بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه هشتم: خودتولیدکنندگی

Theory of computation - 002 - S08 - self-reproducibility

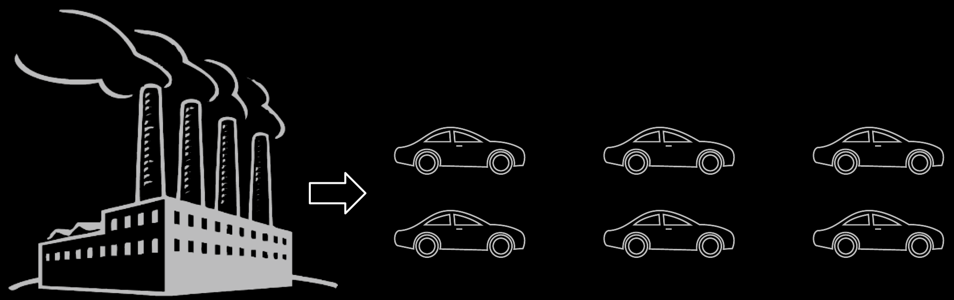
Self-reproduction Paradox

Self-reproduction Paradox

Suppose a Factory makes Cars

Self-reproduction Paradox

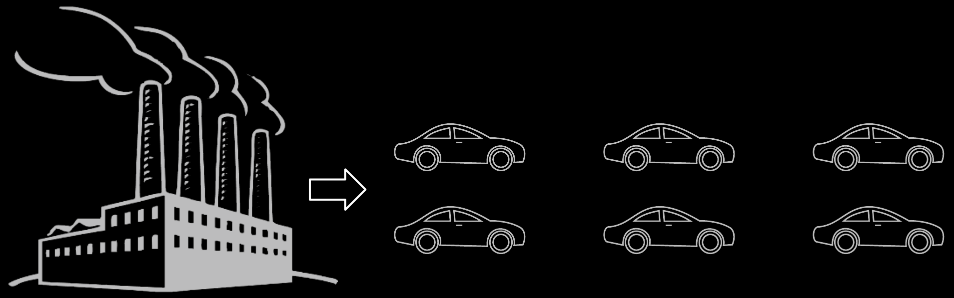
Suppose a Factory makes Cars



Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

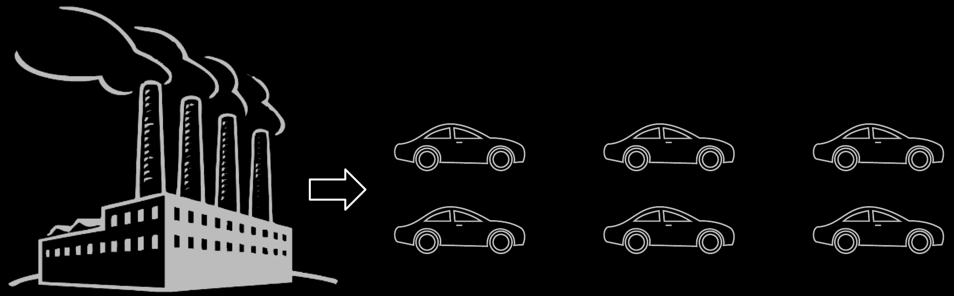


Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

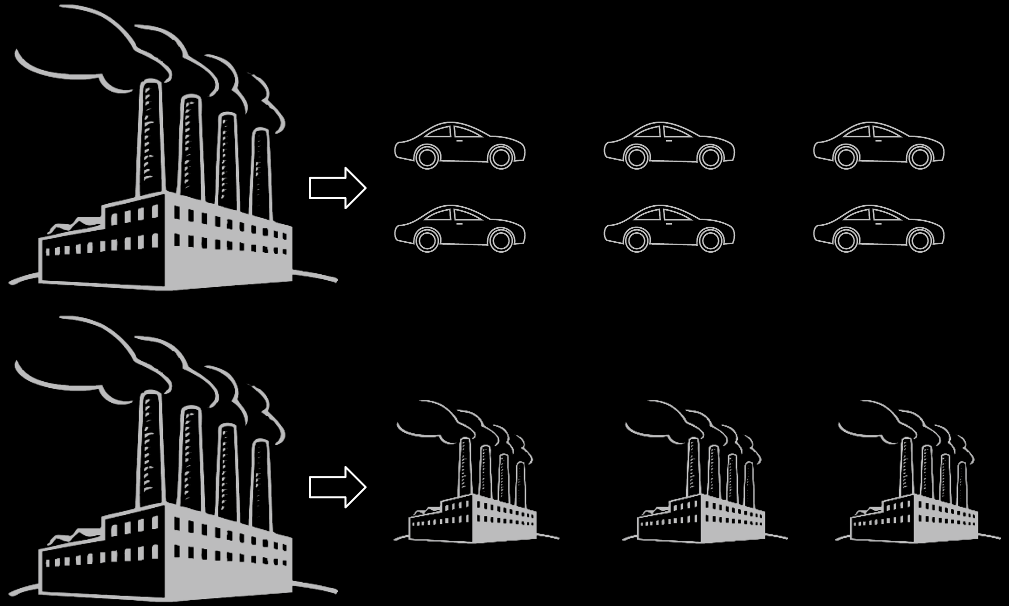


Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?



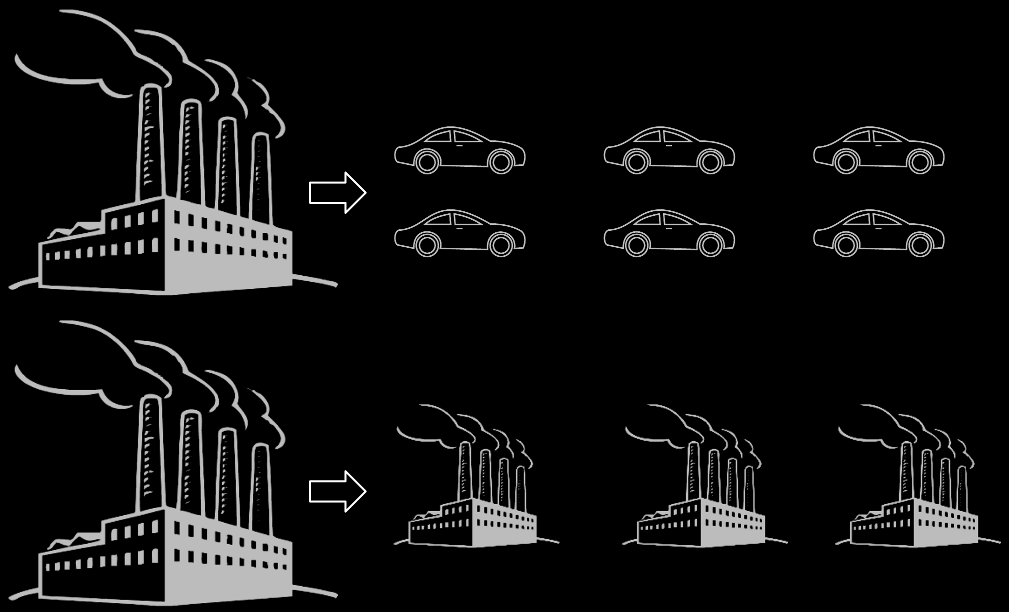
Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?



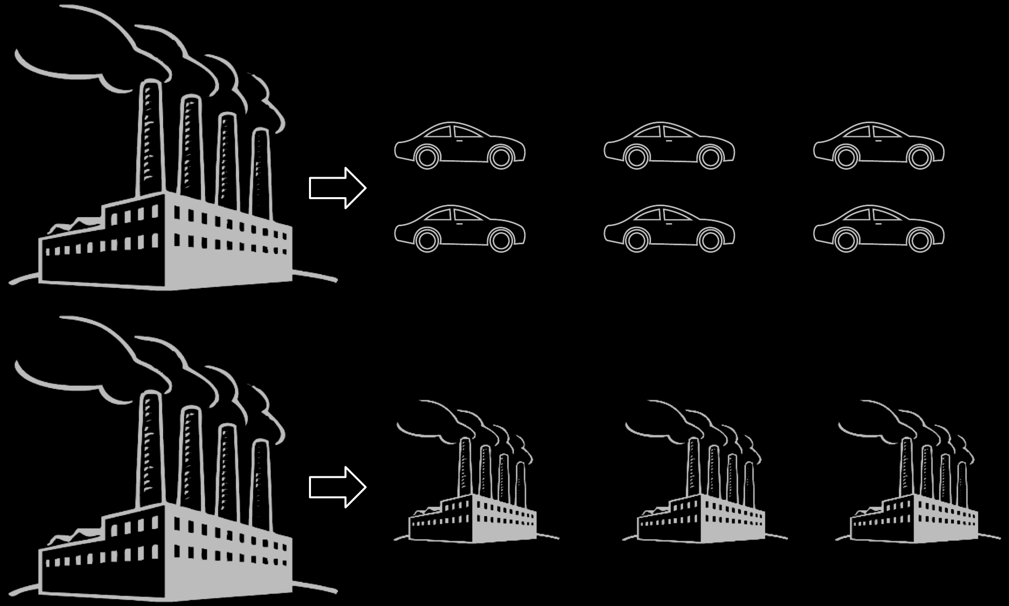
Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine



Self-reproduction Paradox

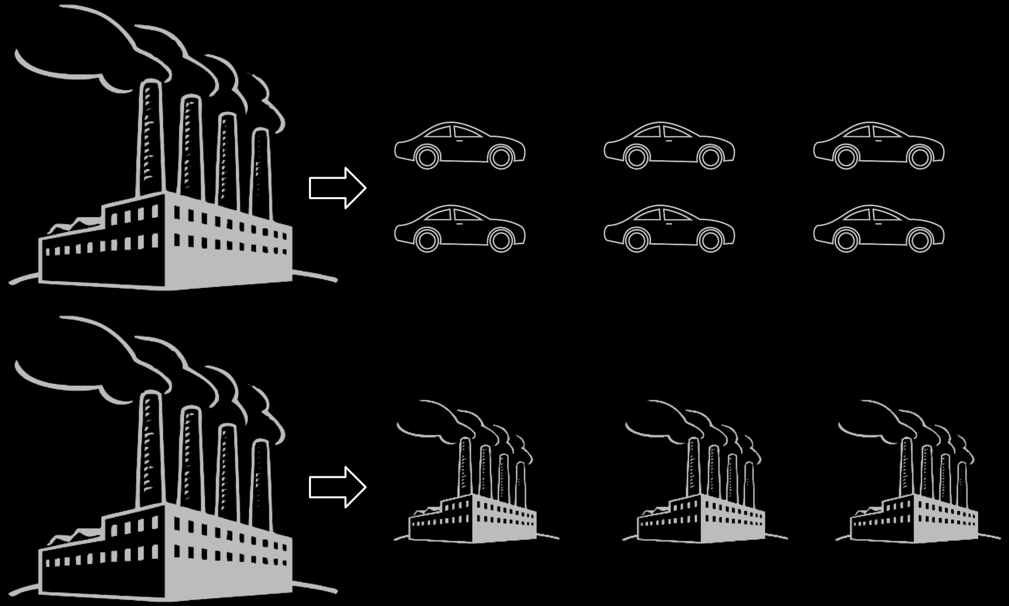
Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine

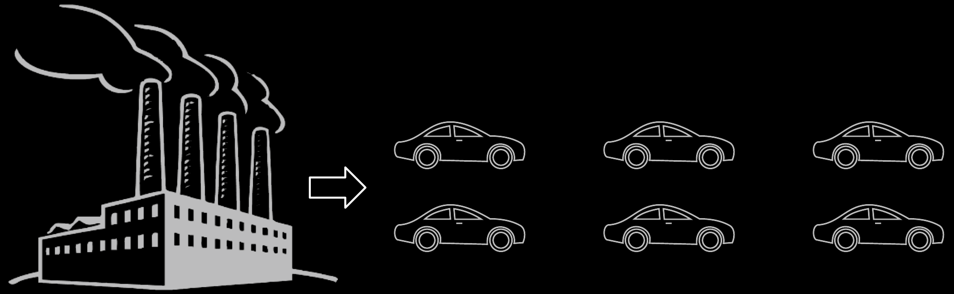
But, living things self-reproduce



Self-reproduction Paradox

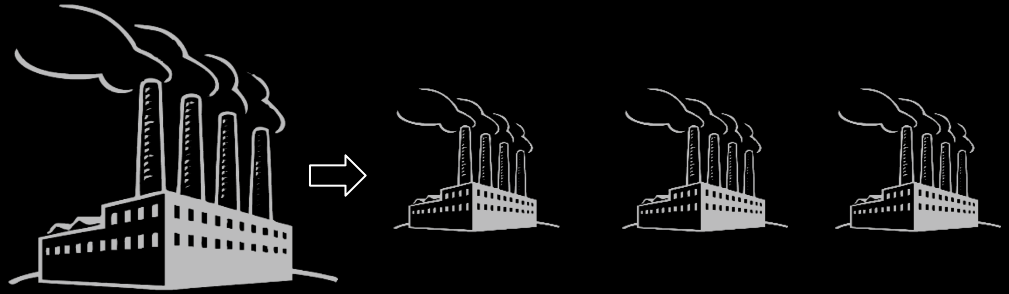
Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)



Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine



But, living things self-reproduce

How to resolve this paradox?

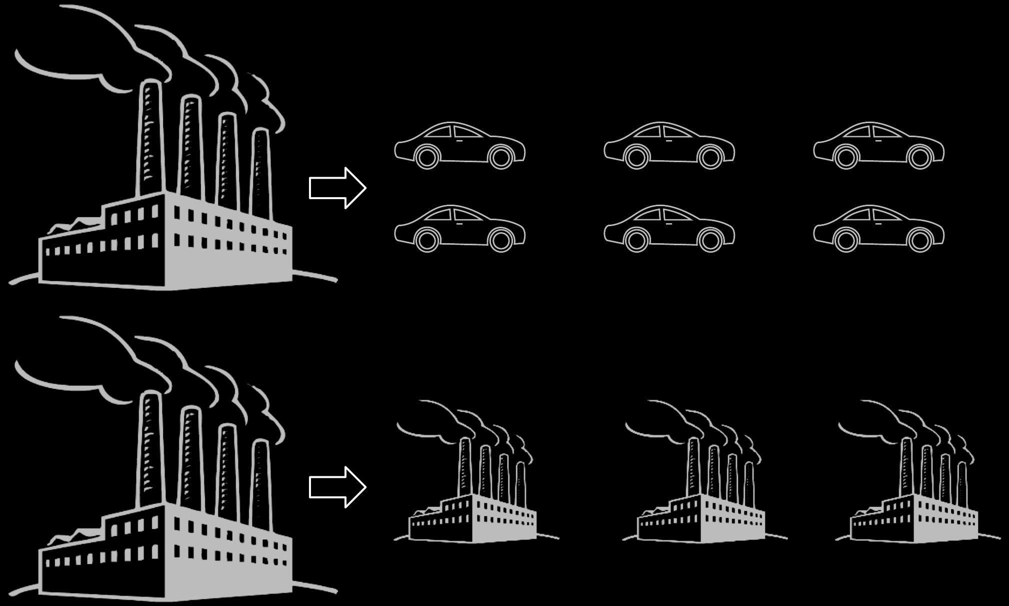
Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine



But, living things self-reproduce

How to resolve this paradox?

Self-reproducing machines are possible!

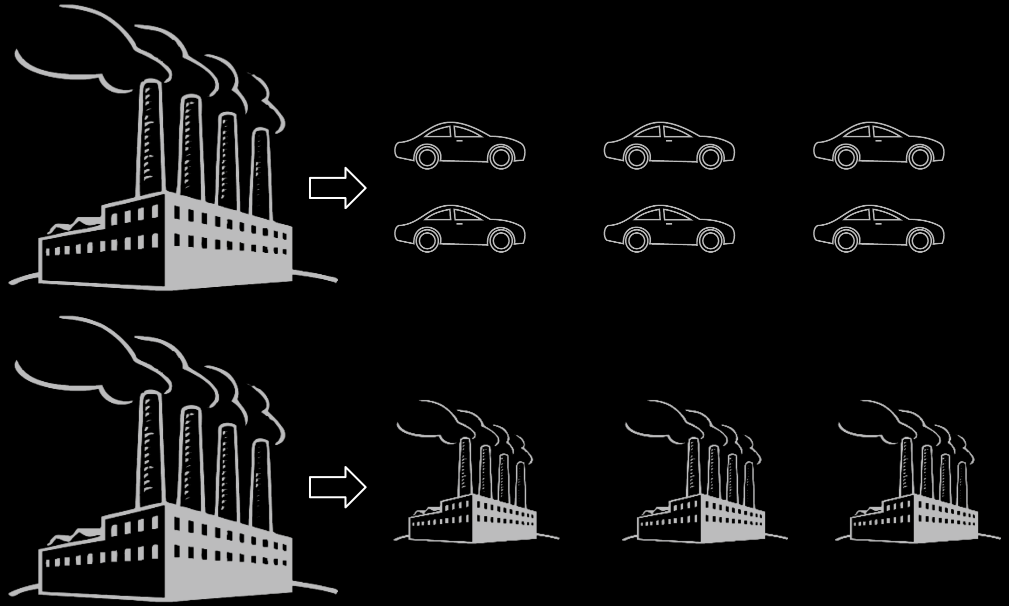
Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine



But, living things self-reproduce

How to resolve this paradox?

Self-reproducing machines are possible!

Self-reproduction Paradox

Suppose a Factory makes Cars

- Complexity of Factory > Complexity of Car
  (because Factory needs instructions for Car + robots, tools, …)

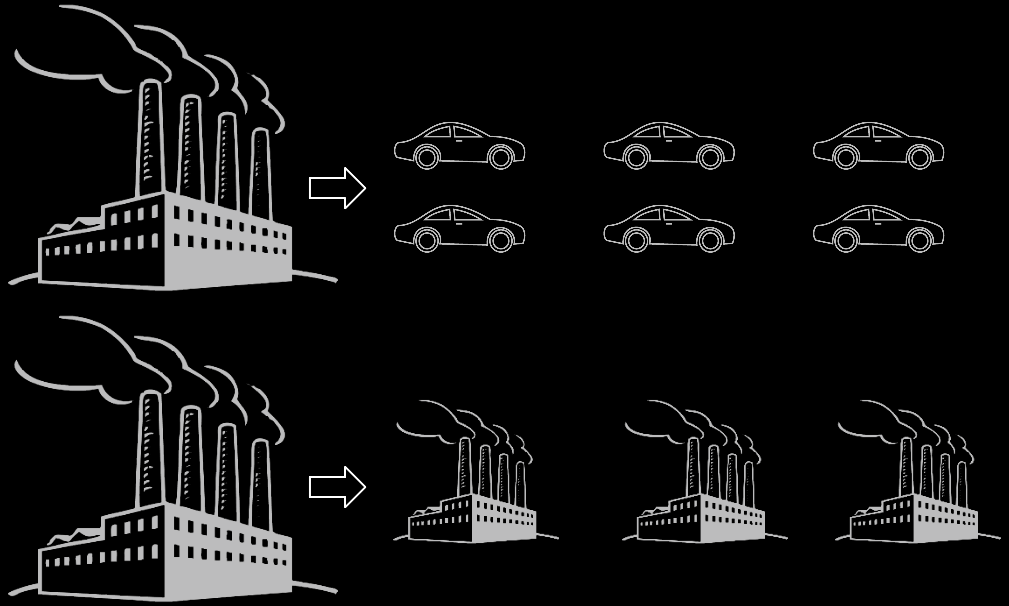
Can a Factory make Factories?

- Complexity of Factory > Complexity of Factory?
- Seems impossible to have a self-reproducing machine

But, living things self-reproduce

How to resolve this paradox?

Self-reproducing machines are possible!



A Self-Reproducing TM

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q : \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

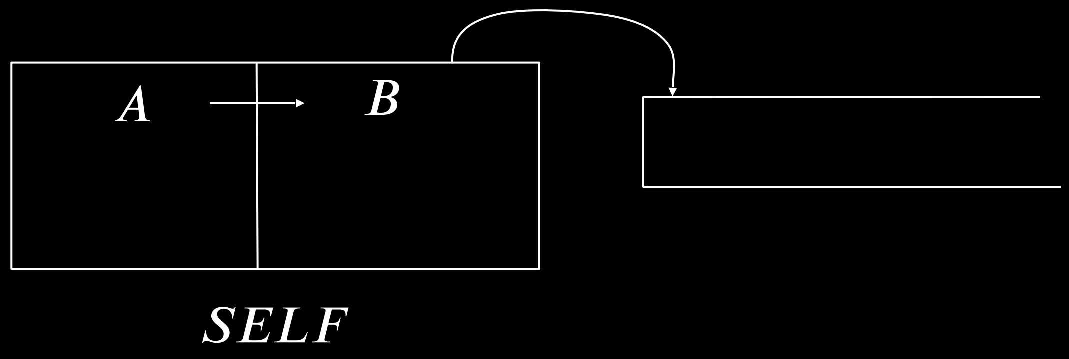
Proof of Theorem: $SELF$ has two parts, A and B.

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.



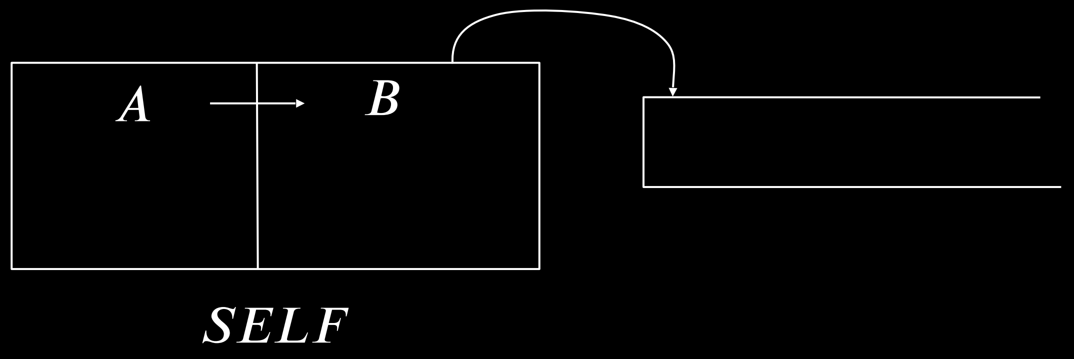
A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q\colon \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

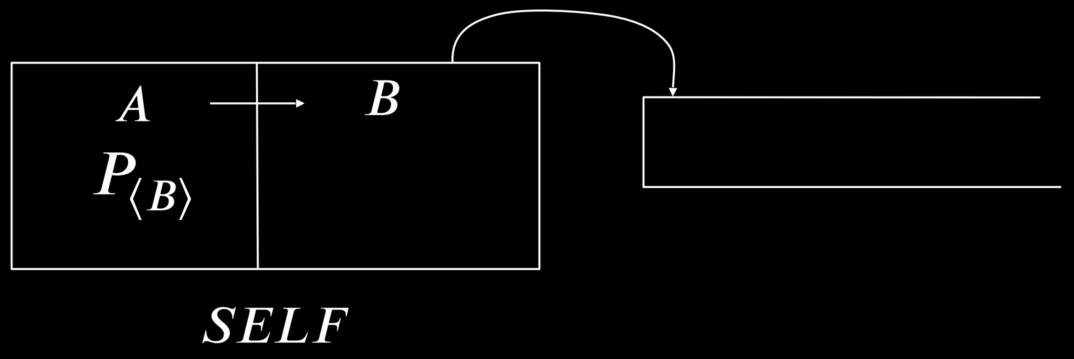


A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.
$A = \ P_{\langle B \rangle}$

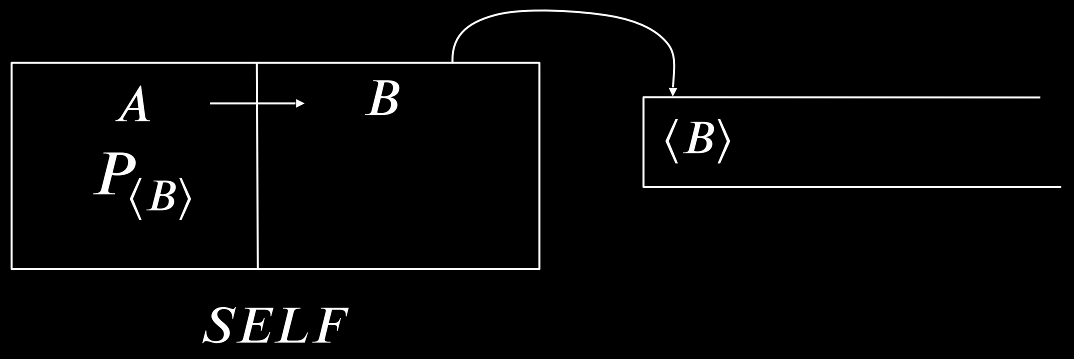


A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.
$A = \ P_{\langle B \rangle}$



A Self-Reproducing TM

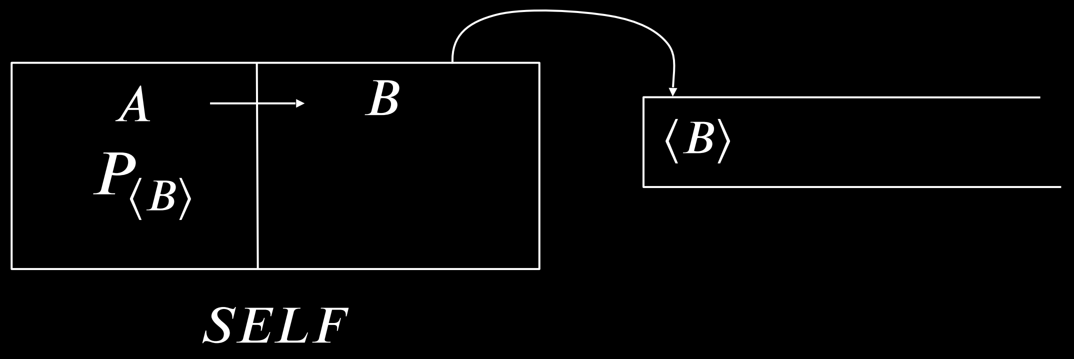
Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$?



A Self-Reproducing TM

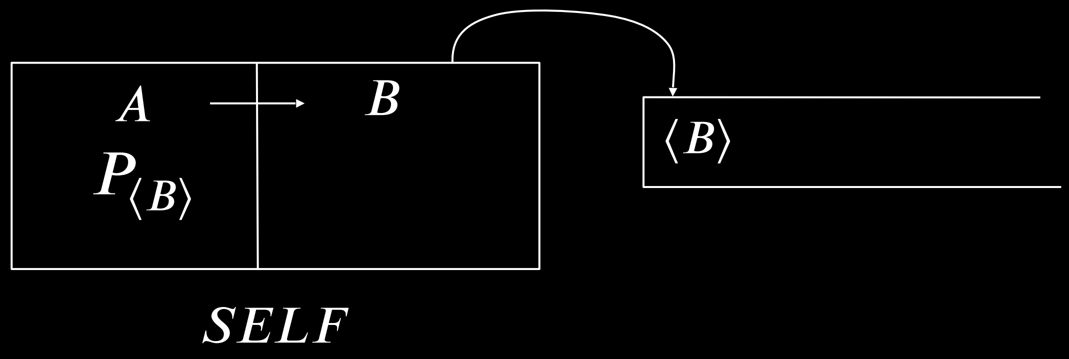
Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q\colon \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \; P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.



A Self-Reproducing TM

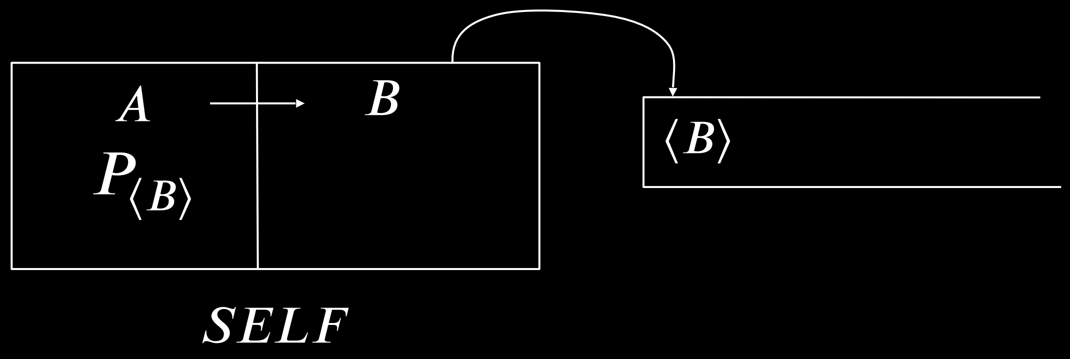
Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q\colon \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

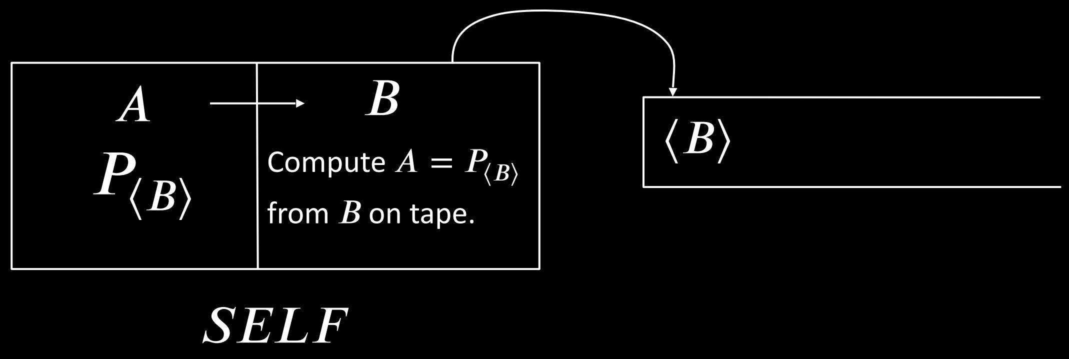


A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.
$A = \; P_{\langle B \rangle}$
$B = P_{\langle A \rangle}$? NO, would be circular reasoning.



A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

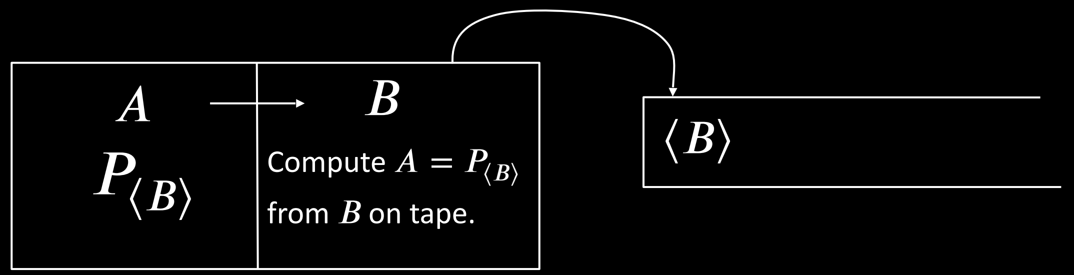
Lemma: There is a computable function $q\colon \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.



$SELF$

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

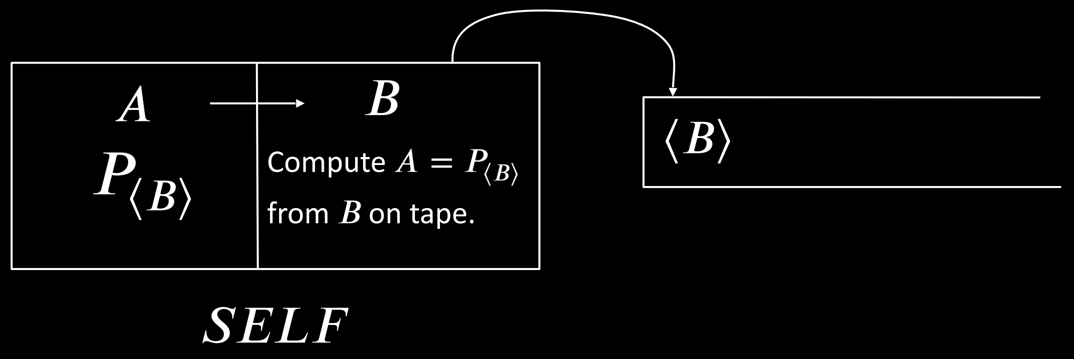
Lemma: There is a computable function $q: \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.



A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF\rangle$ on the tape.

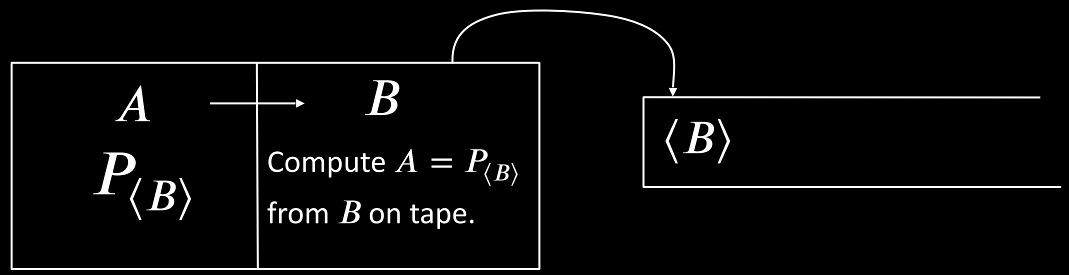
Lemma: There is a computable function $q\colon\Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w\rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B\rangle}$

$B = P_{\langle A\rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.
3. Halt with $\langle SELF\rangle$ on tape."



$SELF$

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

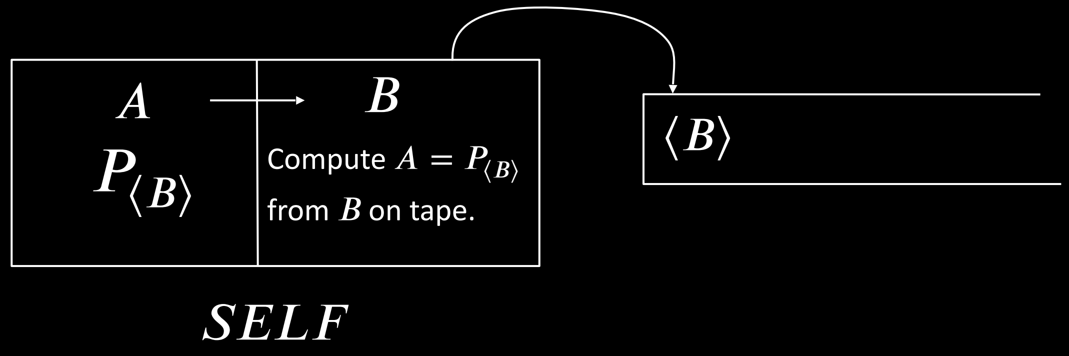
Proof of Theorem: $SELF$ has two parts, A and B.

$A = \; P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.
3. Halt with $\langle SELF \rangle$ on tape."

Given on the tape



A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q\colon \Sigma^* \rightarrow \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

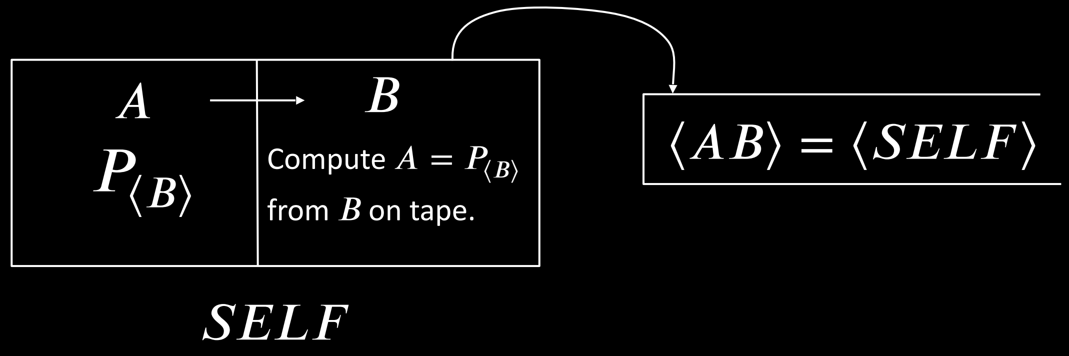
Proof of Theorem: $SELF$ has two parts, A and B.

$A = \;\; P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B = \;\;$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.
3. Halt with $\langle SELF \rangle$ on tape."

Given on the tape



A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q : \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

Proof of Theorem: $SELF$ has two parts, A and B.

$A = \quad P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.
3. Halt with $\langle SELF \rangle$ on tape."

Given on the tape

A	B
$P_{\langle B \rangle}$	Compute $A = P_{\langle B \rangle}$ from B on tape.

$SELF$

$\langle AB \rangle = \langle SELF \rangle$

Can implement in any programming language.

A Self-Reproducing TM

Theorem: There is a TM $SELF$ which (on any input) halts with $\langle SELF \rangle$ on the tape.

Lemma: There is a computable function $q: \Sigma^* \to \Sigma^*$ such that $q(w) = \langle P_w \rangle$ for every w, where P_w is the TM $P_w =$ "Print w on the tape and halt".
Proof: Straightforward.

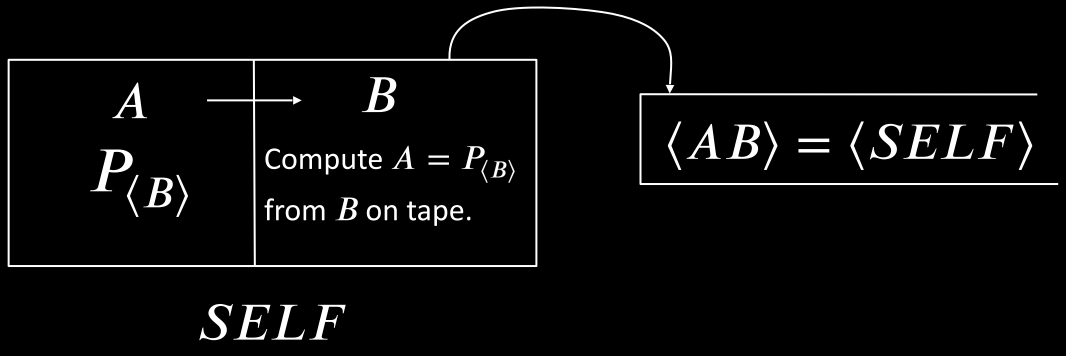
Proof of Theorem: $SELF$ has two parts, A and B.

$A = \ P_{\langle B \rangle}$

$B = P_{\langle A \rangle}$? NO, would be circular reasoning.

$B =$ "1. Compute q(tape contents) to get A.
2. Combine with B to get $AB = SELF$.
3. Halt with $\langle SELF \rangle$ on tape."

Given on the tape



Can implement in any programming language.

English Implementation

English Implementation

Write “Hello World”

English Implementation

Write "Hello World"

Hello World

English Implementation

Write “Hello World”

Hello World

Write this sentence

English Implementation

Write “Hello World”

Hello World

Write this sentence

Write this sentence

English Implementation

Write “Hello World”

Hello World

Write this sentence

Write this sentence

Cheating: TMs don’t have this self-reference primitive.

English Implementation

Write “Hello World”

Hello World

Write this sentence

Write this sentence

Cheating: TMs don’t have this self-reference primitive.

Write the following twice, the second time in quotes “Hello World”

English Implementation

Write “Hello World”

Hello World

Write this sentence

Cheating: TMs don’t have this self-reference primitive.

Write this sentence

Write the following twice, the second time in quotes “Hello World”

Hello World “Hello World”

English Implementation

Write “Hello World”

Hello World

Write this sentence

Write this sentence

Cheating: TMs don’t have this self-reference primitive.

Write the following twice, the second time in quotes “Hello World”

Hello World “Hello World”

Write the following twice, the second time in quotes
“Write the following twice, the second time in quotes”

Write the following twice, the second time in quotes
“Write the following twice, the second time in quotes”

English Implementation

Write “Hello World”

Hello World

Write this sentence

Cheating: TMs don’t have this self-reference primitive.

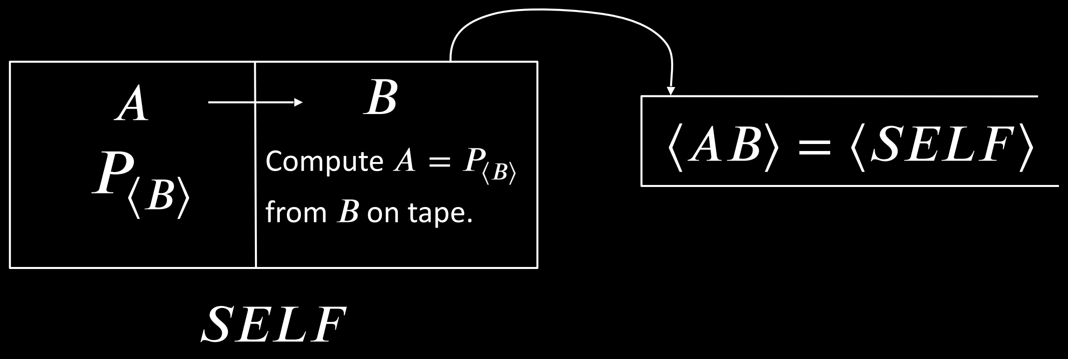
Write this sentence

Write the following twice, the second time in quotes “Hello World”

Hello World “Hello World”

Write the following twice, the second time in quotes
“Write the following twice, the second time in quotes”

Write the following twice, the second time in quotes
“Write the following twice, the second time in quotes”



English Implementation

Write "Hello World"

Hello World

Write this sentence

Write this sentence

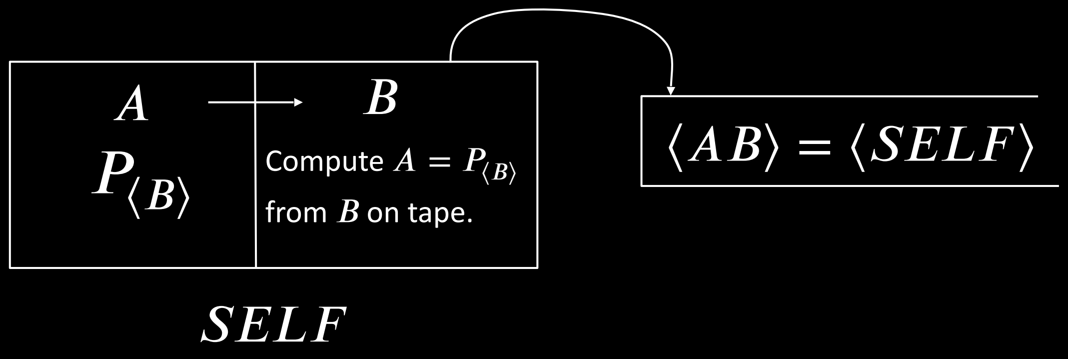
Cheating: TMs don't have this self-reference primitive.

Write the following twice, the second time in quotes "Hello World"

Hello World "Hello World"

Write the following twice, the second time in quotes
"Write the following twice, the second time in quotes"

Write the following twice, the second time in quotes
"Write the following twice, the second time in quotes"



Check-in 11.1

English Implementation

Check-in 11.1

Implementations of the Recursion Theorem have two parts, a <u>Template</u> and an <u>Action</u>. In the TM and English implementations, which is the <u>Action</u> part?

(a) A and the upper phrase

(b) A and the lower phrase

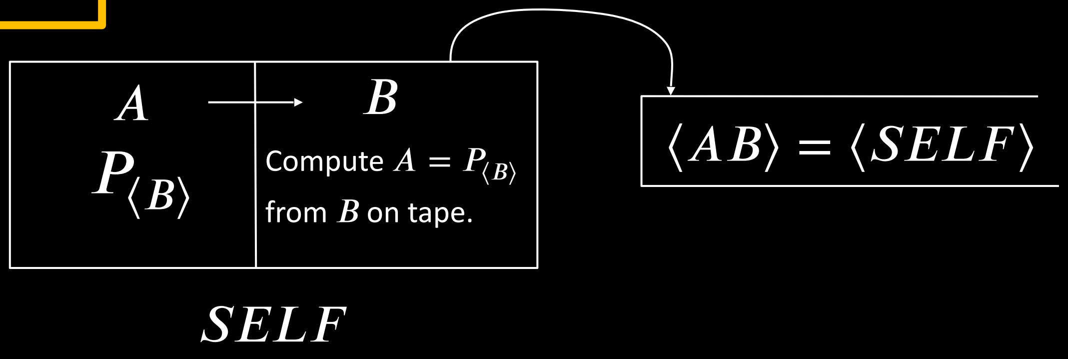
(c) B and the upper phrase

(d) B and the lower phrase.

Hello World "Hello World"

Write the following twice, the second time in quotes
"Write the following twice, the second time in quotes"

Write the following twice, the second time in quotes
"Write the following twice, the second time in quotes"



Check-in 11.1

English Implementation

Check-in 11.1

Implementations of the Recursion Theorem have two parts, a Template and an Action. In the TM and English implementations, which is the Action part?

(a) A and the upper phrase

(b) A and the lower phrase

(c) B and the upper phrase

(d) B and the lower phrase.

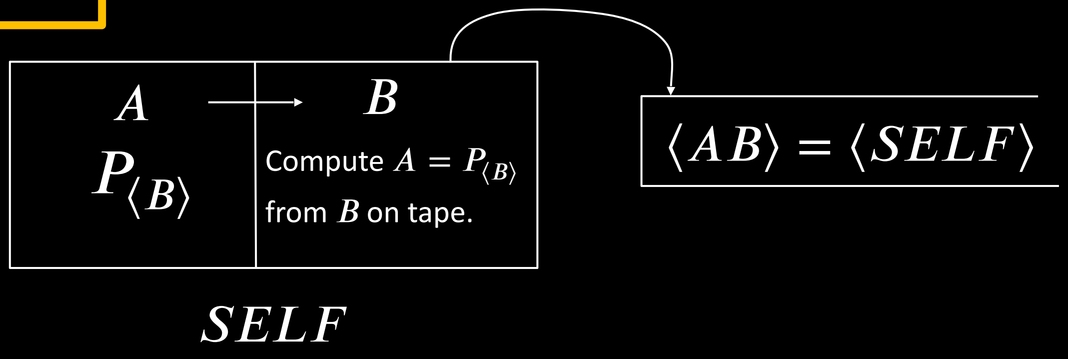
Hello World "Hello World"

Write the following twice, the second time in quotes

"Write the following twice, the second time in quotes"

Write the following twice, the second time in quotes

"Write the following twice, the second time in quotes"



Note on Pset Problem 6: Don't need to worry about quoting.

Check-in 11.1

The Recursion Theorem

The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w

R on input w operates in the same way as T on input $\langle w, R\rangle$.

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w

R on input w operates in the same way as T on input $\langle w, R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.

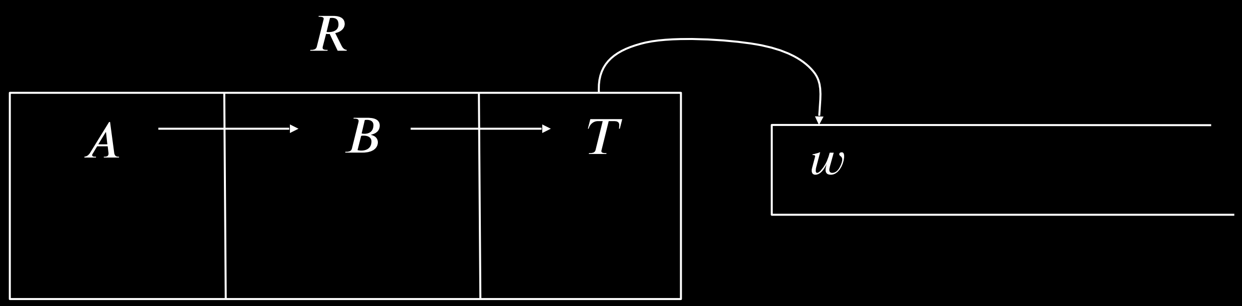
The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w

R on input w operates in the same way as T on input $\langle w,\ R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.

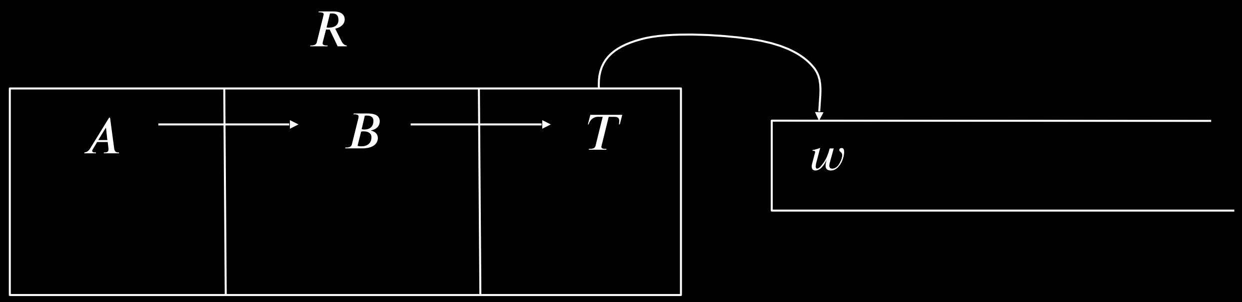


The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.
T is given

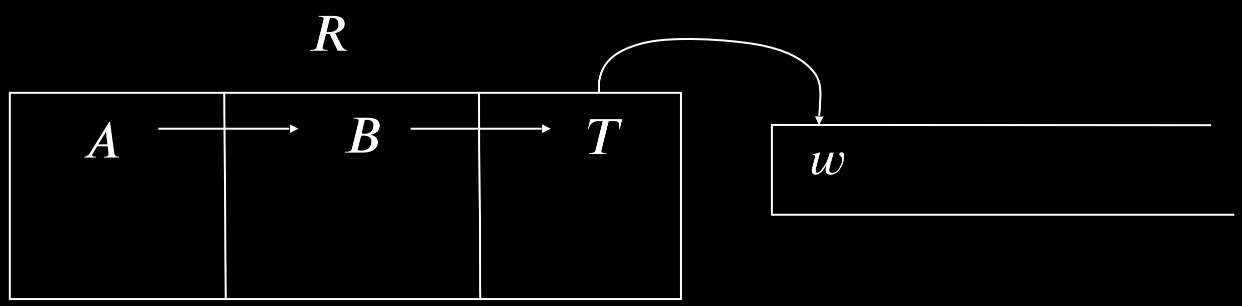


The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, \ R \rangle$.

Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \ \ P_{\langle BT \rangle}$



The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

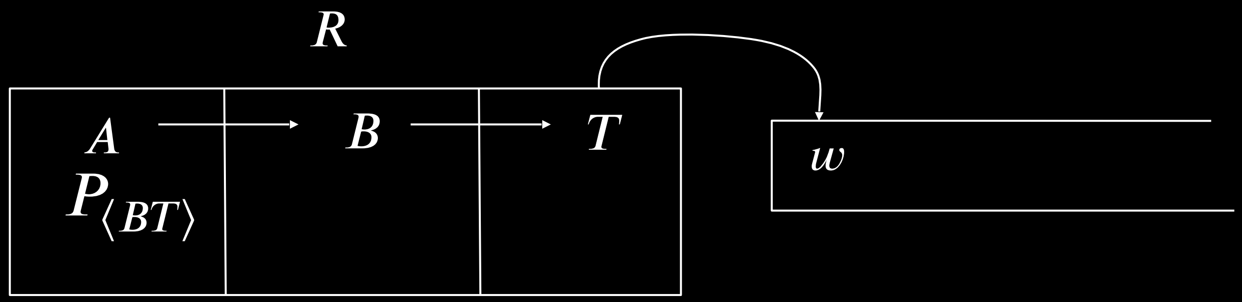
Theorem: For any TM T there is a TM R where for all w

R on input w operates in the same way as T on input $\langle w,\ R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.

T is given

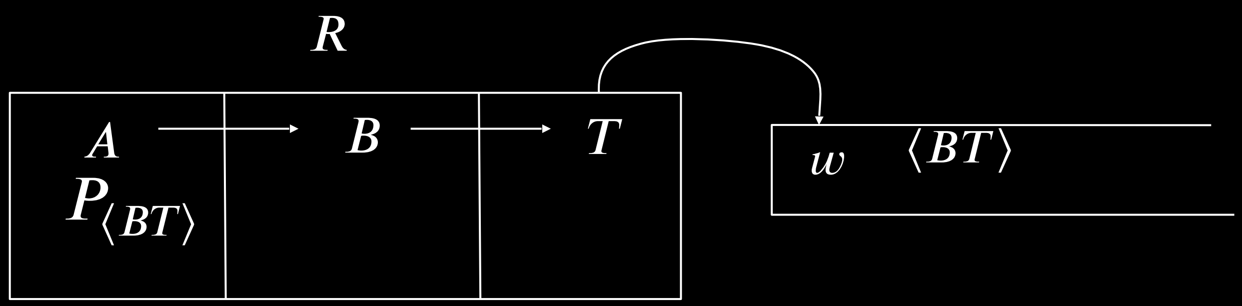
$A = \ P_{\langle BT\rangle}$



The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w,\ R\rangle$.



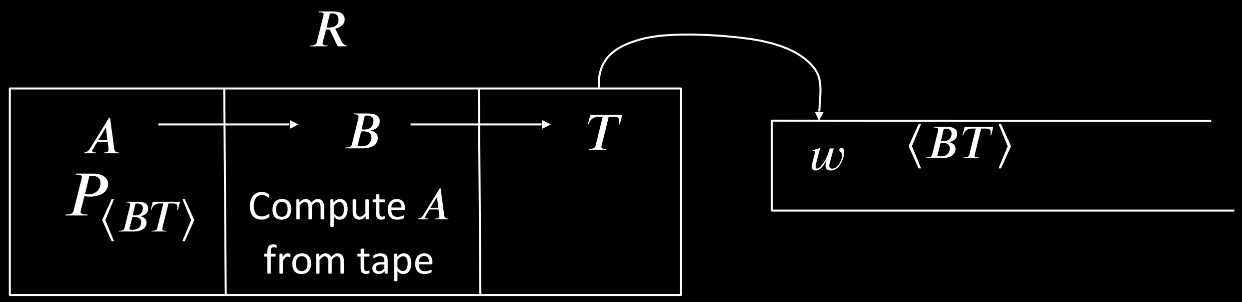
Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \ \ P_{\langle BT\rangle}$

The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R \rangle$.

Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \ P_{\langle BT \rangle}$

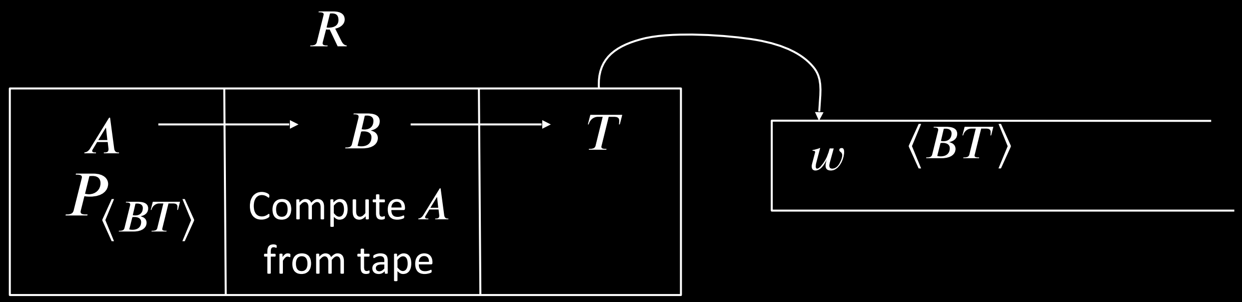


The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \quad P_{\langle BT\rangle}$
$B = \quad$ “1. Compute q(tape contents after w) to get A.

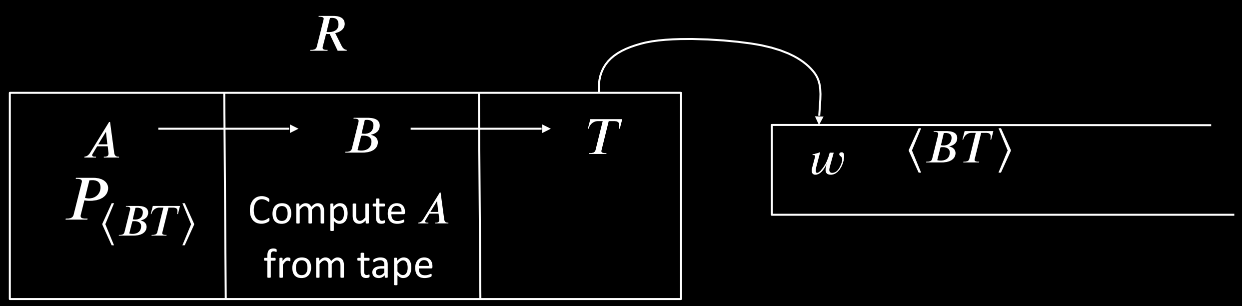


The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w

R on input w operates in the same way as T on input $\langle w, R\rangle$.



Proof of Theorem: R has three parts: A, B, and T.

T is given

$A = \quad P_{\langle BT\rangle}$

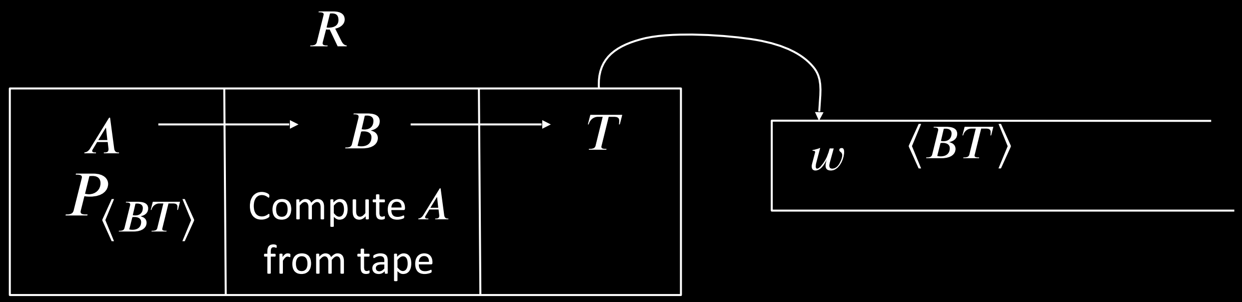
$B = \quad$ "1. Compute q(tape contents after w) to get A.

2. Combine with BT to get $ABT = R$.

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R\rangle$.

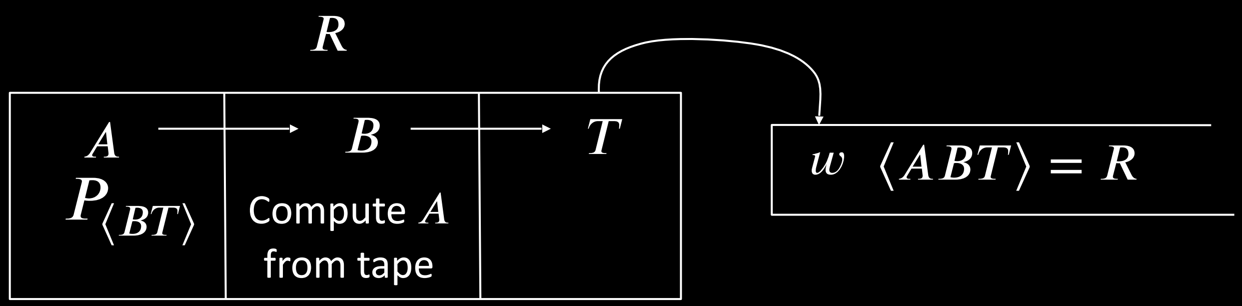


Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \quad P_{\langle BT\rangle}$
$B = \quad$ "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w, R\rangle$."

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R\rangle$.



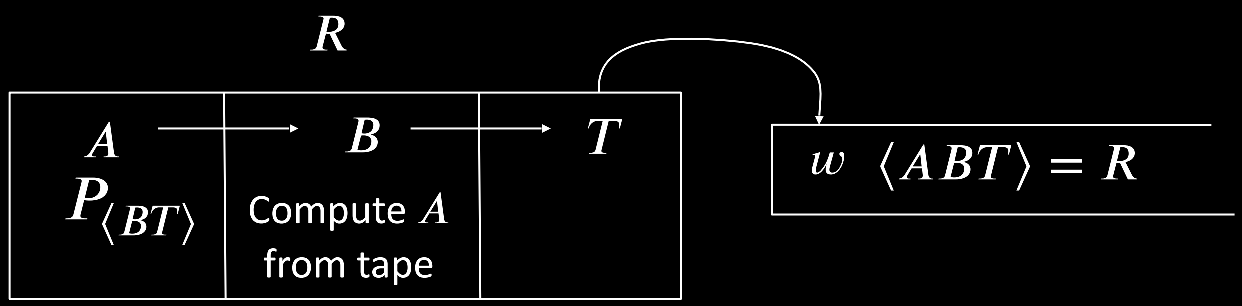
Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \quad P_{\langle BT\rangle}$
$B = \quad$ "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w, R\rangle$."

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w, R\rangle$.

Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \; P_{\langle BT\rangle}$
$B = \;$ "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w, R\rangle$."

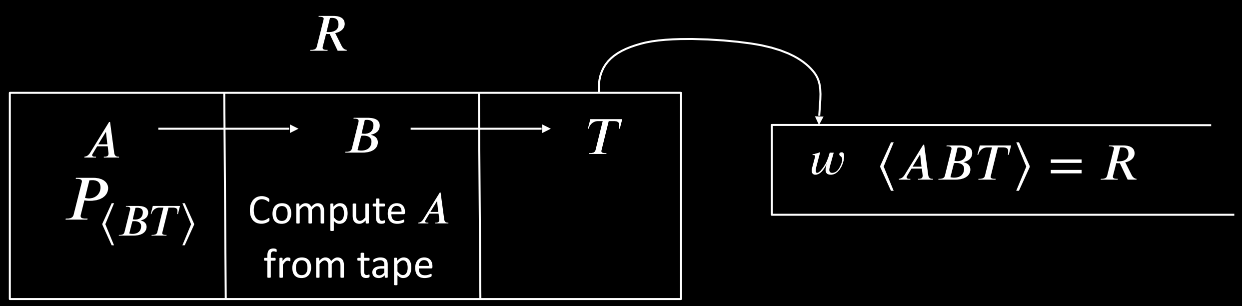


Moral: You can use "compute your own description" in describing TMs.

The Recursion Theorem

A compiler which implements "compute your own description" for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w,\ R\rangle$.



Proof of Theorem: R has three parts: A, B, and T.
T is given
$A = \ \ P_{\langle BT\rangle}$
$B = $ "1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w,\ R\rangle$."

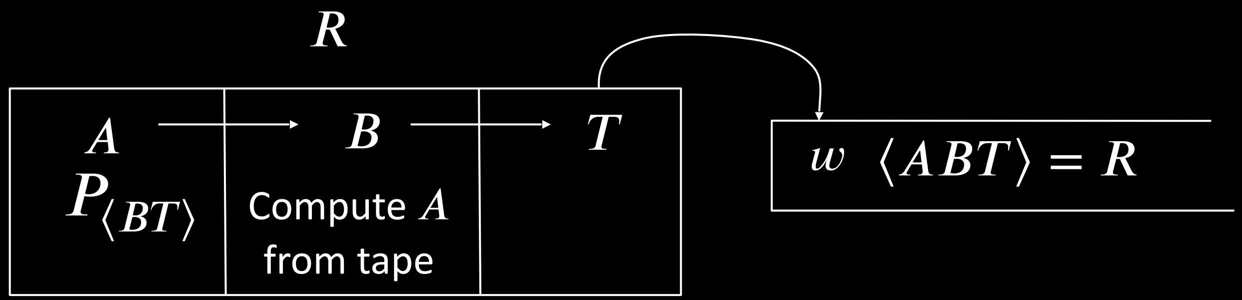
Moral: You can use "compute your own description" in describing TMs.

Check-in 11.2

The Recursion Theorem

A compiler which implements “compute your own description” for a TM.

Theorem: For any TM T there is a TM R where for all w
R on input w operates in the same way as T on input $\langle w,\ R\rangle$.



Proof of Theorem: R has three parts: A, B, and T.

T is given

$A = \quad P_{\langle BT\rangle}$

$B = \quad$ “1. Compute q(tape contents after w) to get A.
2. Combine with BT to get $ABT = R$.
3. Pass control to T on input $\langle w,\ R\rangle$.”

Check-in 11.2

Can we use the Recursion Theorem to design a TM T where $L(T) = \{\langle T\rangle\}$?

(a) Yes.

(b) No.

Moral: You can use “compute your own description” in describing TMs.

Check-in 11.2

Ex 1: ATM is undecidable - new proof

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

$R =$ "On input w

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

$R =$ "On input w

1. Get own description $\langle R \rangle$.

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

$R =$ "On input w

1. Get own description $\langle R \rangle$.
2. Use H on input $\langle R, w \rangle$ to determine whether R accepts w.

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

$R =$ "On input w

1. Get own description $\langle R \rangle$.
2. Use H on input $\langle R, w \rangle$ to determine whether R accepts w.
3. Do the opposite of what H says."

Ex 1: ATM is undecidable - new proof

Theorem: A**TM** is not decidable

Proof by contradiction: Assume some TM H decides ATM.

Consider the following TM R:

$R =$ "On input w

1. Get own description $\langle R \rangle$.
2. Use H on input $\langle R, w \rangle$ to determine whether R accepts w.
3. Do the opposite of what H says."