CS3102 Theory of Computation

www.cs.virginia.edu/~njb2b/cstheory/s2020

Warm up:

How might we try to show that this language is not computable:

$$Infinite = \{w \mid L(\mathcal{M}(w)) \text{ is infinite}\}$$

How to show things aren't computable

1. Ask "can I have an always-halting Turing machine M_p for language/function/problem P?"

2. Show that, if M_p exists, it can be used to make an impossible machine M_{imp}

How do we know a machine is impossible?

Option 1: It contradicts itself (e.g. M_{SR})
Option 2: Someone has done this before (e.g. M_{acc})

Proving Other Problems are Uncomputable

- Reduction
  - Convert some problem into a known uncomputable one (using only computable steps)
  - Show how you can use a solution to one problem to help you to solve another

Non-Computable Problems

MacGyver’s Reduction

Problem **know** is impossible

A

Opening a door

Solution for $\boldsymbol{A}$

Keg cannon battering ram

Aim duct at door, insert keg

Put fire under the Keg

Reduction

Problem we **think** is impossible

B

Lighting a fire

If

Solution for $\boldsymbol{B}$

Alcohol, wood, matches

Example: $FINITE$

- $FINITE(w) = \begin{cases} 1 \text{ if } L\big(\mathcal{M}(w)\big) \text{is finite} \\ 0 \text{ if } L\big(\mathcal{M}(w)\big) \text{is infinite} \end{cases}$
- To show $FINITE$ is uncomputable
  - Show how to use a TM for $FINITE$ to solve $HALT$
    - $FINITE \geq HALT$
    - $HALT$ reduces to $FINITE$

$FINITE$ Reduction

Problem **know** is impossible

A

$HALT$

Does $\mathcal{M}(w)$ halt on input x?

Build (but don't run) M_{wx}

Problem we **think** is impossible

B

$FINITE$

Is $\mathrm{L}(\mathcal{M}(w))$ finite?

Assume

Give M_{wx} as input to M_{finite} answer opposite

M_{HALT} computes $HALT$

M_{finite} computes $FINITE$

Reduction

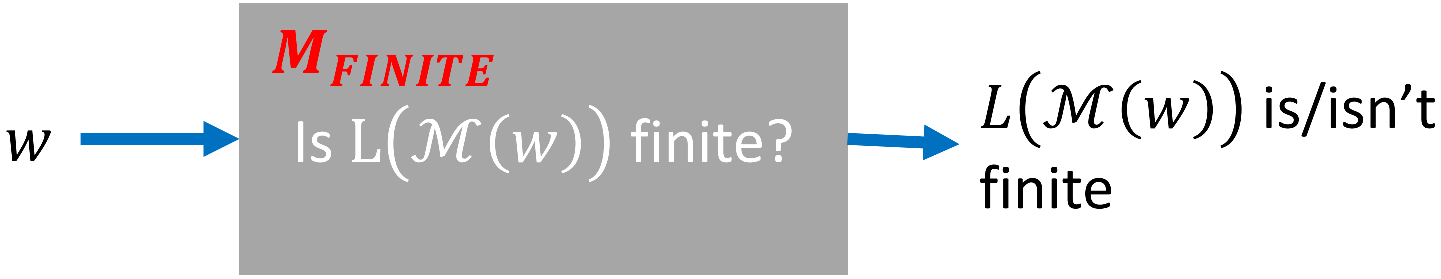
What's the Language of M_{wx}?

- If $\mathcal{M}(w)(x)$ halts:
  - M_{wx} always returns 1
  - $L(M_{wx}) = \Sigma^*$ (all strings)
  - $L(M_{wx})$ is infinite
- If $\mathcal{M}(w)(x)$ doesn't halt:
  - M_{wx} gets "stuck" in step 1 and never returns
  - $L(M_{wx}) = \emptyset$
  - $|L(M_{wx})| = 0$

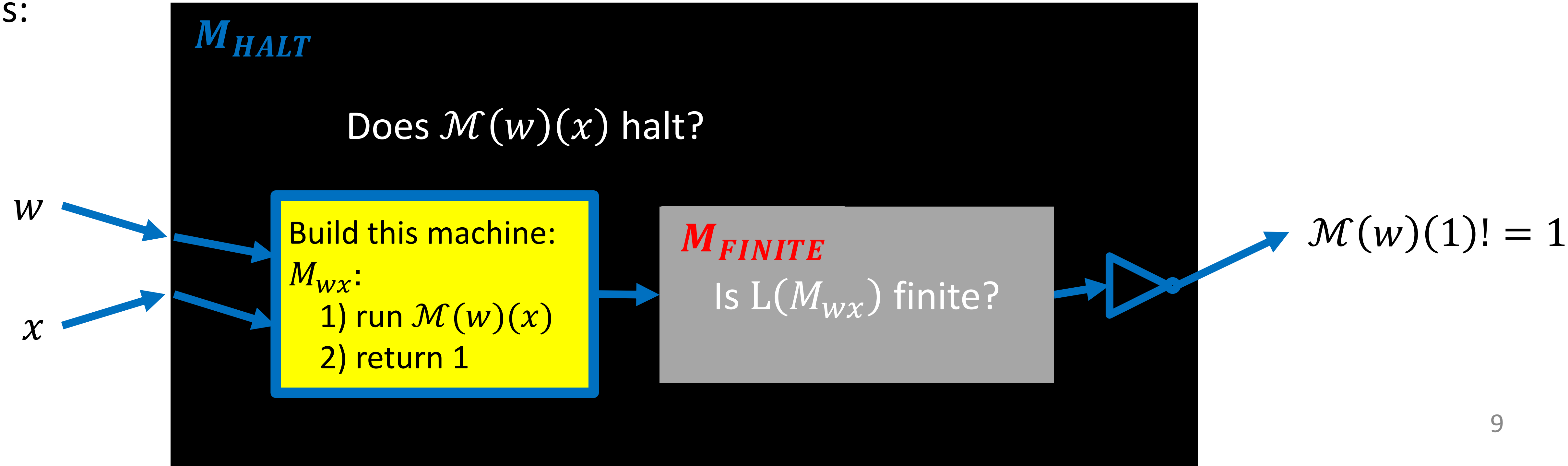
Build this machine:
M_{wx}:
1) run $\mathcal{M}(w)(x)$
2) return 1

Using $FINITE$ to build $HALT$

Assume we have M_{FINITE} which computes $FINITE$:



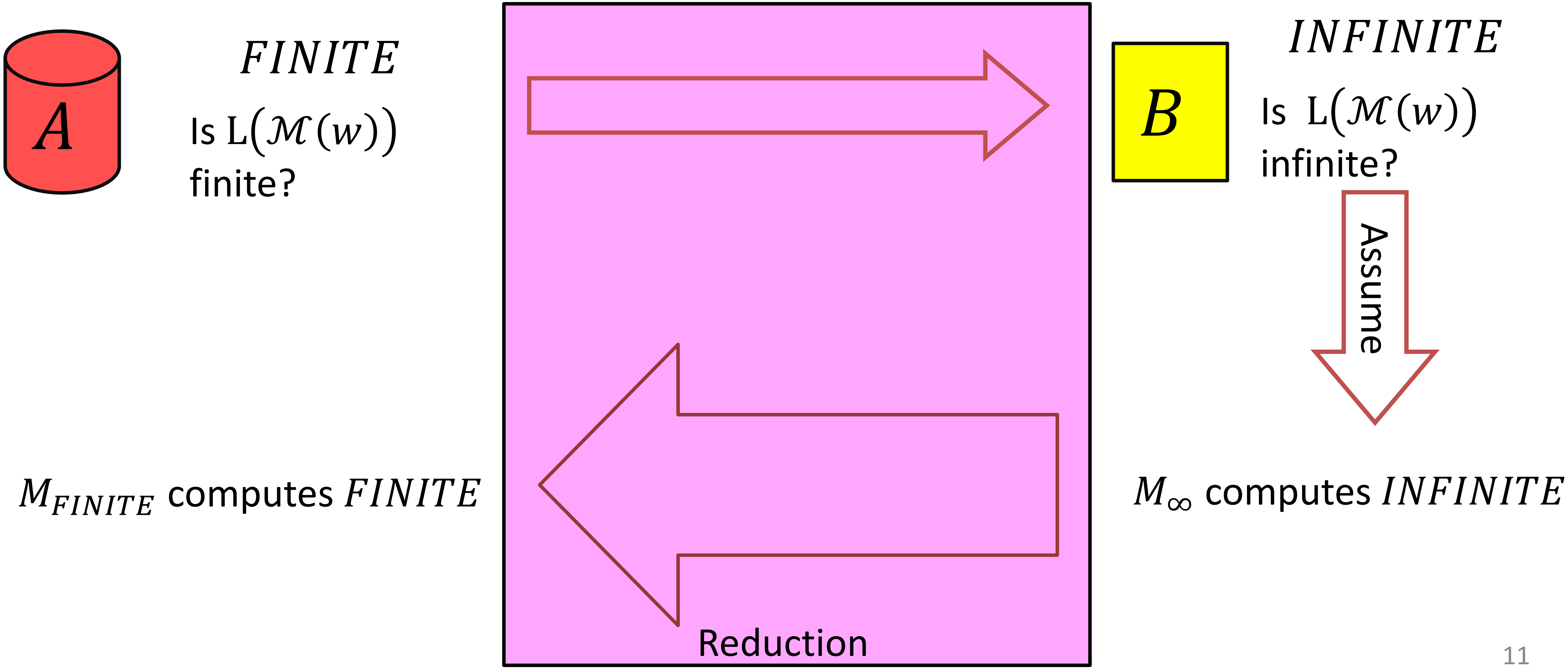
We could then build M_{HALT} which computes $HALT$ like this:



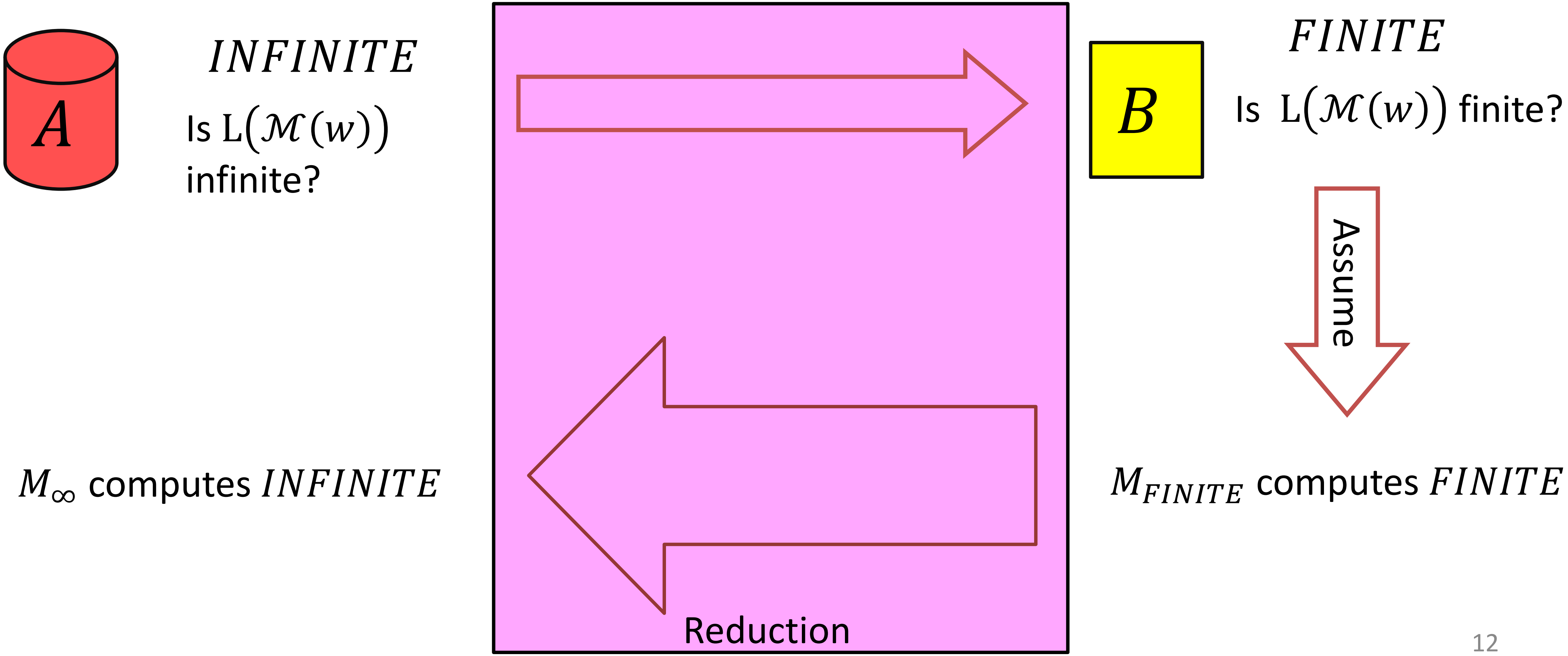
Showing $INFINITE$ is not computable

- Use ______ to build ______.

$INFINITE$ Reduction

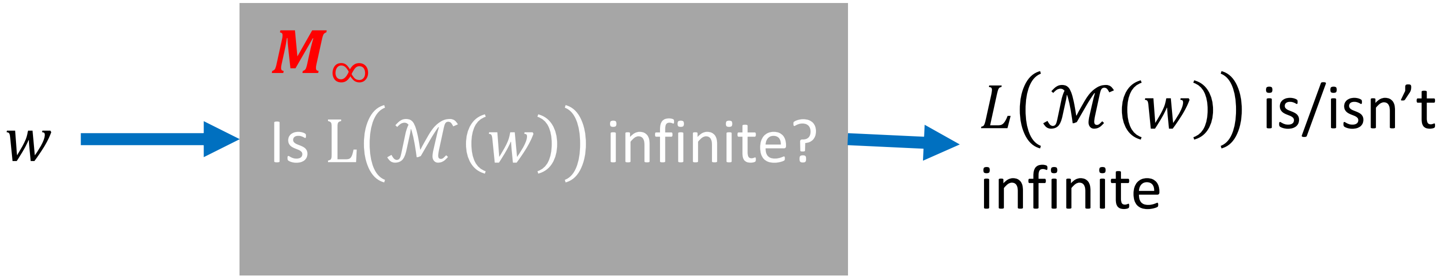


$INFINITE$ Reduction

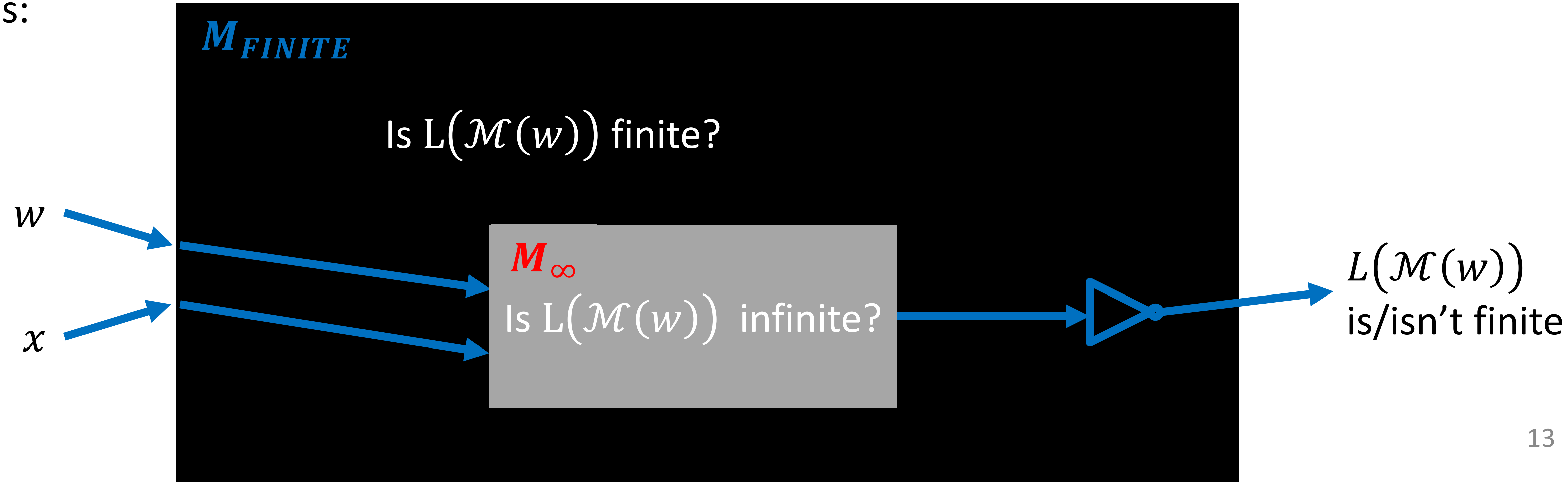


Using $INFINITE$ to build $FINITE$

Assume we have M_{∞} which computes $INFINITE$:



We could then build M_{FINITE} which computes $FINITE$ like this:



Language $NonReg$

- $NonReg = \{w \mid L(\mathcal{M}(w)) \text{ is not regular}\}$
- We will show this is not computable by using $NonReg$ to compute $HALT$
- Idea: given an input for $HALT$, w and x, build a machine M_{wx} such that $L(M_{wx})$ is regular if and only if $\mathcal{M}(w)(x)$ runs forever.

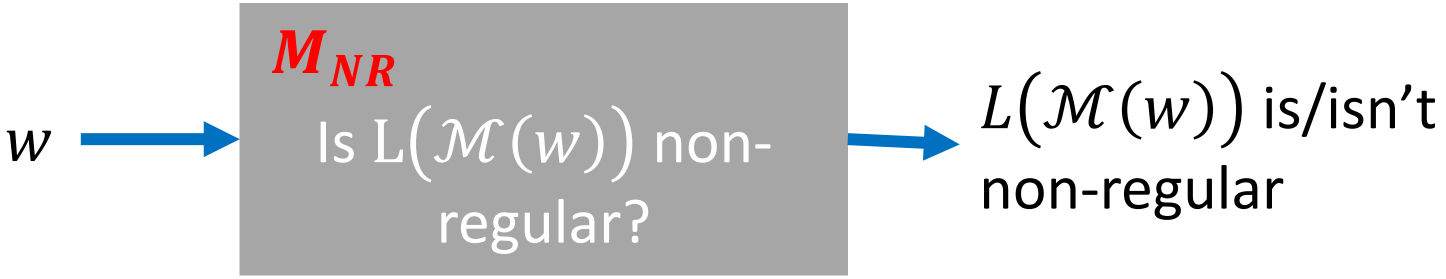
Building M_{wx}

Build this machine:
$M_{wx}(y)$:
1) run $\mathcal{M}(w)(x)$
2) return $XOR(y)$

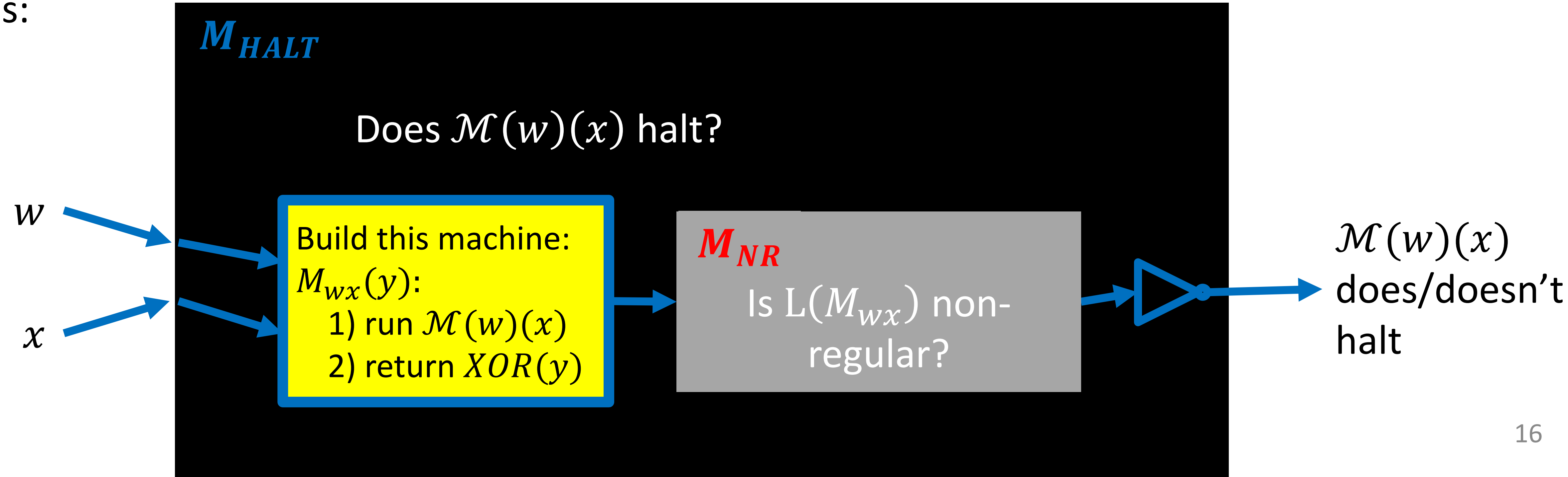
- If $\mathcal{M}(w)(x)$ halts:
  - M_{wx} returns 1 if $XOR(y) = 1$
  - $L(M_{wx}) = XOR(y)$, which is not regular
- If $\mathcal{M}(w)(x)$ doesn't halt:
  - M_{wx} gets "stuck" in step 2 and never returns 1
  - $L(M_{wx}) = \emptyset$, which is regular

Using $NonReg$ to build $HALT$

Assume we have M_{NR} which computes $NonReg$:



We could then build M_{HALT} which computes $HALT$ like this:



Language $Rejects101$

- $Rejects101 = \{w \mid \mathcal{M}(w)(101) = 0\}$
- We will show this is not computable by using $Rejects101$ to compute $HALT$
- Idea: given an input for $HALT$, w and x, build a machine M_{wx} such that $101 \in L(m_{wx})$ if and only if $\mathcal{M}(w)(x)$ runs forever.

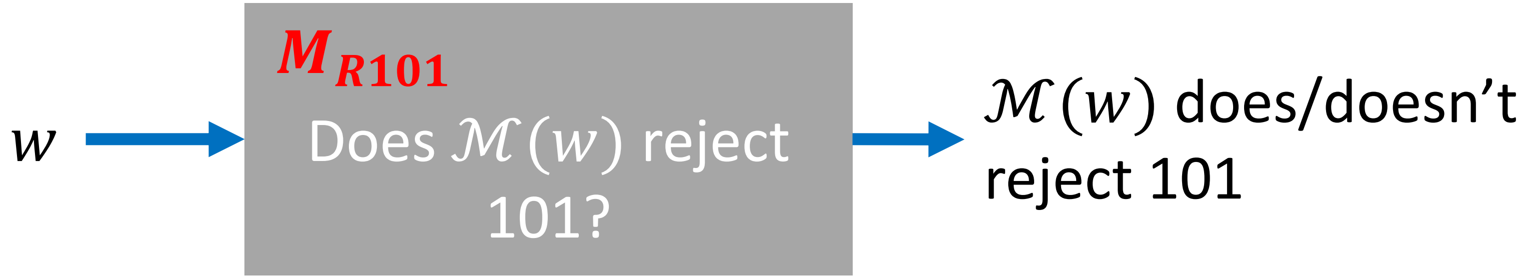
Building M_{wx}

Build this machine:
$M_{wx}(y)$:
1) run $\mathcal{M}(w)(x)$
2) return $y == 101$

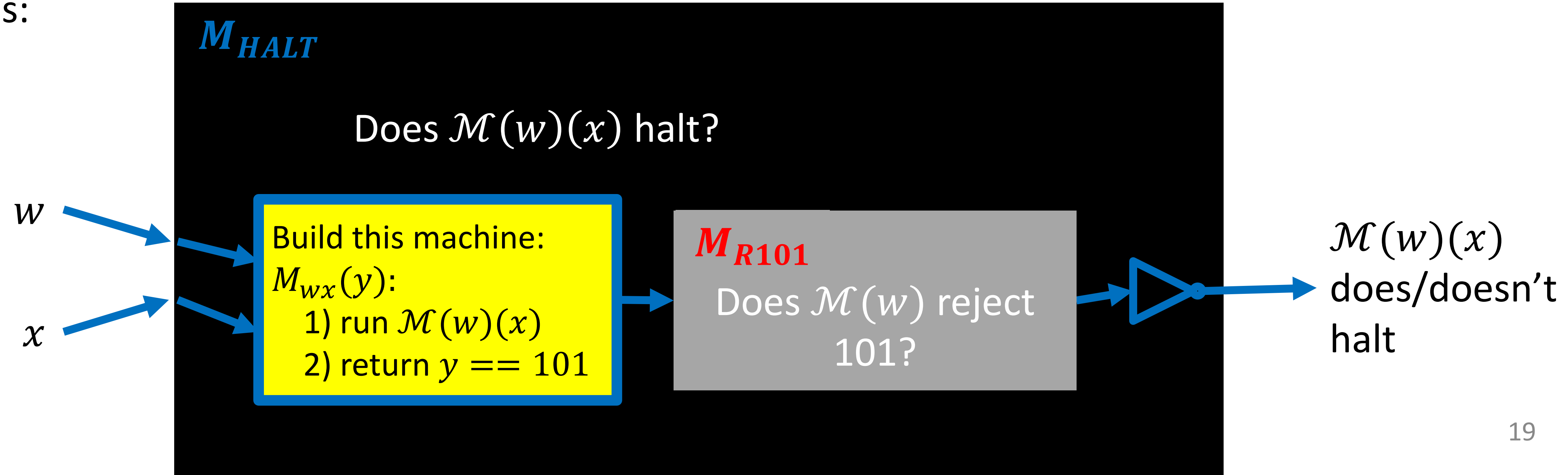
- If $\mathcal{M}(w)(x)$ halts:
  - M_{wx} returns 1 if $y == 101$
  - $L(M_{wx}) = \{101\}$, so it does not reject 101
- If $\mathcal{M}(w)(x)$ doesn't halt:
  - M_{wx} gets "stuck" in step 2 and never returns 1
  - $L(M_{wx}) = \emptyset$, so it does reject 101

Using $Rejects101$ to build $HALT$

Assume we have M_{R101} which computes $NonReg$:



We could then build M_{HALT} which computes $HALT$ like this:



Sematic Property

- Turing machines M, M' are **Functionally Equivalent** if $\forall x \in \Sigma^*, M(x) == M(x')$
  – i.e. they compute the same function/language
- A **Semantic Property** of a Turing machine is one that depends only on the input/output behavior of the machine
  – Formally, if P is semantic, then for machine M, M' that are functionally equivalent, $P(M) == P(M')$
  – If M, M' have the same input/output behavior, and P is a semantic property, then either bother M and M' have property P, or neither of them do.

Examples

- These properties are Semantic:
  - Is the language of this machine finite?
  - Is the language of this machine Regular?
  - Does this machine reject 101?
  - Does this machine return 1001 for input 001?
  - Does this machine only ever return odd numbers?
  - Is the language of this machine computable?
- These properties are not Semantic:
  - Does this machine ever overwrite cell 204 of its tape?
  - Does this machine use more than 3102 cells of its tape on input 101?
  - Does this machine take at least 2020 transitions for input ε?
  - Does this machine ever overwrite the ∇ symbol?

Rice's Theorem

- For any Semantic property P of Turing Machines, either:
  - Every Turing machine has property P
  - No Turing machines have property P
  - P is uncomputable

P is "trivial"

- In other words:
  - If P is semantic, and computable, then one of these two machines computes it:

Return 1

Return 0

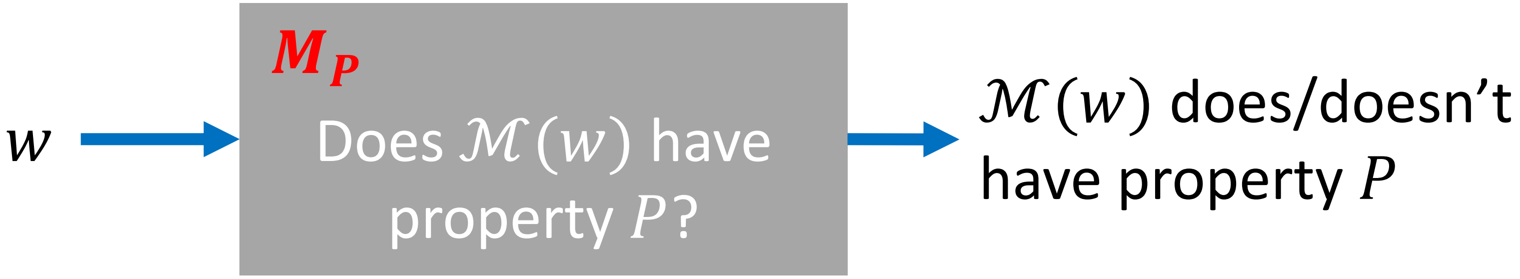
Proof of Rice's Theorem

- Let P be a semantic property of a Turing machine
- Assume $M_\emptyset$ (a machine whose language is $\emptyset$) has property P (otherwise substitute $\neg P$, then answer opposite)
- Let M_{FALSE} be a machine that doesn't have property P
- Idea:
  - If $\mathcal{M}(w)(x)$ halts, $L(M_{wx}) = L(M_{FALSE})$
  - If $\mathcal{M}(w)(x)$ doesn't halt, $L(M_{wx}) = \mathrm{L}(M_\emptyset) = \emptyset$
  - $L(M_{wx})$ has property P if and only if $\mathcal{M}(w)(x)$ runs forever

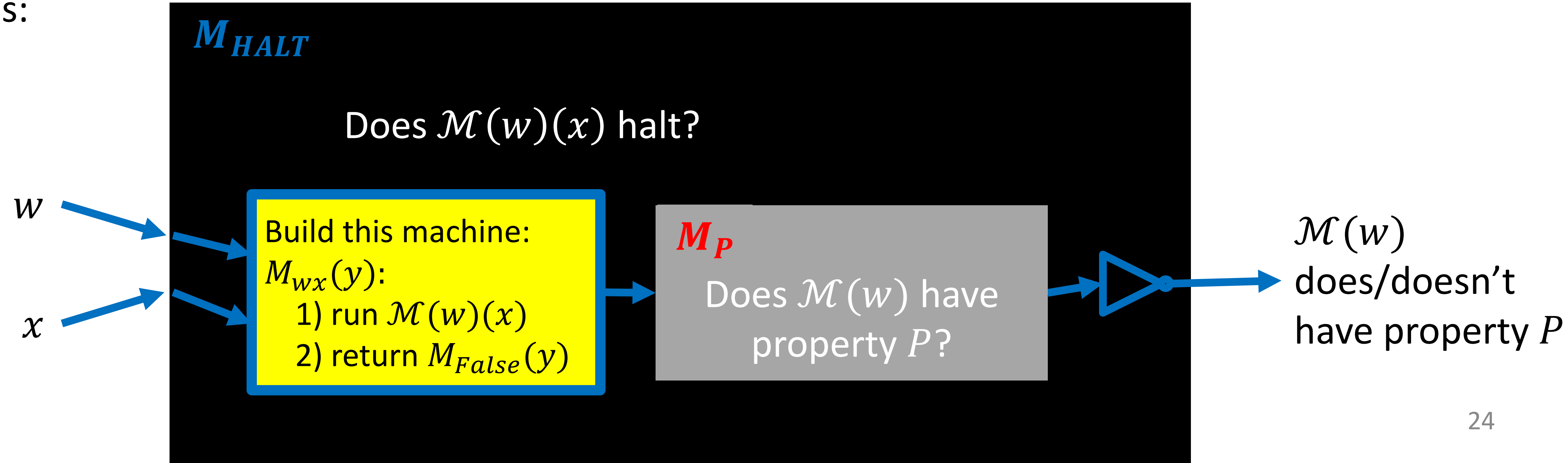
Build this machine:
$M_{wx}(y)$:
1) run $\mathcal{M}(w)(x)$
2) return $M_{False}(y)$

Using P to build $HALT$

Assume we have M_P which computes P:



We could then build M_{HALT} which computes $HALT$ like this:



What if P is "trivial"?

- If P applies to no Turing machines:
  - M_{False} can't exist
- If P applies to all Turing machines:
  - It applies to $M_\emptyset$, $\neg P$ applies to all machines

Build this machine:
$M_{wx}(y)$:
1) run $\mathcal{M}(w)(x)$
2) return $M_{False}(y)$

Using Rice's Theorem

- These properties are Semantic:
  - Is the language of this machine finite?
  - Is the language of this machine Regular?
  - Does this machine reject 101?
  - Does this machine return 1001 for input 001?
  - Does this machine only ever return odd numbers?
  - Is the language of this machine computable?
- These properties are not Semantic:
  - Does this machine ever overwrite cell 204 of its tape?
  - Does this machine use more than 3102 cells of its tape on input 101?
  - Does this machine take at least 2020 transitions for input ε?
  - Does this machine ever overwrite the ∇ symbol on input ε?

$Steps2020$

- $Steps2020 = \{w | \mathcal{M}(w) \text{ takes at least 2020 steps}\}$
- Is $Steps2020$ computable?

$Overwrite\nabla$

- $Overwrite\nabla = \{\text{w} \mid \mathcal{M}(w) \text{ overwrites } \nabla \text{ on input } \varepsilon\}$
- Is $Overwrite\nabla$ computable?