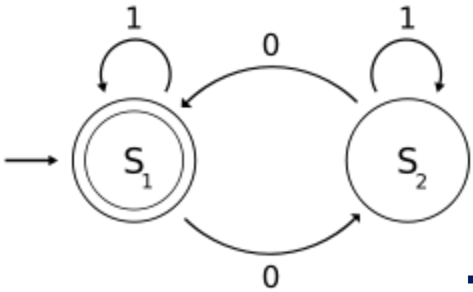
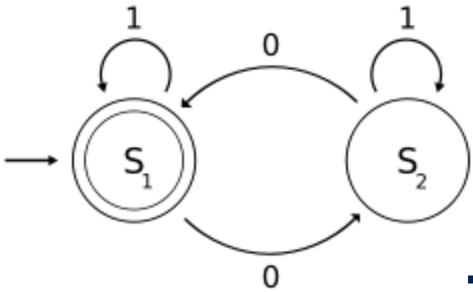
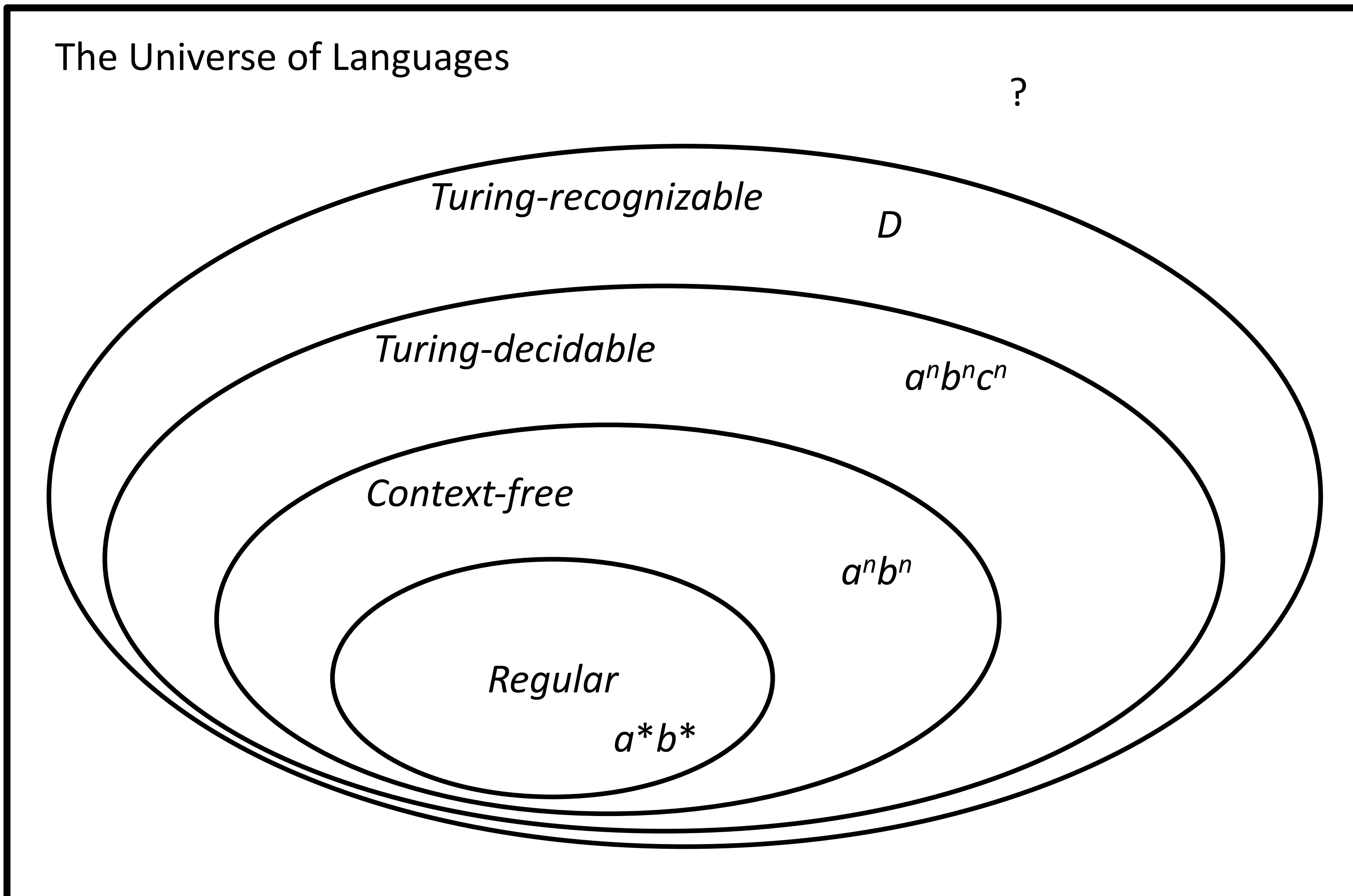


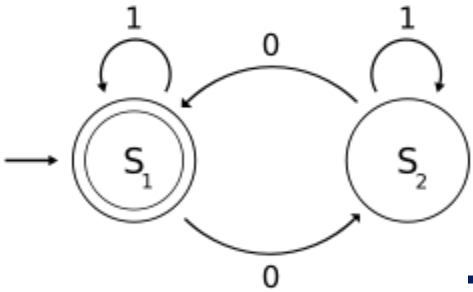
What Machines Cannot Do

Sipser: Section 4.2 pages 201 - 210



Will This Ever End?

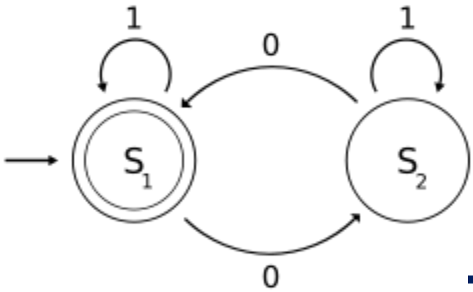




The Sizes of Sets

- Comparing the sizes of two finite sets is easy
- Do all infinite sets have the same size? How can we compare the relative sizes of two infinite sets?

$\mathbb{W}$	$\mathbb{N}$	$\mathbb{E}$
0	1	2
1	2	4
2	3	6
3	4	8
4	5	10
5	6	12
…	…	…

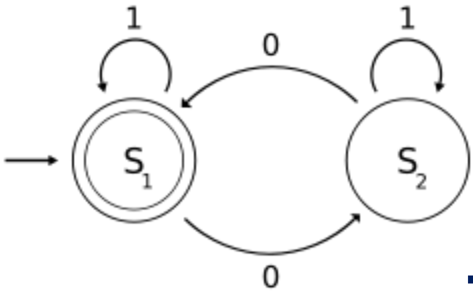


The Sizes of Sets

- Two sets have the same size if the elements of one set can be paired with the elements of the other set.
- A function that is both one-to-one and onto is called a ***correspondence*** (bijection). Two sets have the same size if there is a correspondence between them.

$\mathbb{W}$	$\mathbb{N}$	$\mathbb{E}$
0	1	2
1	2	4
2	3	6
3	4	8
4	5	10
5	6	12
…	…	…

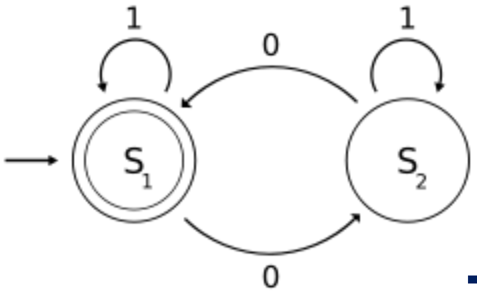
A set is ***countable*** if either it is finite or it has the same size as $\mathbb{N}$.



$\mathbb{R}$ is uncountable (proof by diagonalization)

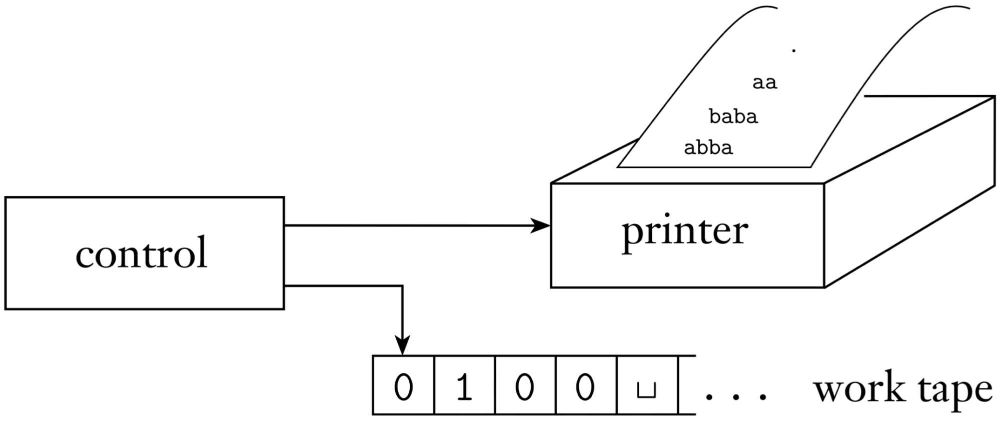
- We show that no correspondence exists between $\mathbb{N}$ and $\mathbb{R}$.
- To reach a contradiction, suppose that a correspondence f does exist between $\mathbb{N}$ and $\mathbb{R}$.
- We will find x in $\mathbb{R}$ that is not paired with anything in $\mathbb{N}$, which will be our contradiction.

n	$f(n)$
1	3.14159…
2	55.55555…
3	0.12345…
4	0.50000…
5	1.414213…
…	…

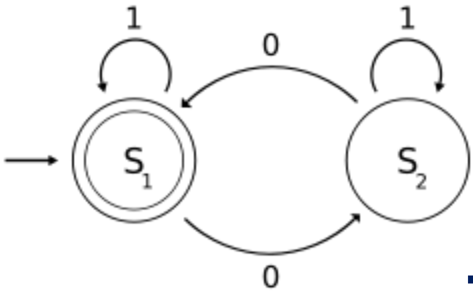


Finite Representation of Languages

- A finite representation of a language must itself be a string over some alphabet Σ. Furthermore, different languages must have distinct representations.

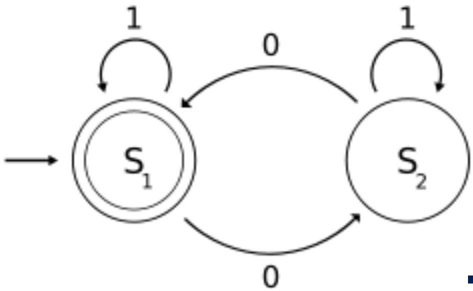


- How many strings can we represent over any given alphabet?



How Many is Many?

Theorem. Let Σ be any finite alphabet containing at least one element. The set of all strings Σ* over Σ is countably infinite.



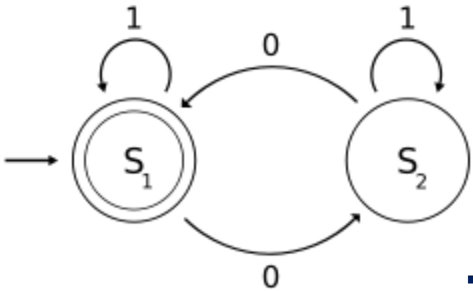
How Many Languages?

Definition. Let 2^{Σ^*}, known as the power set of Σ^*, be the set of all subsets of Σ^*, i.e., the set of all languages over Σ.

Theorem. The set 2^{Σ^*} is uncountable.

Proof. For each language $A \in 2^{\Sigma^*}$, create a unique infinite binary sequence.

$\Sigma^* = \{$	ε,	0,	1,	00,	01,	10,	11,	000,	001,	…	$\}$
$A = \{$		0,		00,	01,			000,	001,	…	$\}$
$f(A) =$	0	1	0	1	1	0	0	1	1	…	



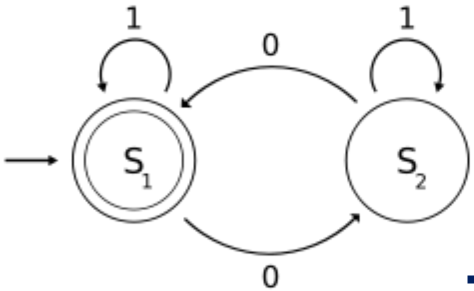
How Many Languages?

Definition. Let 2^{Σ^*}, known as the power set of Σ^*, be the set of all subsets of Σ^*, i.e., the set of all languages over Σ.

Theorem. The set 2^{Σ^*} is uncountable.

Proof. For each language $A \in 2^{\Sigma^*}$, create a unique infinite binary sequence.

$\Sigma^* = \{$	ε,	0,	1,	00,	01,	10,	11,	000,	001,	…	$\}$
$A = \{$	ε,	0,			01,	10,			001,	…	$\}$
$f(A) =$	1	1	0	0	1	1	0	0	1	…	



How Many Languages?

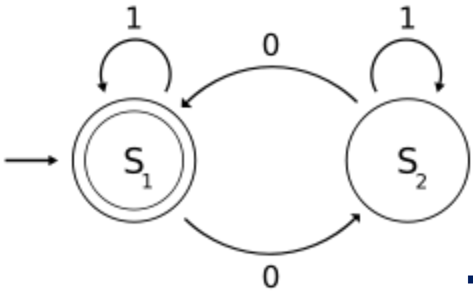
Definition. Let 2^{Σ^*}, known as the power set of Σ^*, be the set of all subsets of Σ^*, i.e., the set of all languages over Σ.

Theorem. The set 2^{Σ^*} is uncountable.

Proof. For each language $A \in 2^{\Sigma^*}$, create a unique infinite binary sequence.

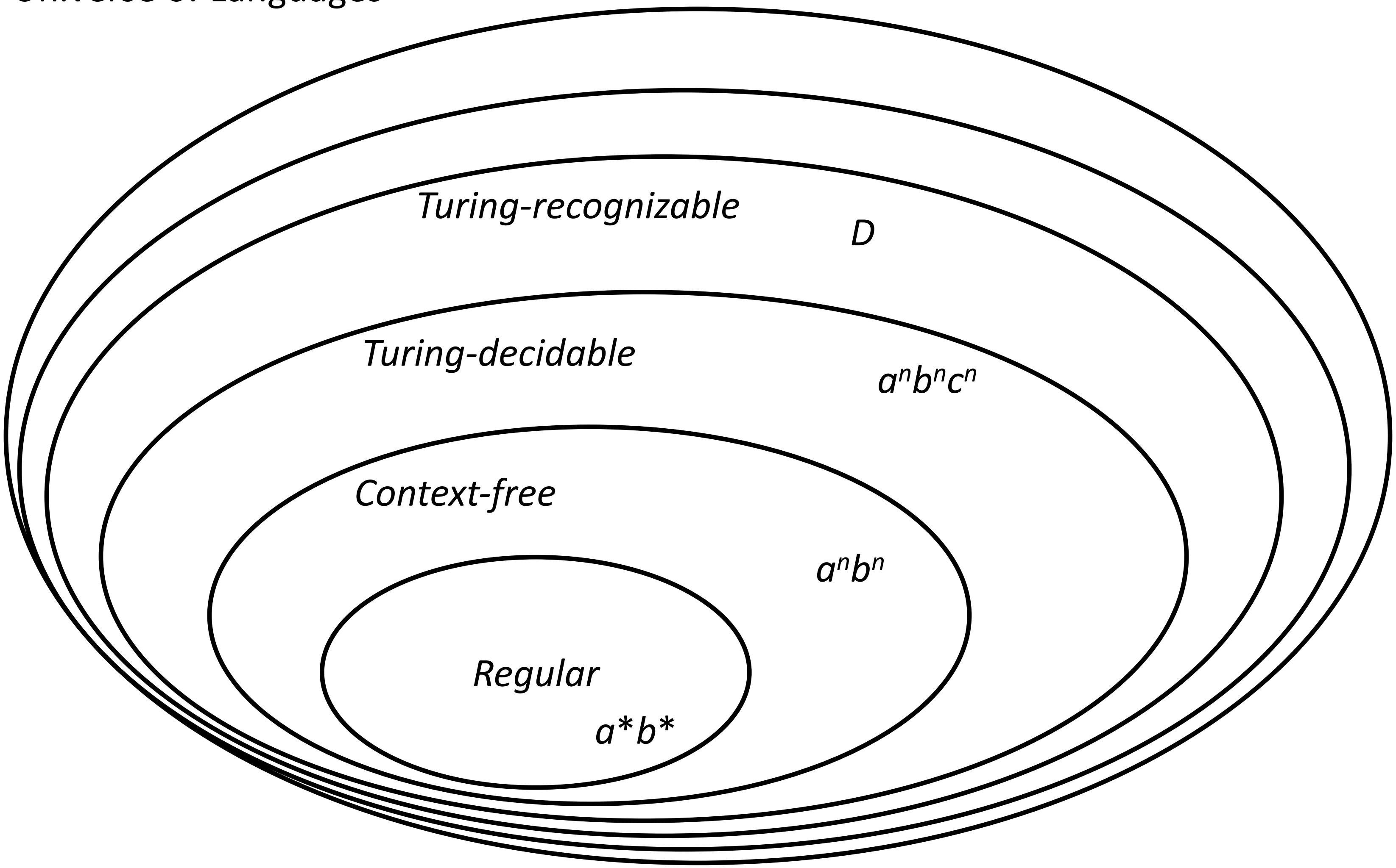
$\Sigma^* = \{$	ε,	0,	1,	00,	01,	10,	11,	000,	001,	...	$\}$
$A = \{$	ε,				01,		11,	000,	001,	...	$\}$
$f(A) =$	1	0	0	0	1	0	1	1	1	...	

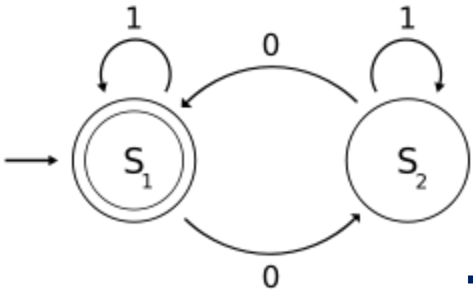
Thus, we have a correspondence f between 2^{Σ^*} and infinite binary sequences. Since the set of infinite binary sequences is uncountable (see homework), so is 2^{Σ^*}.



The Sad Conclusion…

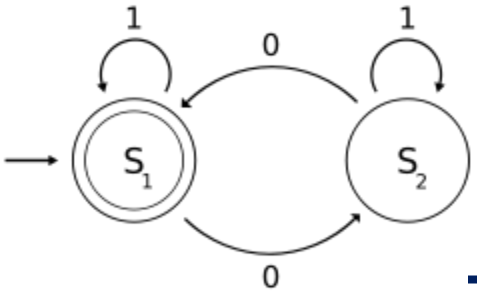
The Universe of Languages





The Trick is to Get all the Good Ones

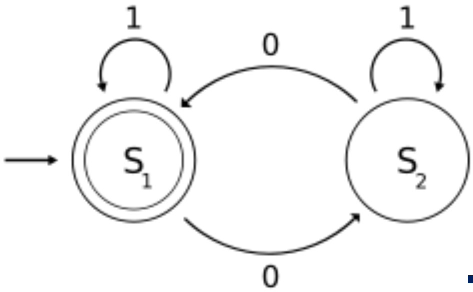
Algorithm	=	Turing Machine



Let's Try This One*

Definition. A_{TM} = { <*M*, *w*> | *M* is a TM and *M* accepts *w* }

* By analogy with our old friends A_{DFA} and A_{CFG}.



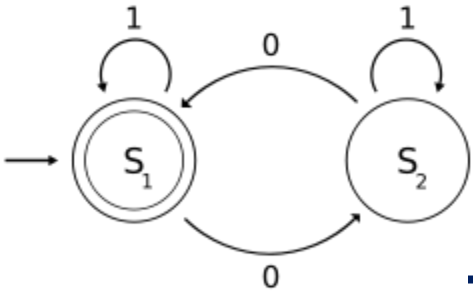
A_{TM} is Turing-Recognizable

U = "On input <*M*, *w*>, where *M* is a TM and *w* a string:

1. Simulate *M* on input *w*.

2. If *M* ever enters its accept state, *accept*. If *M* ever enters its reject state, *reject*."

The *universal Turing machine.*



The Halting Problem

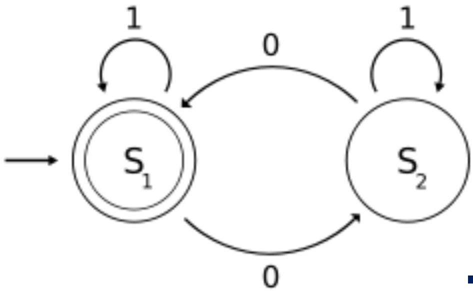
We could use *U* to decide A_{TM} if we had some way to determine whether *M* would halt on input *w*.

"On input <*M*, *w*>, where *M* is a TM and *w* a string:

1. Determine whether *M* on input *w* will ever halt. If not, then *reject*.

2. Otherwise, simulate *M* on input *w*.

3. If *M* enters its accept state, *accept*. If *M* enters its reject state, *reject*."

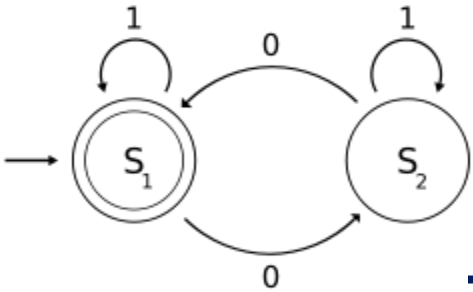


Some People Don't Know When to Stop

Theorem. A_{TM} = { <M, w> | M is a TM and M accepts w } is undecidable.

Proof. Suppose TM H decides A_{TM}. That is,

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w \end{cases}$$



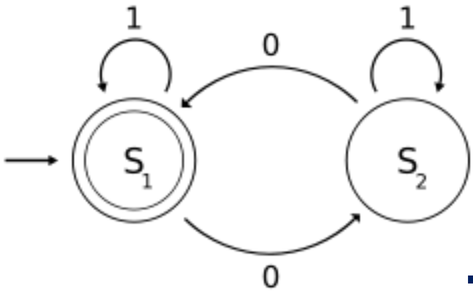
Calling *H* as a Subroutine

Define the contrary TM *D*:

D = "On input <*M*>, where *M* is a TM:

1. Run *H* on input <*M*, <*M*>>.*
2. Output the opposite of what *H* outputs.

* Think of a Python compiler written in Python.



Calling *H* as a Subroutine

Define the contrary TM *D*:

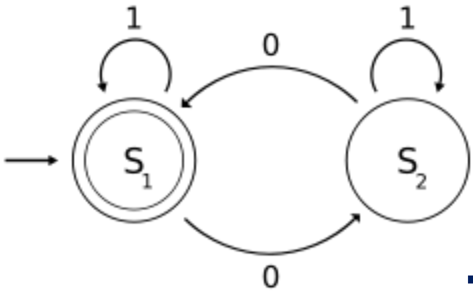
D = "On input <*M*>, where *M* is a TM:

1. Run *H* on input <*M*, <*M*>>.*
2. Output the opposite of what *H* outputs.

That is,

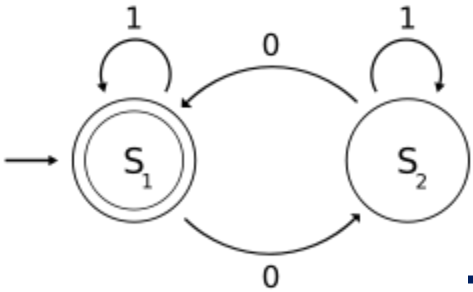
$$D(\langle M \rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\ \textit{reject} & \text{if } M \text{ accepts } \langle M \rangle \end{cases}$$

* Think of a Python compiler written in Python.



Calling *D* on Itself

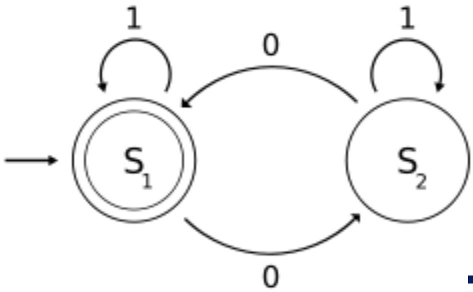
$$D(\langle D \rangle) = \begin{cases} \textit{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\ \textit{reject} & \text{if } D \text{ accepts } \langle D \rangle \end{cases}$$



$\bar{A}_{TM}$ is not even Turing-recognizable

Corollary. $\bar{A}_{TM}$ is not Turing-recognizable.

Proof. If so, then both A_{TM} and $\bar{A}_{TM}$ would be Turing-recognizable. But, then …



Out of Bounds