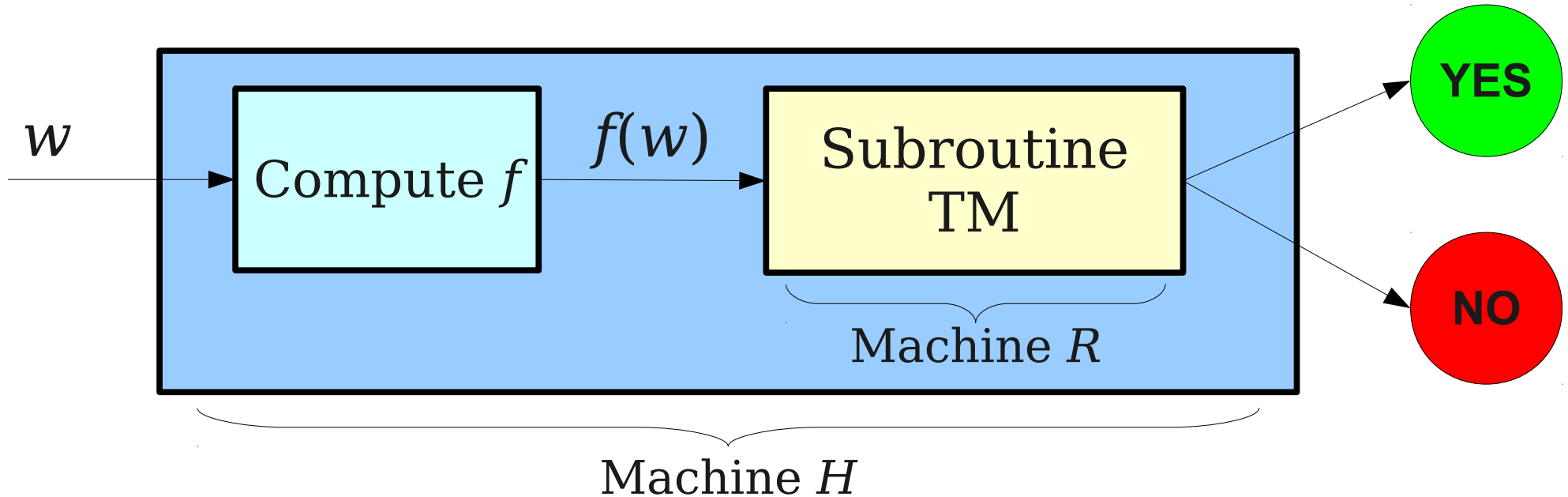


Reducibility

Part II

Problem set 7
due in the box
up front.

The General Pattern



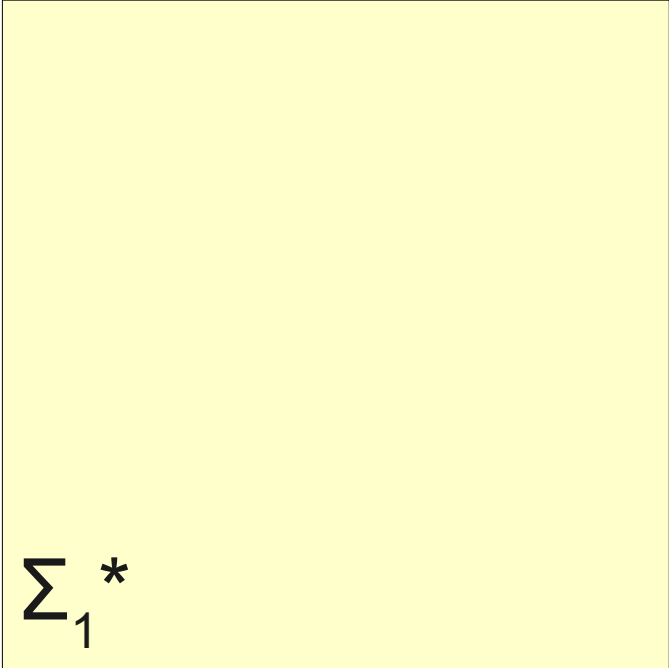
$H =$ "On input w :

- Transform the input w into $f(w)$.
- Run machine R on $f(w)$.
- If R accepts $f(w)$, then H accepts w .
- If R rejects $f(w)$, then H rejects w ."

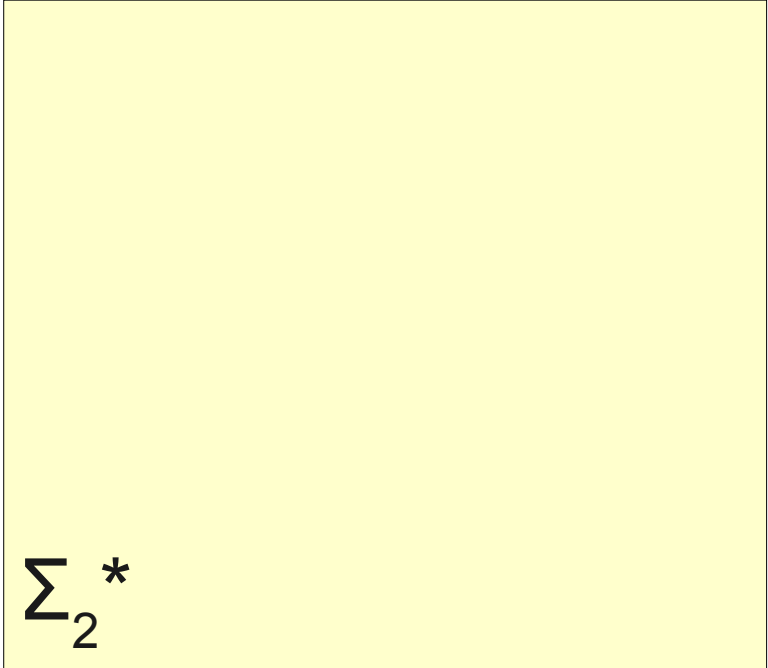
Defining Reductions

- A **reduction** from A to B is a function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ such that

For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$



Σ_1^*

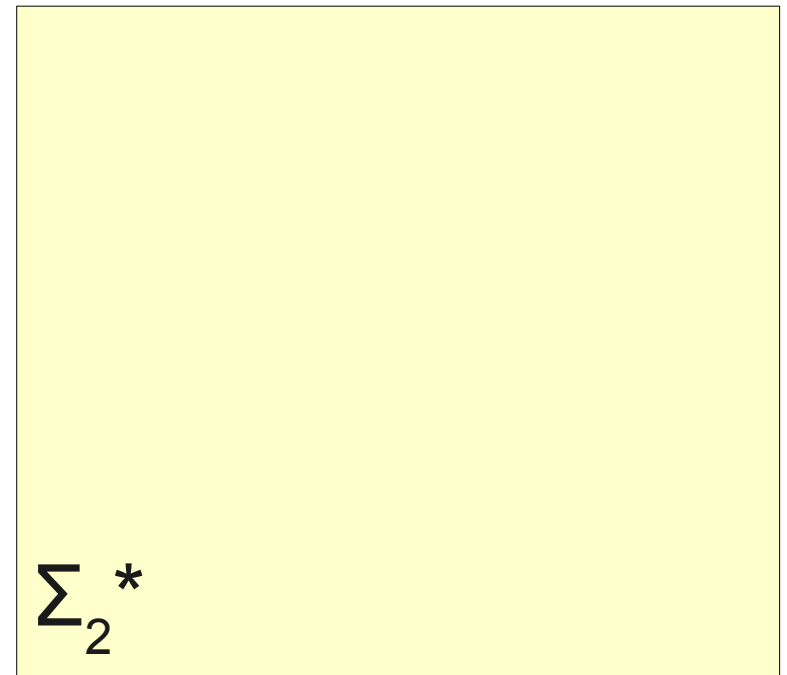
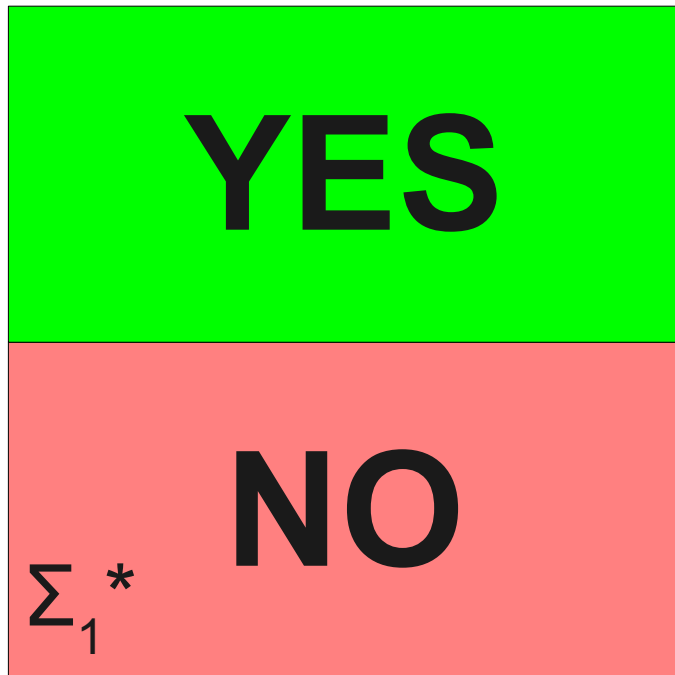


Σ_2^*

Defining Reductions

- A **reduction** from A to B is a function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ such that

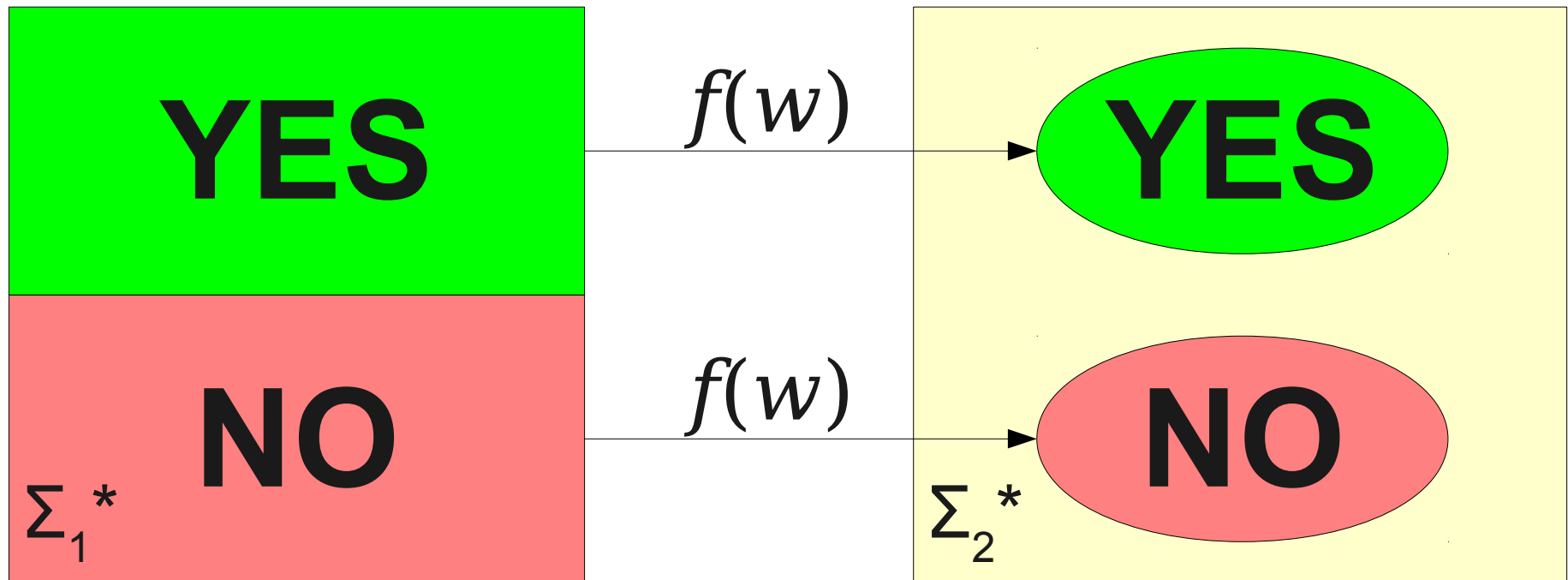
For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$



Defining Reductions

- A **reduction** from A to B is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$



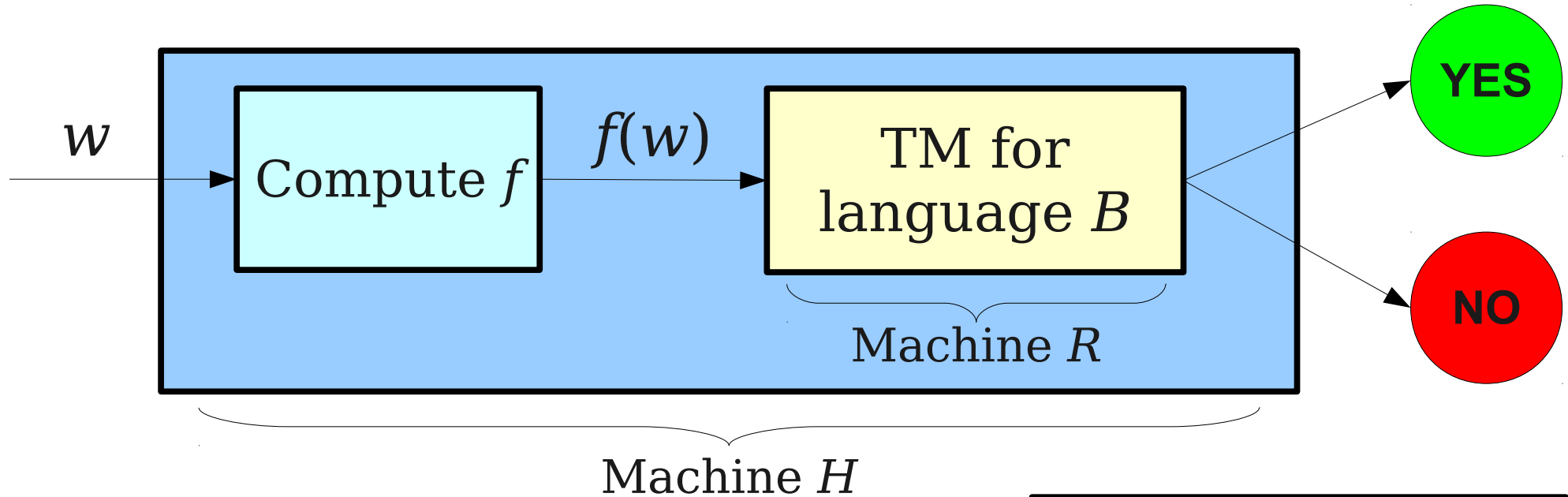
Defining Reductions

- A **reduction** from A to B is a function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ such that

For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$

- Every $w \in A$ maps to some $f(w) \in B$.
- Every $w \notin A$ maps to some $f(w) \notin B$.
- f does not have to be injective or surjective.

$w \in A$ iff $f(w) \in B$



$H =$ "On input w :

- Transform the input w into $f(w)$.
- Run machine R on $f(w)$.
- If R accepts $f(w)$, then H accepts w .
- If R rejects $f(w)$, then H rejects w ."

H accepts w

iff

R accepts $f(w)$

iff

$f(w) \in B$

iff

$w \in A$

Mapping Reductions

- A function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is called a **mapping reduction** from A to B iff
 - For any $w \in \Sigma_1^*$, $w \in A$ iff $f(w) \in B$.
 - f is a computable function.
- Intuitively, a mapping reduction from A to B says that a computer can transform any instance of A into an instance of B such that the answer to B is the answer to A .

Mapping Reducibility

- If there is a mapping reduction from language A to language B , we say that language A is **mapping reducible** to language B .
- Notation: $A \leq_M B$ iff language A is mapping reducible to language B .
- Note that we reduce *languages*, not *machines*.

Why Mapping Reducibility Matters

- **Theorem:** If $B \in \mathbf{R}$ and $A \leq_M B$, then $A \in \mathbf{R}$.
- **Theorem:** If $B \in \mathbf{RE}$ and $A \leq_M B$, then $A \in \mathbf{RE}$.
- **Theorem:** If $B \in \mathbf{co-RE}$ and $A \leq_M B$, then $A \in \mathbf{co-RE}$.
- *Intuitively:* $A \leq_M B$ means “A is not harder than B.”

Why Mapping Reducibility Matters

- **Theorem:** If $A \notin \mathbf{R}$ and $A \leq_M B$, then $B \notin \mathbf{R}$.
- **Theorem:** If $A \notin \mathbf{RE}$ and $A \leq_M B$, then $B \notin \mathbf{RE}$.
- **Theorem:** If $A \notin \mathbf{co-RE}$ and $A \leq_M B$, then $B \notin \mathbf{co-RE}$.
- *Intuitively:* $A \leq_M B$ means “ B is at least as hard as A .”

Why Mapping Reducibility Matters

If this one is "easy"
(R, RE, co-RE)...

$$A \leq_M B$$

... then this one is
"easy" (R, RE,
co-RE) too.

Why Mapping Reducibility Matters

If this one is "hard"
(not R , not RE , or not
 $co-RE$)...

$$A \leq_M B$$

... then this one is
"hard" (not R , not
 RE , or not $co-RE$)
too.

Using Mapping Reductions

Revisiting our Proofs

- Consider the language

$$L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \}$$

- We have already proven that this language is in **RE** by building a TM for it.
- Let's repeat this proof using mapping reductions.
- Specifically, we will prove

$$L \leq_M A_{\text{TM}}$$

$$L = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } \varepsilon \}$$

- To prove $L \leq_M A_{\text{TM}}$, we will need to find a computable function f such that

$$\langle M \rangle \in L \quad \text{iff} \quad f(\langle M \rangle) \in A_{\text{TM}}$$

- Since A_{TM} is a language of TM/string pairs, let's assume $f(\langle M \rangle) = \langle N, w \rangle$ for some TM N and string w (which we'll pick later):

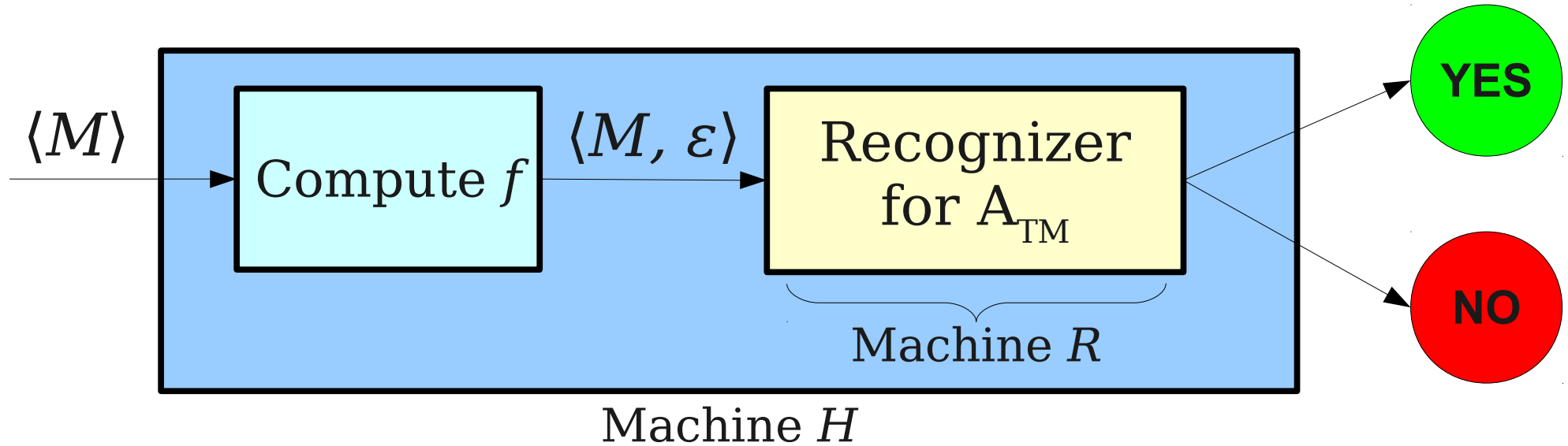
$$\langle M \rangle \in L \quad \text{iff} \quad \langle N, w \rangle \in A_{\text{TM}}$$

- Substituting definitions:

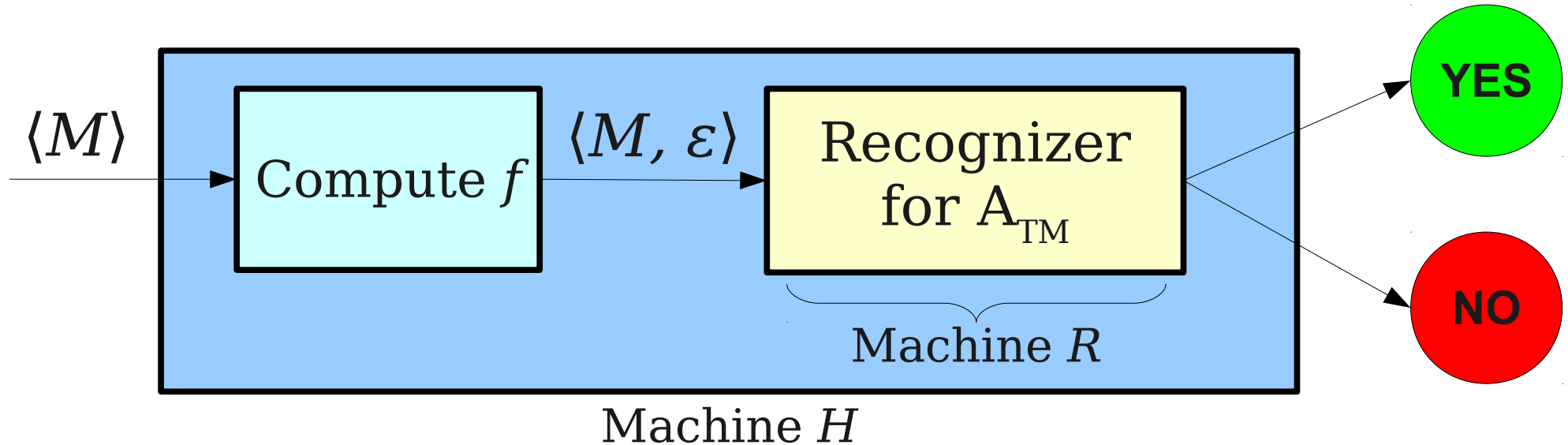
$$M \text{ accepts } \varepsilon \quad \text{iff} \quad N \text{ accepts } w$$

- Choose $N = M$, $w = \varepsilon$. So $f(\langle M \rangle) = \langle M, \varepsilon \rangle$.

One Interpretation of the Reduction



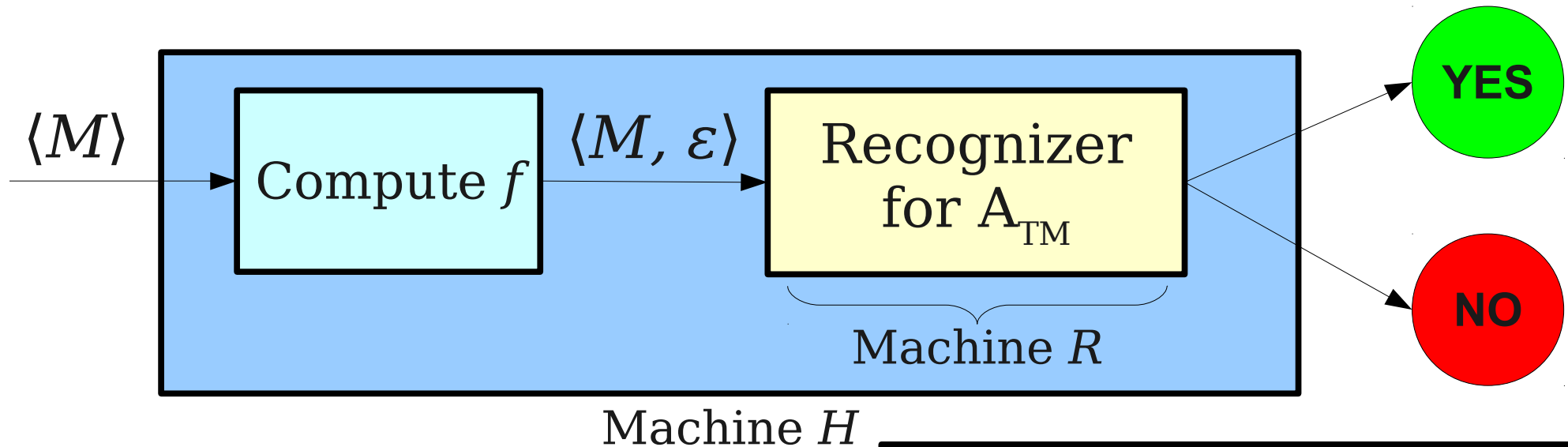
One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

One Interpretation of the Reduction

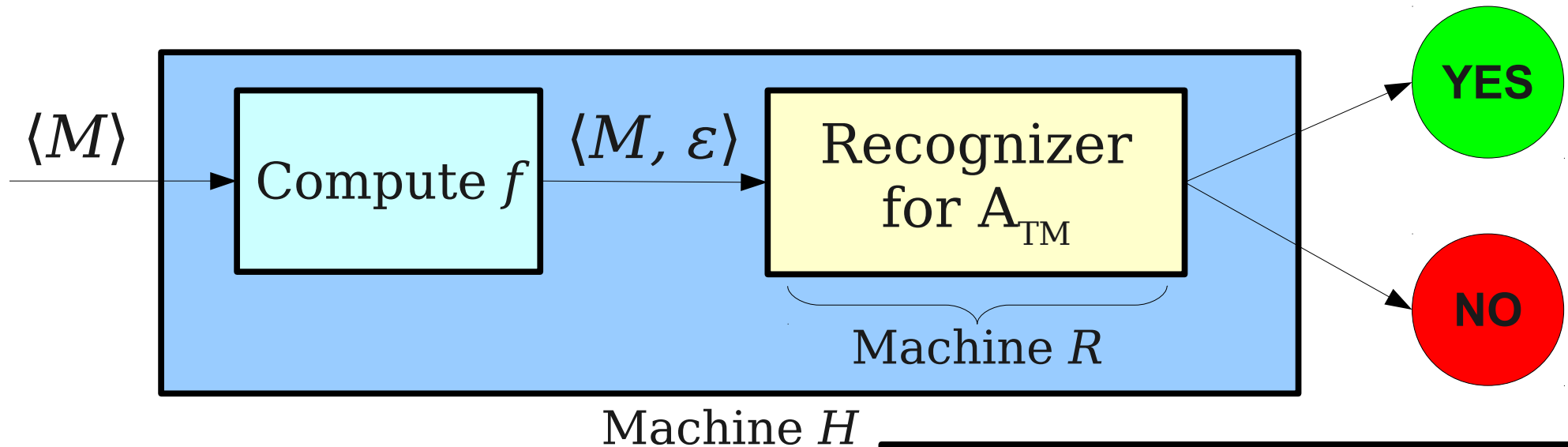


$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

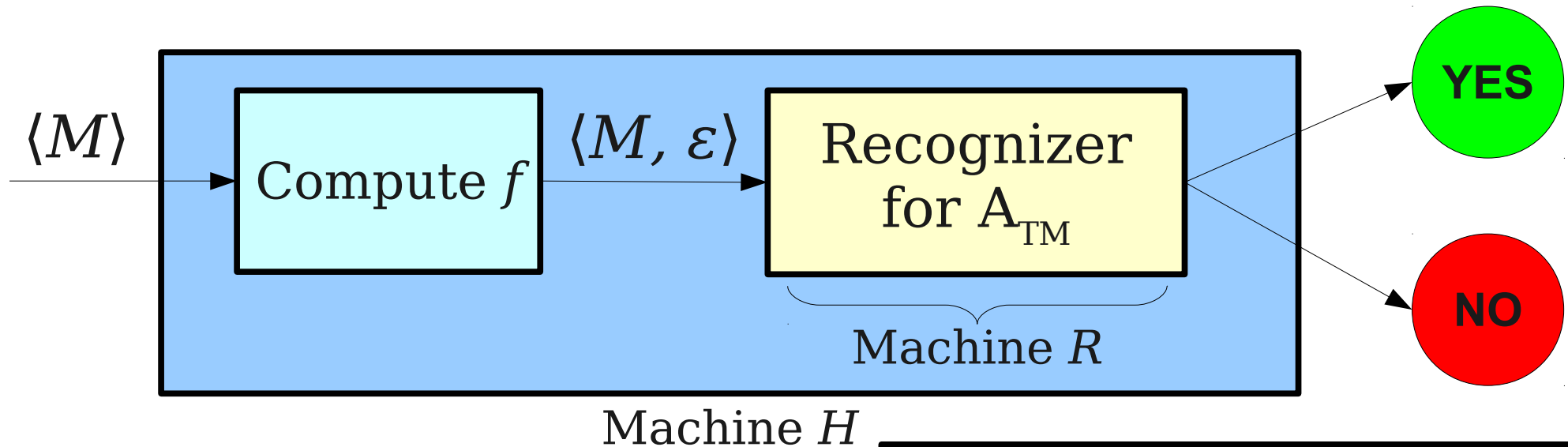
- Run machine R on $\langle M, \epsilon \rangle$.
- If R accepts $\langle M, \epsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \epsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \epsilon \rangle$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

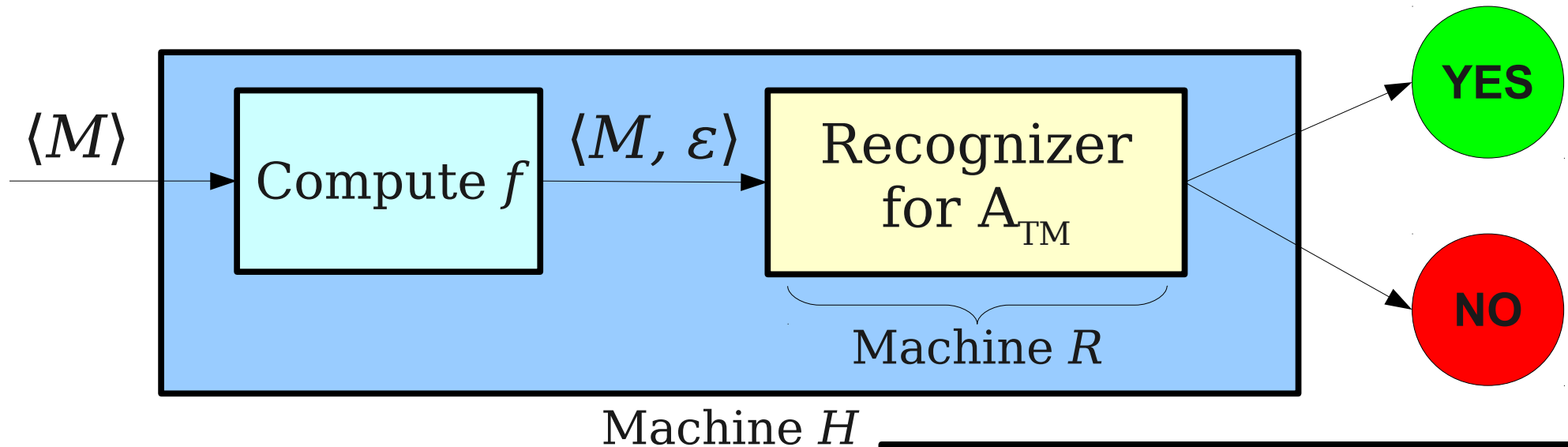
iff

R accepts $\langle M, \varepsilon \rangle$

iff

M accepts ε

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \varepsilon \rangle$

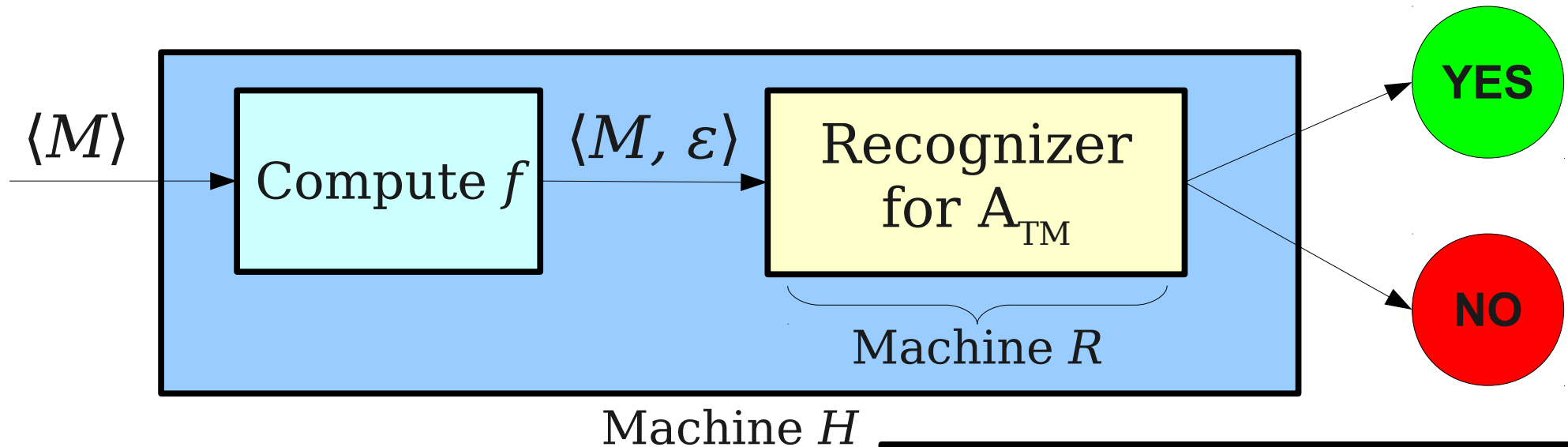
iff

M accepts ε

iff

$\langle M \rangle \in L$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \varepsilon \rangle$

iff

M accepts ε

iff

$\langle M \rangle \in L$

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} .

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. To do this, consider any TM M .

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

To do this, consider any TM M . Note that by the definition of L , we see $\langle M \rangle \in L$ iff M accepts ε .

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

To do this, consider any TM M . Note that by the definition of L , we see $\langle M \rangle \in L$ iff M accepts ε . By the definition of A_{TM} , we know that M accepts ε iff

$$\langle M, \varepsilon \rangle \in A_{\text{TM}}.$$

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

To do this, consider any TM M . Note that by the definition of L , we see $\langle M \rangle \in L$ iff M accepts ε . By the definition of A_{TM} , we know that M accepts ε iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. Combining these statements together, we have that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

To do this, consider any TM M . Note that by the definition of L , we see $\langle M \rangle \in L$ iff M accepts ε . By the definition of A_{TM} , we know that M accepts ε iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. Combining these statements together, we have that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

This means that f is a mapping reduction from L to A_{TM} , so $L \leq_M A_{\text{TM}}$, as required.

$$L = \{ \langle M \rangle \mid M \text{ is a TM that accepts } \varepsilon \}$$

Theorem: $L \in \mathbf{RE}$.

Proof: We will prove that $L \leq_M A_{\text{TM}}$. Since $A_{\text{TM}} \in \mathbf{RE}$, this proves $L \in \mathbf{RE}$ as well.

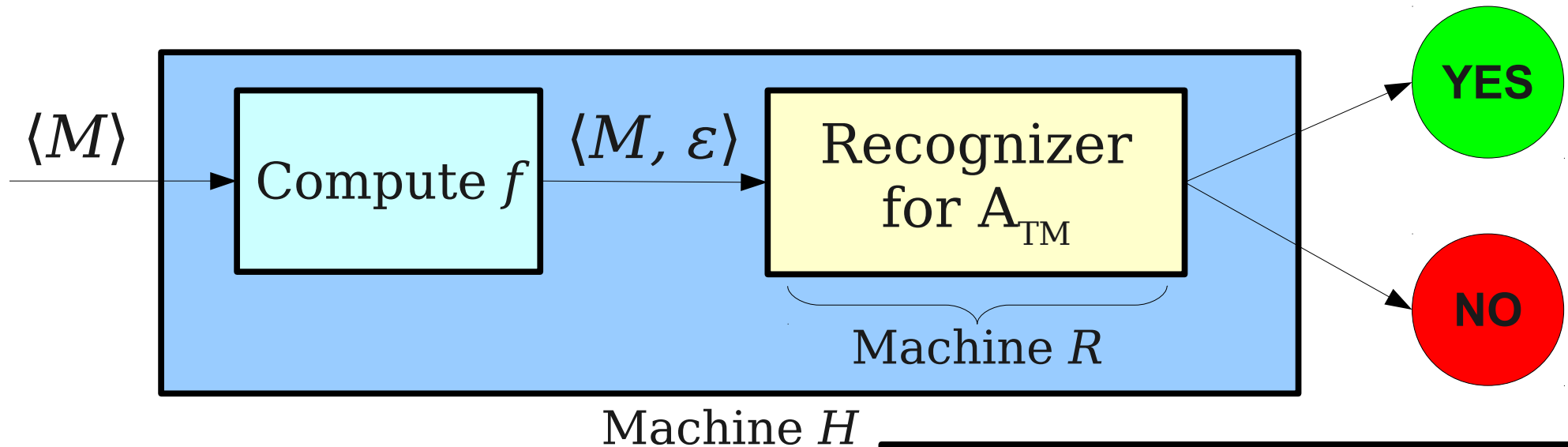
To prove this, we will give a mapping reduction from L to A_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \varepsilon \rangle$. This function can be computed by a Turing machine.

Now, we will prove that f is a mapping reduction by proving for all TMs M that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

To do this, consider any TM M . Note that by the definition of L , we see $\langle M \rangle \in L$ iff M accepts ε . By the definition of A_{TM} , we know that M accepts ε iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$. Combining these statements together, we have that $\langle M \rangle \in L$ iff $\langle M, \varepsilon \rangle \in A_{\text{TM}}$.

This means that f is a mapping reduction from L to A_{TM} , so $L \leq_M A_{\text{TM}}$, as required. ■

What Did We Prove?



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \epsilon \rangle$.
- If R accepts $\langle M, \epsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \epsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \epsilon \rangle$

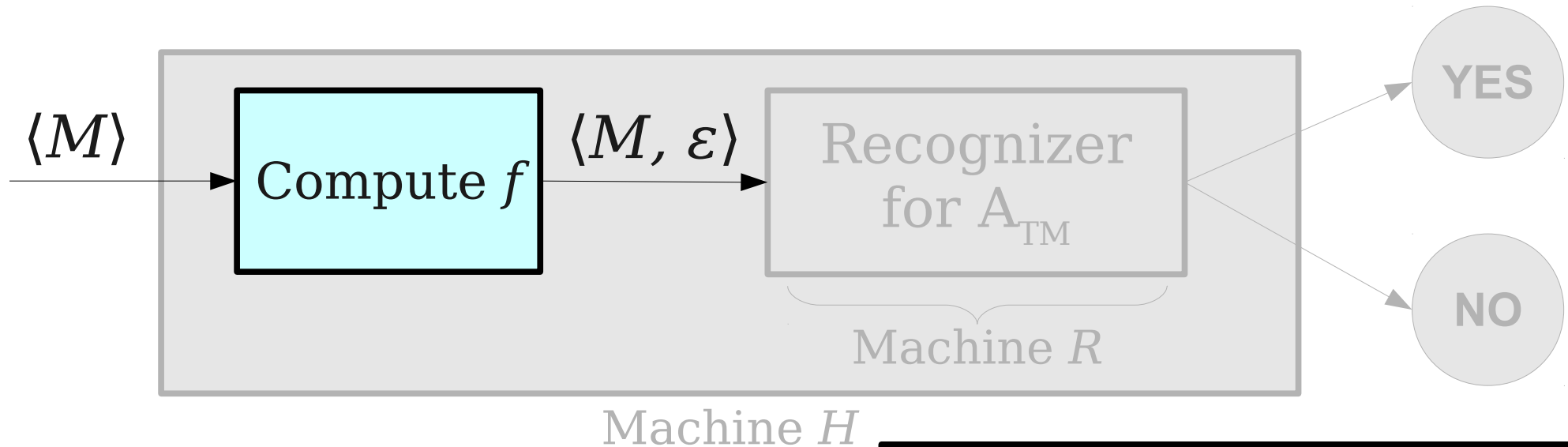
iff

M accepts ϵ

iff

$\langle M \rangle \in L$

What Did We Prove?



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \varepsilon \rangle$.
- If R accepts $\langle M, \varepsilon \rangle$, then H accepts w .
- If R rejects $\langle M, \varepsilon \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \varepsilon \rangle$

iff

M accepts ε

iff

$\langle M \rangle \in L$

Interpreting Mapping Reductions

- If $A \leq_M B$, there is a known construction to turn a TM for B into a TM for A .
- When doing proofs with mapping reductions, you do *not* need to show the overall construction.
- You just need to prove that
 - f is a computable function, and
 - $w \in A$ iff $f(w) \in B$.

Another Mapping Reduction

L_D and \overline{A}_{TM}

- Earlier, we proved $\overline{A}_{TM} \notin \mathbf{RE}$ by proving that

If $\overline{A}_{TM} \in \mathbf{RE}$, then $L_D \in \mathbf{RE}$.

- The proof constructed this TM, assuming R was a recognizer for \overline{A}_{TM} .

$H =$ “On input $\langle M \rangle$:

- Construct the string $\langle M, \langle M \rangle \rangle$.
- Run R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts $\langle M \rangle$.
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects $\langle M \rangle$.”

- Let's do another proof using mapping reductions.

$$L_D \leq_M \overline{A}_{TM}$$

- To prove that $\overline{A}_{TM} \notin \mathbf{RE}$, we will prove

$$L_D \leq_M \overline{A}_{TM}$$

- By our earlier theorem, since $L_D \notin \mathbf{RE}$, we have that $\overline{A}_{TM} \notin \mathbf{RE}$.
- *Intuitively:* \overline{A}_{TM} is “at least as hard” as L_D , and since $L_D \notin \mathbf{RE}$, this means $\overline{A}_{TM} \notin \mathbf{RE}$.

$$L_D \leq_M \overline{A}_{TM}$$

- Goal: Find a computable function f such that

$$\langle M \rangle \in L_D \quad \text{iff} \quad f(\langle M \rangle) \in \overline{A}_{TM}$$

- Simplifying this using the definition of L_D

$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad f(\langle M \rangle) \in \overline{A}_{TM}$$

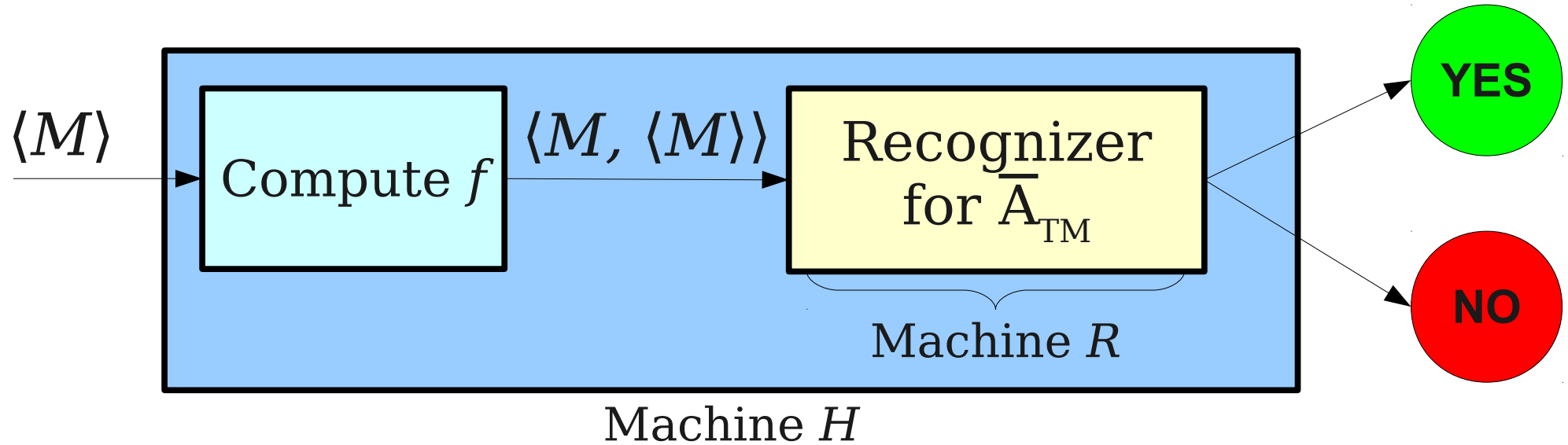
- Let's assume that $f(\langle M \rangle)$ has the form $\langle N, w \rangle$ for some TM N and string w . This means that

$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad \langle N, w \rangle \in \overline{A}_{TM}$$

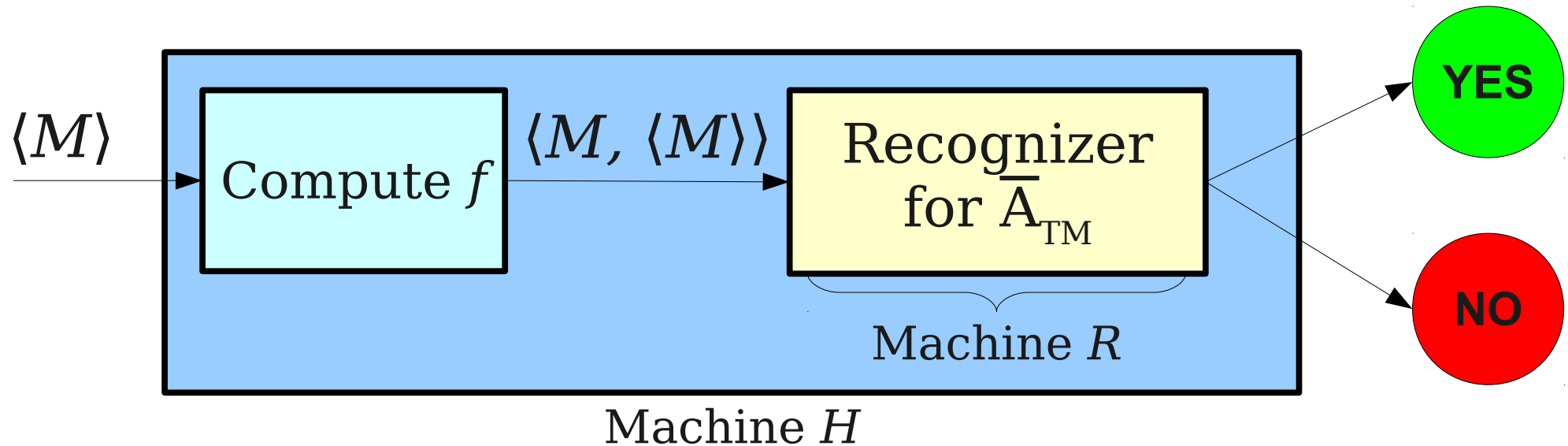
$$M \text{ does not accept } \langle M \rangle \quad \text{iff} \quad N \text{ does not accept } w$$

- If we can choose w and N such that the above is true, we will have our reduction from L_D to \overline{A}_{TM} .
- Choose $N = M$ and $w = \langle M \rangle$.

One Interpretation of the Reduction



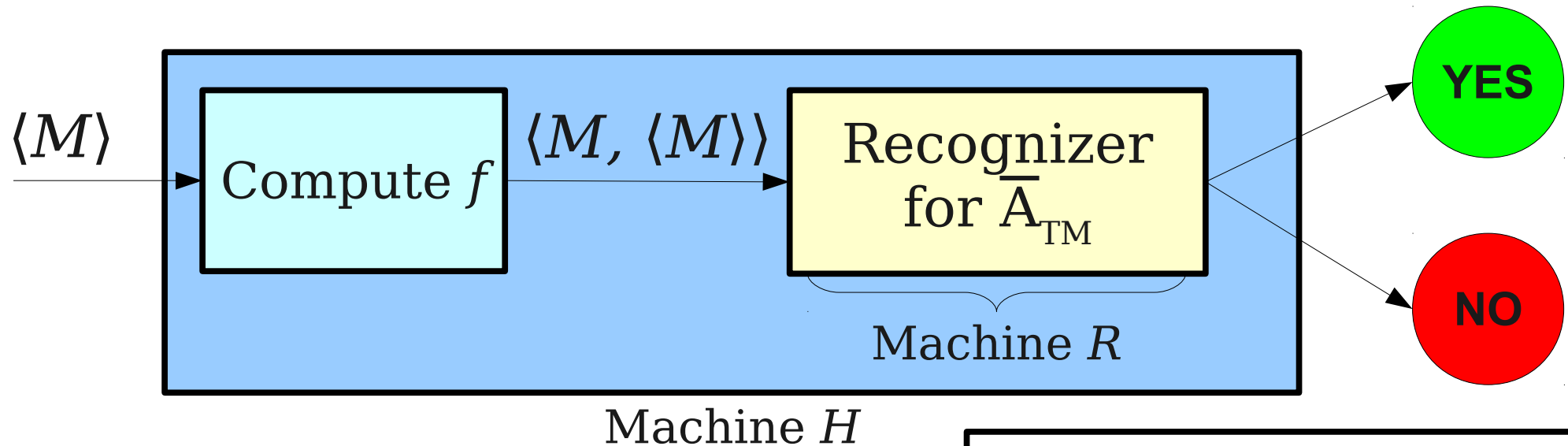
One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

One Interpretation of the Reduction

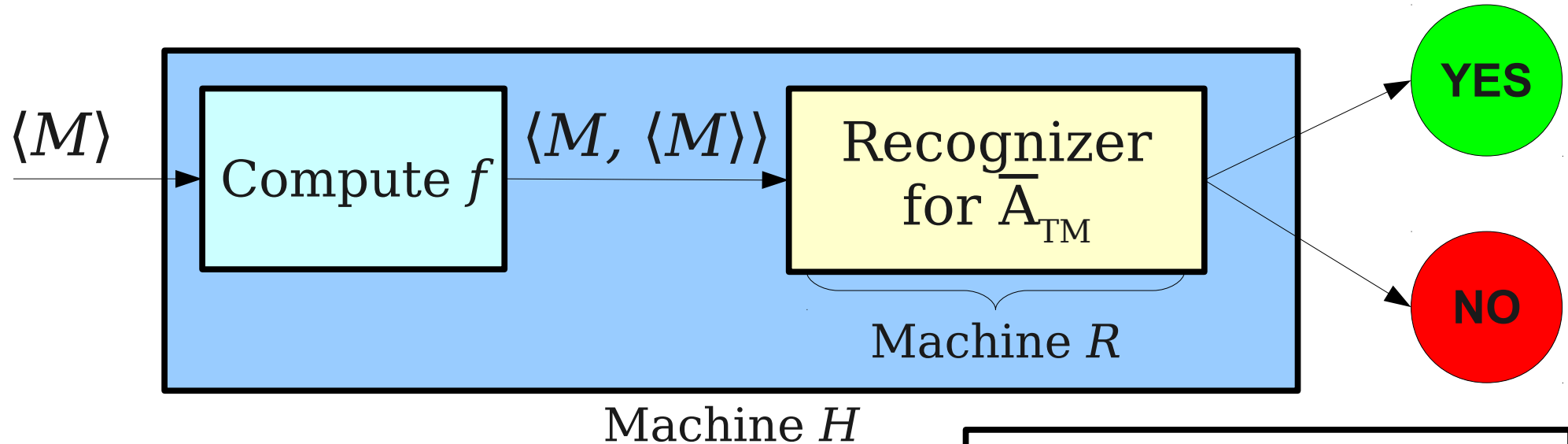


$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

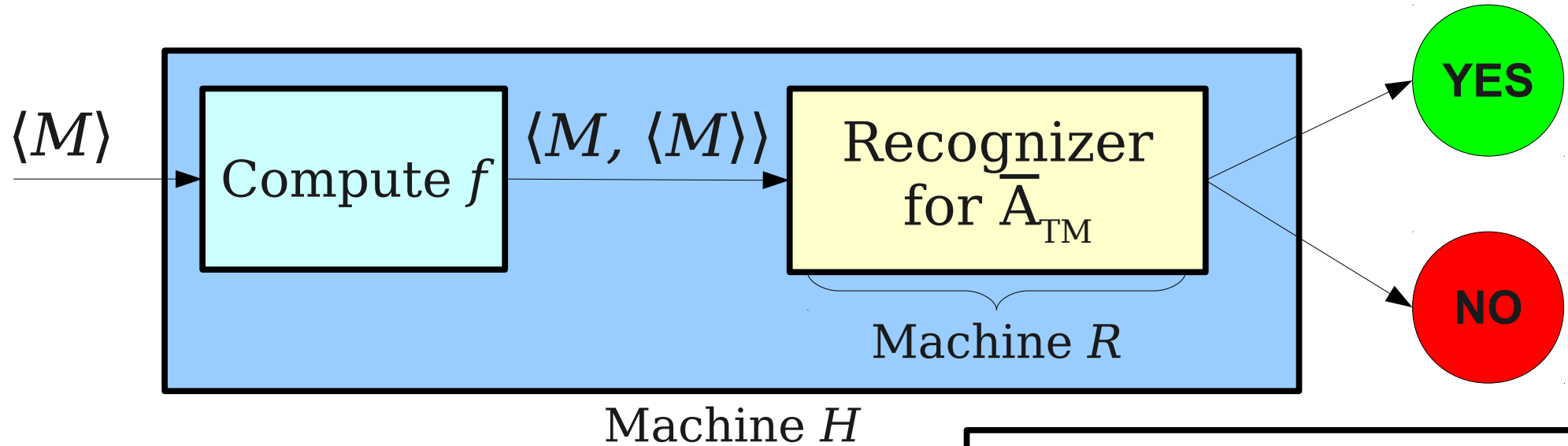
- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \langle M \rangle \rangle$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

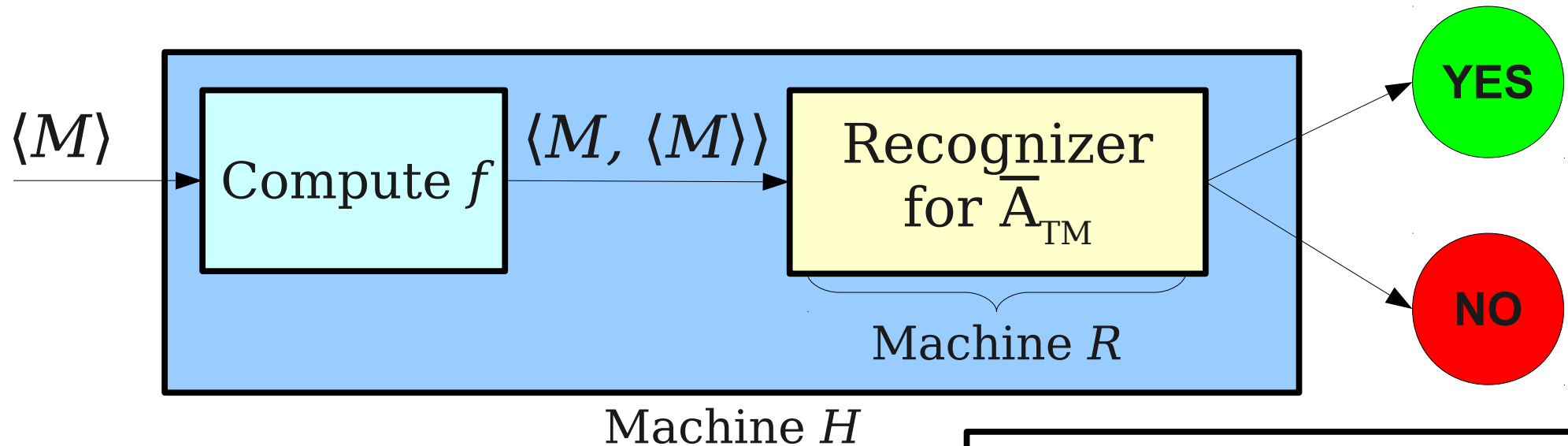
iff

R accepts $\langle M, \langle M \rangle \rangle$

iff

M does not accept $\langle M \rangle$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \langle M \rangle \rangle$

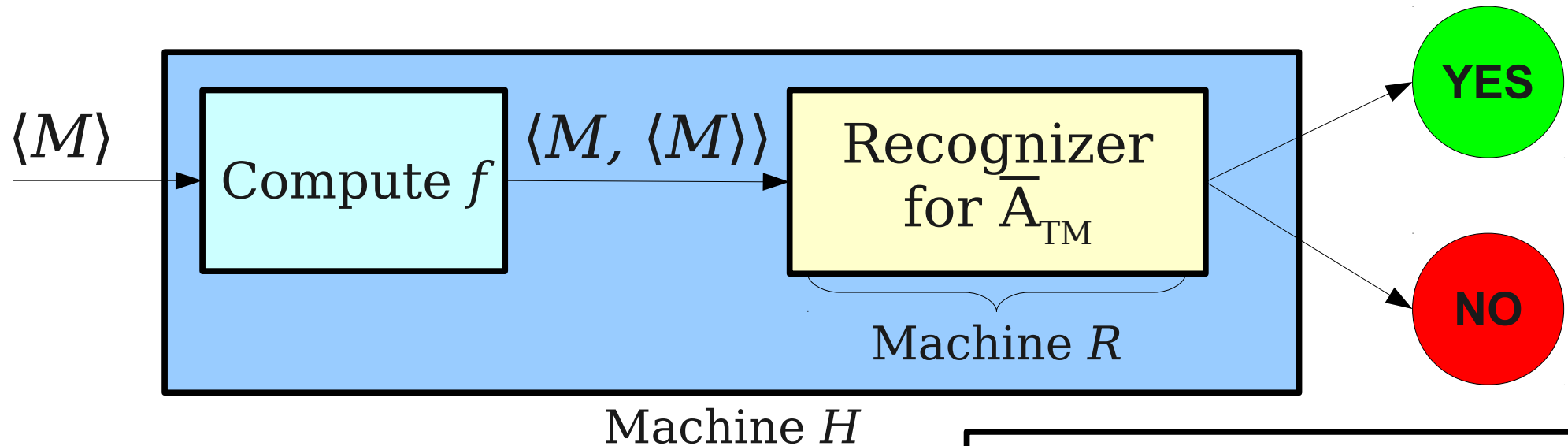
iff

M does not accept $\langle M \rangle$

iff

$\langle M \rangle \in L_D$

One Interpretation of the Reduction



$H =$ "On input $\langle M \rangle$:

- Run machine R on $\langle M, \langle M \rangle \rangle$.
- If R accepts $\langle M, \langle M \rangle \rangle$, then H accepts w .
- If R rejects $\langle M, \langle M \rangle \rangle$, then H rejects w ."

H accepts $\langle M \rangle$

iff

R accepts $\langle M, \langle M \rangle \rangle$

iff

M does not accept $\langle M \rangle$

iff

$\langle M \rangle \in L_D$

Theorem: $\overline{A}_{\text{TM}} \notin \mathbf{RE}$.

Proof: We will prove that $L_D \leq_M \overline{A}_{\text{TM}}$. Since $L_D \notin \mathbf{RE}$, this proves that $\overline{A}_{\text{TM}} \notin \mathbf{RE}$.

To show that $L_D \leq_M \overline{A}_{\text{TM}}$, we will give a mapping reduction from L_D to \overline{A}_{TM} . For any TM M , let $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$. This function f is computable.

To prove that f is a mapping reduction from L_D to \overline{A}_{TM} , we will prove for all TMs M that $\langle M \rangle \in L_D$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{\text{TM}}$. By the definition of L_D , we know $\langle M \rangle \in L_D$ iff M does not accept $\langle M \rangle$. Similarly, by definition of \overline{A}_{TM} , we know that M does not accept $\langle M \rangle$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{\text{TM}}$. Combining these statements together, we see $\langle M \rangle \in L_D$ iff $\langle M, \langle M \rangle \rangle \in \overline{A}_{\text{TM}}$. Thus f is a mapping reduction from L_D to \overline{A}_{TM} , so $L_D \leq_M \overline{A}_{\text{TM}}$, as required. ■

The Amplifier Machine

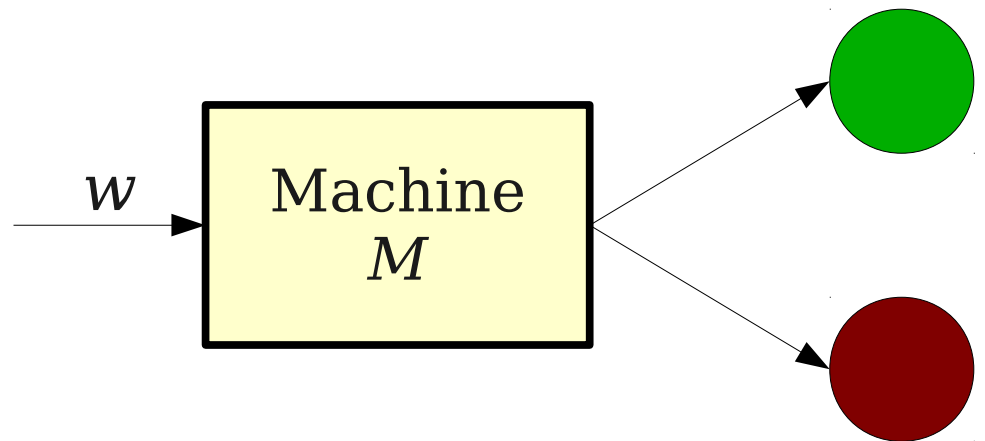
TMs in TMs

- As we've seen, Turing machines can run other Turing machines as subroutines.
- In order to reduce certain problems to one another, it is useful / necessary to embed Turing machines inside of one another.
 - We'll see an example in a second.
- One construction, in particular, is useful for reductions like these.

The Amplifier Machine

For any TM M and string w , let $\text{Amp}(M, w)$ be this TM:

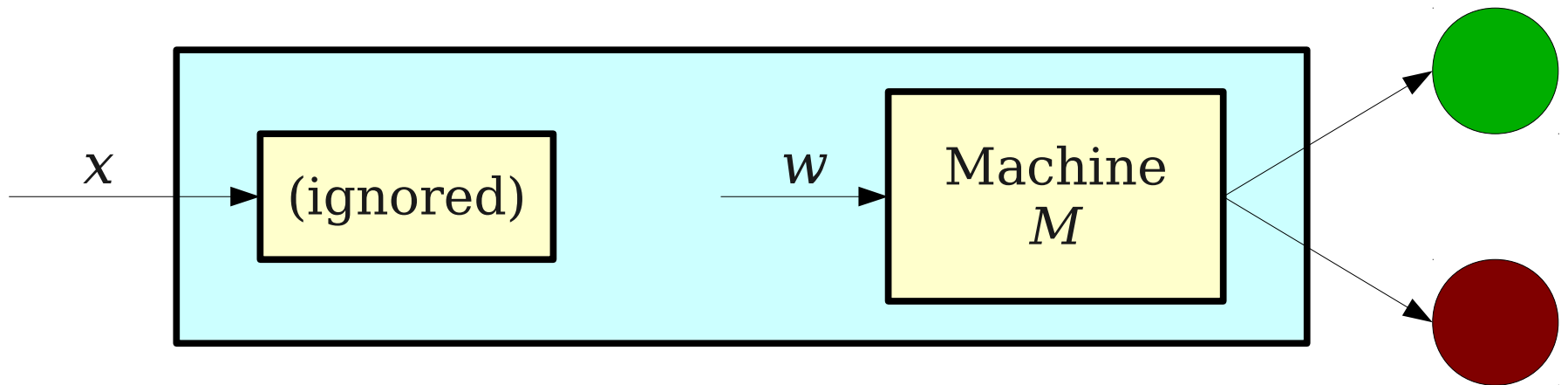
$\text{Amp}(M, w) =$ “On input x :
Ignore x .
Run M on w .
If M accepts w , then $\text{Amp}(M, w)$ accepts x .
If M rejects w , then $\text{Amp}(M, w)$ rejects x .”



The Amplifier Machine

For any TM M and string w , let $\text{Amp}(M, w)$ be this TM:

$\text{Amp}(M, w) =$ “On input x :
Ignore x .
Run M on w .
If M accepts w , then $\text{Amp}(M, w)$ accepts x .
If M rejects w , then $\text{Amp}(M, w)$ rejects x .”



The Amplifier Machine

For any TM M and string w , let $\text{Amp}(M, w)$ be this TM:

$\text{Amp}(M, w) =$ “On input x :
Ignore x .
Run M on w .
If M accepts w , then $\text{Amp}(M, w)$ accepts x .
If M rejects w , then $\text{Amp}(M, w)$ rejects x .”

Theorem 1: If M accepts w , then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If M does not accept w , then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

Corollary 1: M accepts w iff $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$

Corollary 2: M does not accept w iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

For any TM M and string w , let $\text{Amp}(M, w)$ be the following TM:

$\text{Amp}(M, w) =$ “On input x :

Ignore x .

Run M on w .

If M accepts w , then $\text{Amp}(M, w)$ accepts x .

If M rejects w , then $\text{Amp}(M, w)$ rejects x .”

Theorem: If M accepts w , then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If M does not accept w , then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

Proof: First, we consider what happens if M accepts w . In this case, consider what happens when we run $\text{Amp}(M, w)$ on an arbitrary input string x . $\text{Amp}(M, w)$ will run M on w , and since M accepts w , $\text{Amp}(M, w)$ accepts x . Since our choice of x was arbitrary, we see that $\text{Amp}(M, w)$ accepts any input, so $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$.

Otherwise, M does not accept w , so M rejects w or M loops on w . Consider the result of running $\text{Amp}(M, w)$ on an arbitrary string x . If M rejects w , then $\text{Amp}(M, w)$ rejects x . Otherwise, $\text{Amp}(M, w)$ loops on x . In both cases, $\text{Amp}(M, w)$ doesn't accept x . Since our choice of x was arbitrary, we see that $\text{Amp}(M, w)$ never accepts any input, so $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. ■

The Amplifier Machine

For any TM M and string w , let $\text{Amp}(M, w)$ be this TM:

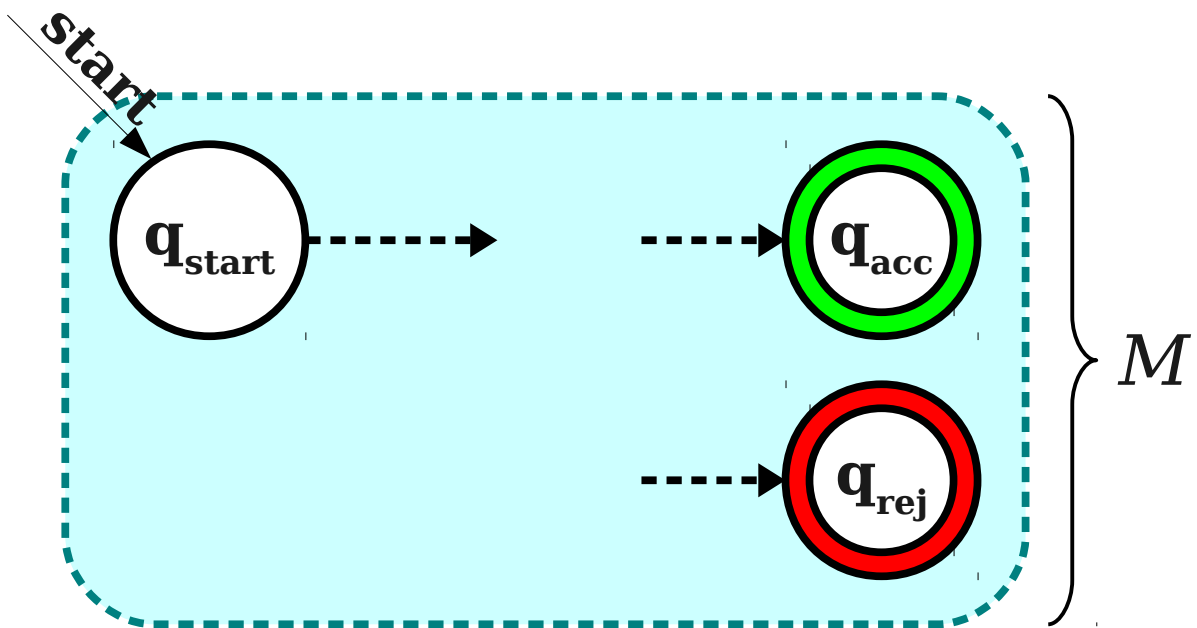
$\text{Amp}(M, w) =$ “On input x :
Ignore x .
Run M on w .
If M accepts w , then $\text{Amp}(M, w)$ accepts x .
If M rejects w , then $\text{Amp}(M, w)$ rejects x .”

Theorem 1: If M accepts w , then $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. If M does not accept w , then $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

Corollary 1: M accepts w iff $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$

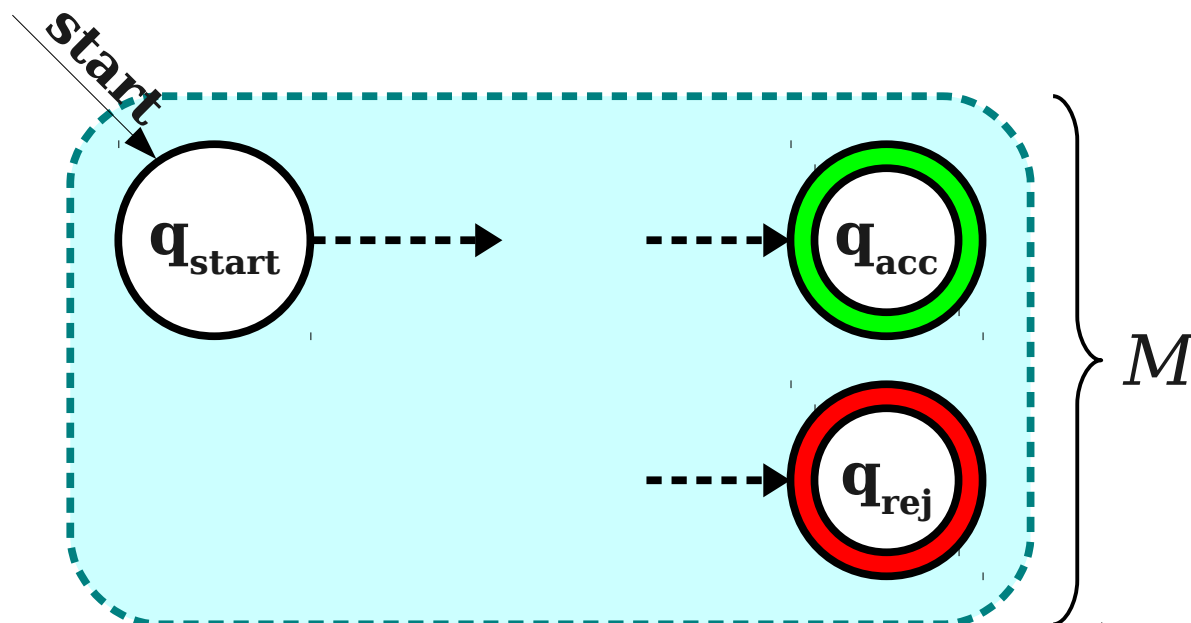
Corollary 2: M does not accept w iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$.

Theorem 2: The function $f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle$ is computable.



“On input x :

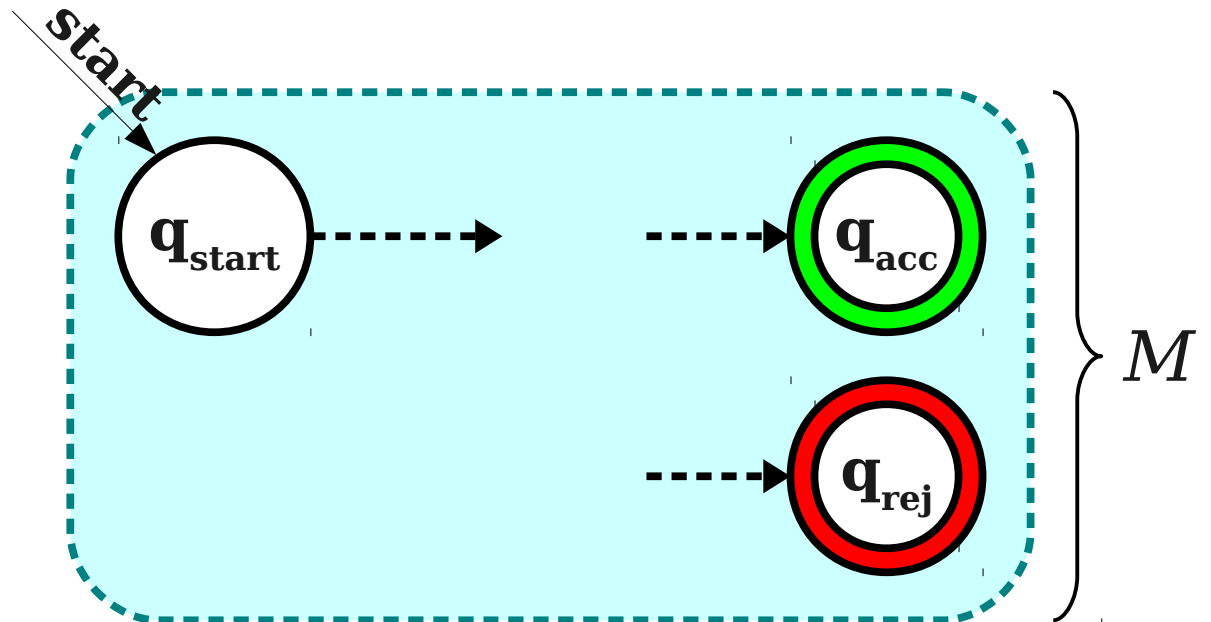
- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

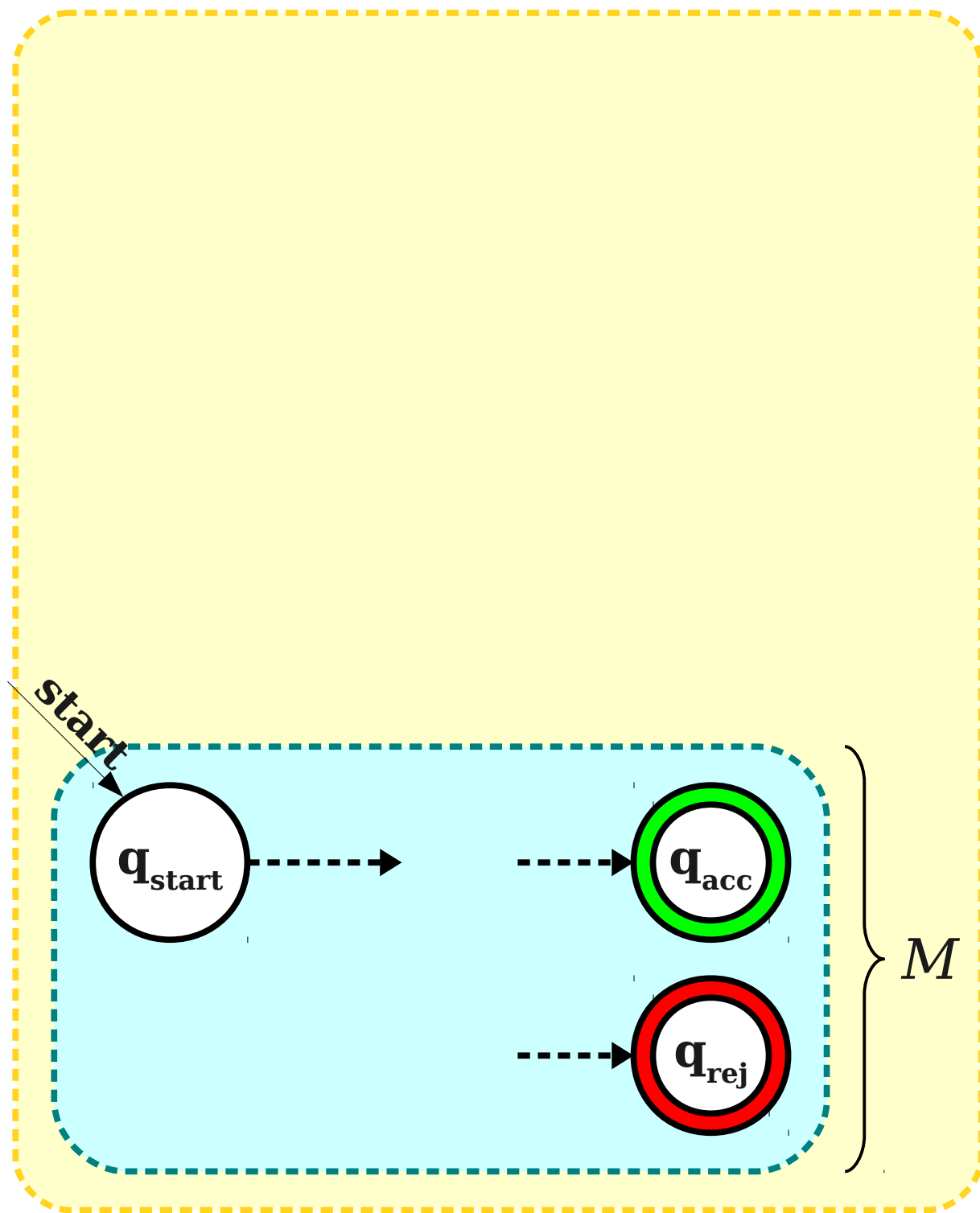
Hypothetically,
assume that w is the
string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

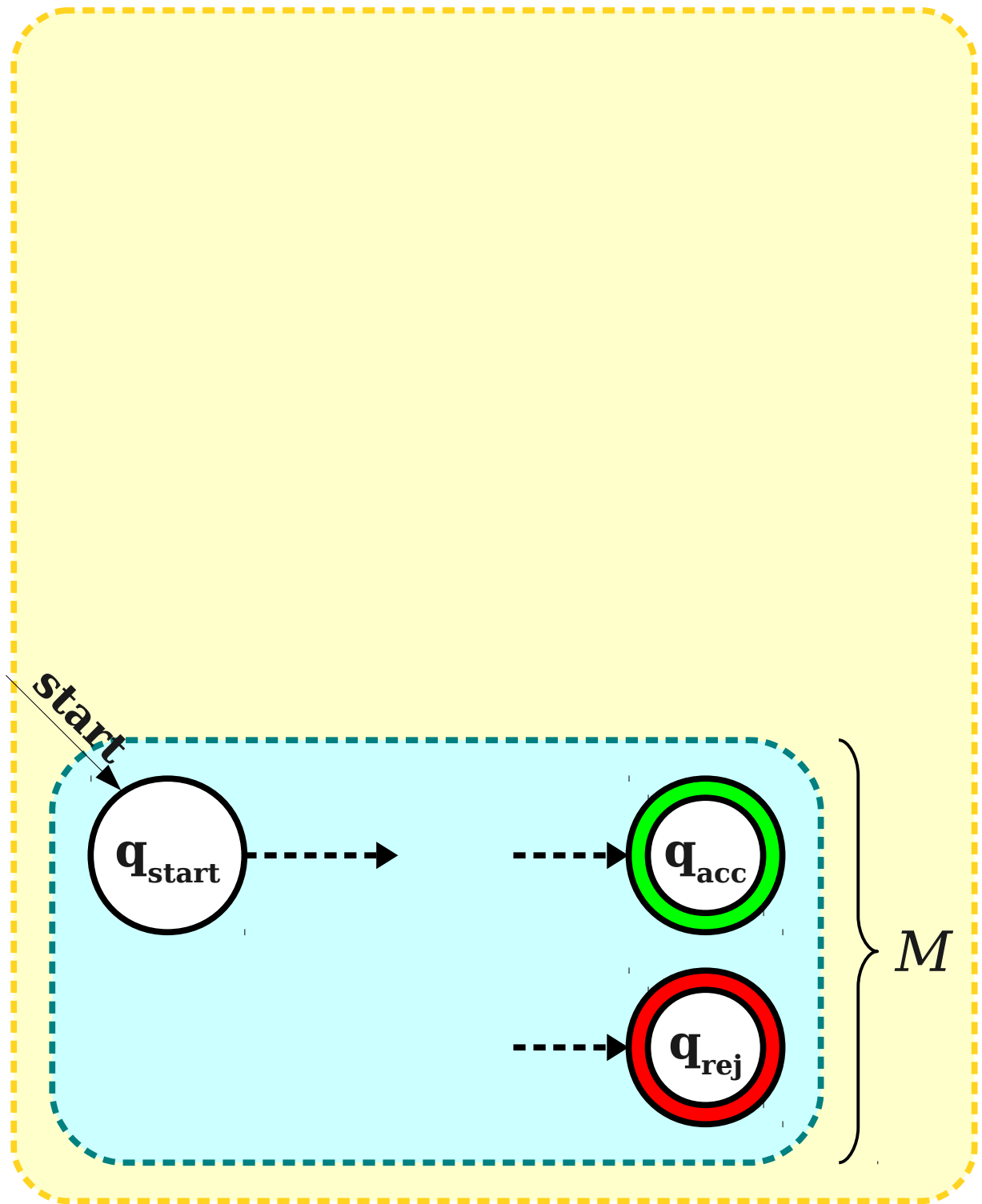
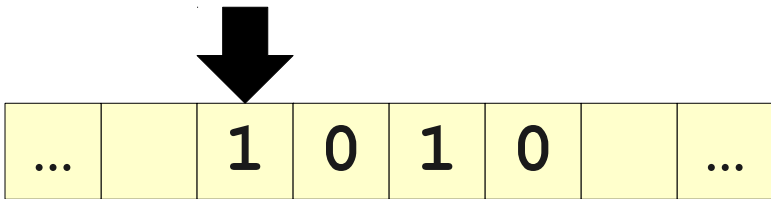
Hypothetically,
assume that w is the
string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

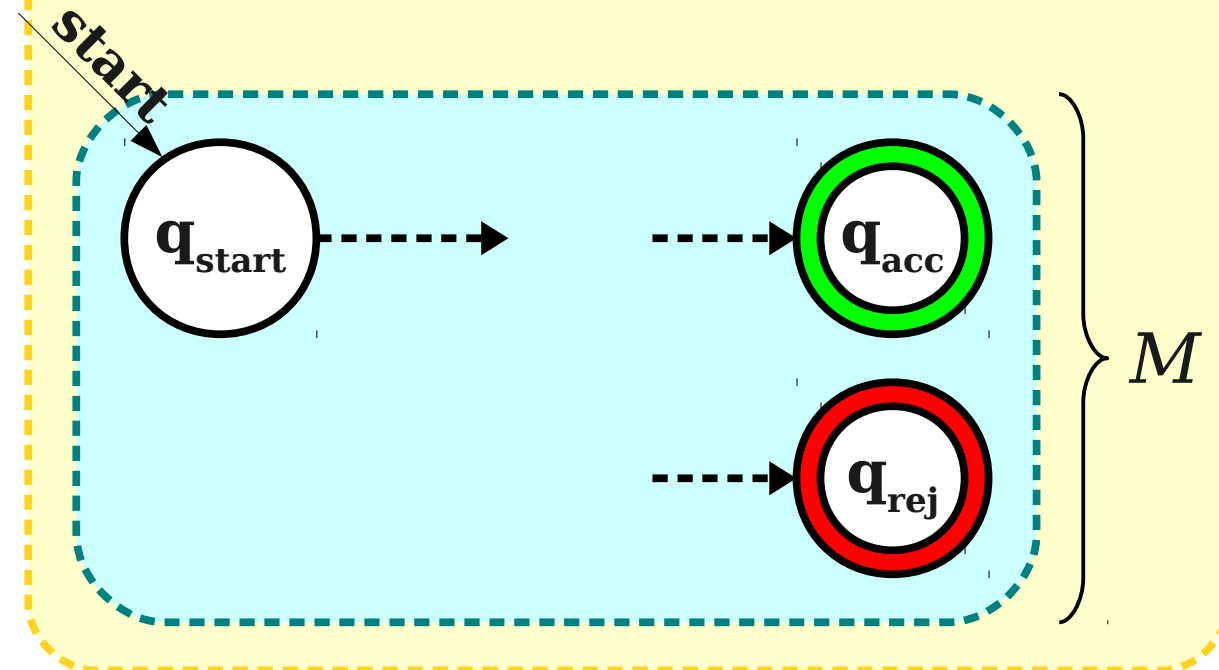
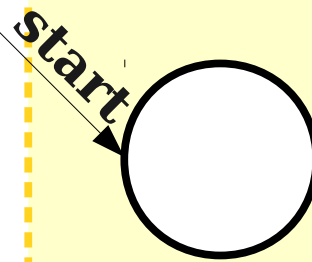
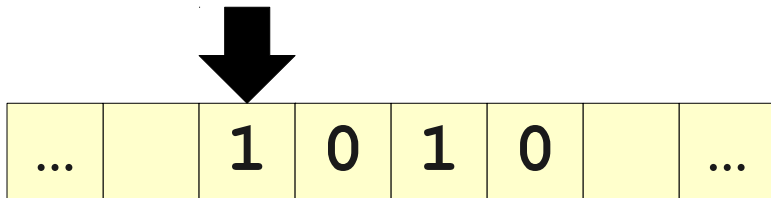
Hypothetically,
assume that w is the
string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

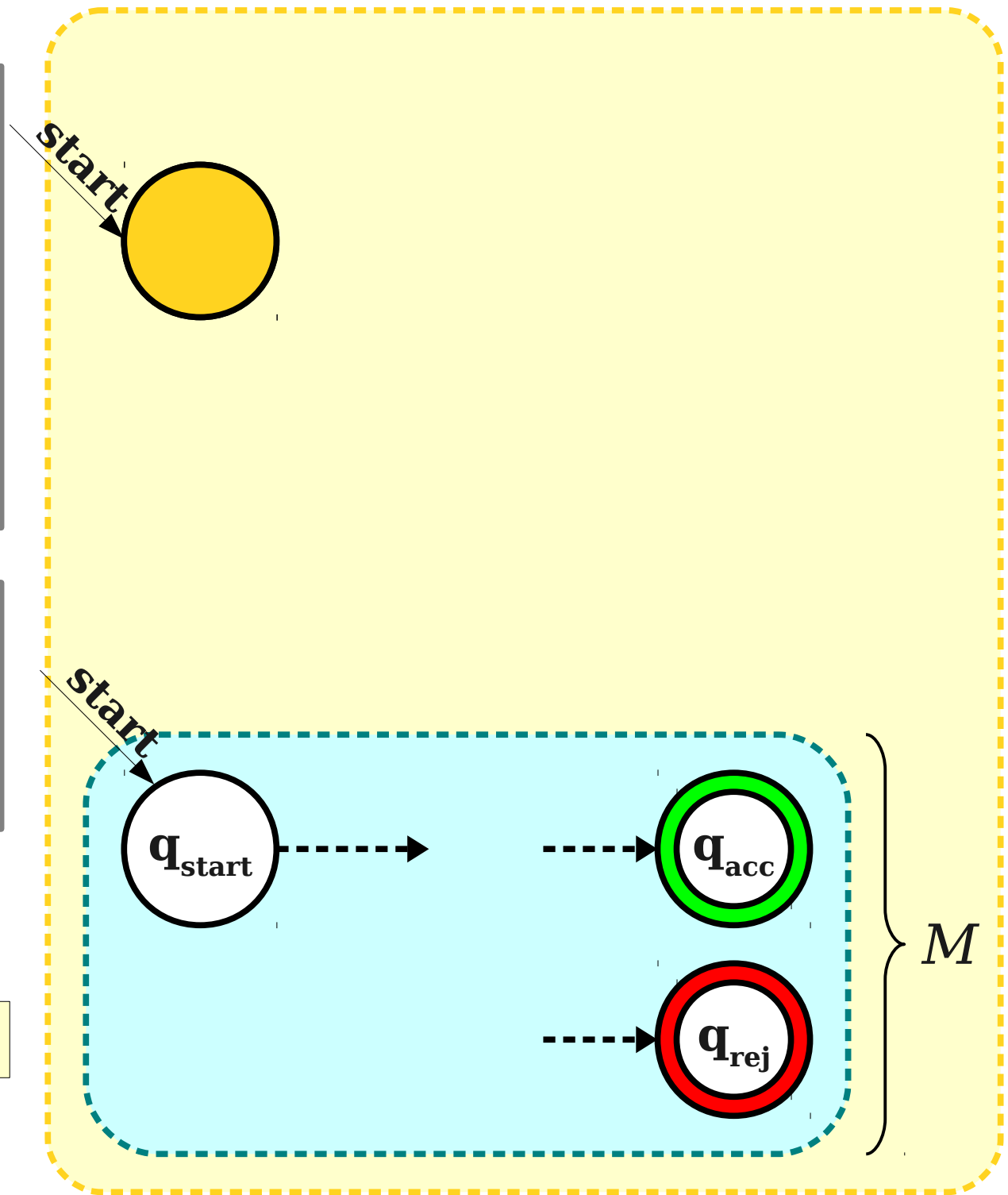
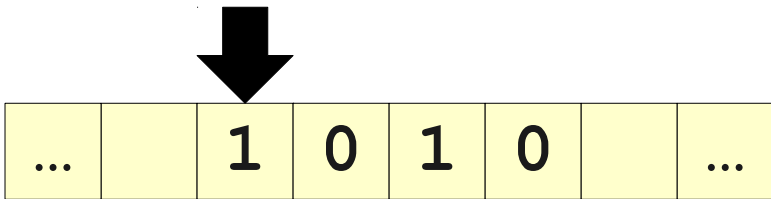
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

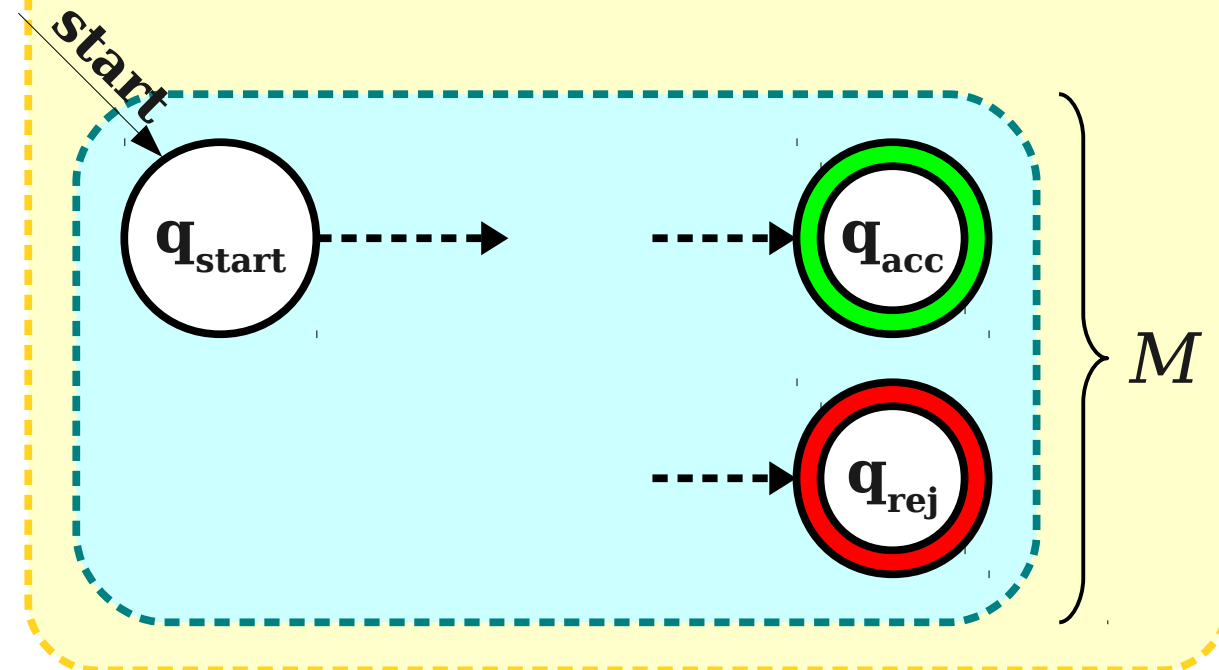
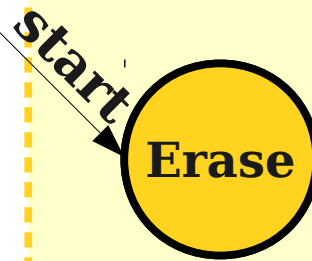
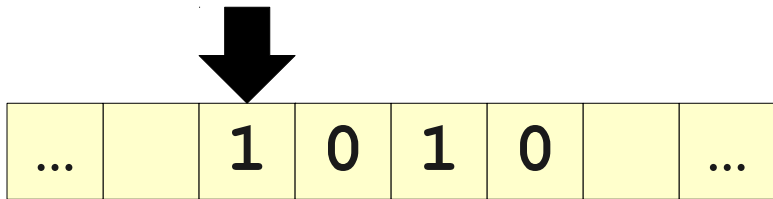
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

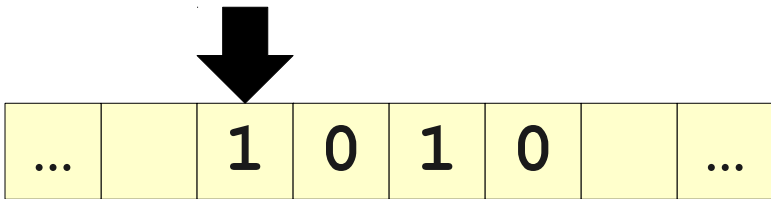
Hypothetically, assume that w is the string **1101**.



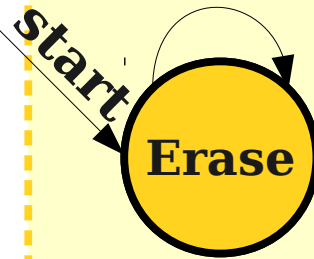
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

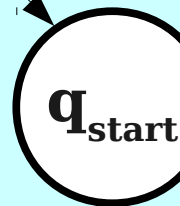
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



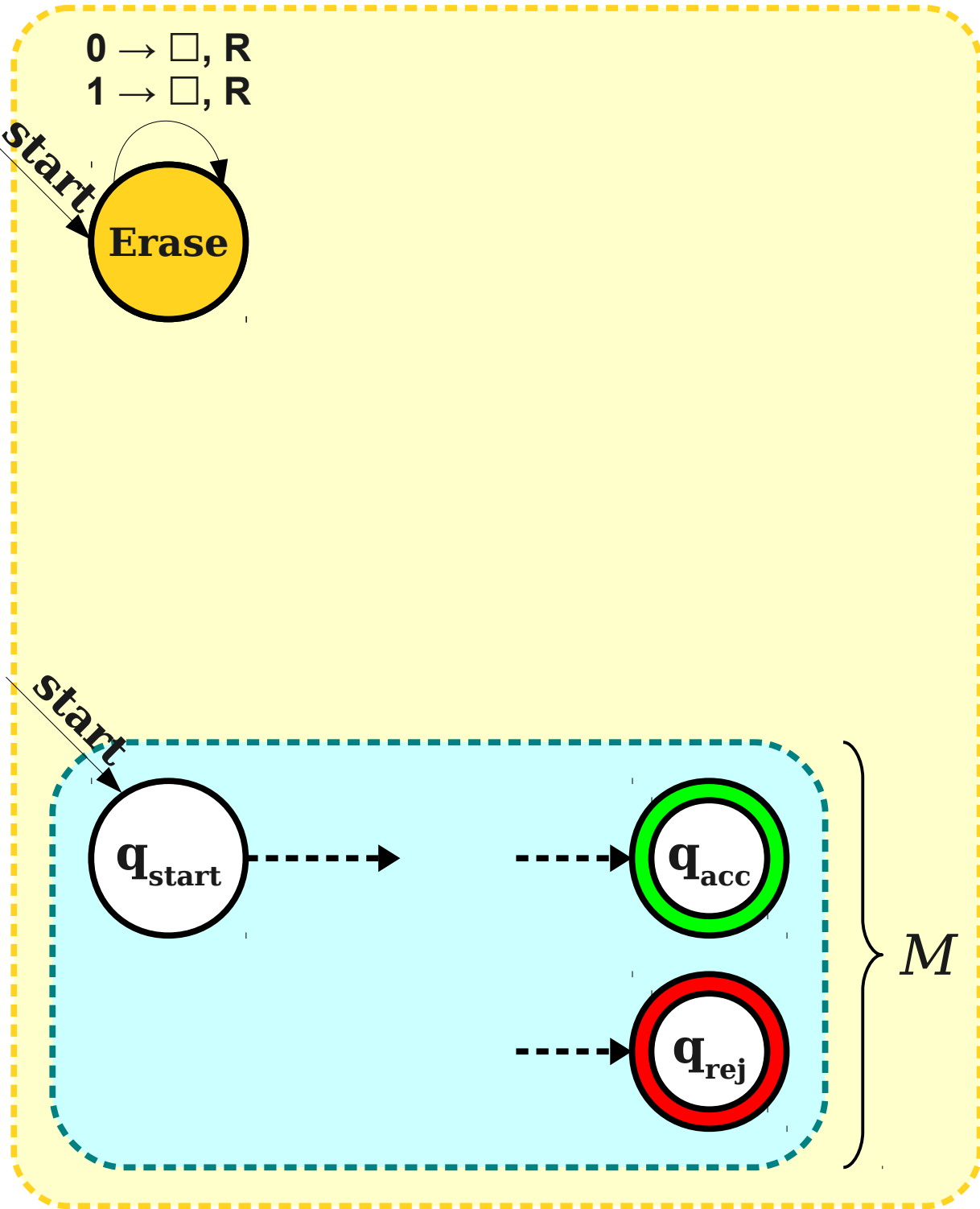
start



q_{acc}

q_{rej}

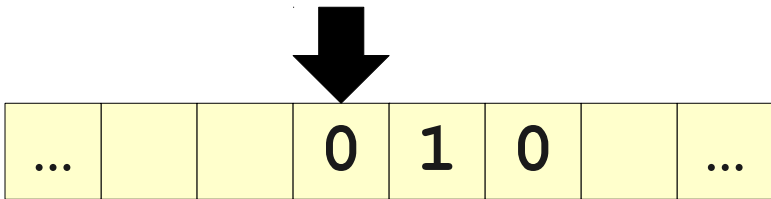
M



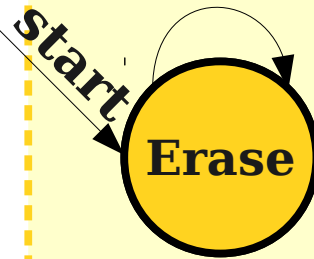
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

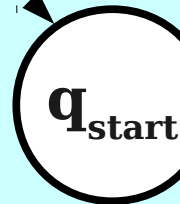
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



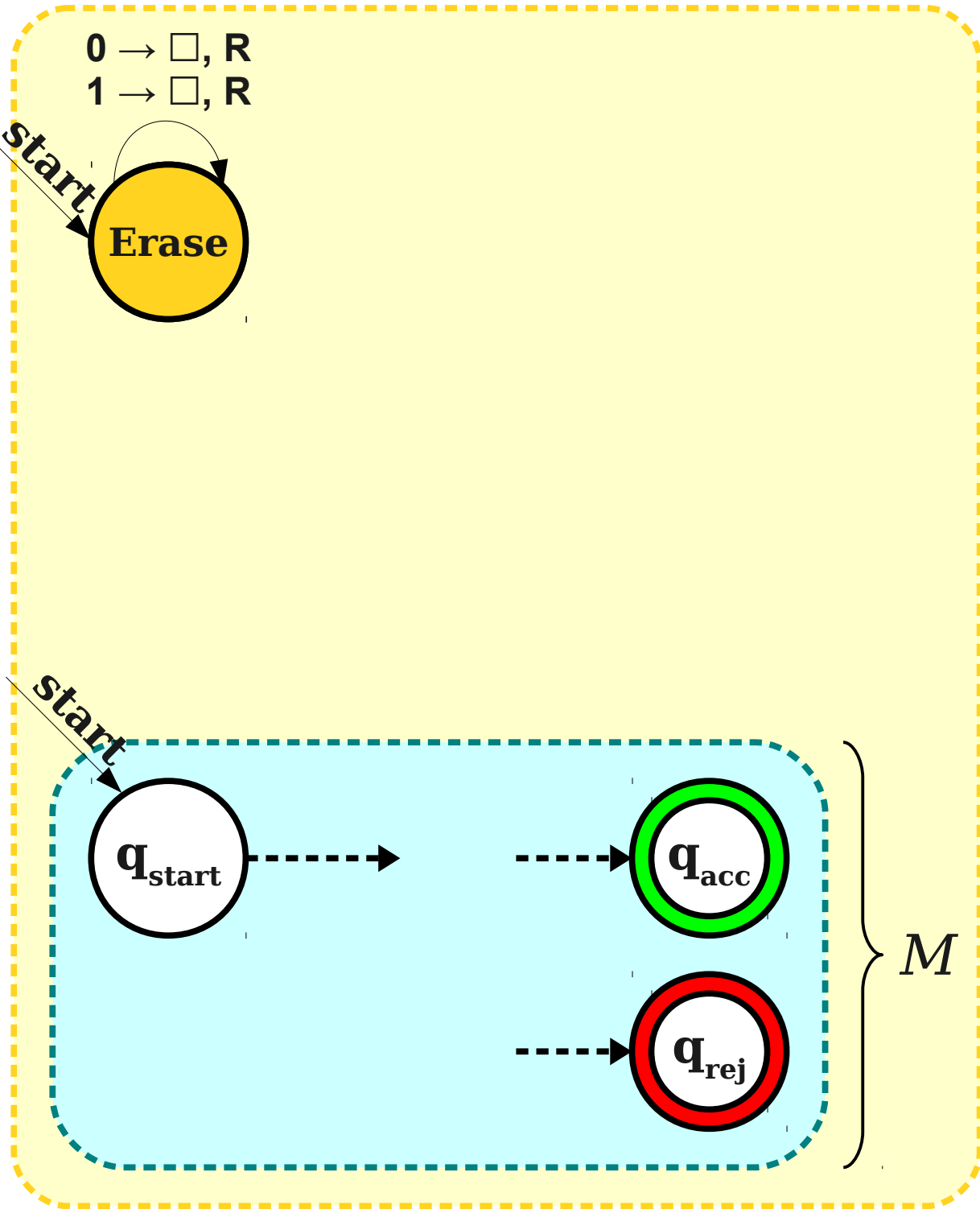
start



q_{acc}

q_{rej}

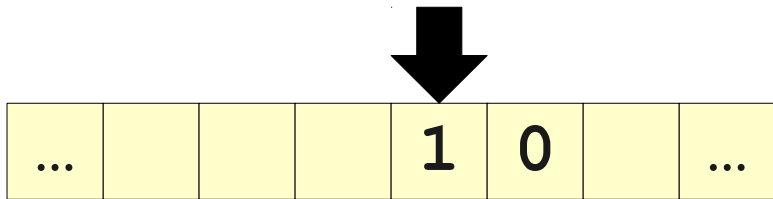
M



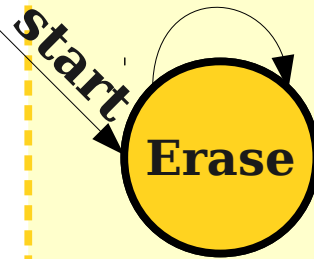
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

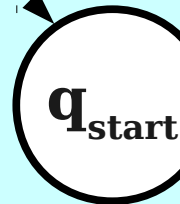
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



start



q_{acc}

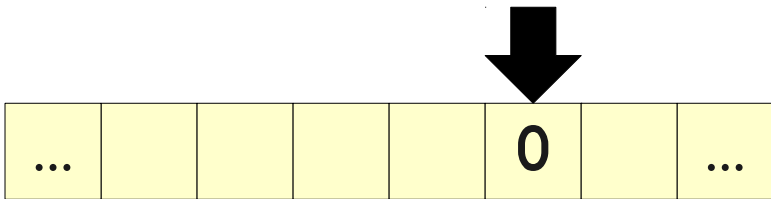
q_{rej}

M

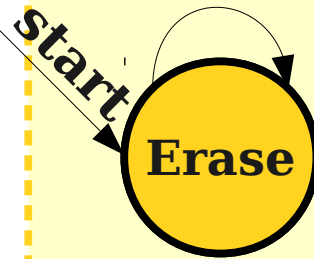
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

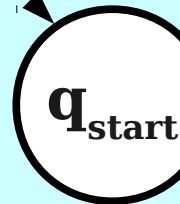
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



start



q_{acc}

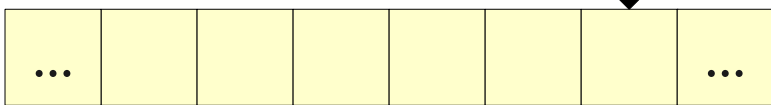
q_{rej}

M

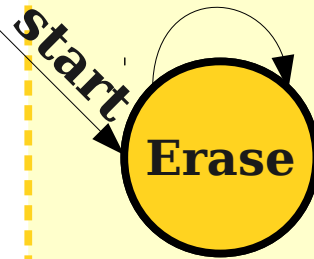
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

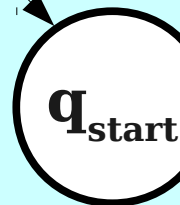
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



start



q_{acc}

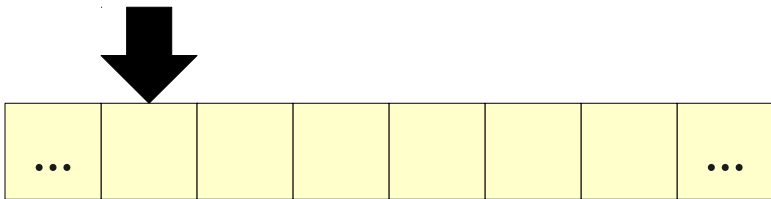
q_{rej}

M

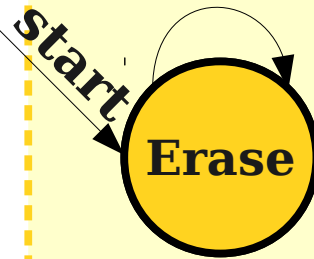
“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

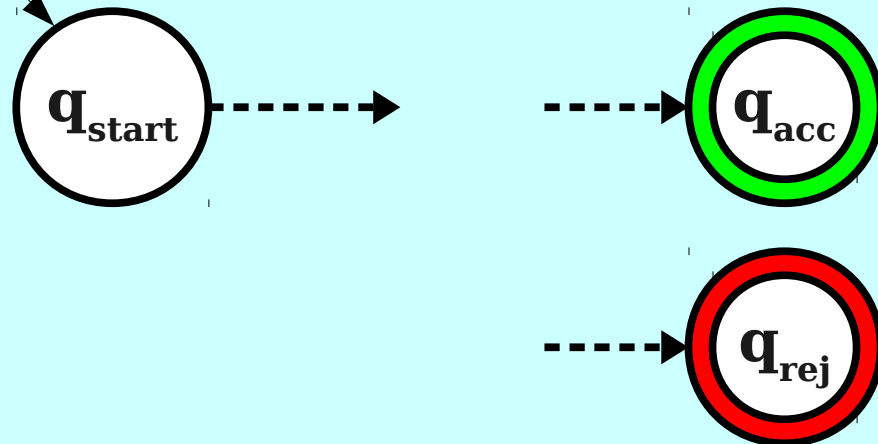
Hypothetically, assume that w is the string **1101**.



$0 \rightarrow \square, R$
 $1 \rightarrow \square, R$



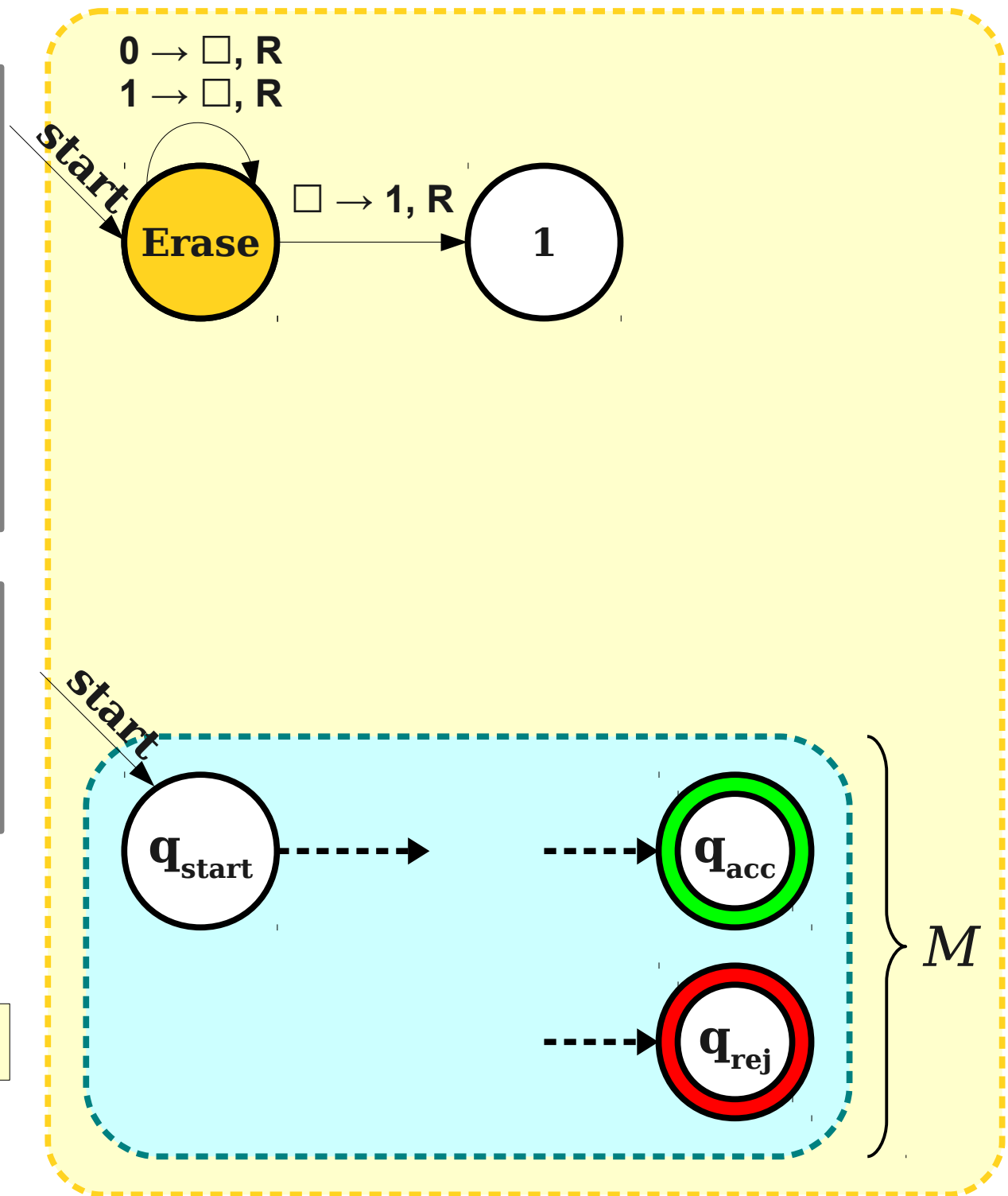
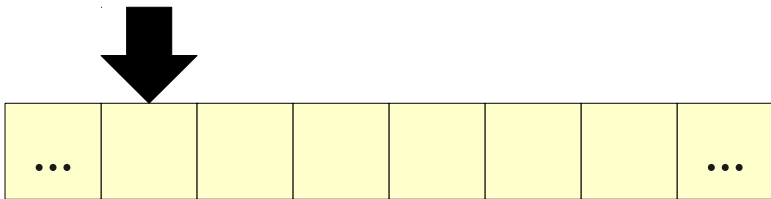
start



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

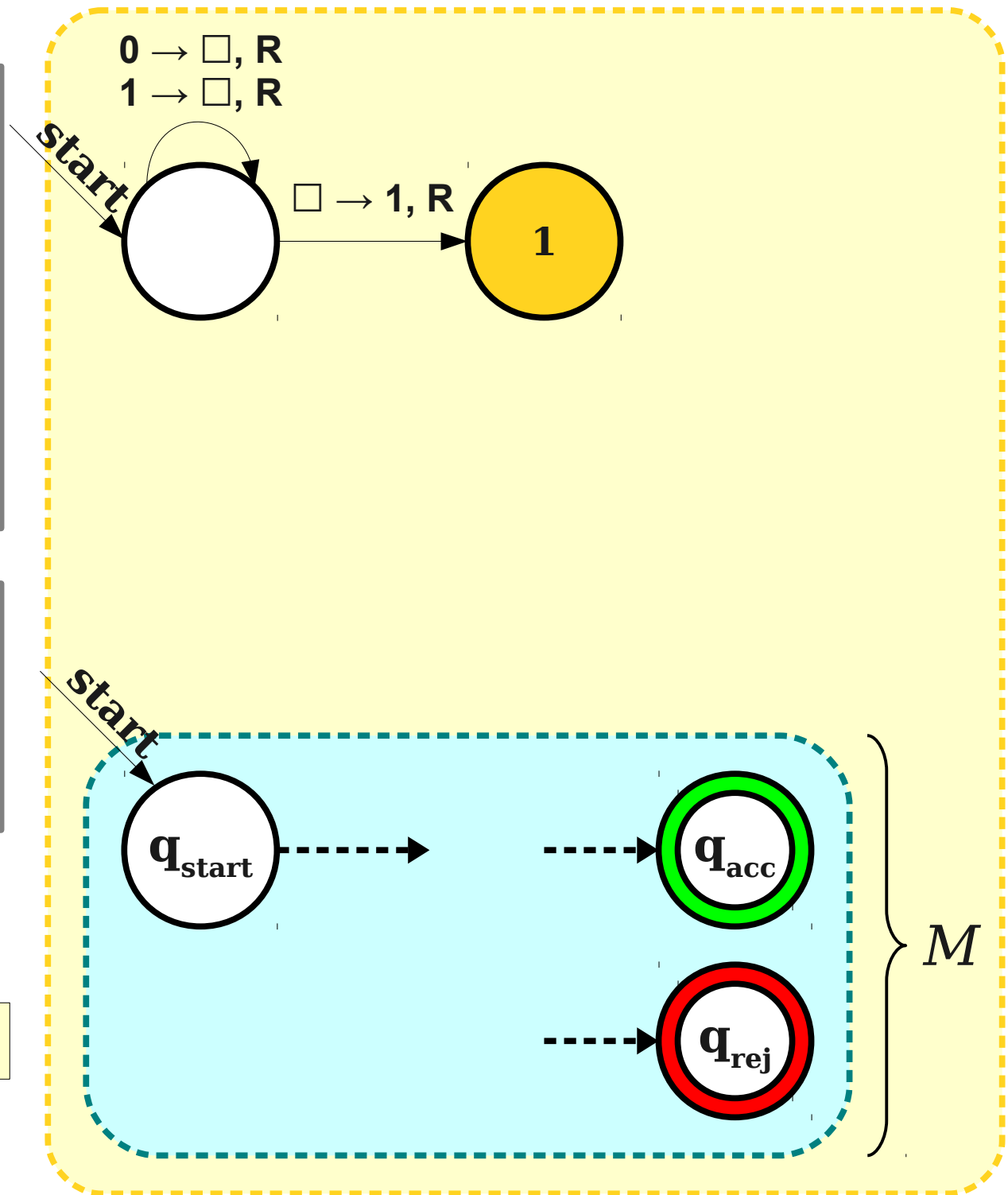
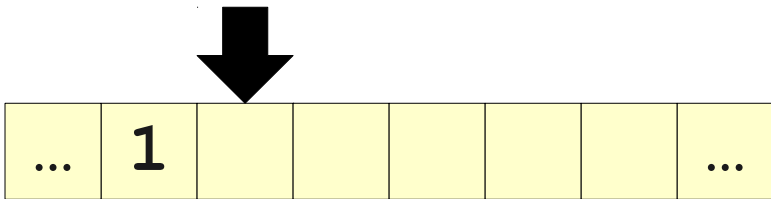
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

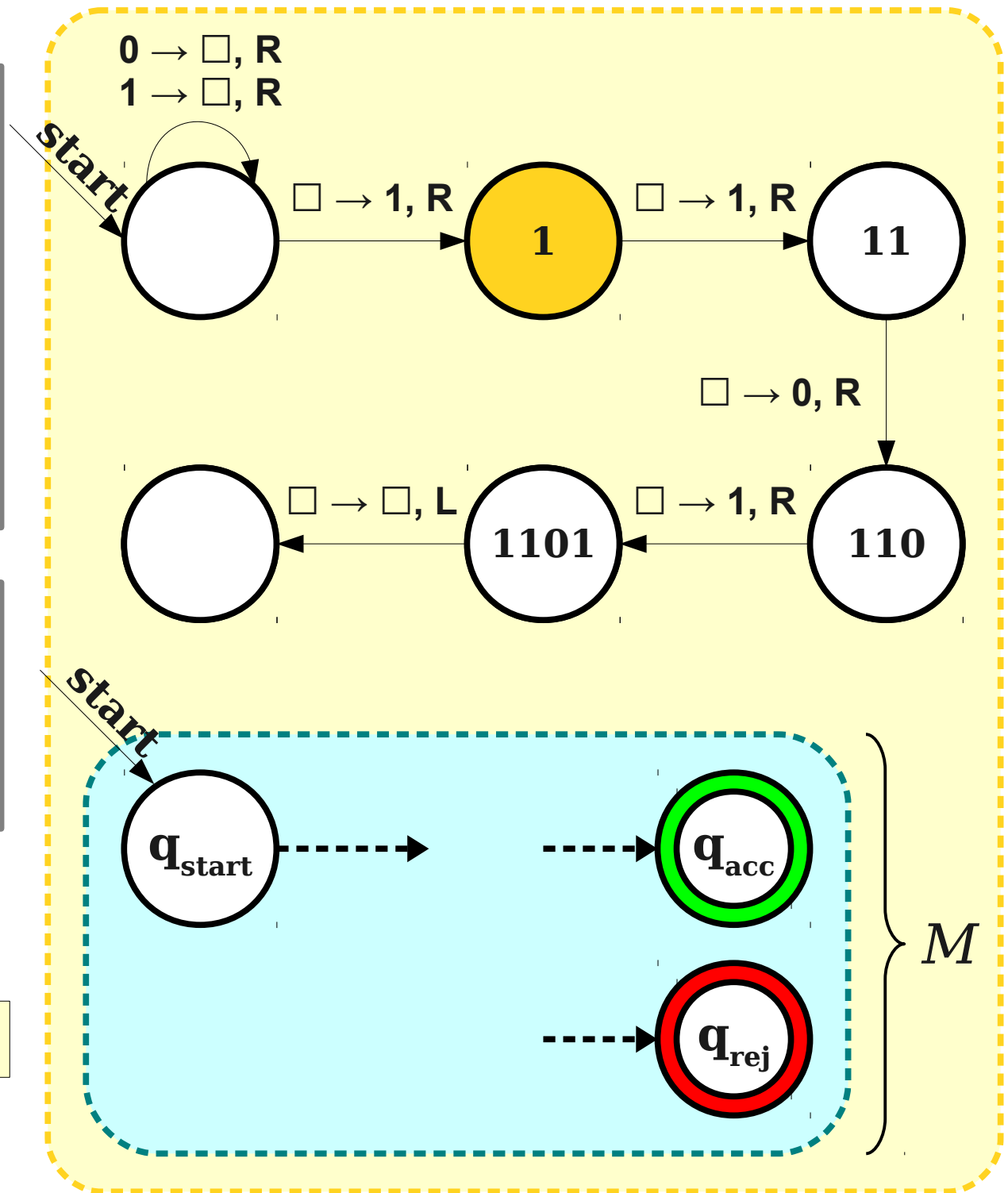
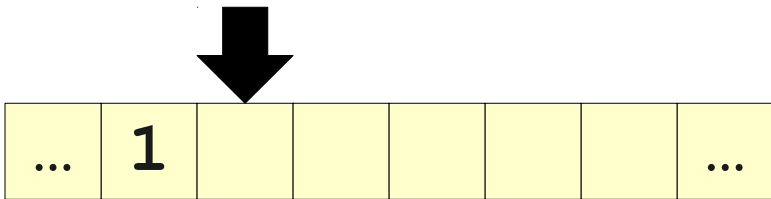
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

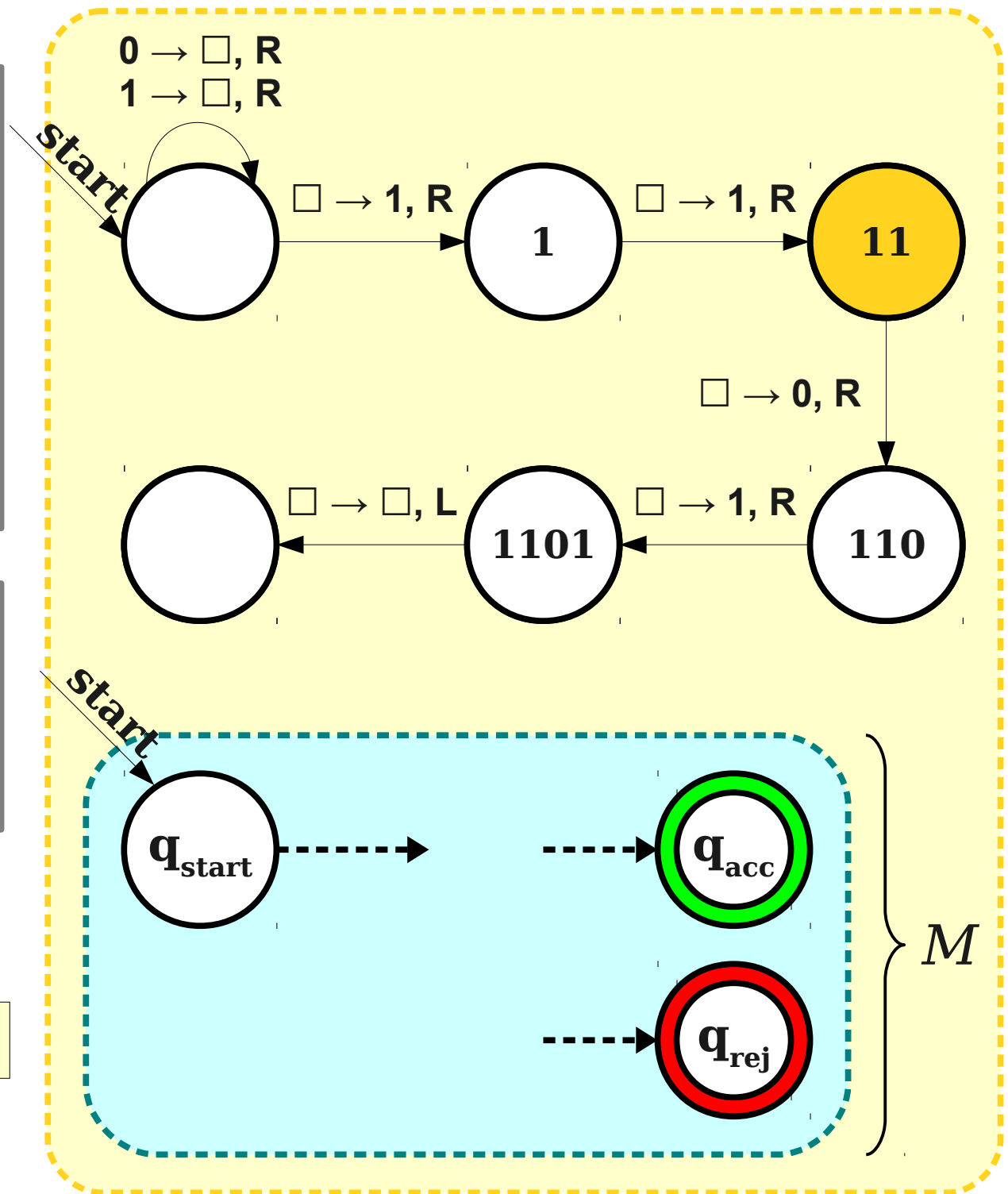
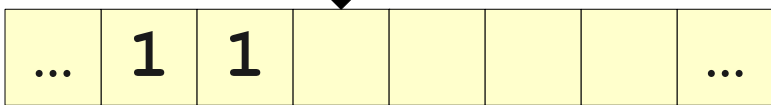
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

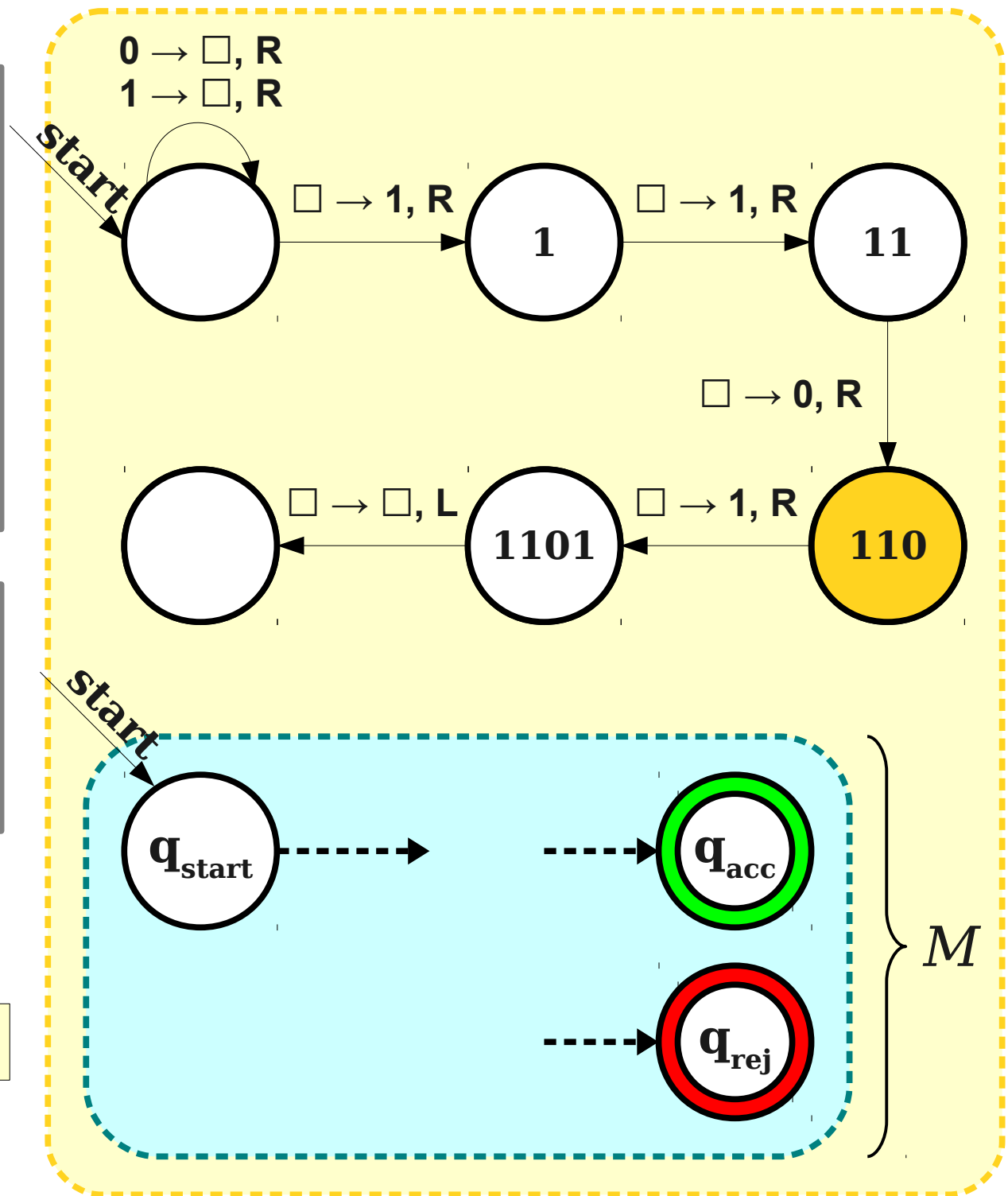
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

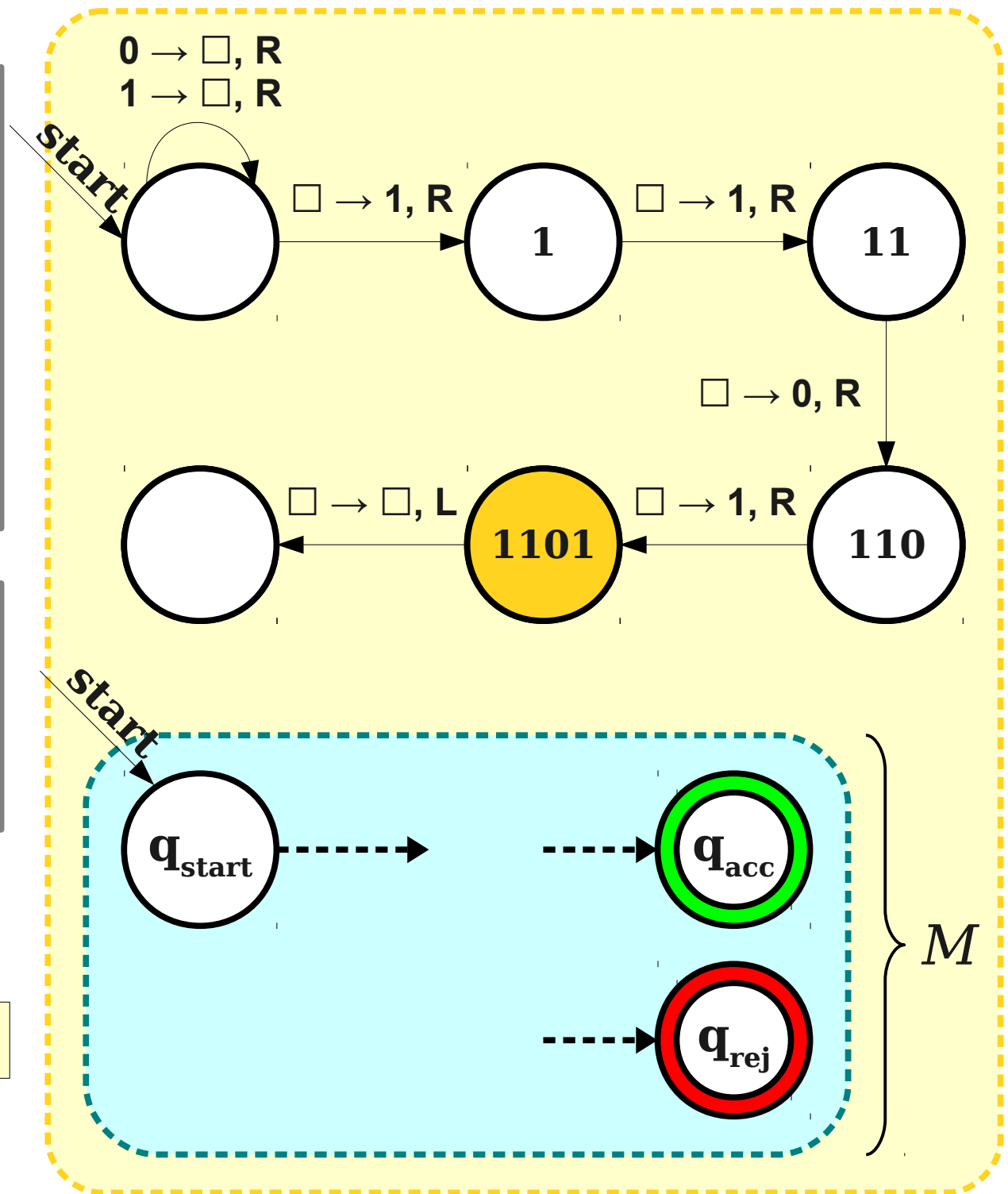
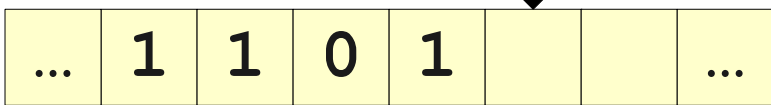
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

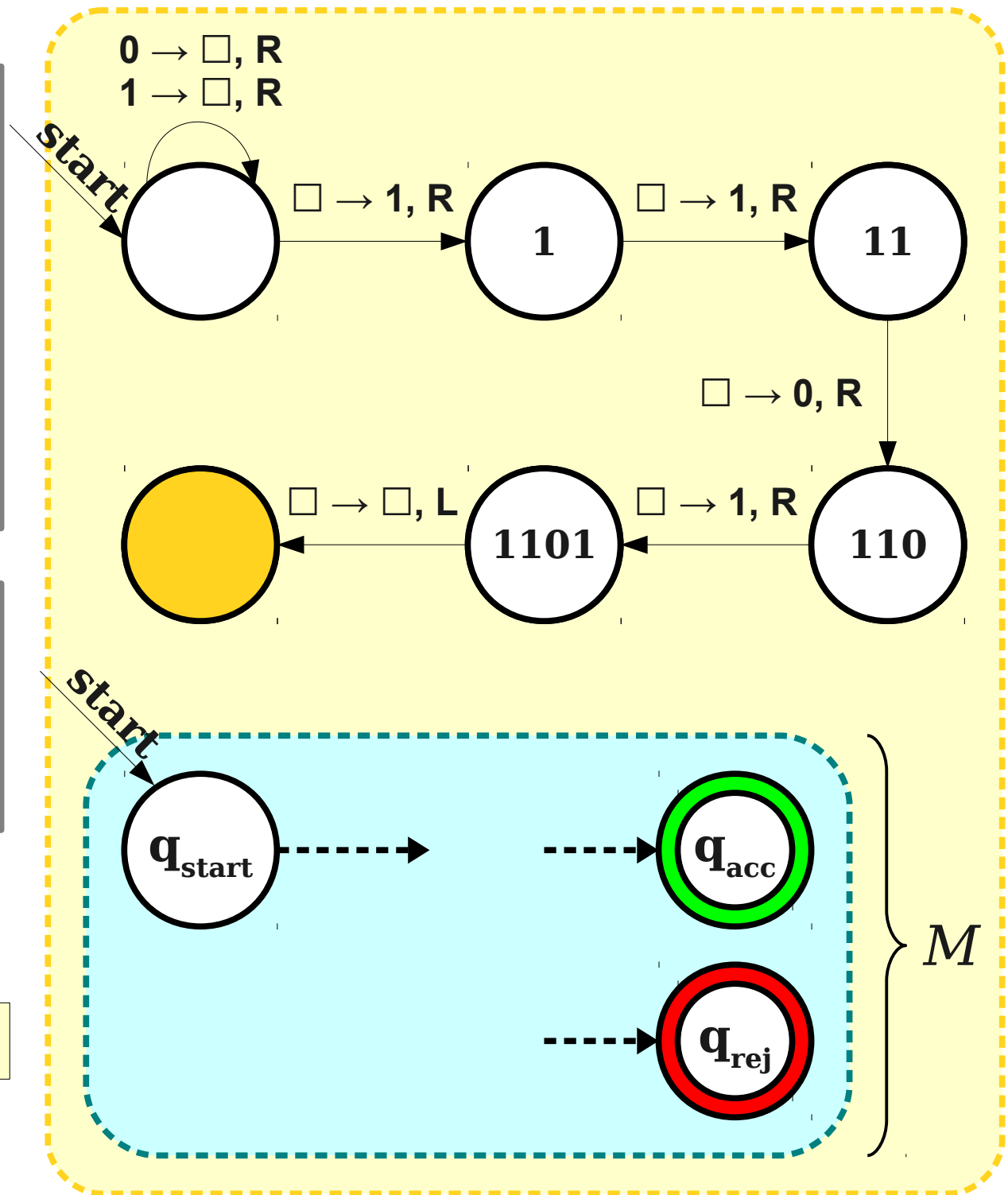
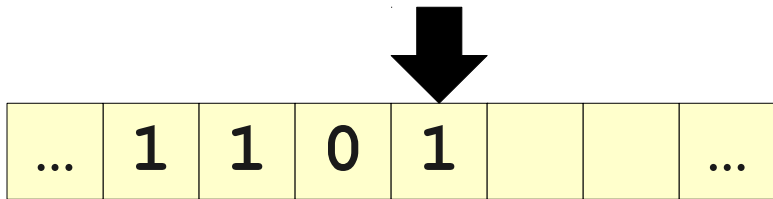
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

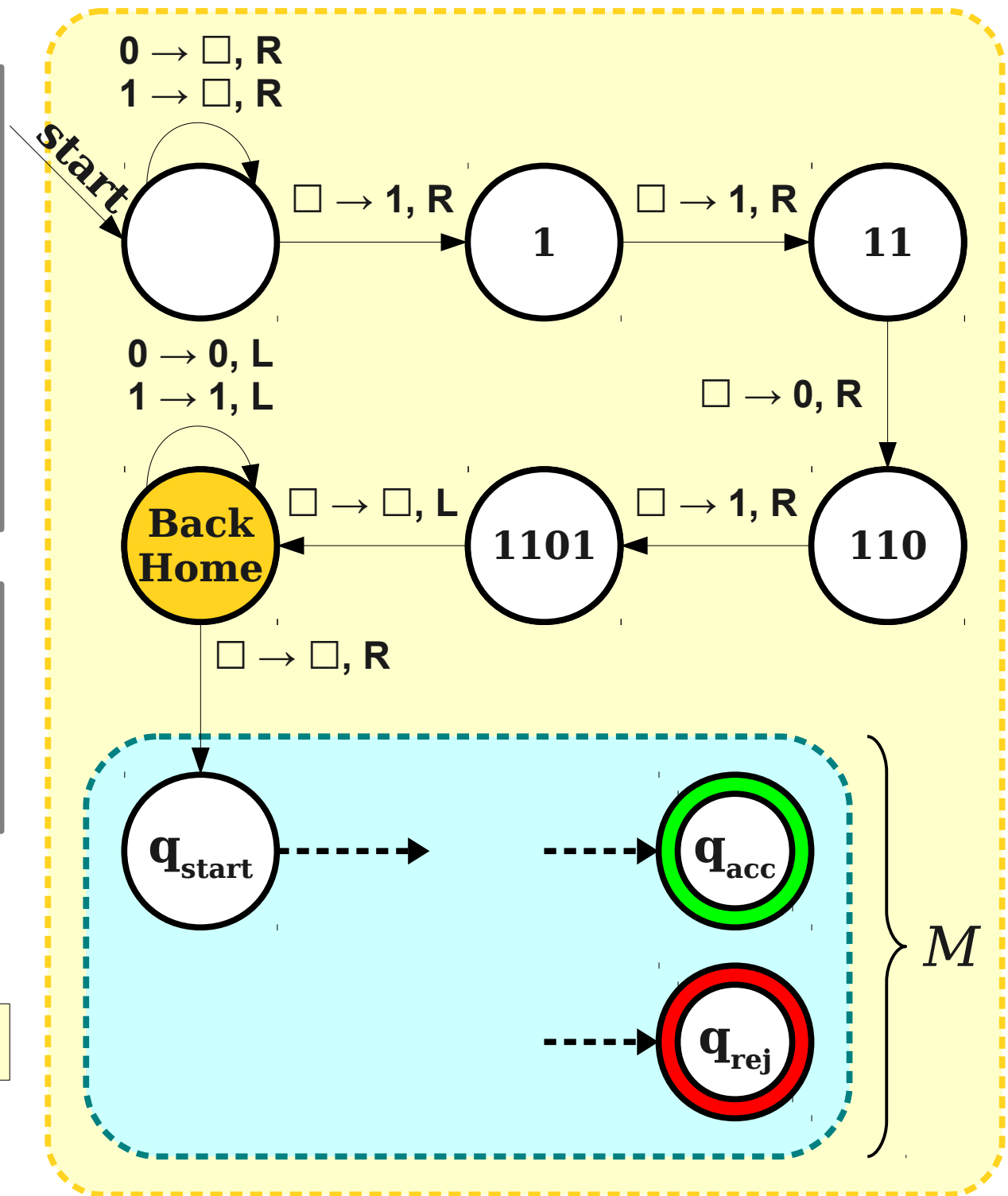
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

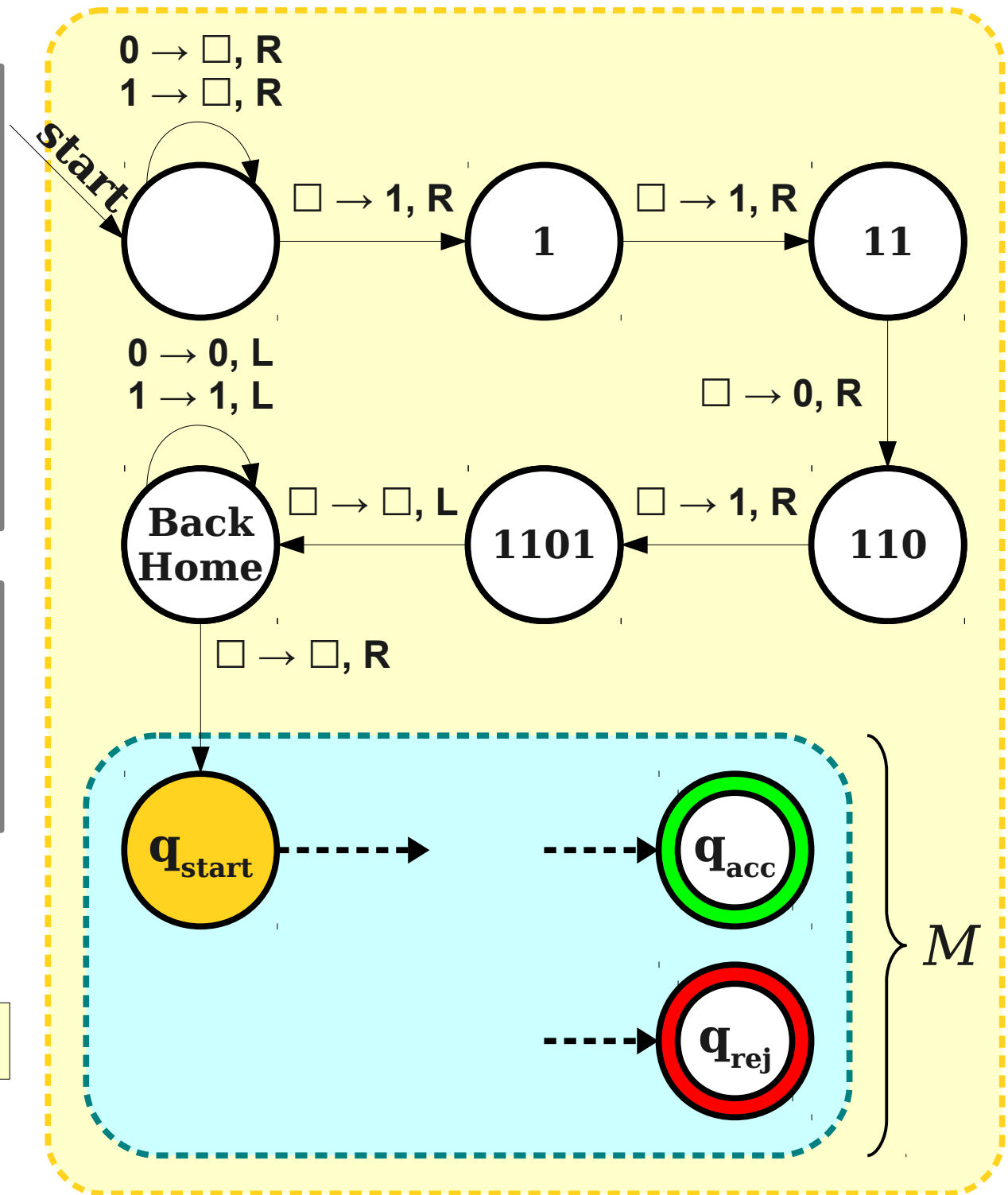
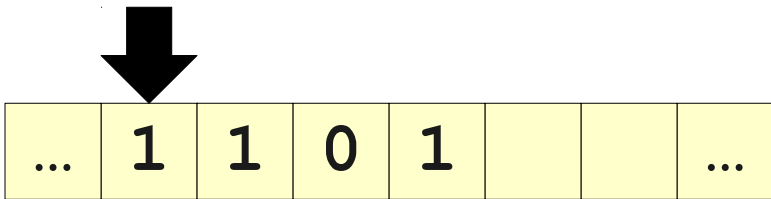
Hypothetically, assume that w is the string **1101**.



“On input x :

- Ignore x .
- Run M on w .
- If M accepts w , we accept x .
- If M rejects w , we reject x .”

Hypothetically, assume that w is the string **1101**.



Using the Amplifier

A More Elaborate Reduction

- Since $\overline{A}_{\text{TM}} \notin \mathbf{RE}$, there is no algorithm for determining whether a TM will not accept a given string.
- Could we check instead whether a TM *never* accepts a string?
- Consider the language
$$L_e = \{ \langle M \rangle \mid M \text{ is a TM and } \mathcal{L}(M) = \emptyset \}$$
- How “hard” is L_e ? Is it **R**, **RE**, **co-RE**, or none of these?

Building an Intuition

- Before we even try to prove how “hard” this language is, we should build an intuition for its difficulty.
- L_e is *probably* not in **RE**, since if we were convinced a TM never accepted, it would be hard to find positive evidence of this.
- L_e is *probably* in co-**RE**, since if we were convinced that a TM *did* accept some string, we could exhaustively search over all strings and try to find the string it accepts.
- Best guess: $L_e \in \text{co-RE} - \mathbf{R}$.

$$\overline{A}_{\text{TM}} \leq_M L_e$$

- We will prove that $L_e \notin \mathbf{RE}$ by showing that $\overline{A}_{\text{TM}} \leq_M L_e$. (This also proves $L_e \notin \mathbf{R}$).

- We want to find a function f such that

$$\langle M, w \rangle \in \overline{A}_{\text{TM}} \quad \text{iff} \quad f(\langle M, w \rangle) \in L_e$$

- Since L_e is a language of TM descriptions, let's assume $f(\langle M, w \rangle) = \langle N \rangle$ for some TM N . Then

$$\langle M, w \rangle \in \overline{A}_{\text{TM}} \quad \text{iff} \quad \langle N \rangle \in L_e$$

- Expanding out definitions, we get

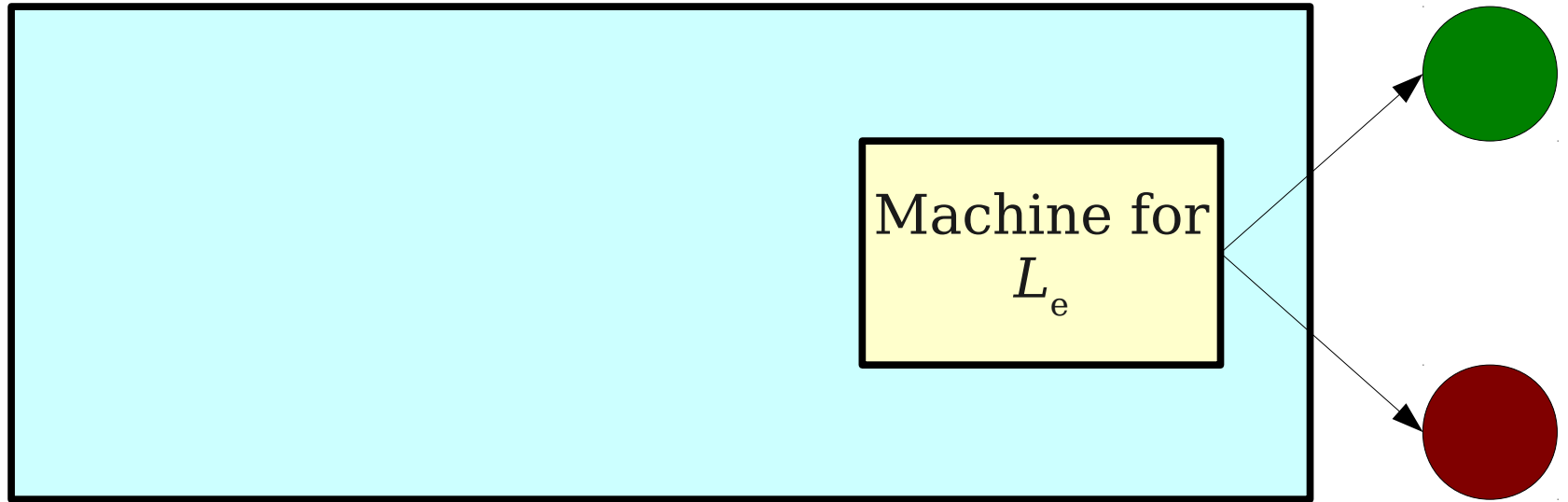
$$\mathbf{M \text{ doesn't accept } w \text{ iff } \mathcal{L}(N) = \emptyset}$$

- How do we pick the machine N ?

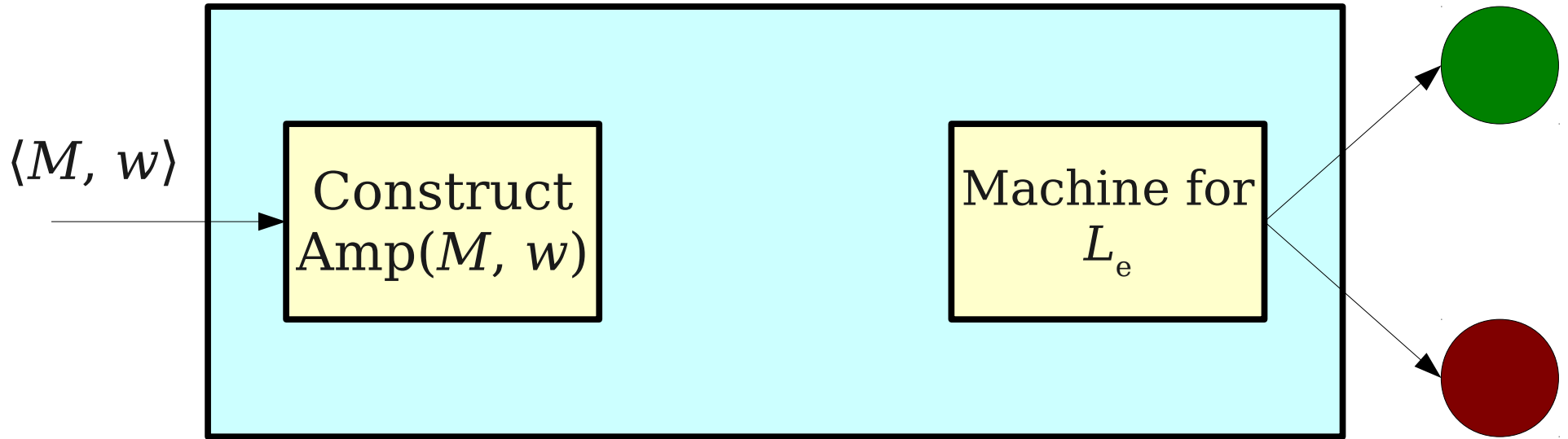
The Reduction

- Choose N such that this holds:
 M doesn't accept w iff $\mathcal{L}(N) = \emptyset$
- We can pick $N = \text{Amp}(M, w)$.
 - **Recall:** $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$ iff M doesn't accept w .
- Since $f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle$ is computable, this is the mapping reduction we need!

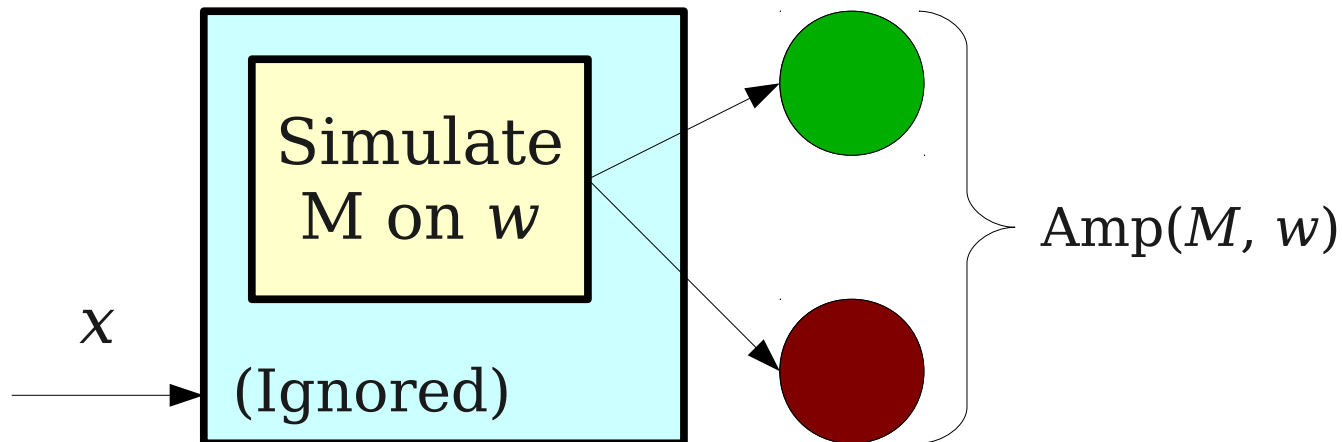
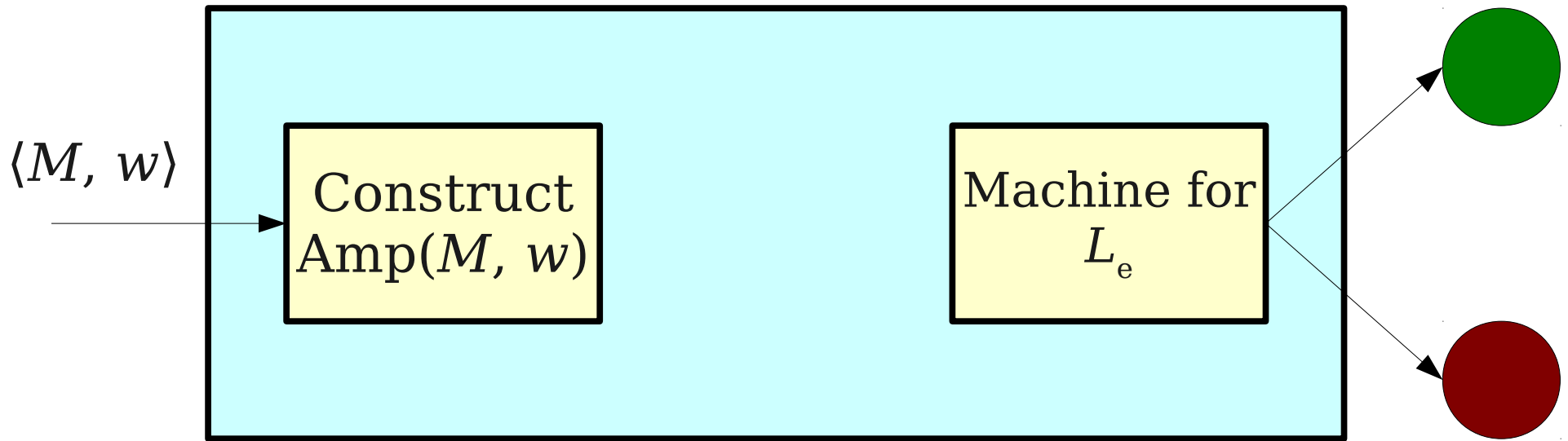
The Reduction



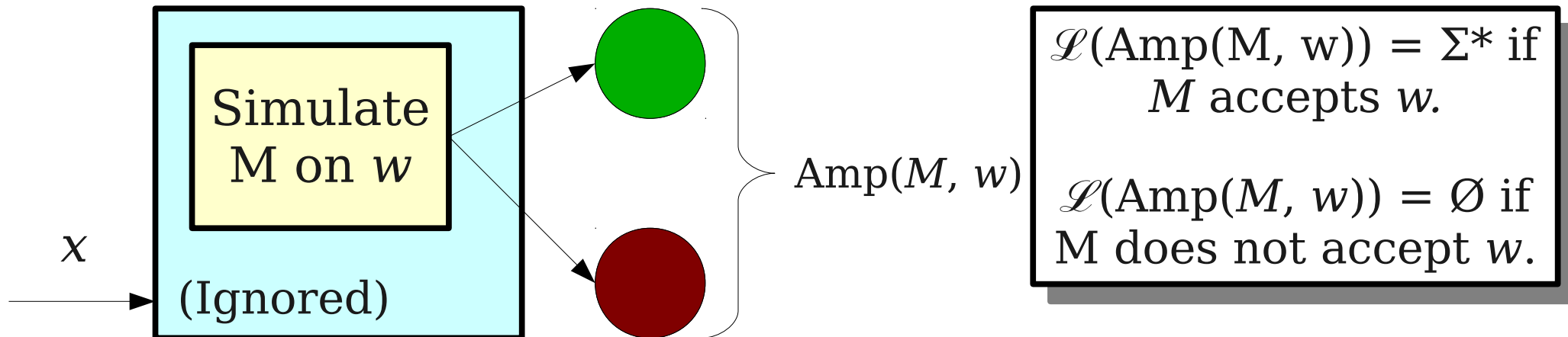
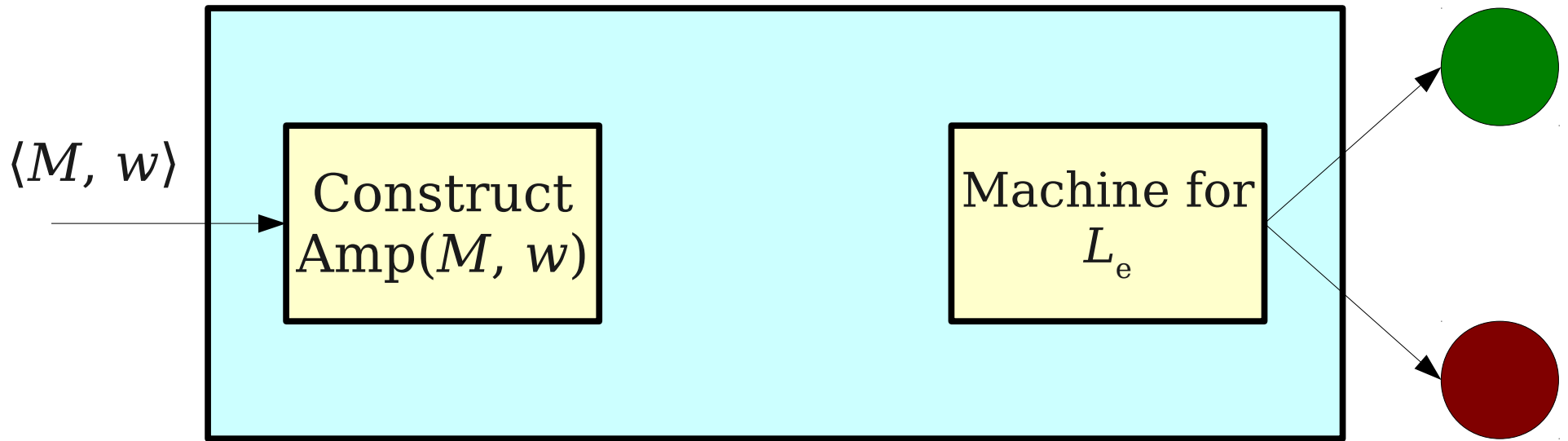
The Reduction



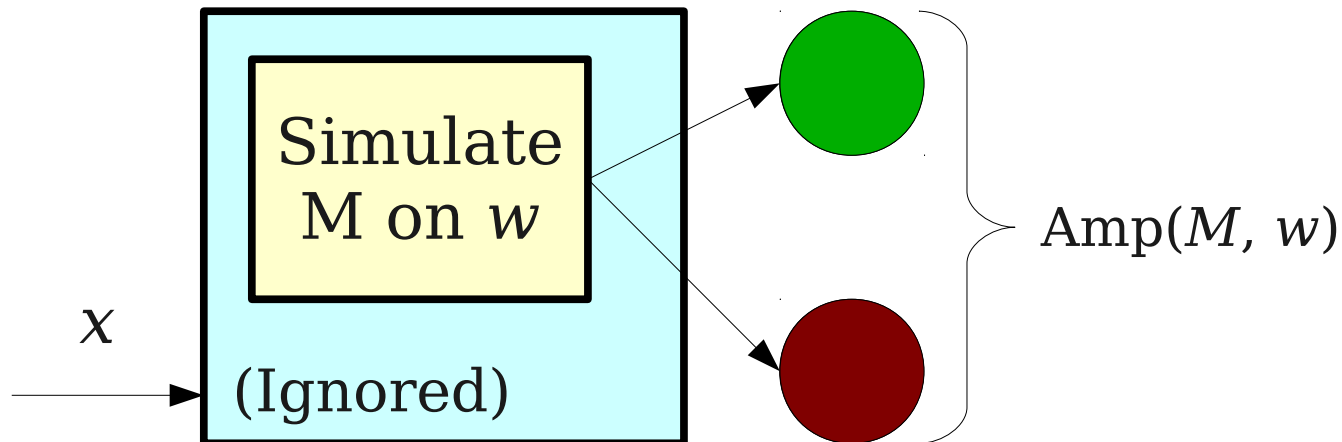
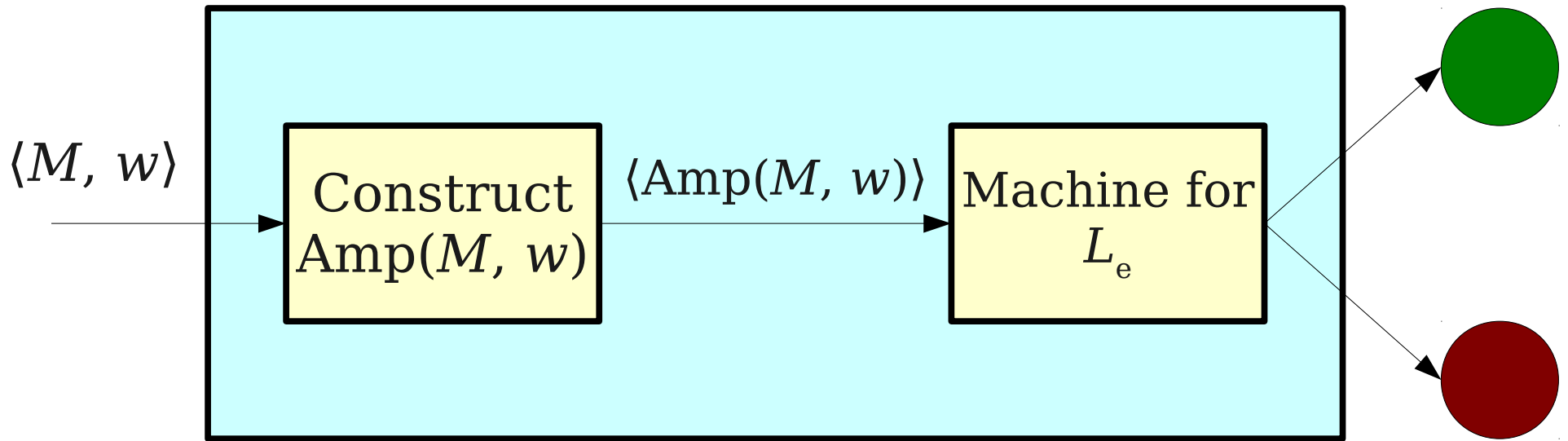
The Reduction



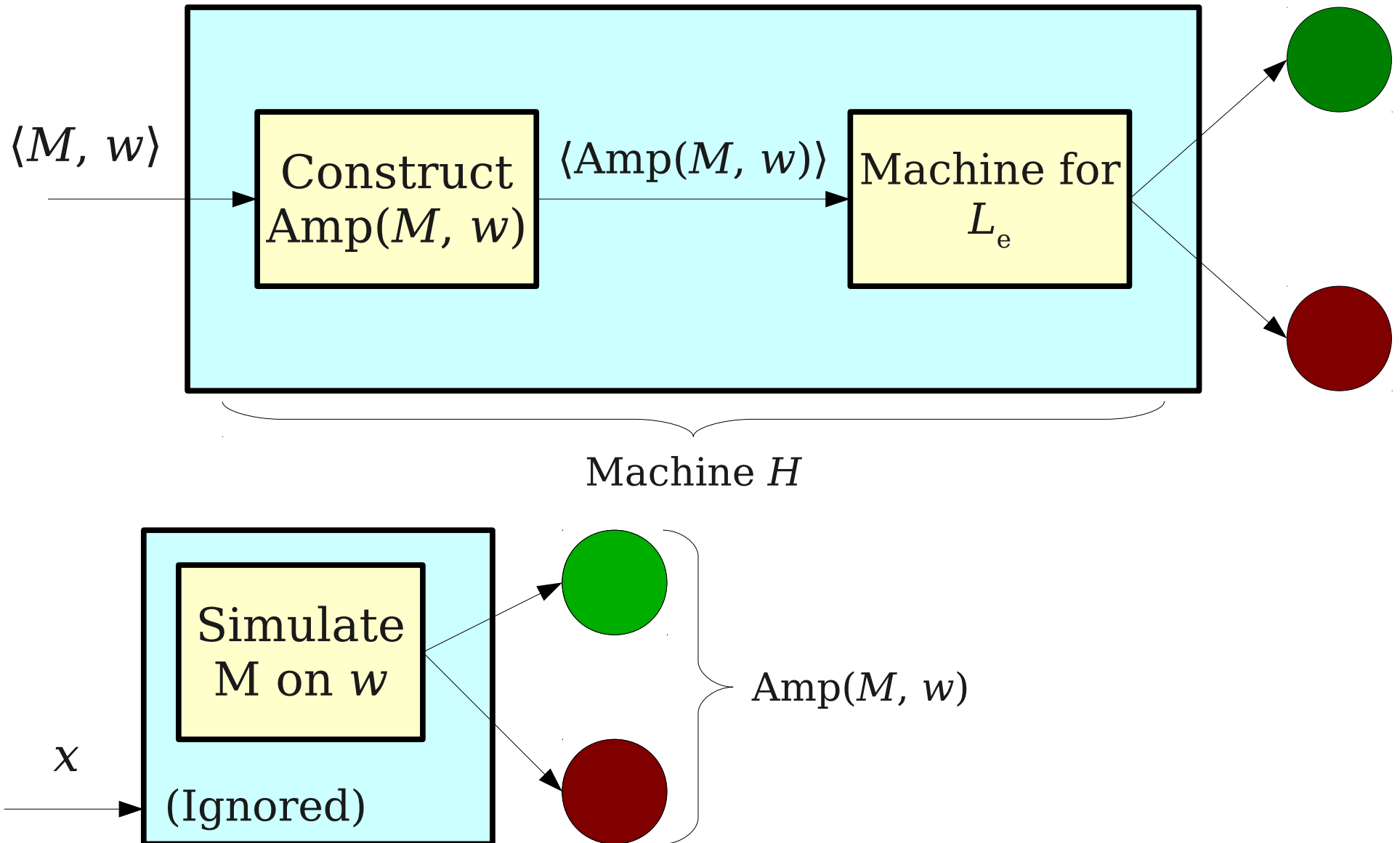
The Reduction



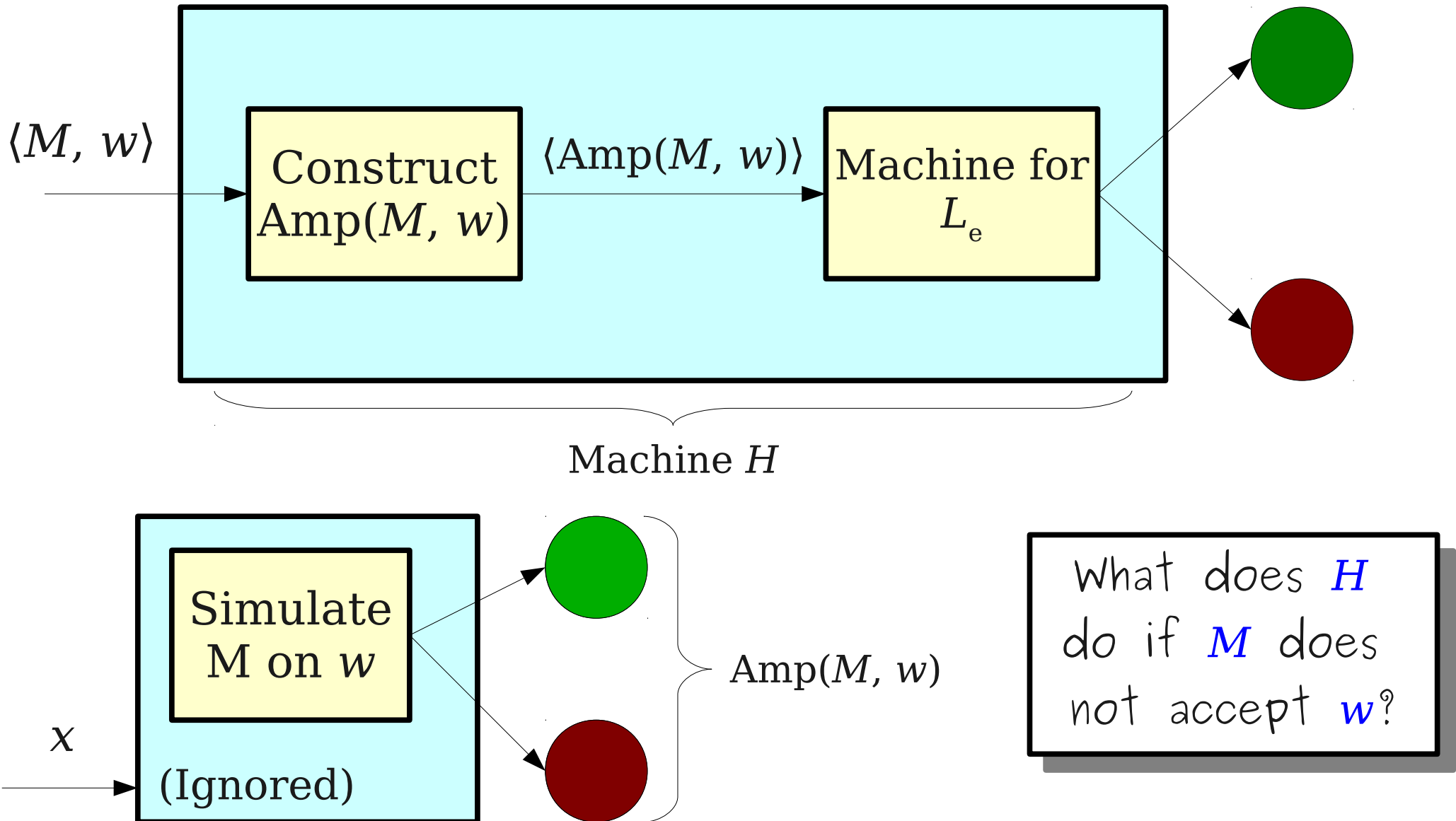
The Reduction



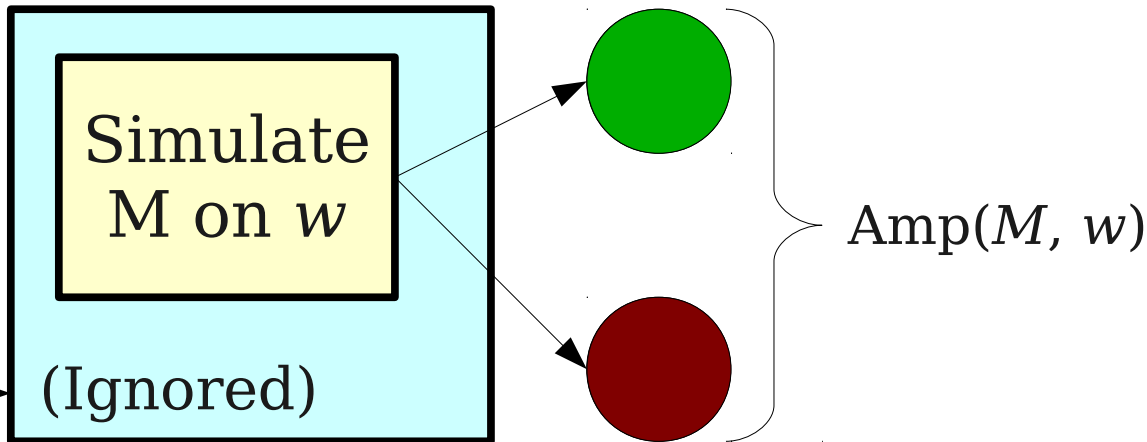
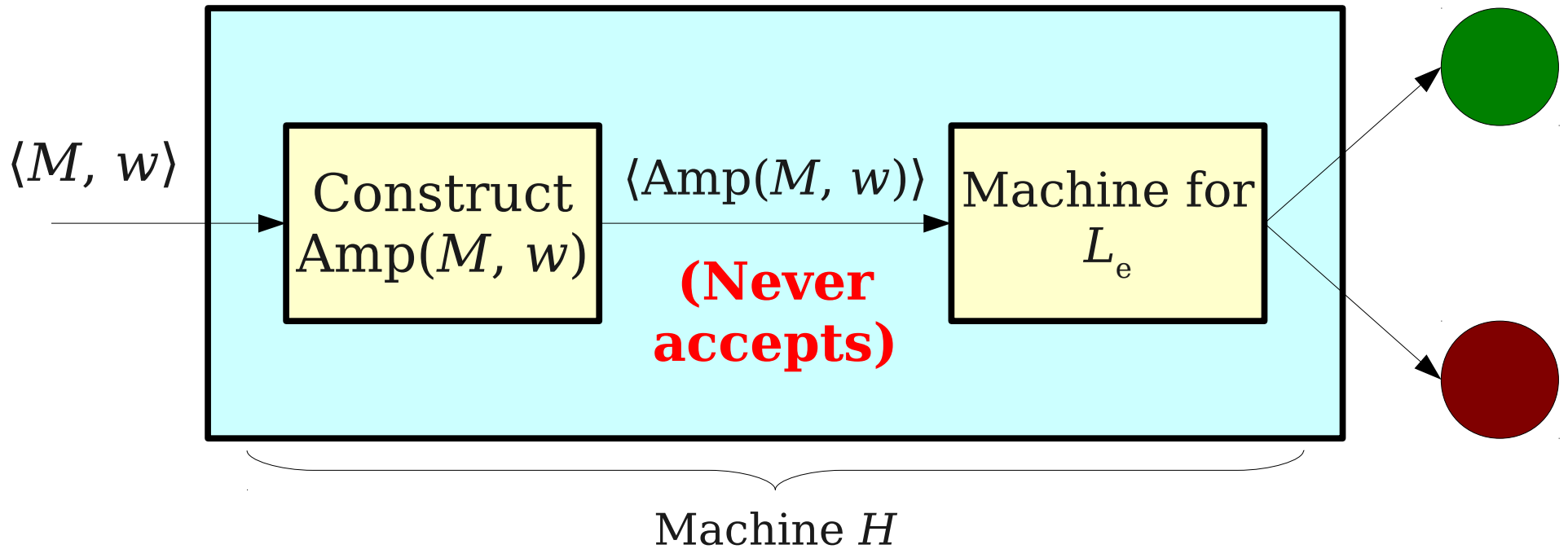
The Reduction



The Reduction

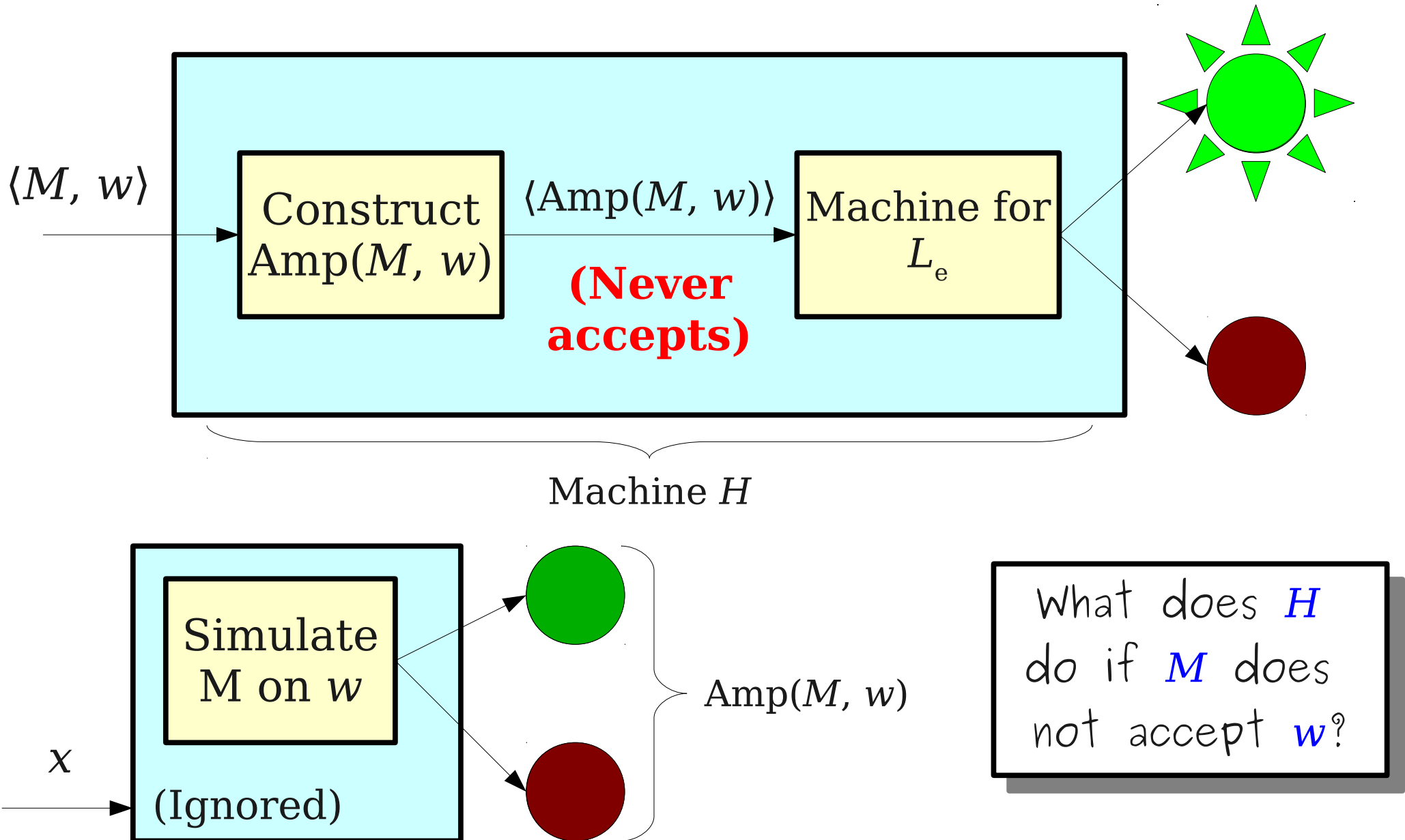


The Reduction

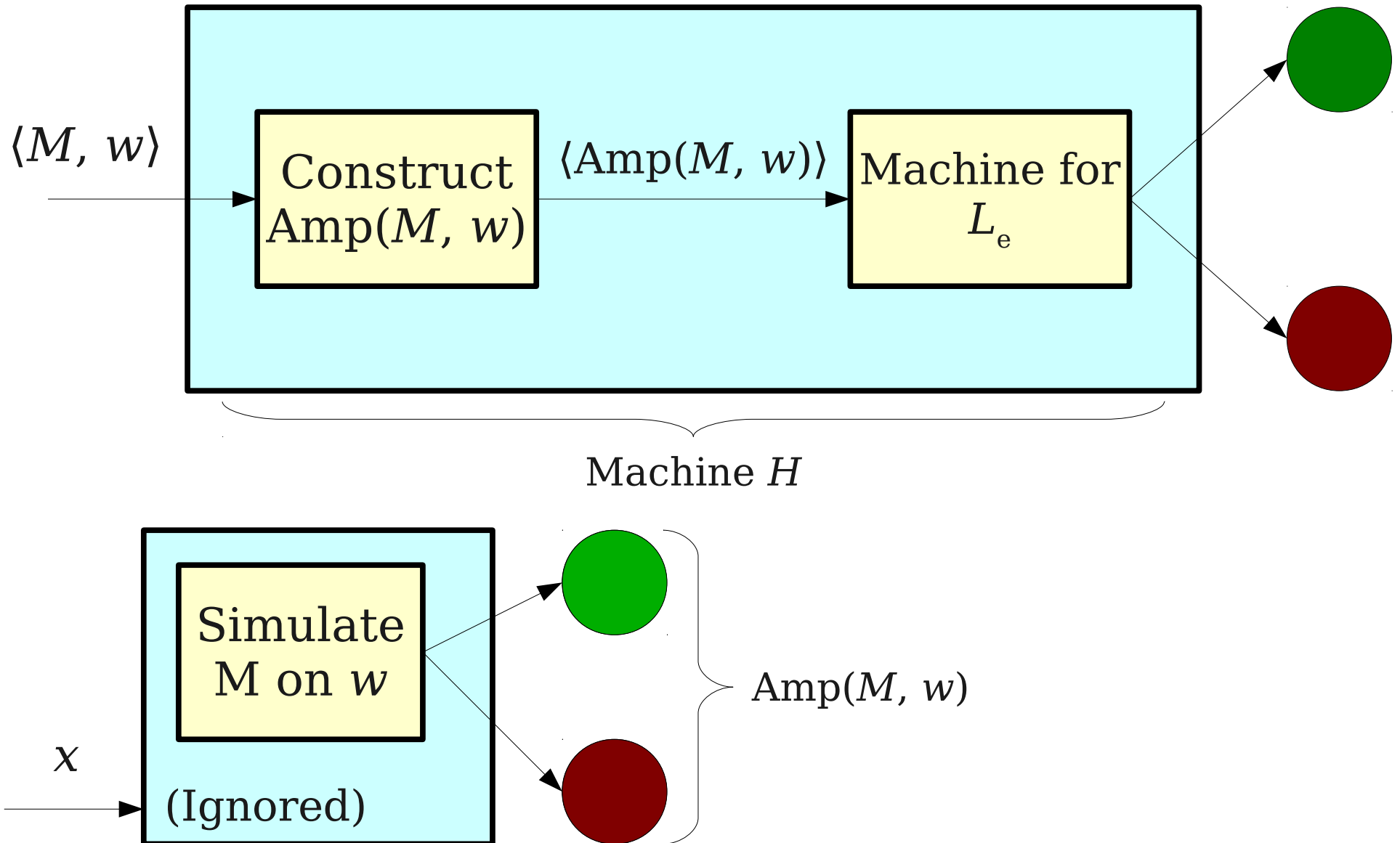


What does H do if M does not accept w ?

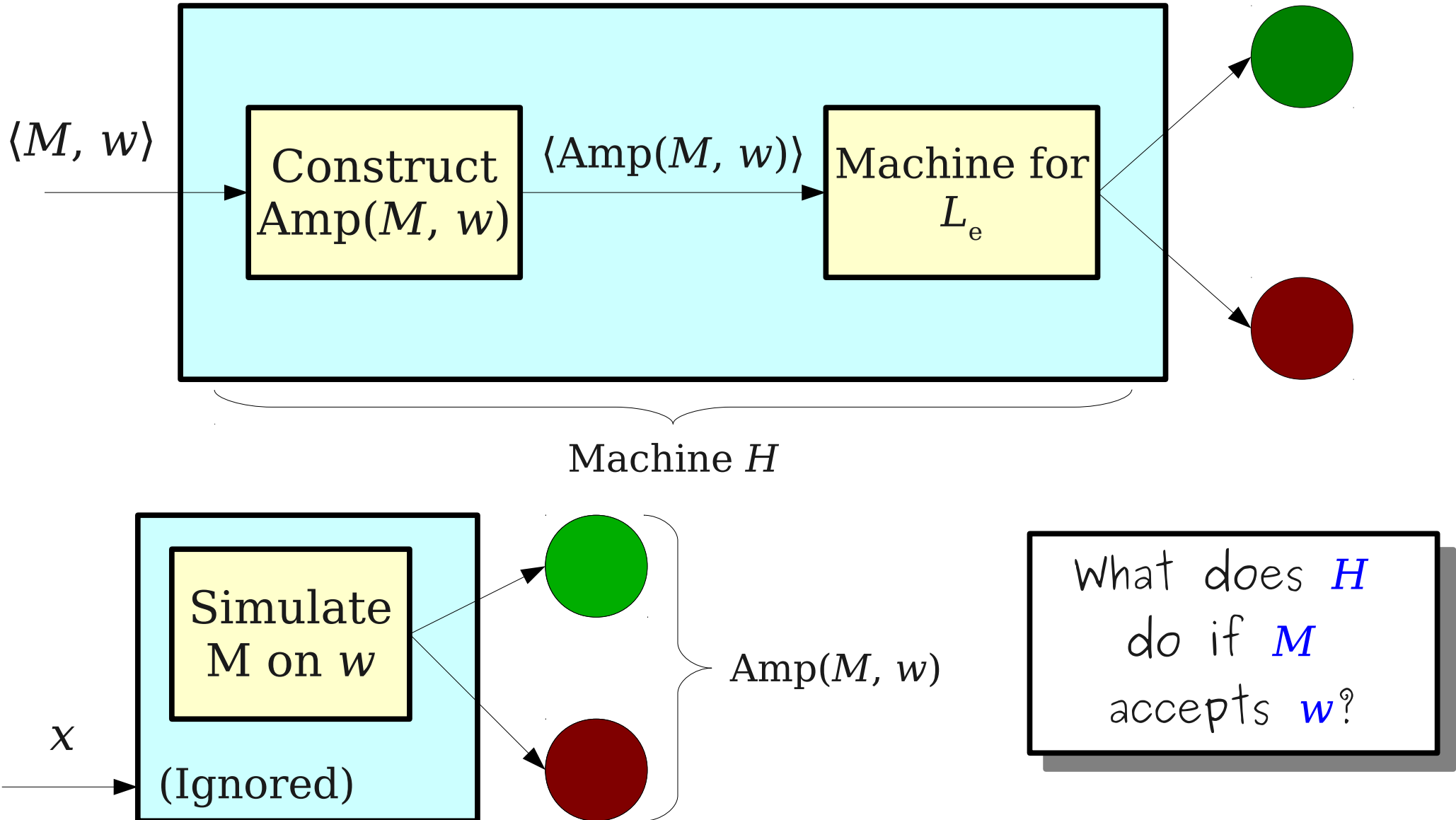
The Reduction



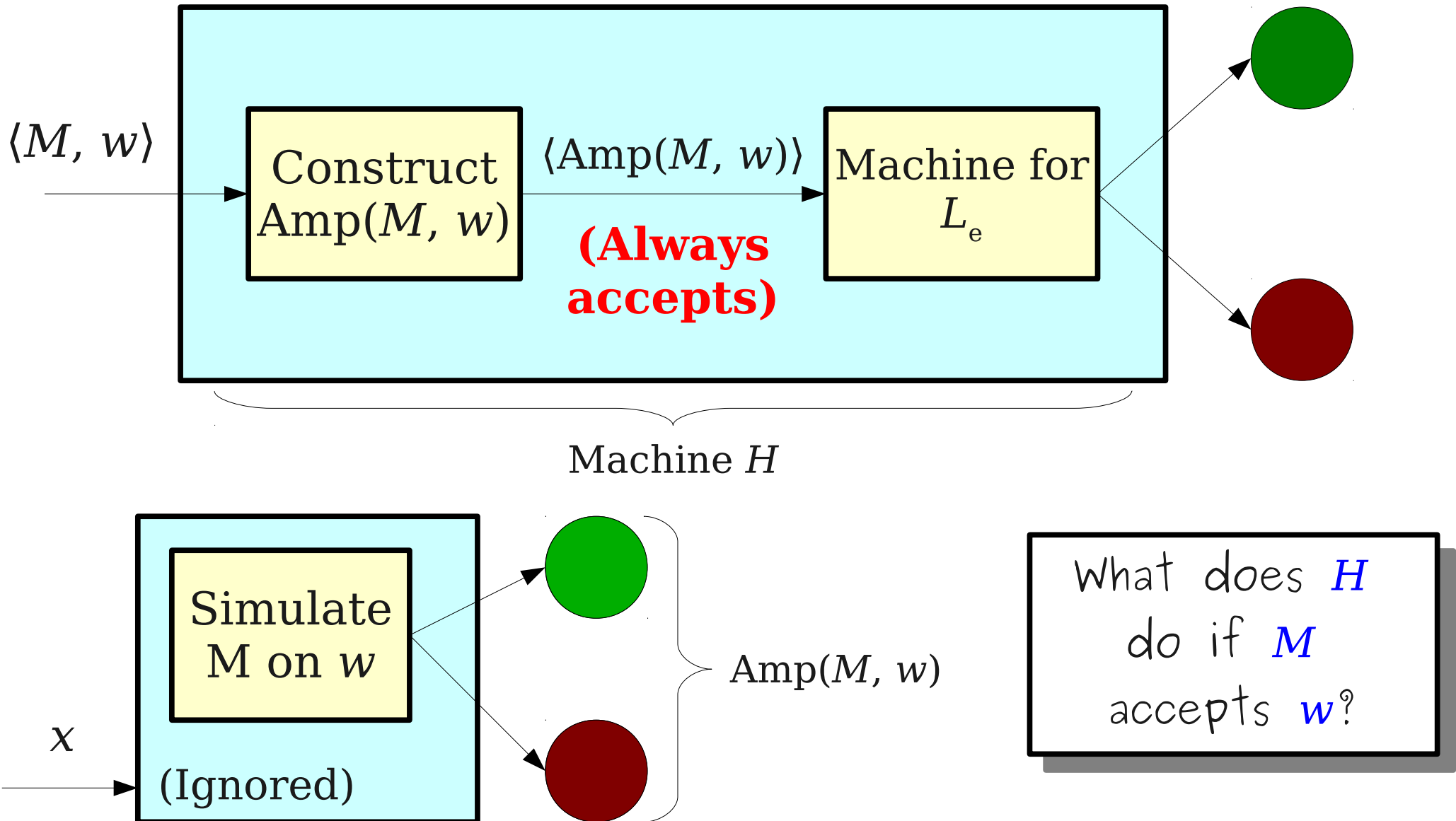
The Reduction



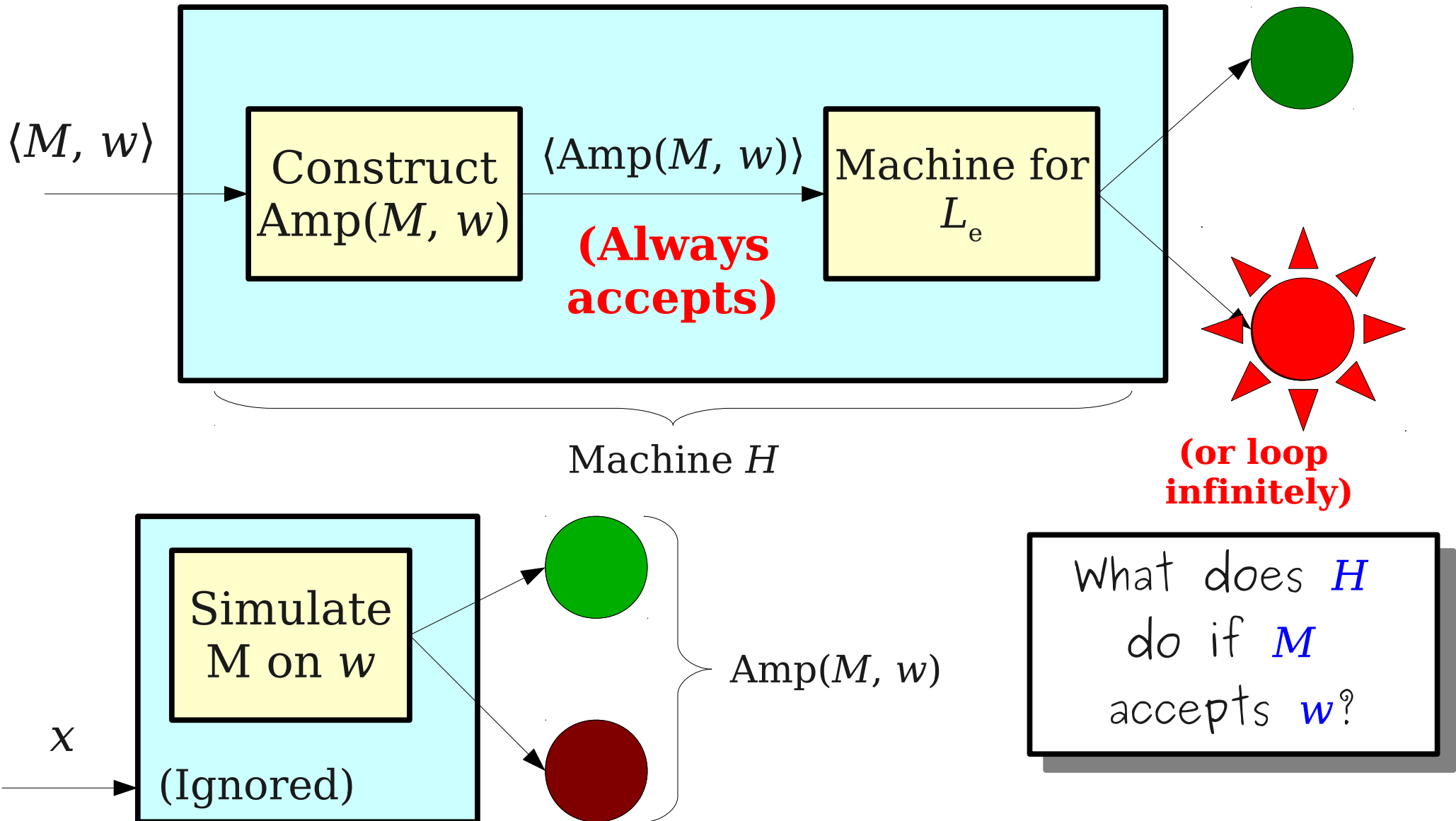
The Reduction



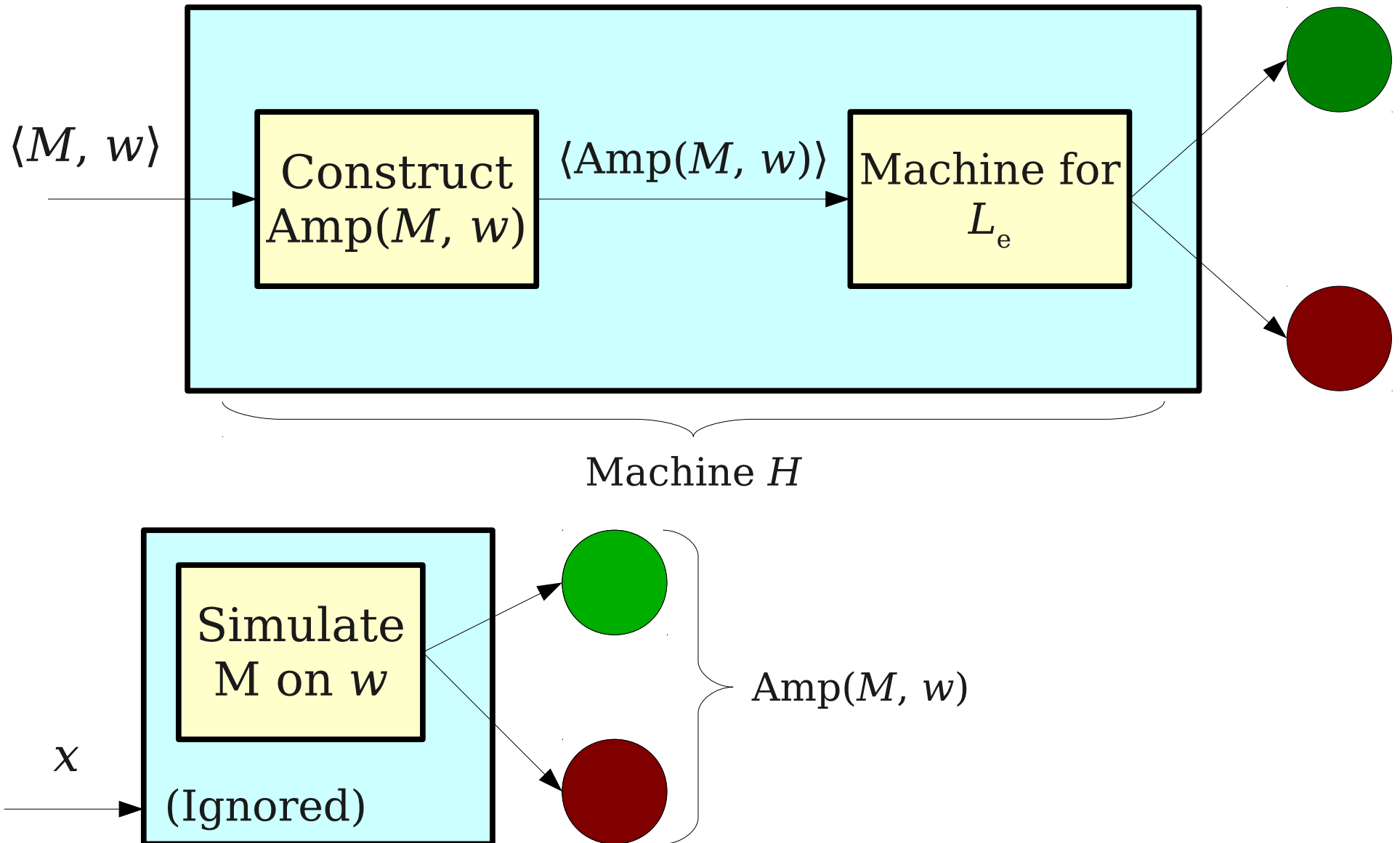
The Reduction



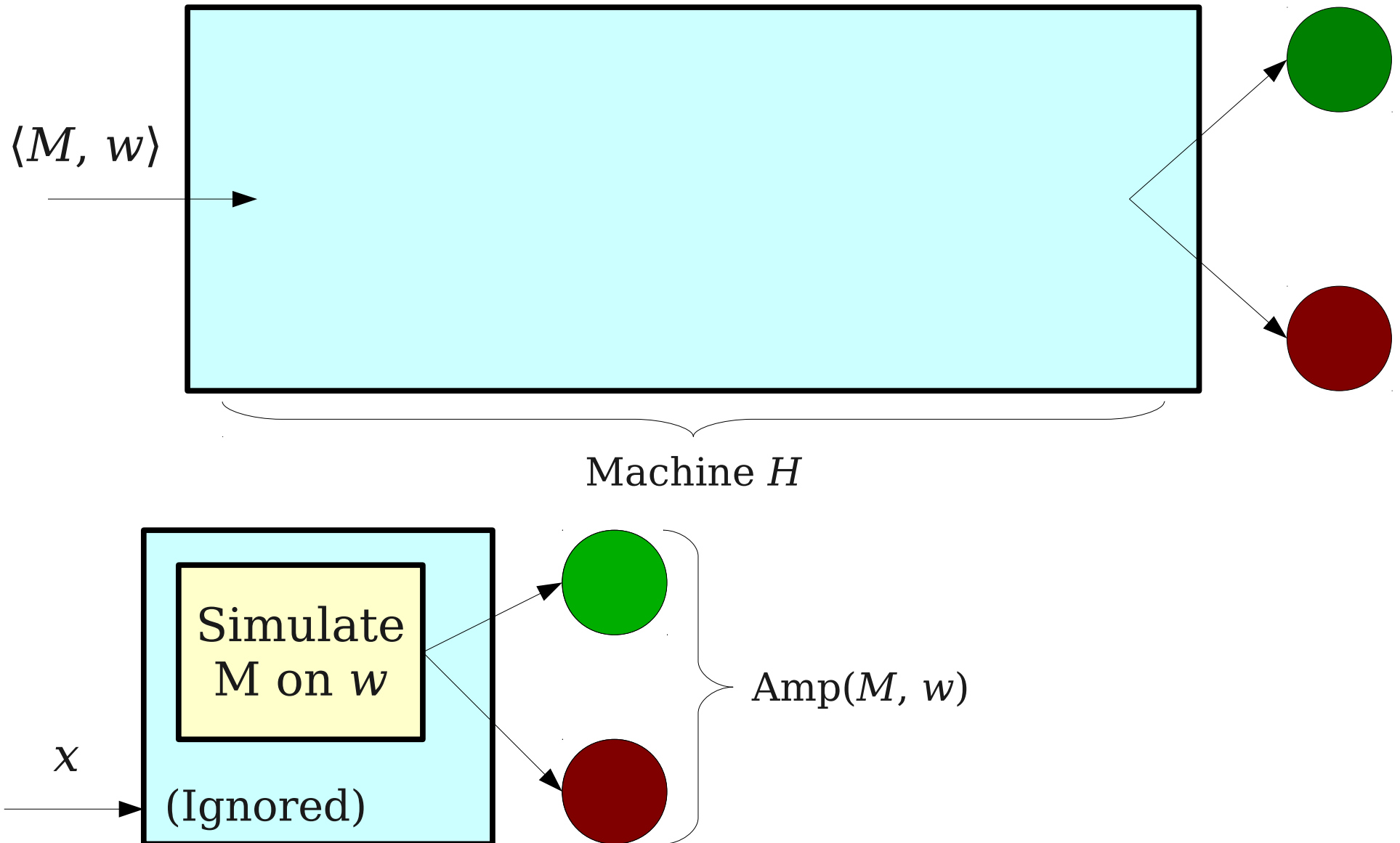
The Reduction



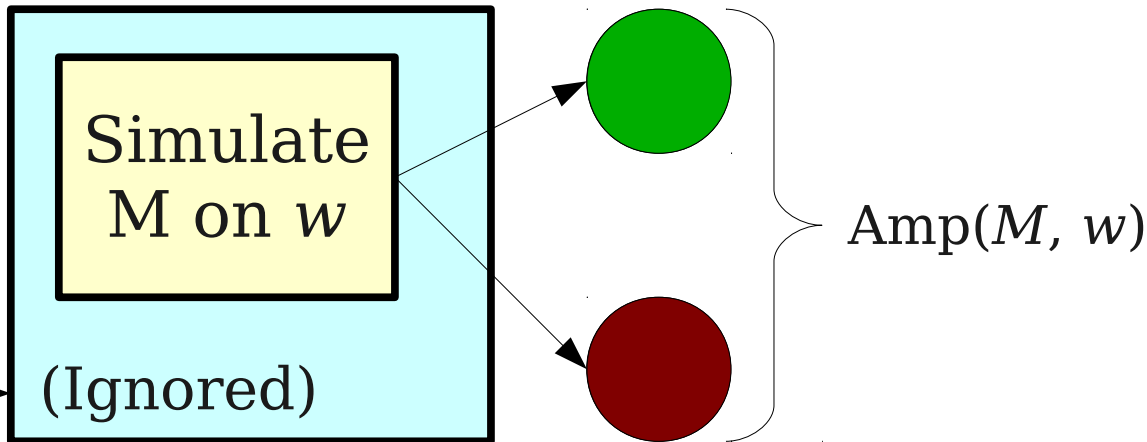
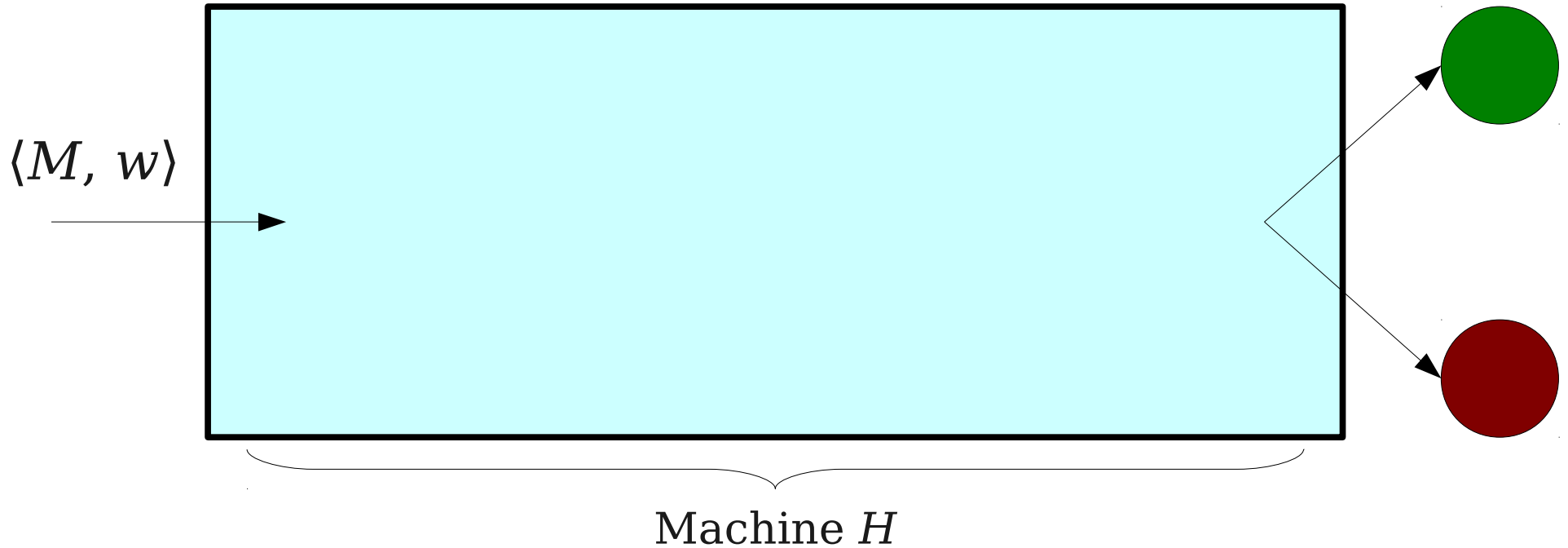
The Reduction



The Reduction

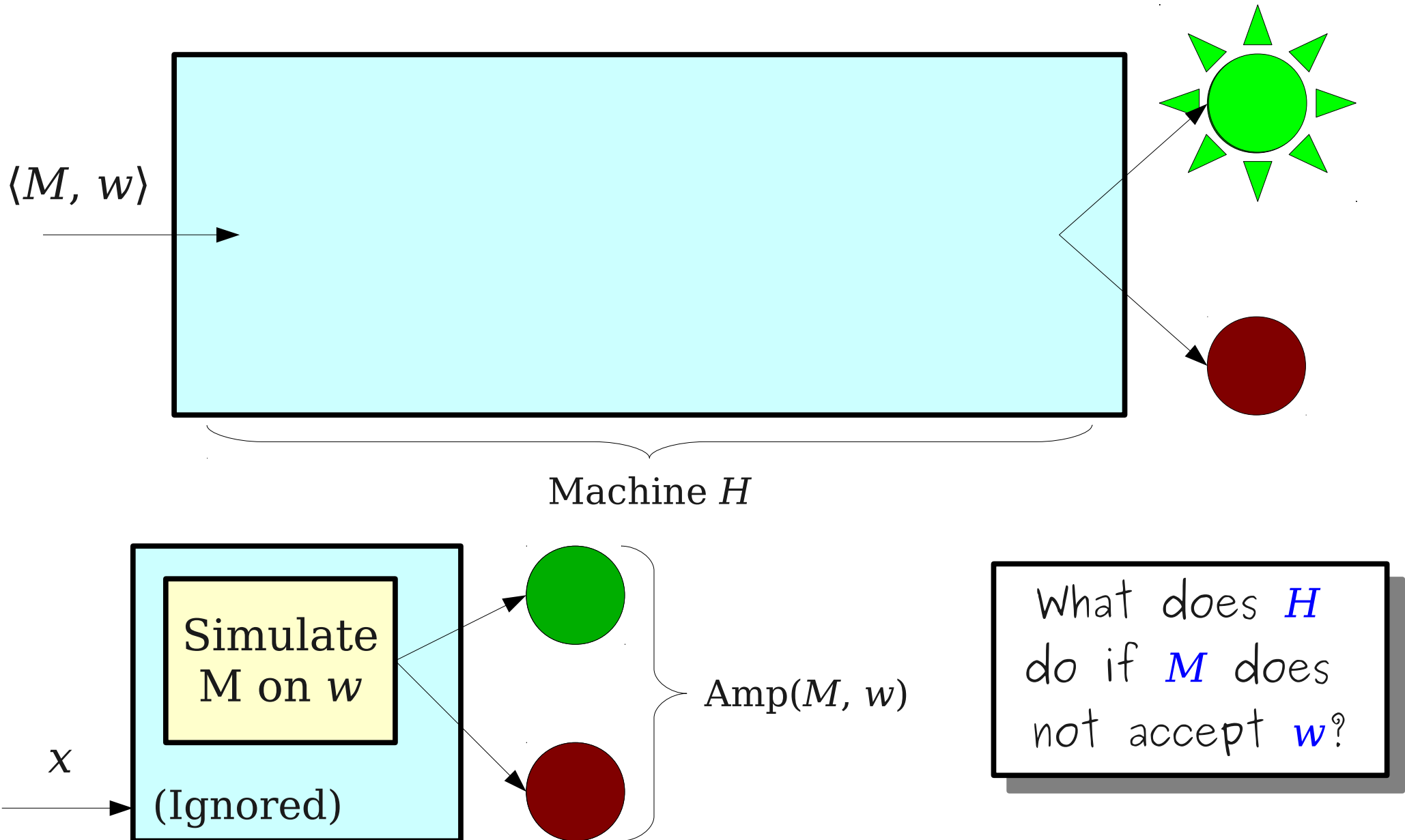


The Reduction

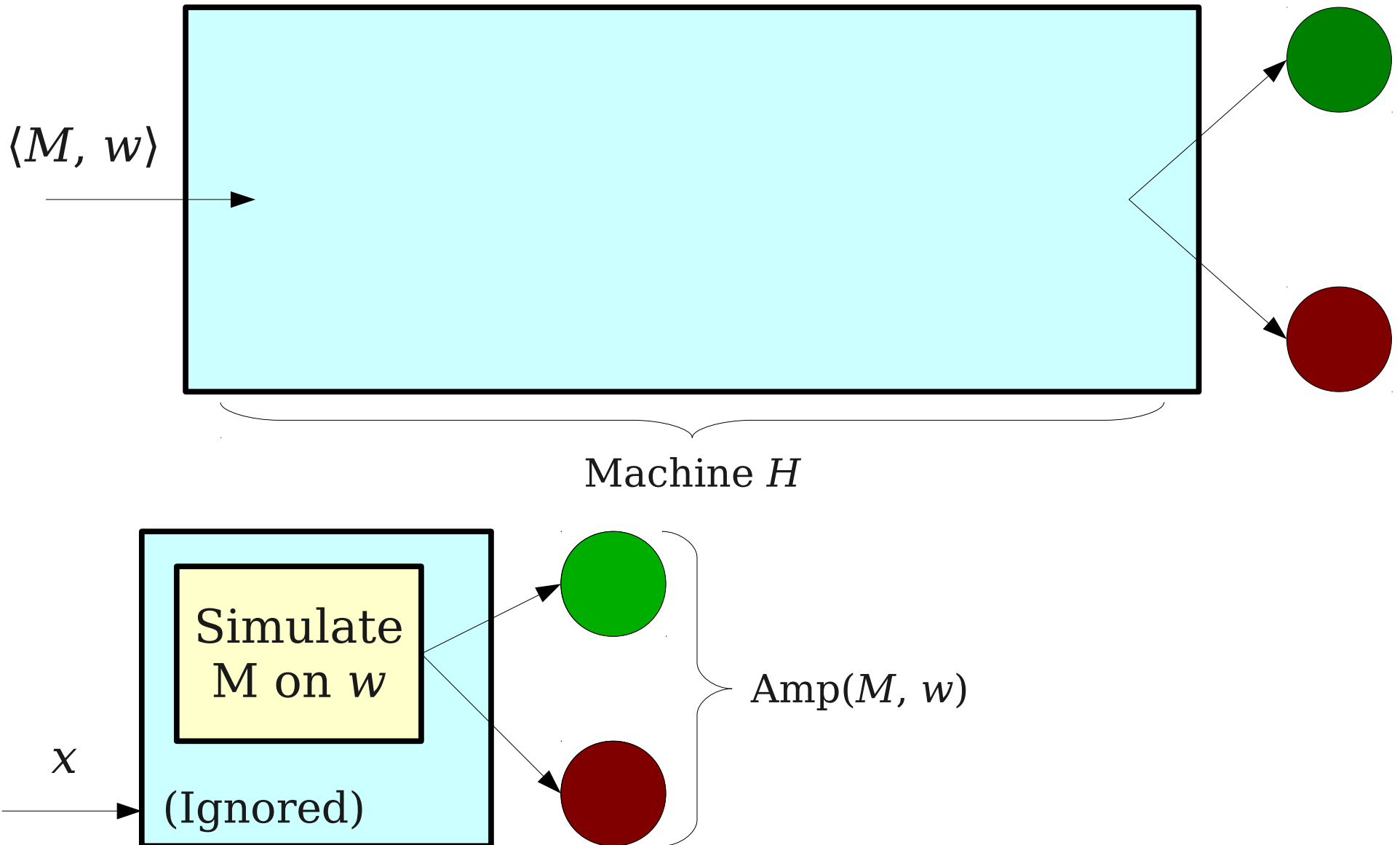


What does H do if M does not accept w ?

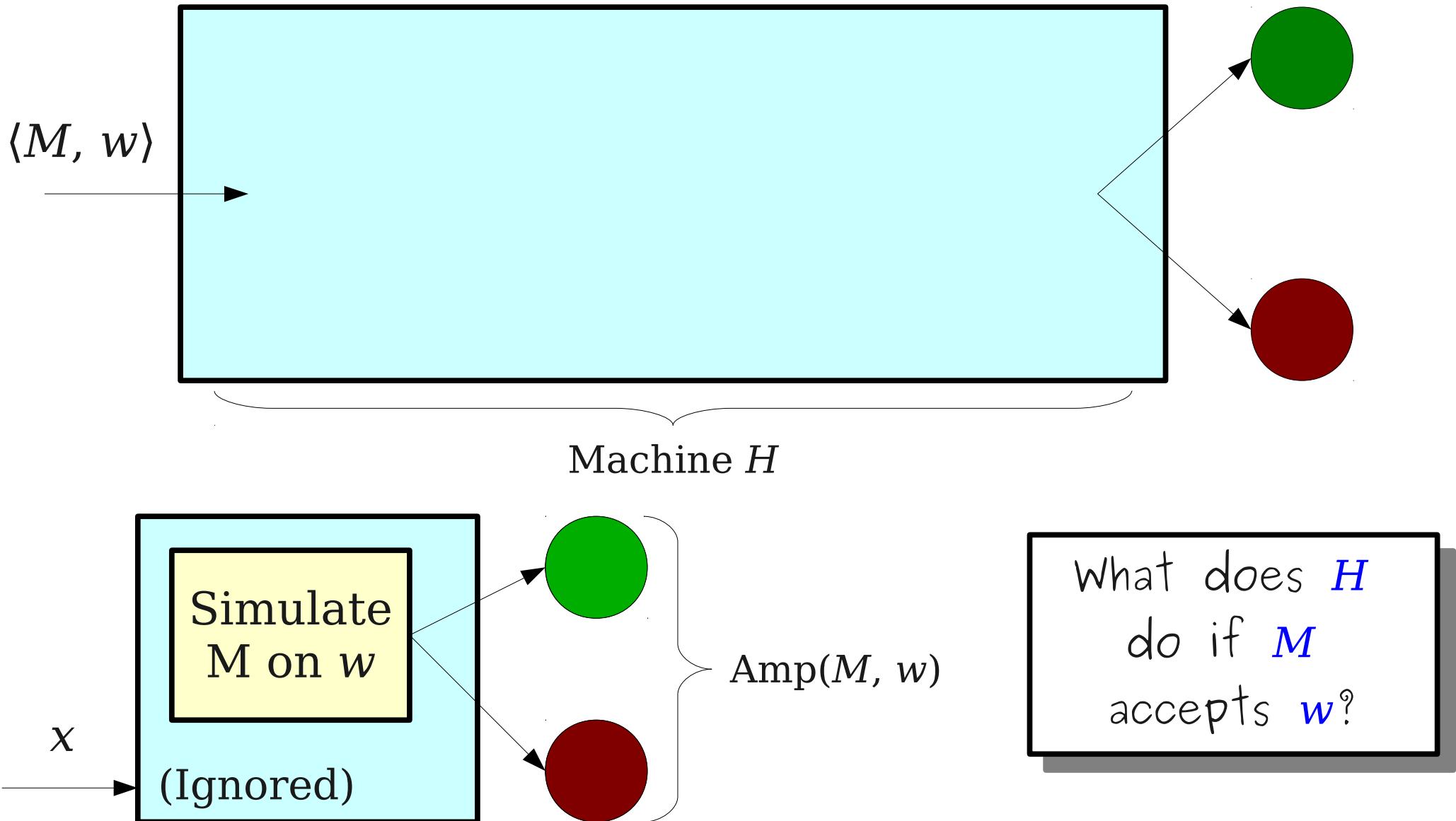
The Reduction



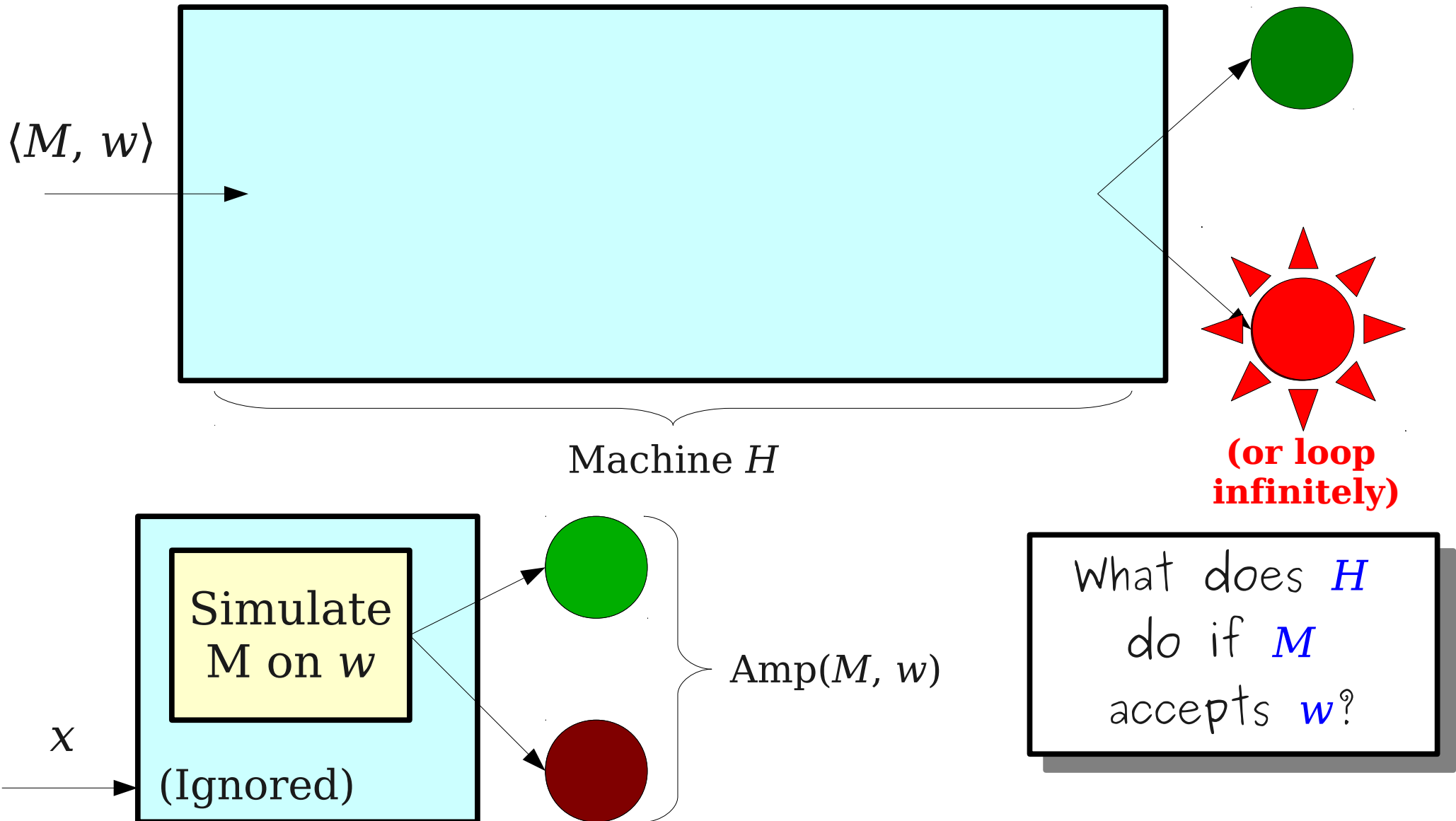
The Reduction



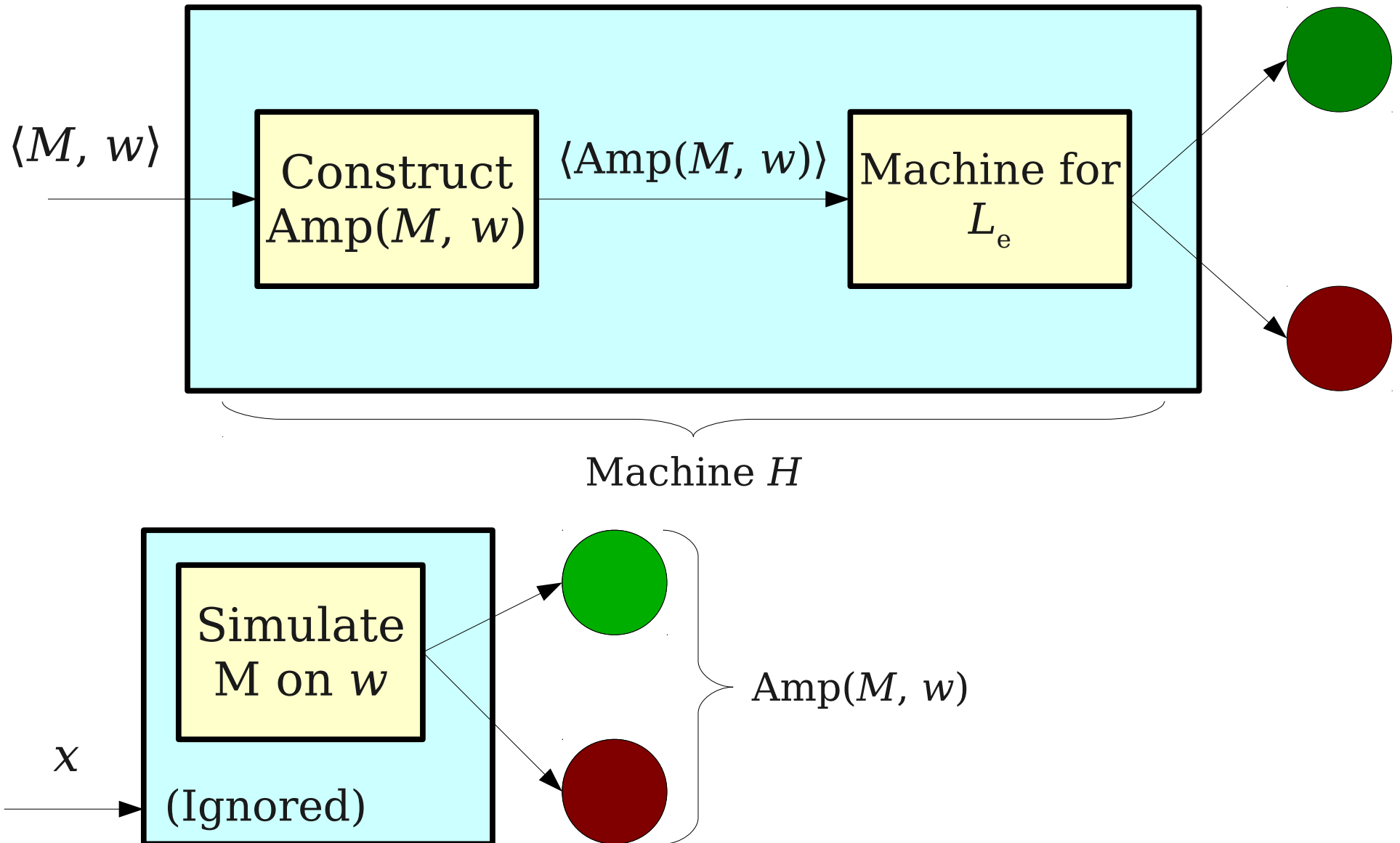
The Reduction



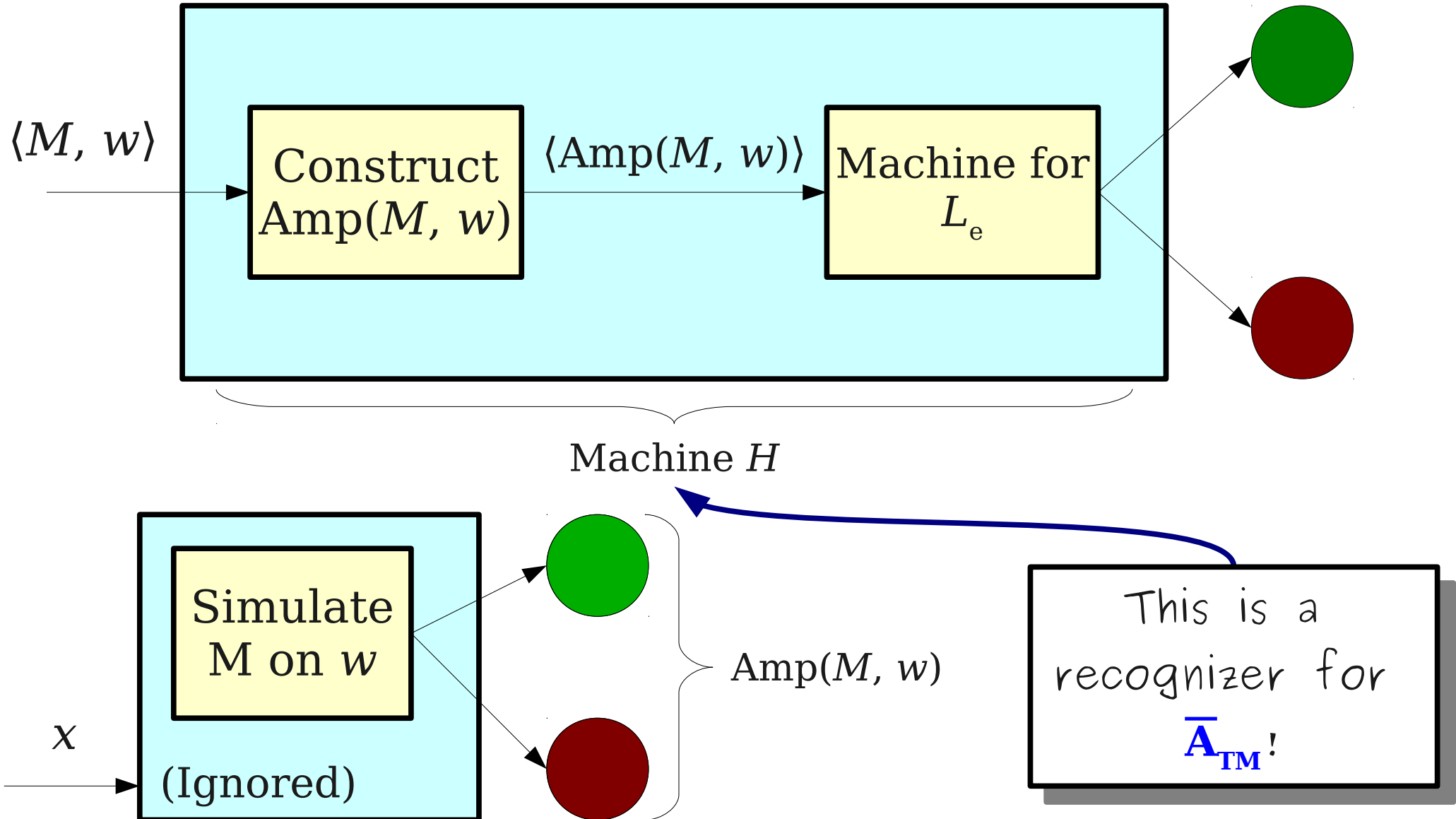
The Reduction



The Reduction



The Reduction



Theorem: $L_e \notin \mathbf{RE}$

Proof: We will prove $\overline{A}_{\text{TM}} \leq_M L_e$. Since $\overline{A}_{\text{TM}} \notin \mathbf{RE}$, this proves that $L_e \notin \mathbf{RE}$, as required. To do so, we will exhibit a mapping reduction from \overline{A}_{TM} to L_e . For any TM/string pair $\langle M, w \rangle$, let $f(\langle M, w \rangle) = \langle \text{Amp}(M, w) \rangle$. By our earlier theorem, this function is computable.

We claim this is a mapping reduction from \overline{A}_{TM} to L_e . To prove this, we will prove that $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff $\langle \text{Amp}(M, w) \rangle \in L_e$. By definition of \overline{A}_{TM} , we see $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff M does not accept w . By our earlier theorem, M does not accept w iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. Finally, by definition of L_e , we see $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$ iff $\langle \text{Amp}(M, w) \rangle \in L_e$. Taken together, we see that $\langle M, w \rangle \in \overline{A}_{\text{TM}}$ iff $\langle \text{Amp}(M, w) \rangle \in L_e$, so f is a mapping reduction from \overline{A}_{TM} to L_e . Therefore, we see $\overline{A}_{\text{TM}} \leq_M L_e$, as required. ■

A Math Joke



Time-Out For Announcements

Problem Set 6 Graded

- On-time Problem Set 6's have all been graded and should be returned after lecture today.
 - Online submissions: contact us if you don't hear back soon.
- Late Problem Set 6's will be returned this Wednesday.

Problem Set 8 Out

- Problem Set 8 goes out right now. It's due the Monday after Thanksgiving break (December 2).
- Some contradictory information:
 - This is the last problem set on which you can use a late period.
 - *We strongly* recommend that you don't, since you'll be pinched trying to finish Problem Set 9 if you do.
- TAs and I will figure out an OH schedule during Thanksgiving week.

Your Questions

“The fact we can't create a TM for \overline{A}_{TM} and L_D is very cool. But it is tough to see why we would want to solve those problems in the first place – what are problems that we actually want to solve but can't, because of limits of computability?”

“Aren't there some cases where we can know a TM is infinite looping? Couldn't we modify the U_{TM} so it keeps a record of IDs and then if it sees the same one twice know it was in a loop? This doesn't guarantee to find all loops, but would it be useful?”

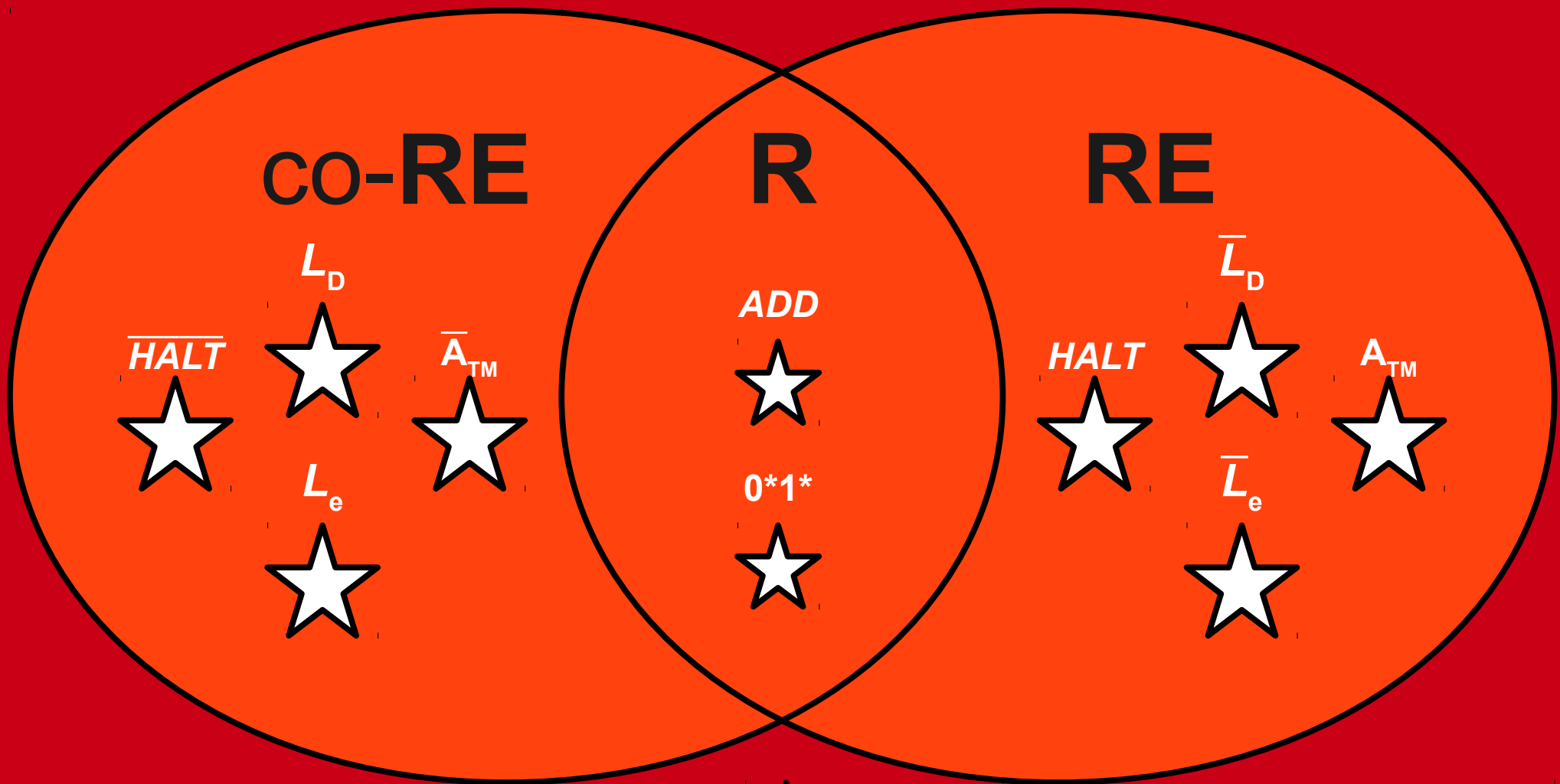
“What's the difference between a language being decidable and having a decider for a language?”

“The generalized hailstone sequence terminating is proven to be undecidable (http://link.springer.com/chapter/10.1007%2F978-3-540-72504-6_49).

What purpose is there to prove something as undecidable? Is undecidable better than not solvable?”

Back to CS103

The Limits of Computability



What's out here?

RE \cup **co-RE** is Not Everything

- Using the same reasoning as the first day of lecture, we can show that there must be problems that are neither **RE** nor **co-RE**.
- There are more sets of strings than TMs.
- There are more sets of strings than twice the number of TMs.
- What do these languages look like?

TM Equality

- There are infinitely many pairs of Turing machines with the same language as one another.

- Good exercise: think about why this is.

- Consider the following language:

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

- Questions:

- Is $\mathbf{EQ_{TM}} \in \mathbf{co-RE}$?

- Is $\mathbf{EQ_{TM}} \in \mathbf{RE}$?

Is $EQ_{TM} \in \text{co-RE}$?

- Intuitively, would we expect EQ_{TM} to be a **co-RE** language?
- Suppose TM M_1 accepts a string w . We'd need to know whether M_2 accepts w as well.
- Co-recognizing this would require us to have a corecognizer that detects whether $\langle M_2, w \rangle \in A_{TM}$, but that's not an **co-RE** language!
- Our guess: *EQ_{TM} is probably not **co-RE**.*

Proving $EQ_{TM} \notin \text{co-RE}$

- To prove that $EQ_{TM} \notin \text{co-RE}$, we can try to find a language L where
 - $L \notin \text{co-RE}$, and
 - $L \leq_M EQ_{TM}$
- A good candidate would be something like A_{TM} , which is a “canonical” non-co-RE languages.
- **Goal:** Prove $A_{TM} \leq_M EQ_{TM}$.

Proving $A_{\text{TM}} \leq_M \text{EQ}_{\text{TM}}$

- Goal: Find a computable function f where

$$\langle M, w \rangle \in A_{\text{TM}} \text{ iff } f(\langle M, w \rangle) \in \text{EQ}_{\text{TM}}$$

- Since EQ_{TM} is a language of pairs of TMs, let's assume $f(\langle M \rangle) = \langle M_1, M_2 \rangle$. Then we want to pick M_1 and M_2 such that

$$\langle M, w \rangle \in A_{\text{TM}} \text{ iff } \langle M_1, M_2 \rangle \in \text{EQ}_{\text{TM}}$$

- Substituting definitions, we want

$$M \text{ accepts } w \text{ iff } \mathcal{L}(M_1) = \mathcal{L}(M_2)$$

- What do we do now?

Using the Amplifier

- We want

M accepts w iff $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

- What happens if we pick M_1 to be $\text{Amp}(M, w)$?
 - If M accepts w , then $\mathcal{L}(M_1) = \Sigma^*$.
 - If M does not accept w , then $\mathcal{L}(M_1) = \emptyset$.
- Choose M_1 to be the amplifier machine and M_2 to be any TM with language Σ^* . Then the above statement is true!

What's Going On?

- Suppose we have an oracle for EQ_{TM} .
- We want to know whether M accepts w .
- To do this:
 - Find a TM S we know has language Σ^* .
 - Ask the oracle “does TM $Amp(M, w)$ have the same language as TM S ?”
 - If so, then M accepts w .
 - If not, then M does not accept w .

Theorem: $EQ_{TM} \notin \text{co-RE}$.

Proof: We will prove $A_{TM} \leq_M EQ_{TM}$. Since $A_{TM} \notin \text{co-RE}$, this proves that $EQ_{TM} \notin \text{co-RE}$. To show $A_{TM} \leq_M EQ_{TM}$, we will exhibit a mapping reduction from A_{TM} to EQ_{TM} .

For any TM/string pair $\langle M, w \rangle$, define $f(\langle M, w \rangle)$ to be the pair of TMs $\langle \text{Amp}(M, w), S \rangle$, where S is the TM “On input x , accept x .” This function is computable, and note that $\mathcal{L}(S) = \Sigma^*$.

We claim that $\langle M, w \rangle \in A_{TM}$ iff $\langle \text{Amp}(M, w), S \rangle \in EQ_{TM}$. To see this, note by definition of A_{TM} that $\langle M, w \rangle \in A_{TM}$ iff M accepts w . By our earlier theorem, M accepts w iff $\mathcal{L}(\text{Amp}(M, w)) = \Sigma^*$. Since $\mathcal{L}(S) = \Sigma^*$, we see M accepts w iff $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(S)$. Finally, by definition of EQ_{TM} , $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(S)$ iff $\langle \text{Amp}(M, w), S \rangle \in EQ_{TM}$.

Collectively, we see $\langle M, w \rangle \in A_{TM}$ iff $\langle \text{Amp}(M, w), S \rangle \in EQ_{TM}$.

Thus f is a mapping reduction from A_{TM} to EQ_{TM} , so

$A_{TM} \leq_M EQ_{TM}$, as required. ■

Is $EQ_{TM} \in \mathbf{RE}$?

- Intuitively, would we expect EQ_{TM} to be a **RE** language?
- Suppose TM M_1 doesn't accept a string w . We'd need to know whether M_2 also doesn't accept w .
- Recognizing this would require us to have a recognizer that detects whether $\langle M_2, w \rangle \in \overline{A}_{TM}$, but that's not an **RE** language!
- Our guess: *EQ_{TM} is probably not **RE**.*

Proving $\bar{A}_{\text{TM}} \leq_M \text{EQ}_{\text{TM}}$

- Goal: Find a computable function f where

$$\langle M, w \rangle \in \bar{A}_{\text{TM}} \text{ iff } f(\langle M, w \rangle) \in \text{EQ}_{\text{TM}}$$

- Since EQ_{TM} is a language of pairs of TMs, let's assume $f(\langle M \rangle) = \langle M_1, M_2 \rangle$. Then we want to pick M_1 and M_2 such that

$$\langle M, w \rangle \in \bar{A}_{\text{TM}} \text{ iff } \langle M_1, M_2 \rangle \in \text{EQ}_{\text{TM}}$$

- Substituting definitions, we want

$$\mathbf{M \text{ does not accept } w \text{ iff } \mathcal{L}(M_1) = \mathcal{L}(M_2)}$$

- What do we do now?

Using the Amplifier

- We want

M does not accept w iff $\mathcal{L}(M_1) = \mathcal{L}(M_2)$

- What happens if we pick M_1 to be $\text{Amp}(M, w)$?
 - If M accepts w , then $\mathcal{L}(M_1) = \Sigma^*$.
 - If M does not accept w , then $\mathcal{L}(M_1) = \emptyset$.
- Choose M_1 to be the amplifier machine and M_2 to be any TM with language \emptyset . Then the above statement is true!

What's Going On?

- Suppose we have an oracle for EQ_{TM} .
- We want to know whether M accepts w .
- To do this:
 - Find a TM E we know has language \emptyset .
 - Ask the oracle “does TM $\text{Amp}(M, w)$ have the same language as TM E ?”
 - If so, then M does not accept w .
 - If not, then M accepts w .

Theorem: $\text{EQ}_{\text{TM}} \notin \mathbf{RE}$.

Proof: We will prove $\bar{\text{A}}_{\text{TM}} \leq_{\text{M}} \text{EQ}_{\text{TM}}$. Since $\bar{\text{A}}_{\text{TM}} \notin \mathbf{RE}$, this proves that $\text{EQ}_{\text{TM}} \notin \mathbf{RE}$. To show $\bar{\text{A}}_{\text{TM}} \leq_{\text{M}} \text{EQ}_{\text{TM}}$, we will exhibit a mapping reduction from $\bar{\text{A}}_{\text{TM}}$ to EQ_{TM} .

For any TM/string pair $\langle M, w \rangle$, define $f(\langle M, w \rangle)$ to be the pair of TMs $\langle \text{Amp}(M, w), E \rangle$, where E is the TM “On input x , reject x .” This function is computable, and note that $\mathcal{L}(E) = \emptyset$.

We claim that $\langle M, w \rangle \in \bar{\text{A}}_{\text{TM}}$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$. To see this, note by definition of $\bar{\text{A}}_{\text{TM}}$ that $\langle M, w \rangle \in \bar{\text{A}}_{\text{TM}}$ iff M does not accept w . By our theorem, M does not accept w iff $\mathcal{L}(\text{Amp}(M, w)) = \emptyset$. Since $\mathcal{L}(E) = \emptyset$, we see M does not accept w iff $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(E)$. Finally, by definition of EQ_{TM} , $\mathcal{L}(\text{Amp}(M, w)) = \mathcal{L}(E)$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$.

Collectively, we see $\langle M, w \rangle \in \bar{\text{A}}_{\text{TM}}$ iff $\langle \text{Amp}(M, w), E \rangle \in \text{EQ}_{\text{TM}}$.

Thus f is a mapping reduction from $\bar{\text{A}}_{\text{TM}}$ to EQ_{TM} , so $\bar{\text{A}}_{\text{TM}} \leq_{\text{M}} \text{EQ}_{\text{TM}}$, as required. ■

The Limits of Computability

EQ_{TM}
★

\overline{EQ}_{TM}
★

CO-RE

R

RE

\overline{HALT} ★ \overline{A}_{TM}
 L_D ★
 L_e ★

ADD ★
 0^*1^* ★

$HALT$ ★ \overline{L}_D ★ A_{TM}
 \overline{L}_e ★