بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه هفتم: محاسبه‌پذیری و محاسبه‌ناپذیری (۳)

Theory of computation - 002 - S07 - non-computability (3)

Mapping Reducibility

Mapping Reducibility

Defn: Function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Mapping Reducibility

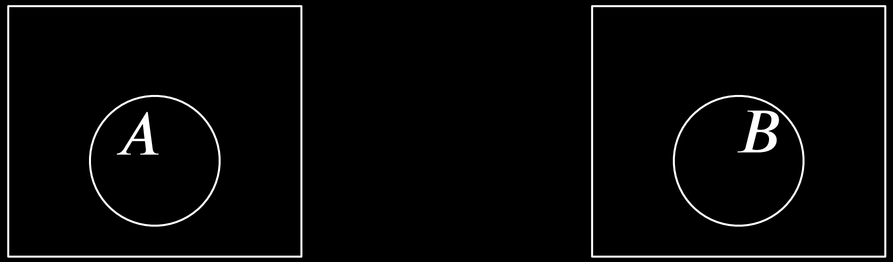
Defn: Function $f:\Sigma^* \to \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: A is mapping-reducible to B ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.

Mapping Reducibility

Defn: Function $f\colon \Sigma^* \to \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

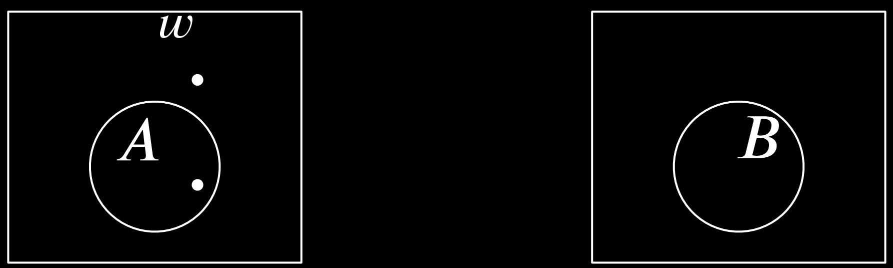
Defn: <u>A is mapping-reducible to B</u> ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Mapping Reducibility

Defn: Function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

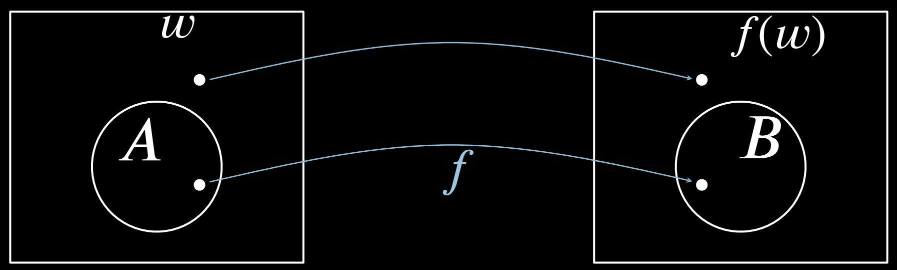
Defn: A is mapping-reducible to B ($A \leq_m B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Mapping Reducibility

Defn: Function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

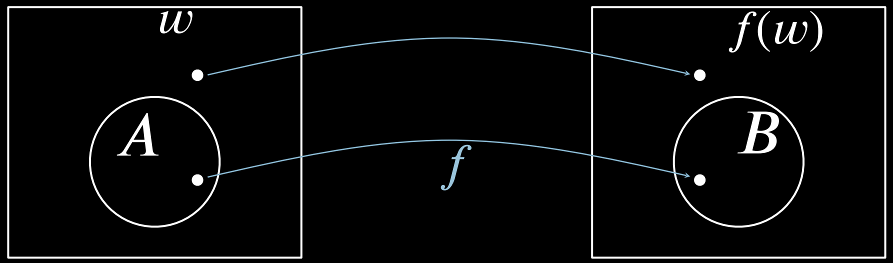
Defn: A is mapping-reducible to B ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Mapping Reducibility

Defn: Function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: A is mapping-reducible to B ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.

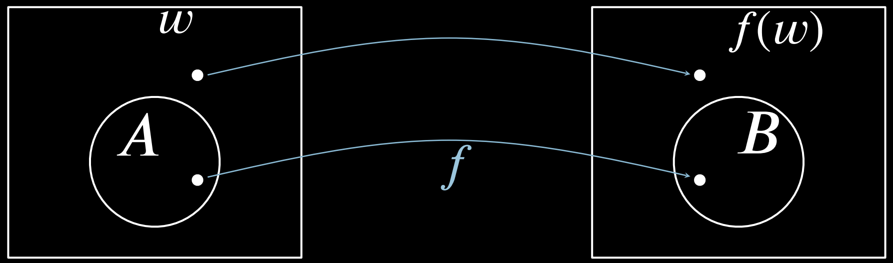


Example: ATM $\leq_{\mathrm{m}} \overline{E\text{TM}}$

Mapping Reducibility

Defn: Function $f : \Sigma^* \rightarrow \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: <u>A is mapping-reducible to B</u> ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



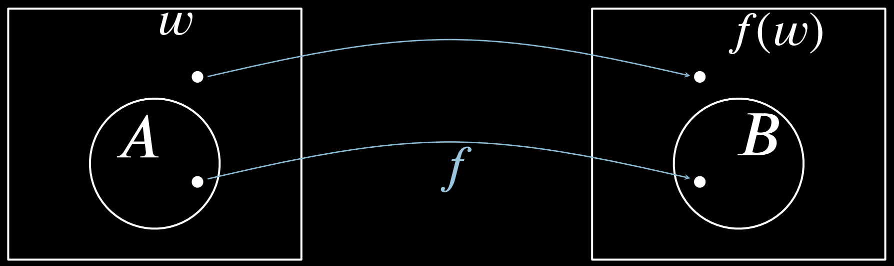
Example: $A_{\mathrm{TM}} \leq_{\mathrm{m}} \overline{E_{\mathrm{TM}}}$

The computable reduction function f is $f(\langle M, w\rangle) = \langle M_w \rangle$

Mapping Reducibility

Defn: Function $f : \Sigma^* \rightarrow \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: <u>A is mapping-reducible to B</u> ($A \leq_m B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Example: $A_{TM} \leq_m \overline{E_{TM}}$

The computable reduction function f is $f(\langle M, w\rangle) = \langle M_w \rangle$

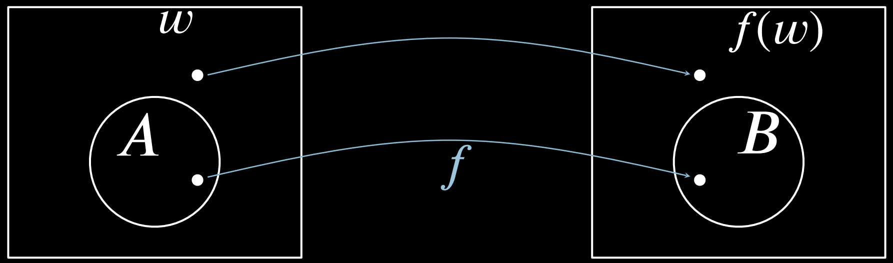
Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."

Mapping Reducibility

Defn: Function $f\colon \Sigma^* \to \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: <u>A is mapping-reducible to B</u> ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Example: A**TM** $\leq_{\mathrm{m}}$ $\overline{E\text{TM}}$

The computable reduction function f is $f(\langle M, w\rangle) = \langle M_w\rangle$

Recall TM $M_w =$ "On input x
1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."

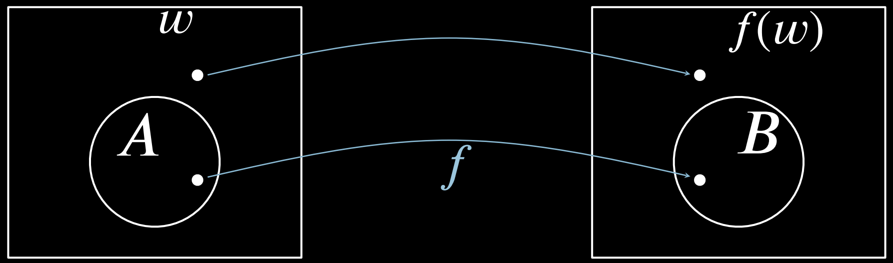
Because $\langle M, w\rangle \in A$TM iff $\langle M_w\rangle \in \overline{E\text{TM}}$

(M accepts w iff $L(\langle M_w\rangle) \neq \varnothing$)

Mapping Reducibility

Defn: Function $f : \Sigma^* \to \Sigma^*$ is <u>computable</u> if there is a TM F where F on input w halts with $f(w)$ on its tape, for all strings w.

Defn: <u>A is mapping-reducible to B</u> ($A \leq_{\mathrm{m}} B$) if there is a computable function f where $w \in A$ iff $f(w) \in B$.



Example: A**TM** $\leq_{\mathrm{m}}$ $\overline{E\text{TM}}$

The computable reduction function f is $f(\langle M, w\rangle) = \langle M_w \rangle$

Because $\langle M, w\rangle \in A$TM iff $\langle M_w \rangle \in \overline{E\text{TM}}$

(M accepts w iff $L(\langle M_w \rangle) \neq \varnothing$)

Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:

$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:

$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:

$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Theorem: If $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A

Proof: Same as above.

Mapping Reductions - properties

Theorem: If $A \leq_m B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:

$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_m B$ and A is undecidable then so is B

Theorem: If $A \leq_m B$ and B is T-recognizable then so is A

Proof: Same as above.

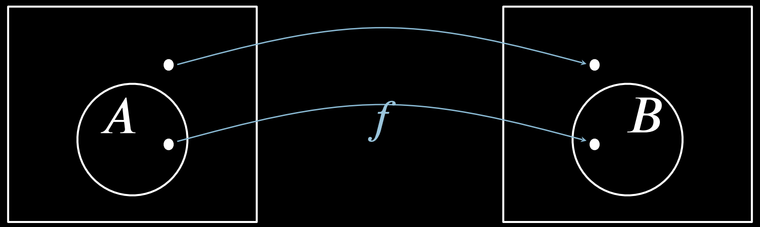
Corollary: If $A \leq_m B$ and A is T-unrecognizable then so is B

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:



$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Theorem: If $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A

Proof: Same as above.

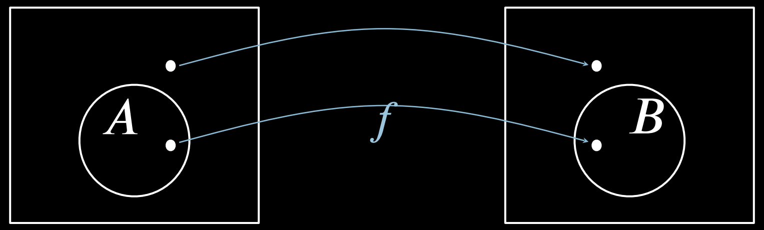
Corollary: If $A \leq_{\mathrm{m}} B$ and A is T-unrecognizable then so is B

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:



$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Theorem: If $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A

Proof: Same as above.

Corollary: If $A \leq_{\mathrm{m}} B$ and A is T-unrecognizable then so is B

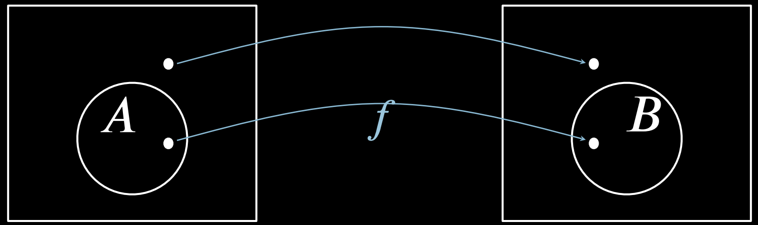
Check-in 9.2

Mapping Reductions - properties

Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable then so is A

Proof: Say TM R decides B.

Construct TM S deciding A:



$S =$ "On input w

1. Compute $f(w)$
2. Run R on $f(w)$ to test if $f(w) \in B$
3. If R halts then output same result."

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable then so is B

Theorem: If $A \leq_{\mathrm{m}} B$ and B is T-recognizable then so is A

Proof: Same as above.

Corollary: If $A \leq_{\mathrm{m}} B$ and A is T-unrecognizable then so is B

Check-in 9.2

Suppose $A \leq_{\mathrm{m}} B$.

What can we conclude?

Check all that apply.

(a) $B \leq_{\mathrm{m}} A$

(b) $\overline{A} \leq_{\mathrm{m}} \overline{B}$

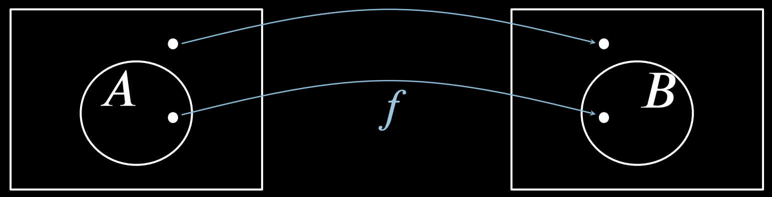
(c) None of the above

Check-in 9.2

Mapping vs General Reducibility

Mapping vs General Reducibility

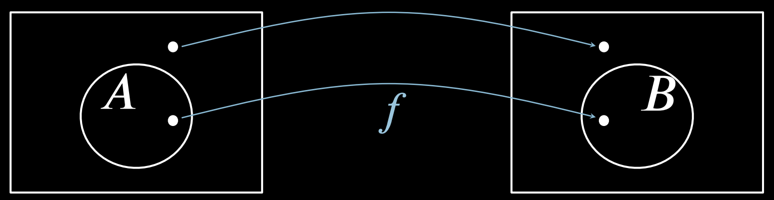
Mapping Reducibility of A to B: Translate A-questions to B-questions.



Mapping vs General Reducibility

Mapping Reducibility of A to B: Translate A-questions to B-questions.

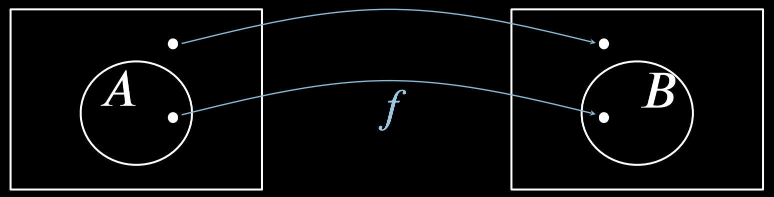
- A special type of reducibility
- Useful to prove T-unrecognizability



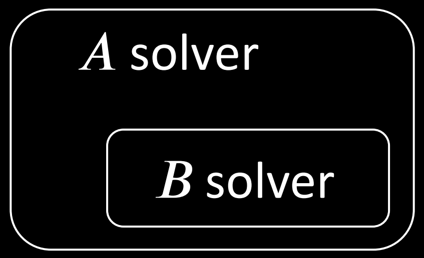
Mapping vs General Reducibility

Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



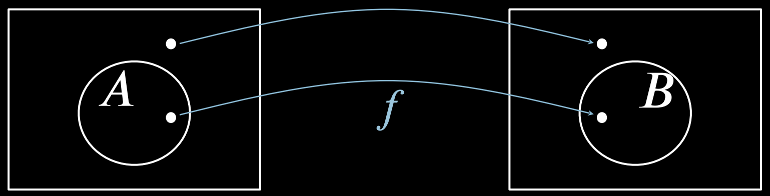
(General) Reducibility of A to B: Use B solver to solve A.



Mapping vs General Reducibility

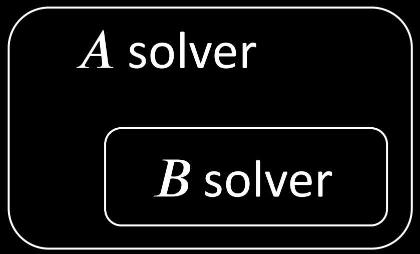
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

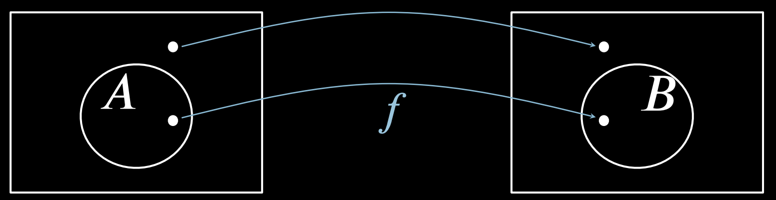
- May be conceptually simpler
- Useful to prove undecidability



Mapping vs General Reducibility

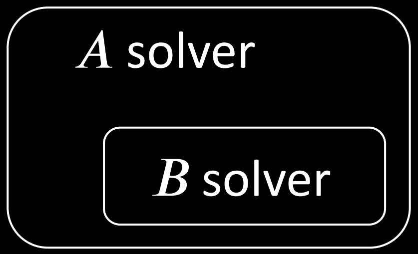
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

- May be conceptually simpler
- Useful to prove undecidability

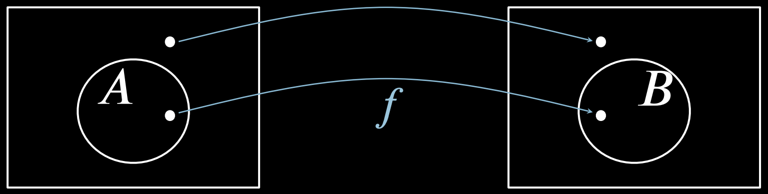


Noteworthy difference:

Mapping vs General Reducibility

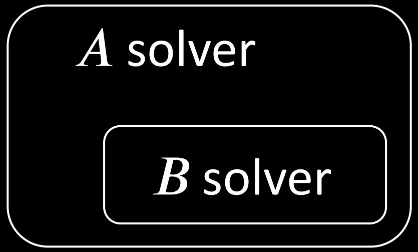
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

- May be conceptually simpler
- Useful to prove undecidability



Noteworthy difference:

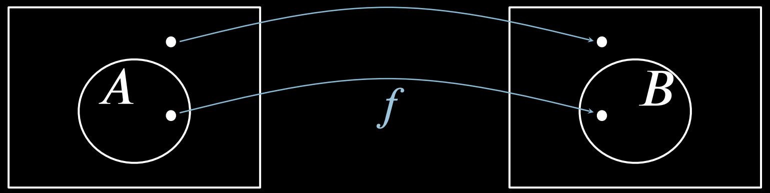
- A is reducible to $\overline{A}$
- A may not be mapping reducible to $\overline{A}$.

For example $\overline{A\mathrm{TM}} \nleq_{\mathrm{m}} A\mathrm{TM}$

Mapping vs General Reducibility

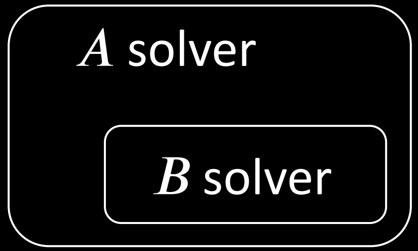
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

- May be conceptually simpler
- Useful to prove undecidability



Noteworthy difference:

- A is reducible to $\overline{A}$
- A may not be mapping reducible to $\overline{A}$.

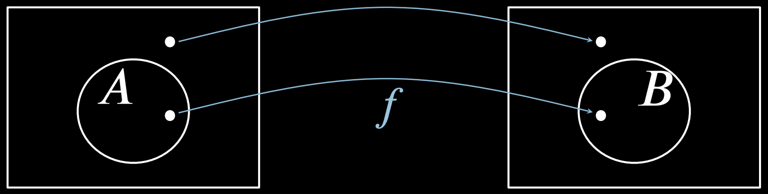
For example $\overline{A\text{TM}} \not\leq_{\text{m}} A\text{TM}$

Check-in 9.3

Mapping vs General Reducibility

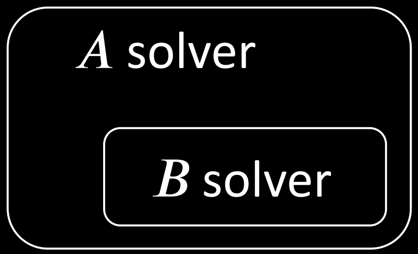
Mapping Reducibility of A to B: Translate A-questions to B-questions.

- A special type of reducibility
- Useful to prove T-unrecognizability



(General) Reducibility of A to B: Use B solver to solve A.

- May be conceptually simpler
- Useful to prove undecidability



Noteworthy difference:

- A is reducible to $\overline{A}$
- A may not be mapping reducible to $\overline{A}$.

For example $\overline{A\text{TM}} \not\leq_{\text{m}} A\text{TM}$

Check-in 9.3

We showed that if $A \leq_{\text{m}} B$ and B is T-recognizable then so is A.

Is the same true if we use general reducibility instead of mapping reducibility?

(a) Yes

(b) No

Check-in 9.3

Reducibility – Templates

Reducibility – Templates

To prove B is undecidable:

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)
- Template: Assume TM R decides B.

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.

To prove B is T-unrecognizable:

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is A_{TM})
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.

To prove B is T-unrecognizable:

- Show T-unrecognizable A is mapping reducible to B. (often A is $\overline{A_{\mathrm{TM}}}$)

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.

To prove B is T-unrecognizable:

- Show T-unrecognizable A is mapping reducible to B. (often A is $\overline{A\text{TM}}$)
- Template: give reduction function f.

Reducibility – Templates

To prove B is undecidable:

- Show undecidable A is reducible to B. (often A is ATM)
- Template: Assume TM R decides B.
  Construct TM S deciding A. Contradiction.

To prove B is T-unrecognizable:

- Show T-unrecognizable A is mapping reducible to B. (often A is $\overline{A\text{TM}}$)
- Template: give reduction function f.

ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid M$ is a TM and $L(M) = \emptyset\}$

ETM is T-unrecognizable

Recall $E\text{TM} = \{\langle M\rangle \mid M$ is a TM and $L(M) = \varnothing\}$

Theorem: E**TM** is T-unrecognizable

ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid M$ is a TM and $L(M) = \emptyset\ \}$

Theorem: E**TM** is T-unrecognizable

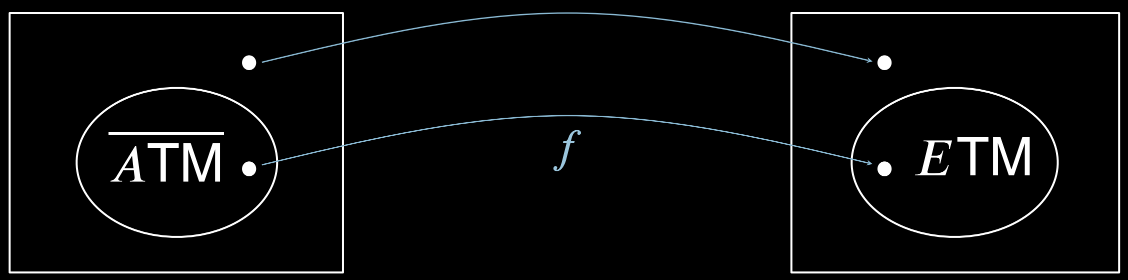
Proof: Show $\overline{A\text{TM}} \leq_{\mathrm{m}} E\text{TM}$

ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid M$ is a TM and $L(M) = \varnothing\}$

Theorem: E**TM** is T-unrecognizable

Proof: Show $\overline{A\text{TM}} \leq_{\text{m}} E$TM



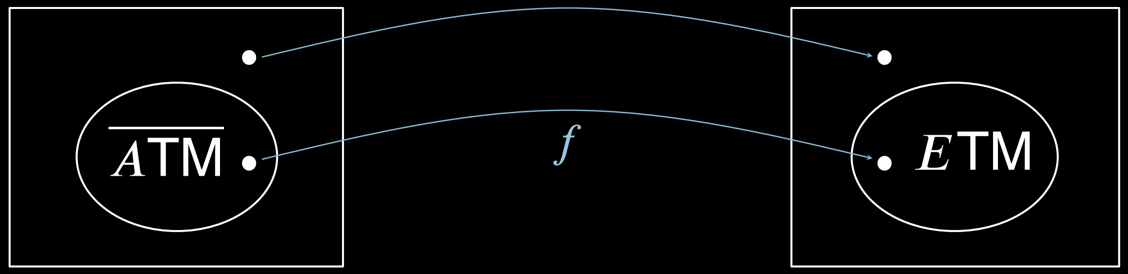
ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid M$ is a TM and $L(M) = \varnothing\}$

Theorem: E**TM** is T-unrecognizable

Proof: Show $\overline{A\text{TM}} \leq_{\text{m}} E\text{TM}$

Reduction function: $f(\langle M, w\rangle) = \langle M_w \rangle$



ETM is T-unrecognizable

Recall ETM $= \{\langle M\rangle \mid M$ is a TM and $L(M) = \varnothing\}$

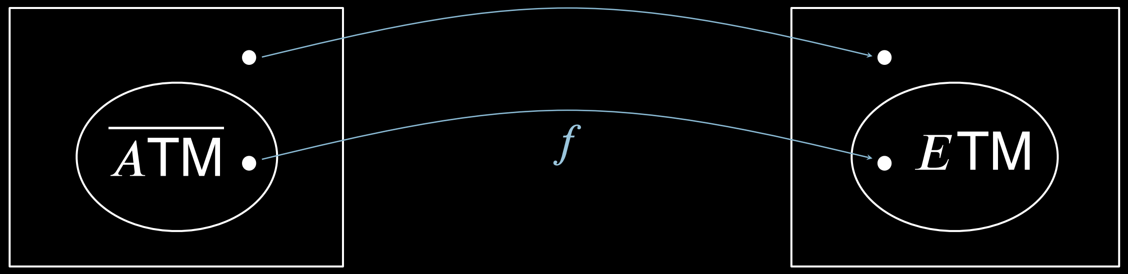
Theorem: E**TM** is T-unrecognizable

Proof: Show $\overline{A\text{TM}} \leq_{\mathrm{m}} E\text{TM}$

Reduction function: $f(\langle M, w\rangle) = \langle M_w\rangle$

Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."



ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid$ M is a TM and $L(M) = \varnothing$ $\}$

Theorem: E**TM** is T-unrecognizable

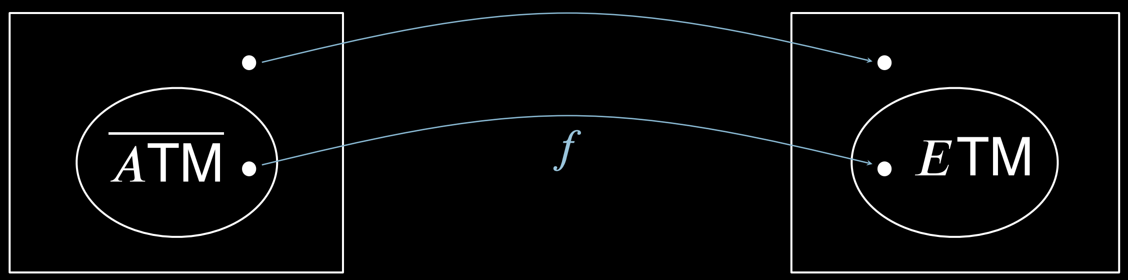
Proof: Show $\overline{A\text{TM}} \leq_m E$TM

Reduction function: $f(\langle M, w \rangle) = \langle M_w \rangle$

Explanation: $\langle M, w \rangle \in \overline{A\text{TM}}$ iff $\langle M_w \rangle \in E$TM

Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."



ETM is T-unrecognizable

Recall ETM $= \{\langle M \rangle \mid M$ is a TM and $L(M) = \varnothing\ \}$

Theorem: E**TM** is T-unrecognizable

Proof: Show $\overline{A\text{TM}} \leq_{\mathrm{m}} E$TM

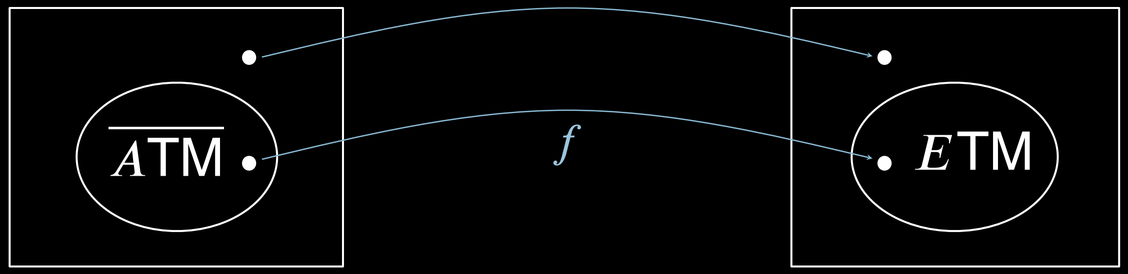
Reduction function: $f(\langle M, w \rangle) = \langle M_w \rangle$

Explanation: $\langle M, w \rangle \in \overline{A\text{TM}}$ iff $\langle M_w \rangle \in E$TM

M rejects w iff $L(\langle M_w \rangle) = \varnothing$

Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."



ETM is T-unrecognizable

Recall ETM $= \{\langle M\rangle \mid M$ is a TM and $L(M) = \varnothing\ \}$

Theorem: E**TM** is T-unrecognizable

Proof: Show $\overline{A\text{TM}} \leq_\mathrm{m} E$TM

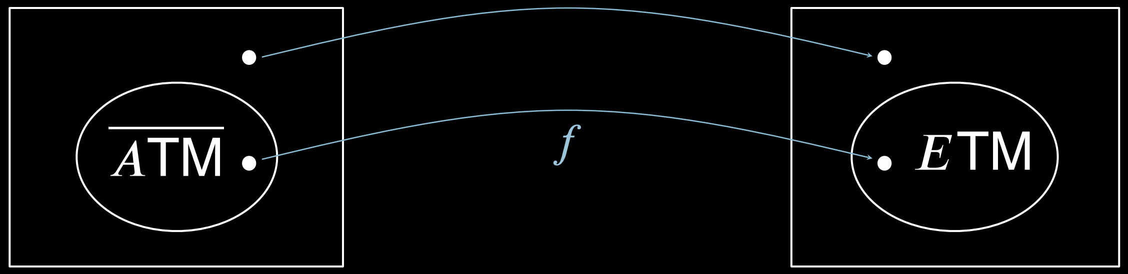
Recall TM $M_w =$ "On input x

1. If $x \neq w$, *reject*.
2. else run M on w
3. *Accept* if M accepts."

Reduction function: $f\big(\langle M, w\rangle\big) = \langle M_w\rangle$

Explanation: $\langle M, w\rangle \in \overline{A\text{TM}}$ iff $\langle M_w\rangle \in E$TM

M rejects w iff $L\big(\langle M_w\rangle\big) = \varnothing$



EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\ \}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

EQTM and $\overline{EQ}$TM are T-unrecognizable

$EQ\text{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: Both EQTM and $\overline{EQ}$TM are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_m EQ\text{TM}$

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\ \}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\mathrm{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\mathrm{m}} \overline{EQ\text{TM}}$

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\text{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\text{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \le_m EQ\text{TM}$

(2) $\overline{A\text{TM}} \le_m \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.

EQTM and $\overline{EQ}$TM are T-unrecognizable

$EQ\text{TM} = \left\{ \langle M_1,\ M_2 \rangle \middle| \ M_1 \text{ and } M_2 \text{ are TMs and } L\left(M_1\right) = L(M_2) \ \right\}$

Theorem: Both EQTM and $\overline{EQ}$TM are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\text{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\text{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \le_{\text{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \le_{\text{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\ \}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\mathrm{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\mathrm{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

(1) Here we give f which maps $\overline{A\text{TM}}$ problems (of the form $\langle M, w\rangle$) to EQTM problems (of the form $\langle T_1, T_2\rangle$).

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\ \}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\text{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\text{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

(1) Here we give f which maps $\overline{A\text{TM}}$ problems (of the form $\langle M, w\rangle$) to EQTM problems (of the form $\langle T_1, T_2\rangle$).

$f(\langle M, w\rangle) = \langle T_w,\ T_{\text{reject}}\rangle$ $\qquad$ T_{reject} is a TM that always rejects.

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \le_m EQ\text{TM}$

(2) $\overline{A\text{TM}} \le_m \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

(1) Here we give f which maps $\overline{A\text{TM}}$ problems (of the form $\langle M, w\rangle$) to EQTM problems (of the form $\langle T_1, T_2\rangle$).

$f(\langle M, w\rangle) = \langle T_w, T_{\text{reject}}\rangle$ T_{reject} is a TM that always rejects.

(2) Similarly $f(\langle M, w\rangle) = \langle T_w, T_{\text{accept}}\rangle$ T_{accept} always accepts.

EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

$EQ\text{TM} = \{\langle M_1,\ M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\ \}$

Theorem: Both EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable

Proof: (1) $\overline{A\text{TM}} \leq_{\text{m}} EQ\text{TM}$

(2) $\overline{A\text{TM}} \leq_{\text{m}} \overline{EQ\text{TM}}$

For any w let $T_w =$ "On input x

1. Ignore x.
2. Simulate M on w."

T_w acts on all inputs the way M acts on w.

(1) Here we give f which maps $\overline{A\text{TM}}$ problems (of the form $\langle M, w\rangle$) to EQTM problems (of the form $\langle T_1, T_2\rangle$).

$f(\langle M, w\rangle) = \langle T_w,\ T_{\text{reject}}\rangle$ $\qquad T_{\text{reject}}$ is a TM that always rejects.

(2) Similarly $f(\langle M, w\rangle) = \langle T_w,\ T_{\text{accept}}\rangle$ $\qquad T_{\text{accept}}$ always accepts.

Reducibility terminology

Why do we use the term “reduce”?

Reducibility terminology

Why do we use the term “reduce”?

When we reduce A to B, we show how to solve A by using B
and conclude that A is no harder than B. (suggests the $\leq_{\mathrm{m}}$ notation)

Reducibility terminology

Why do we use the term "reduce"?

When we reduce A to B, we show how to solve A by using B
and conclude that A is no harder than B. (suggests the $\leq_{\mathrm{m}}$ notation)

Possibility 1: We bring A's difficulty down to B's difficulty.

Reducibility terminology

Why do we use the term "reduce"?

When we reduce A to B, we show how to solve A by using B
and conclude that A is no harder than B. (suggests the $\leq_{\mathrm{m}}$ notation)

Possibility 1: We bring A's difficulty down to B's difficulty.

Possibility 2: We bring B's difficulty up to A's difficulty.

Reducibility terminology

Why do we use the term “reduce”?

When we reduce A to B, we show how to solve A by using B
and conclude that A is no harder than B. (suggests the $\leq_{\mathrm{m}}$ notation)

Possibility 1: We bring A’s difficulty down to B’s difficulty.

Possibility 2: We bring B’s difficulty up to A’s difficulty.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \dots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a match is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \dots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \dots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \quad \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a match is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \cdots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Problem: Given P, is there a match?

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a match is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \ldots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Problem: Given P, is there a match?
Theorem: Undecidable!

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad, \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \quad \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \left\{ \langle P \rangle \,\middle|\, P \text{ has a match } \right\}$

Proof: Show ATM is reducible to PCP.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \ldots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a match is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{\langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \quad \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match: $\begin{array}{|lll} \text{a} & \text{b} & \\ \text{a} & \text{b} & \text{a} \end{array}$

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

a b | a a |

a b a | a b a |

Problem: Given P, is there a match?

Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \cdots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

| a b | a a | b a |
| a b a | a b a | a a |

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{\langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \cdots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

a b | a a | b a | a a |

a b a | a b a | a a | a b a |

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \cdots \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

| a b | a a | b a | a a | a b a b |
| a b a | a b a | a a | a b a | b |

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \cdots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \quad \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} \; = \; b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \quad \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \quad \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

| a b | a a | b a | a a | a b a b |
| a b a | a b a | a a | a b a | b |

✓

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match } \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \quad \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \quad \ldots \quad , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed) where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \quad \ldots \quad \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

| a b | a a | b a | a a | a b a b |
| a b a | a b a | a a | a b a | b |

✓

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{\langle P \rangle \mid P \text{ has a match }\}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Check-in 10.1

Post Correspondence Problem

Given a collection of pairs of strings as dominoes:

$$P = \left\{ \begin{bmatrix} t_1 \\ b_1 \end{bmatrix}, \begin{bmatrix} t_2 \\ b_2 \end{bmatrix}, \ \ldots \ , \begin{bmatrix} t_k \\ b_k \end{bmatrix} \right\}$$

a <u>match</u> is a finite sequence of dominos in P (repeats allowed)
where the concatenation of the t's = the concatenation of the b's.

Match = $\begin{bmatrix} t_{i_1} \\ b_{i_1} \end{bmatrix} \begin{bmatrix} t_{i_2} \\ b_{i_2} \end{bmatrix} \ \ldots \ \begin{bmatrix} t_{i_l} \\ b_{i_l} \end{bmatrix}$ where

$$t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$$

Example: $P = \left\{ \begin{bmatrix} \text{ab} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{aa} \\ \text{aba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{aa} \end{bmatrix}, \begin{bmatrix} \text{abab} \\ \text{b} \end{bmatrix} \right\}$

Match:

a b	a a	b a	a a	a b a b

a b a	a b a	a a	a b a	b

✓

Check-in 10.1

Let

$$P_1 = \left\{ \begin{bmatrix} \text{aa} \\ \text{aaba} \end{bmatrix}, \begin{bmatrix} \text{ba} \\ \text{ab} \end{bmatrix}, \begin{bmatrix} \text{ab} \\ \text{ba} \end{bmatrix} \right\}$$

Does P_1 have a match?

(a) Yes.

(b) No.

Problem: Given P, is there a match?
Theorem: Undecidable!

Let $PCP = \{ \langle P \rangle \mid P \text{ has a match} \}$

Proof: Show ATM is reducible to PCP.

First: the Computation History Method.

Check-in 10.1

TM Configurations

TM Configurations

Defn: A configuration of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.

TM Configurations

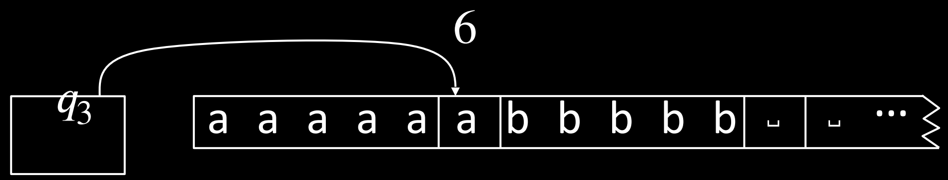
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



TM Configurations

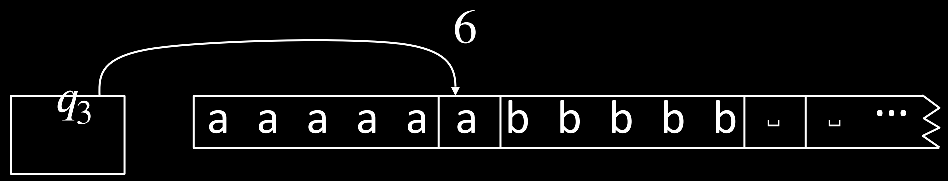
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



Configuration: $(q_3,\ 6,$aaaaaabbbbb$)$

TM Configurations

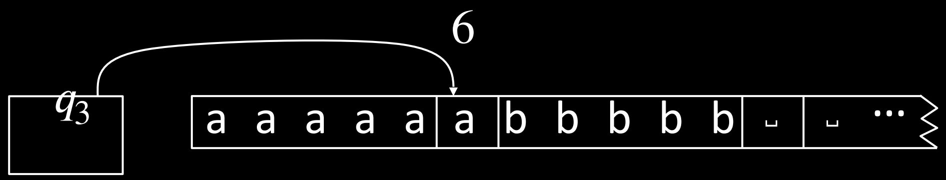
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



Configuration: $(q_3,\ 6,\text{aaaaaabbbbb})$

Encode configuration (q, p, t) as the string t_1qt_2 where
$t = t_1t_2$ and the head position is on the first symbol of t_2.

TM Configurations

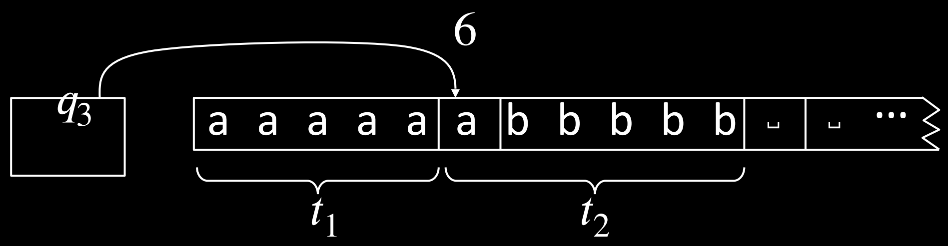
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



Configuration: $(q_3,\ 6,$aaaaaabbbbb$)$

Encode configuration (q, p, t) as the string $t_1 q t_2$ where

$t = t_1 t_2$ and the head position is on the first symbol of t_2.

TM Configurations

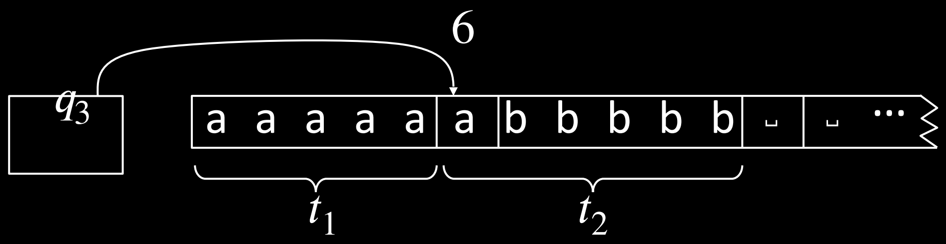
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



Configuration: $(q_3,\ 6,$aaaaaabbbbb$)$

Encoding as a string: aaaaa$q_3$abbbbb

Encode configuration (q, p, t) as the string $t_1 q t_2$ where

$t = t_1 t_2$ and the head position is on the first symbol of t_2.

TM Configurations

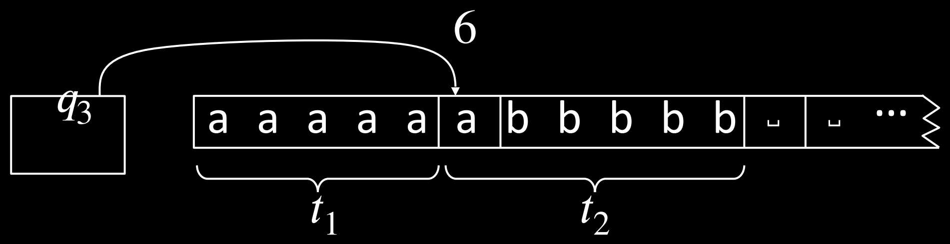
Defn: A <u>configuration</u> of a TM is a triple (q, p, t) where

q = the state,

p = the head position,

t = tape contents

representing a snapshot of the TM at a point in time.



Configuration: $(q_3,\ 6,$aaaaaabbbbb$)$

Encoding as a string: aaaaa$q_3$abbbbb

Encode configuration (q, p, t) as the string $t_1 q t_2$ where

$t = t_1 t_2$ and the head position is on the first symbol of t_2.

TM Computation Histories

Defn: An <u>(accepting) computation history</u> for TM M on input w is a sequence of configurations $C_1,\ C_2,\ \ldots, C$accept that M enters until it accepts.

TM Computation Histories

Defn: An (accepting) computation history for TM M on input w is a sequence of configurations $C_1,\ C_2,\ \ldots, C_{\text{accept}}$ that M enters until it accepts.

Encode a computation history
$C_1,\ C_2,\ \ldots, C_{\text{accept}}$ as the string
$C_1 \# C_2 \# \cdots \# C_{\text{accept}}$ where each
configuration C_i is encoded as a string.

TM Computation Histories

Defn: An <u>(accepting) computation history</u> for TM M on input w is a sequence of configurations $C_1, C_2, \ldots, C_{\text{accept}}$ that M enters until it accepts.

Encode a computation history $C_1, C_2, \ldots, C_{\text{accept}}$ as the string $C_1 \# C_2 \# \cdots \# C_{\text{accept}}$ where each configuration C_i is encoded as a string.

A computation history for M on $w = w_1 w_2 \cdots w_n$.

Here say $\delta(q_0, w_1) = (q_7, \text{a}, \text{R})$ and $\delta(q_7, w_2) = (q_8, \text{c}, R)$.

$q_0 w_1 w_2 \cdots w_n \quad \# \quad \text{a} q_7 w_2 \cdots w_n \quad \# \quad \text{ac} q_8 w_3 \cdots w_n \quad \# \quad \cdots \quad \# \cdots q_{\text{accept}} \cdots$

TM Computation Histories

Defn: An <u>(accepting) computation history</u> for TM M on input w is a sequence of configurations $C_1,\ C_2,\ \ldots, C_{\text{accept}}$ that M enters until it accepts.

Encode a computation history $C_1,\ C_2,\ \ldots, C_{\text{accept}}$ as the string $C_1 \# C_2 \# \cdots \# C_{\text{accept}}$ where each configuration C_i is encoded as a string.

A computation history for M on $w = w_1 w_2 \cdots w_n$.
Here say $\delta\left(q_0, w_1\right) = (q_7, \text{a}, \text{R})$ and $\delta\left(q_7, w_2\right) = (q_8, \text{c}, R)$.

$$\overbrace{q_0 w_1 w_2 \cdots w_n}^{C_1} \ \# \ \overbrace{\text{a} q_7 w_2 \cdots w_n}^{C_2} \ \# \ \overbrace{\text{ac} q_8 w_3 \cdots w_n}^{C_3} \ \# \ \cdots \ \# \ \overbrace{\cdots\ q_{\text{accept}} \cdots}^{C_{\text{accept}}}$$

TM Computation Histories

Defn: An <u>(accepting) computation history</u> for TM M on input w is a sequence of configurations $C_1, C_2, \ldots, C$accept that M enters until it accepts.

Encode a computation history $C_1, C_2, \ldots, C$accept as the string $C_1 \# C_2 \# \cdots \# C$accept where each configuration C_i is encoded as a string.

A computation history for M on $w = w_1 w_2 \cdots w_n$.
Here say $\delta(q_0, w_1) = (q_7, \text{a}, \text{R})$
and $\delta(q_7, w_2) = (q_8, \text{c}, R)$.

$$\underbrace{q_0 w_1 w_2 \cdots w_n}_{C_1} \;\#\; \underbrace{\text{a} q_7 w_2 \cdots w_n}_{C_2} \;\#\; \underbrace{\text{ac} q_8 w_3 \cdots w_n}_{C_3} \;\#\; \cdots \;\#\; \underbrace{\cdots q_{\text{accept}} \cdots}_{C_{\text{accept}}}$$

Quick review of today

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.
3. ETM is undecidable.

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.
3. ETM is undecidable.
4. ETM is T-unrecognizable.

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.
3. ETM is undecidable.
4. ETM is T-unrecognizable.
5. EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable.

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.
3. ETM is undecidable.
4. ETM is T-unrecognizable.
5. EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable.
6. PCP is undecidable

Quick review of today

1. Introduced The Reducibility Method to prove undecidability and T-unrecognizability.
2. Defined mapping reducibility as a type of reducibility.
3. ETM is undecidable.
4. ETM is T-unrecognizable.
5. EQTM and $\overline{EQ\text{TM}}$ are T-unrecognizable.
6. PCP is undecidable