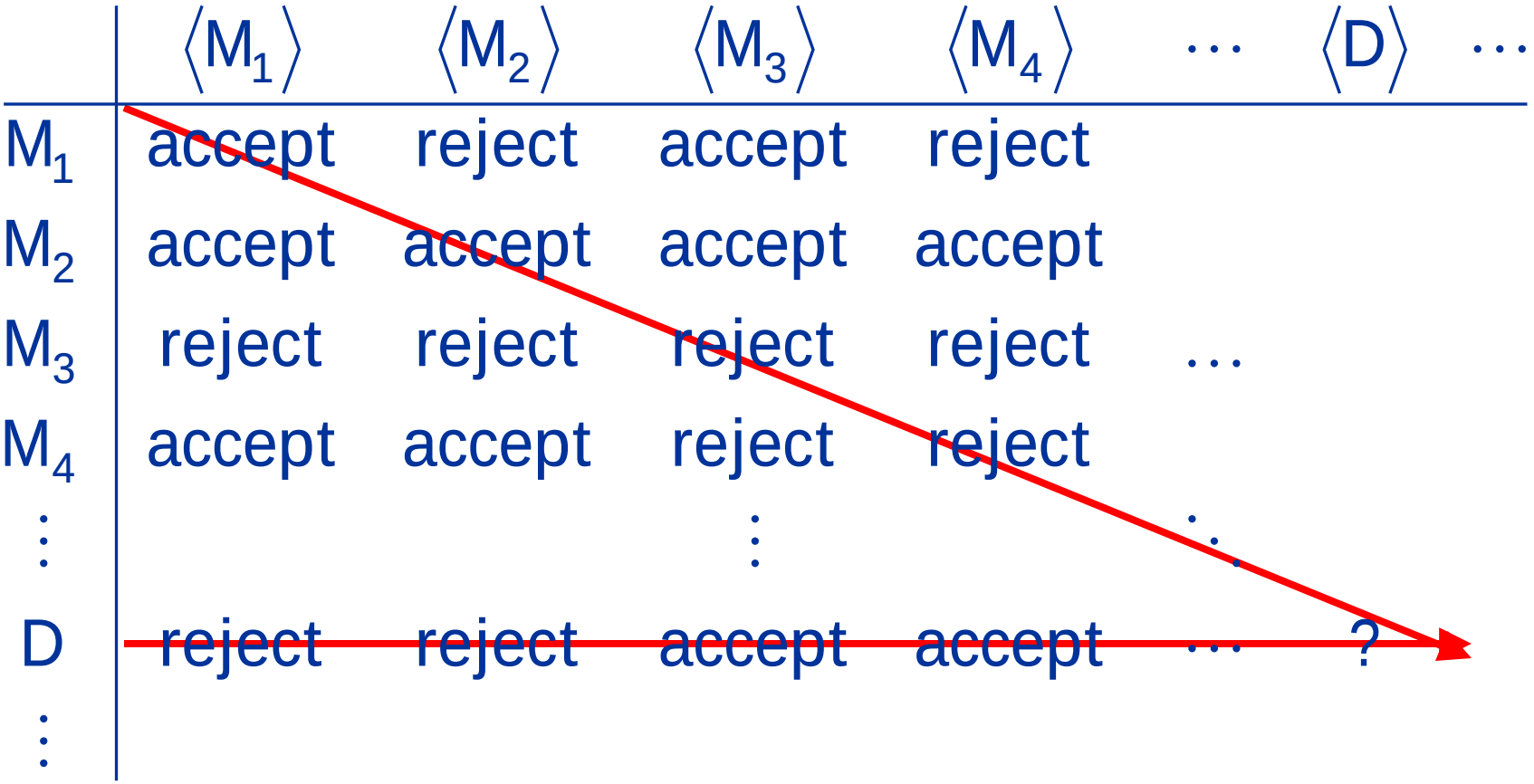


'Deciding behavior' of G on $\langle M_i, \langle M_j \rangle \rangle$

Review of Proof (3)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	…	$\langle D \rangle$	…
M_1	accept	reject	accept	reject			
M_2	accept	accept	accept	accept			
M_3	reject	reject	reject	reject	…		
M_4	accept	accept	reject	reject			
.			.		.		

Review of Proof (4)



Contradiction for D on input <D>.

Another View of the Problem

The **"Self-referential paradox"** occurs when we force the TM D to disagree with itself.

On the one hand, D knows what it is going to do on input <D>, but then it decides to do something else instead.

"You cannot know for sure what you will do in the future, because then you could decide to change your actions and create a paradox."

Self-Reference in Math

The diagonalization method implements the self-reference paradox in a mathematical way.

In logic this approach is often used to prove that certain things are impossible.

Kurt Gödel gave a mathematical equivalent of "This sentence is not true."

Old puzzle: In a town, there is a barber who shaves all those who do not shave themselves.
Who shaves the barber ?

Self-Reference in CSE

What happens if a computer program M tries to answer questions about itself <M>?

Sometimes this is perfectly okay:
- How big is <M>?
- Is <M> a proper TM?

Other questions lead to paradoxes:
- Does <M> halt or not?
- Is there a smaller program M' that is equivalent?

TM-Unrecognizable

A_{TM} is not TM-decidable, but it is TM-recognizable. What about a language that is not recognizable?

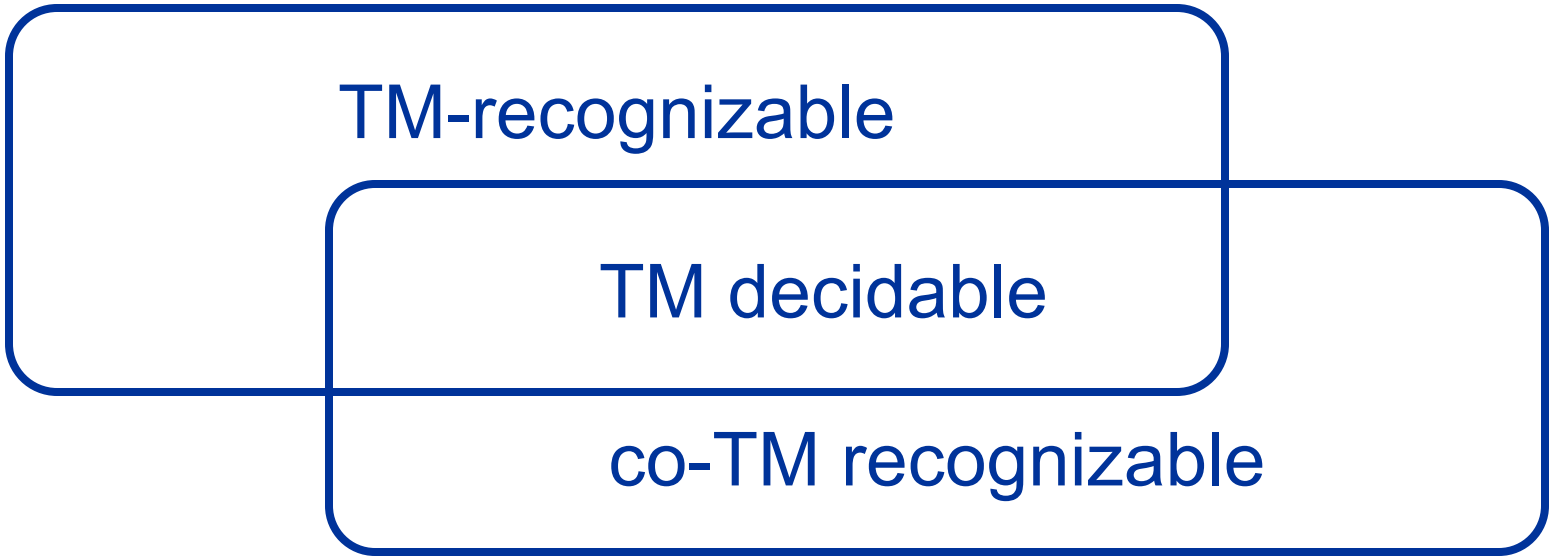
Theorem 4.22: If a language A is recognizable and its complement $\bar{A}$ is recognizable, then A is Turing machine decidable.

Proof: Run the recognizing TMs for A and $\bar{A}$ in parallel on input x. Wait for one of the TMs to accept. If the TM for A accepted: “accept x”; if the TM for $\bar{A}$ accepted: “reject x”.

$\bar{A}_{TM}$ is not TM-Recognizable

By the previous theorem it follows that $\bar{A}_{TM}$ cannot be TM-recognizable, because this would imply that A_{TM} is TM decidable (Corollary 4.23).

We call languages like $\bar{A}_{TM}$ co-TM recognizable.



Things that TMs Cannot Do:

The following languages are also unrecognizable:

E_{TM} = { <G> | G is a TM with L(G)=∅ }

EQ_{TM} = { <G,H> | G and H are TMs with L(G)=L(H) }

To be precise:

- E_{TM} is co-TM recognizable
- EQ_{TM} is not even co-Turing recognizable

How can we prove these facts?

Next: reducibility

- We still need to ***prove*** that the Halting problem is undecidable.
- Do more examples of undecidable problems.
- Try to get a general technique for proving undecidability.

Halting problem

- Assume that it is decidable. So there is a TM S that decides

HALT={<M,w>|M is a TM and M halts on w}

- Use S as a subroutine to get a TM S to decide

A_{TM} = {<M,w> | M is a TM that accepts w }

- Therefore A_{TM} is decidable. CONTRADICTION!
- Details follow ….

Halting problem - 2

S = “On input <M,w>

- Run TM R on input <M,w>.
- If R rejects, REJECT.
- If R accepts, simulate M on w until it halts.
- If M has accepted, ACCEPT, else REJECT”

More undecidability

E_{TM} = {<M>| M is a TM and L(M) = ϕ}
We mentioned that E_{TM} is co-TM recognizable.
We will prove next that E_{TM} is undecidable.

Intuition: You cannot solve this problem UNLESS you solve the halting problem!!

But this is hard to formalize, so we use A_{TM}. Instead.

E_{TM} is undecidable

Assume R decides E_{TM}. Use R to design TM S to decide A_{TM}.

- Given a TM M and input w, define a new TM M':
  - If $x \neq w$, reject
  - If x=w, accept iff M accepts w

S = "On input <M,w>

- Construct M' as above.
- Run TM R on input <M'>.
- If R accepts, REJECT; If R rejects, ACCEPT."

EQ_{TM} is undecidable

If this is decidable, then we can solve E_{TM}!! (You need to check equality with TM M_1 that rejects all inputs)

Assume R decides EQ_{TM}. Use R to design TM S to decide E_{TM}.

S = "On input <M>

- Run TM R on input <M, M_1>.
- If R accepts, ACCEPT; If R rejects, REJECT."

The running idea

All our proofs had a common structure

- The first undecidable proof was hard – used diagonalization/self-reference.
- For the rest, we assumed decidable and used it as a subroutine to design TM's that decide known undecidable problems.
- Can we make this technique more structured?

Mapping Reducibility

Thus far, we used reductions informally:

If "knowing how to solve A" implied "knowing how to solve B", then we had a **reduction** from B to A.

Sometimes we had to negate the answer to the "$\in$A?" question, sometimes not. In general, it was unspecified which transformations were allowed around the "$\in$A?"-part of the reduction.

Let us make this formal…

Computable Functions

A function $f:\Sigma^* \rightarrow \Sigma^*$ is a TM-computable function if there is a Turing machine that on every input $w \in \Sigma^*$ halts with just f(w) on the tape.

All the usual computations (addition, multiplication, sorting, minimization, etc.) are all TM-computable.

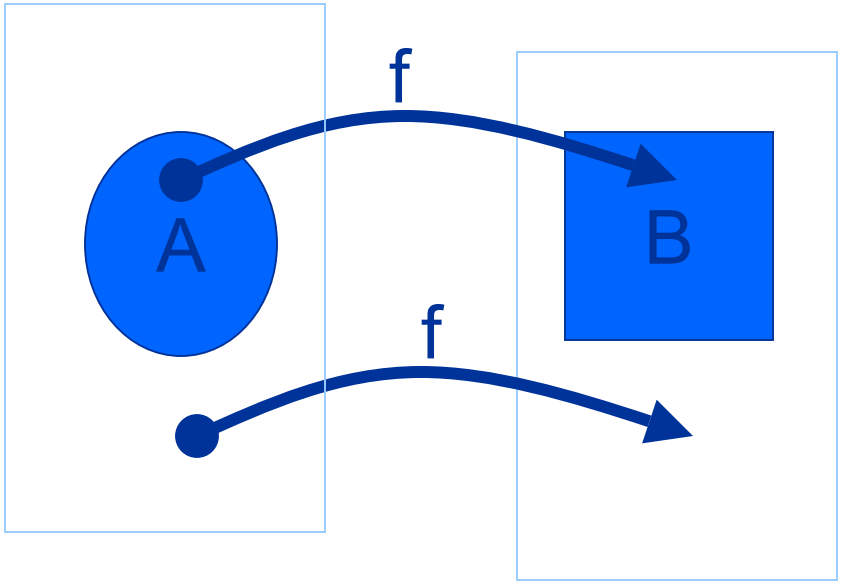
Note: alterations to TMs, like "given a TM M, we can make an M' such that…" can also be described by computable functions that satisfy f(<M>) = <M'>.

Mapping Reducible

A language A is mapping reducible to a another language B if there is a TM-computable function $f:\Sigma^* \rightarrow \Sigma^*$ such that: $\mathbf{w \in A \iff f(w) \in B}$ for every $w \in \Sigma^*$.

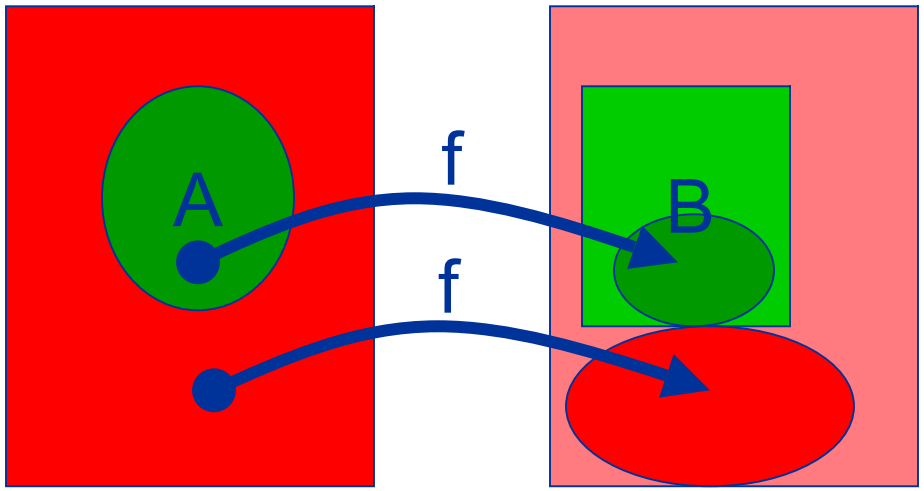
Terminology/notation:

- $A \leq_m B$
- function f is the reduction of A to B
- also called: "*many-one reducible*"



$A \leq_m B$

The language B can be more difficult than A.



Typically, the image f(A) is only a subset of B, and f(Σ*\A) a subset of Σ*\B.

"Image f(A) can be the easy part of B".

Previous mappings used

$A_{TM} \leq_m HALT_{TM}$

F = "On input <M,w>

- Construct TM M' = "on input x:
  - Run M on x
  - If M accepts, ACCEPT
  - If M rejects, enter infinite loop."
- Output <M',w>"

Check: f maps < M,w> to <M', w'>.

$<M,w> \in A_{TM} \Longleftrightarrow <M',w> \in HALT_{TM}$

Previous mappings used - 2

Recall: M_1 rejects all inputs. Assume R decides EQ_{TM}. Use R to design TM S to decide E_{TM}.

S = "On input <M>

- Run TM R on input <M, M_1>.
- If R accepts, ACCEPT; If R rejects, REJECT."

Check: f maps < M> to <M, M_1>.

<M> $\in E_{TM} \Longleftrightarrow$ <M , M_1> $\in EQ_{TM}$

Decidability obeys $\leq_m$ Ordering

Theorem 5.22: If $A \leq_m B$ and B is TM-decidable, then A is TM-decidable.

Proof: Let M be the TM that decides B and f the reducing function from A to B. Consider the TM:

On input w:

1) Compute f(w)
2) Run M on f(w) and give the same output.

By definition of f: if $w \in A$ then $f(w) \in B$.

M "accepts" f(w) if $w \in A$, and

M "rejects" f(w) if $w \notin A$.

Undecidability obeys $\leq_m$ Order

Corollary 5.23: If $A \leq_m B$ and A is undecidable, then B is undecidable as well.
Proof: Language A undecidable and B decidable contradicts the previous theorem.

Extra: If $A \leq_m B$, then also for the complements $(\Sigma^* \backslash A) \leq_m (\Sigma^* \backslash B)$
Proof: Let f be the reducing function of A to B with $w \in A \Longleftrightarrow f(w) \in B$. This same computable function also obeys "$v \in (\Sigma^* \backslash A) \Longleftrightarrow f(v) \in (\Sigma^* \backslash B)$" for all $v \in \Sigma^*$

Recognizability and $\leq_m$

Theorem 5.28: If $A \leq_m B$ and B is TM-recognizable, then A is TM-recognizable.

Proof: Let M be the TM that recognizes B and f the reducing function from A to B. Again the TM:

On input w:

1) Compute f(w)
2) Simulate M on f(w) and give the same result.

By definition of f: $w \in A$ equivalent with $f(w) \in B$.

M "accepts" f(w) if $w \in A$, and

M "rejects" f(w)/does not halt on f(w) if $w \notin A$.

Unrecognizability and $\leq_m$

Corollary 5.29: If $A \leq_m B$ and A is not Turing-recognizable, then B is not recognizable as well.

Proof: Language A not TM-recognizable and B recognizable contradicts the previous theorem.

Extra: If $A \leq_m B$ and A is not co-TM recognizable, then B is not co-Turing-recognizable as well.

Proof: If A is not co-TM-recognizable, then the complement ($\Sigma^* \backslash A$) is not TM recognizable.
By $A \leq_m B$ we also know that $(\Sigma^* \backslash A) \leq_m (\Sigma^* \backslash B)$.
Previous corollary: ($\Sigma^* \backslash B$) not TM recognizable, hence B not co-Turing-recognizable .

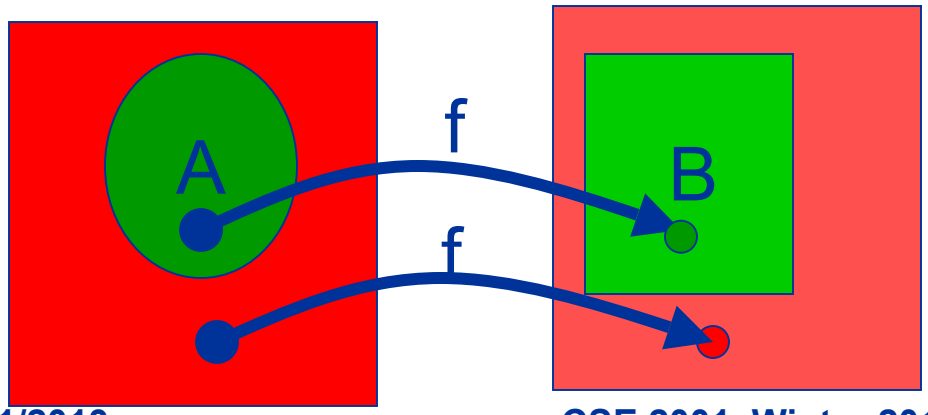
Decidable $A \leq_m B$

If A is a decidable language, then $A \leq_m B$ for every nontrivial B. (Let $1 \in B$ and $0 \notin B$.)

Because A is decidable, there exists a TM M such that M outputs "accept" on every $x \in A$, and "reject" on $x \notin A$.

We can use this M for a TM-computable function f with

$f(x)=1 \in B$ if $x \in A$ and $f(x)=0 \notin B$ if $x \notin A$



"The function f does all the decision-work"

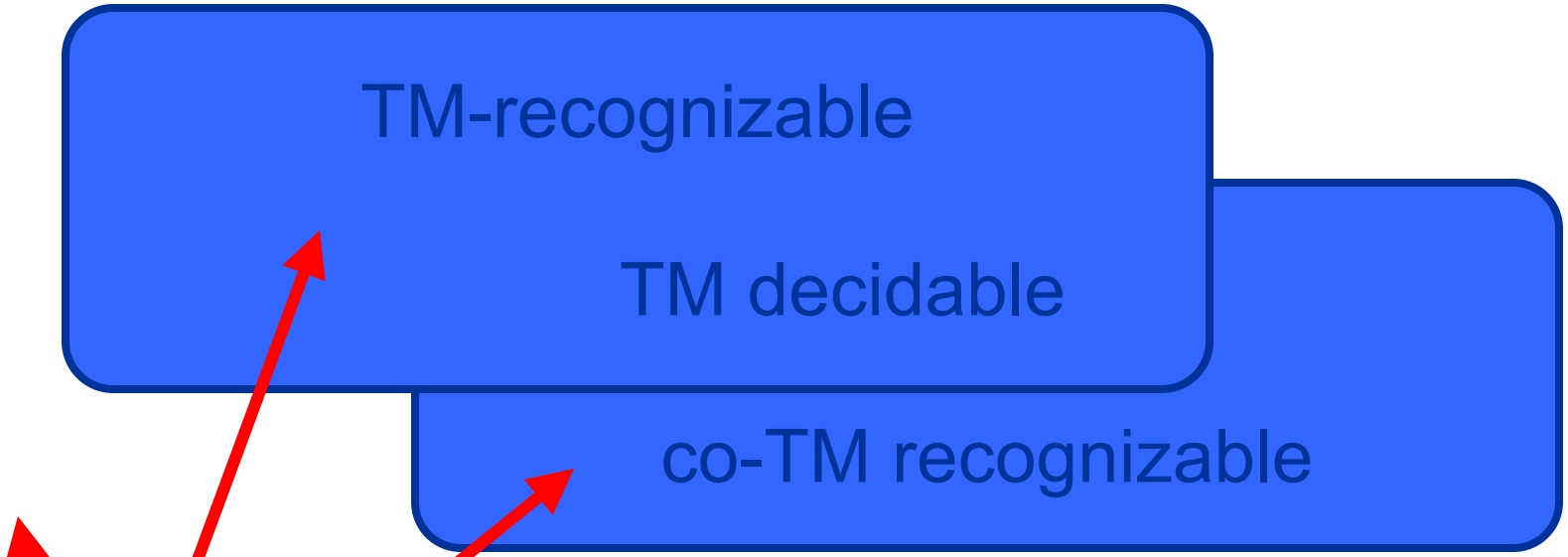
E_{TM} Revisited

Recall: The emptiness language was defined as
$E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM with } L(M)=\varnothing \}$
E_{TM} is not Turing recognizable.

Simple proof via ($\bar{A}_{TM} \leq_m E_{TM}$):
Let f on input $\langle M,w \rangle$ give $\langle M' \rangle$ as output with:
M': Ignore input
    Run M on w
    If M accepted w then "accept"
    otherwise "reject"

Now: $\langle M,w \rangle \in \bar{A}_{TM} \Longleftrightarrow f(\langle M,w \rangle) = \langle M' \rangle \in E_{TM}$

Something still unproven…



A_{TM} = { <M,w> | M is a TM that accepts w }

E_{TM} = { <M> | M is a TM with L(M)=∅ }

EQ_{TM} = { <G,H> | G,H TMs with L(G)=L(H) }

EQ_{TM} is not TM Recognizable

Proof (by showing $\bar{A}_{TM} \leq_m EQ_{TM}$):

Let f on input $\langle M,w \rangle$ give $\langle M_1,M_2 \rangle$ as output with:
M_1: "reject" on all inputs
M_2: Ignore input
Run M on w
"accept" if M accepted w

We see that with this TM-computable f:
$\langle M,w \rangle \in \bar{A}_{TM} \Longleftrightarrow f(\langle M,w \rangle) = \langle M_1,M_2 \rangle \in EQ_{TM}$

Because $\bar{A}_{TM}$ is not recognizable, so is EQ_{TM}.

EQ_{TM} not co-TM Recognizable

Proof (by showing $A_{TM} \leq_m EQ_{TM}$):

Let f on input $\langle M,w \rangle$ give $\langle M_1,M_2 \rangle$ as output with:
M_1: "accept" on all inputs
M_2: Ignore input
Run M on w
"accept" if M accepted w

We see that with this TM-computable f:
$\langle M,w \rangle \in A_{TM} \Longleftrightarrow f(\langle M,w \rangle) = \langle M_1,M_2 \rangle \in EQ_{TM}$

Because A_{TM} is not co-recognizable, so is EQ_{TM}.

Partial $\leq_m$ Ordering