Draw a DFA (or NFA!) that describes your typical day.

Full name

R. 9/13

Which would you prefer?

1. DFAs

2. NFAs

3. Regular expressions
e.g., `10*1`

Which is more powerful?

1. DFAs

2. NFAs

3. Regular expressions
e.g., `10*1`

Regular Languages

NFAs ≡ DFAs ≡ Regular Expressions

(Kleene's theorem)

What counts as a problem?

Decision problems on finite, bitstring inputs.

What kinds of **problems** can **computers** solve?

DFAs, NFAs, REs.

What counts as a computer?

Some interesting decision problems

Let's create a DFA (or NFA or RE) for these

1. $L = \{a^N b^N \mid N > 0\}$ // equality?

   this means N repetitions of the character `a`

2. $L = \{a^N b^{2N} \mid N > 0\}$ // multiplication?

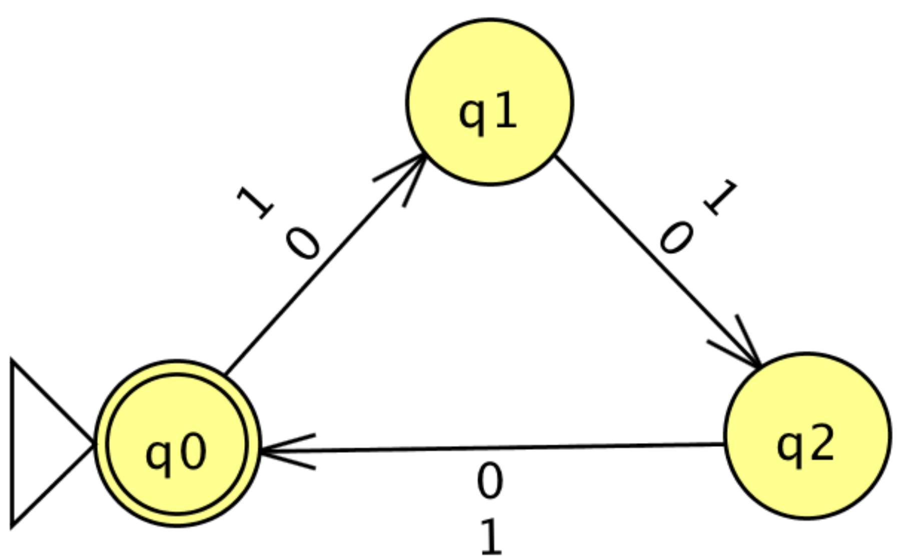
3. $L = \{a^N b^M c^{(N+M)} \mid N,M > 0\}$ // addition?

Not Regular

we cannot build a DFA that accepts L

Recall: Distinguishability theorem

If a set of strings $S = \{w_1, w_2, \ldots, w_n\}$ is pairwise distinguishable for a language L, then any DFA that accepts L must have at least n states.



		w_1 λ	w_2 1	w_3 11
w_1	λ	–	11	1
w_2	1	–	–	1
w_3	11	–	–	–

$L = \{w \mid w$'s length is divisible by $3\}$ has a DFA with at least 3 states.

Recall: Distinguishability theorem

What if…

A set of strings $S = \{w_1, w_2, \ldots\}$ is pairwise distinguishable for a language L, and the size of S is infinite?!

$$L = \{a^N b^N \mid N > 0\}$$

$$S = \{a, aa, aaa, \ldots\}$$

w_i	w_j	z	Accept	Reject
a	aa	b	ab	aab
a	aaa	b	ab	aaab
a	aaaa	b	ab	aaaab
a	aaaaa	b	ab	aaaaab

*…and so on for **every** pair of unequal strings in S…*

When (not) to use regular languages

Regular languages **are** useful for

- processes that require a finite number of steps
- recognizing text without needing to remember arbitrary amounts of previous input

Regular languages are **not** useful for

- modeling the full power of a computer

What counts as a problem?

Decision problems on finite, bitstring inputs.

What kinds of **problems** can **computers** solve?

DFAs, NFAs, REs can't solve very many kinds of decision problems!

What counts as a computer?

Deterministic Finite Automaton

Formal definition

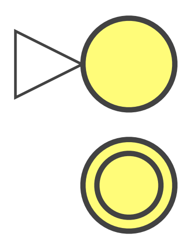
A machine M that consists of:

an **alphabet** Σ

a finite set of **states**, including:

initial state

accepting state(s)



transitions between states

for every state, every letter in Σ labels one and only one transition

Given a string w,
M **accepts** w
if consuming w causes
M to terminate in
an accepting state.

what's missing?

Turing Machines

Formal definition

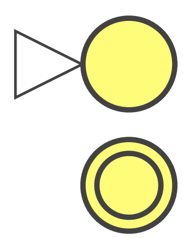
A machine *M* that consists of:

an **alphabet** Σ

a finite set of **states**, including:

initial state

accepting state(s)



transitions between states

an infinitely large **tape**, which can be read or written

the tape is akin to memory

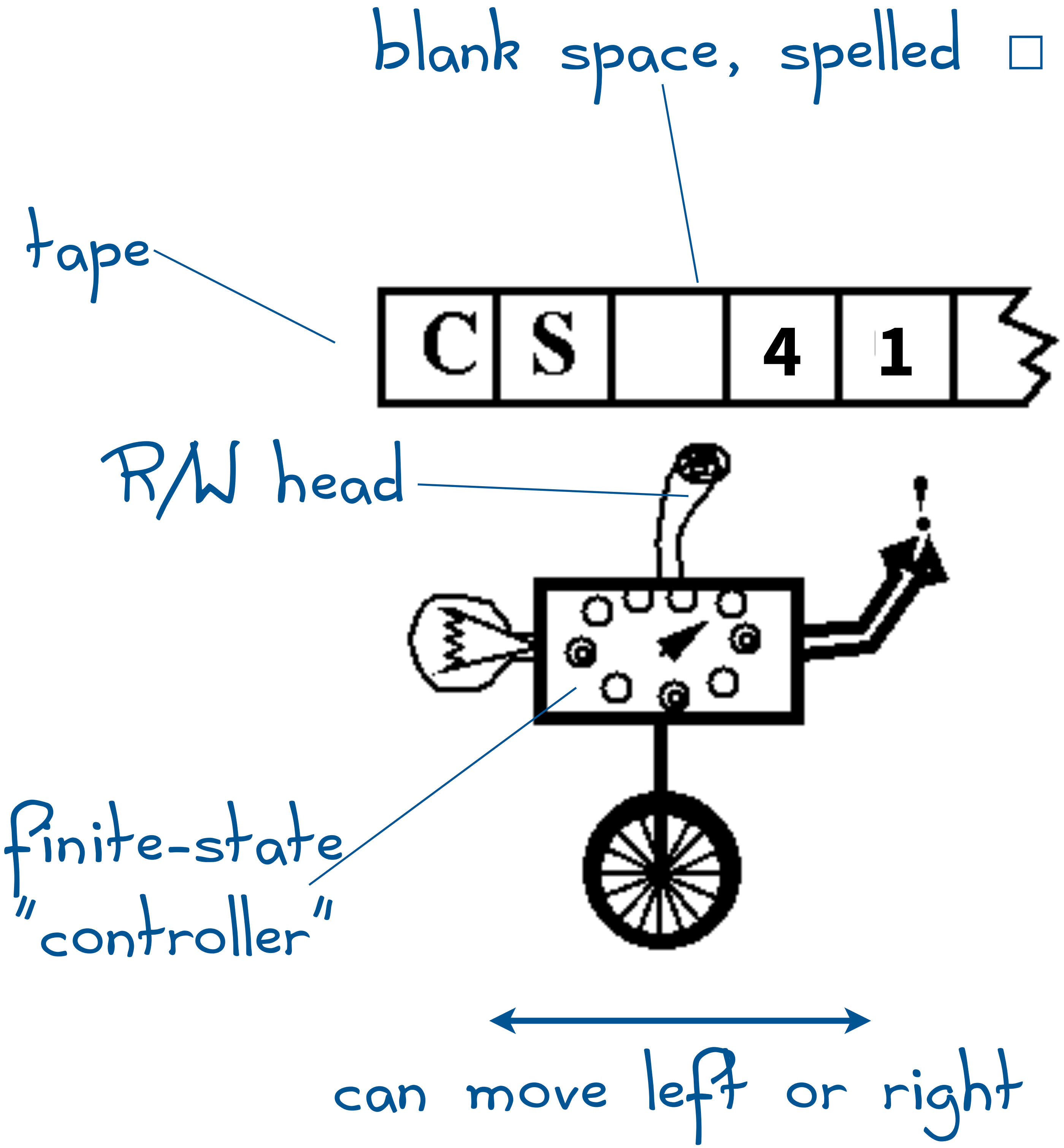
a **current location** on the tape

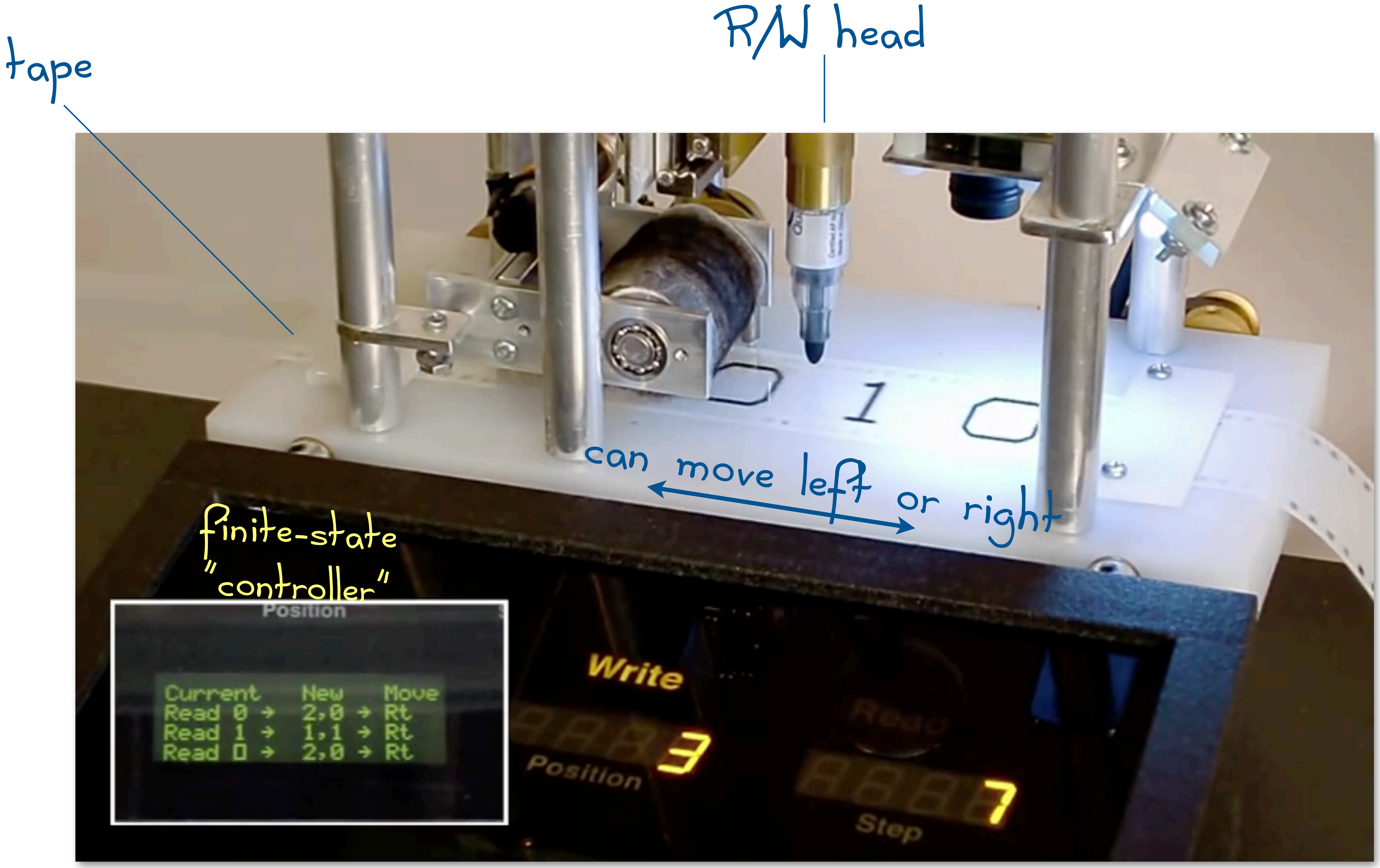
called the “read/write head”

Given a string *w*,
M **accepts** *w*
if consuming *w* causes
M to terminate in
an accepting state.

Turing Machine

Artist's conception

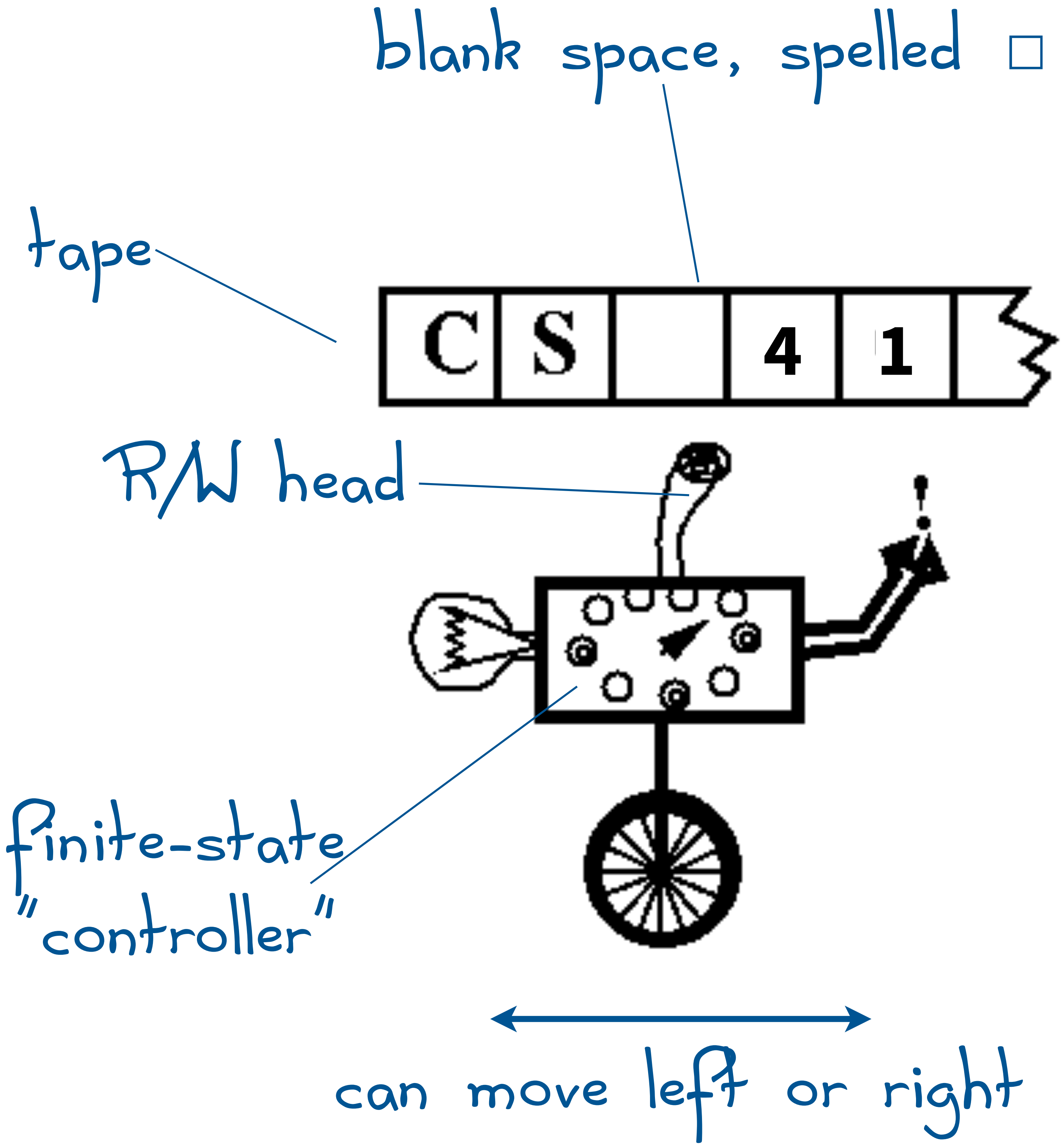




https://youtu.be/E3keLeMwfHY

Turing Machine

Artist's conception



One transition:

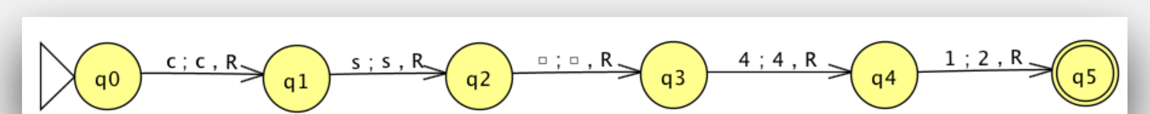
"If I see a C..."

"...write a C..."

"...and move right..."

C ; C ; R

A finite-state controller:



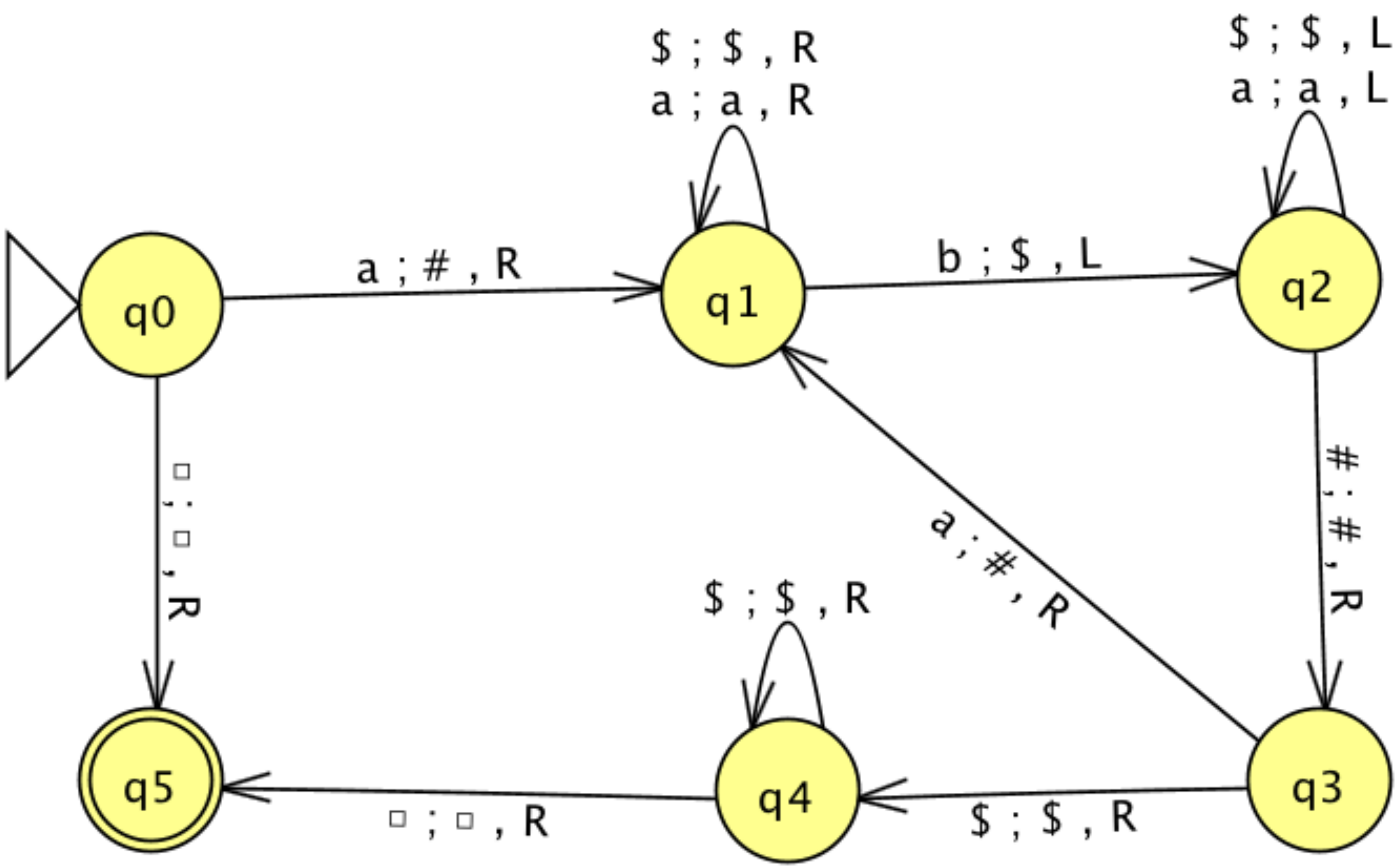
Rewrites "CS 41" to "CS 42"

Let’s practice!

What does this machine do for the input aabb?

What does this machine do for the input abb?

What does this machine do in general?



cdn1.medicalnewstoday.com/content/images/articles/322/322156/platypus-swimming.jpg

Turing Machines FTW!

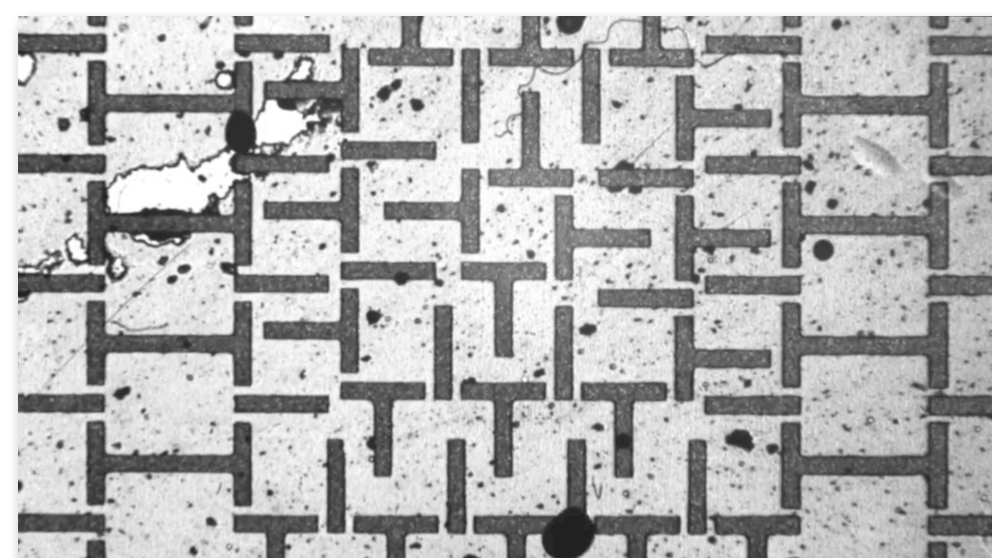
(1) $L = \{a^N b^N \mid N > 0\}$ // equality?

(2) $L = \{a^N b^{2N} \mid N > 0\}$ // multiplication?

(3) $L = \{a^N b^M c^{(N+M)} \mid N,M > 0\}$ // addition?

So far, all known computational devices are equivalent to Turing Machines…

Quantum computers
Molecular computers
Parallel computers
Integrated circuits
Water-based computation
…



What counts as a problem?

Decision problems on finite, bitstring inputs.

What kinds of **problems** can **computers** solve?

Turing Machines can solve way more problems than DFAs!

What counts as a computer?

Turing Machines FTW?

Here's a strategy for doing every HW assignment in every class:

(1) Spend a week writing a program that takes as input a description of any assignment and outputs the solution.

(2) There is no step two.

Turing Machines FTW?

Here's a strategy for winning mathematical fame and glory:

(1) Write a program that searches for an even integer *n* greater than 2 that is not the sum of two prime numbers. The program halts when it finds *n*.

This program looks for a counter-example to the unproven *Goldbach Conjecture*.

(2) Write a second program that takes as input the *first* program (!), then returns false if that program will ever halt and true otherwise.

We must know.
We will know.

David Hilbert

upload.wikimedia.org/wikipedia/commons/7/79/Hilbert.jpg

We must know.
We will know.

David Hilbert

upload.wikimedia.org/wikipedia/commons/7/79/Hilbert.jpg

No.

Kurt Gödel

plus.maths.org/issue39/features/dawson/Godel_Einstein.jpg

This sentence is false.

The Halting Problem is undecidable

L = {All programs that halt and give an answer}

"undecidable" means:

"We cannot create a Turing machine that can tell us whether every possible string is in this language or not."

The Halting Problem is undecidable

L = {All programs that halt and give an answer}

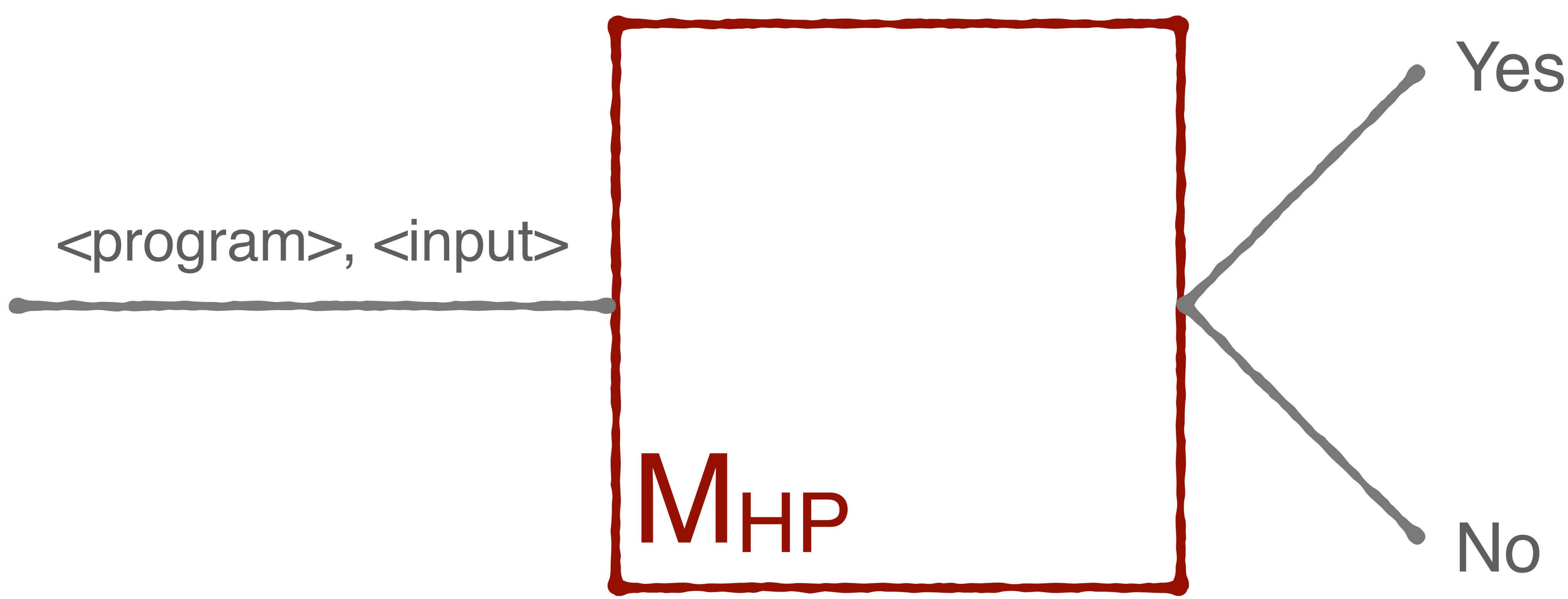
Proof sketch (proof by contradiction):

1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a contradiction.
3. Therefore the assumption (that the HP is decidable) is false.

The Halting Problem is undecidable

L = {All programs that halt and give an answer}

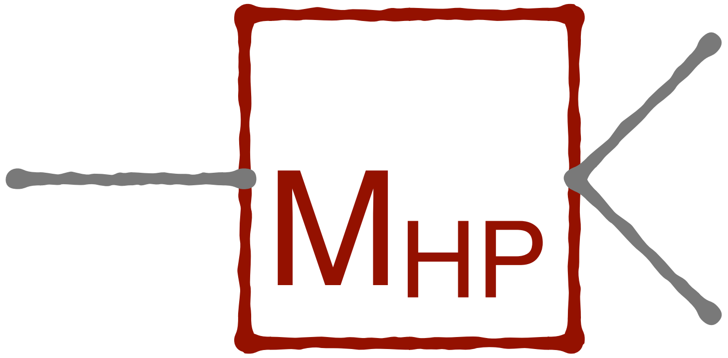
1. Assume that Halting Problem is decidable.



The Halting Problem is undecidable

L = {All programs that halt and give an answer}

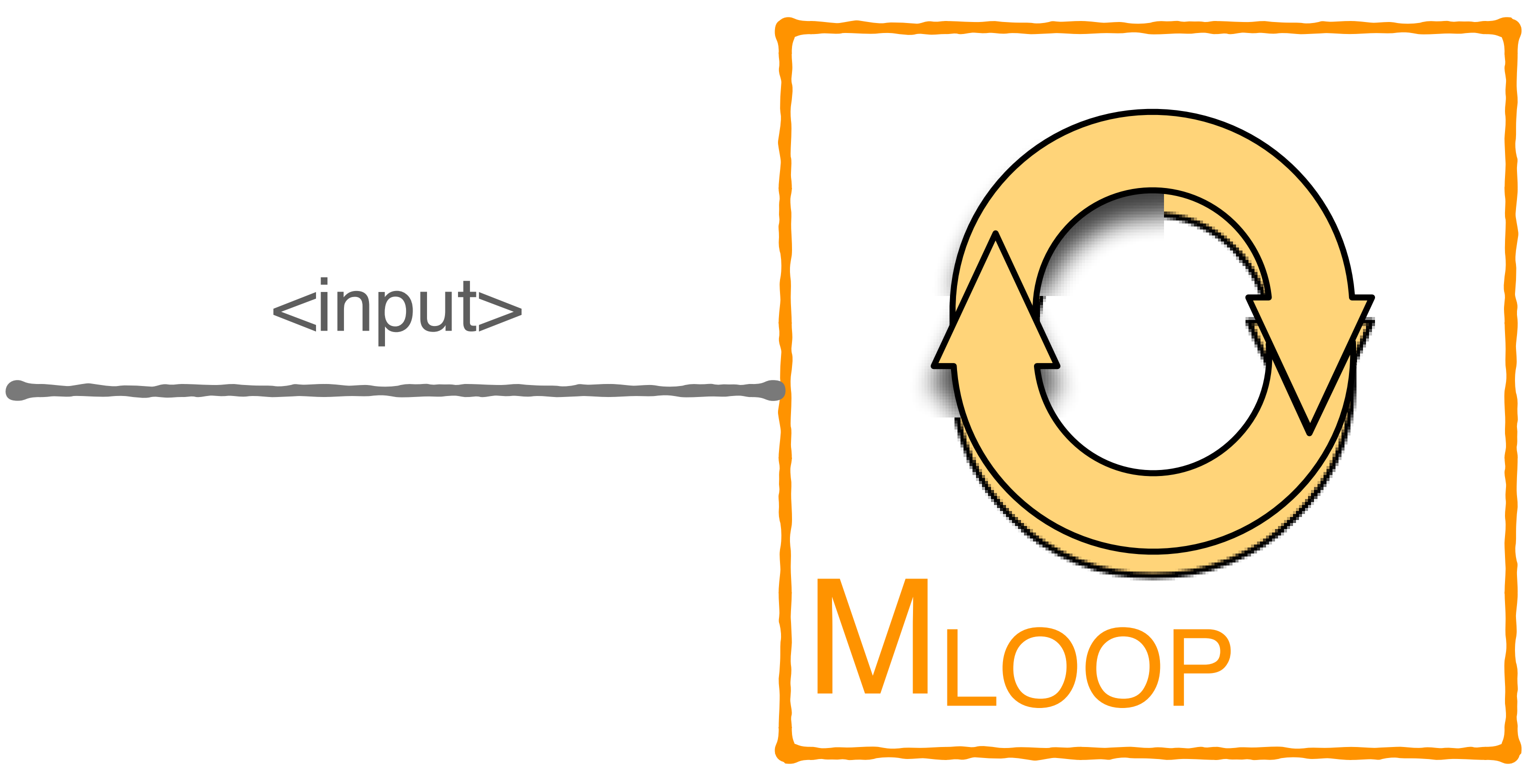
1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a contradiction.

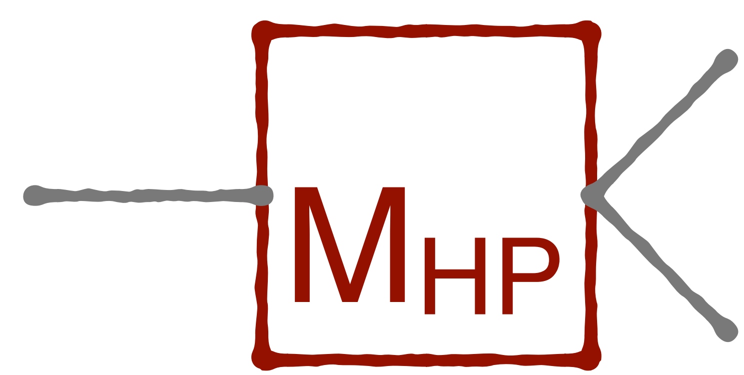


The Halting Problem is undecidable

L = {All programs that halt and give an answer}

1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a contradiction.

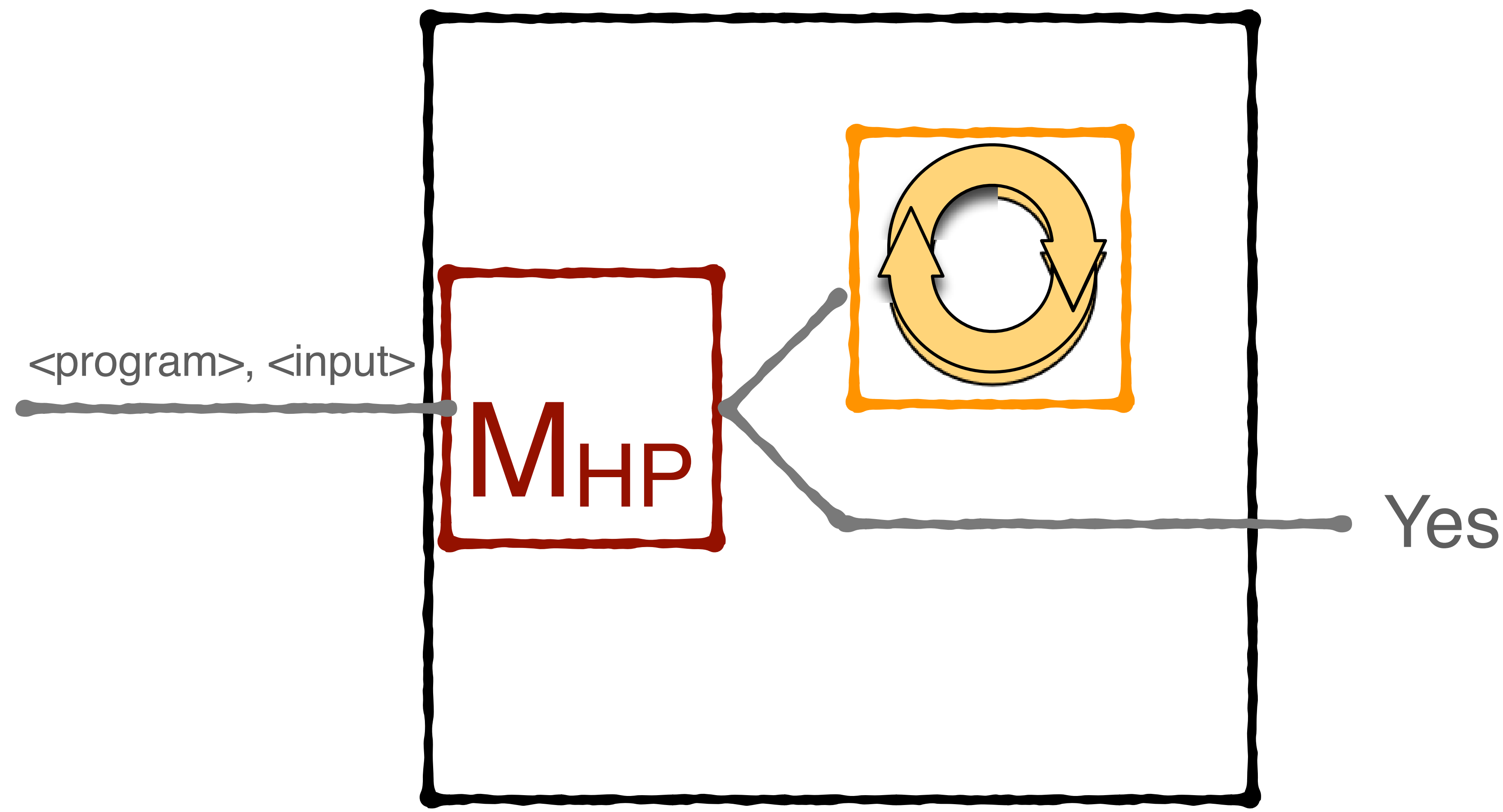




The Halting Problem is undecidable

L = {All programs that halt and give an answer}

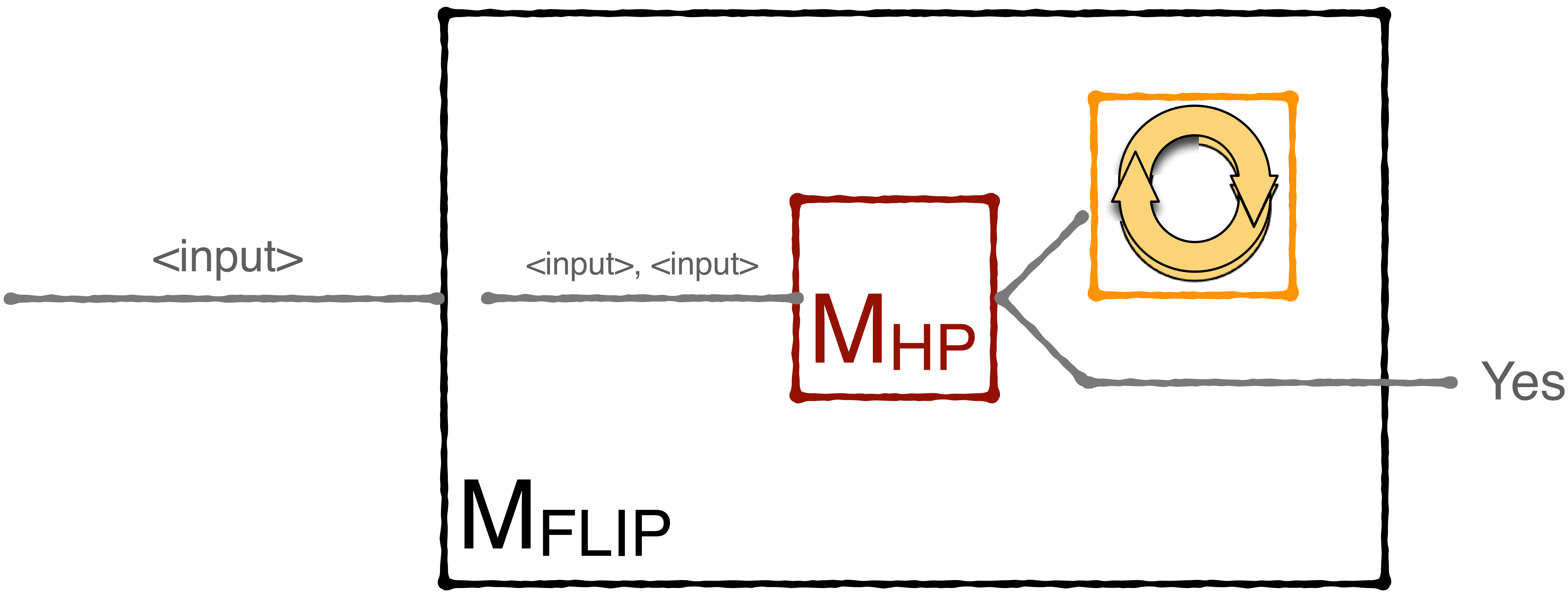
1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a contradiction.



The Halting Problem is undecidable

L = {All programs that halt and give an answer}

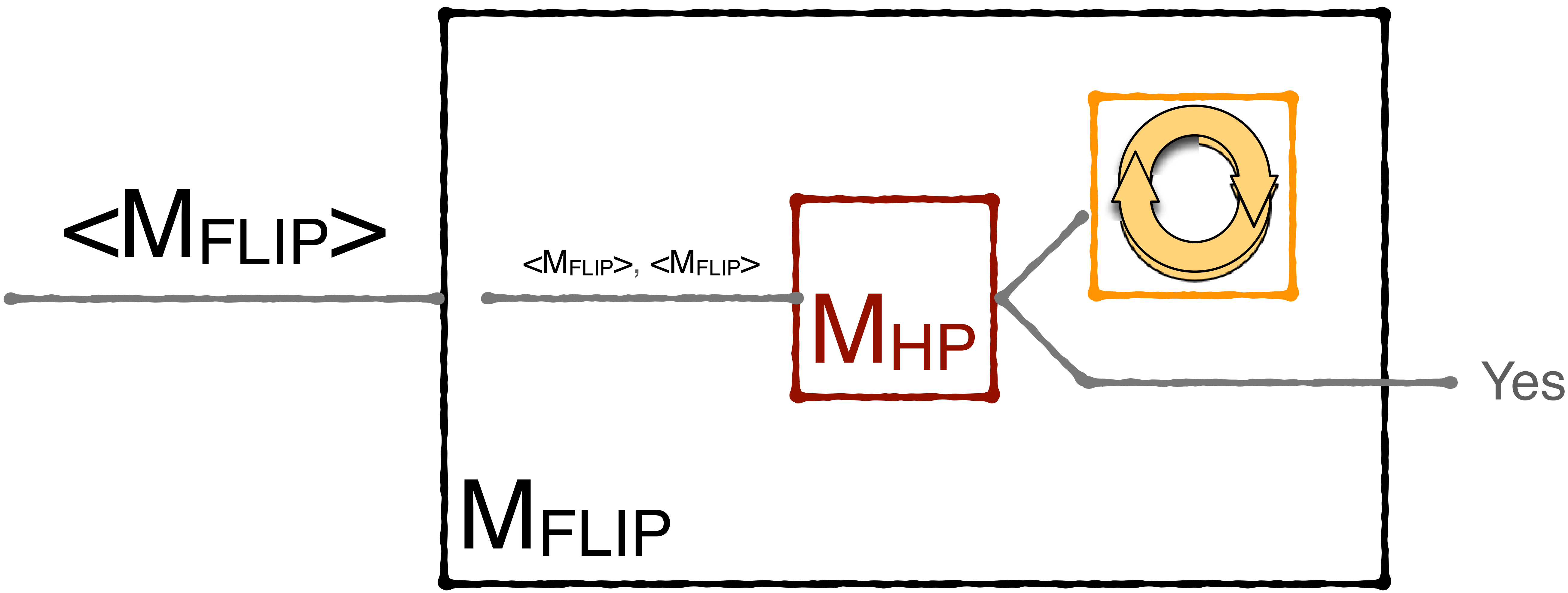
1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a contradiction.



The Halting Problem is undecidable

L = {All programs that halt and give an answer}

1. Assume that Halting Problem is decidable.
2. Show that this assumption leads to a **contradiction**.



Programs $\equiv$ Data