More undecidable languages

Last lecture

- K = { $\langle M \rangle$ | M is a TM and M accepts $\langle M \rangle$ } is undecidable (Turing-undecidable, non-recursive).
- A_{TM} = { $\langle M,y \rangle$ | M is a TM and M accepts y }

Today

- $Halt_{TM}$ = { $\langle M,y \rangle$ | M is a TM and M halts on input y }
- E_{TM} = { $\langle M \rangle$ | M is a Turing machine and L(M) is empty }
- Π_p = { $\langle M \rangle$ | **p**(L(M)) = **true**} where p is any non-trivial property.
- …
- A general technique to prove undecidability.

Compare the difficulty

Consider two decision problems A and B.

- Mickey: provides an algorithm for A, but is unable to solve B.
- Mickey's conclusion: A is easier than B.

- Minnie: can't solve either problem, but
  - knows how to solve A if an algorithm for B is given.
- Fact. A cannot be more difficult than B.

Mapping Reducibility

Consider any two languages $A, B \subseteq \Sigma^*$.

A is said to be **mapping reducible** to B, denoted $A \leq_m B$, if there is a computable function **f**: $\Sigma^* \rightarrow \Sigma^*$ such that

- for every $x \in \Sigma^*$, $\mathbf{x} \in \mathbf{A}$ <u>if and only if</u> $\mathbf{f(x)} \in \mathbf{B}$.

$$\begin{tikzcd} x \arrow[r] & \boxed{f} \arrow[r] & f(x) \end{tikzcd}$$

NB. Mapping reducibility is also known as many-one reducibility

Mapping Reducibility

Consider any two languages $A, B \subseteq \Sigma^*$.

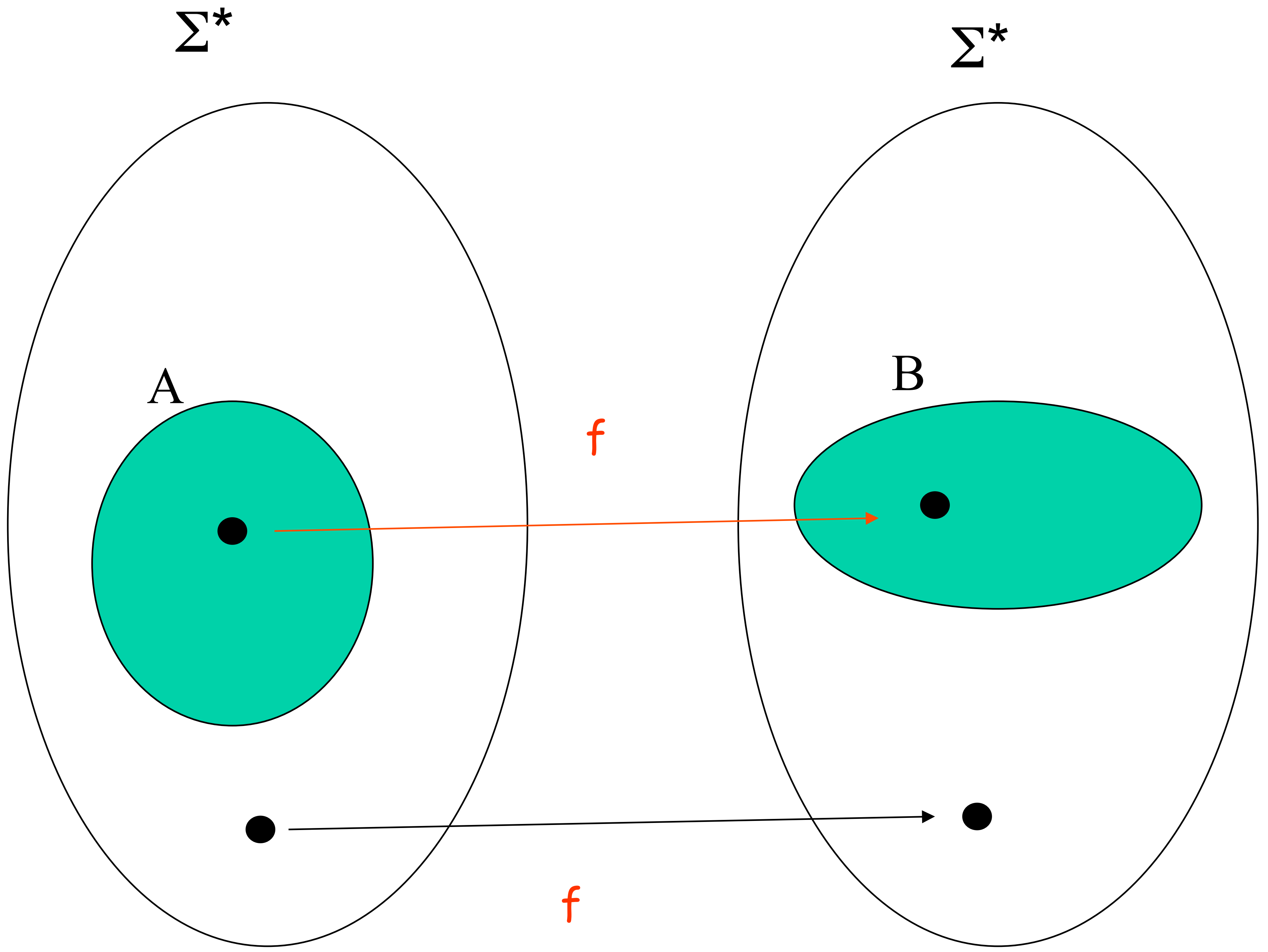
A is said to be **mapping reducible** to B, denoted $A \leq_m B$, if there is a computable function **f**: $\Sigma^* \rightarrow \Sigma^*$ such that

- for every $x \in \Sigma^*$, $\mathbf{x} \in \boldsymbol{A}$ if and only if $\mathbf{f(x)} \in \mathbf{B}$.

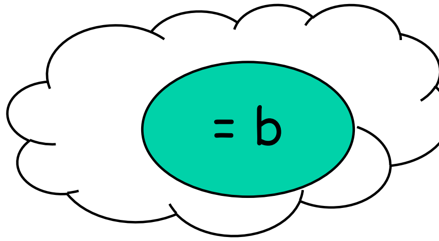
$$\begin{tikzcd} x \arrow[r] & \boxed{f} \arrow[r] & f(x) \end{tikzcd}$$

- Intuitively, $A \leq_m B$ means that we can transform the problem "Is $\mathbf{x} \in \boldsymbol{A}$?" to the problem "Is $\mathbf{f(x)} \in \mathbf{B}$?".

NB. Mapping reducibility is also known as many-one reducibility


Σ*
Σ*
A
B
f
f

Example



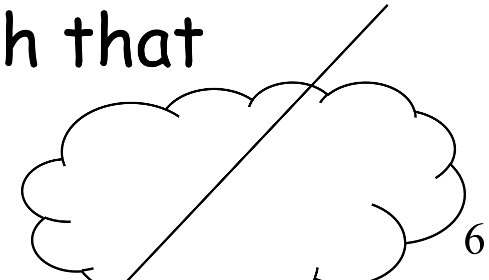
Knapsack problem: Given positive integers (a_1 , a_2 , ... , a_n , b), determine whether some a_i's have a sum equal to b.

- Precisely, does there exist $S \subseteq \{1, 2, ..., n\}$ such that $\Sigma_{i \in S} a_i = b$
- To simplify our discussion, we assume that $b < (a_1 + a_2 + ... + a_n)/2$.

Partition problem: Given m positive integers (w_1 , w_2 , w_3 ,..., w_m), determine whether these m numbers can be split into two parts with equal sums.

- Precisely, does there exist $Y \subseteq \{1, 2, ..., m\}$ such that

$\Sigma_{i \in Y} w_i = \Sigma_{i \notin Y} w_i$?



Knapsack problem $\leq_m$ Partition problem

iven an instance of the knapsack problem, $X = (a_1 , a_2 , \ldots , a_n , b$
we construct the following instance of the partition problem:

$f(X) = (w_1 , w_2 , \ldots , w_m)$, where $m = n+1$;
$w_1 = a_1$; $w_2 = a_2$; … $w_{m-1} = a_n$;
$w_m = (a_1 + a_2 + \ldots + a_n) - 2b$.

et $A = a_1 + a_2 + \ldots + a_n$, let $W = w_1 + w_2 + \ldots + w_m$.
hen $W = A + A -2b = 2A - 2b$.

Correctness

X has the answer Yes

$\Rightarrow$ there exists $S \subseteq \{1, 2, \dots, n\}$ such that $\Sigma_{i \in S}\, a_i = b$

$\Rightarrow \Sigma_{i \in S \cup \{m\}}\, w_i \quad = b + A - 2b$

$= A - b$

$= W/2 \qquad = \Sigma_{i \notin S \cup \{m\}}\, w_i$

$\Rightarrow$ f(X) has the answer Yes

f(X) has the answer Yes

$\Rightarrow$ there exists $Y \subseteq \{1, 2, \ldots, n+1\}$ such that $\Sigma_{i \in Y} w_i = \Sigma_{i \notin Y} w_i = W/2$, and Y contains m=n+1

$$\begin{aligned} \Rightarrow \Sigma_{i \in Y - \{m\}} a_i &= W/2 - (A - 2b) \\ &= A\text{-}b - (A\text{-}2b) \\ &= b \end{aligned}$$

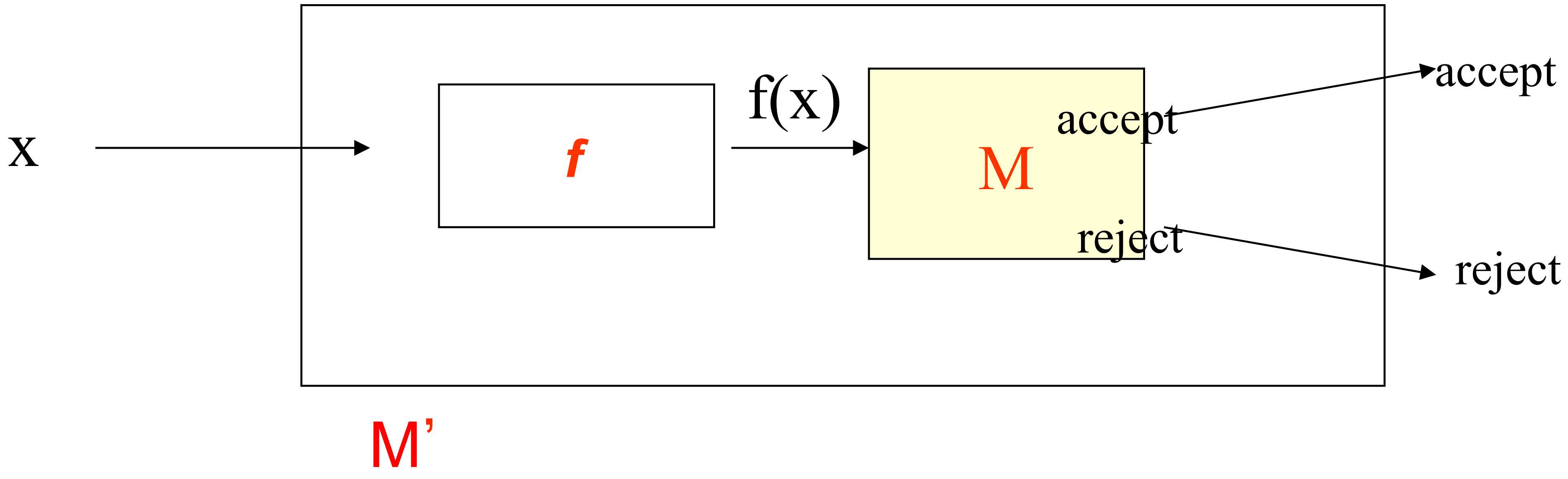
$\Rightarrow$ X has the answer Yes. [S = Y – {m}]

Therefore, the knapsack problem is mapping reducible to the partition problem.

Properties of mapping reducibility

Theorem 1. If $A \leq_m B$ and B is decidable, then A is decidabl

Proof: Let M be a TM deciding B. Construct a machine M' to decide A as follows: On input x,



M' decides A

For any $x \in \Sigma^*$,

- $x \in A \Rightarrow$ $f(x) \in B$ (by def. of f)
  $\Rightarrow$ M accepts f(x)
  $\Rightarrow$ M' accepts x

- $x \notin A \Rightarrow$ $f(x) \notin B$ (by def. of f)
  $\Rightarrow$ M rejects f(x)
  $\Rightarrow$ M' rejects x

NB. To show a language L is decidable, we can show that for some decidable language L', $L \leq_m L'$.

A more useful property

Corollary 2. If $A \leq_m B$ and A is undecidable, then B is undecidable.

Proof: Suppose on the contrary that B is decidable. Then by Theorem 1, A is decidable. A contradiction occurs.

Example:

$K \leq_m A_{TM}$, where

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

$K = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \langle M \rangle \}$.

By Corollary 2, as K is undecidable, A_{TM} is undecidable.

Example 1

emma. $K \leq_m A_{TM}$, where

$A_{TM} = \{ \langle M,y \rangle \mid M$ is a TM and M accepts $y \}$, and

$K = \{ \langle M \rangle \mid M$ is a TM and M accepts $\langle M \rangle \}$.

roof:

What is f ?

Is f computable ?

$x \in K \Leftrightarrow f(x) \in A_{TM}$?

$K \leq_m A_{TM}$

$A_{TM} = \{ \langle M,y \rangle \mid M$ is a TM and M accepts $y \}$
$K = \{ \langle M \rangle \mid M$ is a TM and M accepts $\langle M \rangle \}$.

For any input x that encodes a Turing machine M (i.e., $x = \langle M \rangle$), define

- $f(x) = \langle M, x \rangle$.

NB. If x is some garbage (not a valid encoding), we assume x encodes a Turing machine M that rejects all inputs.

The function f is computable: checking the encoding + duplicating the input.

$X \in K$

$\Leftrightarrow x=\langle M \rangle$ and M accepts x

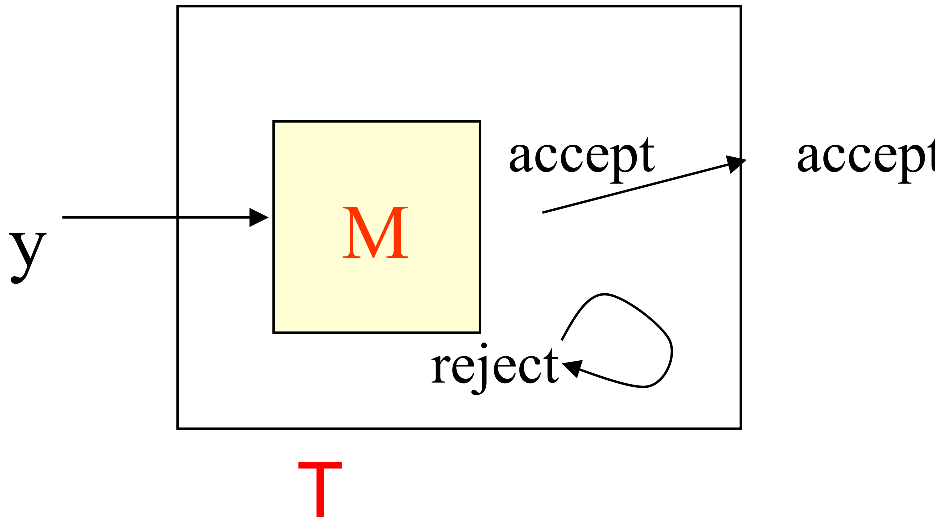
$\Leftrightarrow \langle M,x \rangle = f(x) \in A_{TM}$

Example 2

Halt_{TM} = { ⟨M,y⟩ | M is a TM and M halts on input y } is undecid

Claim: $A_{TM} \leq_m \text{Halt}_{TM}$.

For any input x = ⟨M,y⟩ that encodes a Turing machine M and an
input y, define T as the following TM & f(x) = ⟨T,y⟩.



B. If x is not in proper format, we assume M denotes a TM rejecting all inputs, and y
n empty string.

Example 2

Halt_{TM} = { ⟨M,y⟩ | M is a TM and M halts on input y } is undecid

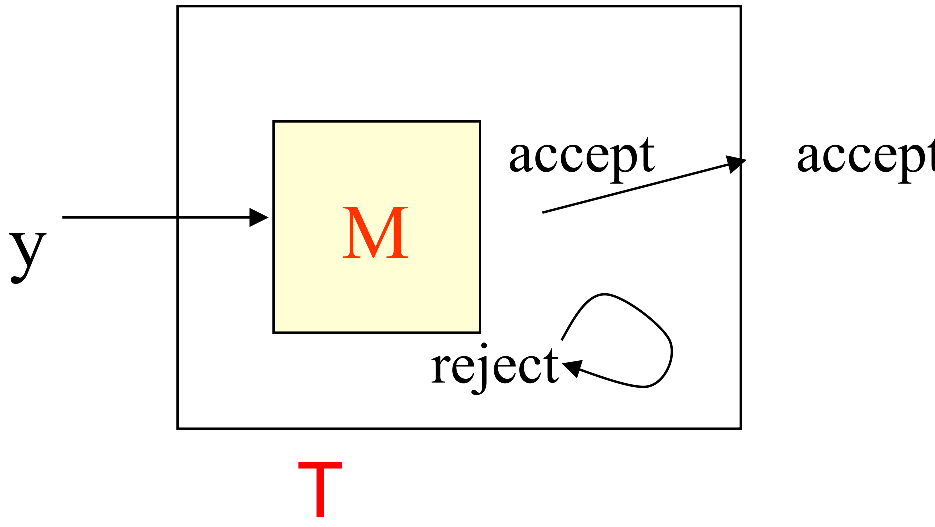
Claim: $A_{TM} \leq_m \text{Halt}_{TM}$.

For any input x = ⟨M,y⟩ that encodes a Turing machine M and an
input y, define T as the following TM & f(x) = ⟨T,y⟩.

Precisely,

- If M accepts y, T accepts.
- If M rejects y, T loops forever.

f(x) = = ⟨T,y⟩ is computable.



B. If x is not in proper format, we assume M denotes a TM rejecting all inputs, and y
n empty string.

Correctness

$x \in A_{TM}$

$\Rightarrow x = \langle M,y \rangle$ and M is a TM and M accepts y

$\Rightarrow$ T accepts y

$\Rightarrow$ T with y as input halts

$\Rightarrow \langle T,y \rangle = f(x) \in Halt_{TM}$

$x \notin A_{TM}$

$\Rightarrow x = \langle M,y \rangle$ and M is a TM and M doesn't accept y

$\Rightarrow$ M rejects y or M loops forever

$\Rightarrow$ T with y as input loops forever

$\Rightarrow \langle T,y \rangle = f(x) \notin Halt_{TM}$

Example 3

Let $E_{TM} = \{ \langle M \rangle \mid M$ is a Turing machine and $L(M)$ is empty $\}$.

Lemma. E_{TM} is undecidable.

Claim: $\sim K \leq_m E_{TM}$.

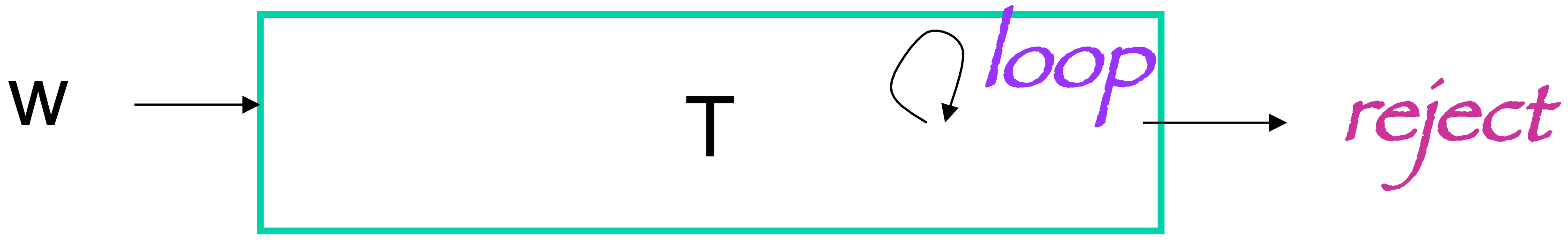
As $\sim K$ is undecidable, E_{TM} is undecidable.

Example

Claim: ~K $\leq_m$ E_{TM}.

Requirement for the reduction function f:

- For any x in ~K, we want f(x) to represent a Turing machine T accepting nothing.

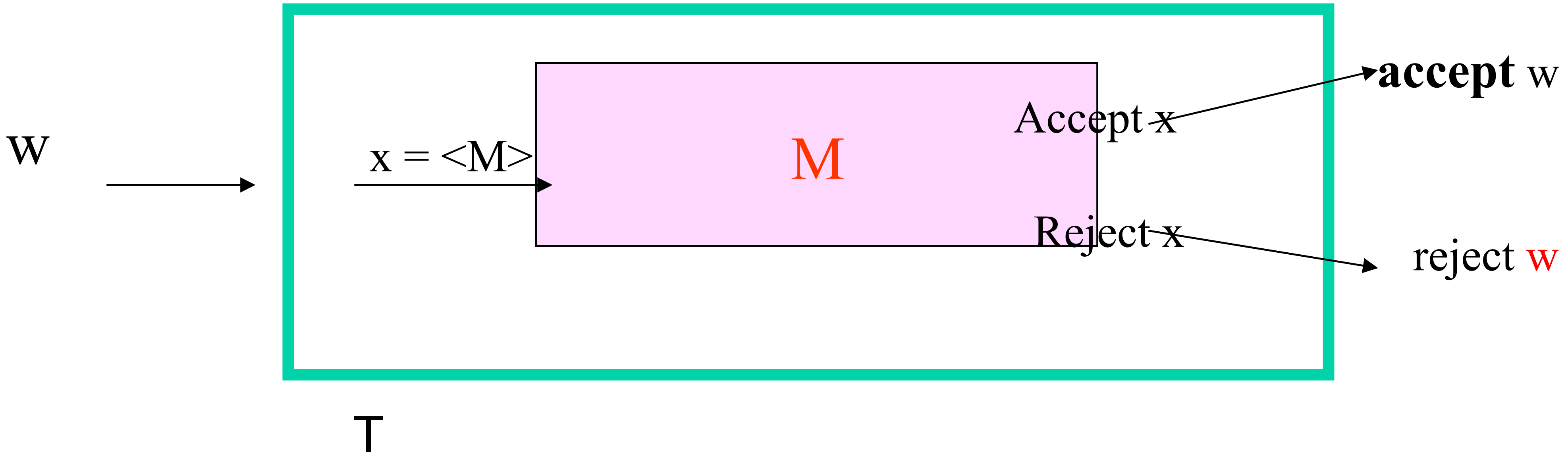


- For any x not in ~K (i.e., x in K), we want f(x) to represent a TM accepting some input.

What is f (x)

For any input x that encodes a Turing machine M (i.e., x = ⟨M⟩), define

- f(x) is the binary encoding of the following TM T, which ignores its input w, and simulates M on x. I.e., f(x) = ⟨T⟩.



Correctness

$x \in {\sim}K \Rightarrow X \notin K$

$\Rightarrow x = \langle M \rangle$ and M is a TM and M doesn't accept x

$\Rightarrow$ M rejects x or M loops forever

$\Rightarrow$ T does not accept w for all inputs w

$\Rightarrow$ L(T) is empty

$\Rightarrow f(x) = \langle T \rangle \in E_{TM}$

$x \notin {\sim}K \Rightarrow x \in K$

$\Rightarrow x = \langle M \rangle$ and M is a TM and M accepts x

$\Rightarrow$ T accepts w for all inputs w

$\Rightarrow$ L(T) is not empty

$\Rightarrow f(x) = \langle T \rangle \notin E_{TM}$

Non-trivial property

The following properties of Turing machines are all undecidable:

Given a Turing machine M,

- is L(M) = empty?
- is L(M) = Σ^*?
- does L(M) satisfy a non-trivial property ***P*** ?

NB. A property is non-trivial if it holds for some, but not all, Turing machines.

Rice Theorem

Theorem. Let **p** be any non-trivial property.
Then $\Pi_p = \{ \langle M \rangle \mid$ **p**$(L(M)) =$ **true**$\}$ is undecidable.

Proof

Claim: $K \leq_m \Pi_p$.

Then, by Corollary 2, as K is undecidable, Π_p is undecidable.

Consider a TM M_o rejecting all inputs. I.e., $L(M_o) = \emptyset$.

Without loss of generality, we assume that **p** $(L(M_o))$ = **p** $(\emptyset)$ = false.

Since **p** is non-trivial, there exists another TM T_o such that **p** $(L(T_o))$ = true.

Proof

Claim: $K \leq_m \Pi_p$.

Then, by Corollary 2, as K is undecidable, Π_p is undecidable.

Consider a TM M_o rejecting all inputs. I.e., $L(M_o) = \varnothing$.

Without loss of generality, we assume that **p** $(L(M_o)) =$ **p** $(\varnothing)$ = false.

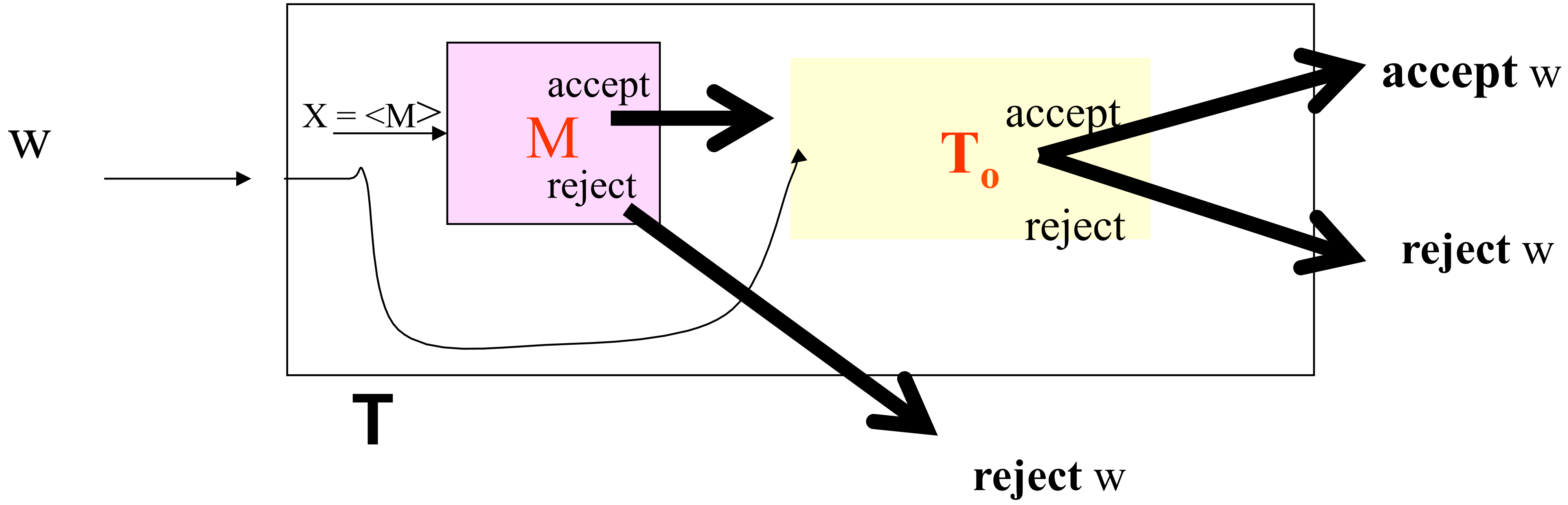
Since **p** is non-trivial, there exists another TM T_o such that **p**$(L(T_o))$ = true.

Reduction: What is f ? Is f computable? $x \in K \Leftrightarrow f(x) \in \Pi_p$?

Intuitively, if $x \in K$, f(x) encodes a TM whose language satisfies **p**; otherwise, f(x) encodes a TM whose language is exactly $\varnothing$ and does not satisfy **p**.

Definition of f (x)

For any input x that encodes a TM M (i.e., x = ⟨M⟩),
let f(x) = ⟨T⟩, where T is a Turing machine defined as follow



Correctness

$\in K$

$x = \langle M \rangle$ and M is a TM and M <u>accepts</u> x

T accepts an input w if and only if T_o accepts W

$L(T) = L(T_0)$

$p(L(T)) = p(L(T_0)) =$ true

$f(x) = \langle T \rangle \in \Pi_p$

Correctness

$\in$ K

$\Rightarrow$ x=⟨M⟩ and M is a TM and M <u>accepts</u> x

$\Rightarrow$ T accepts an input w if and only if T_0 accepts W

$\Rightarrow$ L(T) = L(T_0)

$\Rightarrow$ ***p*** (L(T)) = ***p*** (L(T_0)) = true

$\Rightarrow$ f(x) = ⟨T⟩ $\in \Pi_p$

x $\notin$ K

$\Rightarrow$ x=⟨M⟩ and M is a TM and M <u>does</u>

<u>accept</u> x

$\Rightarrow$ M rejects x or M loops forever

$\Rightarrow$ T does not accept any input w

$\Rightarrow$ L(T) = $\varnothing$

$\Rightarrow$ ***p*** (L(T)) = ***p*** ($\varnothing$) = false

3. Can you give a proof

*the case that **p**($\varnothing$) = true?*

Other properties of $\leq_m$

- If $A \leq_m B$ and B is recognizable, then A is recognizable.

NB. The proof is the same as that of Theorem 1.

- If $A \leq_m B$ and A is not recognizable, then B is not recognizable.

- If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.
- $A \leq_m A$

- Question: $A \leq_m \sim A$? No.

Post Correspondence Problem

- Input: a collection of dominos, each containing two strings
- Make a list of dominos (repetition permitted) so that the string composed of the upper strings matches that of the lower strings.

1	2	3	4
b	a	ca	abc
ca	ab	a	c

- Consider the list 21324;
- upper string = abcaaabc; lower string = abcaaabc
- Decision problem: Does such a list exist?

Proof: a reduction from A_{TM}.

Supplementary reading

- Sipser Chaper 5
- Hopcroft et al. : 9.3

List of undecidable problems:

http://en.wikipedia.org/wiki/List_of_undecidable_problems

Examples:

1. Given a set of 7 or more 3 × 3 matrices with integer entries (or in general, a finite set of n x n matrices),
   - determine whether they can be multiplied in some order, possibly with repetition, to yield the zero matrix.
2. Given 2 context free grammars,
   - determine if they generate the same set of strings. [Or one is the subset of the other.]