COMP0017
Computability and Complexity Theory

Fabio Zanasi
http://www0.cs.ucl.ac.uk/staff/F.Zanasi/

Lecture eight

Previously on COMP0017

We studied in depth what Turing machines **are able to do**.

Decide/recognise problems and languages.

Simulate diverse computational models, from register machines to programming languages.

Be universal and programmable.

In this lecture

We begin the study of what Turing machines **cannot do**.

We introduce our first **undecidable** problem:
the **halting** problem.

This informs us about the **limits of algorithmic computation**.

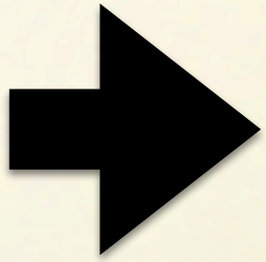
Remember the Church-Turing thesis?

Any problem that can be solved by a human being though a well-defined step-by-step procedure can be solved by a Turing machine.

Implications of Church-Turing thesis

Explored last week

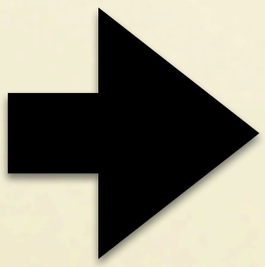
Solvable by a well-defined step-by-step procedure



Solvable by a Turing Machine

To be explored from now on.

Unsolvable by a Turing Machine



Unsolvable by a well-defined step-by-step procedure

Not by a C++/Python/… program
Not by a quantum computer
Etc.

Languages and TMs: a reminder

A TM $\mathcal{M}$ **decides** a language (decision problem) L if:

- When $x \in L$, then $\mathcal{M}$ accepts x (= halts in state Y).
- When $x \notin L$, then $\mathcal{M}$ rejects x (= halts in state N).

A TM $\mathcal{M}$ **recognises** a language (decision problem) L if:

- When $x \in L$, then $\mathcal{M}$ halts on x.
- When $x \notin L$, then $\mathcal{M}$ does not halt on x.

Degrees of (un)solvability

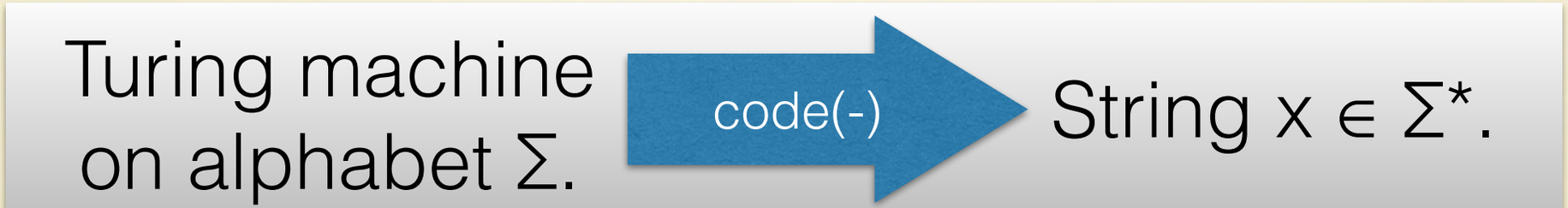
Decided by a TM	~	Solvable (There exists an algorithm answering “Yes” and "No")
Not decided by any TM but recognised by a TM	~	Unsolvable (Semi-decidable: no algorithm has all the “No" answers)
Not even recognised by any TM	~	Unsolvable (Completely undecidable: any algorithm will miss both “Yes" and “No" answers)

An unsolvable problem

We now show a language that is unsolvable: it is recognisable but **not decidable**.

An unsolvable problem

First, assume an encoding code(-):



The translation used to define the Universal Turing machine is an example of such encoding.

The halting problem

We define the language of the **halting problem**:

$HALT = \{ \langle y, x \rangle \in \Sigma^* x \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Theorem The halting problem is recognisable but undecidable.

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

The easy part: *HALT* is recognisable.

Exercise: how would you sketch a Turing machine recognising *HALT*?

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

The remaining part: show that *HALT* is not decidable.

We make a *proof by contradiction*.

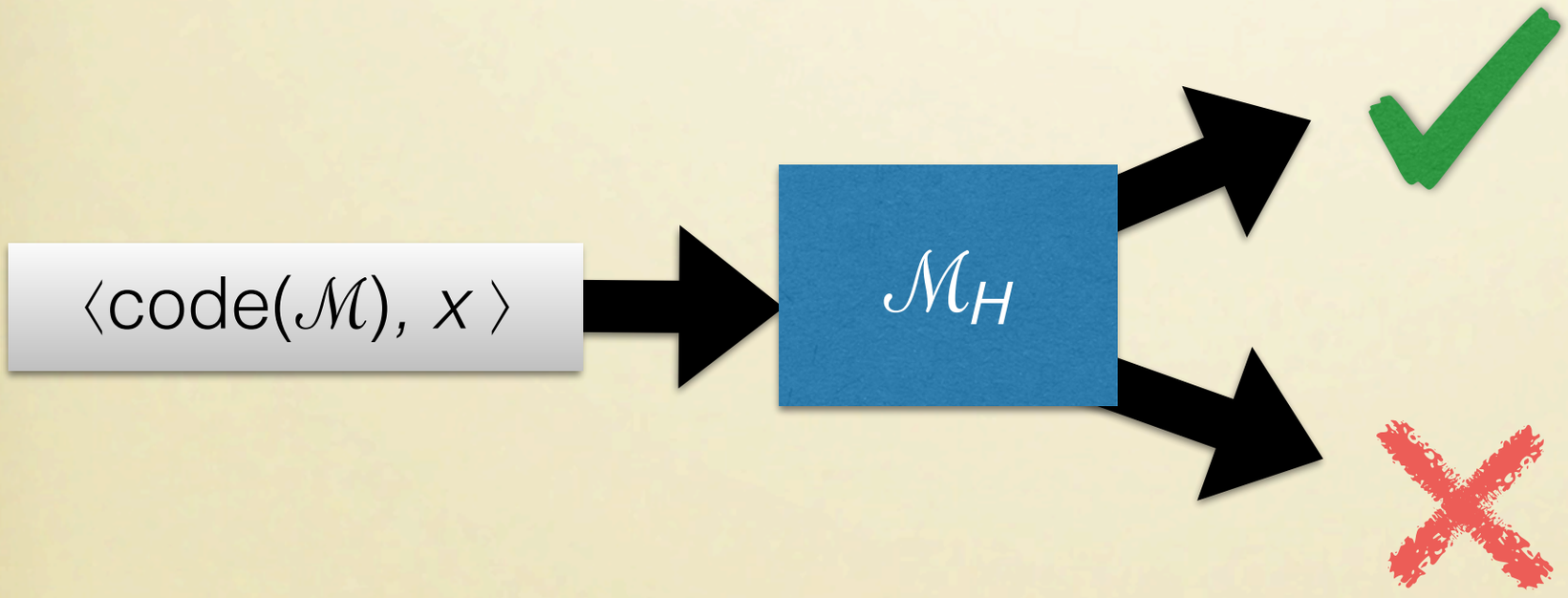
So suppose that *HALT is* decidable, and call $\mathcal{M}_H$ the TM deciding it.

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

When $y = \text{code}(\mathcal{M})$ for a certain $\mathcal{M}$:



If $\mathcal{M}$ halts on x.

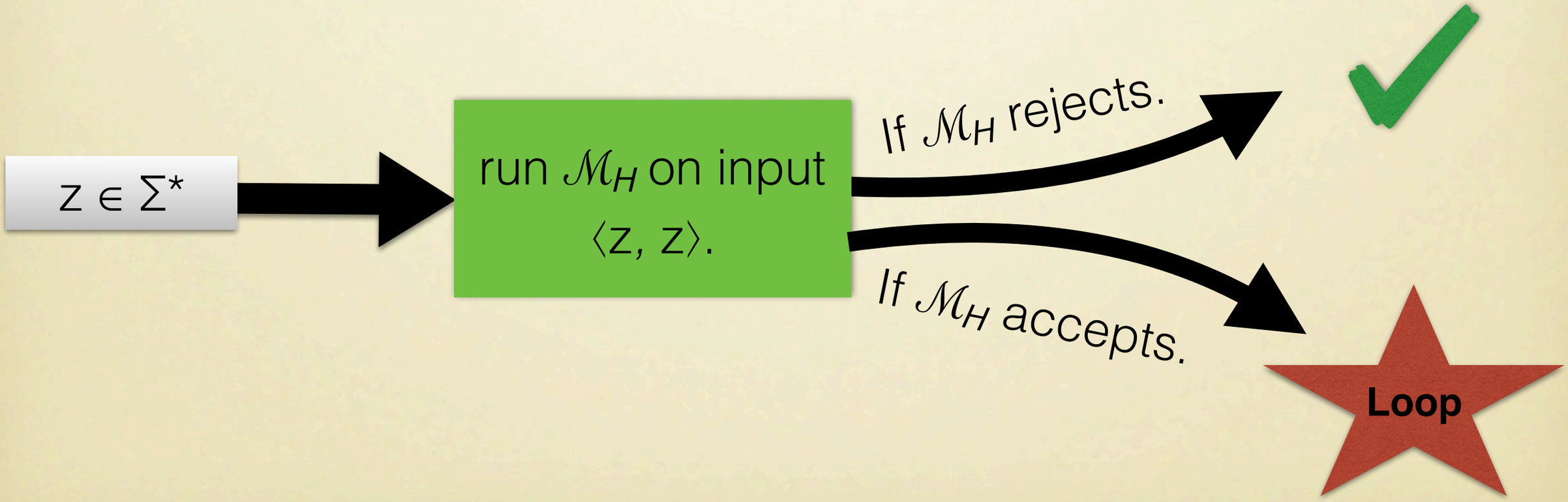
If $\mathcal{M}$ does not halt on x.

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* x \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

Now we are able to define a new TM called $\mathcal{M}'$ as follows.

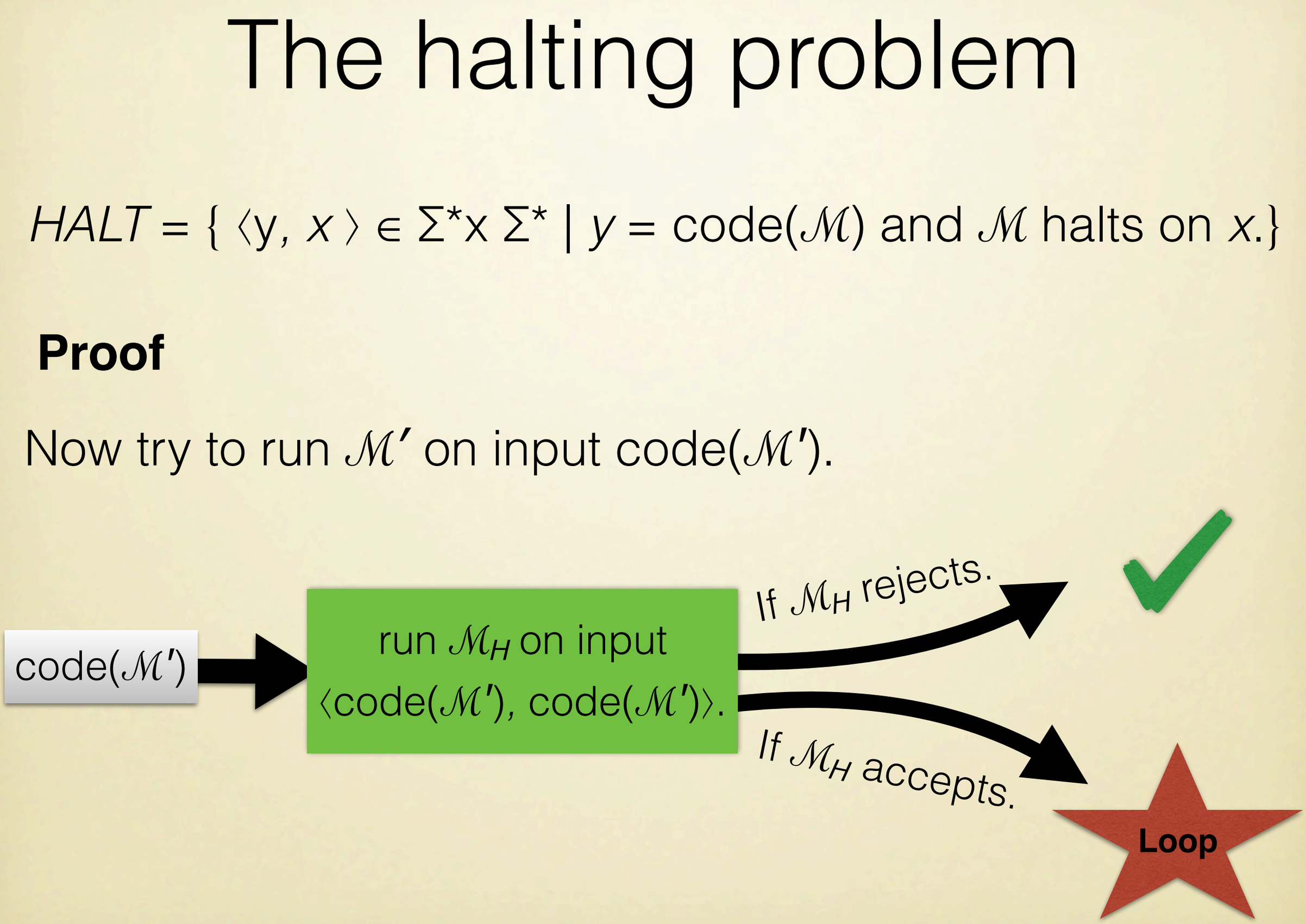


The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* x \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

Now try to run $\mathcal{M}'$ on input code($\mathcal{M}'$).



The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* x \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

$\mathcal{M}_H$ accepts $\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle$.

by def of $\mathcal{M}'$ ⇘

$\mathcal{M}'$ does not halt (loops) on code($\mathcal{M}'$).

⇗ by def of *HALT*

$\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle \notin HALT$

⇘ by def of $\mathcal{M}_H$

$\mathcal{M}_H$ does not accept $\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle$.

Contradiction. Let's try the other option…

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

$\mathcal{M}_H$ rejects $\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle$.

⇓ by def of $\mathcal{M}'$

$\mathcal{M}'$ halts on code($\mathcal{M}'$).

⇑ by def of *HALT*

$\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle \in HALT$

⇓ by def of $\mathcal{M}_H$

$\mathcal{M}_H$ accepts $\langle \text{code}(\mathcal{M}'), \text{code}(\mathcal{M}') \rangle$.

Either cases, we reach a contradiction.

The halting problem

$HALT = \{ \langle y, x \rangle \in \Sigma^* x \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

Proof

The only assumption needed for the construction of $\mathcal{M}'$ was that a TM $\mathcal{M}_H$ deciding *HALT* existed.

So $\mathcal{M}_H$ cannot exist, meaning that *HALT* is undecidable.

Where we are, so far

Decided by a TM	~	Solvable (There exists an alg answering “Yes”
Not decided by any TM but recognised by a TM	~	Unsolvable (Semi-decidable: no algorithm has all the “No" ans
Not even recognised by any TM	~	Unsolvable (Completely undecidable: any algorithm will miss both “Yes" and “No" answers)

the Halting problem

?

Un-recognisable problems

Theorem the complement $HALT^-$ of the Halting problem is not recognised by any Turing machine.

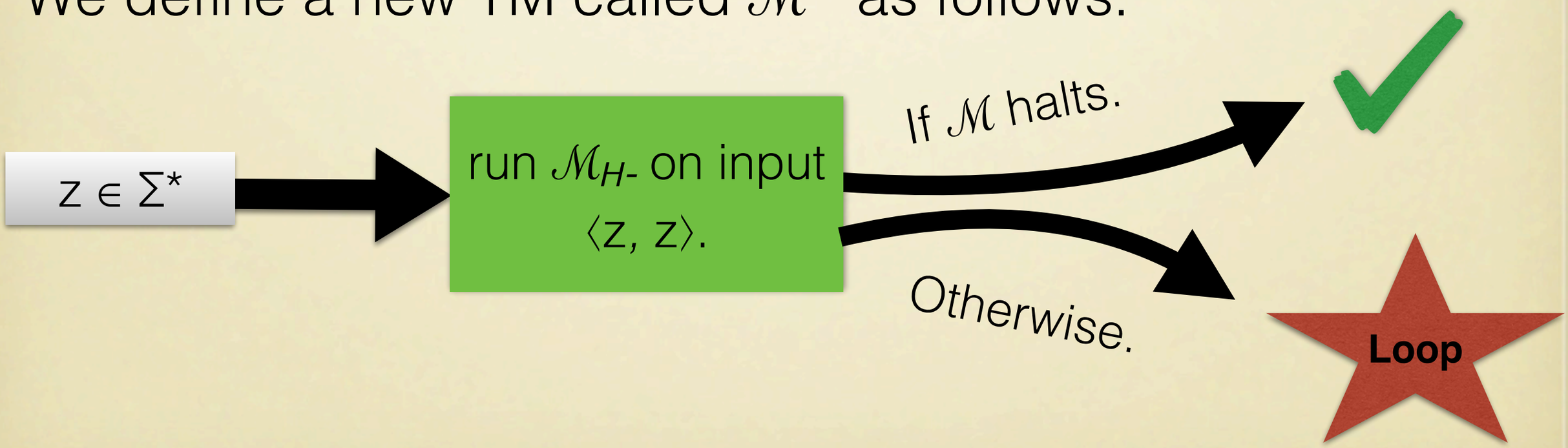
$HALT = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ halts on } x. \}$

$HALT^- = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M}) \text{ for any } \mathcal{M} \text{ or } y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ does not halt on } x. \}$

Un-recognisable problems

$HALT^{-} = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M}) \text{ for any } \mathcal{M} \text{ or } y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ does not halt on } x. \}$

Proof Again we make a proof by contradiction. Suppose that $HALT^{-}$ is recognisable and let $\mathcal{M}_{H^-}$ be the TM recognising it. We define a new TM called $\mathcal{M}''$ as follows.



Un-recognisable problems

$HALT^{-} = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M}) \text{ for any } \mathcal{M} \text{ or } y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ does not halt on } x. \}$

Proof

$\mathcal{M}_{H^-}$ halts on $\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle$.

by def of $\mathcal{M}''$ ⇘

$\mathcal{M}''$ halts on code($\mathcal{M}''$).

⇗ by def of $HALT^{-}$

$\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle \notin HALT^{-}$

↘ by def of $\mathcal{M}_{H^-}$

$\mathcal{M}_{H^-}$ does not halt on $\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle$.

Contradiction. Let's try the other option…

Un-recognisable problems

$HALT^{-} = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M}) \text{ for any } \mathcal{M} \text{ or } y = \text{code}(\mathcal{M}) \text{ and } \mathcal{M} \text{ does not halt on } x. \}$

Proof

$\mathcal{M}_{H^-}$ does not halt on $\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle$.

⇘ by def of $\mathcal{M}''$

$\mathcal{M}''$ does not halt on code($\mathcal{M}''$).

⇗ by def of $HALT^-$

$\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle \in HALT^-$

↘ by def of $\mathcal{M}_{H^-}$

$\mathcal{M}_{H^-}$ halts on $\langle \text{code}(\mathcal{M}''), \text{code}(\mathcal{M}'') \rangle$.

Either cases, we reach a contradiction.

Un-recognisable problems

$HALT^{-} = \{ \langle y, x \rangle \in \Sigma^* \times \Sigma^* \mid y \neq \text{code}(\mathcal{M})$ for any $\mathcal{M}$ or $y = \text{code}(\mathcal{M})$ and $\mathcal{M}$ does not halt on x.$\}$

Proof

The only assumption needed for the construction of $\mathcal{M}''$ was that a TM $\mathcal{M}_{H^-}$ recognising $HALT^{-}$ existed.

So $\mathcal{M}_{H^-}$ cannot exist, meaning that $HALT^{-}$ is not recognisable.

Where we are, so far

Decided by a TM	~	Solvable (There exists an alg answering “Yes”
Not decided by any TM but recognised by a TM	~	Unsolvable (Semi-decidable: no a has all the “No" an
Not even recognised by any TM	~	Unsolvable (Completely undecidable: any algorithm will miss both “Yes" and “No" answers)

the Halting problem

The Complement of the Halting problem

A different proof

Theorem the complement $\overline{HALT}$ of the Halting problem is not recognised by any Turing machine.

There is a more “abstract” and indirect proof of this theorem. We can show:

Theorem If $\overline{HALT}$ was recognisable, then *HALT* would be decidable.

Corollary Therefore, because *HALT* is undecidable, $\overline{HALT}$ is not recognisable.

A different proof

Theorem If $HALT^-$ was recognisable, then *HALT* would be decidable.

Proof

We proved before that *HALT* is recognisable, say by a TM $\mathcal{M}_{HR}$.

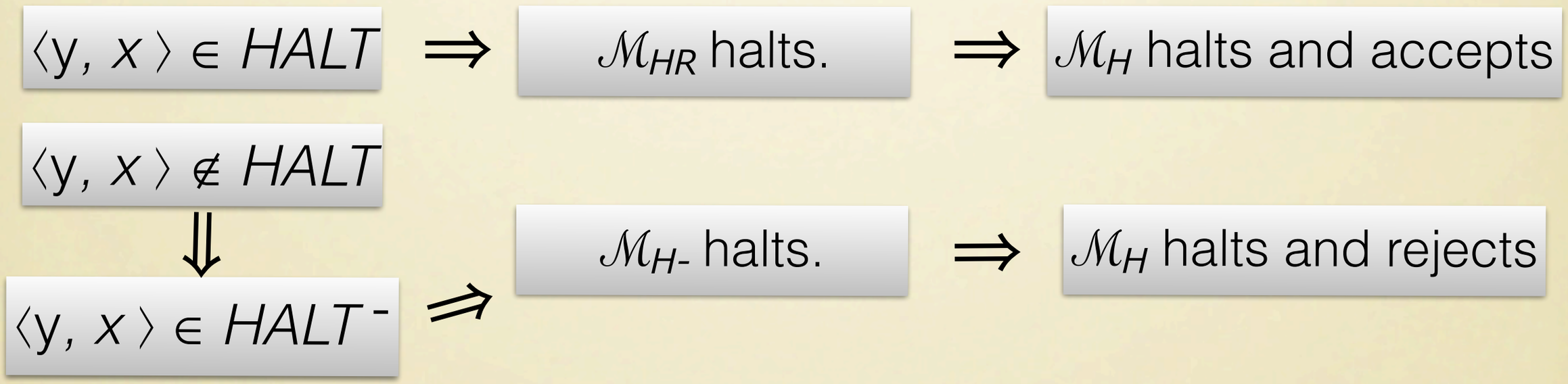
Suppose that $HALT^-$ is also recognisable and let $\mathcal{M}_{H^-}$ be the TM recognising it.

We can now construct a TM $\mathcal{M}_H$ *deciding HALT as* follows.

A different proof

Proof Definition of $\mathcal{M}_H$:

on input $\langle y, x \rangle$, run $\mathcal{M}_{HR}$ and $\mathcal{M}_{H^-}$ *in parallel* on $\langle y, x \rangle$. If $\mathcal{M}_{HR}$ halts, accept. If $\mathcal{M}_{H^-}$ halts, reject.



Therefore $\mathcal{M}_H$ decides *HALT*.

A different proof

Thus we have proved:

Theorem If $HALT^-$ was recognisable, then *HALT* would be decidable.

Corollary Therefore, because *HALT* is undecidable, $HALT^-$ is not recognisable.

This triggers two observations.

Observation #1
Complement of a Recognisable Language

The proof that we gave does not use in any way that HALT is a problem about machines that may or may not halt.
It would have worked with any recognisable language.

Theorem If L and L^- are recognisable, then L is decidable.

Proof The same we gave for $L = HALT$.

Observation #1
Complement of a Recognisable Language

Using

Theorem If L and L^- are recognisable, then L is decidable.

Theorem *HALT* is recognisable but not decidable.

We obtain:

Corollary Recognisable languages are *not* closed under complementation.

Proof *HALT* is recognisable. If its complement $HALT^-$ was recognisable, *HALT* would be decidable, contradiction.

Observation #2
Reduction arguments

In giving our second proof of the fact that $HALT^-$ is not recognisable, we used an argument of the kind:

If we could recognise *L*, *then* we could decide *L'*.
As L' is undecidable, *then* we cannot recognise L.

In the reminder of the course we will use this reduction procedure multiple times, in order to show that other interesting languages are not recognisable/decidable.