Unsolvable Problems

Part Two

Recap from Last Time

What exactly is the class **RE**?

An Intuition for **RE**

- Intuitively, a language L is in **RE** if a TM can search for positive proof that a string w belongs to L.
- Such a machine could work as follows:
  - Find a possible proof.
  - Check the proof.
  - If correct, accept!
  - If not, try the next proof.

Verifiers

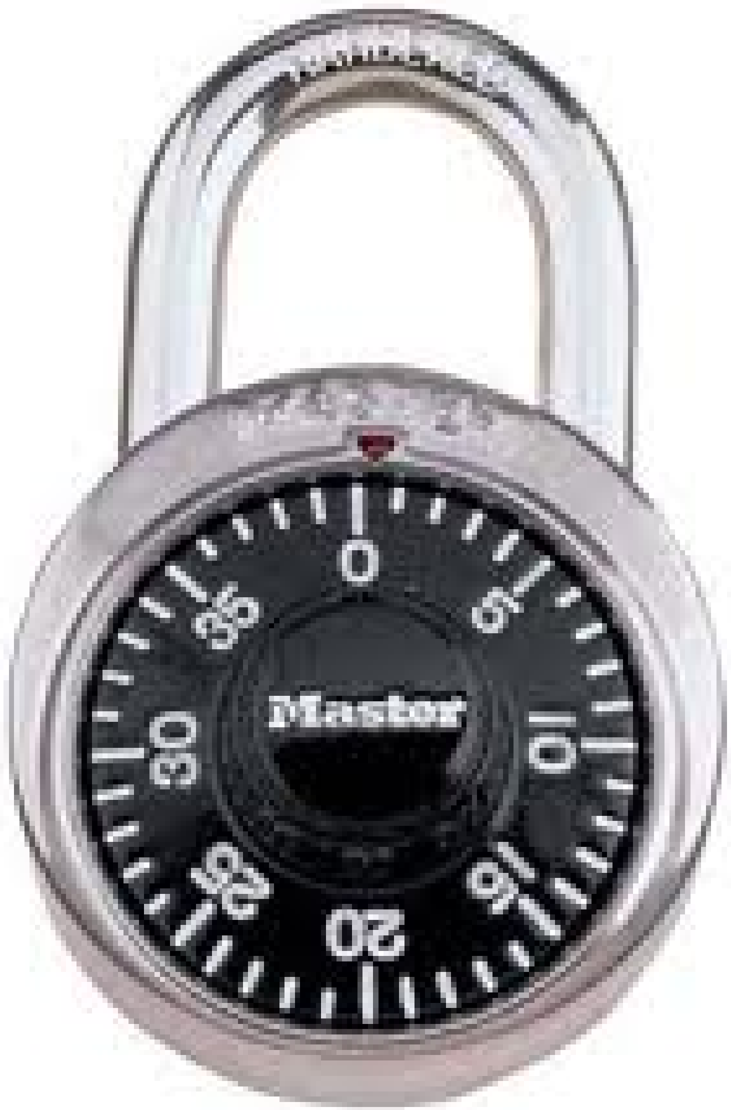
- A ***verifier*** for a language L is a TM V with the following properties:
  - V is a decider (that is, V halts on all inputs.)
  - For any string $w \in \Sigma^*$, the following is true:

$$w \in L \leftrightarrow \exists c \in \Sigma^*. \ V \text{ accepts } \langle w, c \rangle$$

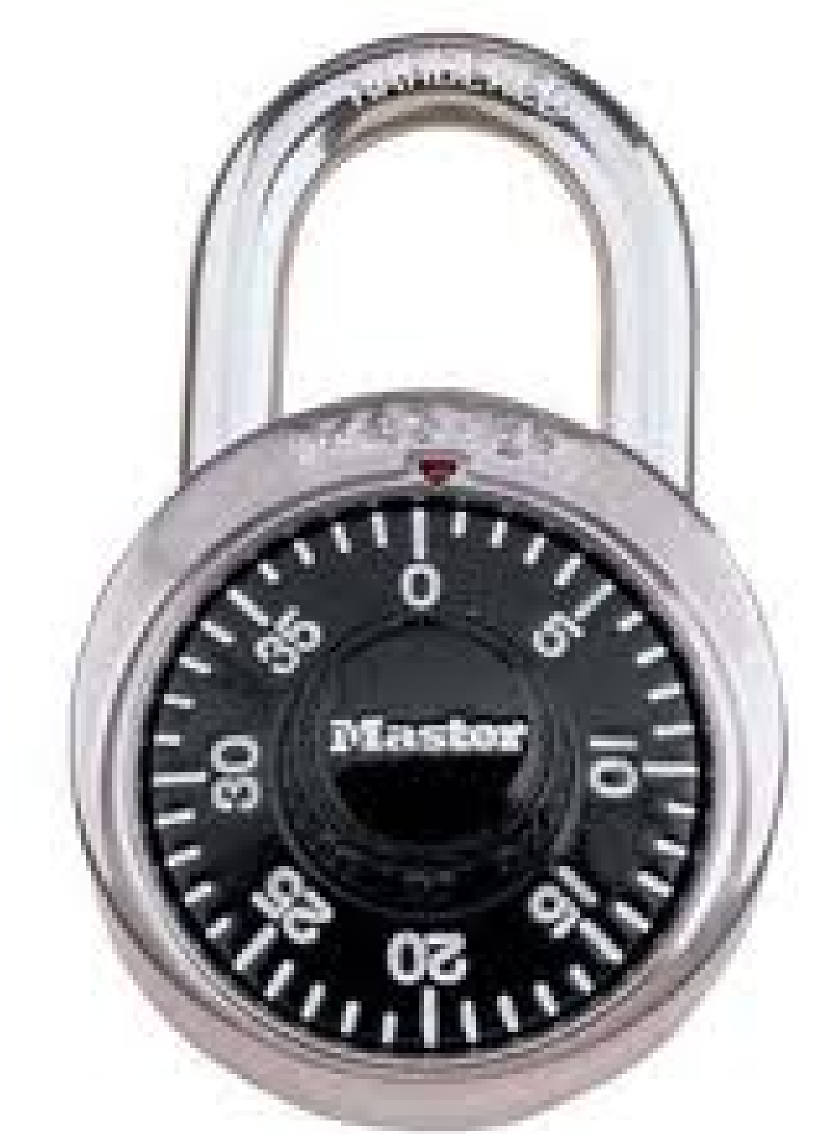
- Intuitively, what does this mean?

Intuiting Verifiers

Intuiting Verifiers



Intuiting Verifiers



Question:
Can this lock be opened?

Verifiers

- A ***verifier*** for a language L is a TM V with the following properties:
  - V is a decider (that is, V halts on all inputs.)
  - For any string $w \in \Sigma^*$, the following is true:

$$w \in L \quad \leftrightarrow \quad \exists c \in \Sigma^*. \; V \text{ accepts } \langle w, c \rangle$$

- Some notes about V:
  - If V accepts $\langle w, c \rangle$, then we're guaranteed $w \in L$.
  - If V does not accept $\langle w, c \rangle$, then either
    - $w \in L$, but you gave the wrong c, or
    - $w \notin L$, so no possible c will work.

Verifiers

- A ***verifier*** for a language L is a TM V with the following properties:
  - V is a decider (that is, V halts on all inputs.)
  - For any string $w \in \Sigma^*$, the following is true:

$$\boldsymbol{w \in L \quad \leftrightarrow \quad \exists c \in \Sigma^*.\ V \text{ accepts } \langle w, c \rangle}$$

- Some notes about V:
  - If $w \in L$, a string c for which V accepts $\langle w, c \rangle$ is called a ***certificate*** for w.
  - V is required to halt, so given any potential certificate c for w, you can check whether the certificate is correct.

Some Verifiers

- Let L be the following language:

$$L = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates } w \}$$

- Here is one possible verifier for L:

> V = "On input $\langle G, w, c \rangle$, where G is a CFG:
>
> Check whether c is a valid derivation of w from the start symbol of G.
>
> If so, accept. If not, reject."

- If the certificate is a correct derivation, we know for a fact that G can generate w.
- If not, we can't tell whether we got a bad certificate or whether G doesn't generate w.

Some Verifiers

- Let L be the following language:

$$L = \{ \langle n \rangle \mid n \in \mathbb{N} \text{ and the hailstone sequence terminates for } n \}$$

V = "On input $\langle n, k \rangle$, where $n, k \in \mathbb{N}$.
Check that $n \neq 0$.
Run the hailstone sequence, starting at n, for at most k steps.
If after k steps we reach 1, accept.
Otherwise, reject."

- Do you see why $\langle n \rangle \in L$ iff there is some k such that V accepts $\langle n, k \rangle$?

What languages are verifiable?

Verifiers and **RE**

- ***Theorem:*** If there is a verifier V for a language L, then $L \in$ **RE**.
- ***Proof idea:*** Build a recognizer that tries every possible certificate to see if $w \in L$.
- ***Proof sketch:*** Show that this TM is a recognizer for L:

M = "On input w:
    For $i = 0$ to ∞
        For each string c of length i:
            Run V on $\langle w, c\rangle$.
            If V accepts $\langle w, c\rangle$, M accepts w."

Verifiers and **RE**

- ***Theorem:*** If $L \in$ **RE**, then there is a verifier for L.
- ***Proof idea:*** If $L \in$ **RE**, there is a recognizer M for L.
- If $w \in L$, M accepts w after some number of steps. If $w \notin L$, M never accepts.
- Have the certificate for w be the number of steps before M accepts w.

Verifiers and **RE**

- ***Theorem:*** If $L \in \mathbf{RE}$, then there is a verifier for L.
- ***Proof sketch:*** Let L be an **RE** language and let M be a recognizer for it. Then show that this is a verifier for L:

V = "On input $\langle w, n \rangle$, where $n \in \mathbb{N}$:
Run M on w for n steps.
If M accepts w within n steps, accept.
If M did not accept w in n steps, reject."

Verifiers and **RE**

- The verifier definition of **RE** gives a new perspective on **RE** languages.
- A language L is in **RE** if there is a way to prove that strings belong to L.
- Equivalently, a problem is an **RE** problem if there is a way to prove that “yes” answers are correct.

$\mathbf{R} \neq \mathbf{RE}$

- We know that $\mathbf{R} \neq \mathbf{RE}$.
- The verifier perspective of **RE** gives a new interpretation to this:

In the limit, it is fundamentally harder to solve a problem than it is to check an answer to a problem.

Nondeterministic Turing Machines

Nondeterministic TMs

- A ***nondeterministic Turing machine*** (or ***NTM***) is a Turing machine in which there can be zero or multiple transitions defined at each state.
- Nondeterministic TMs do not have ε-transitions; they have to read or write something and move the tape at each step.
- As with NFAs, NTMs accept if any path accepts. In other words, an NTM for a language L is one where

  $w \in L$ iff there is some series of choices N can make that causes N to accept w.

- In particular, if $w \in L$, N only needs to accept w along one branch. The rest can loop infinitely or reject.

Designing an NTM

- A ***tautonym*** is a word that consists of the same string repeated twice.
- Some examples:
  - **`dikdik`** (an adorable petite antelope)
  - **`hotshots`** (people who aren't very fun to be around)

Designing an NTM

- A ***tautonym*** is a word that consists of the same string repeated twice.
- Some examples:
  - `dikdik` (an adorable petite antelo
  - `hotshots` (people who aren't very

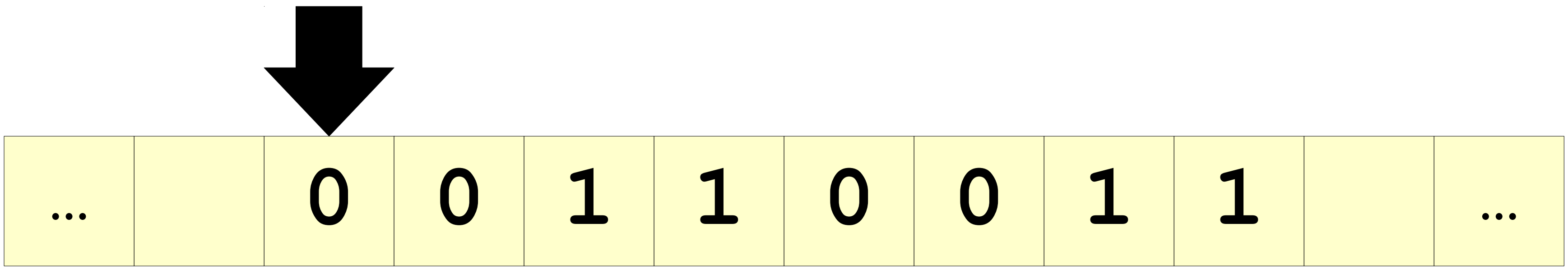
Designing an NTM

- A ***tautonym*** is a word that consists of the same string repeated twice.
- Some examples:
  - **`dikdik`** (an adorable petite antelope)
  - **`hotshots`** (people who aren't very fun to be around)
- Consider the following language over $\Sigma = \{\texttt{0}, \texttt{1}\}$:

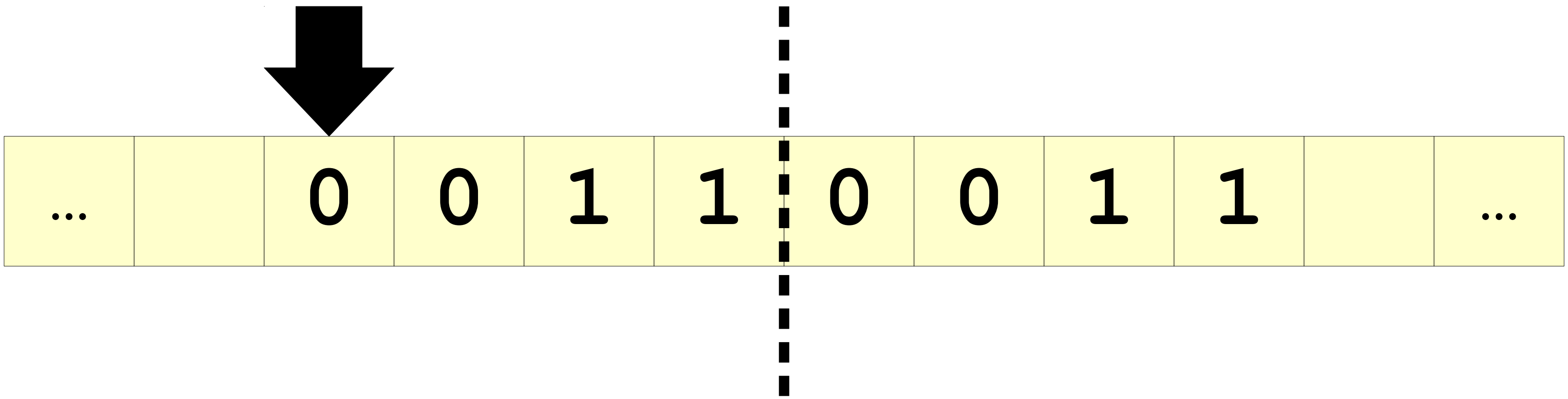
$$L = \{ ww \mid w \in \Sigma^* \text{ and } w \neq \varepsilon \}$$

- This is the set of all nonempty tautonyms.
- How might we design a TM for this language?

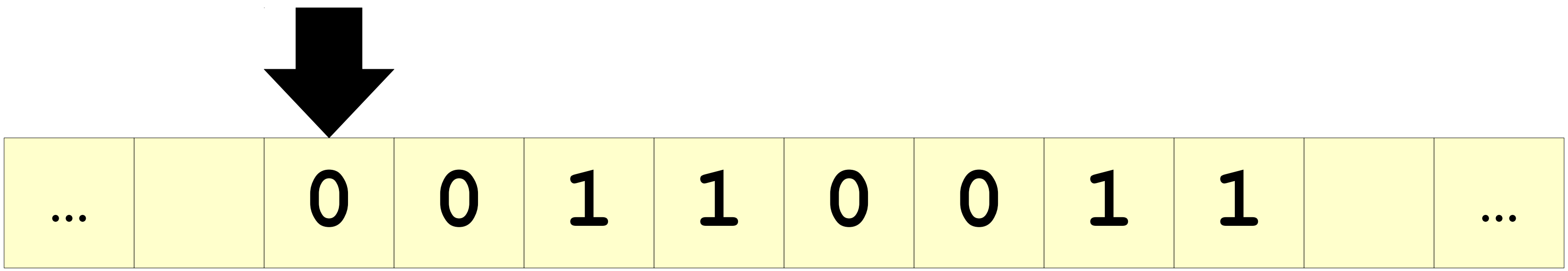
What's Tricky



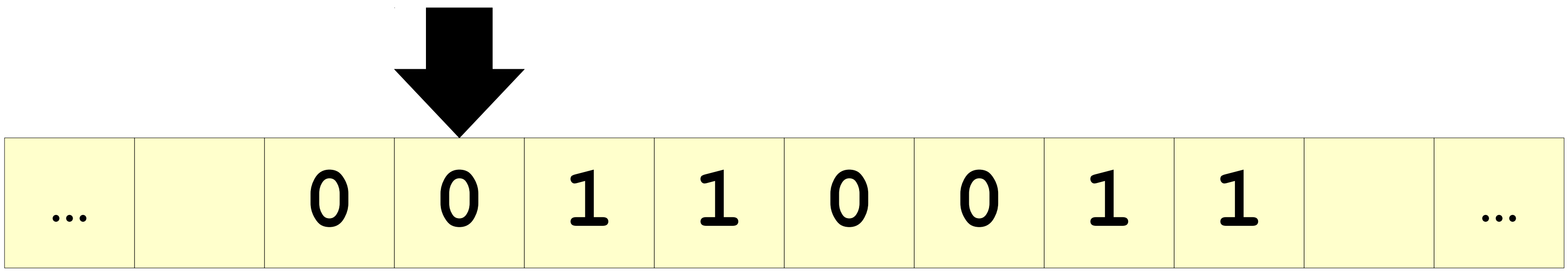
What's Tricky



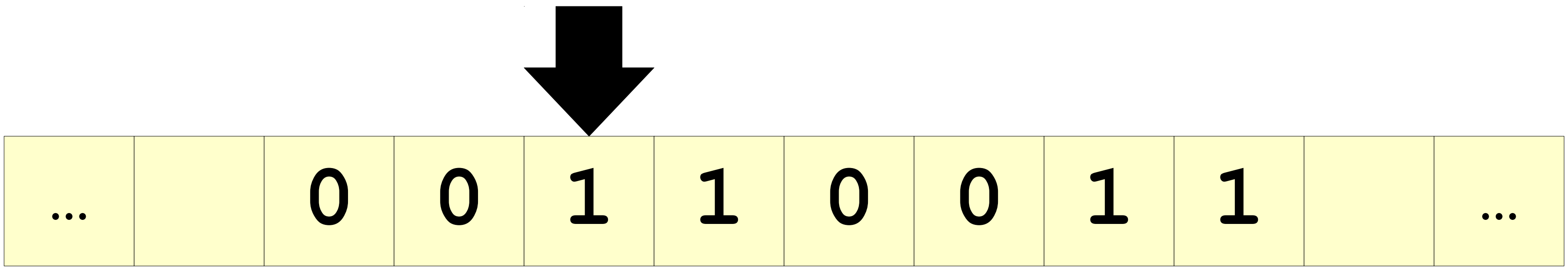
Using Nondeterminism



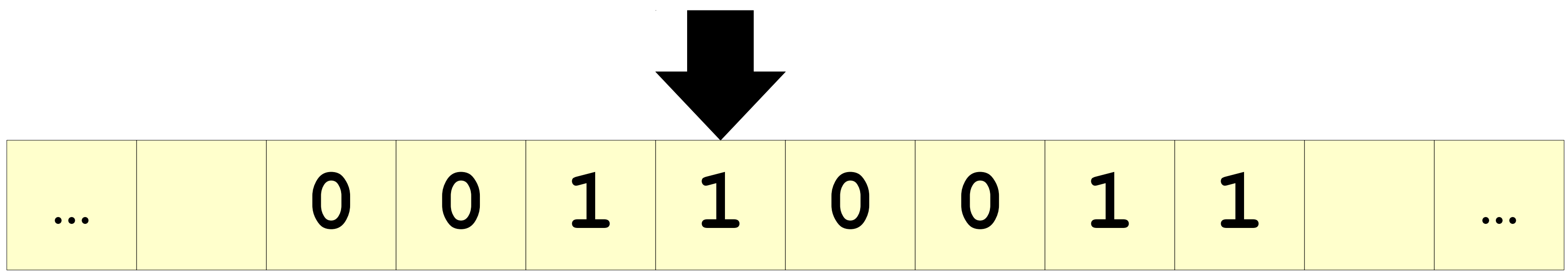
Using Nondeterminism



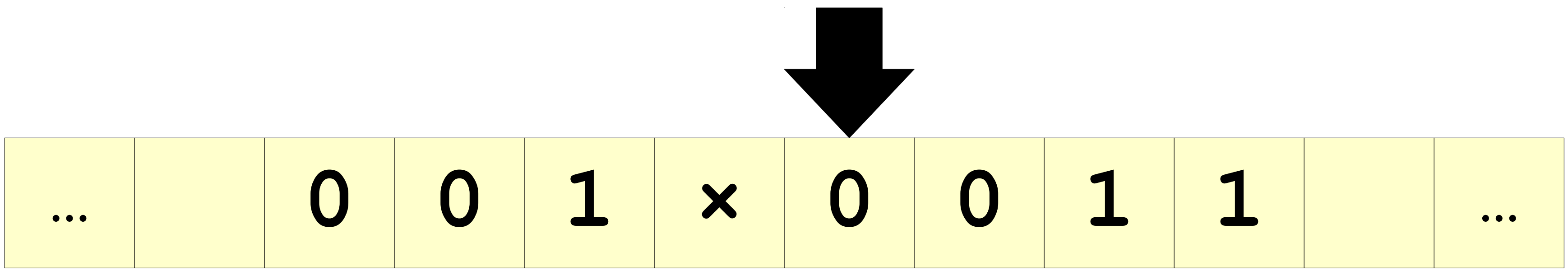
Using Nondeterminism



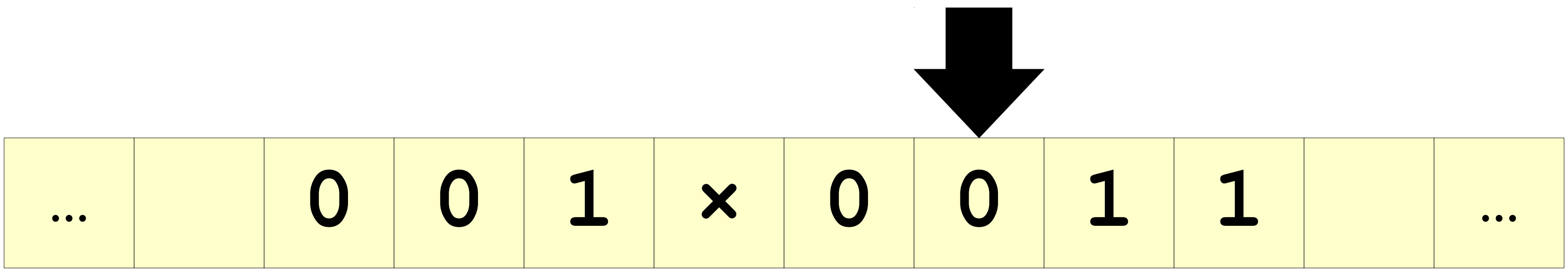
Using Nondeterminism



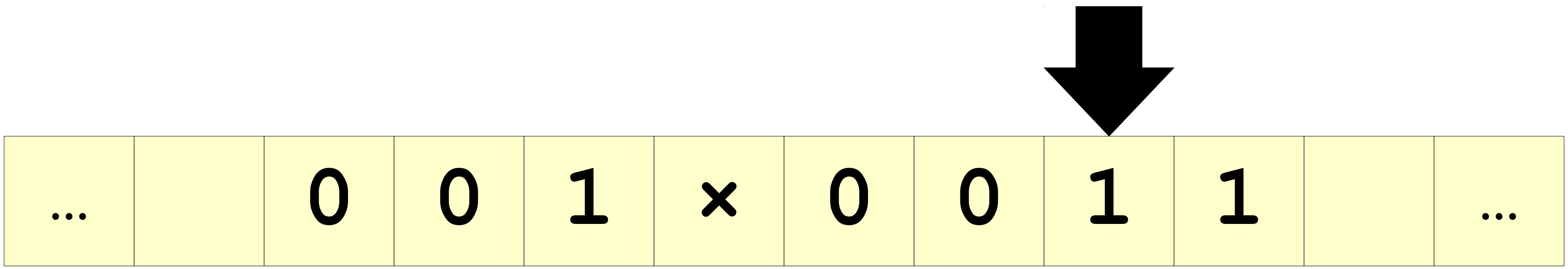
Using Nondeterminism



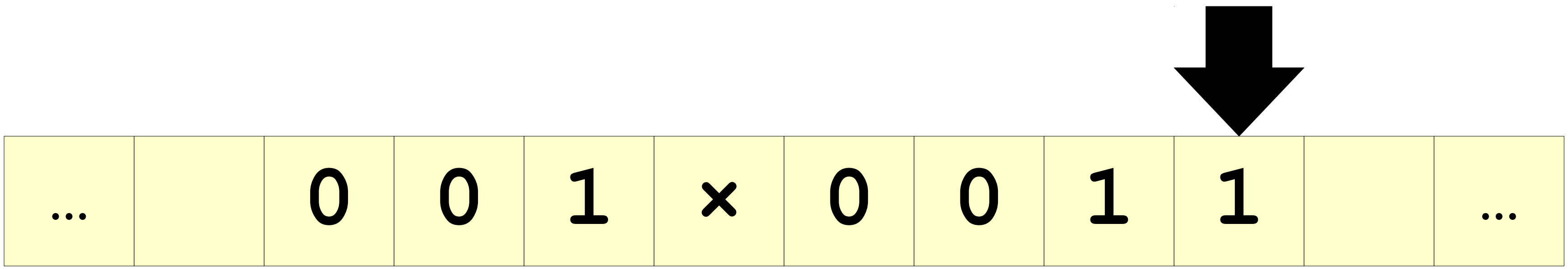
Using Nondeterminism



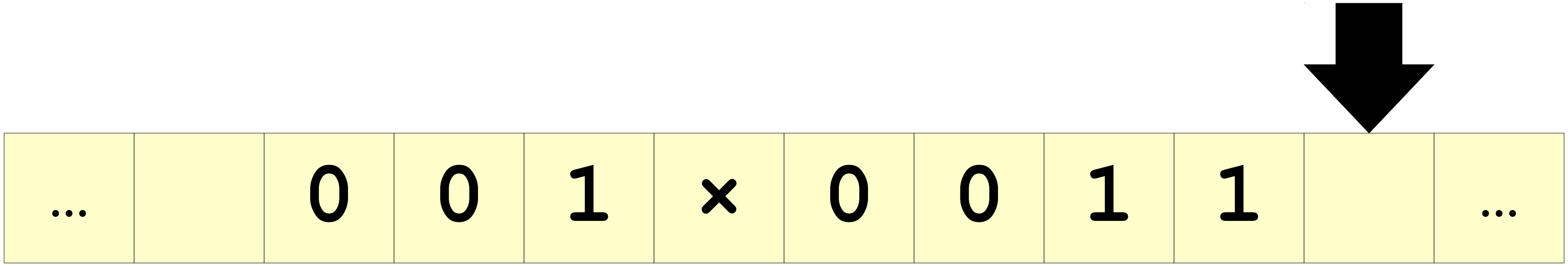
Using Nondeterminism



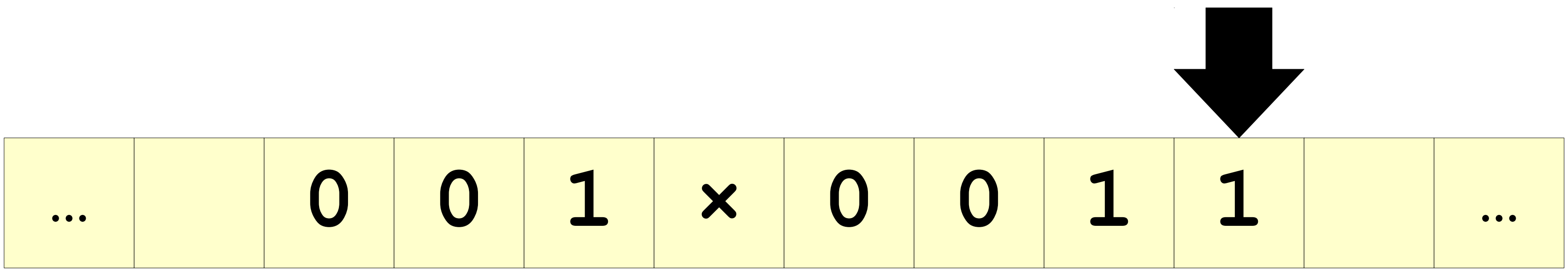
Using Nondeterminism



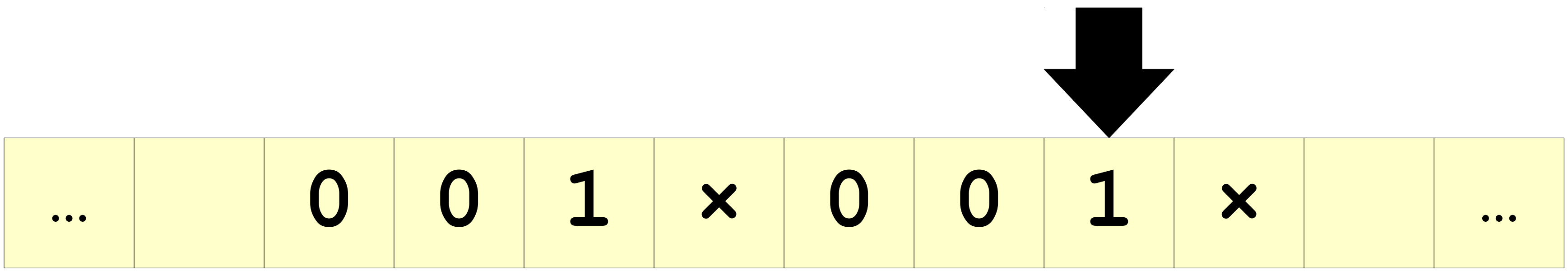
Using Nondeterminism

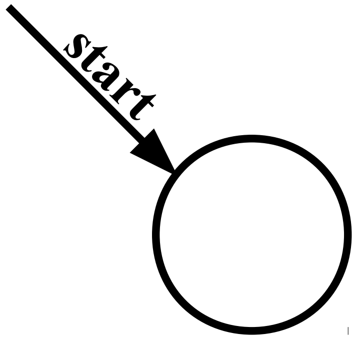


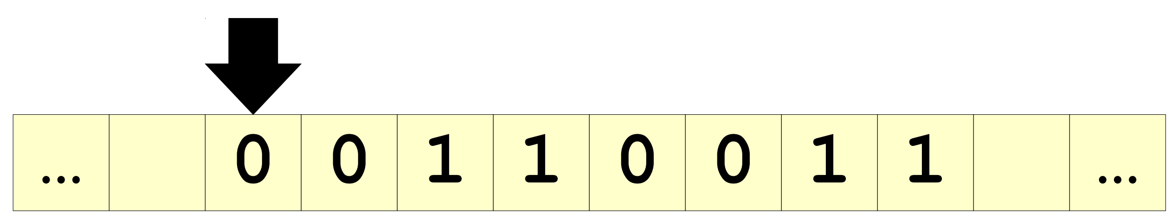
Using Nondeterminism

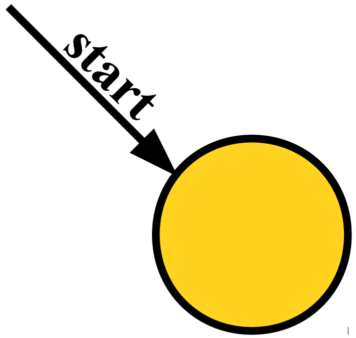


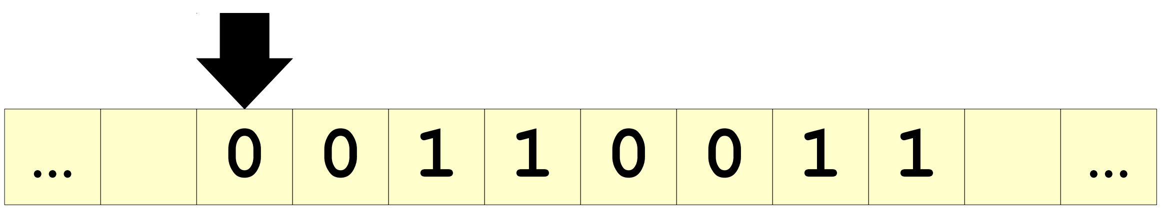
Using Nondeterminism

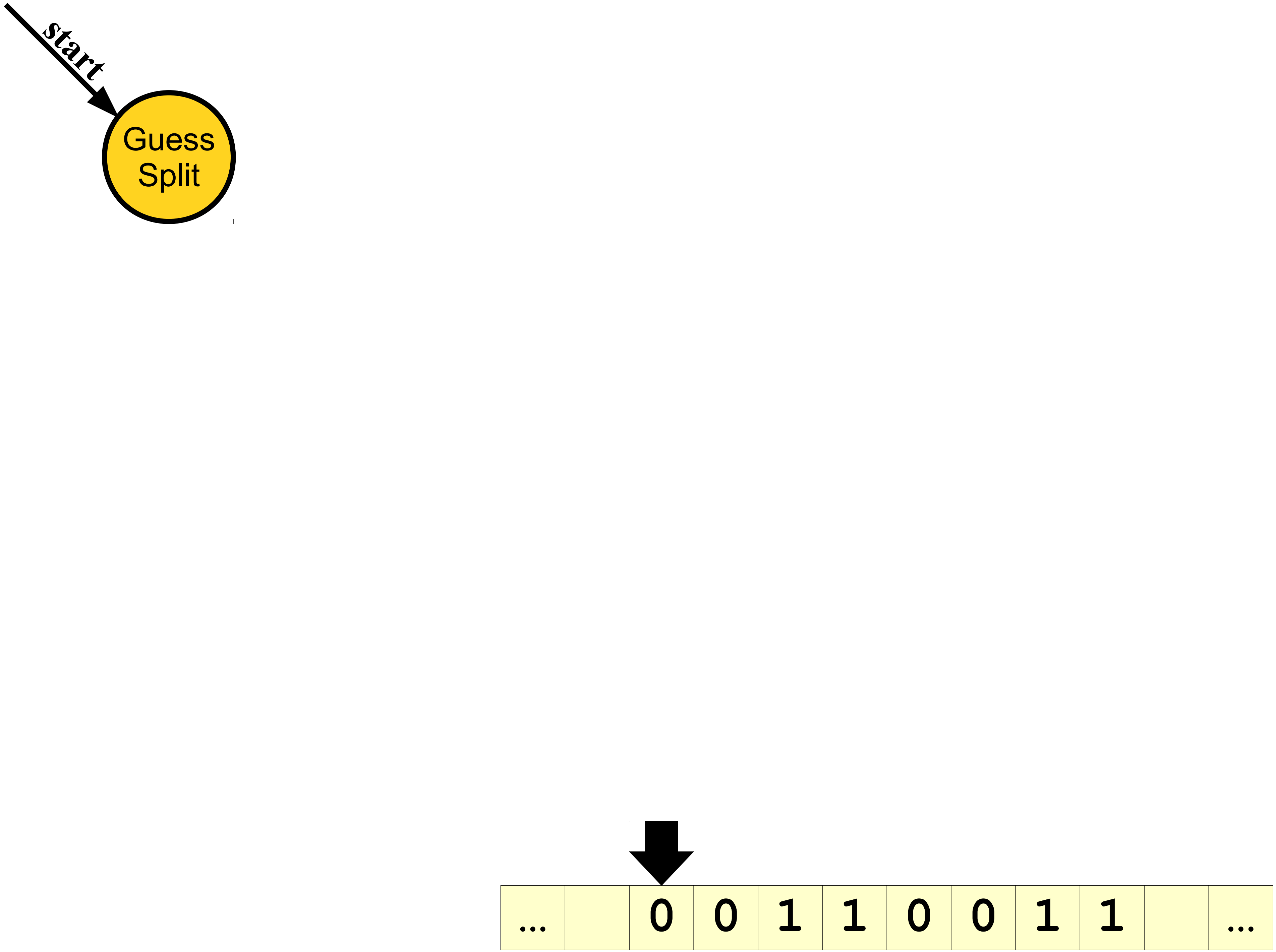


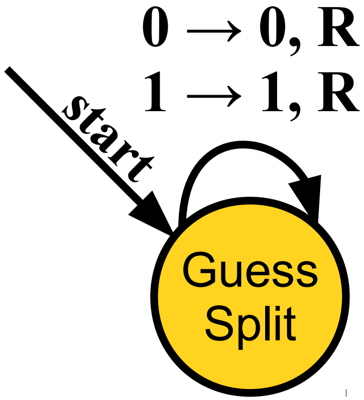

start

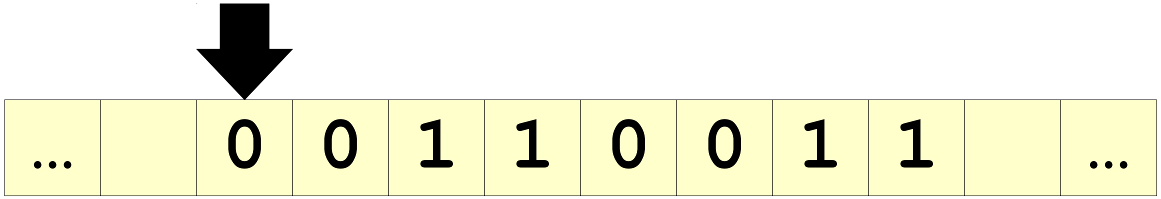

... 0 0 1 1 0 0 1 1 ...

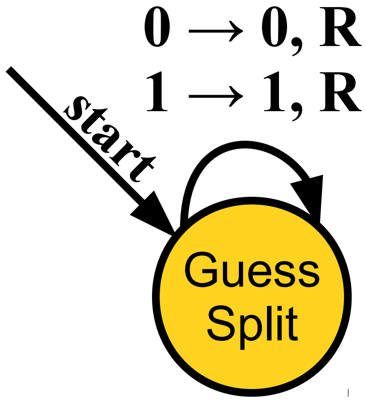

start

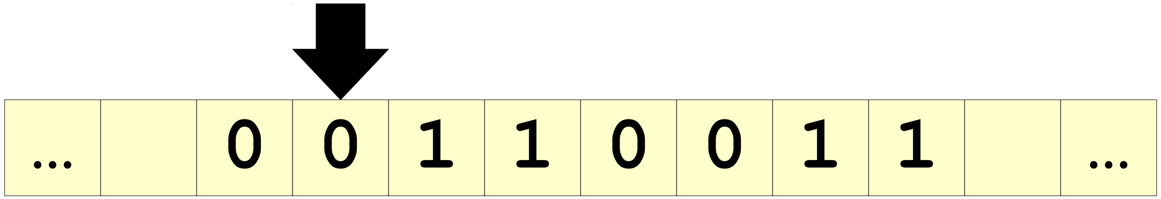

...
0 0 1 1 0 0 1 1
...

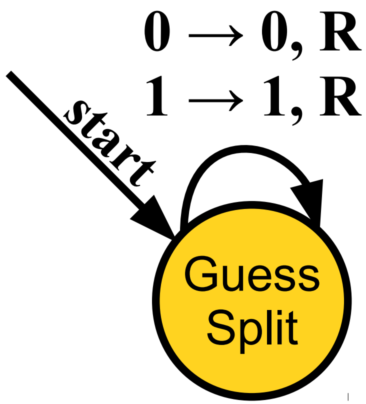

start
Guess Split
...
0 0 1 1 0 0 1 1
...

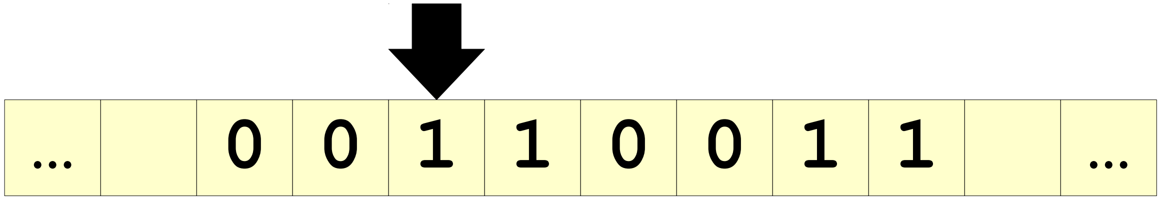

0 → 0, R
1 → 1, R
start
Guess Split

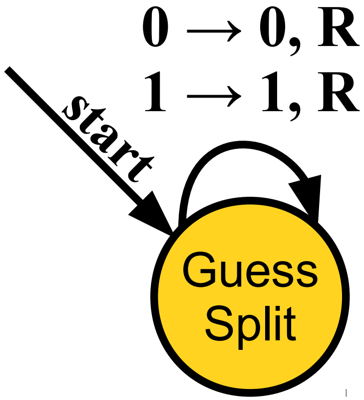

...
0
0
1
1
0
0
1
1
...

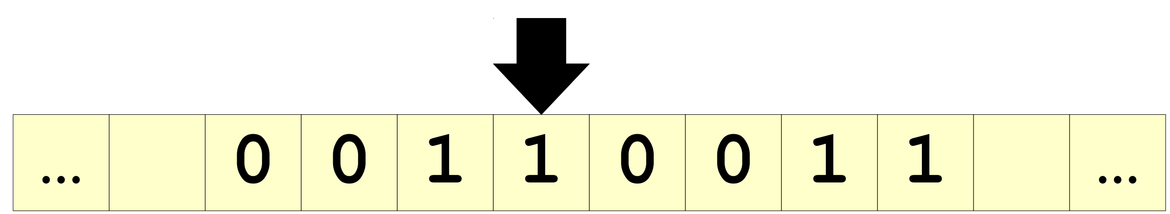

0 → 0, R
1 → 1, R
start
Guess Split

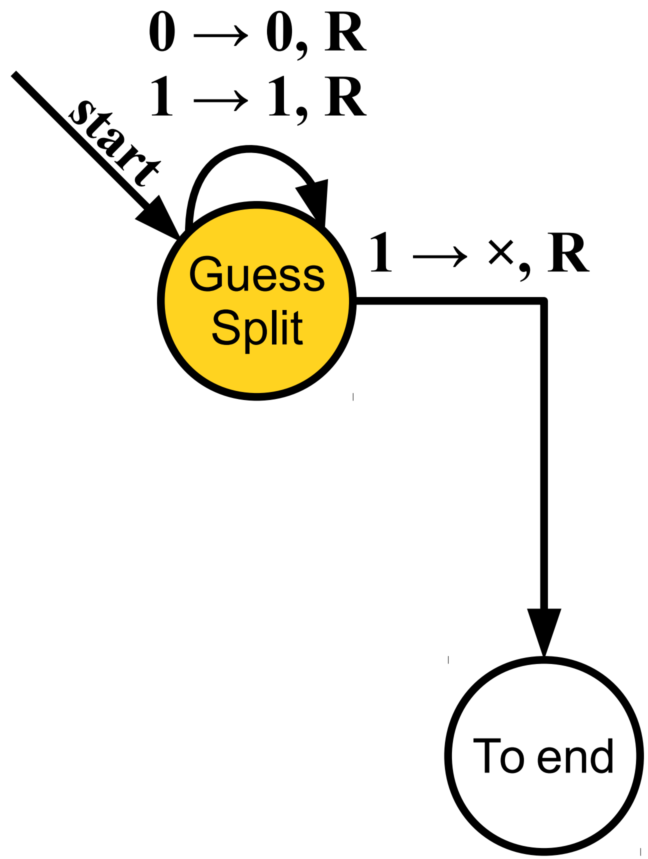

... 0 0 1 1 0 0 1 1 ...

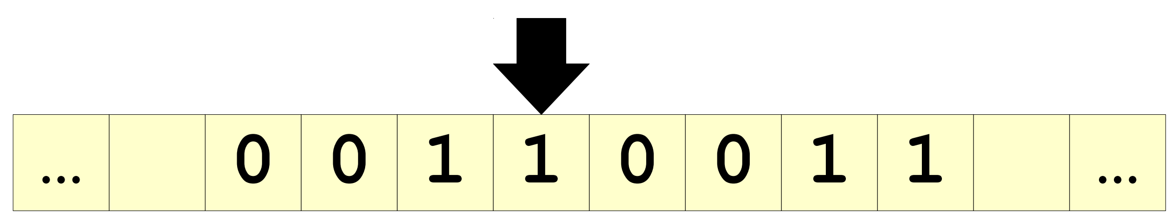

0 → 0, R
1 → 1, R
start
Guess Split

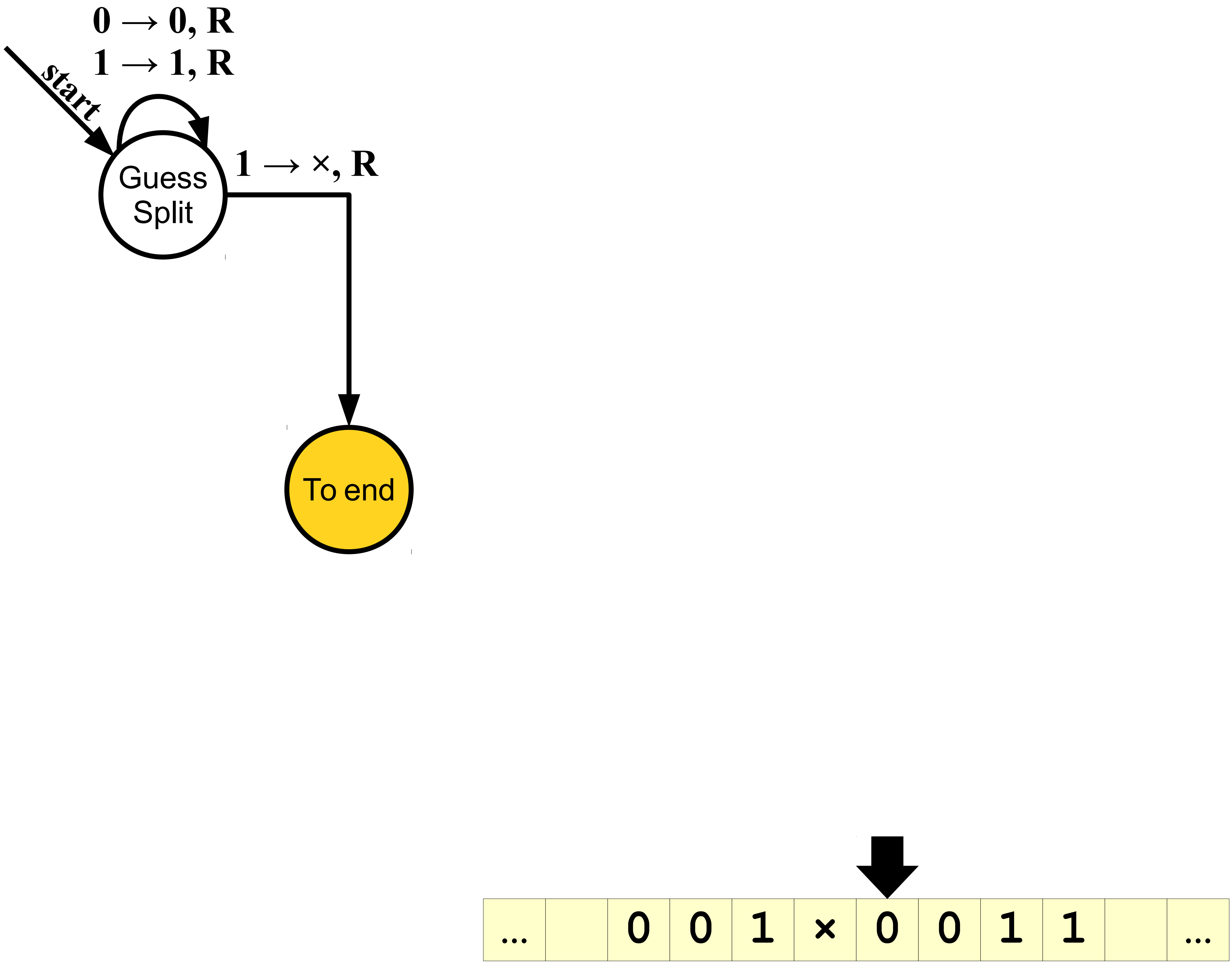

...
0
0
1
1
0
0
1
1
...

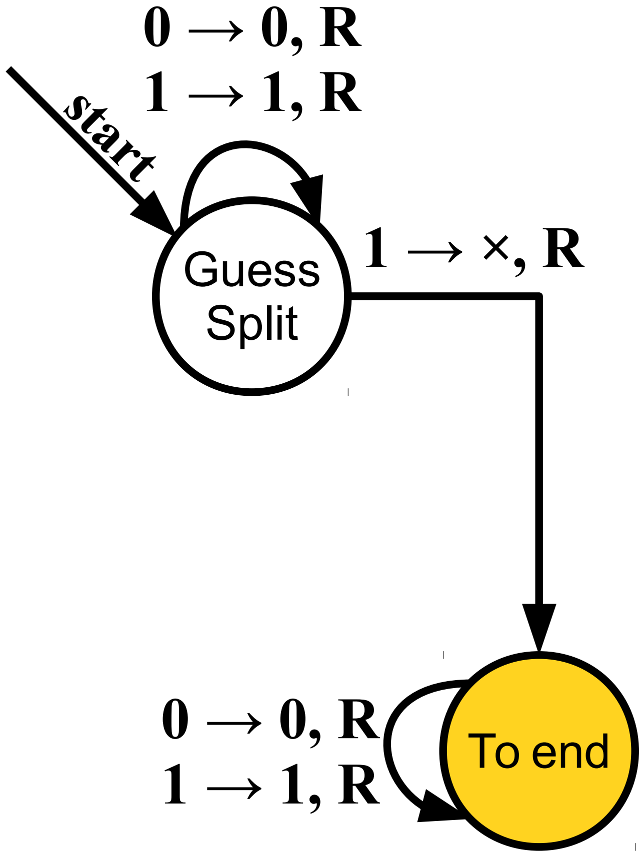

0 → 0, R
1 → 1, R
start
Guess Split

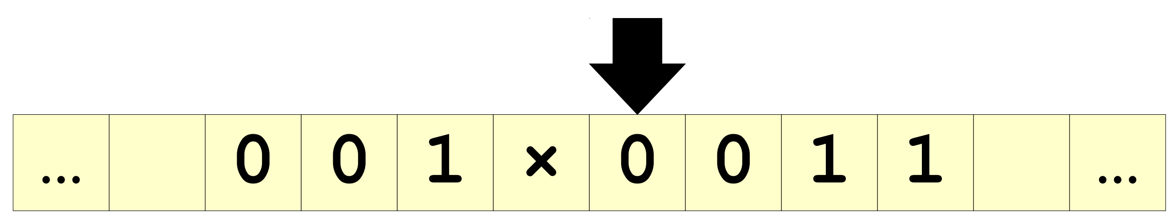

... 0 0 1 1 0 0 1 1 ...

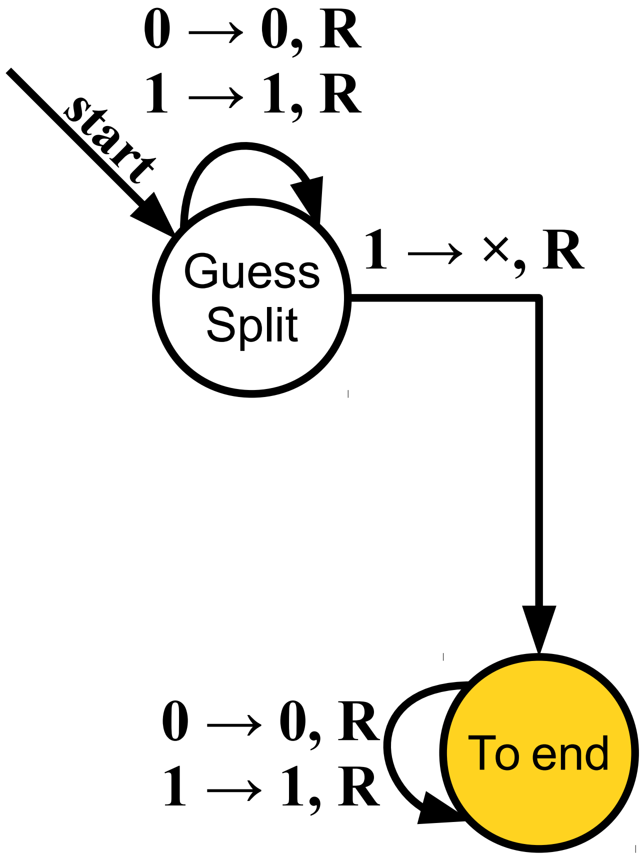

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
To end

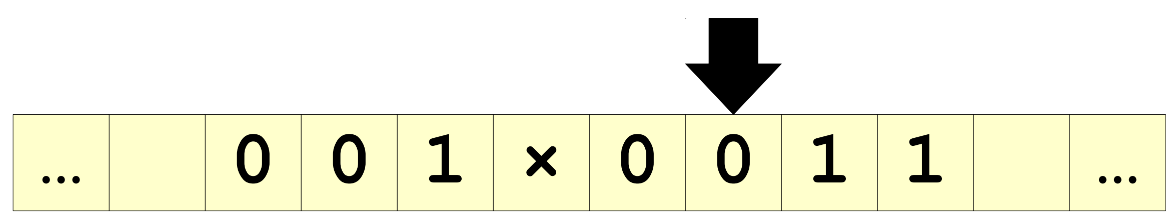

...
0
0
1
1
0
0
1
1
...

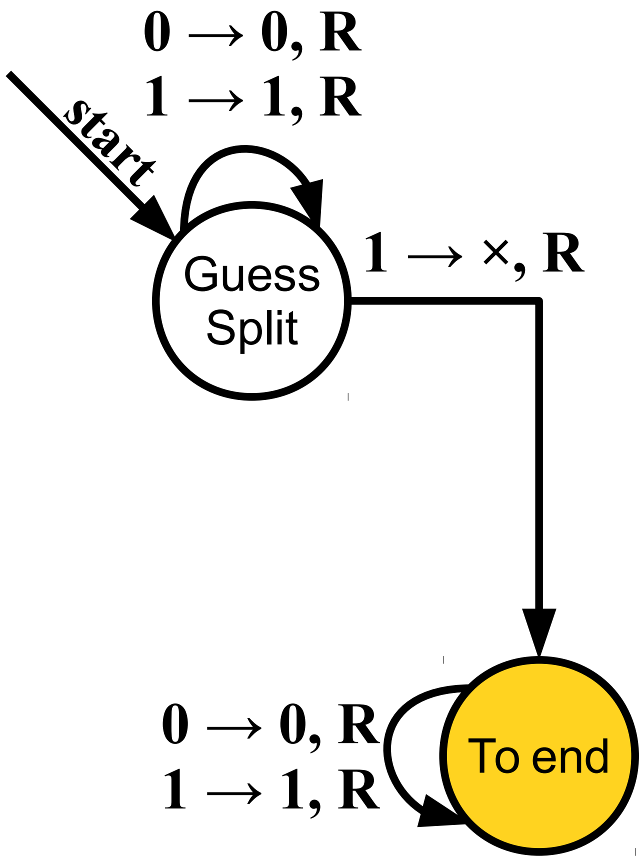

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
To end
...
0
0
1
×
0
0
1
1
...

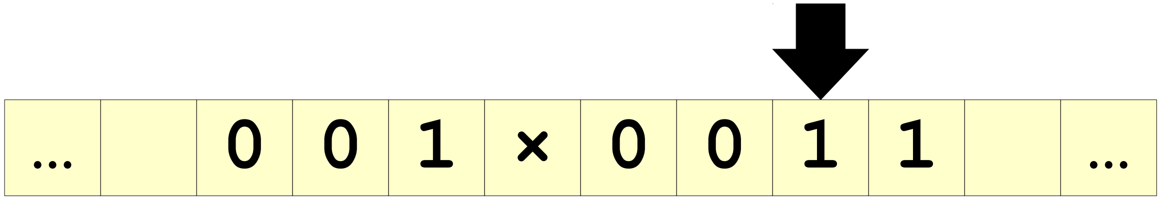

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end

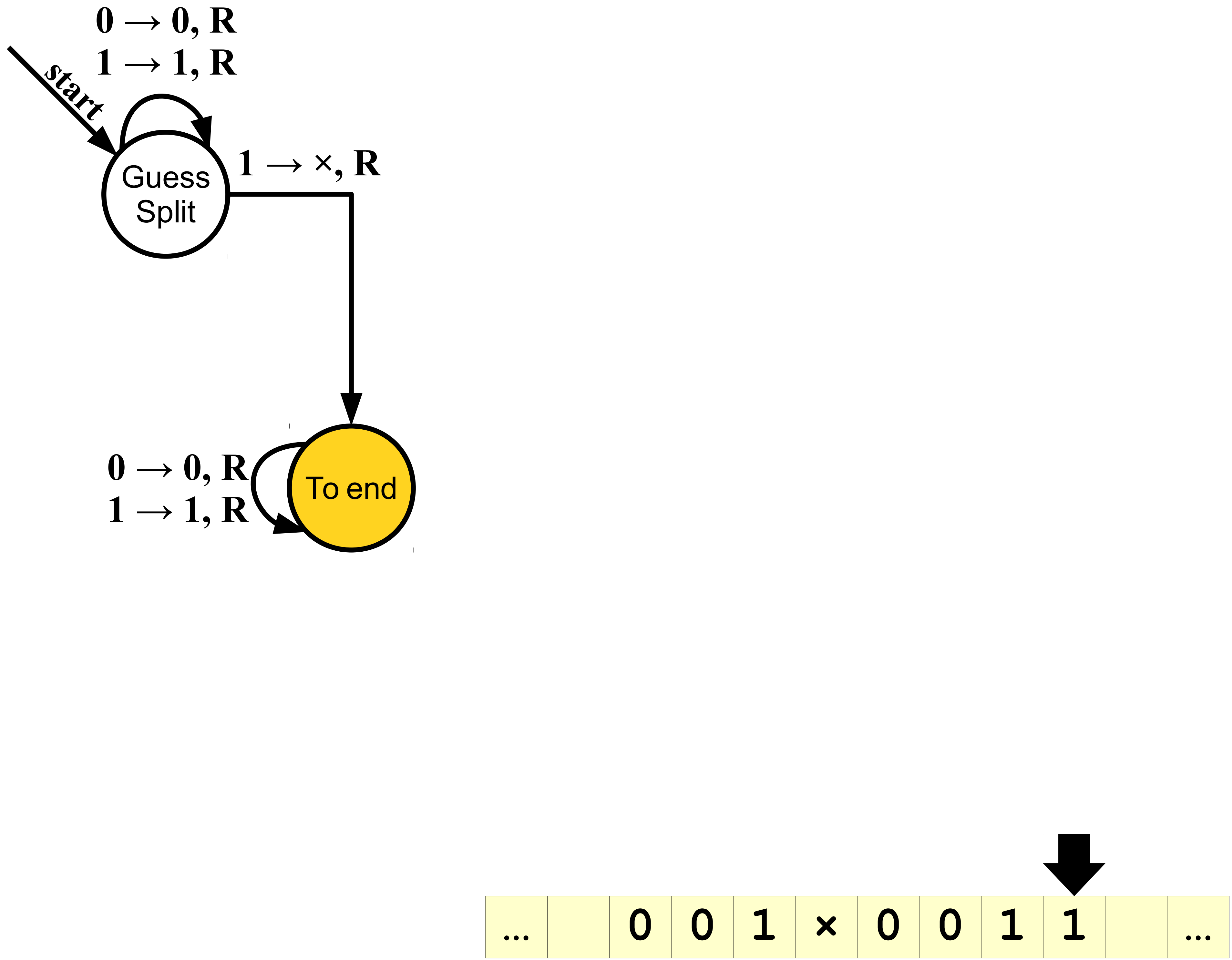

…
0
0
1
×
0
0
1
1
…


0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end

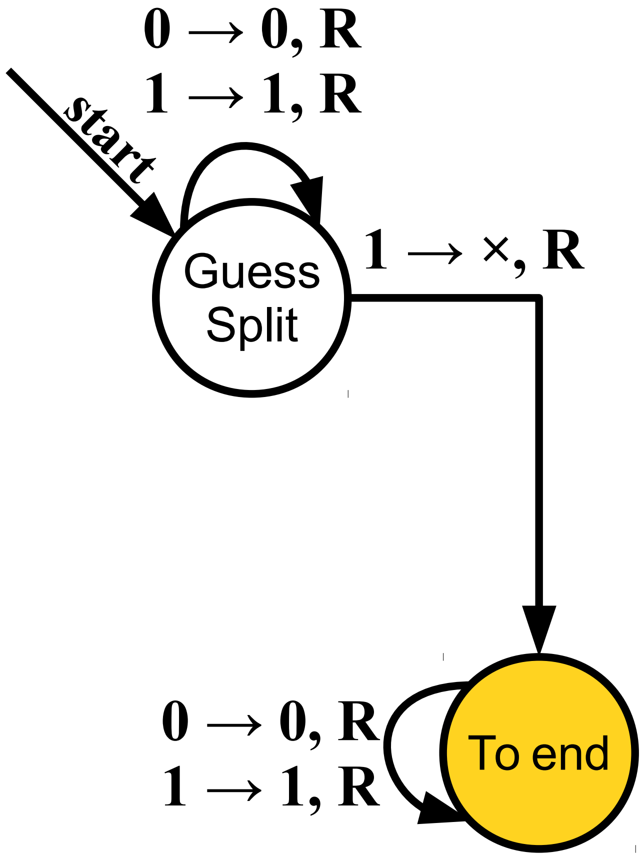

...
0
0
1
×
0
0
1
1
...

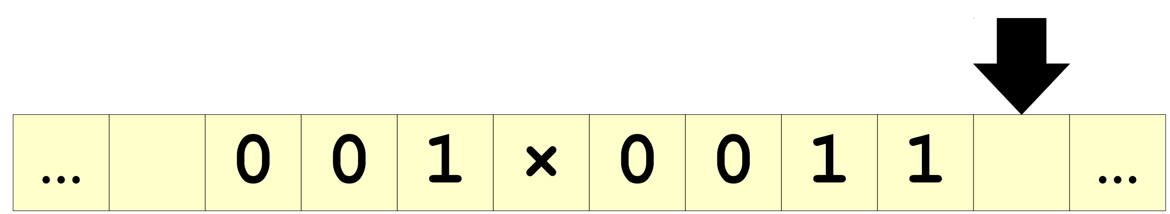

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end

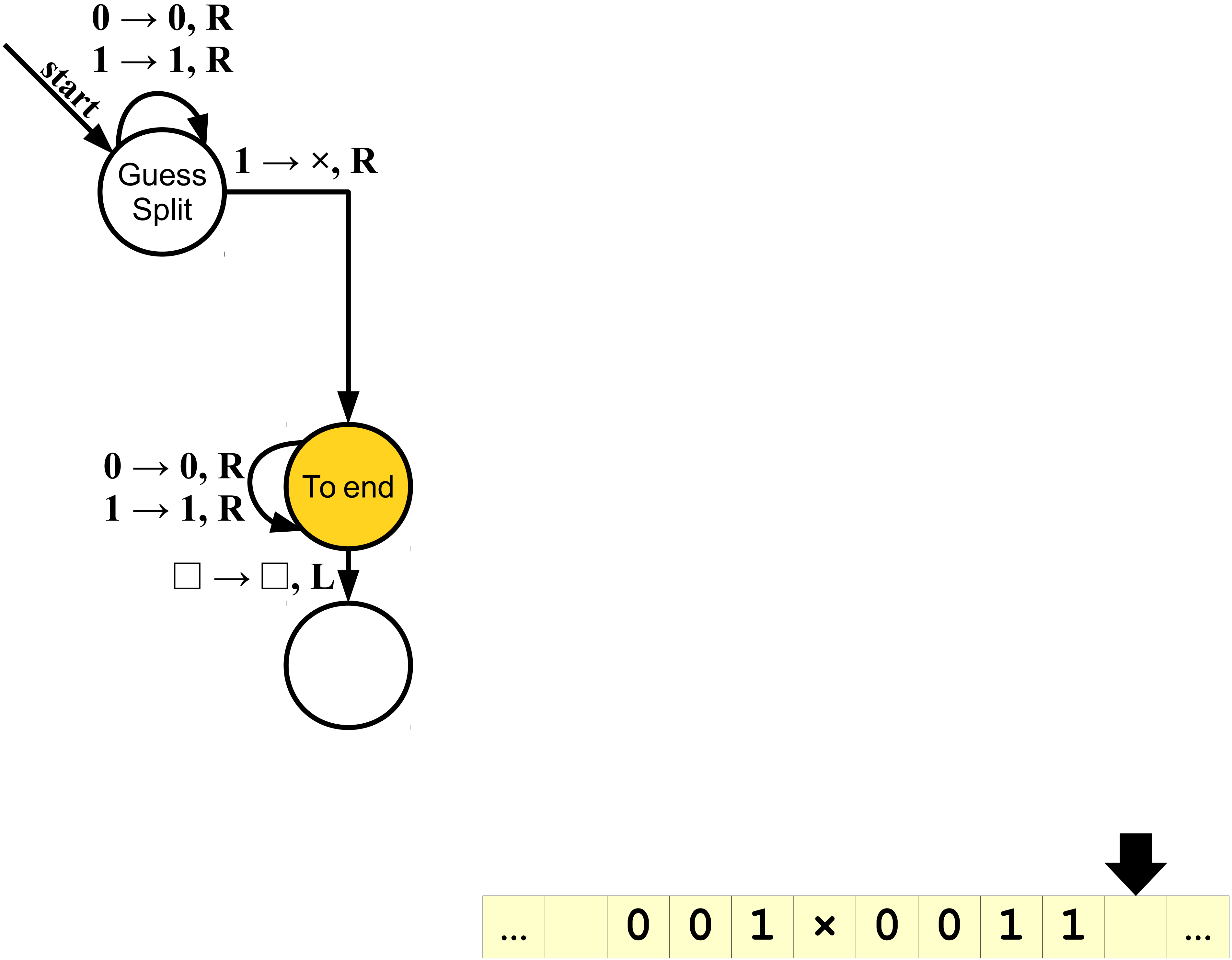

...
0
0
1
×
0
0
1
1
...

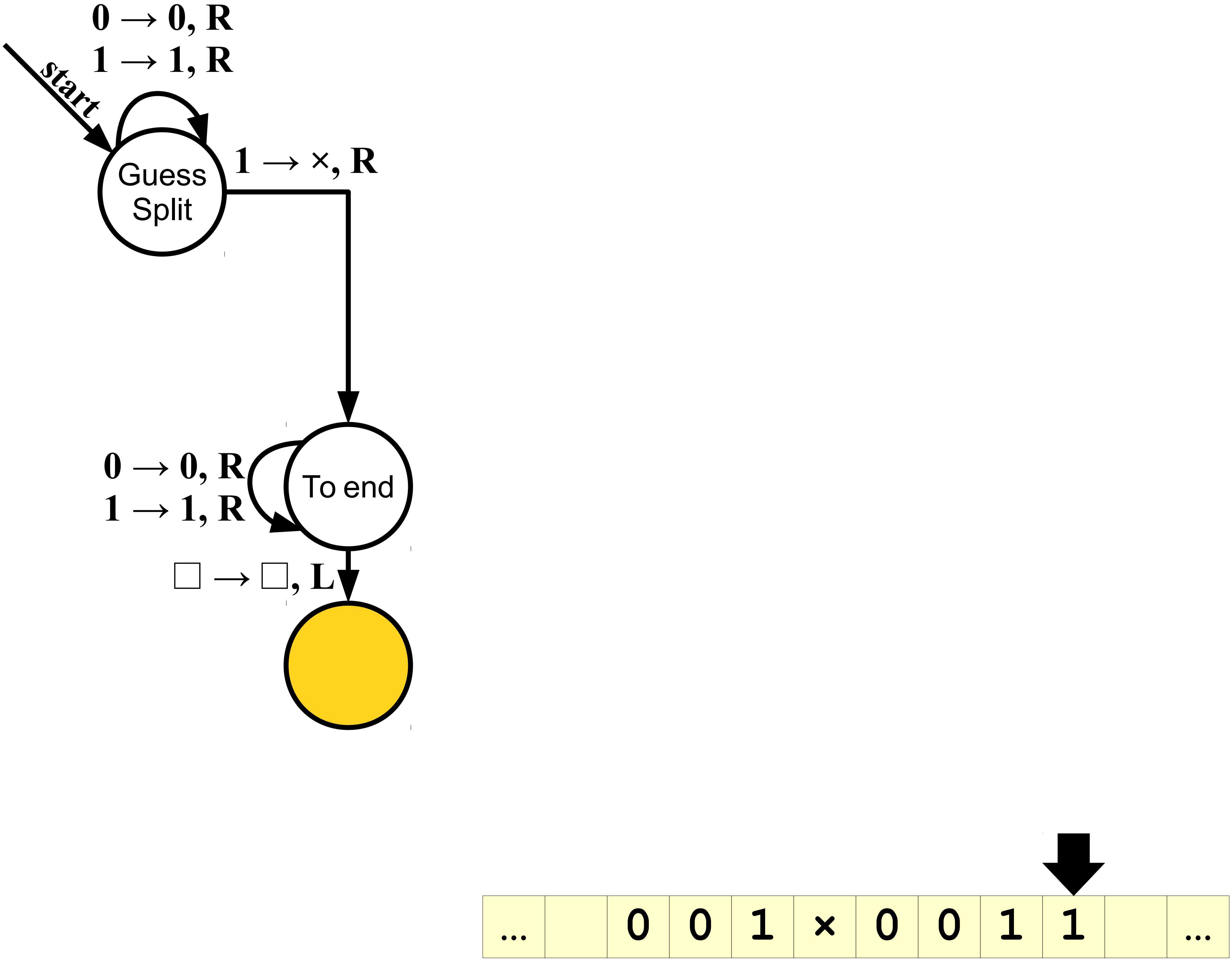


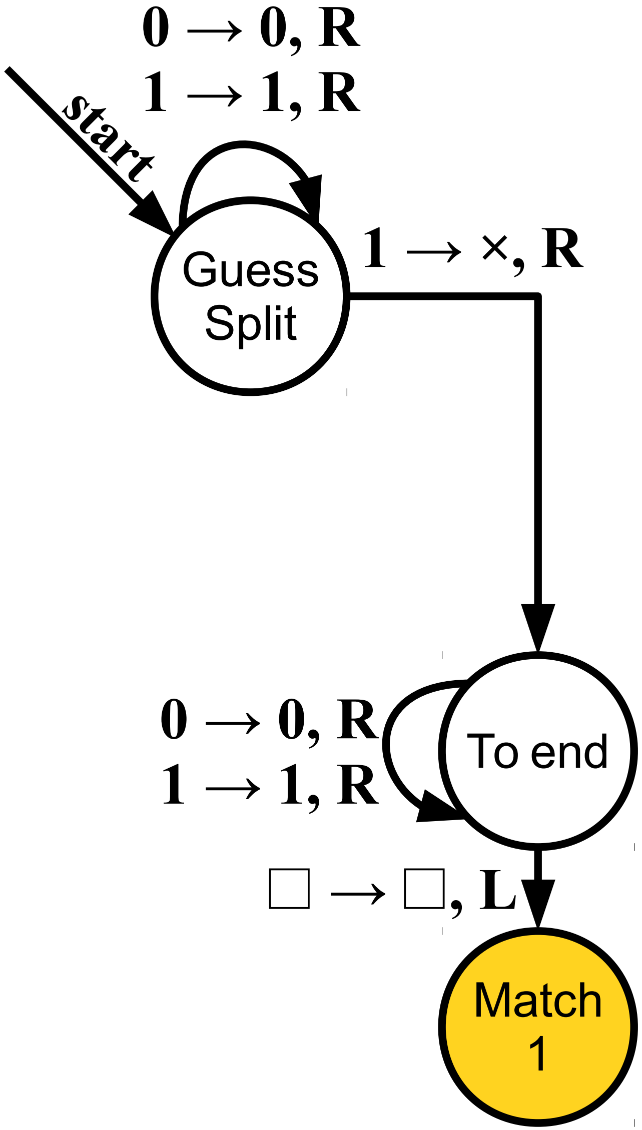
0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
… 0 0 1 × 0 0 1 1 …

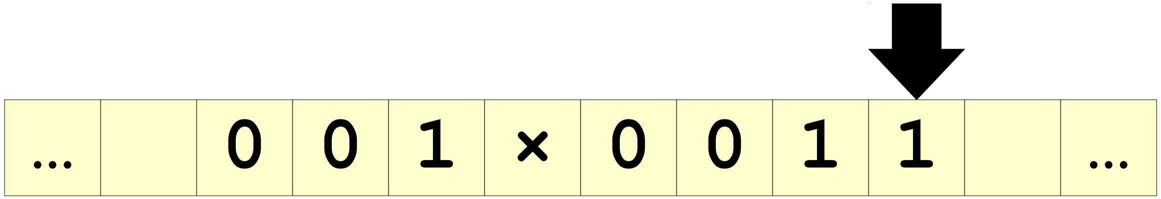

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end

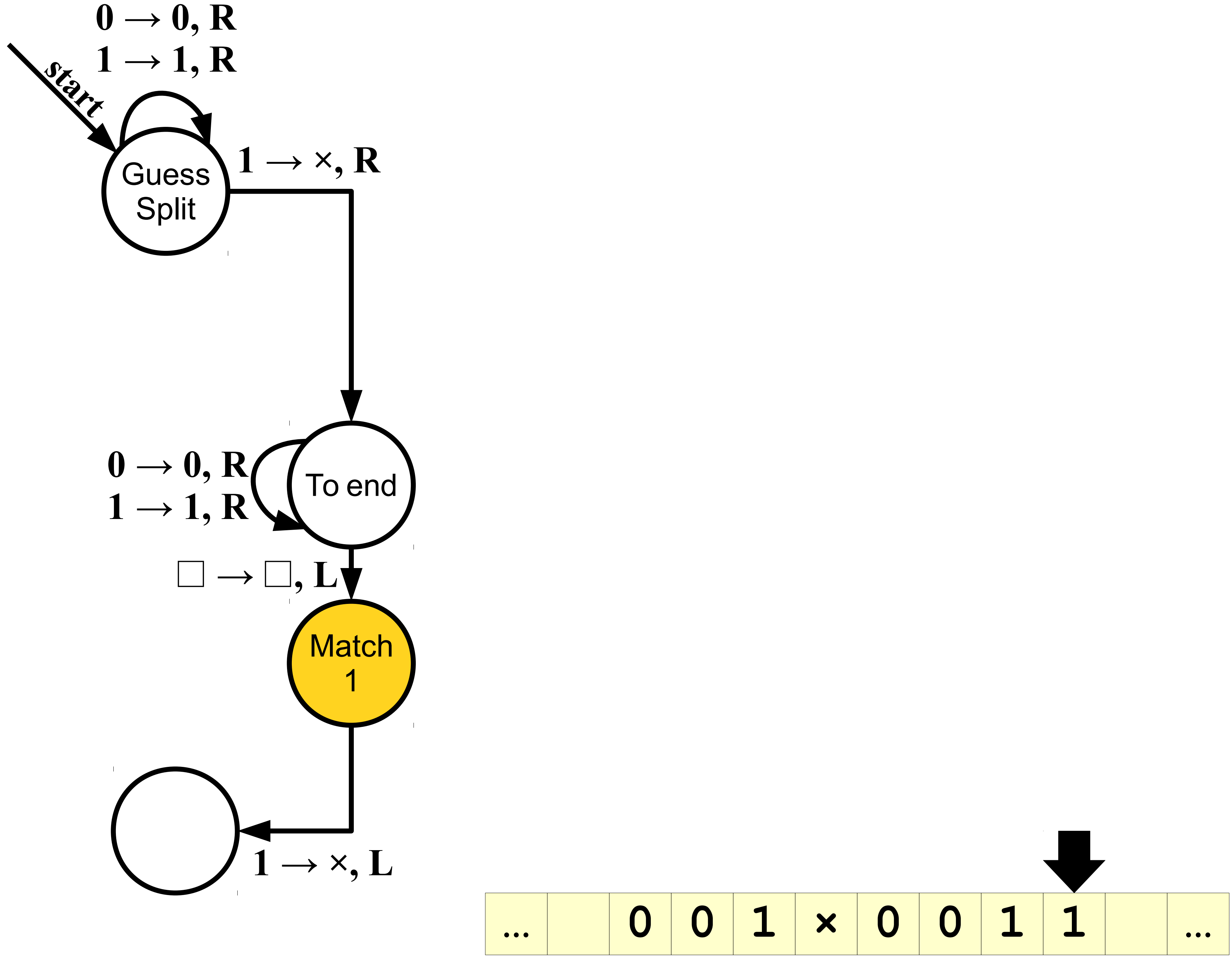

... 0 0 1 × 0 0 1 1 ...

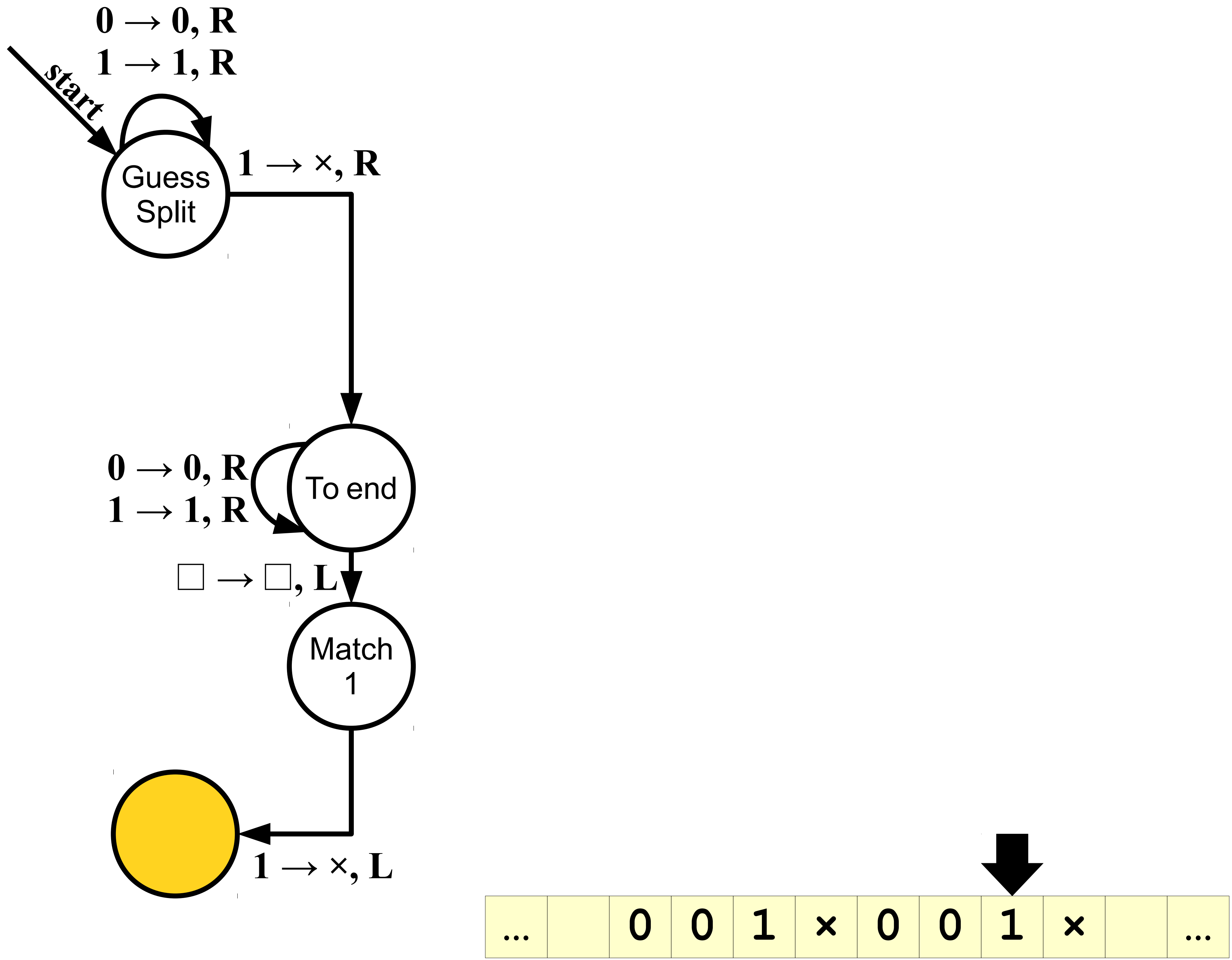

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
... 0 0 1 × 0 0 1 1 ...


0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
... 0 0 1 × 0 0 1 1 ...

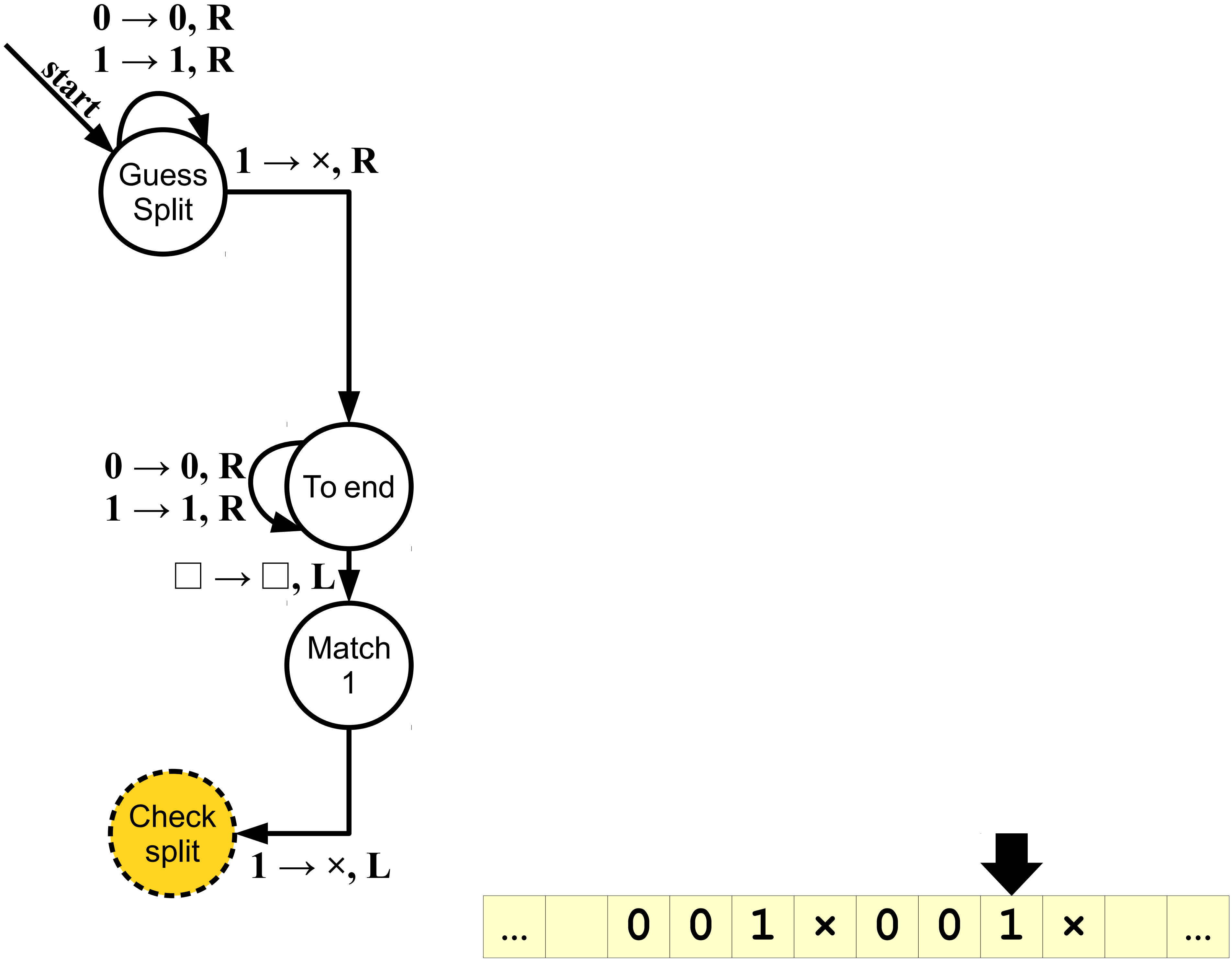

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1

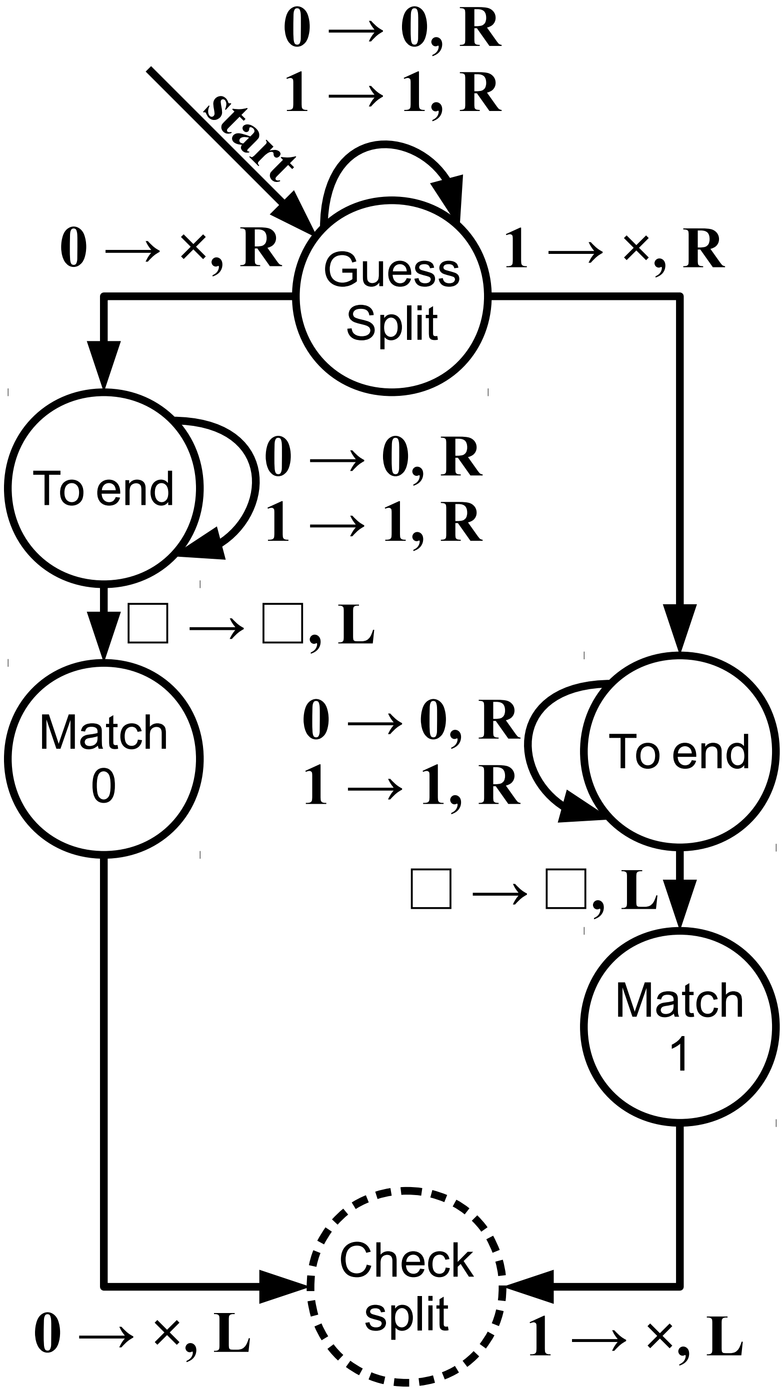

...
0
0
1
×
0
0
1
1
...

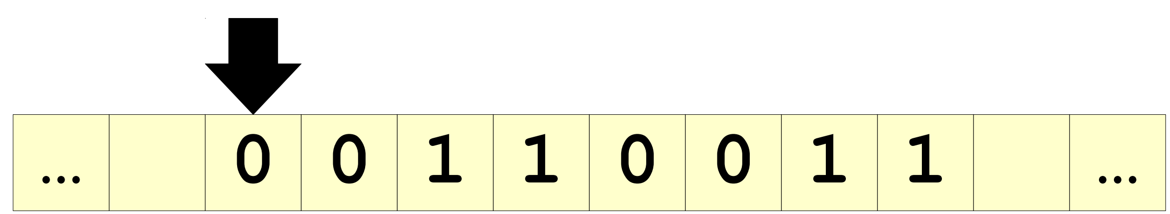

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
1 → ×, L
... 0 0 1 × 0 0 1 1 ...



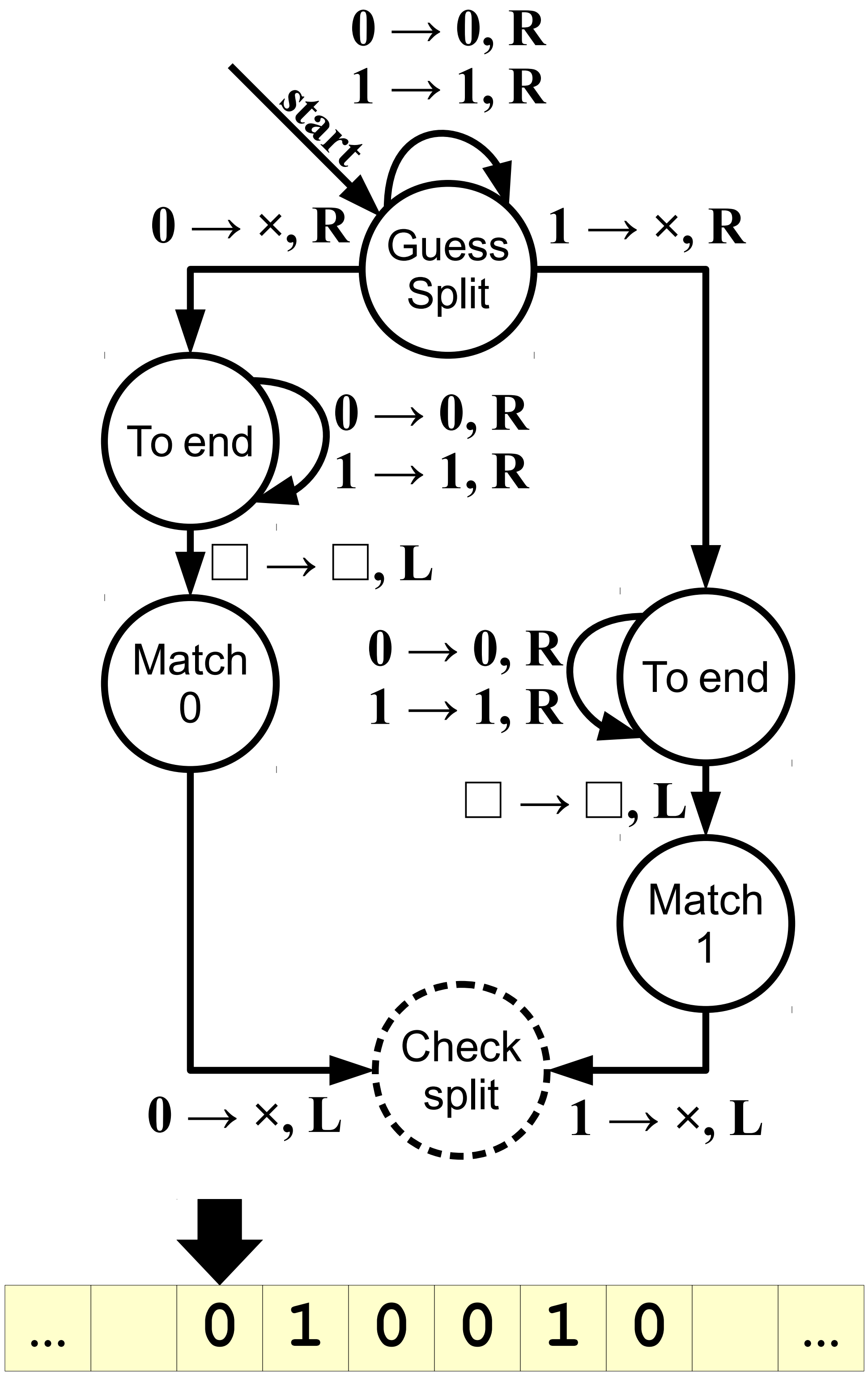
0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
1 → ×, L
…
0
0
1
×
0
0
1
×
…

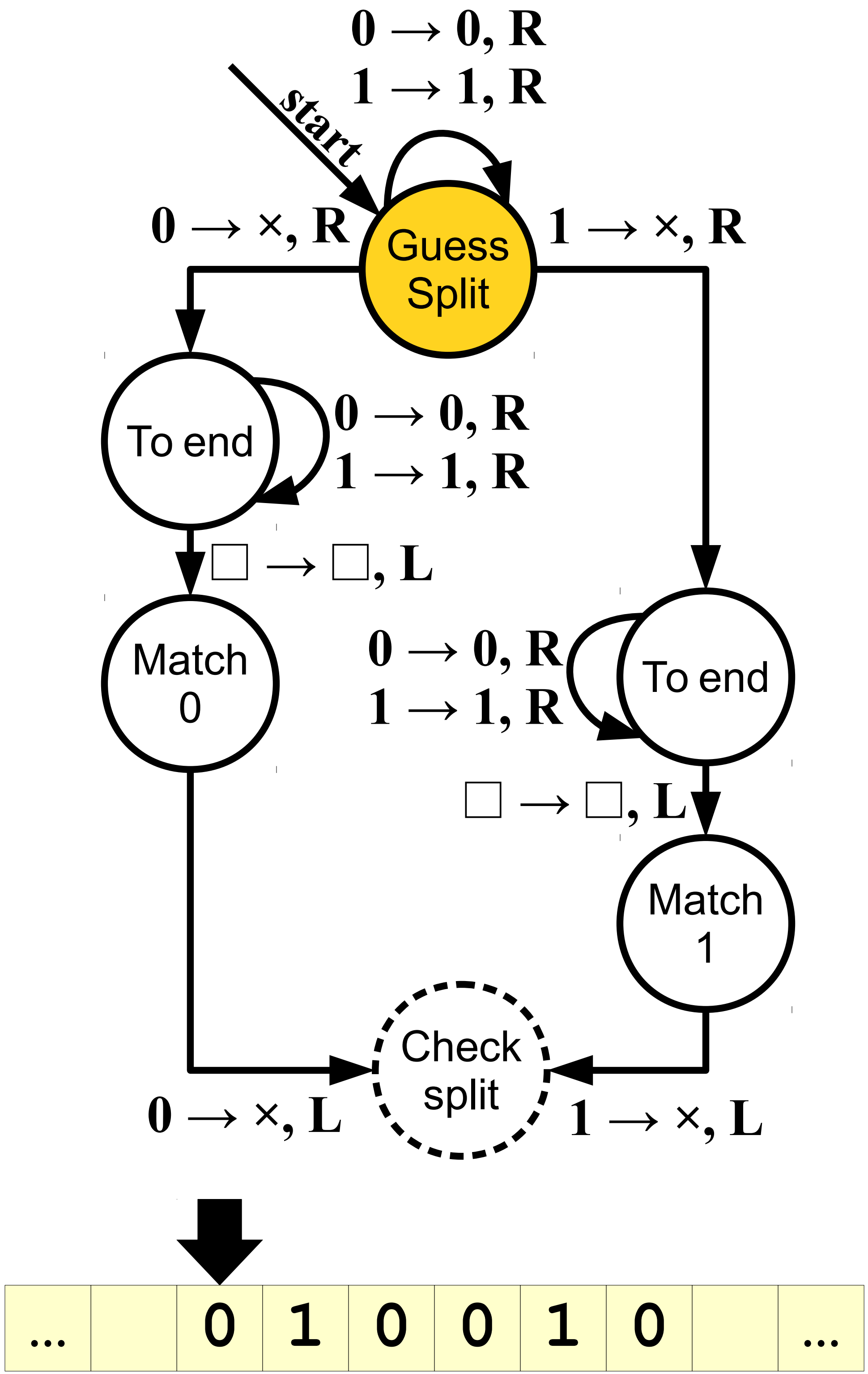

0 → 0, R
1 → 1, R
start
Guess Split
1 → ×, R
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
Check split
1 → ×, L
...
0
0
1
×
0
0
1
×
...


0 → 0, R
1 → 1, R
start
Guess Split
0 → ×, R
1 → ×, R
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L

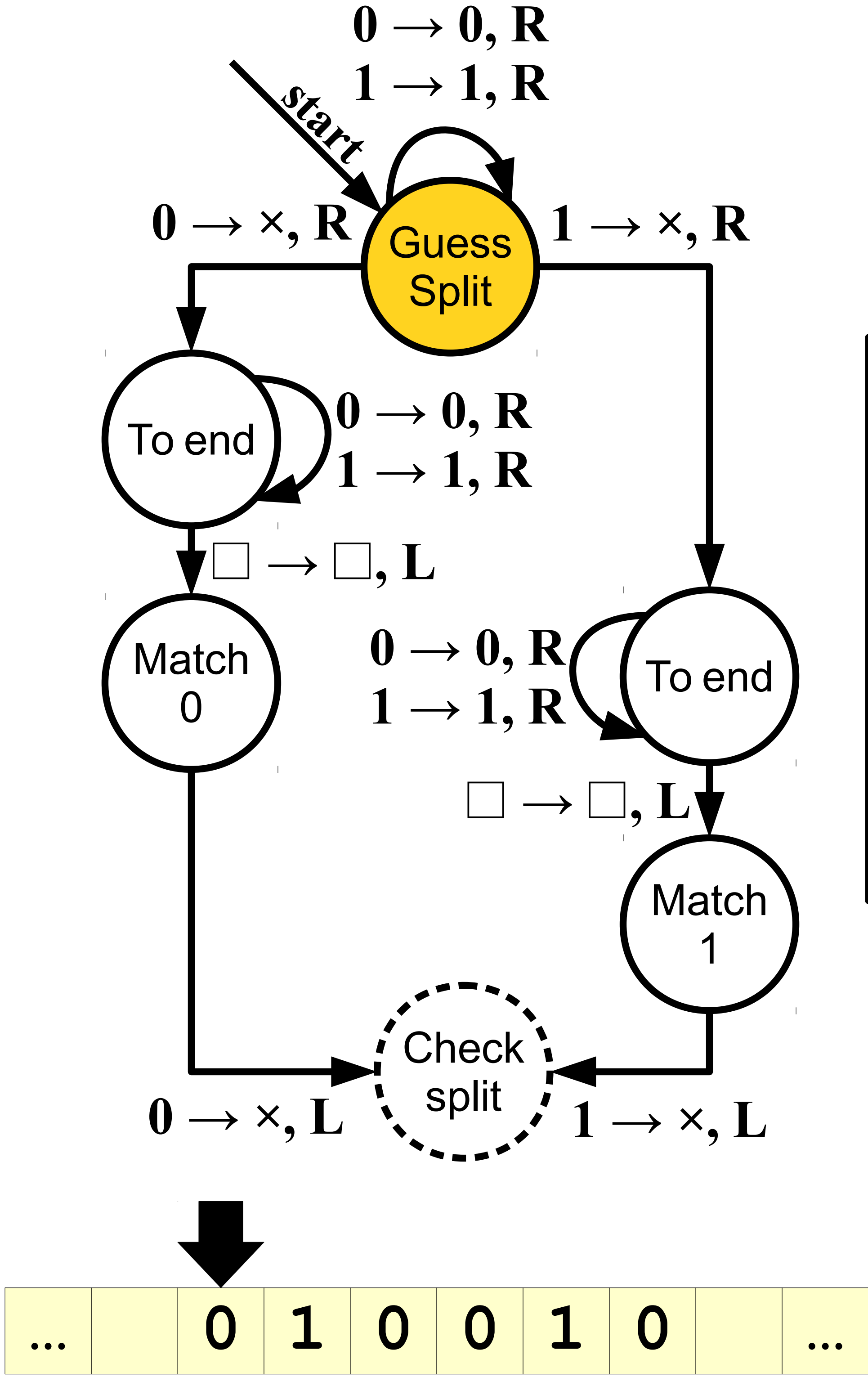

…
0 0 1 1 0 0 1 1
…

A ***huge*** difference between NTMs and NFAs.

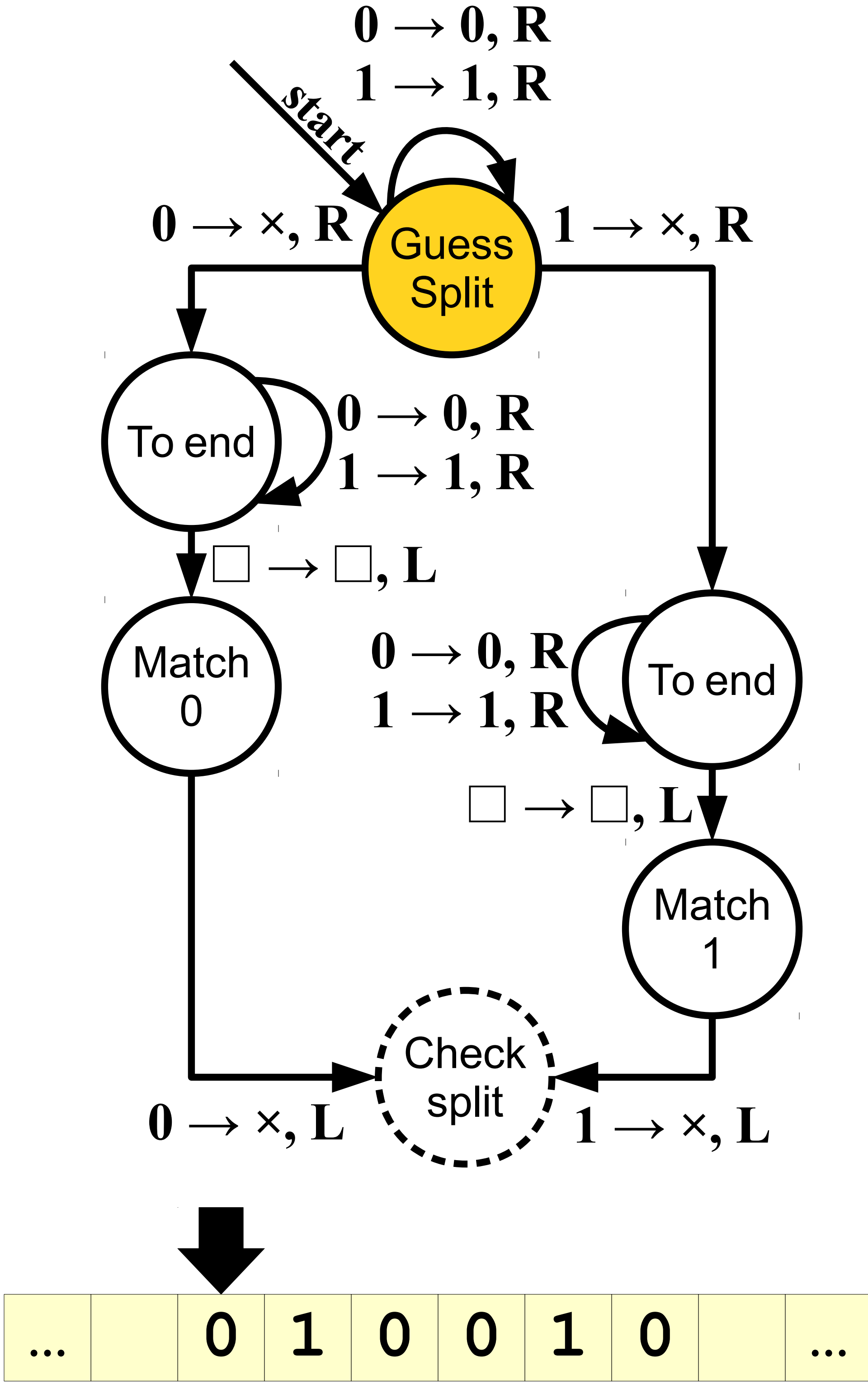

0 → 0, R
1 → 1, R
start
0 → ×, R
1 → ×, R
Guess Split
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
…
0
1
0
0
1
0
…



0 → 0, R
1 → 1, R
start
0 → ×, R
1 → ×, R
Guess Split
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
… 0 1 0 0 1 0 …

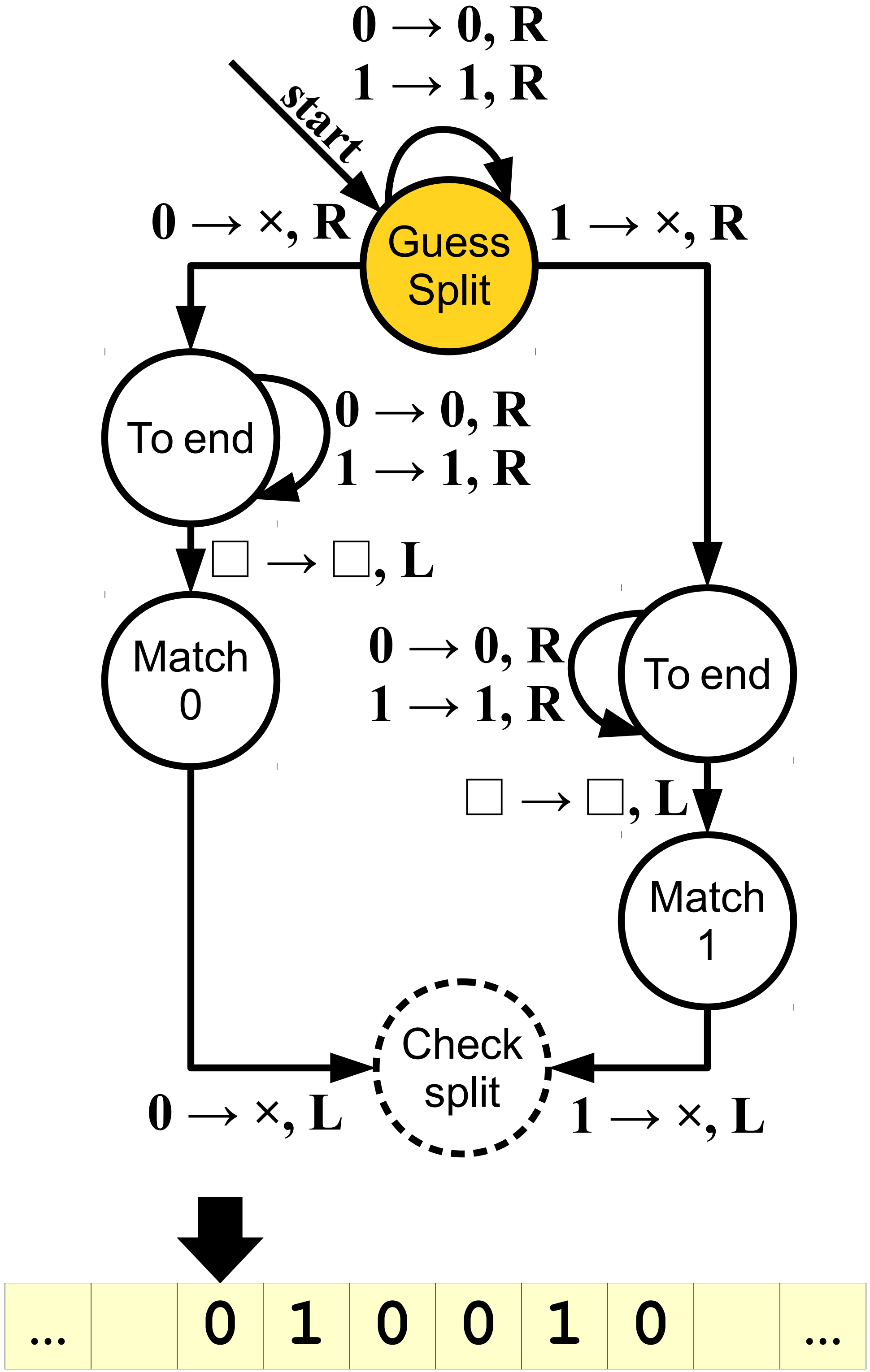
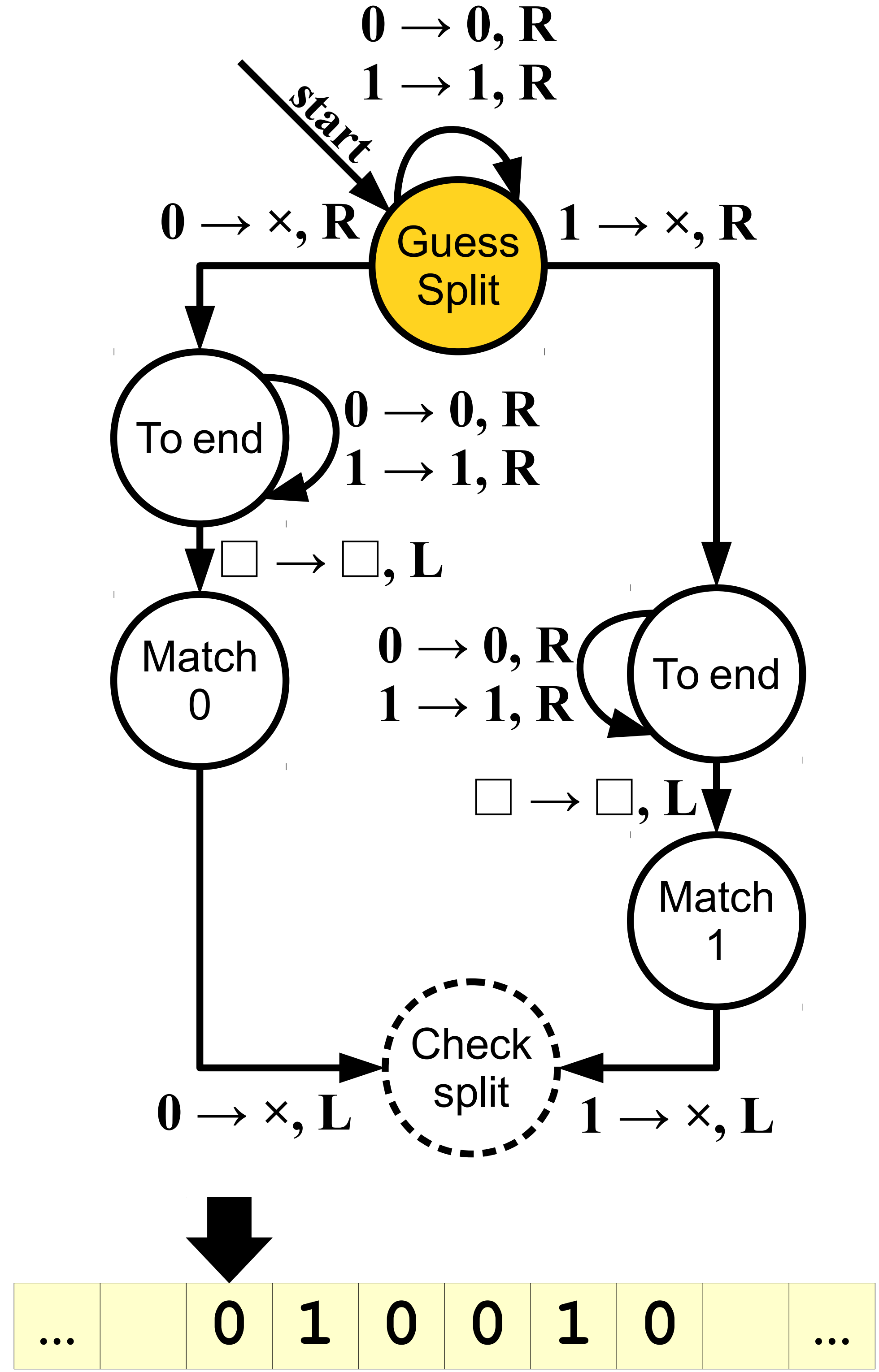


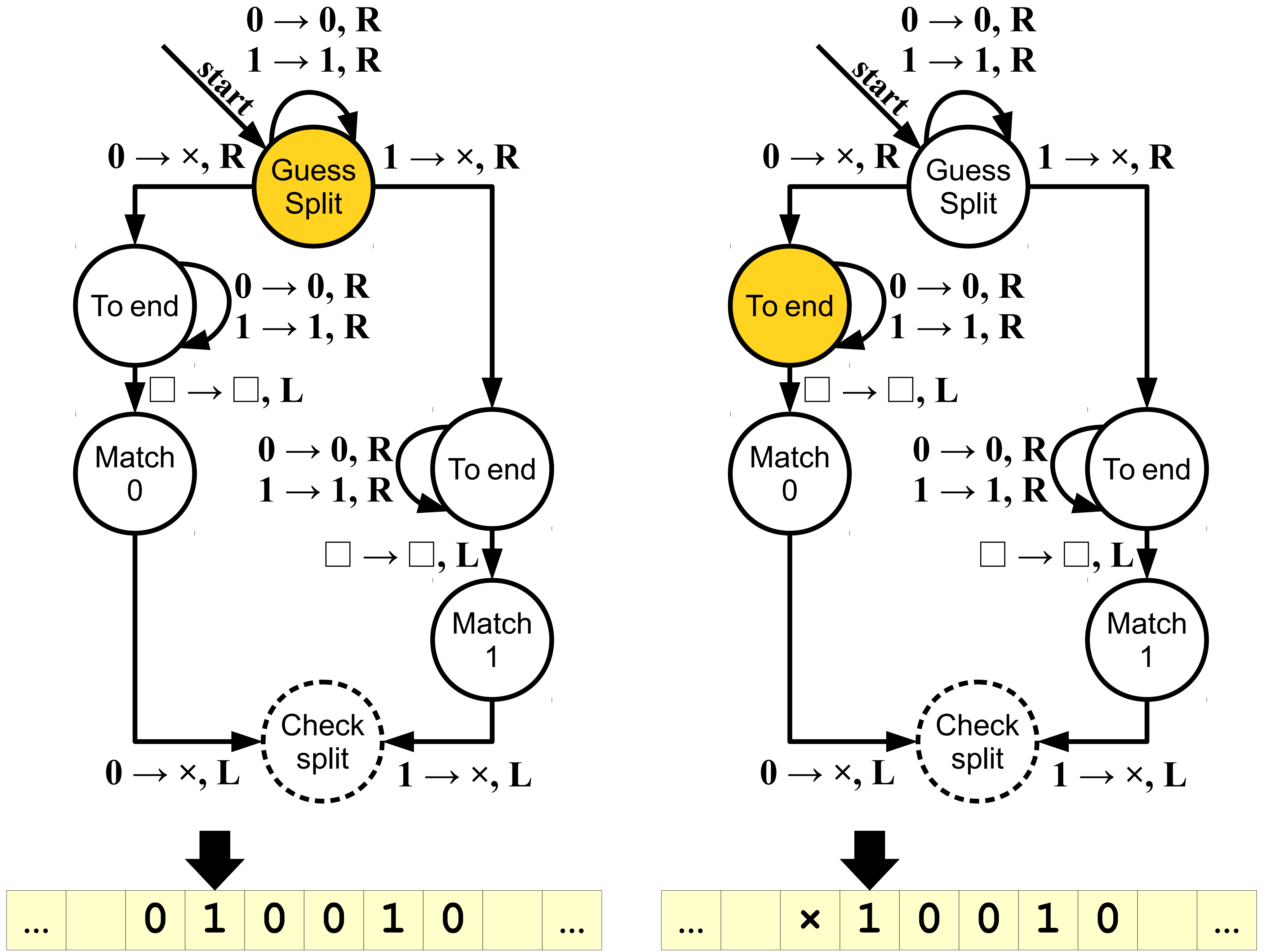
In an NFA, we can follow multiple transitions at once by just being in many states at the same time.



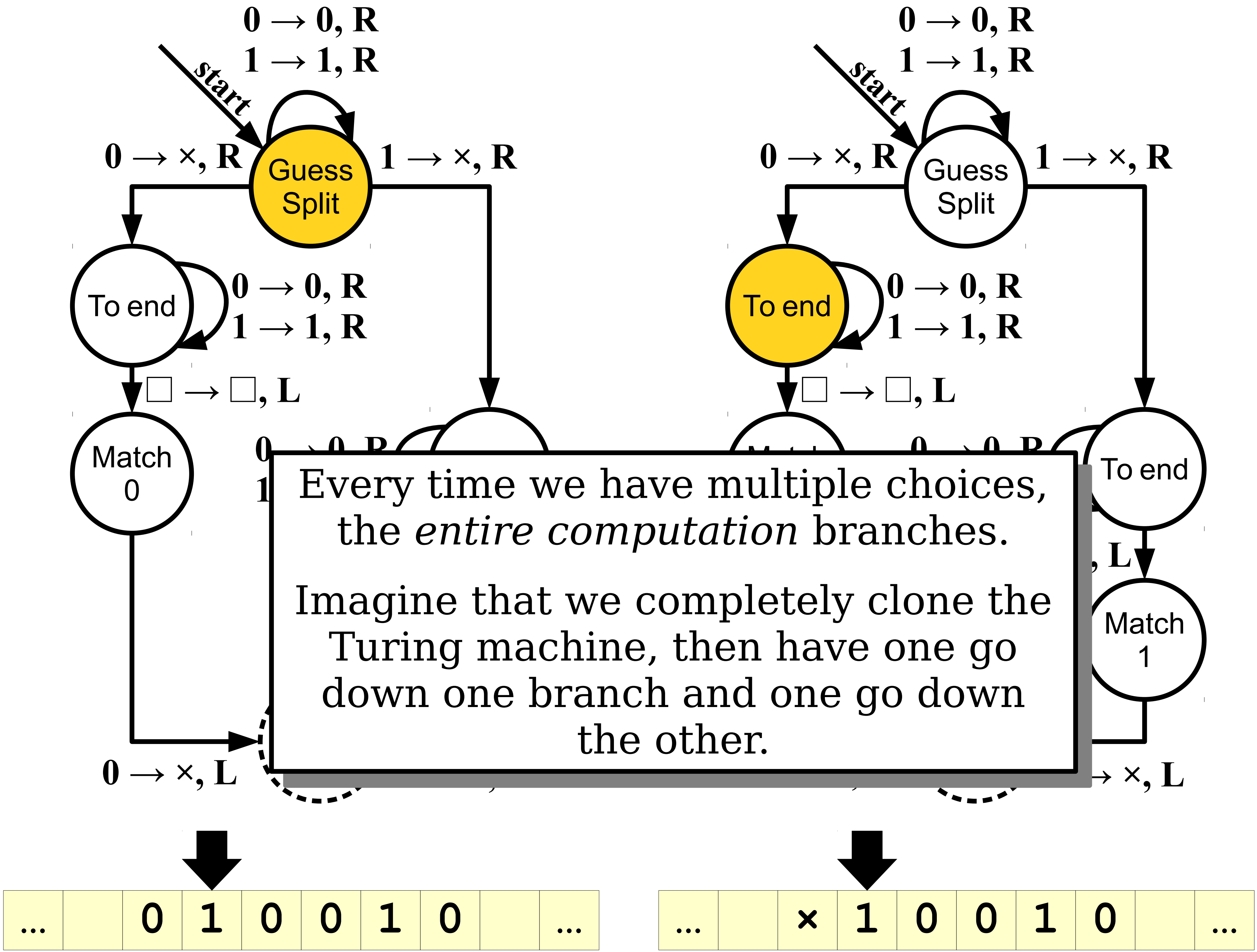
In an NFA, we can follow multiple transitions at once by just being in many states at the same time.

That doesn't work with NTMs! The tapes will be different in each case.


0 → 0, R
1 → 1, R
start
0 → ×, R
Guess Split
1 → ×, R
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
...
0
1
0
0
1
0
...

0 → 0, R
1 → 1, R
start
0 → ×, R
Guess Split
1 → ×, R
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
0 → 0, R
1 → 1, R
To end
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
...
0
1
0
0
1
0
...



start
0 → 0, R
1 → 1, R
Guess Split
0 → ×, R
1 → ×, R
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
… 0 1 0 0 1 0 …
start
0 → 0, R
1 → 1, R
Guess Split
0 → ×, R
1 → ×, R
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 0
To end
0 → 0, R
1 → 1, R
□ → □, L
Match 1
Check split
0 → ×, L
1 → ×, L
… × 1 0 0 1 0 …



Every time we have multiple choices, the *entire computation* branches.

Imagine that we completely clone the Turing machine, then have one go down one branch and one go down the other.

Intuiting Nondeterministic TMs

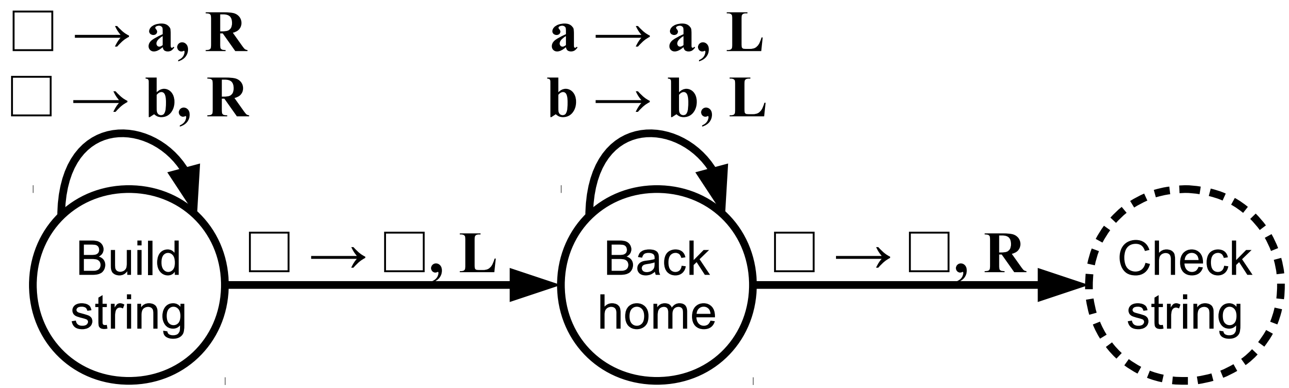
- Two of our previous NTM intuitions are useful here:
- ***Perfect guessing:*** If there is some choice of transitions that leads to an accepting state, the NTM can perfectly guess those transitions.
  - There's just one NTM, and it makes the right guess if one exists.
- ***Massive parallelism:*** The NTM tries all options. Each time it follows multiple transitions, it copies the current state of the machine once for each option, then tries each option.
  - Each step of the computation creates multiple new NTMs to try out each branch.
- The "guess-and-check" intuition from NFAs still applies here and is probably the best way to design NTMs.

Guessing Arbitrary Objects

- NTMs can use their nondeterminism to guess an arbitrary discrete, finite object.
- ***Idea:*** The NTM nondeterministically chooses a string to write on its tape, then does something with the string it just wrote.

Guessing an Arbitrary String

- As an example, here's how an NTM can guess an arbitrary string, then go do something with it:



- As a high-level description:

> *N* = "On input *w*:
>
> Nondeterministically guess a string $x \in \Sigma^*$.
>
> Deterministically check whether […]"

Just How Powerful are NTMs?

NTMs and DTMs

- ***Theorem:*** If $L \in \mathbf{RE}$, then there is an NTM for L.
- ***Proof Sketch:*** Every deterministic TM (DTM) can be thought of as an NTM with no nondeterminism, so if L is the language of a DTM, it's also the language of an NTM. ■

NTMs and DTMs

- ***Theorem:*** If L is the language of an NTM, then $L \in$ **RE**.
- ***Faulty Proof Idea:*** Use the subset construction.
- Why doesn't this work?
  - In an NFA, the only "memory" is which states are active, so creating one state per configuration simulates the NFA with a DFA.
  - In an NTM, the memory is the current state plus the tape contents, so building one state per configuration is impossible.

NTMs and DTMs

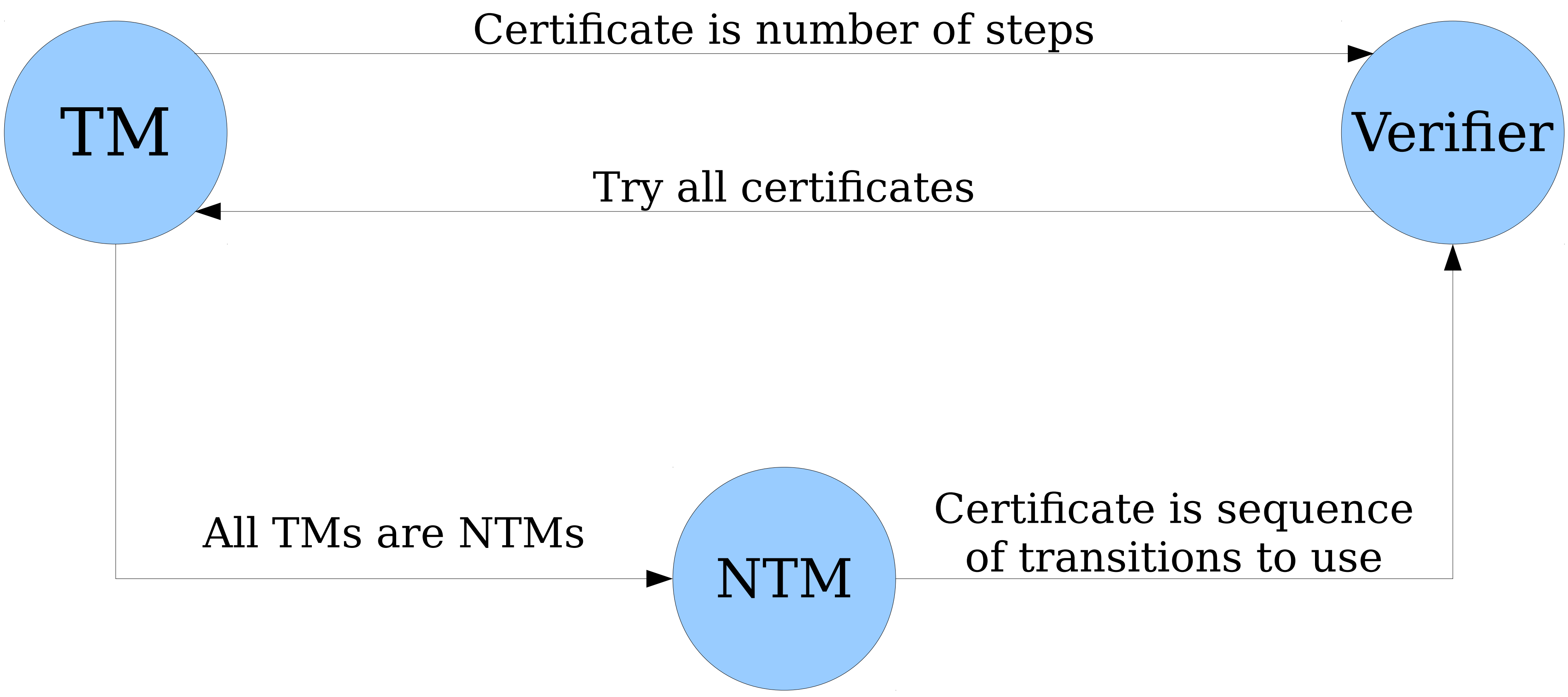
- ***Theorem:*** If L is the language of an NTM, then $L \in \mathbf{RE}$.
- ***Proof Idea:*** Show how to construct an verifier for L using the NTM.
- We showed how to build a verifier for an arbitrary TM M by having the certificate for some $w \in L$ be the number of steps it takes for M to accept w.
- With an NTM N, there might be many possible executions of length n on the string w.
- **Idea:** Our certificate will be the series of transitions that N is supposed to follow to accept w.

NTMs and DTMs

- ***Theorem:*** If L is the language of an NTM, then $L \in$ **RE**.
- ***Proof Sketch:*** Let N be an NTM for L. Then we can prove that this is a verifier for L:

V = "On input $\langle w, T \rangle$, where T is a sequence of transitions:
- Run N on w, following transitions in the order specified in T.
- If any of the transitions in T are invalid or can't be followed, reject.
- If after following the transitions N accepts w, accept; otherwise reject.

Three Views of **RE**



Why This Matters

- We now have three perspectives on the **RE** languages:
  - They're languages where a TM can *search* for proof that a string in the language.
  - They're languages where a verifier can *check* a proof that a string is in the language.
  - They're languages where an NTM can *guess* a proof that a string is in the language.
- All of this comes back to the notion of *proving strings in the language*: you might not be able to *determine* whether a string is in an **RE** language, but if a string is in an **RE** language, there is some way to prove it.

Time-Out for Announcements!

Solution Sets

- Solutions for PS6 are available for pickup outside lecture.
- We've also just released the solutions to the final set of practice problems.
- If you need to pick up solutions after class, check out the filing cabinets in the Gates building; directions are up on the course website.
- We'll be recycling PS1 – PS3 and related solution sets later this week – grab them if you want them!

Words of Encouragement

- Good luck on the exam tomorrow!
- Stressed? Worried? Losing perspective? Check the course website for two links:
  - The video "A Math Major Talks about Fear."
  - The article "How I Faced my Fears and Learned to be Good at Math."
- Hope this helps!

Upcoming Event

- Senator Dianne Feinstein will be visiting Stanford a week from tomorrow to talk about what role congress does and should play in overseeing the CIA and NSA.
- Talk will be Thursday, May 28 at 6:30PM at Cemex.
- RSVP required: ***click here***!

Your Questions

"What are your thoughts on the Stanford core undergraduate requirements? Do you ever find it embarrassing that engineering students here can graduate without ever learning things like linear algebra, optics, and differential equations?"

So, um, I never learned optics. I did learn differential equations and have never used them, though linear algebra is super useful and really cool.

I really like the idea of breadth requirements, though I think that we as a university need to do a better job getting people excited about them. If anything, we should have more breadth requirements in more areas.

“What are some things that you wish Stanford could change or do differently?”

A few thoughts:

1. Help students gain perspective on the purpose of their education and empower them to make more informed decisions about how to approach their classes and their course selection.

2. Stress the importance of accountability and honesty and give students space to make mistakes and recover from them.

3. Open better lines of communication between students and faculty members to ensure that concerns are heard and that all parties involved in education understand the needs and goals of all others.

Back to CS103!

Beyond **RE**

- What would it mean for a language not to be in class **RE**?
- That would mean that
  - There's no algorithm for checking whether a string is in the language (because $\mathbf{R} \subseteq \mathbf{RE}$).
  - There's no algorithm for checking a proof that a string is in the language.
- In a sense, languages outside of **RE** are so complex, even if you know for a fact a string is in the language, you may not be able to prove it!

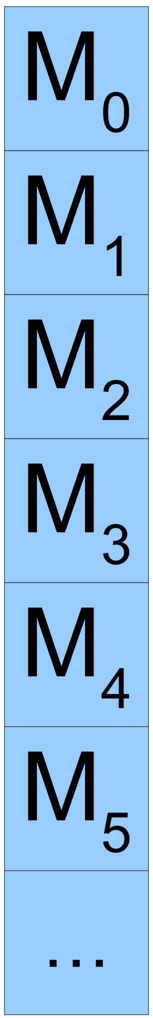
Going Beyond **RE**

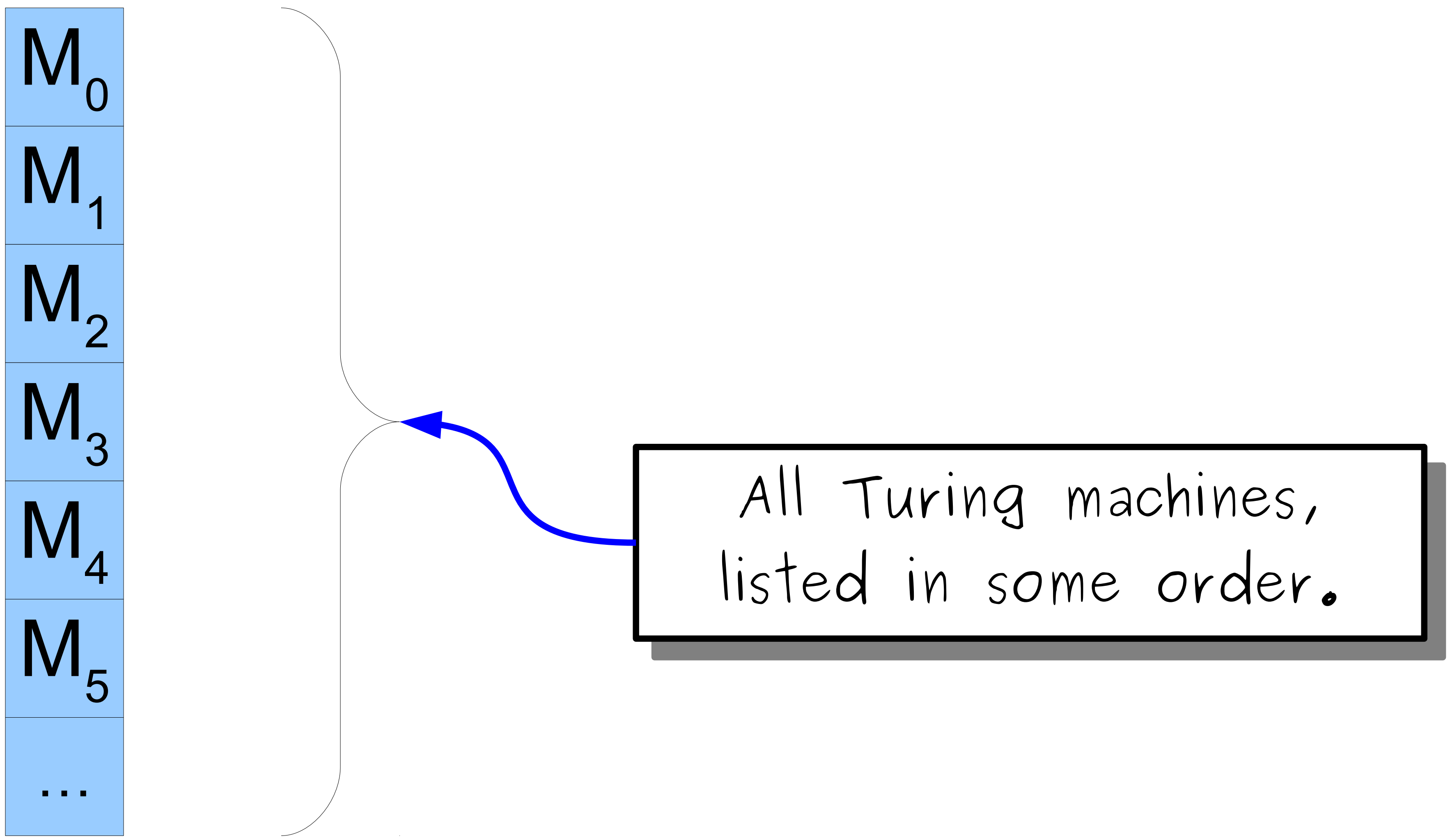
Languages, TMs, and TM Encodings

- Recall: The language of a TM M is the set

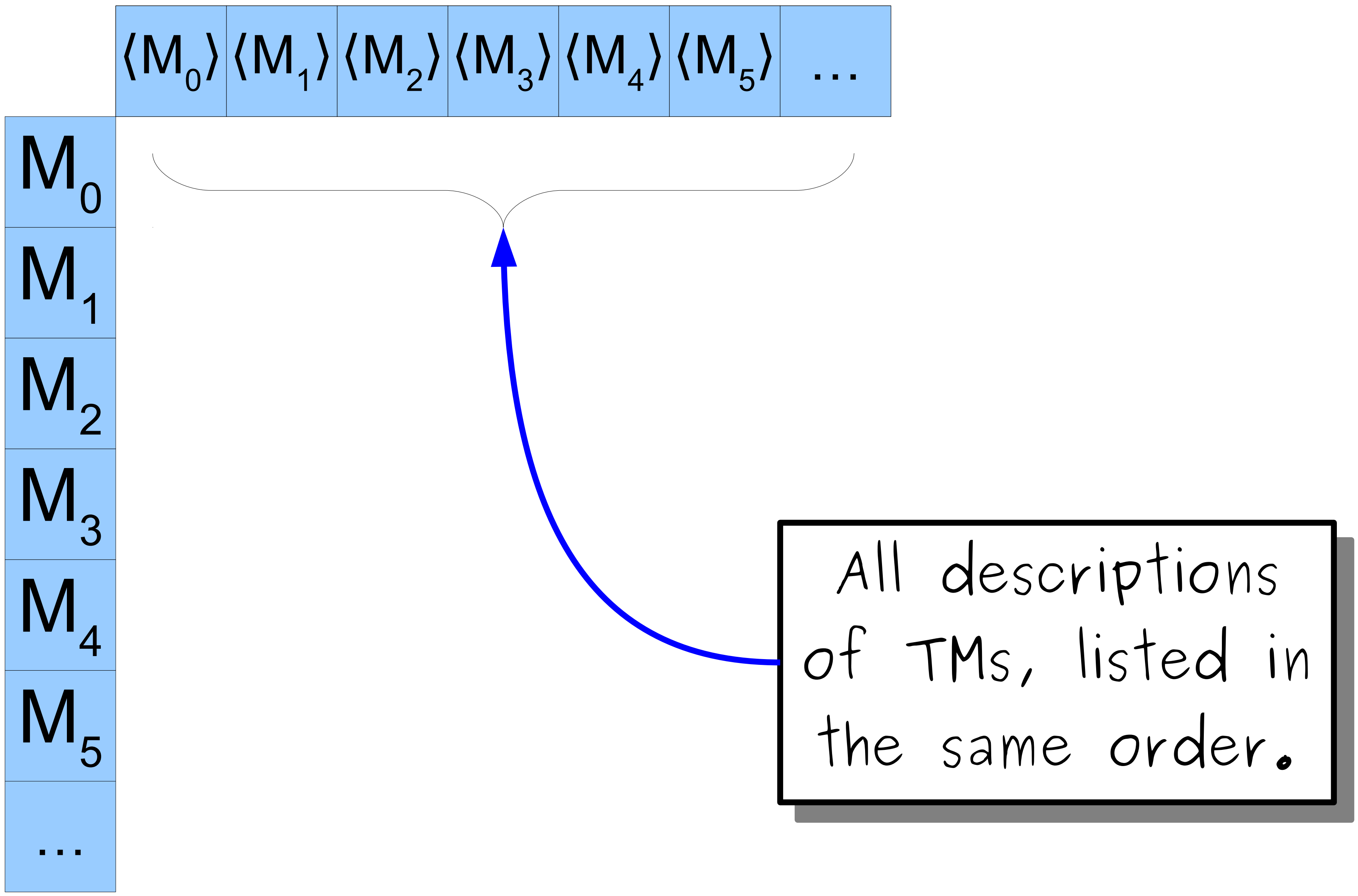
$$\mathscr{L}(\boldsymbol{M}) = \{ \boldsymbol{w} \in \boldsymbol{\Sigma^*} \mid \boldsymbol{M} \textbf{ accepts } \boldsymbol{w} \}$$

- Some of the strings in this set might be descriptions of TMs.
- What happens if we just focus on the set of strings that are legal TM descriptions?


M_0
M_1
M_2
M_3
M_4
M_5
…


M0
M1
M2
M3
M4
M5
...
All Turing machines, listed in some order.

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0							
M_1							
M_2							
M_3							
M_4							
M_5							
…							


⟨M0⟩ ⟨M1⟩ ⟨M2⟩ ⟨M3⟩ ⟨M4⟩ ⟨M5⟩ ...
M0
M1
M2
M3
M4
M5
...
All descriptions of TMs, listed in the same order.

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1							
M_2							
M_3							
M_4							
M_5							
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2							
M_3							
M_4							
M_5							
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3							
M_4							
M_5							
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4							
M_5							
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5							
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…							

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

Acc	Acc	Acc	No	Acc	No	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

Flip all "accept" to "no" and vice-versa

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

What TM has this behavior?

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…
	No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…
	No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

No TM has this behavior!

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

	⟨M_0⟩	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩	⟨M_5⟩	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

"The language of all TMs that do not accept their own description."

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

$\{ \langle M \rangle \mid M$ is a TM that does not accept $\langle M \rangle \}$

	$\langle M_0 \rangle$	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	…
M_0	Acc	No	No	Acc	Acc	No	…
M_1	Acc	Acc	Acc	Acc	Acc	Acc	…
M_2	Acc	Acc	Acc	Acc	Acc	Acc	…
M_3	No	Acc	Acc	No	Acc	Acc	…
M_4	Acc	No	Acc	No	Acc	No	…
M_5	No	No	Acc	Acc	No	No	…
…	…	…	…	…	…	…	…

No	No	No	Acc	No	Acc	…

$$\{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$$

Diagonalization Revisited

- The ***diagonalization language***, which we denote $\boldsymbol{L}_{\mathbf{D}}$, is defined as

$$\boldsymbol{L}_{\mathbf{D}} = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$$

- That is, L_{D} is the set of descriptions of Turing machines that do not accept themselves.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

Because $\mathscr{L}(R) = L_D$, we know that a string belongs to one set if and only if it belongs to the other.

$L_D = \{ \langle M \rangle \mid M$ is a TM and $\langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

$$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$$

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

We've replaced the left-hand side of this biconditional with an equivalent statement.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$.

$L_D = \{ \langle M \rangle \mid M$ is a TM and $\langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$.

A nice consequence of a universally-quantified statement is that it should work in all cases.

$L_D = \{ \langle M \rangle \mid M$ is a TM and $\langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathscr{L}(R) \quad \text{iff} \quad \langle R \rangle \in \mathscr{L}(R) \qquad (3)$$

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin \mathbf{RE}$.

Proof: By contradiction; assume that $L_D \in \mathbf{RE}$. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathscr{L}(R) \text{ iff } \langle R \rangle \in \mathscr{L}(R) \qquad (3)$$

This is clearly impossible.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathscr{L}(R) \quad \text{iff} \quad \langle R \rangle \in \mathscr{L}(R) \qquad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong.

$L_D = \{ \langle M \rangle \mid M$ is a TM and $\langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \text{ iff } \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathscr{L}(R) \text{ iff } \langle R \rangle \in \mathscr{L}(R) \qquad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong. Thus $L_D \notin$ **RE**.

$L_D = \{ \langle M \rangle \mid M \text{ is a TM and } \langle M \rangle \notin \mathscr{L}(M) \}$

Theorem: $L_D \notin$ **RE**.

Proof: By contradiction; assume that $L_D \in$ **RE**. Then there must be some TM R such that $\mathscr{L}(R) = L_D$.

Since $\mathscr{L}(R) = L_D$, we know that if M is any TM, then

$$\langle M \rangle \in L_D \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (1)$$

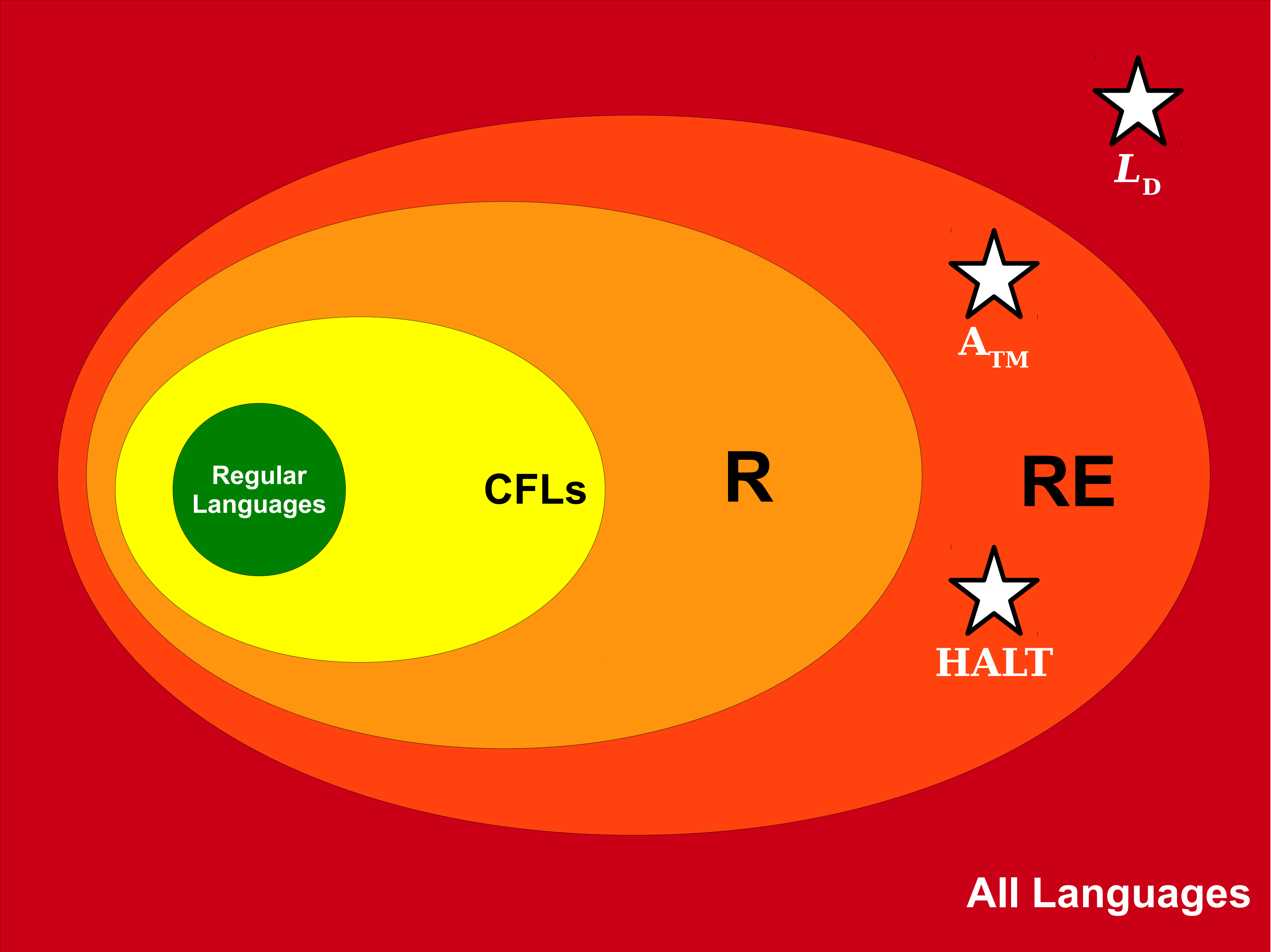
From the definition of L_D, we see that $\langle M \rangle \in L_D$ iff $\langle M \rangle \notin \mathscr{L}(M)$. Combining this with statement (1) tells us that

$$\langle M \rangle \notin \mathscr{L}(M) \quad \text{iff} \quad \langle M \rangle \in \mathscr{L}(R) \qquad (2)$$

Statement (2) holds for any TM M, so in particular it should hold when $M = R$. If we pick $M = R$, we see that

$$\langle R \rangle \notin \mathscr{L}(R) \quad \text{iff} \quad \langle R \rangle \in \mathscr{L}(R) \qquad (3)$$

This is clearly impossible. We have reached a contradiction, so our assumption must have been wrong. Thus $L_D \notin$ **RE**. ■



L_D
A_{TM}
Regular Languages
CFLs
R
RE
HALT
All Languages

Non-**RE** Languages

- We've just discovered a non-**RE** language.
- In a sense, though, we did it by cheating: we specifically constructed this problem so that *by construction* it cannot be solved!
- This technique won't generalize to other cases very well. For that, we'll need a new set of techniques.
- That said, the technique we used here is quite clever; you'll explore it in more depth in Problem Set Eight.