Probabilistic Language Inclusion Problems

Tobias Winkler

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

DFG Research Training Group 2236

European Research Council

IMDEA Software Institute, Madrid — 19.09.2023 aka “Probabilistic Model Checking”

Probabilistic Language Inclusion Problems

Tobias Winkler

Software Modeling and Verification Chair | RWTH AACHEN UNIVERSITY

DFG Research Training Group 2236

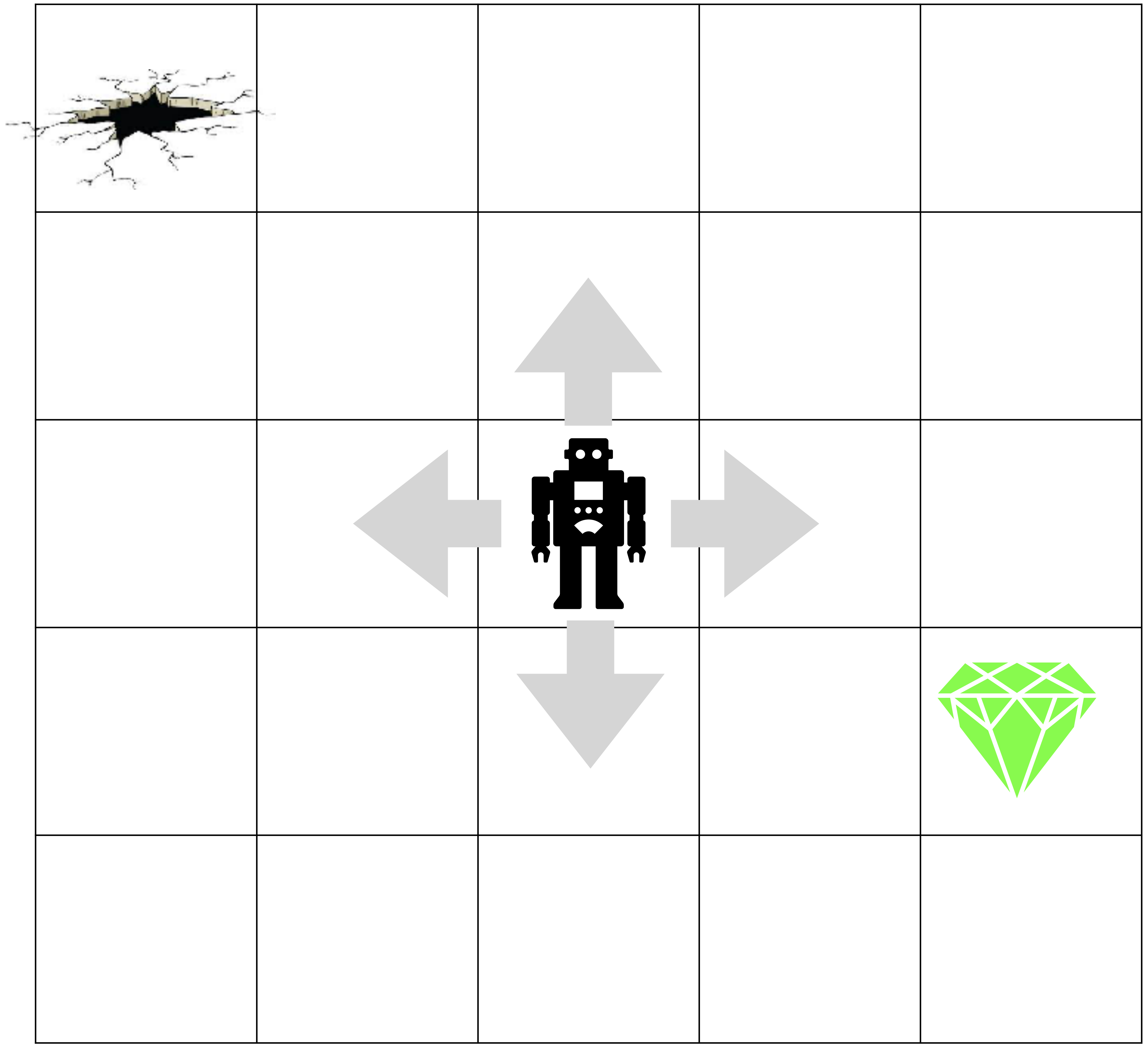
European Research Council

IMDEA Software Institute, Madrid — 19.09.2023

Language Inclusion

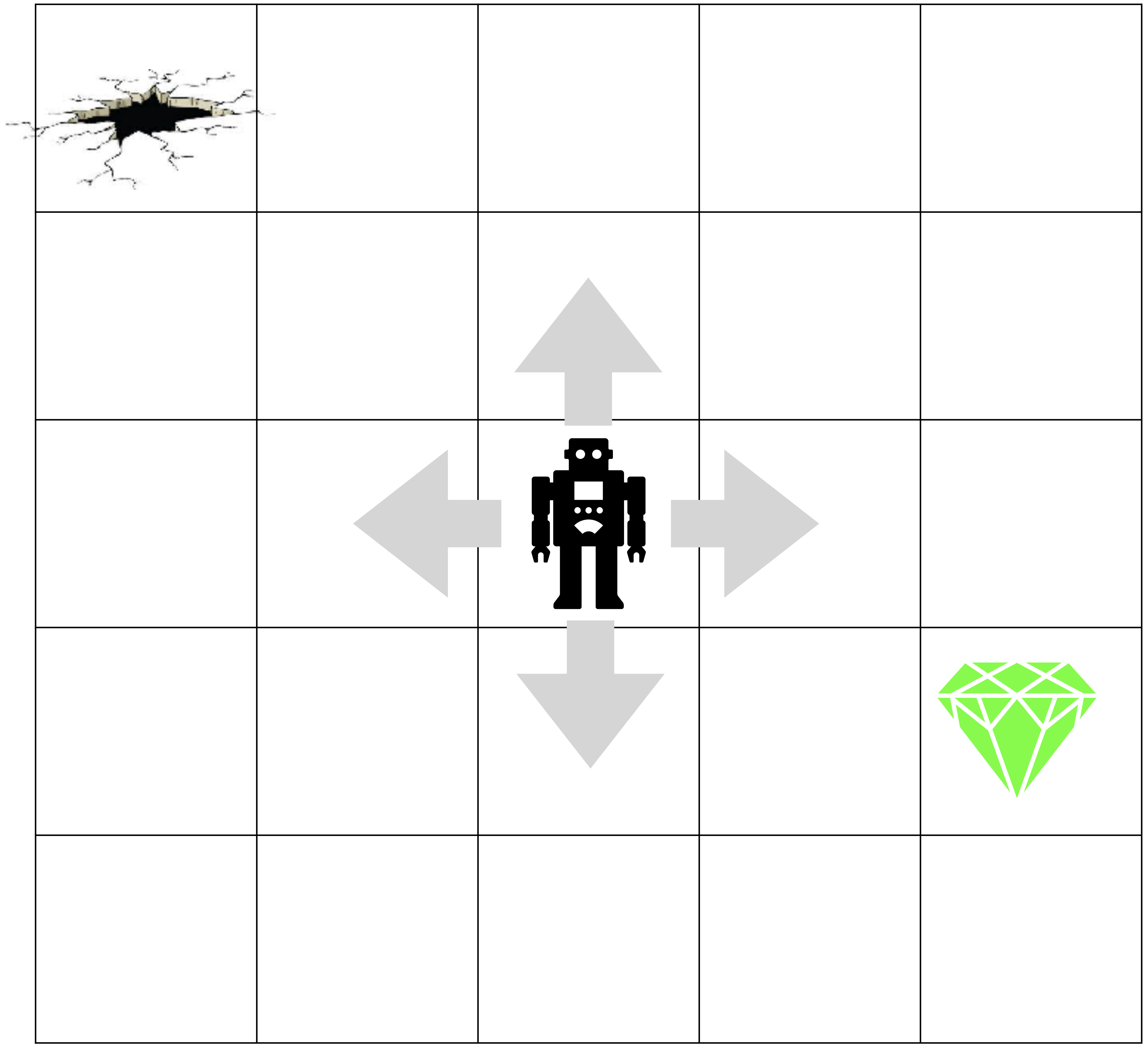
Is $L \subseteq M$ true?

Robot on a Grid

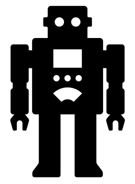


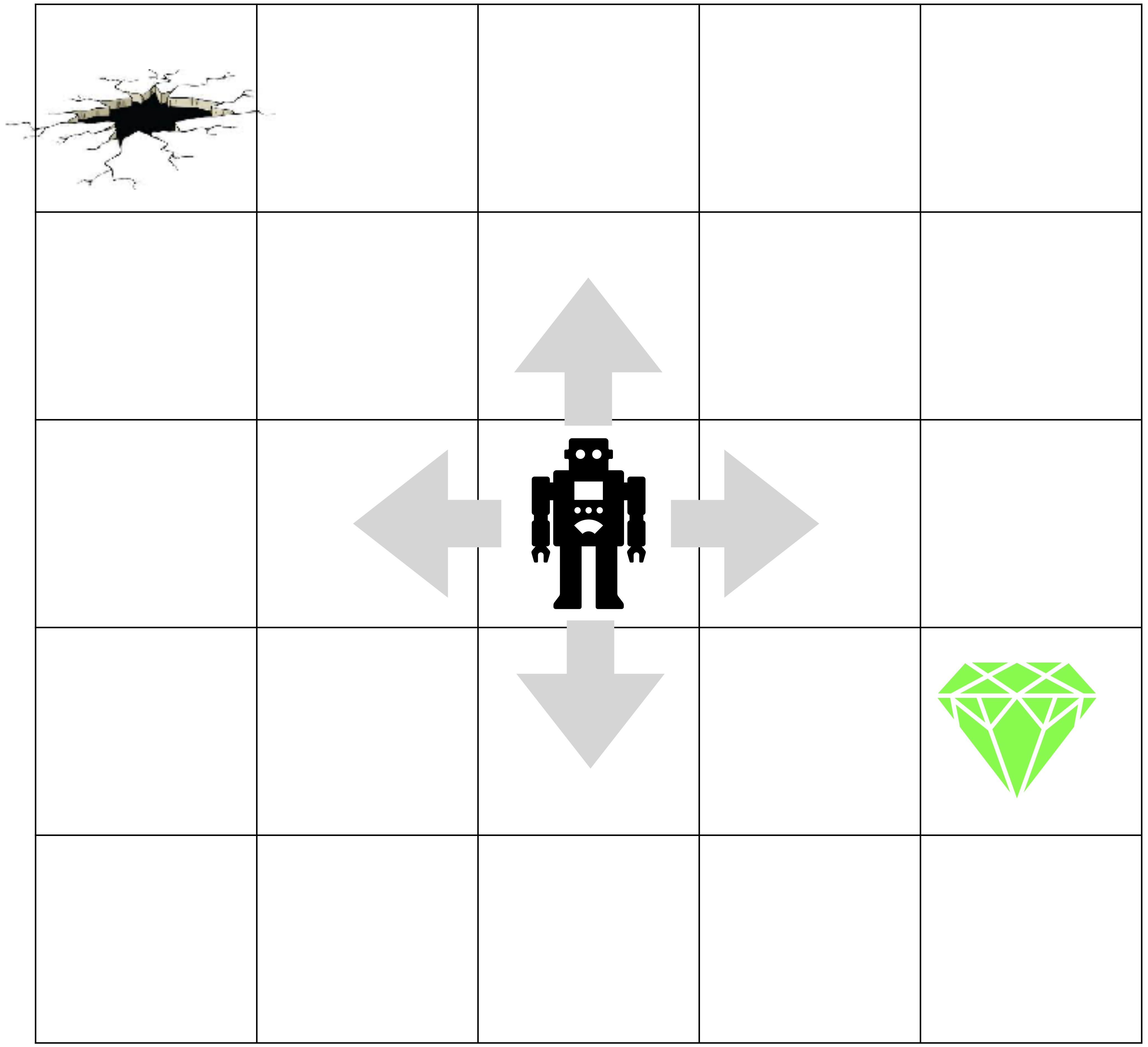
Robot on a Grid

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$

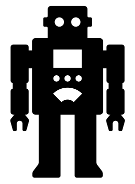


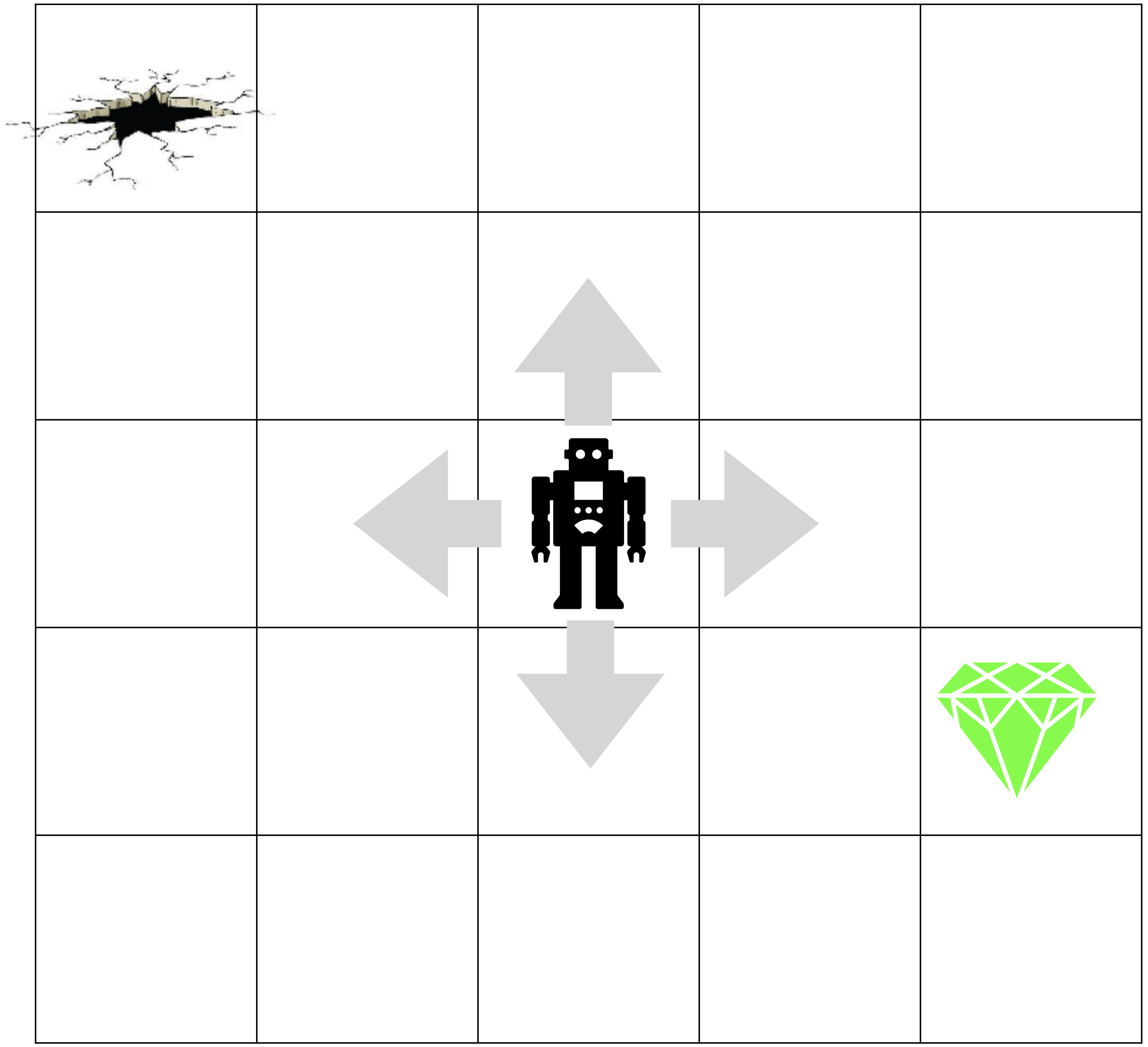
Robot on a Grid

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- L = all possible walks of  on the grid

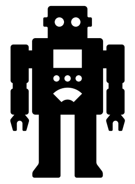


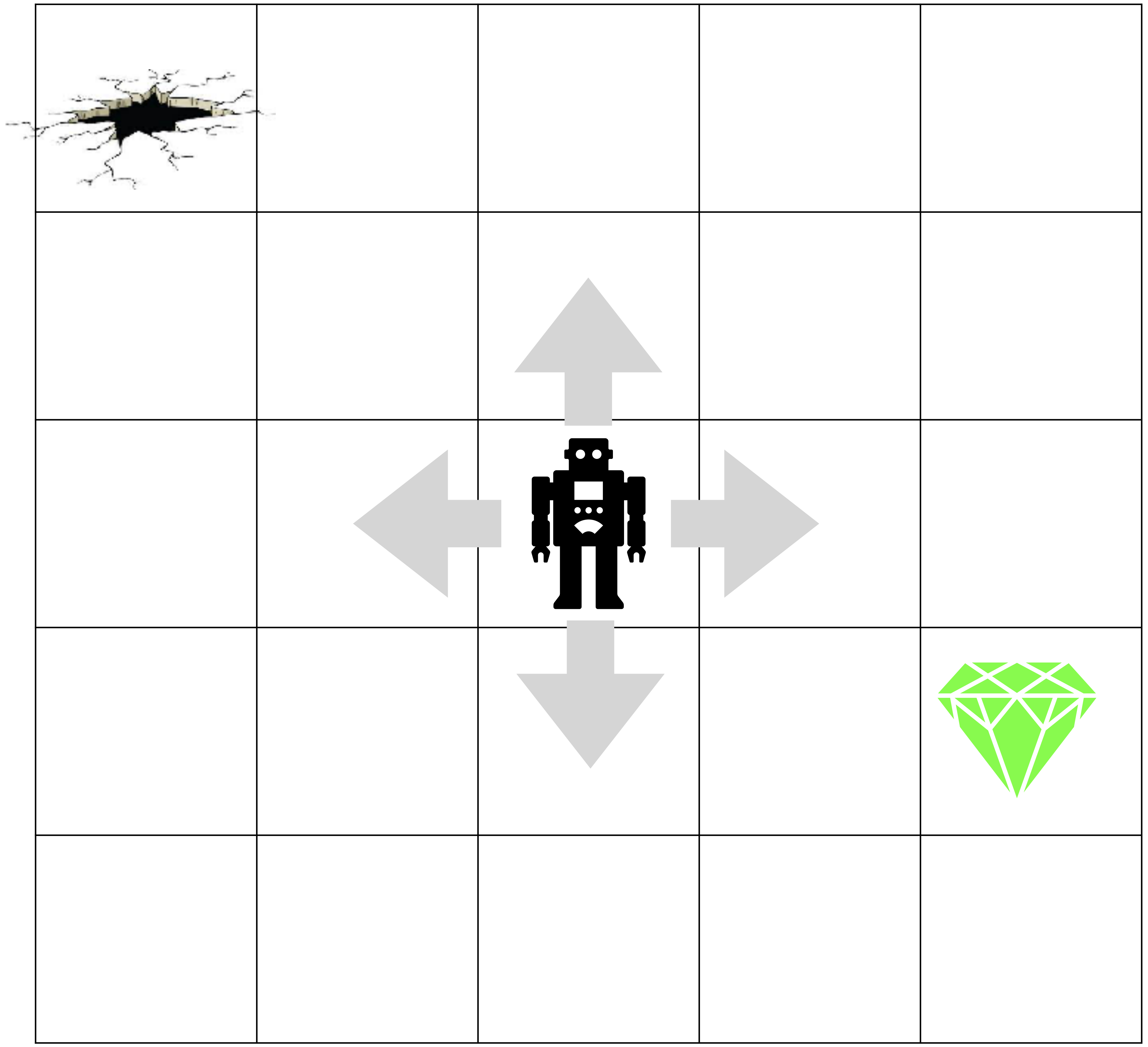
Robot on a Grid

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- L = all possible walks of "" on the grid
- M = " before "

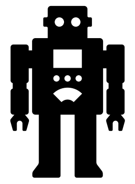


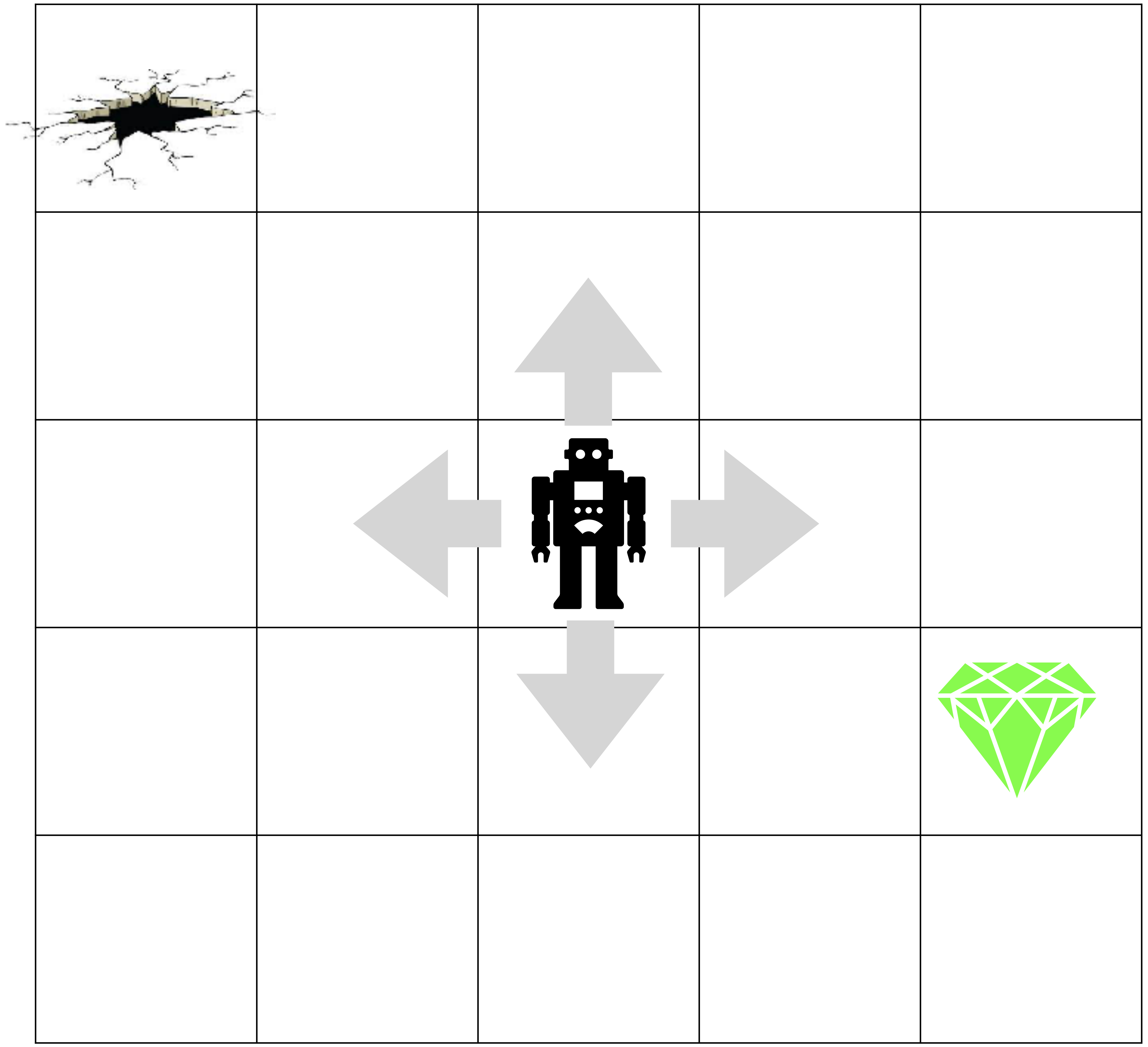
Robot on a Grid

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- $L =$ all possible walks of "" on the grid
- $M =$ " before "
- $L \subseteq M$ is false



Robot on a Grid

- $\Sigma = \{(x, y) \mid 0 \le x, y < 5\}$
- $L =$ all possible walks of  on the grid
- $M =$ "  before  "
- $L \subseteq M$ is false 



Probabilistic Language Inclusion

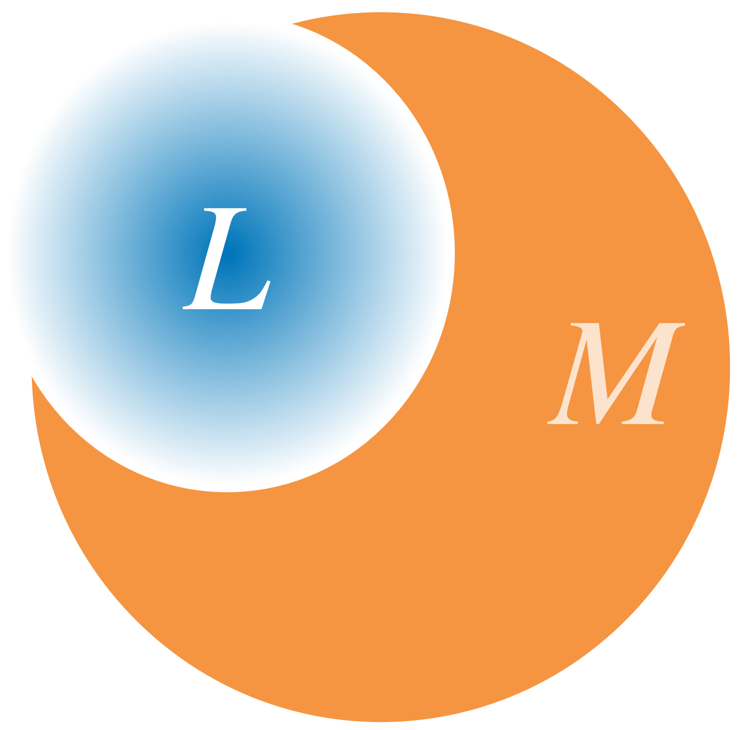
How true is $L \subseteq M$?

Probabilistic Language Inclusion

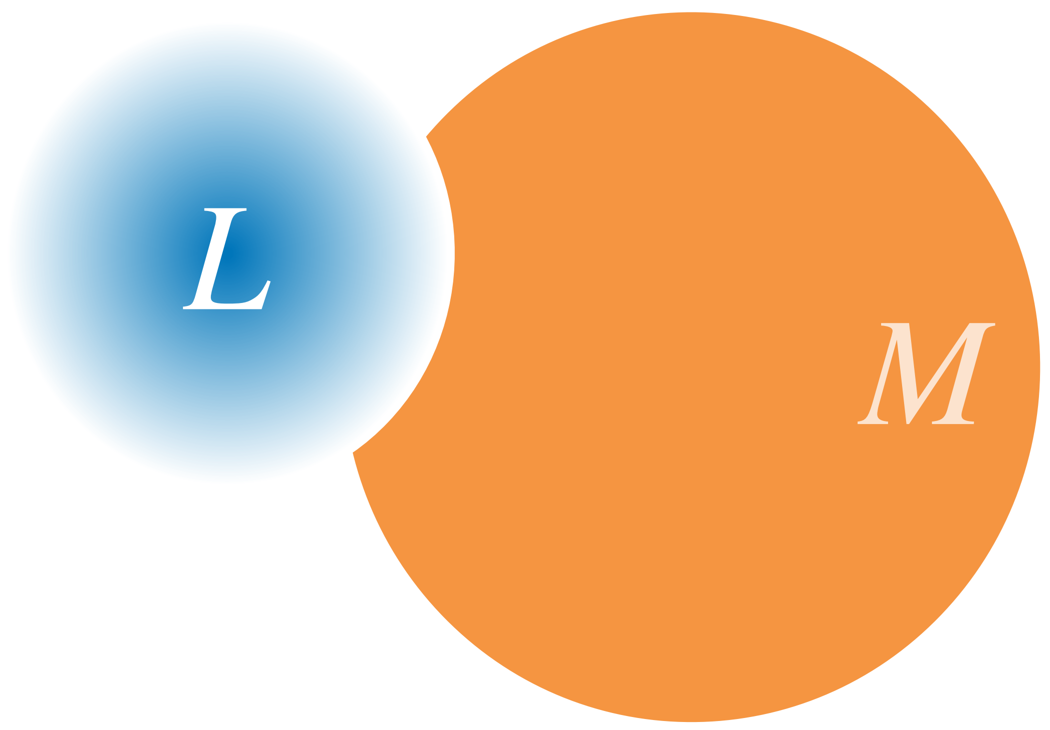
How true is $L \subseteq M$?

Probabilistic Language Inclusion

Problem Definition



0.95

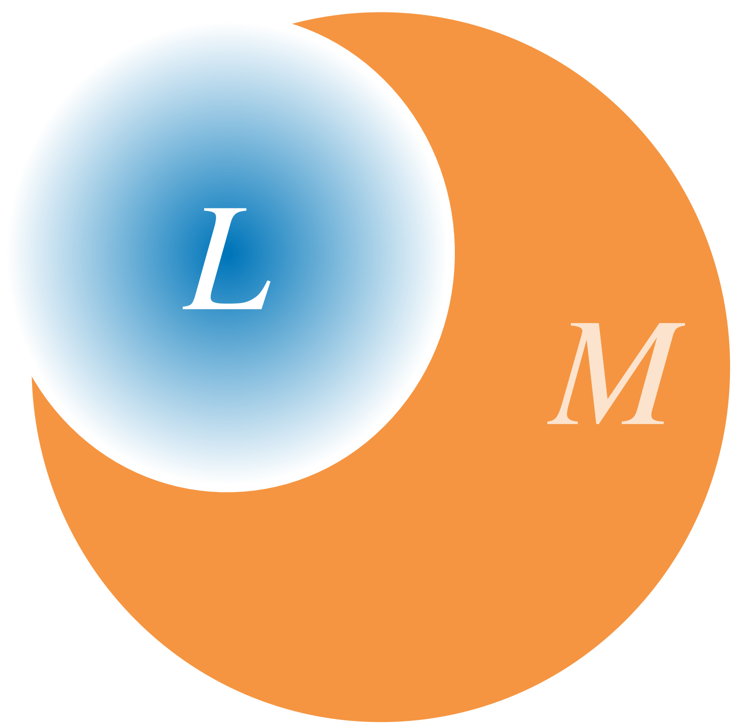


0.1

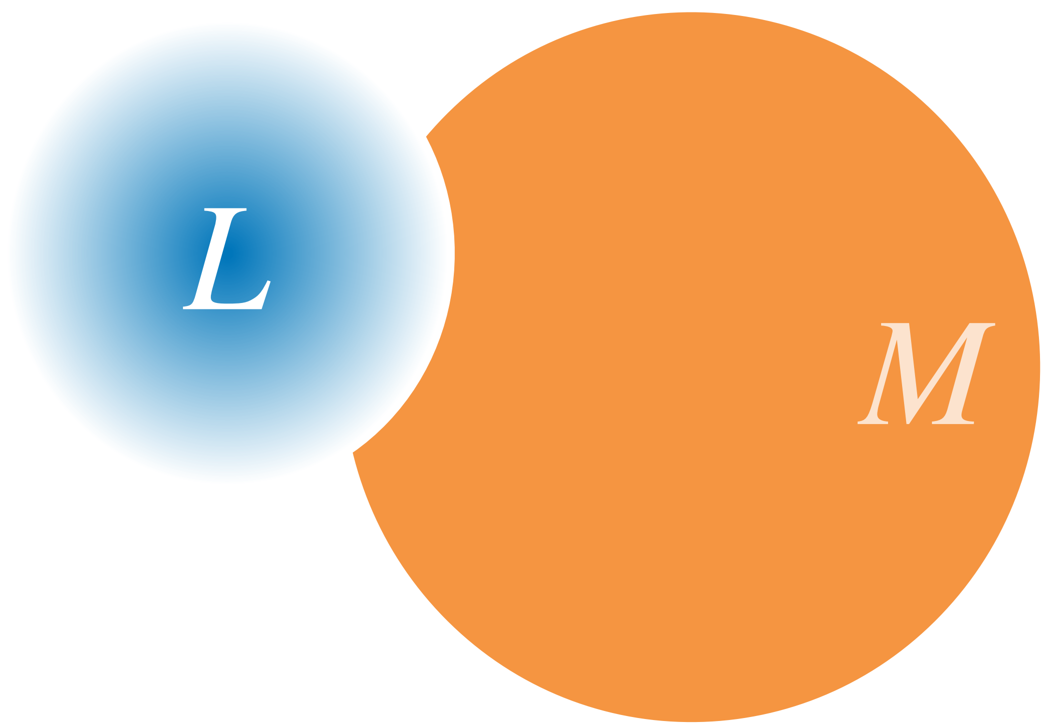
Probabilistic Language Inclusion

Problem Definition

- Given: probability measure L on Σ^ω, $M \subseteq \Sigma^\omega$ measurable



0.95

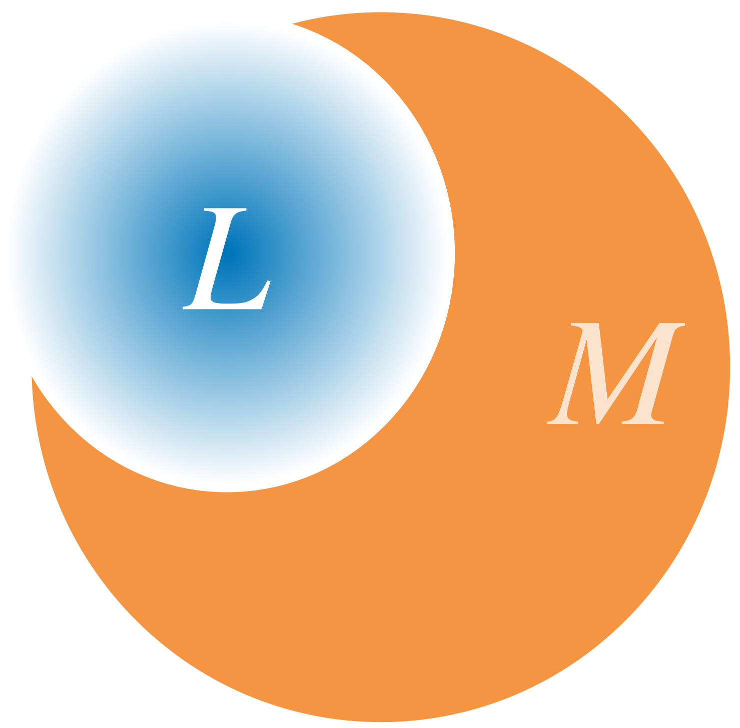


0.1

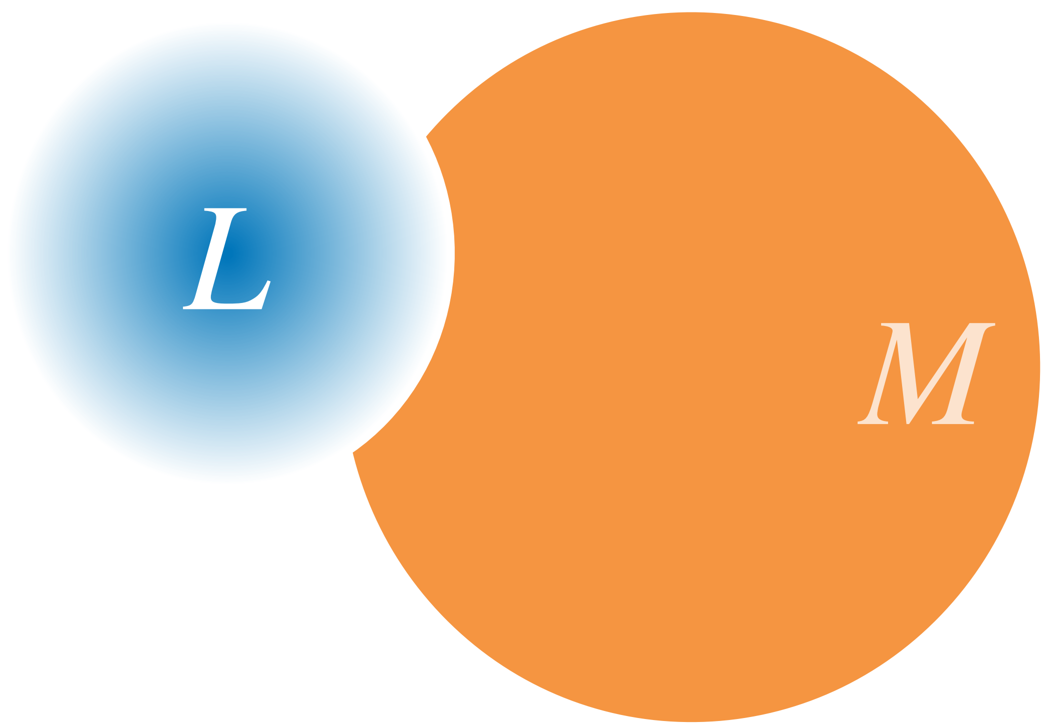
Probabilistic Language Inclusion

Problem Definition

- Given: probability measure L on Σ^ω, $M \subseteq \Sigma^\omega$ measurable
- Question: What is $Pr_{w \sim L}(w \in M) = L(M)$



0.95

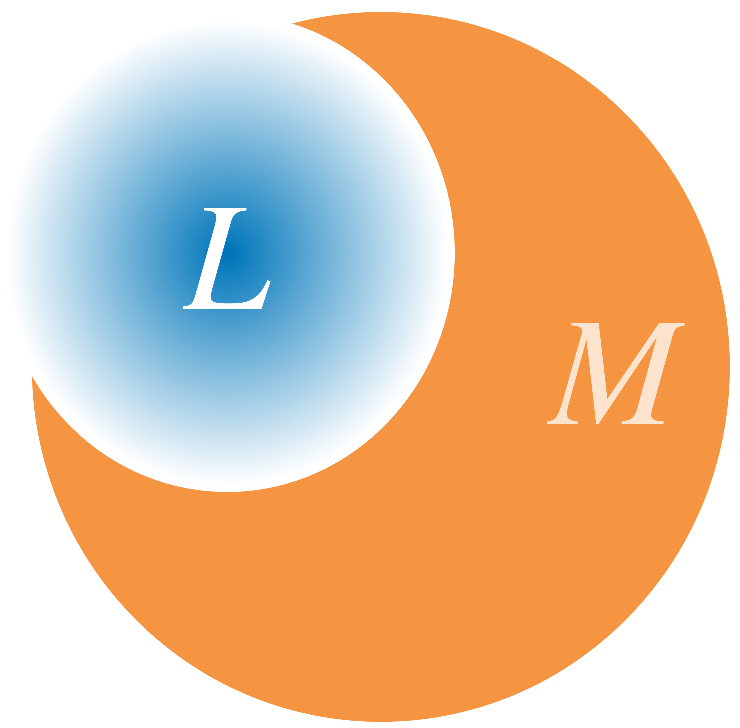


0.1

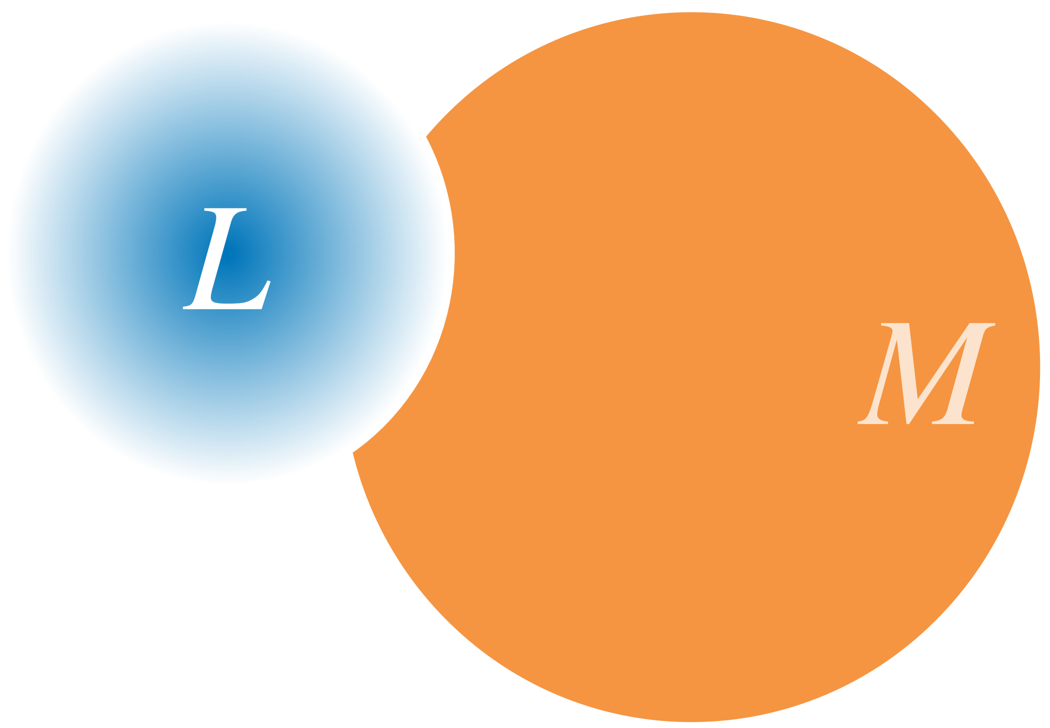
Probabilistic Language Inclusion

Problem Definition

- Given: probability measure L on Σ^ω, $M \subseteq \Sigma^\omega$ measurable
- Question: What is $Pr_{w \sim L}(w \in M) = L(M)$
- Variant: Is $Pr_{w \sim L}(w \in M) = 1$? "almost-sure inclusion"

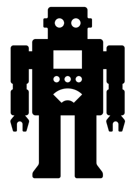


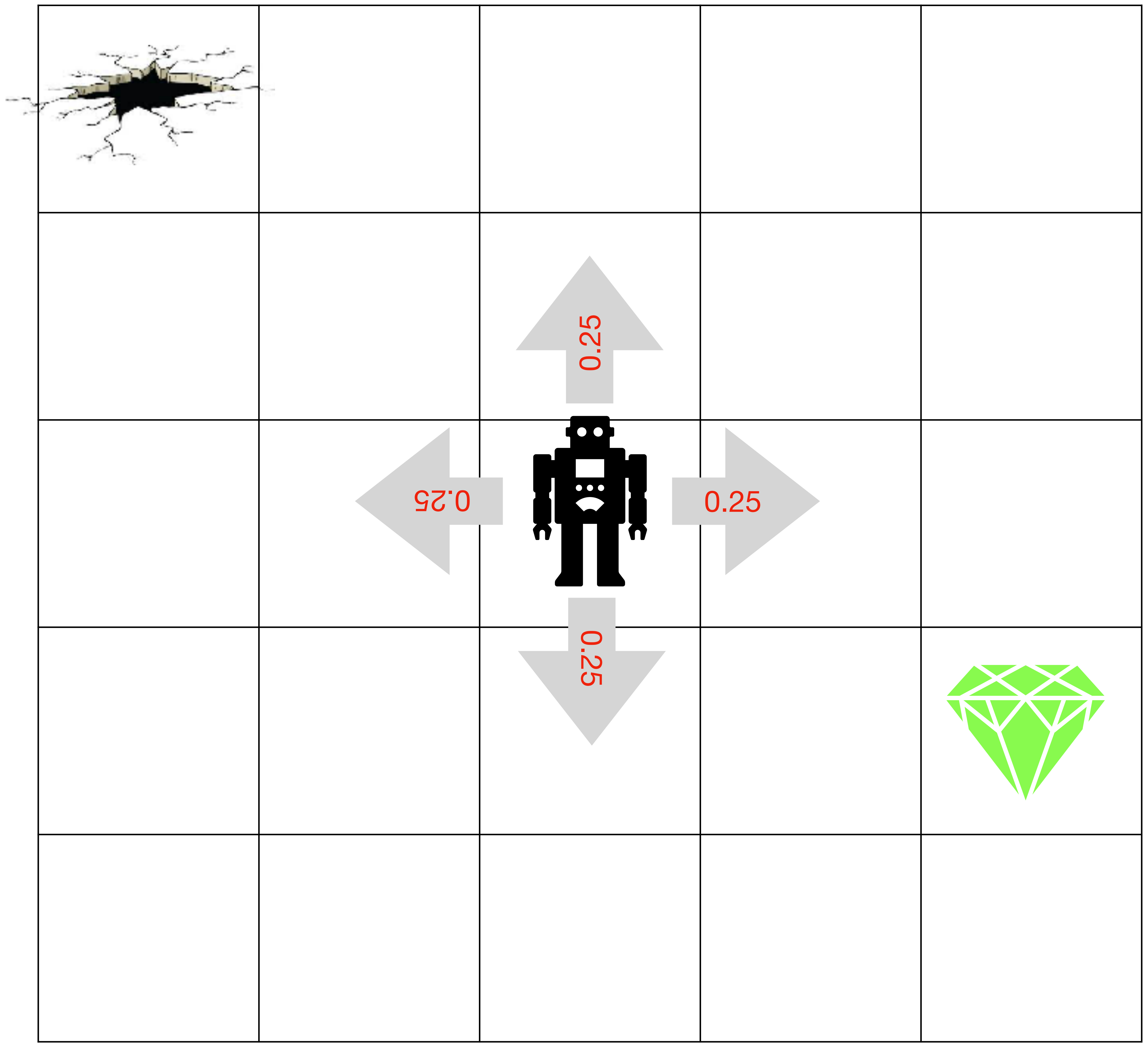
0.95



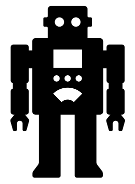
0.1

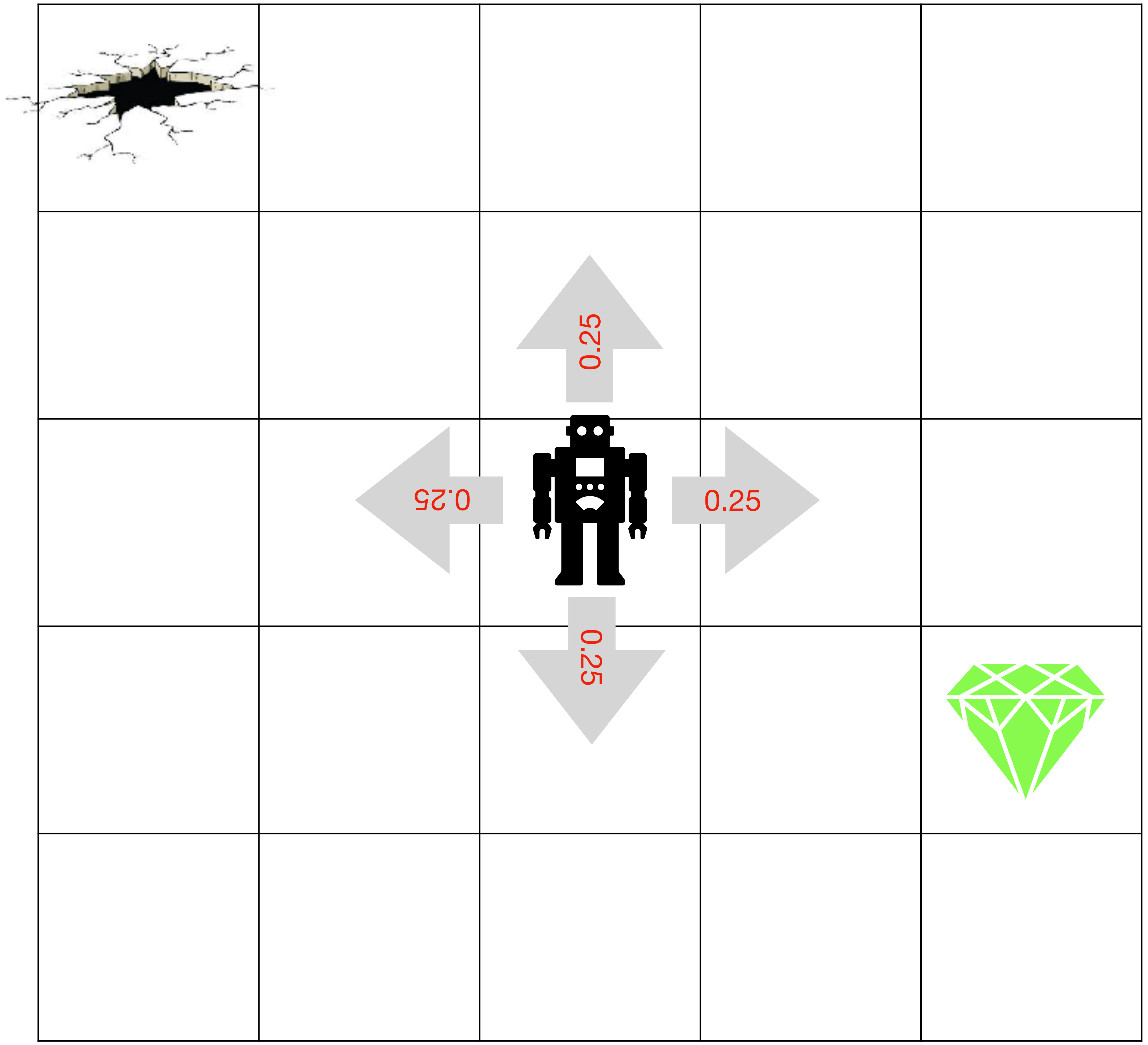
Robot on a Grid (probabilistic)

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- $L =$ random walks of  on the grid
- $M =$ "  before  "



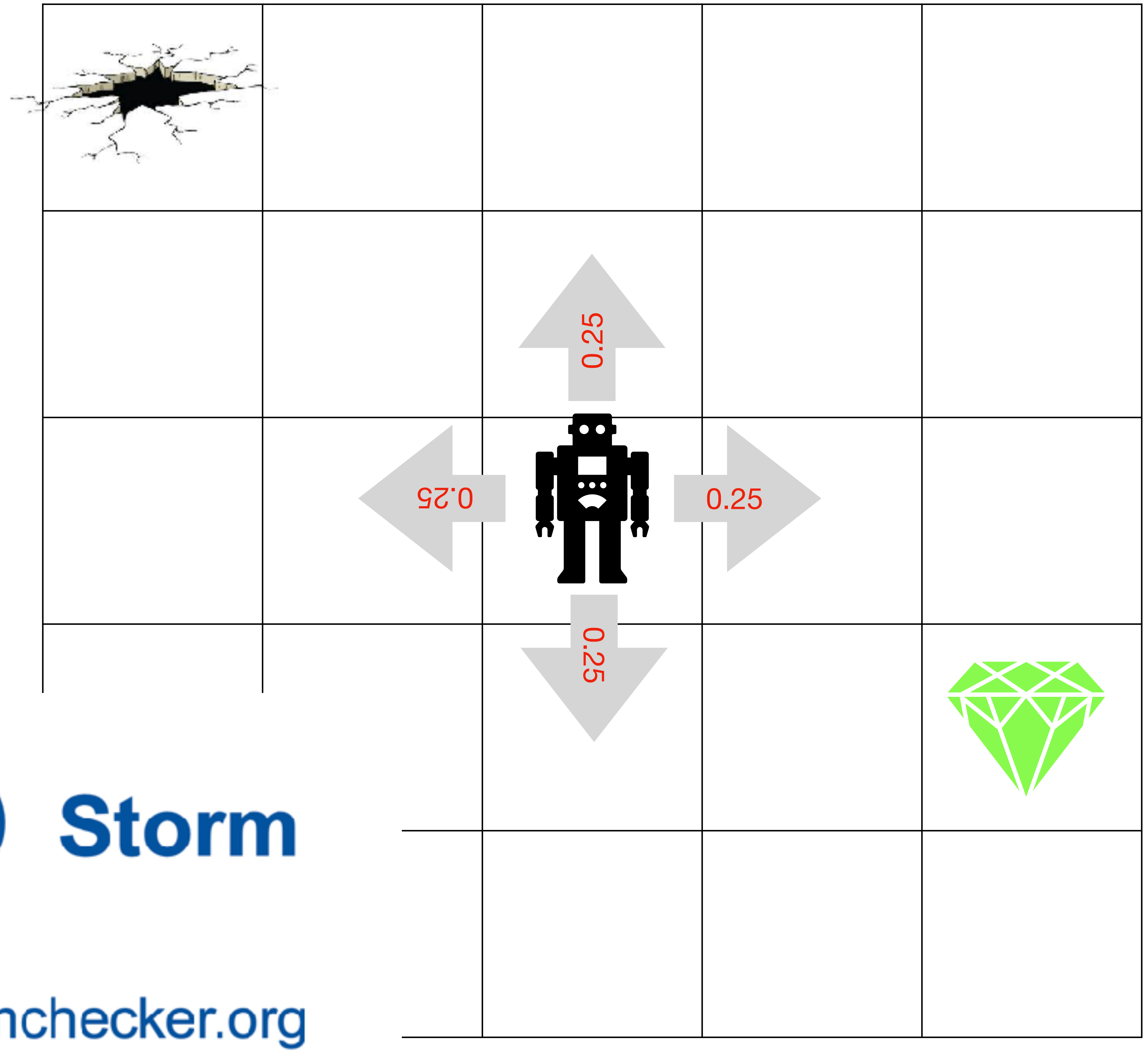
Robot on a Grid (probabilistic)

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- L = random walks of  on the grid
- M = " before "
- $Pr_{w \sim L}(w \in M) = \frac{7050}{12113} \approx 0.58$



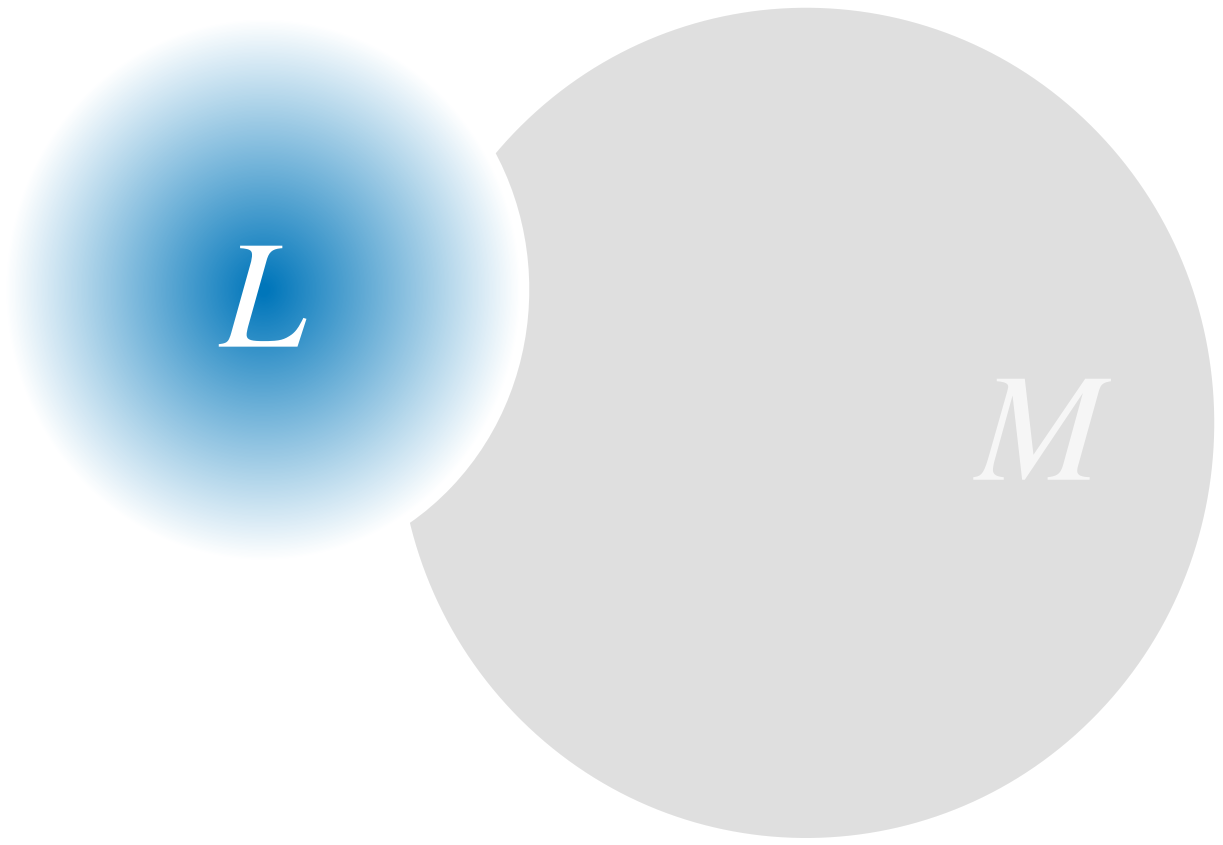
Robot on a Grid (probabilistic)

- $\Sigma = \{(x, y) \mid 0 \leq x, y < 5\}$
- L = random walks of on the grid
- M = " before "
- $Pr_{w \sim L}(w \in M) = \frac{7050}{12113} \approx 0.58$

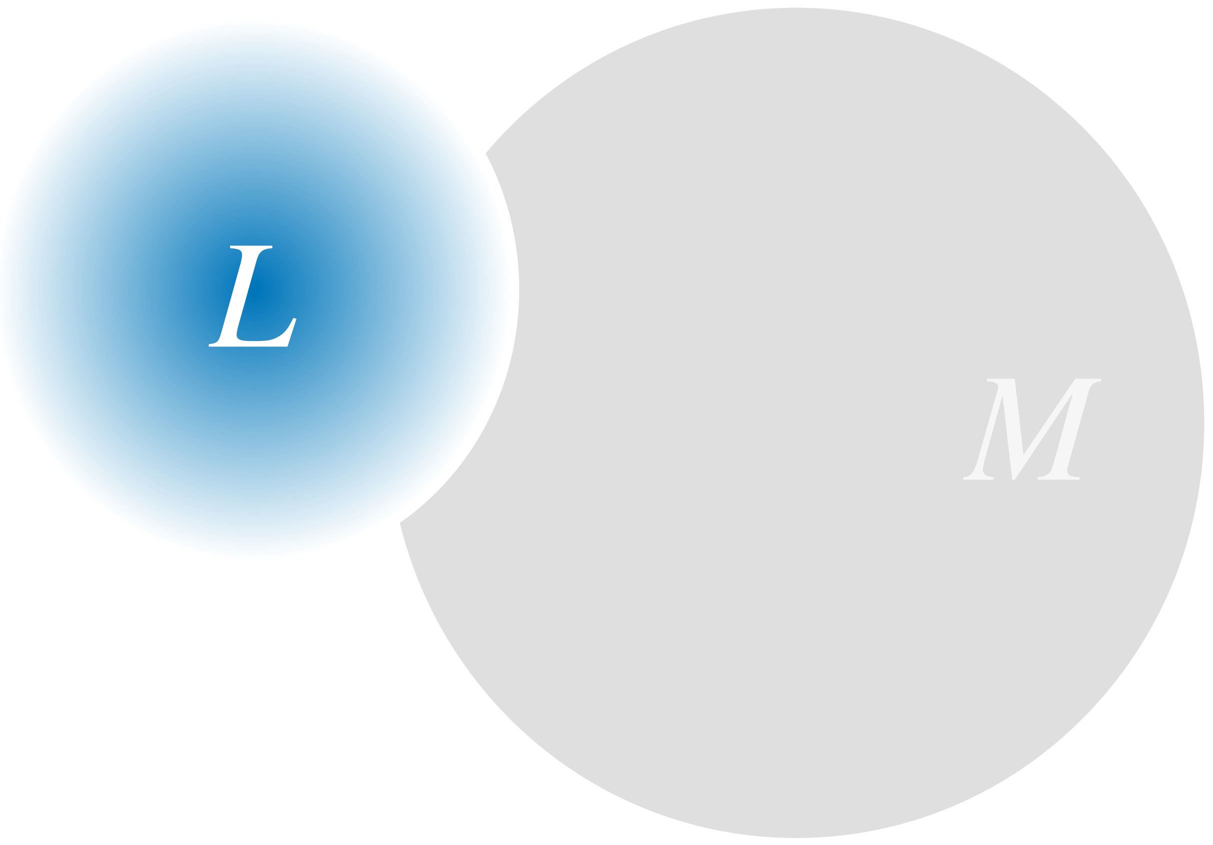


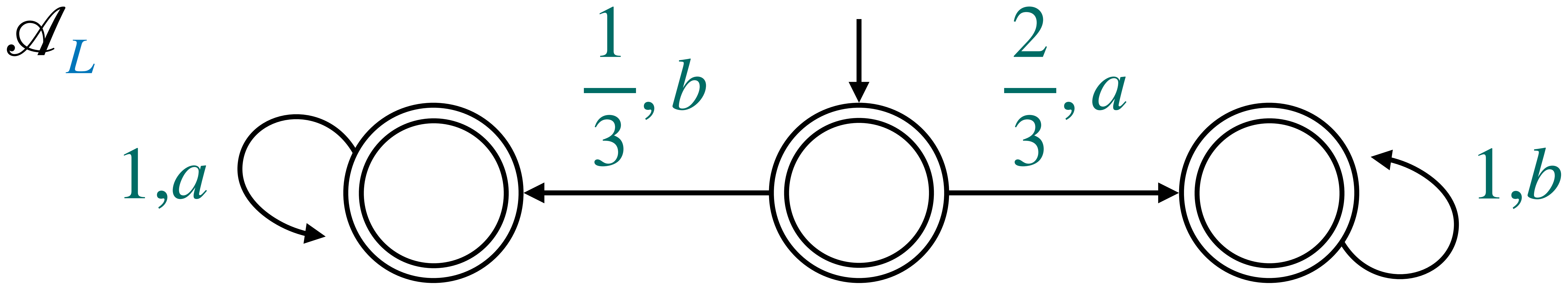
Storm
www.stormchecker.org

How to define L?

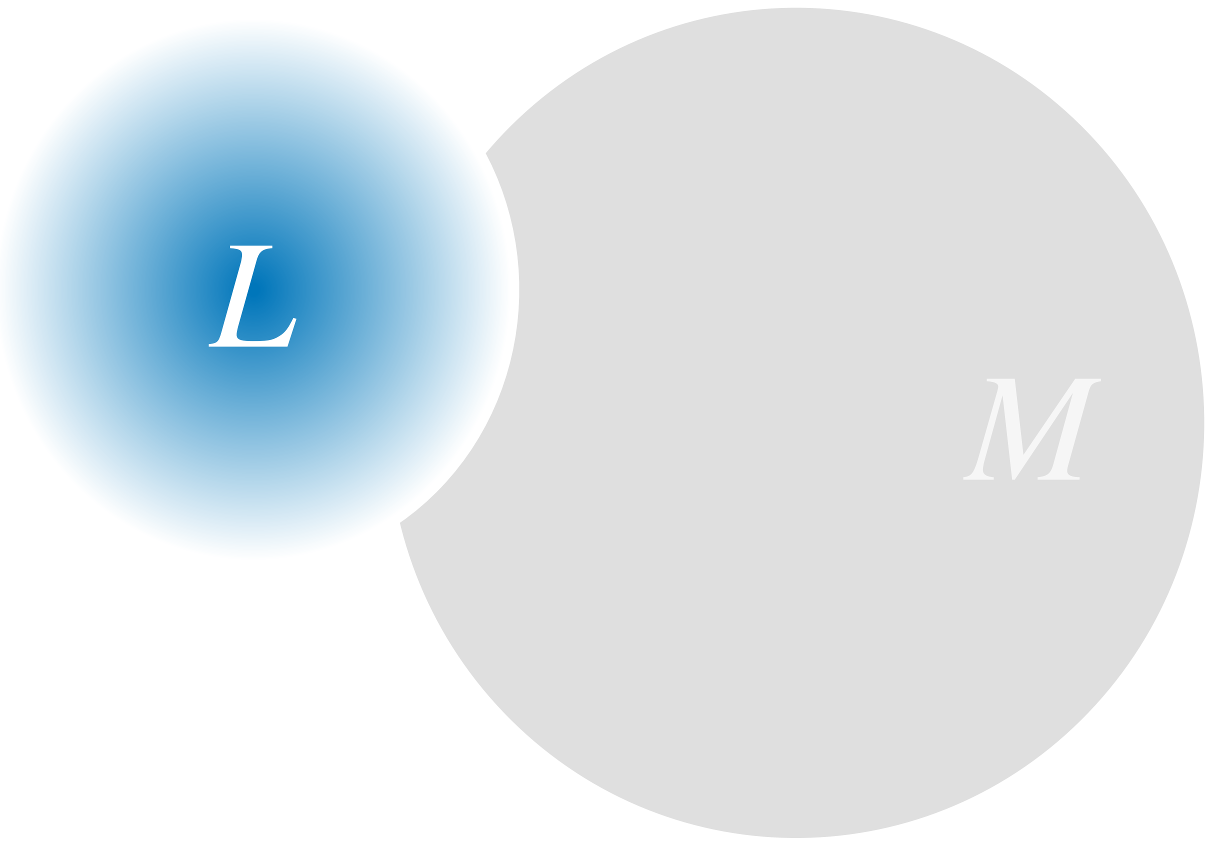


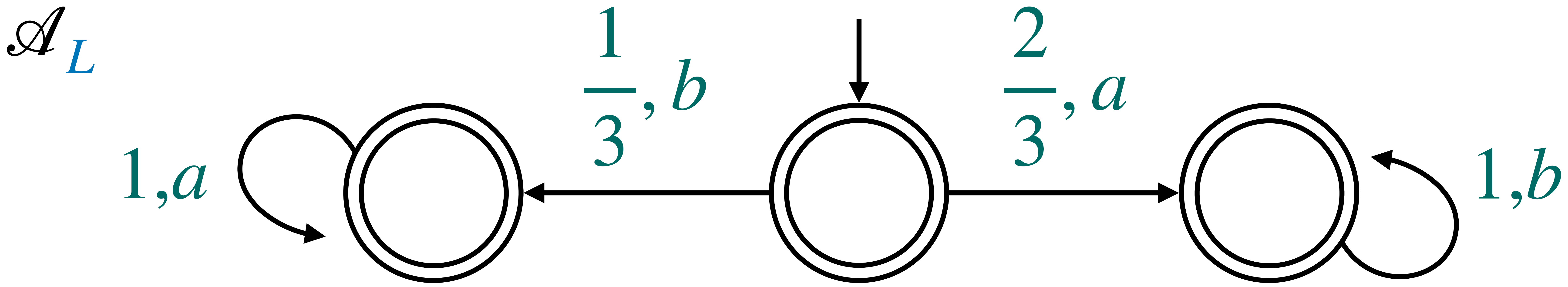
How to define L?





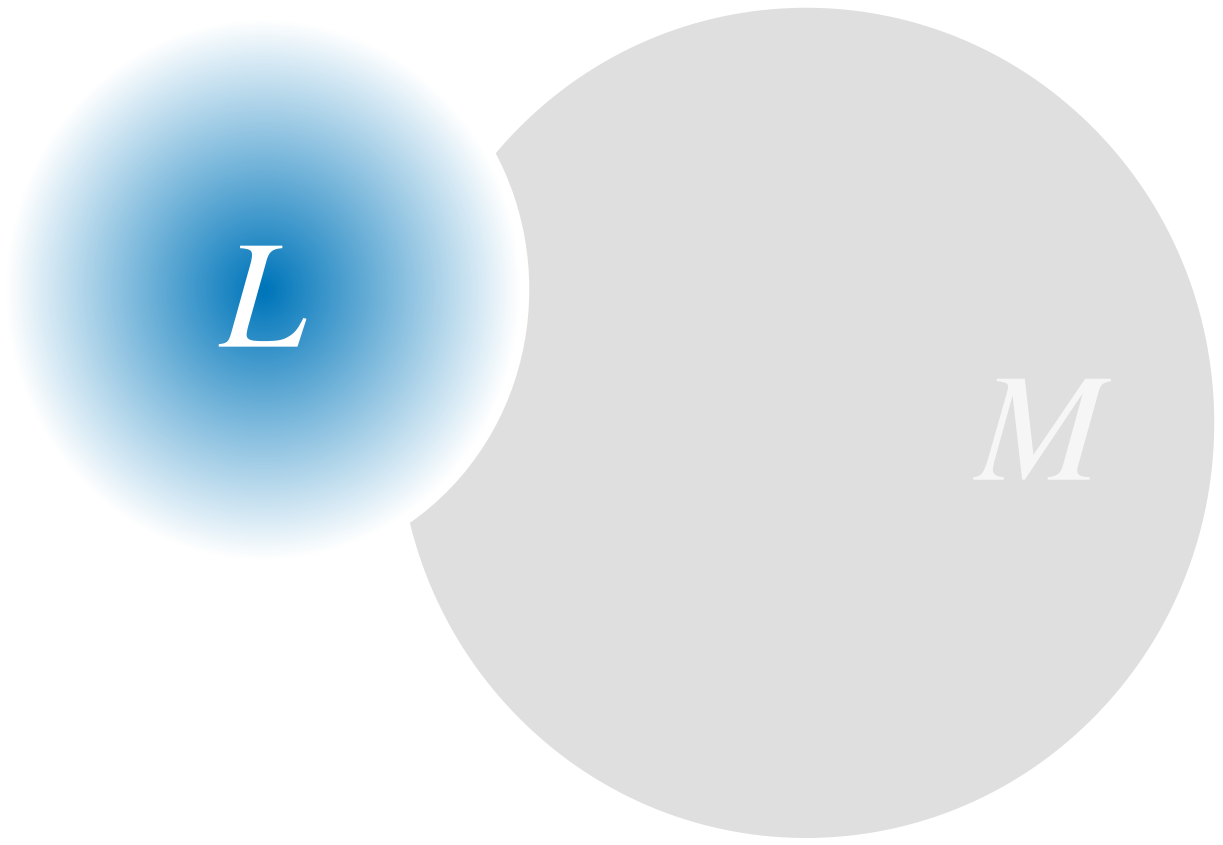
How to define L?

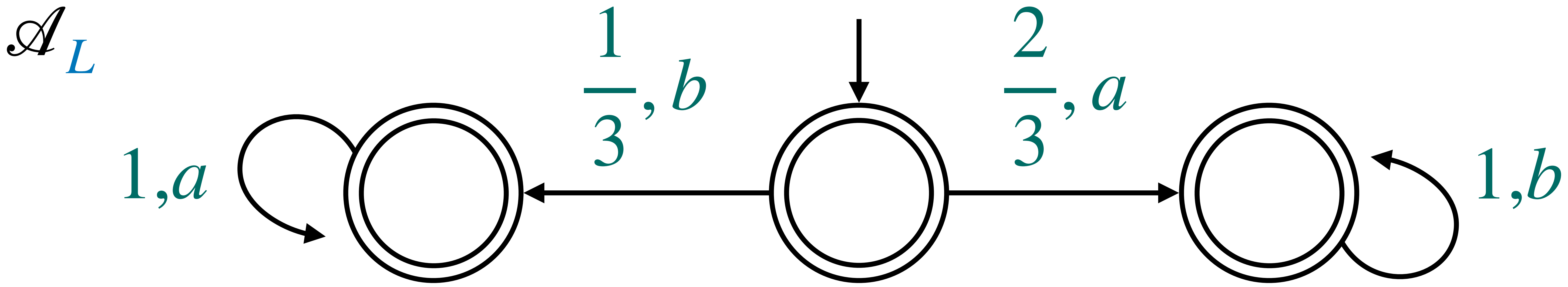




- $L = \{\frac{1}{3} : ba^\omega, \frac{2}{3} : ab^\omega\}$ (discrete, finite support)

How to define L?

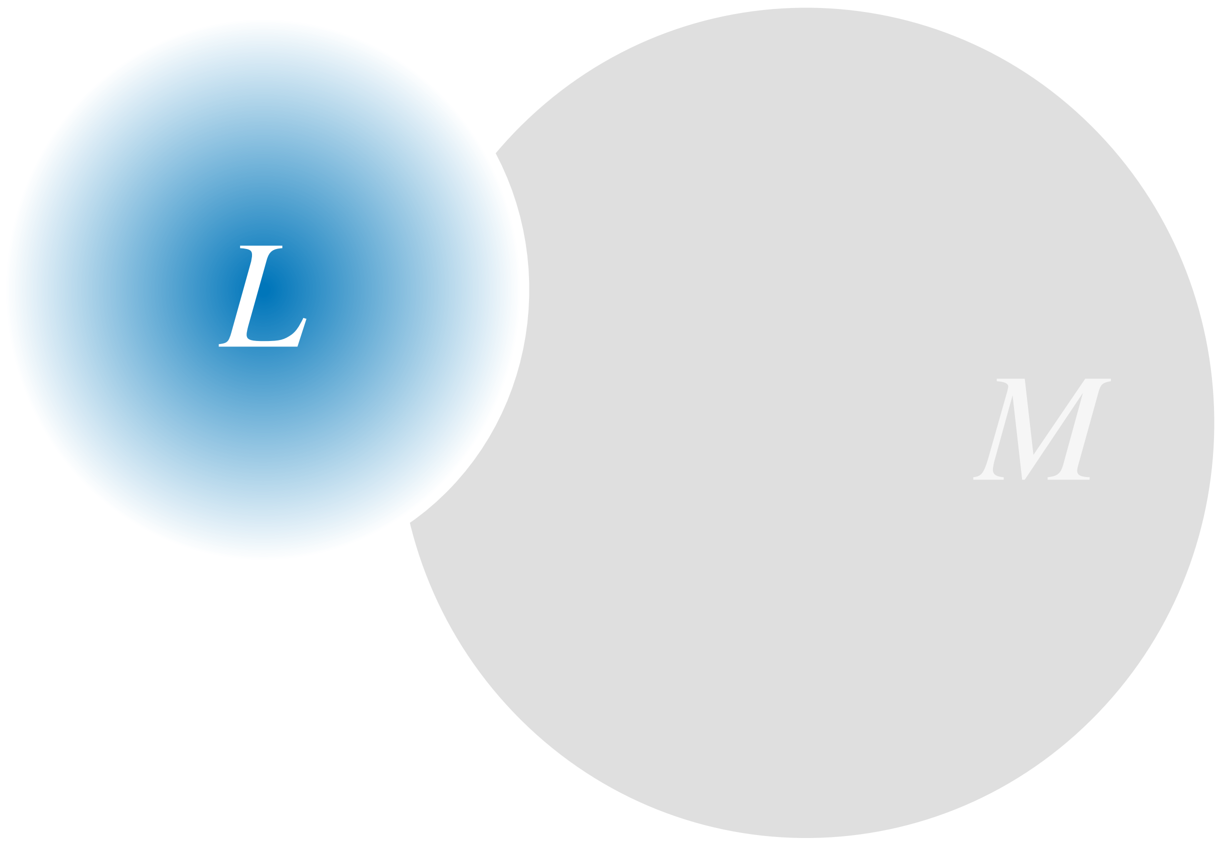


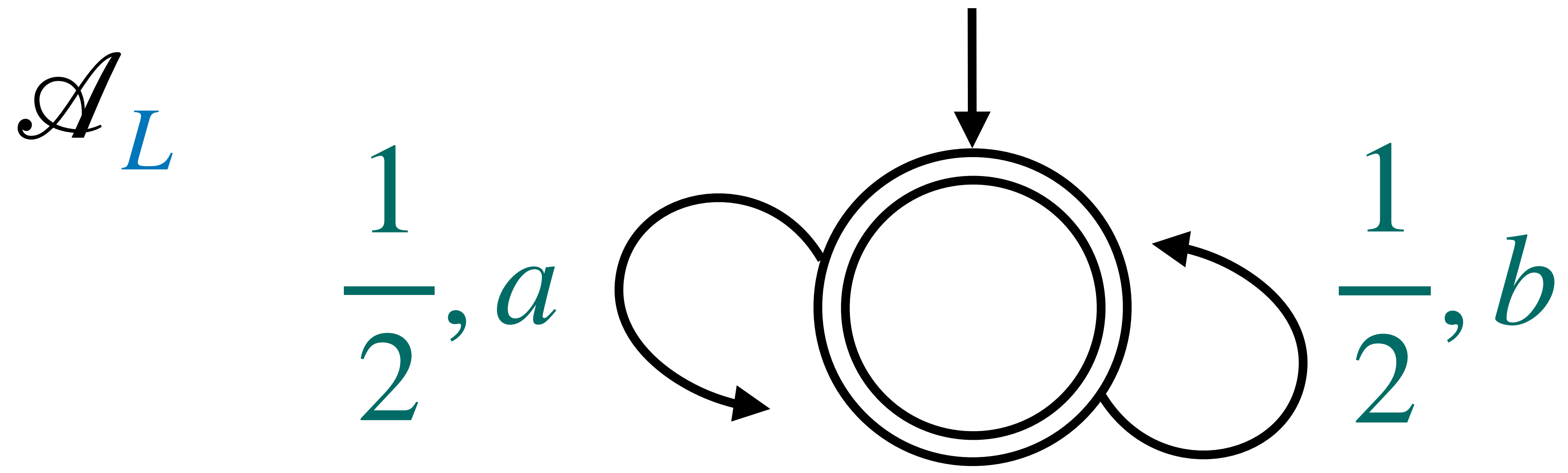


- $L = \{\frac{1}{3}: ba^\omega, \frac{2}{3}: ab^\omega\}$ (discrete, finite support)

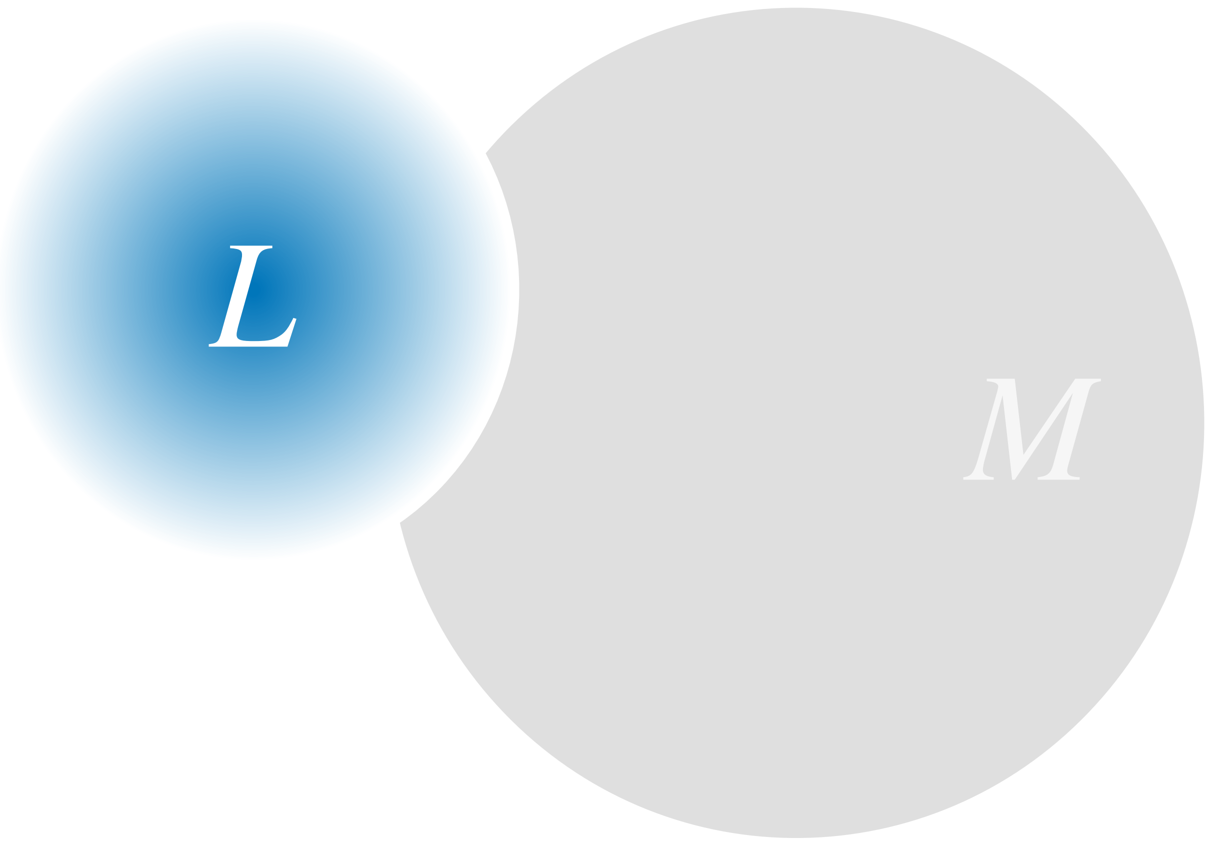
- $\mathscr{A}_L$ = "Generative" probabilistic automaton with trivial Büchi acceptance
  = finite Markov chain with labeled transitions

How to define L?

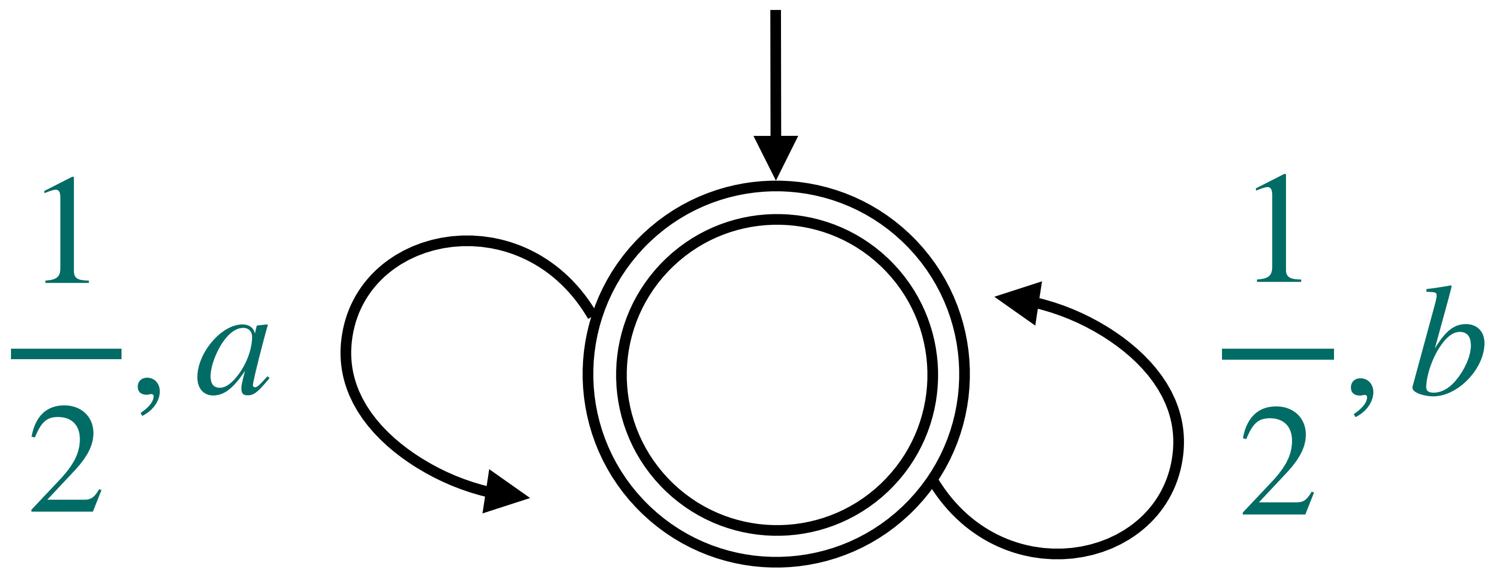




How to define L?

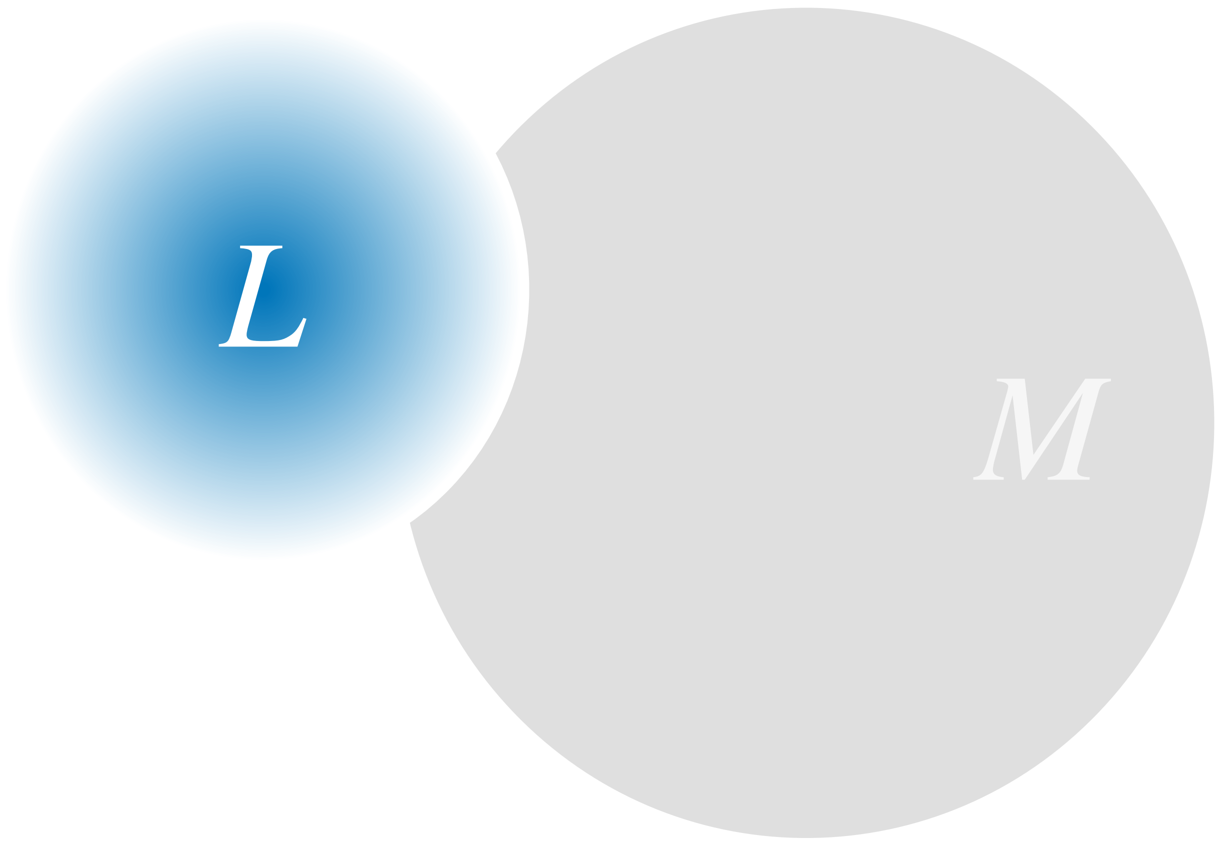


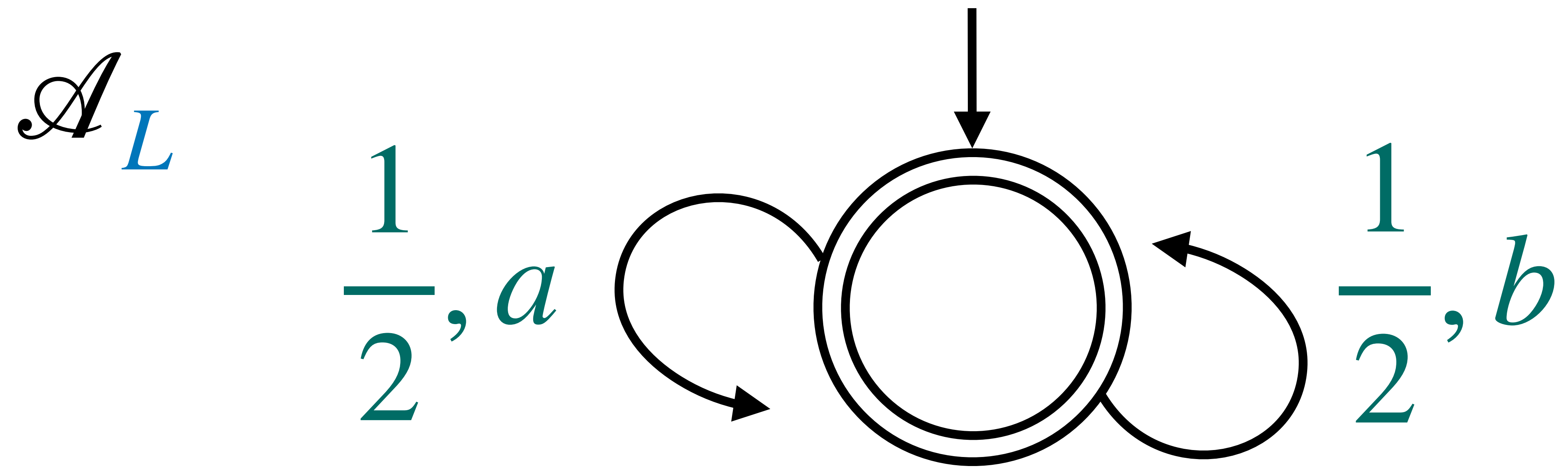
$\mathscr{A}_L$



- L = uniform probability measure on $\{a, b\}^\omega$ (continuous!)

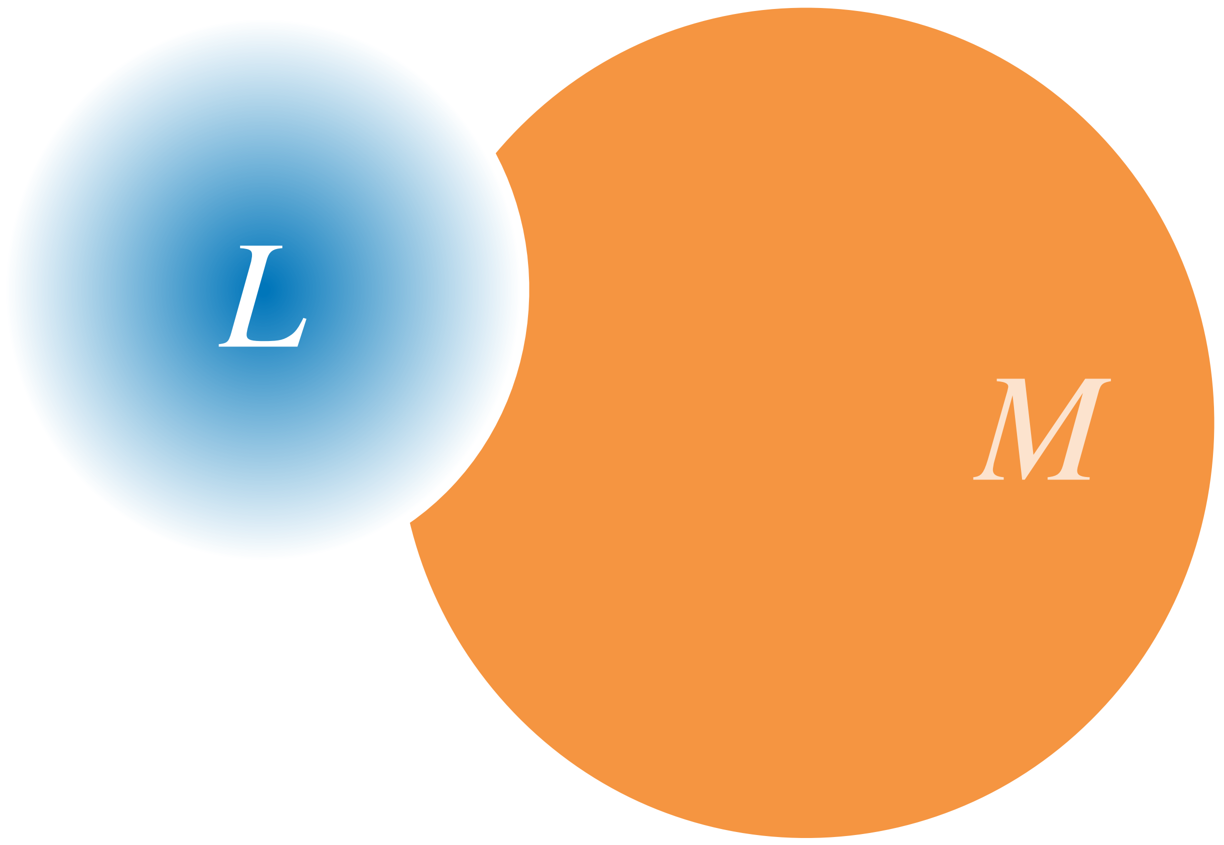
How to define L?

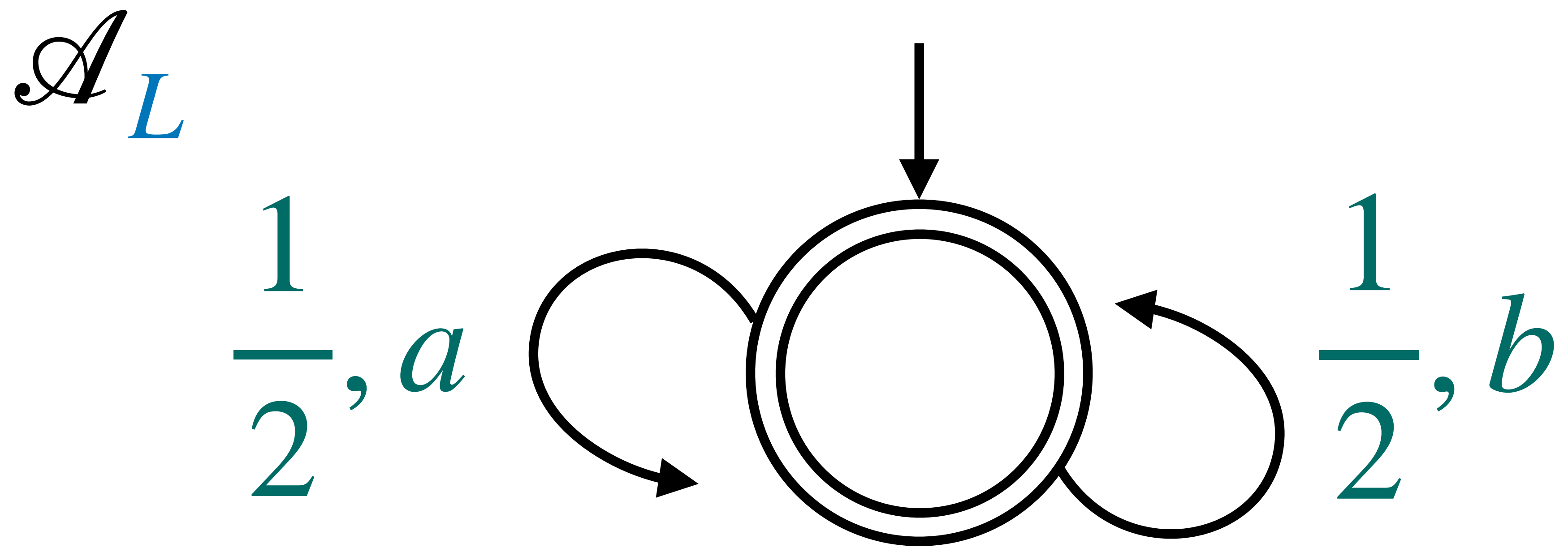


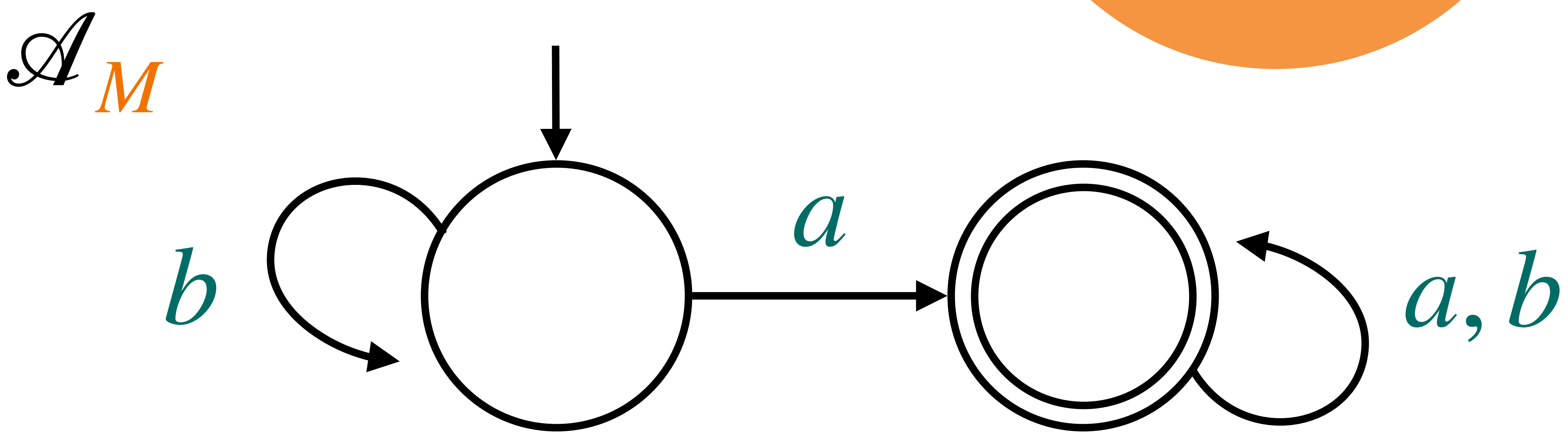


- L = uniform probability measure on $\{a, b\}^\omega$ (continuous!)
- Each individual word $w \in \{a, b\}^\omega$ occurs with probability 0

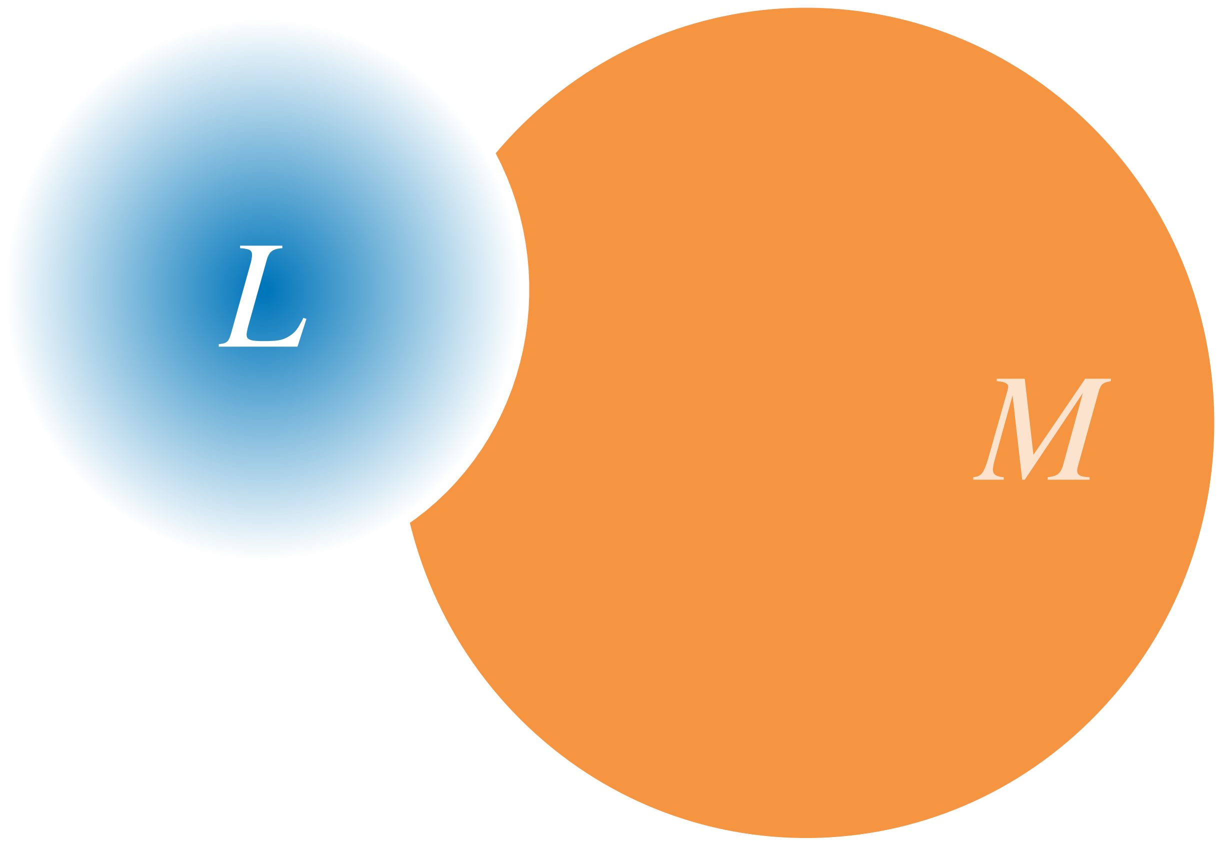
Example

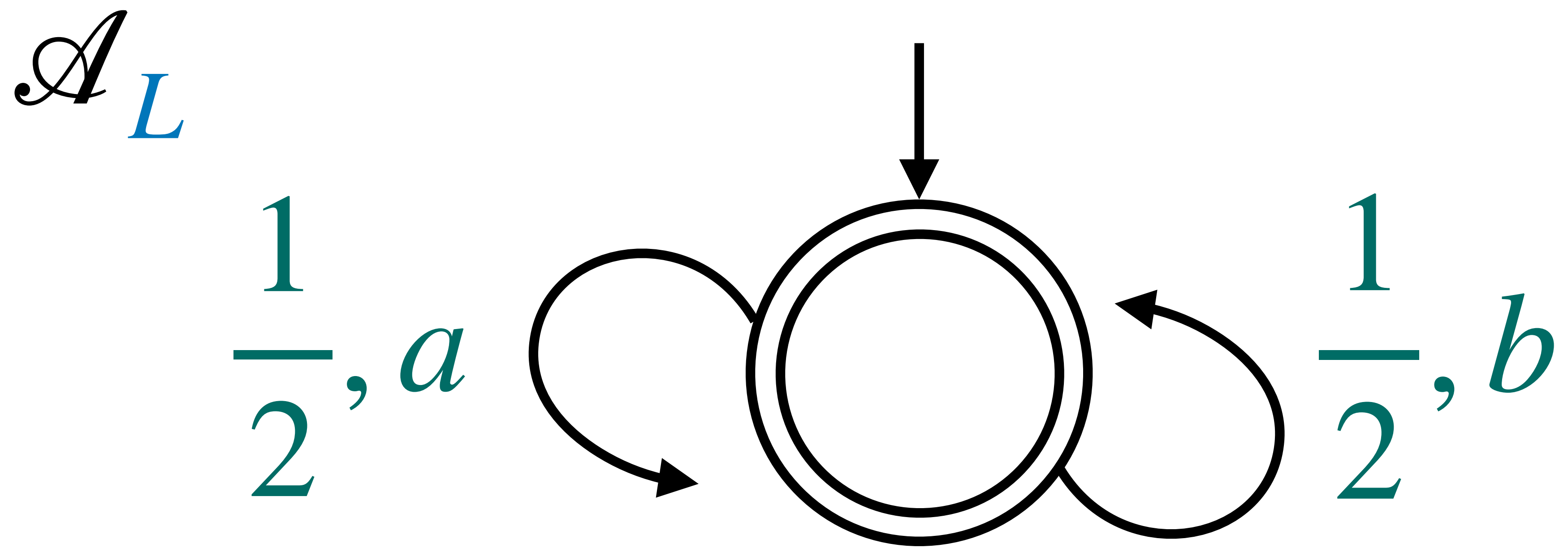


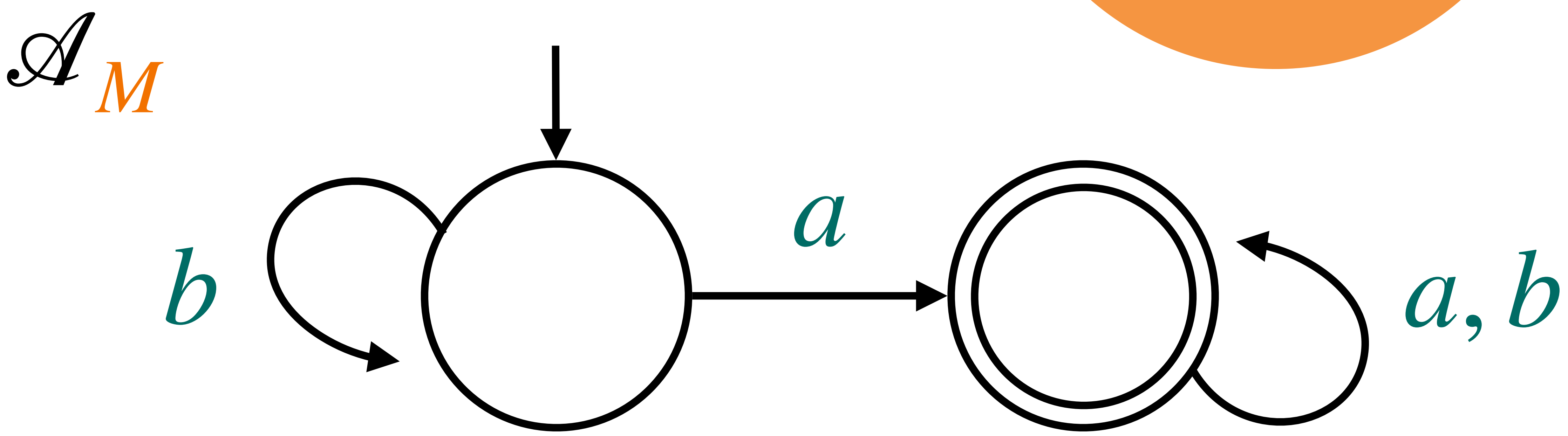




Example

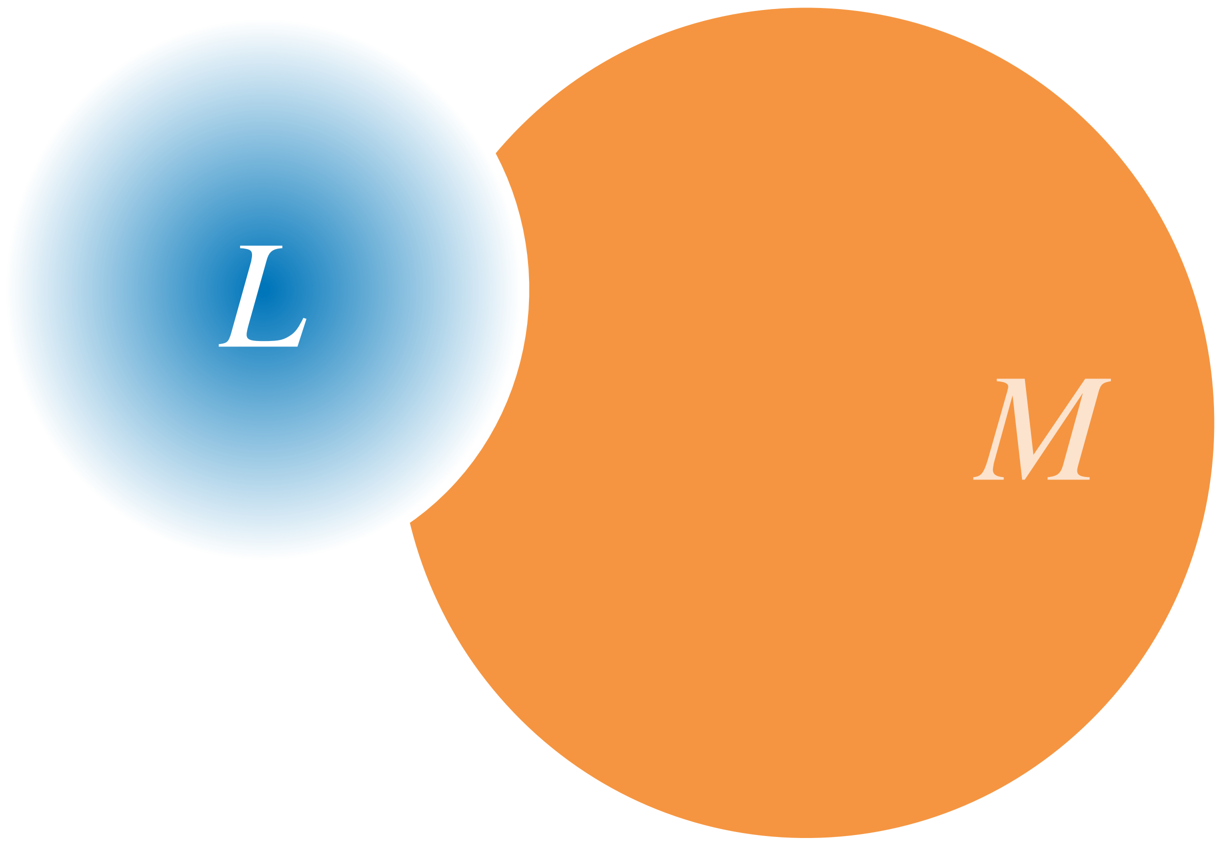




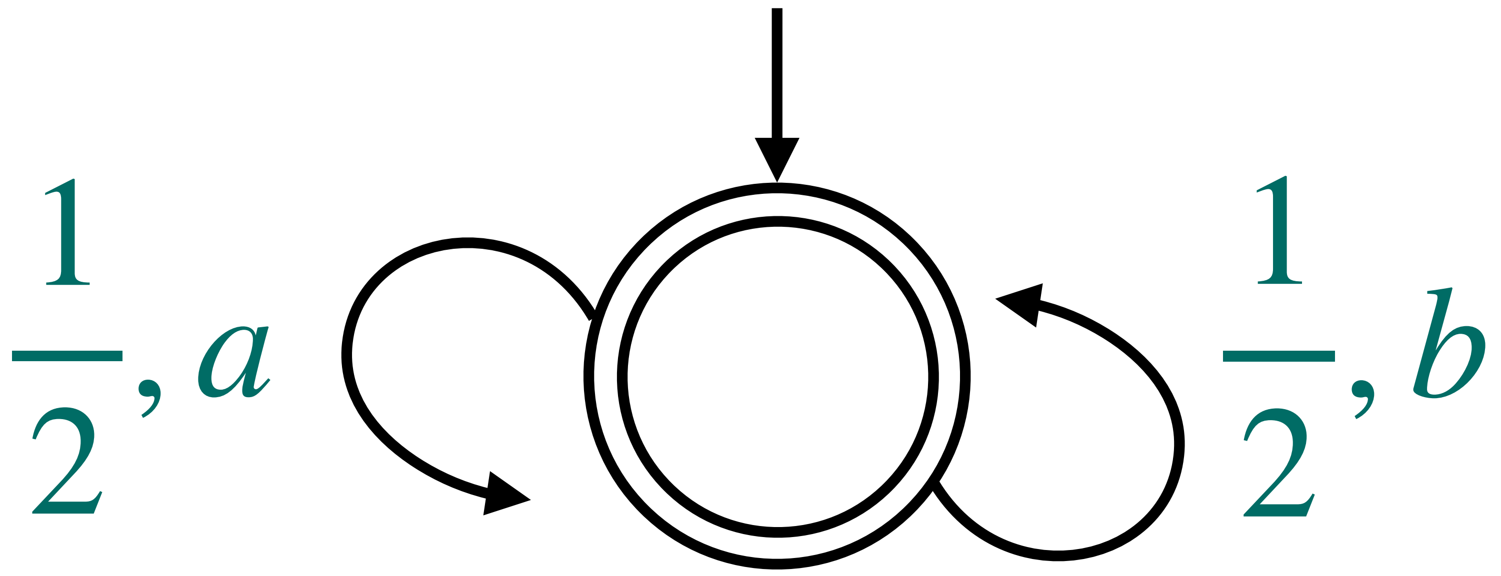


- L = uniform probability measure on $\{a, b\}^\omega$ (continuous)

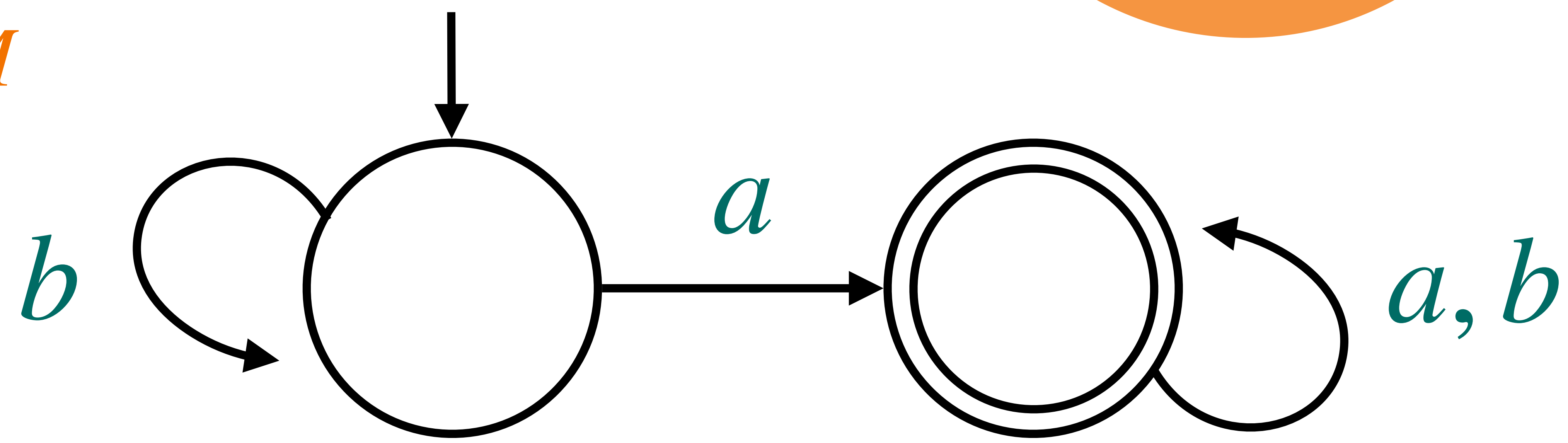
Example



$\mathscr{A}_L$

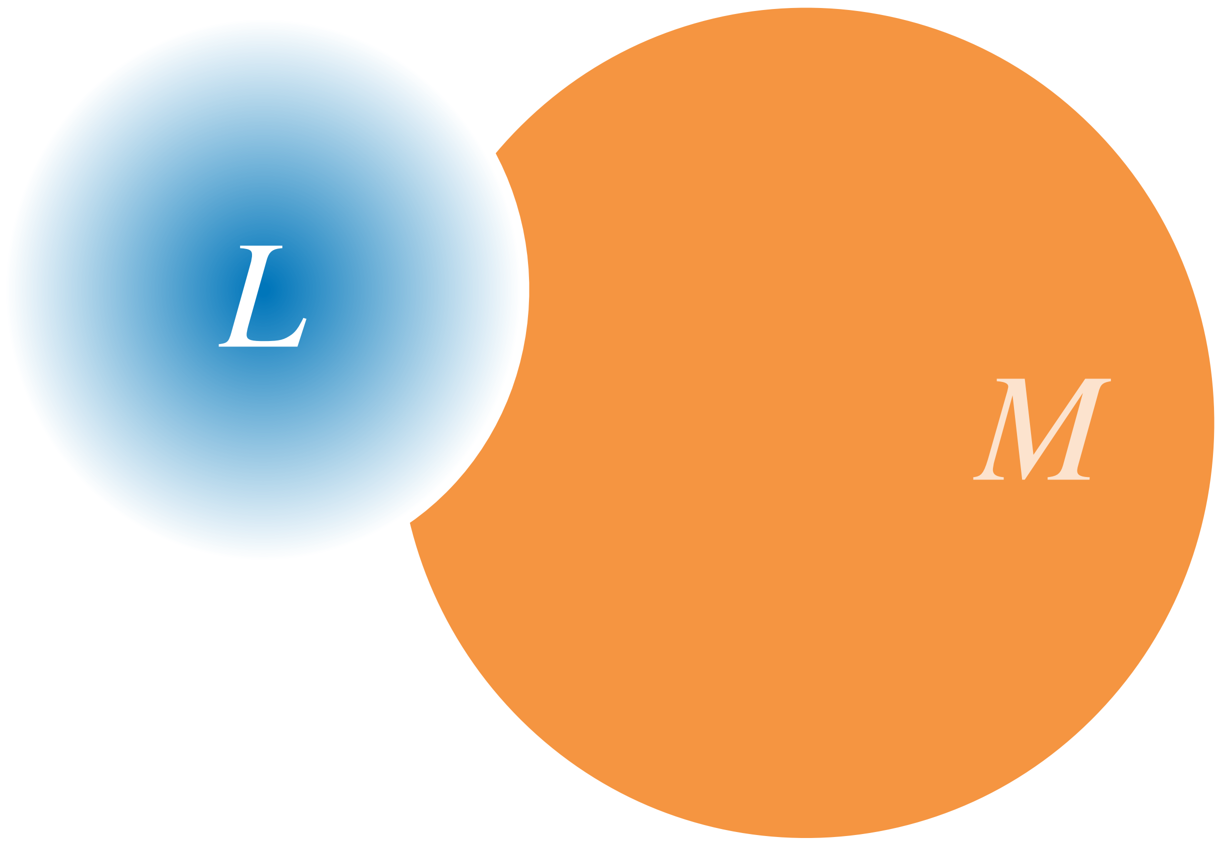


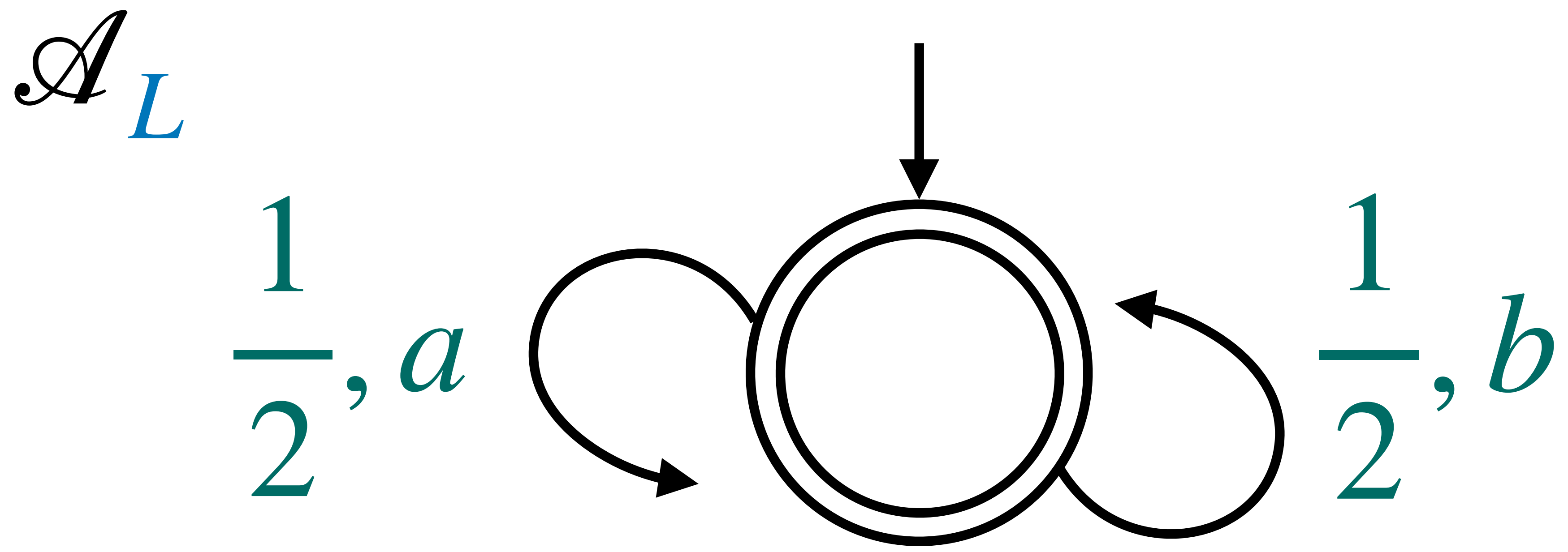
$\mathscr{A}_M$

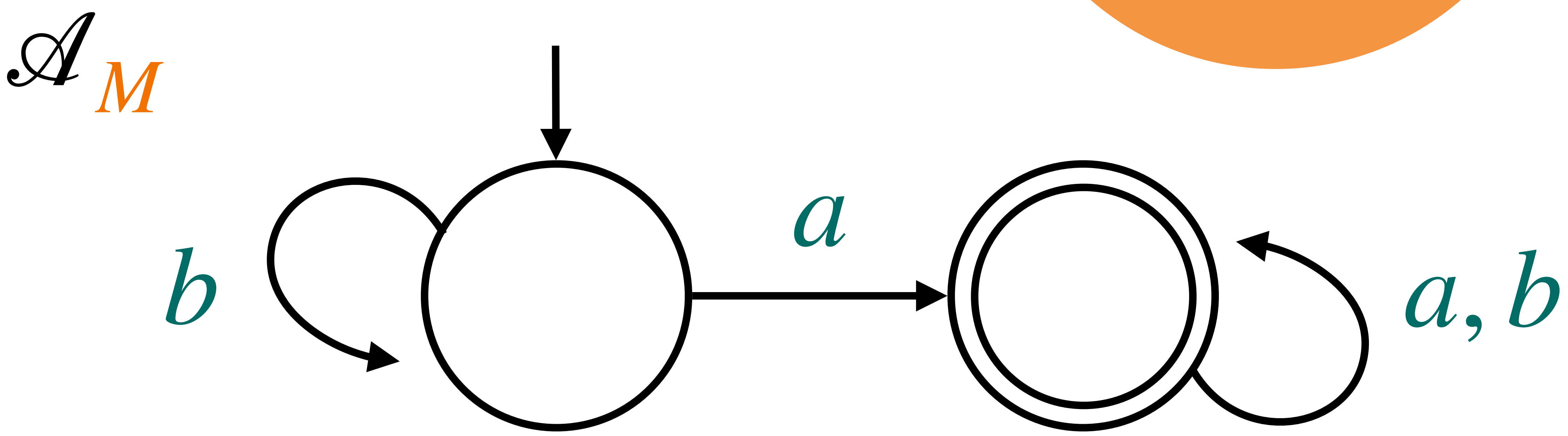


- L = uniform probability measure on $\{a, b\}^\omega$ (continuous)
- $M = b^*a(a+b)^\omega$ = "eventually a occurs"

Example

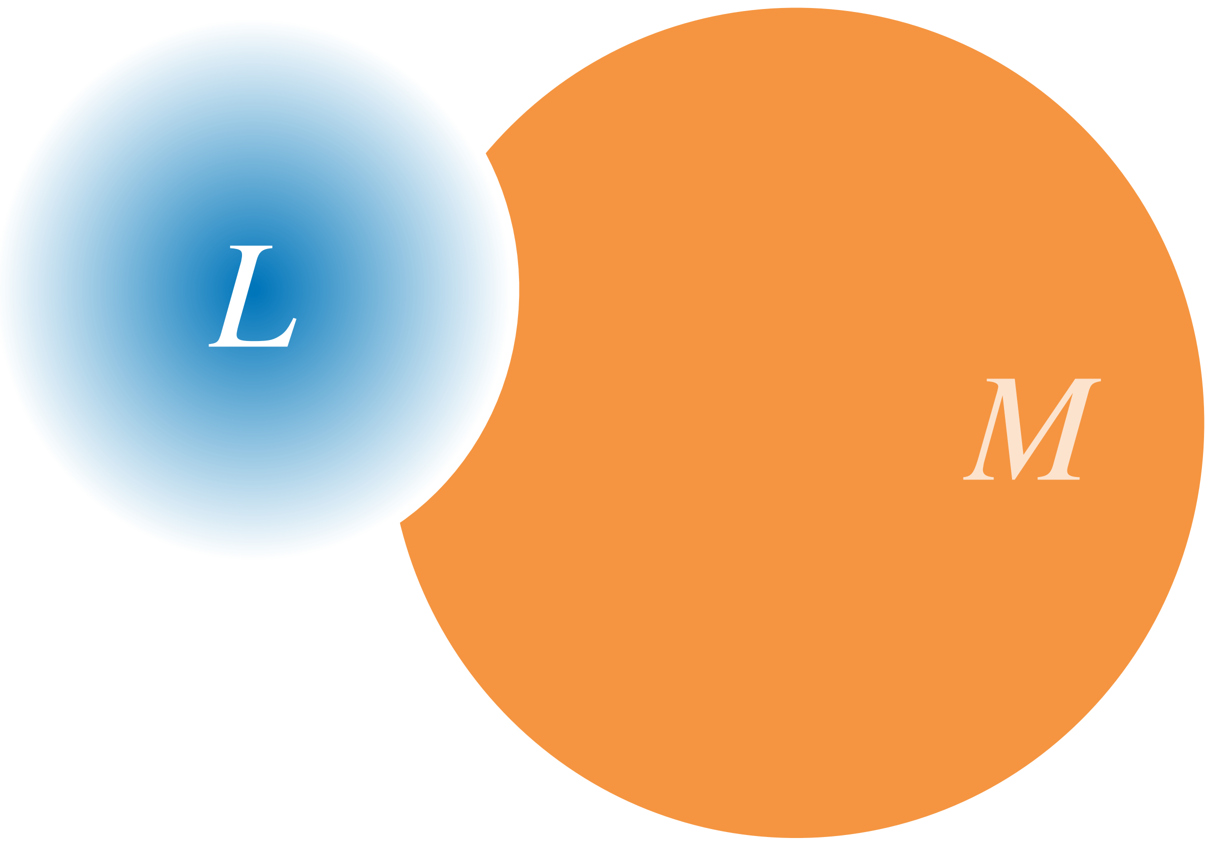


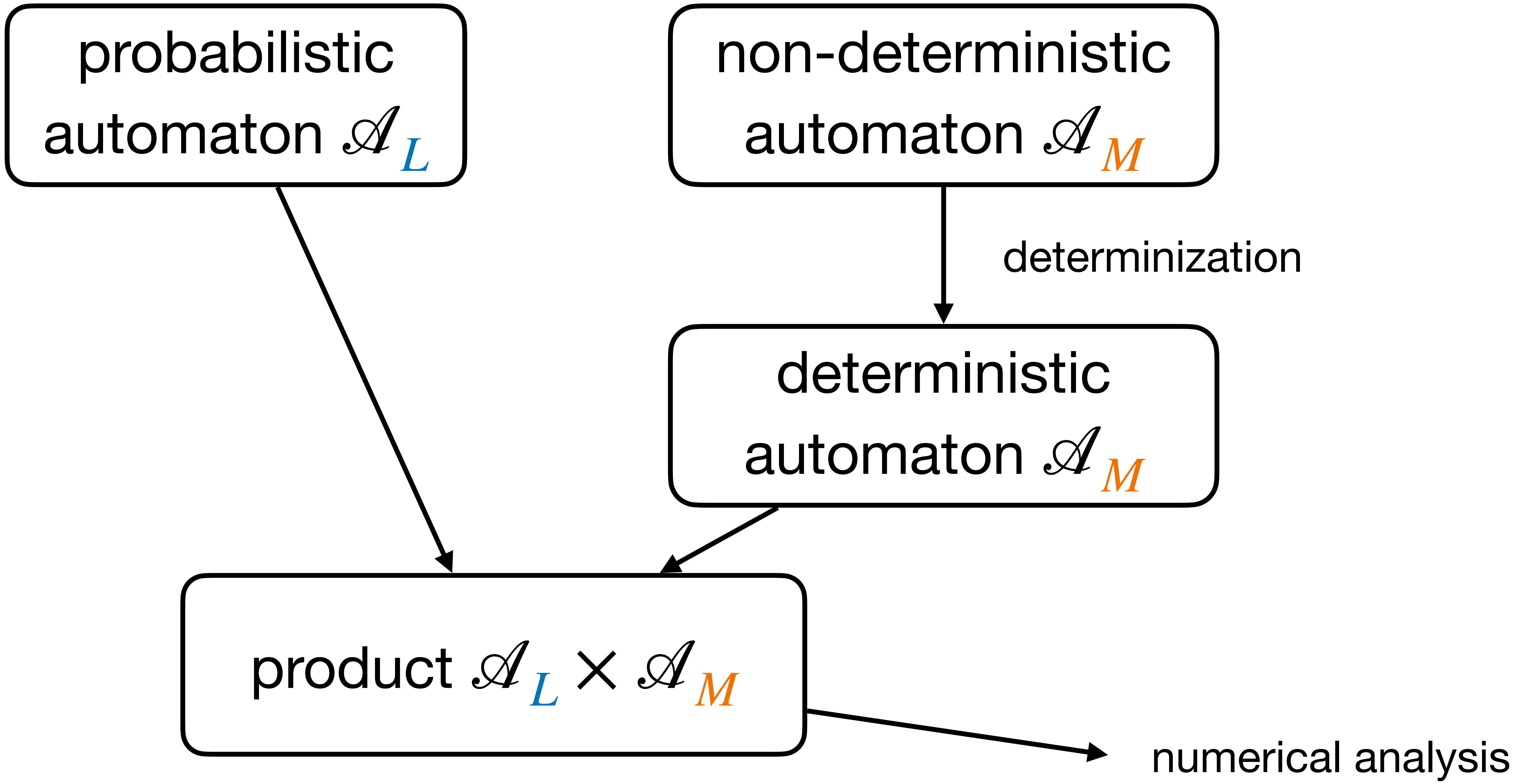




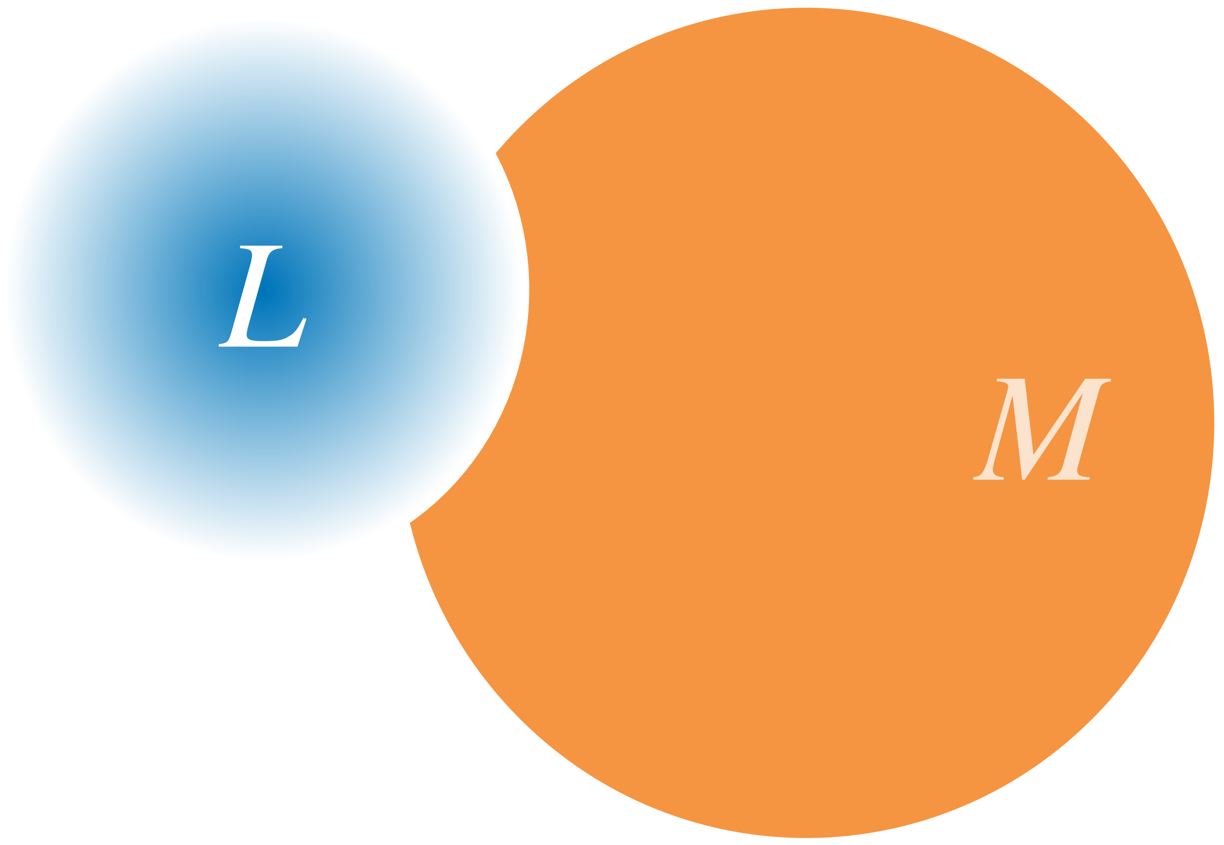
- L = uniform probability measure on $\{a, b\}^\omega$ (continuous)
- $M = b^*a(a+b)^\omega$ = "eventually a occurs"
- $Pr_{w \sim L}(w \in M)$ = ?

Basic Approach





Basic Approach



probabilistic automaton $\mathscr{A}_L$

non-deterministic automaton $\mathscr{A}_M$

determinization

often high complexity, loss of expressive power

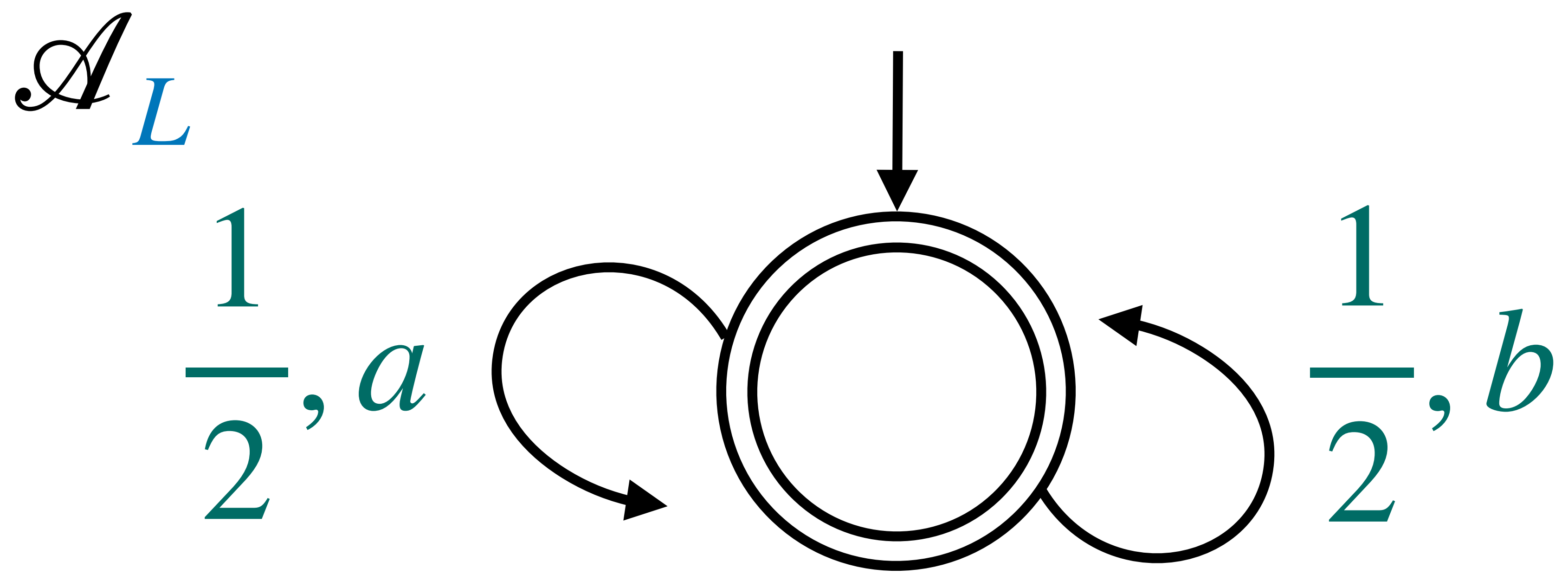
deterministic automaton $\mathscr{A}_M$

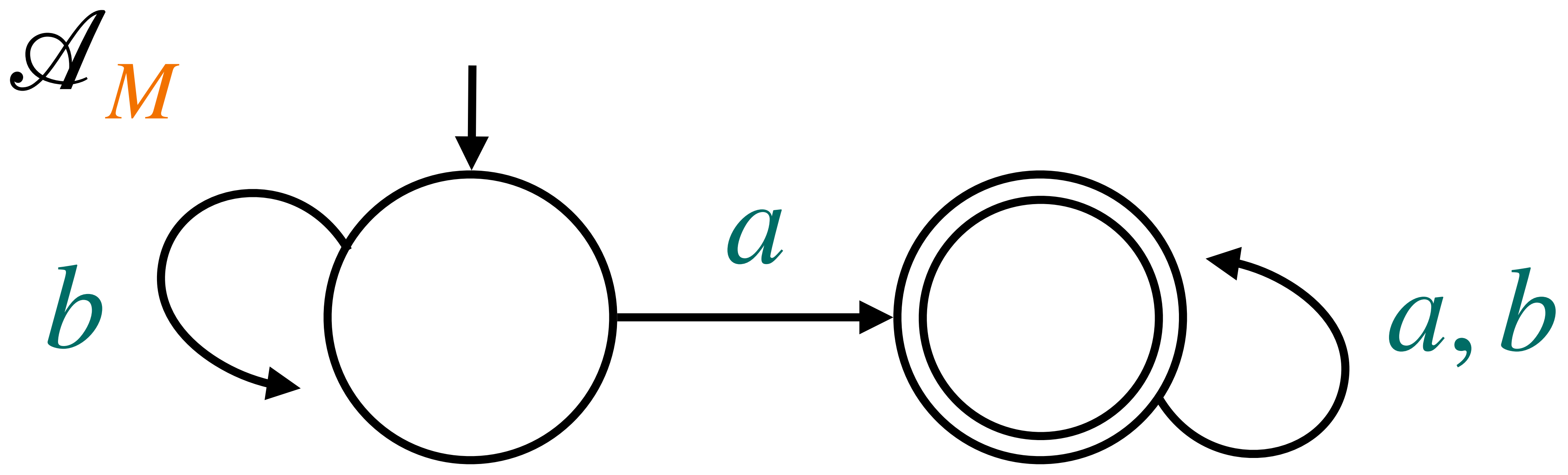
product $\mathscr{A}_L \times \mathscr{A}_M$

numerical analysis

Example cont'd

The Product Construction



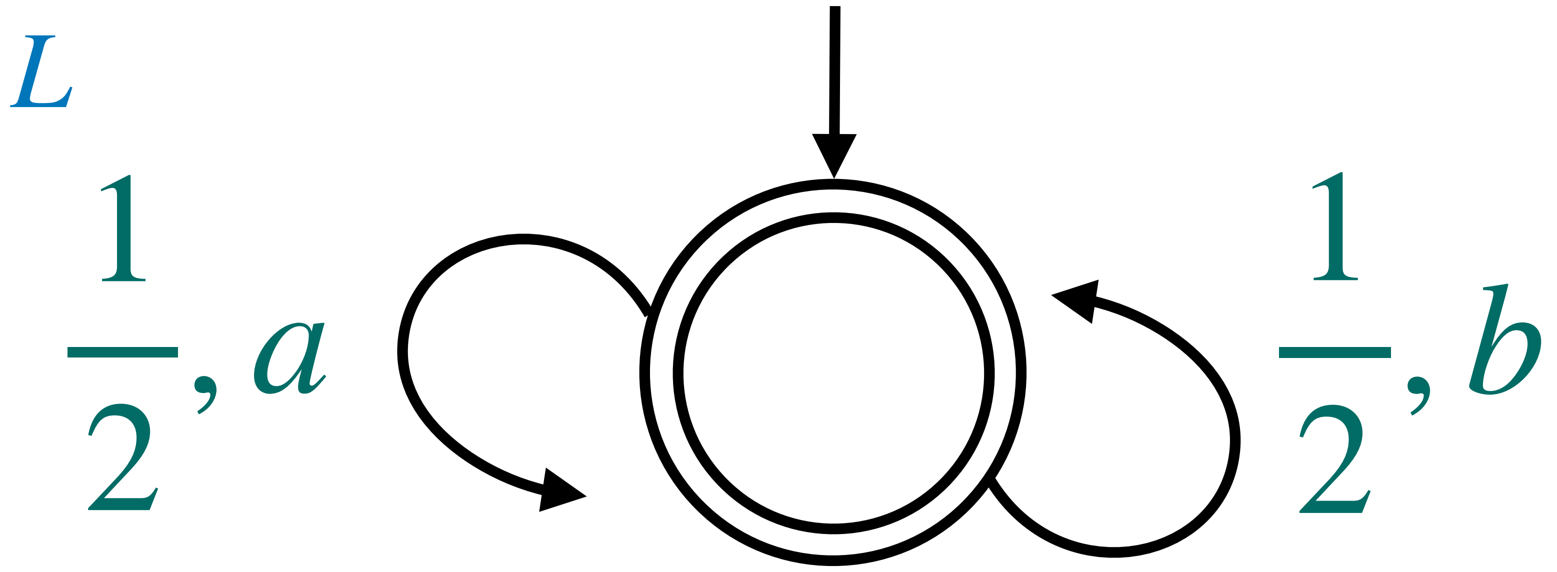


Example cont'd

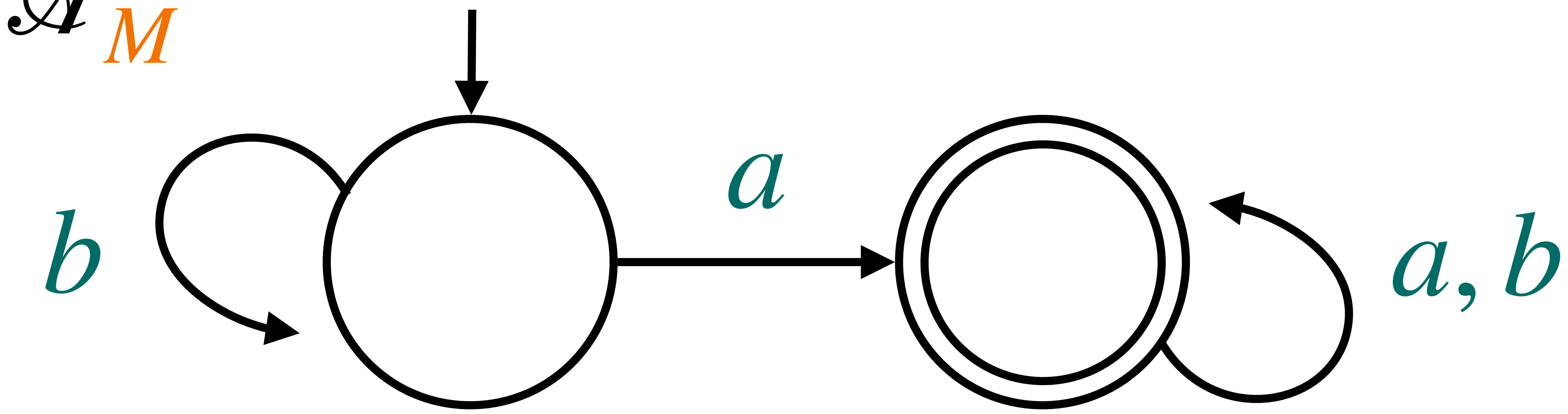
The Product Construction

already deterministic 🥳

$\mathscr{A}_L$



$\mathscr{A}_M$

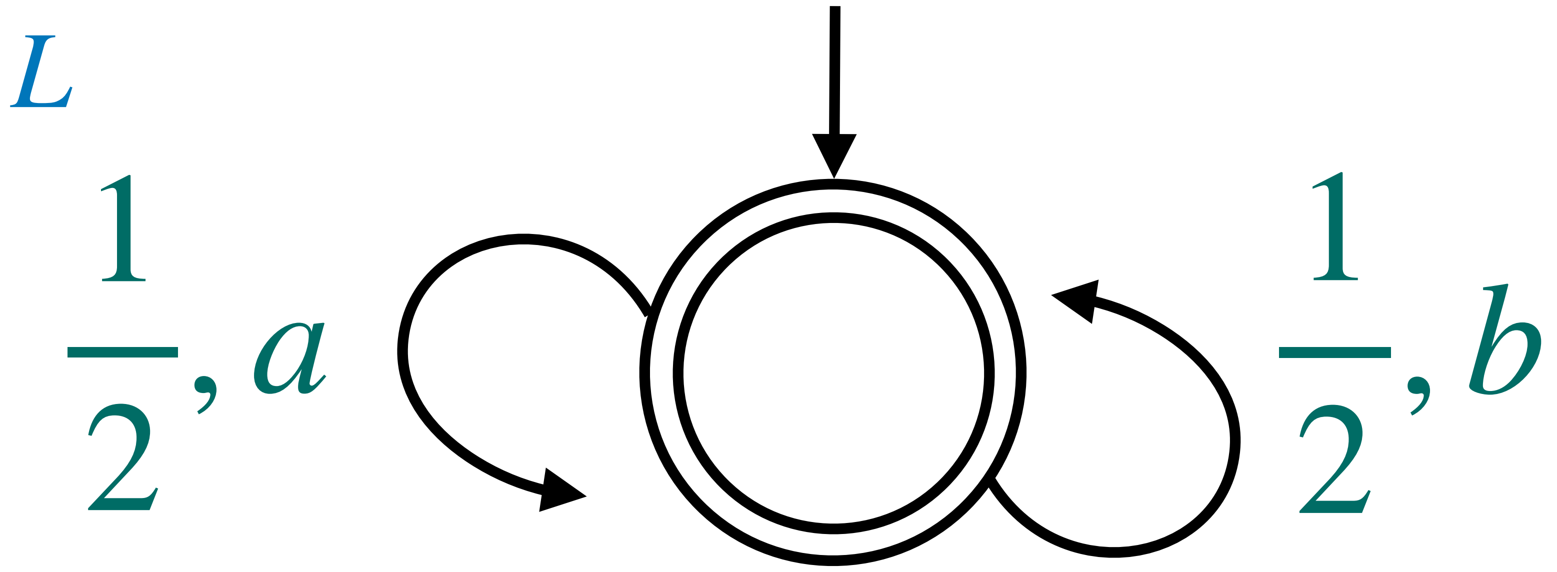


Example cont'd

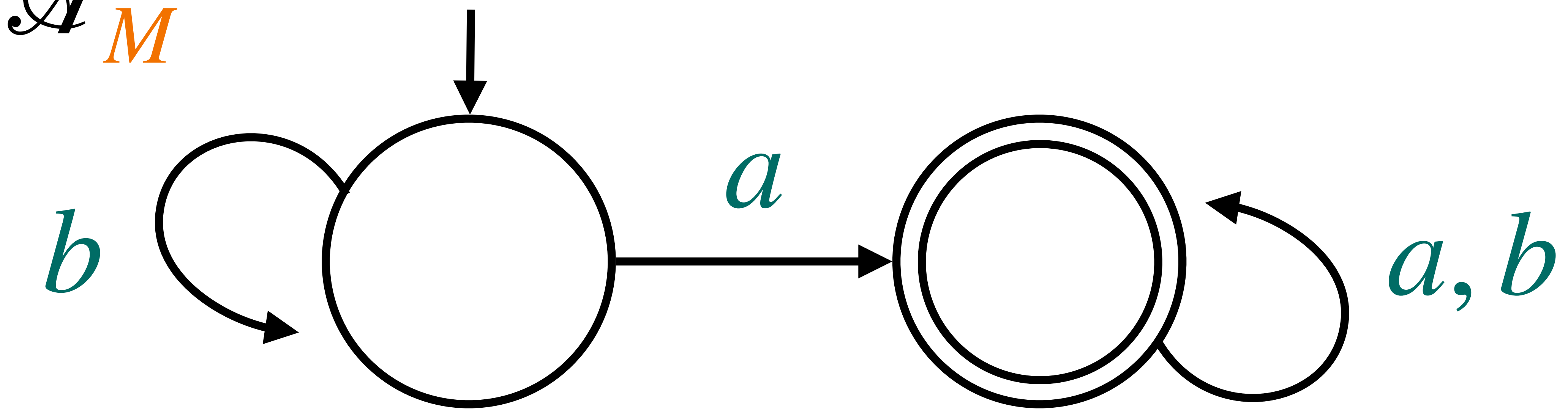
The Product Construction

already deterministic 🥳

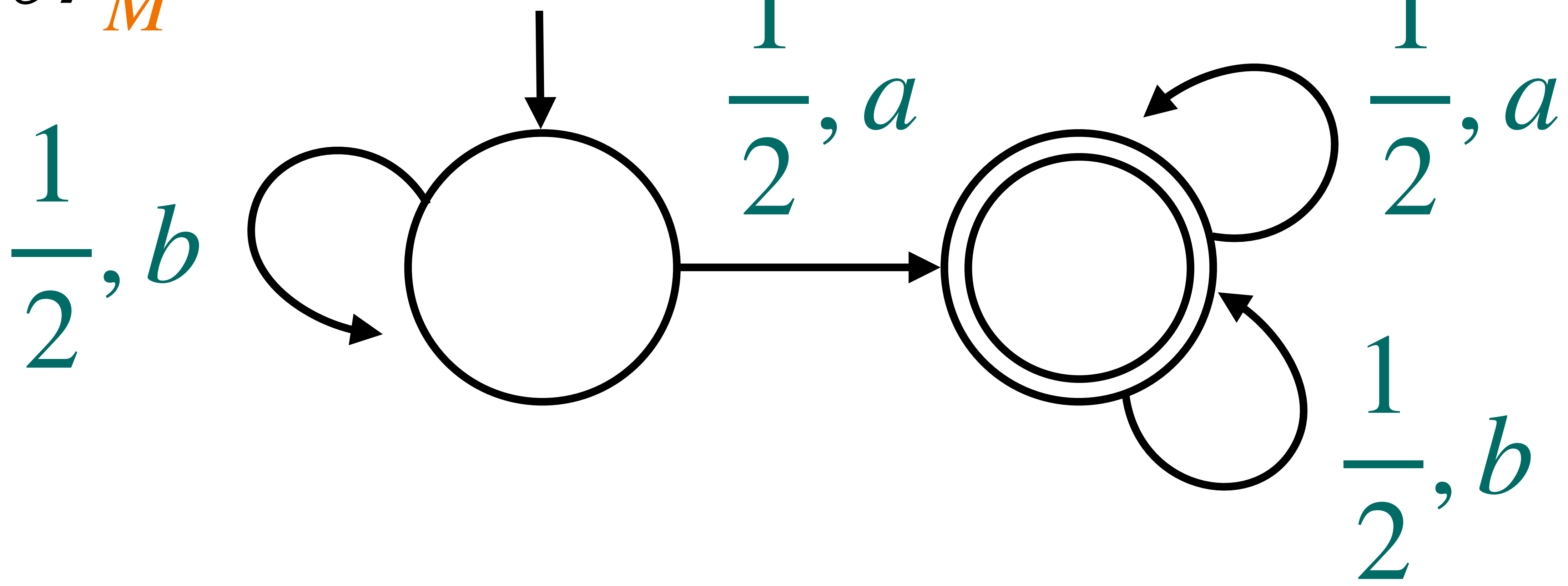
$\mathscr{A}_L$



$\mathscr{A}_M$



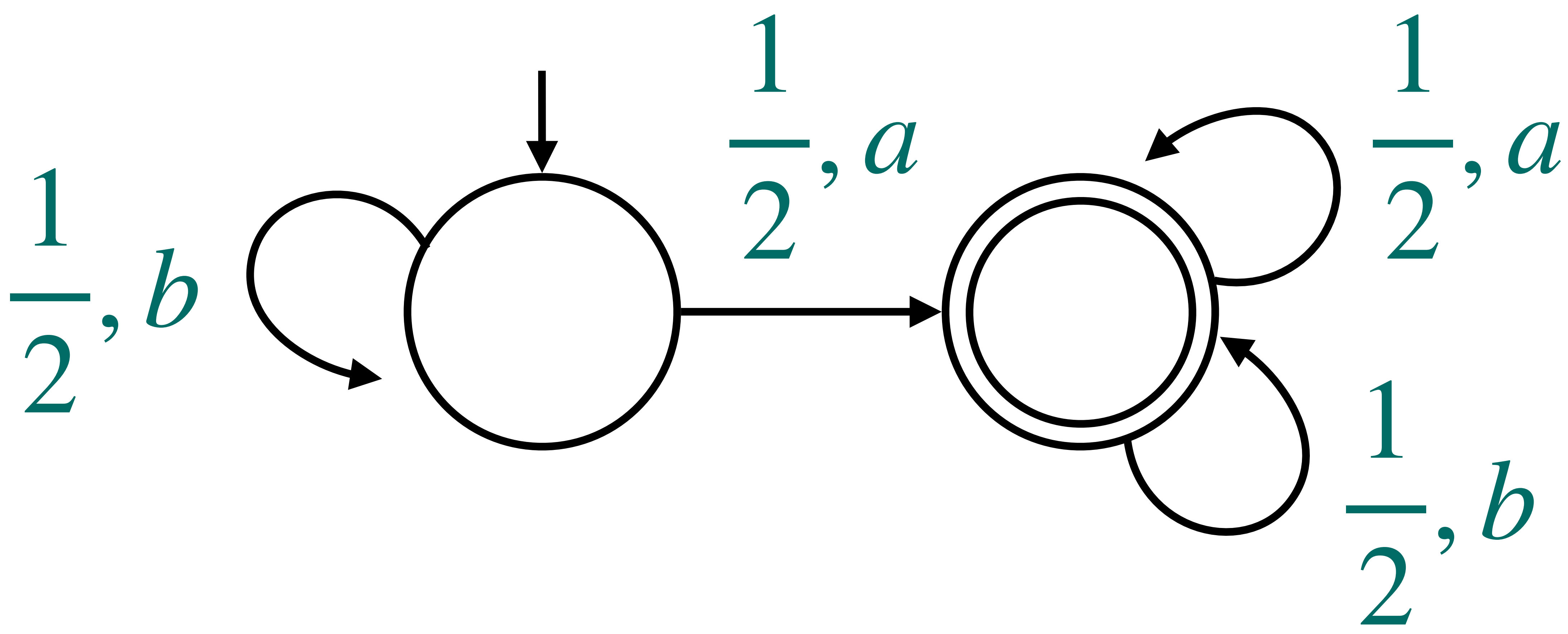
$\mathscr{A}_L \times \mathscr{A}_M$



Example cont'd

Meaning of the Product

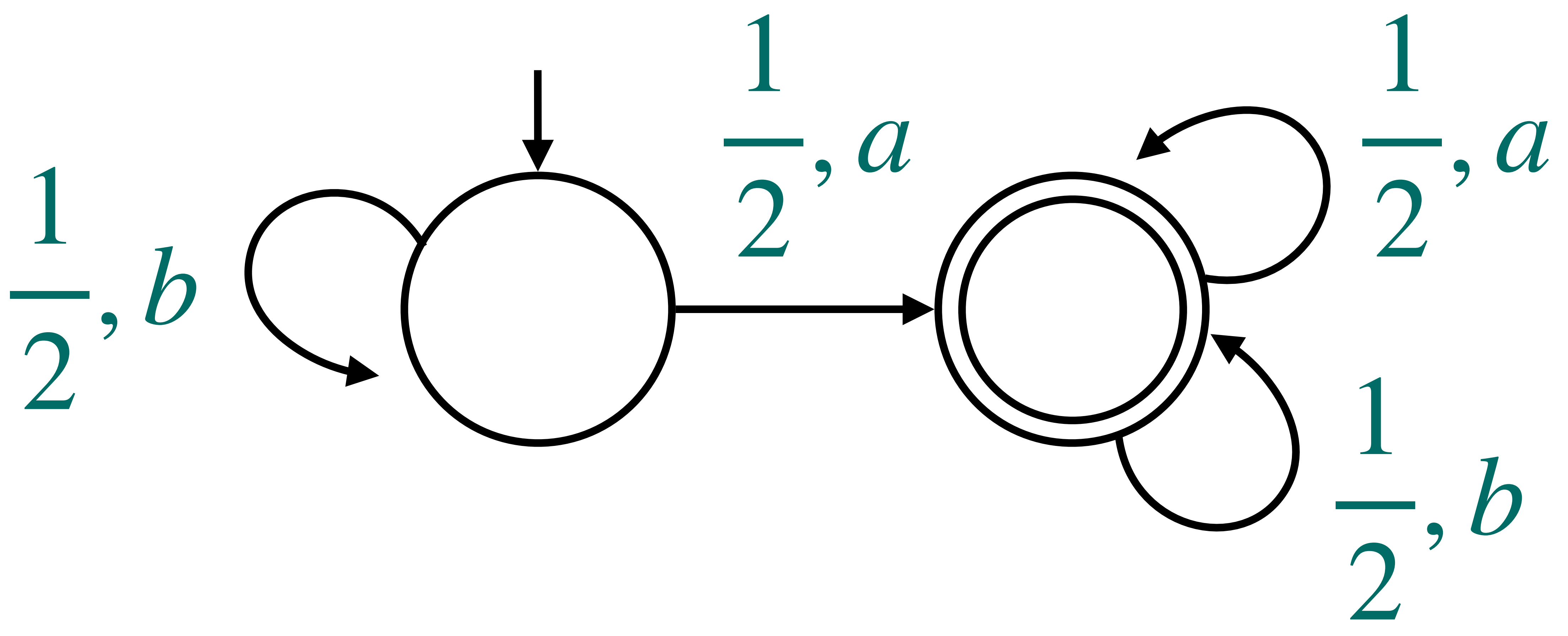
$\mathscr{A}_L \times \mathscr{A}_M$



Example cont'd

Meaning of the Product

$\mathscr{A}_L \times \mathscr{A}_M$

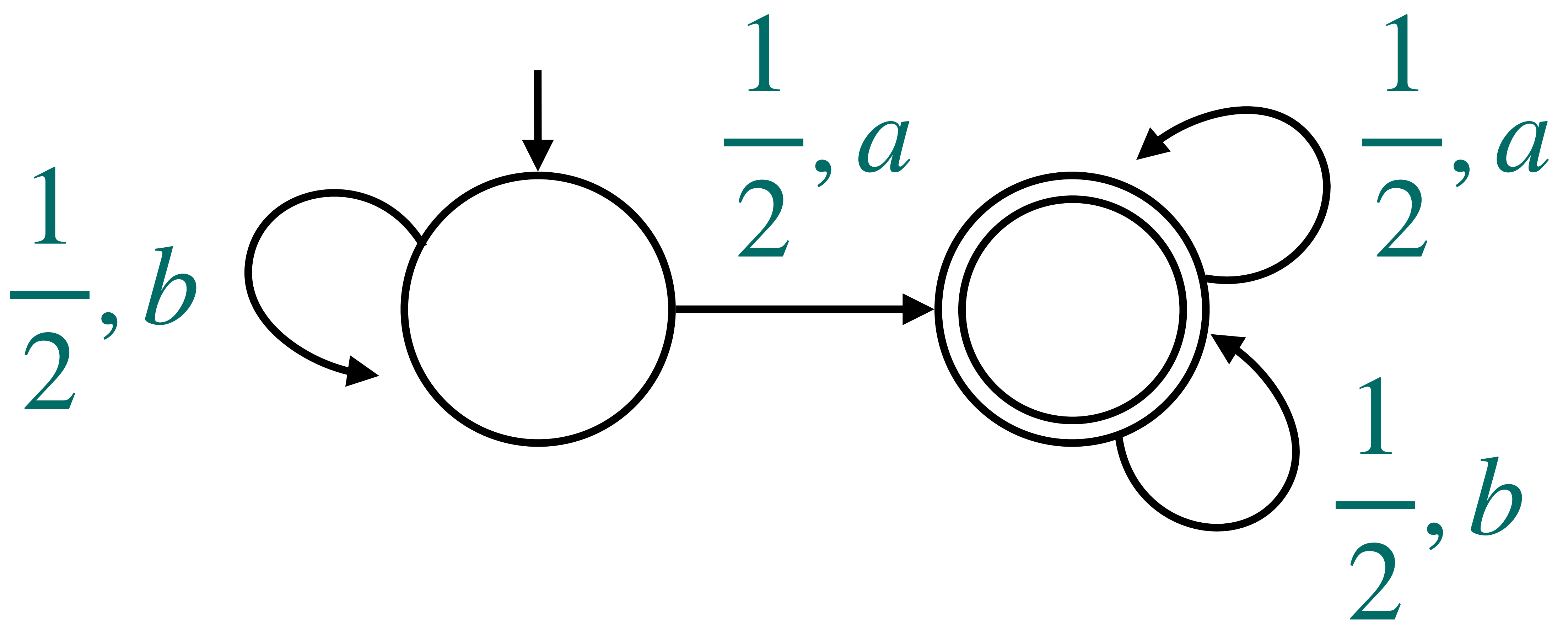


- $\mathscr{A}_L \times \mathscr{A}_M =$ probabilistic automaton with non-trivial Büchi condition

Example cont'd

Meaning of the Product

$\mathscr{A}_L \times \mathscr{A}_M$

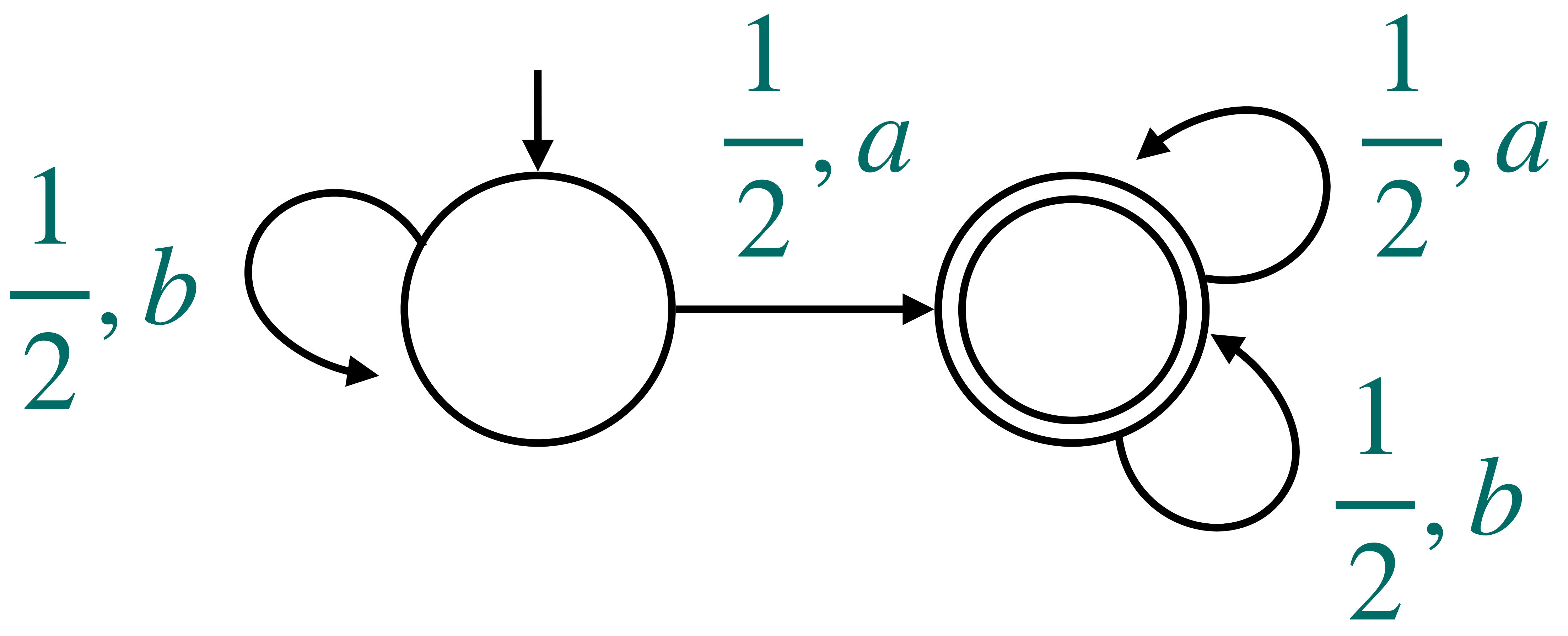


- $\mathscr{A}_L \times \mathscr{A}_M =$ probabilistic automaton with non-trivial Büchi condition
  - Defines sub-probability measure on words

Example cont'd

Meaning of the Product

$\mathscr{A}_L \times \mathscr{A}_M$

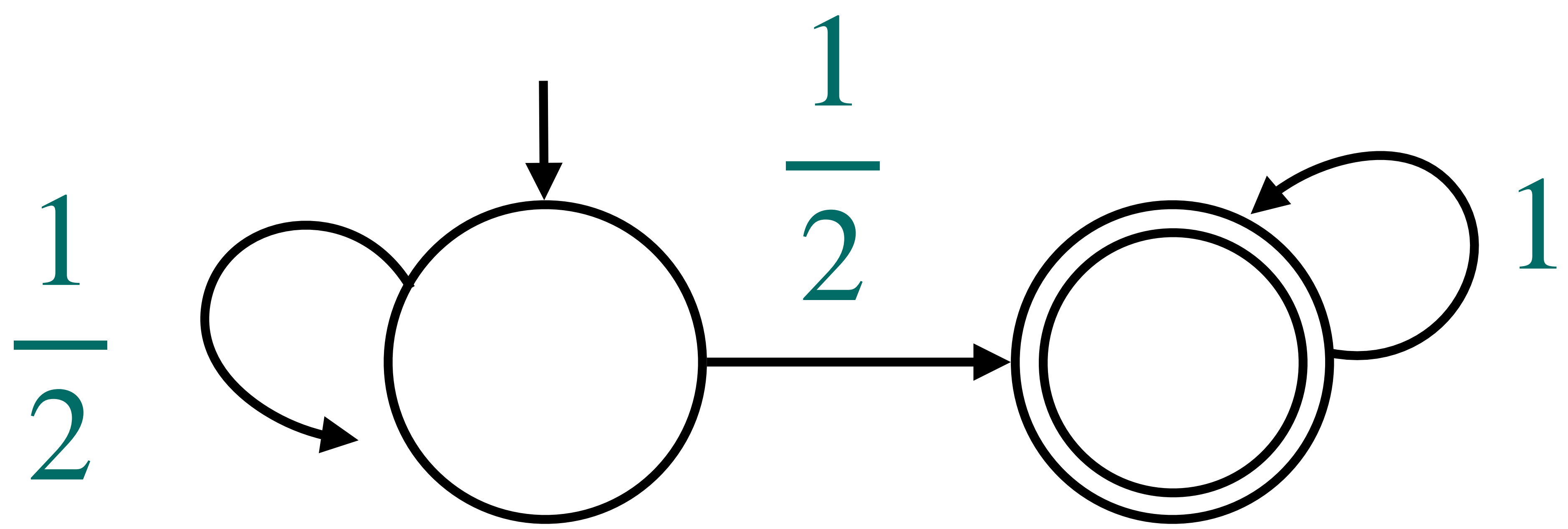


- $\mathscr{A}_L \times \mathscr{A}_M =$ probabilistic automaton with non-trivial Büchi condition
  - Defines sub-probability measure on words
  - $Pr_{w \sim L}(w \in M)$ = mass of that sub-probability measure
    = probability $\mathscr{A}_L \times \mathscr{A}_M$ generates a run it accepts

Example cont'd

Computing the Acceptance Probability

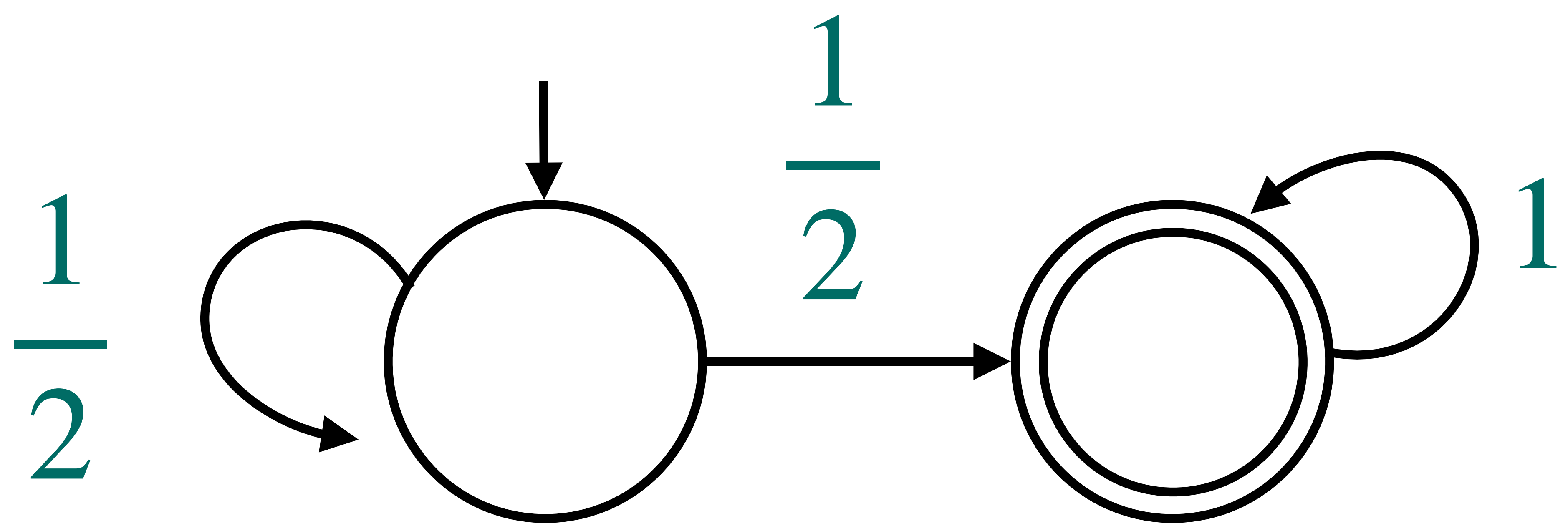
Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:



Example cont'd

Computing the Acceptance Probability

Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:

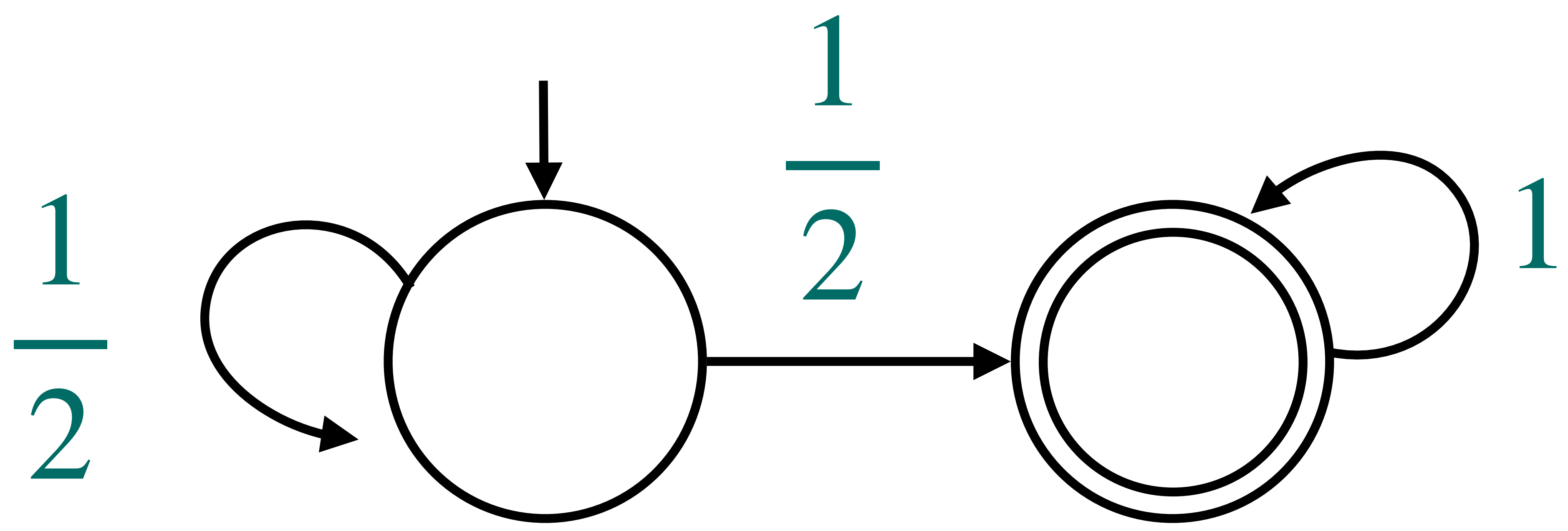


- Two facts:

Example cont'd

Computing the Acceptance Probability

Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:

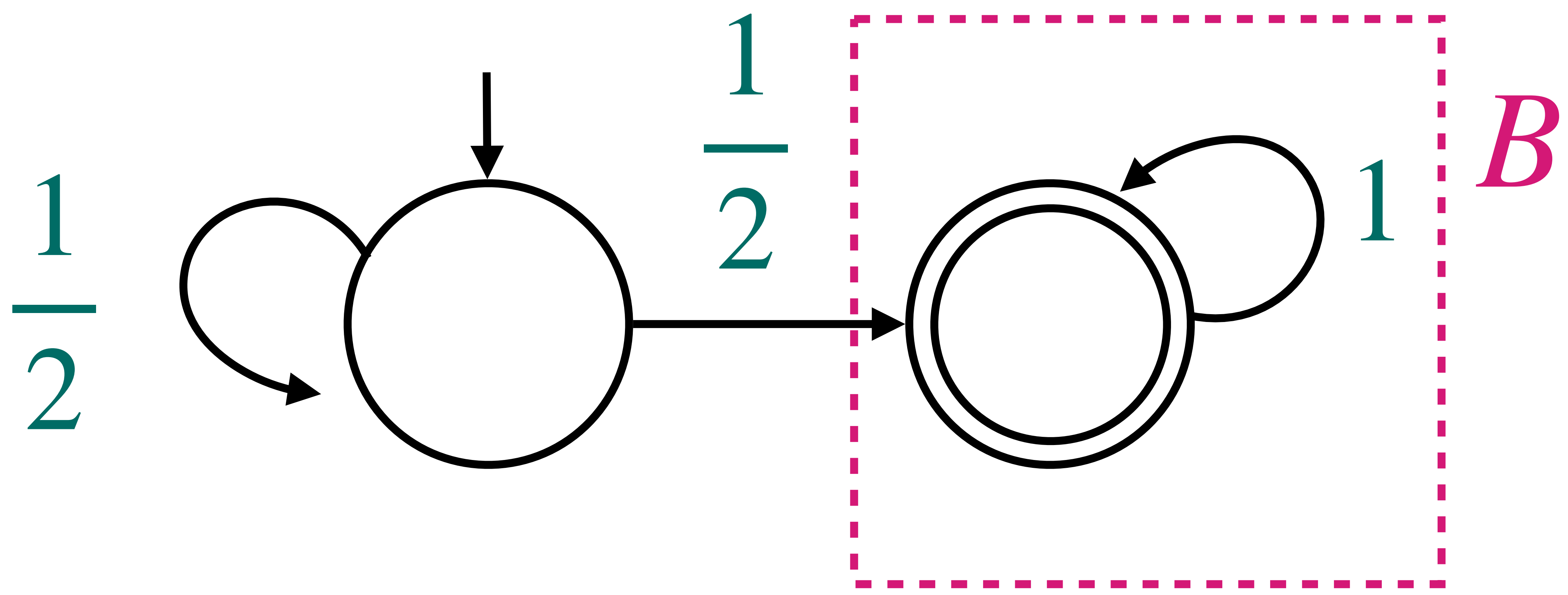


- Two facts:
  - A finite Markov chain reaches a bottom strongly-connected component (BSCC) with probability 1

Example cont'd

Computing the Acceptance Probability

Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:

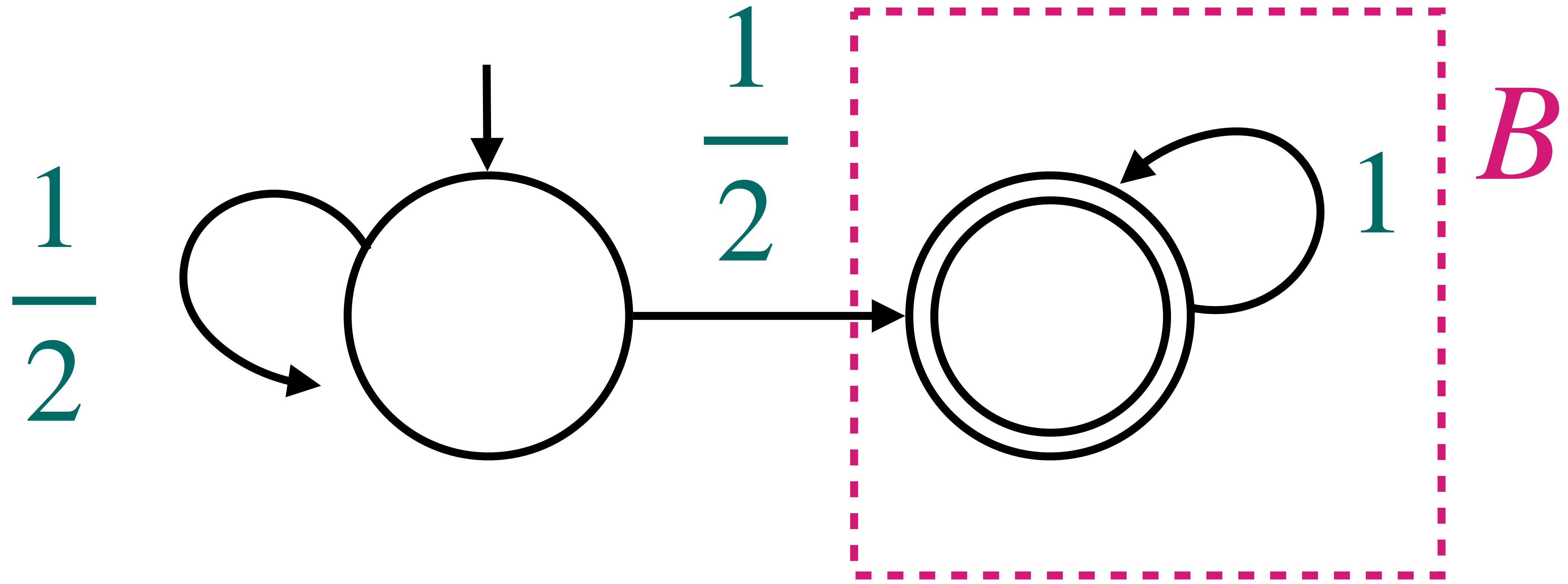


- Two facts:
  - A finite Markov chain reaches a bottom strongly-connected component (BSCC) with probability 1

Example cont'd

Computing the Acceptance Probability

Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:

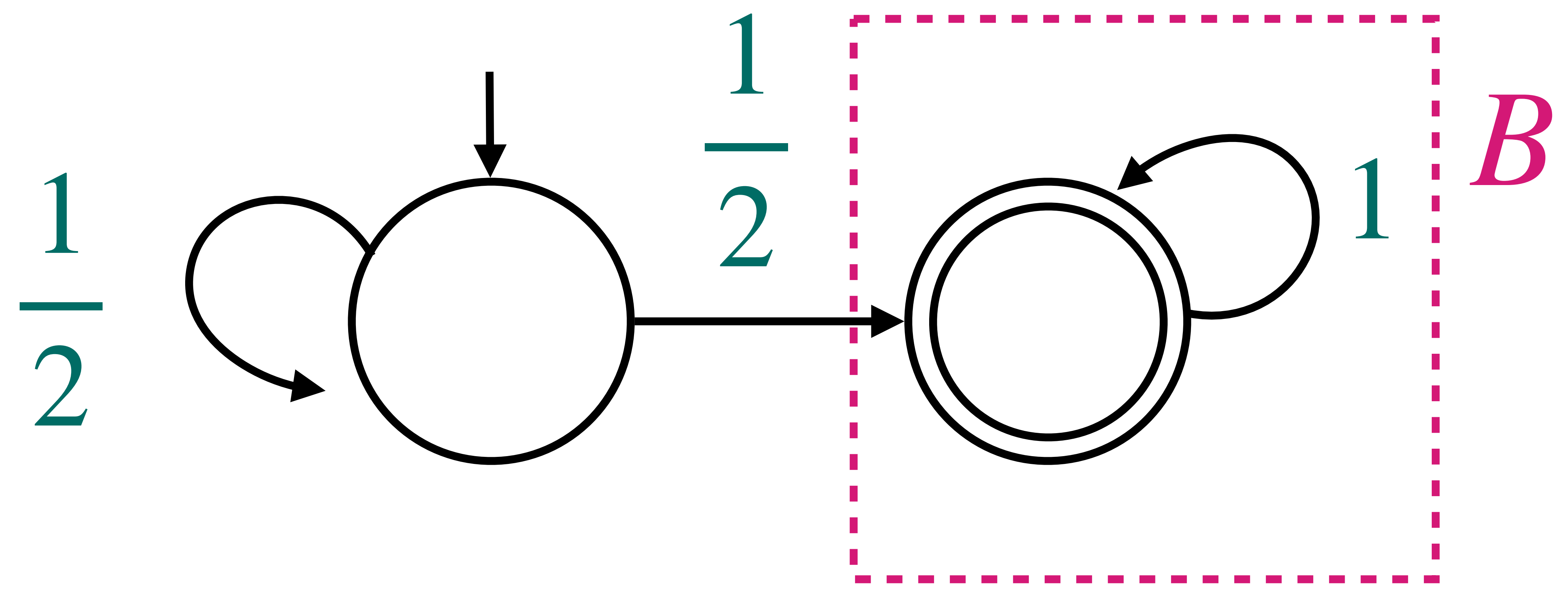


- Two facts:
  - A finite Markov chain reaches a bottom strongly-connected component (BSCC) with probability 1
  - If BSCC B is reached, then all states in B are visited ∞-often with probability 1

Example cont'd

Computing the Acceptance Probability

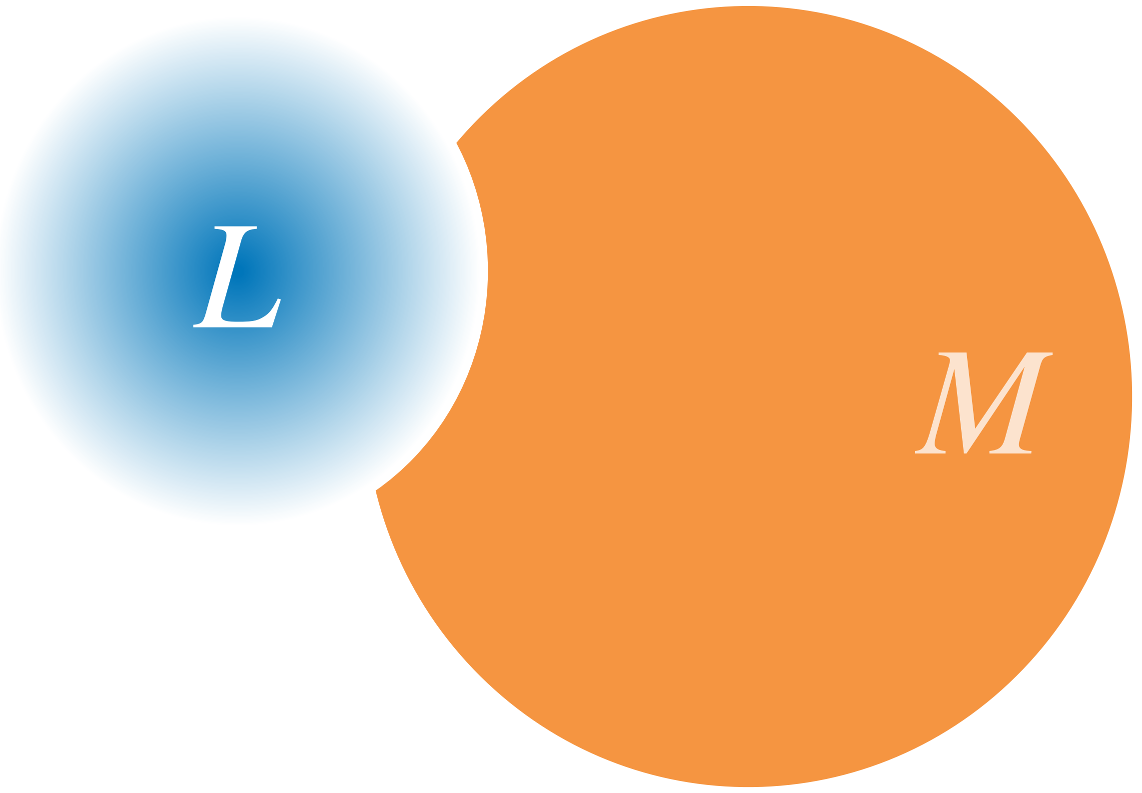
Dropping a, b from $\mathscr{A}_L \times \mathscr{A}_M$ yields finite Markov chain:



- Two facts:
  - A finite Markov chain reaches a bottom strongly-connected component (BSCC) with probability 1
  - If BSCC B is reached, then all states in B are visited ∞-often with probability 1
- $Pr_{w \sim L}(w \in M) = Pr(\text{reach a BSCC containing an accepting state}) = 1$

Some Known Results

(Büchi Acceptance)



$\mathscr{A}_L$	$\mathscr{A}_M$	$Pr_{w \sim L}(w \in M) = 1$?
probabilistic FA (Markov chain)	DFA	PTIME
	unambiguous FA	PTIME [Baier et al.]
	NFA	PSPACE [Courcoubetis & Yannakakis]
probabilistic PDA	NFA	EXPTIME [Etessami & Yannakakis]
	non-deterministic PDA	undecidable [Dubslaff et al.]
probabilistic visibly PDA	deterministic visibly PDA	in PSPACE [W. et al.]
	unambiguous visibly PDA	?
	non-deterministic visibly PDA	EXPTIME [W. et al.]

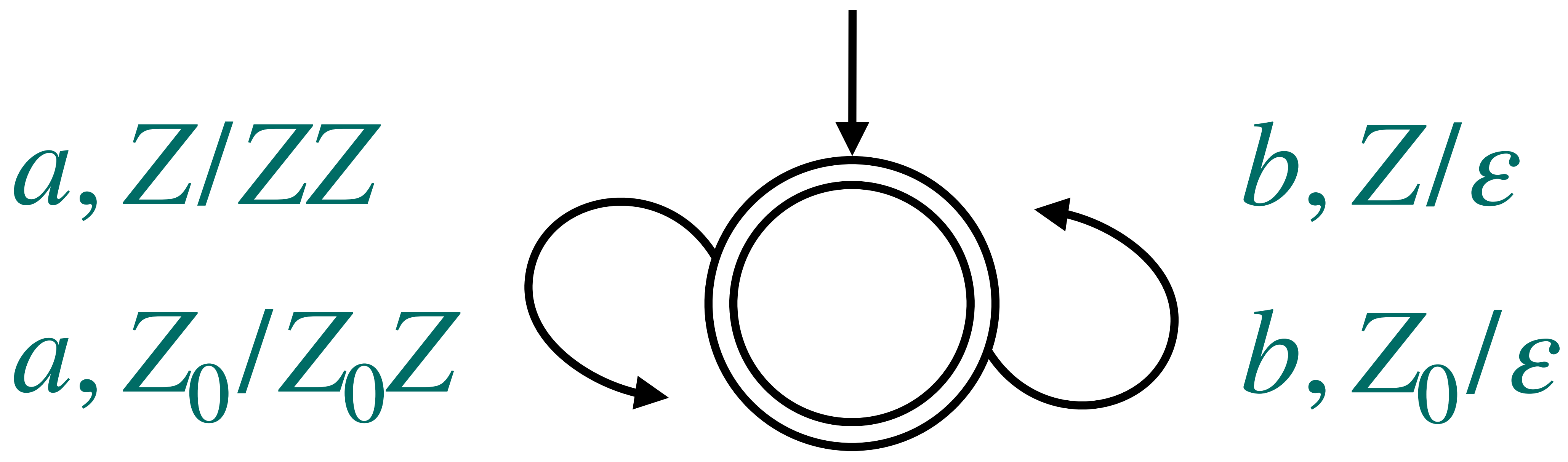
- Many other versions by varying type of L, M, acceptance condition, etc.

Part II
Probabilistic Visibly Pushdown Language Inclusion

Based on a FoSSaCS'22 paper with Christina Gehnen and Joost-Pieter Katoen

Visibly Pushdown Automata (VPA)

[Alur & Madhusudan ’04]



Stack alphabet $\Gamma = \{Z_0, Z\}$

- Z_0 initially on stack
- Z_0 cannot be popped nor pushed

Visibly Pushdown Automata (VPA)

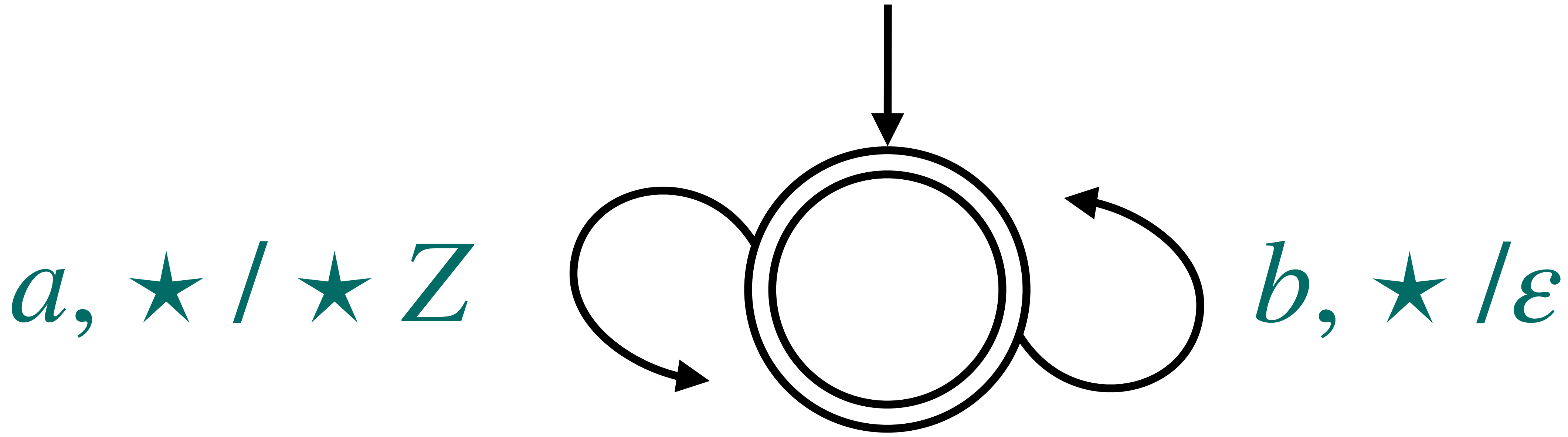
[Alur & Madhusudan ’04]

Stack alphabet $\Gamma = \{Z_0, Z\}$

- Z_0 initially on stack
- Z_0 cannot be popped nor pushed

Visibly Pushdown Automata (VPA)

[Alur & Madhusudan ’04]

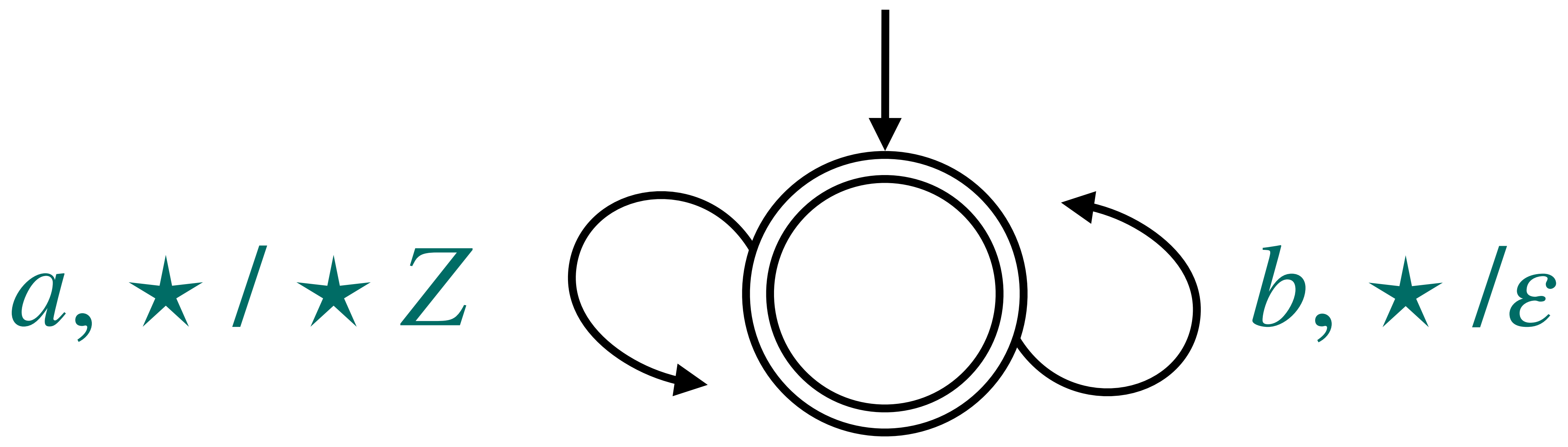


Stack alphabet $\Gamma = \{Z_0, Z\}$

- Z_0 initially on stack
- Z_0 cannot be popped nor pushed

Visibly Pushdown Automata (VPA)

[Alur & Madhusudan ’04]

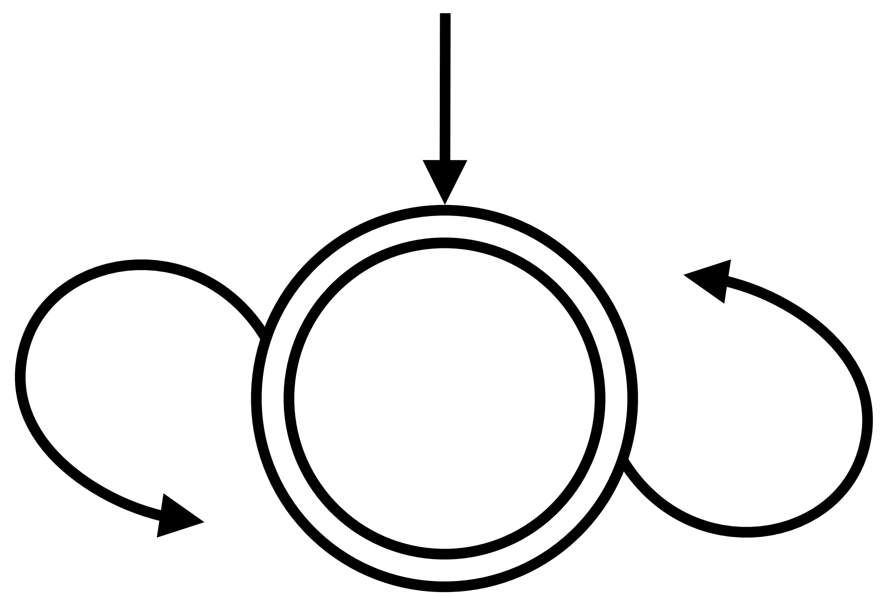


Stack alphabet $\Gamma = \{Z_0, Z\}$

- Z_0 initially on stack
- Z_0 cannot be popped nor pushed

Visibly Pushdown Automata (VPA)

[Alur & Madhusudan '04]



$\Sigma = \Sigma_{push} \uplus \Sigma_{pop} \uplus \Sigma_{int} = \{a\} \uplus \{b\} \uplus \varnothing$

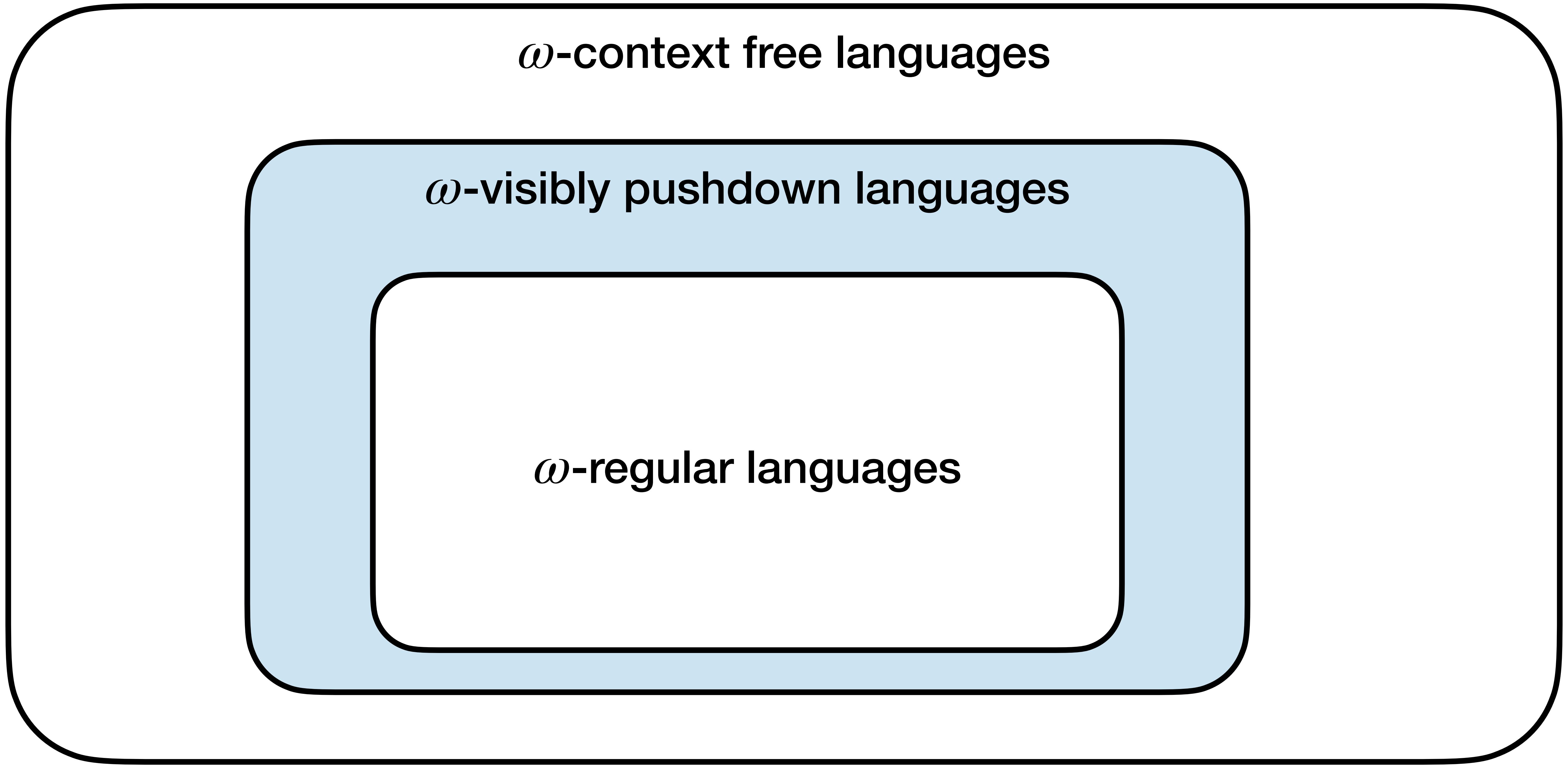
- Reading a must trigger a push
- Reading b must trigger a pop
- (Reading symbol from Σ_{int} must not change stack height)

Stack alphabet $\Gamma = \{Z_0, Z\}$

- Z_0 initially on stack
- Z_0 cannot be popped nor pushed

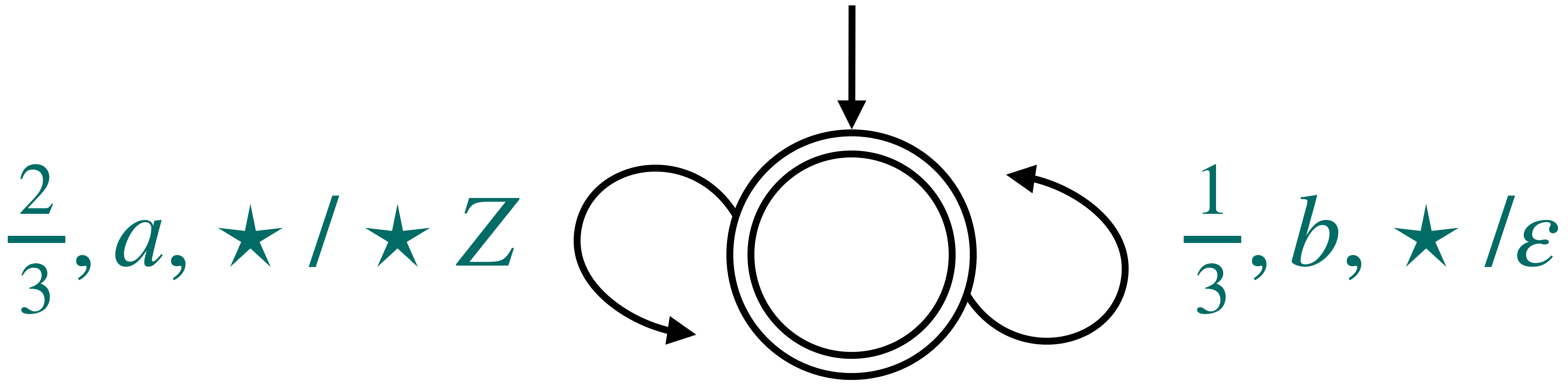
ω-Visibly Pushdown Languages (ωVPL)

L is an ω-visibly pushdown language if $L = L(\mathscr{A})$ for a Büchi VPA $\mathscr{A}$

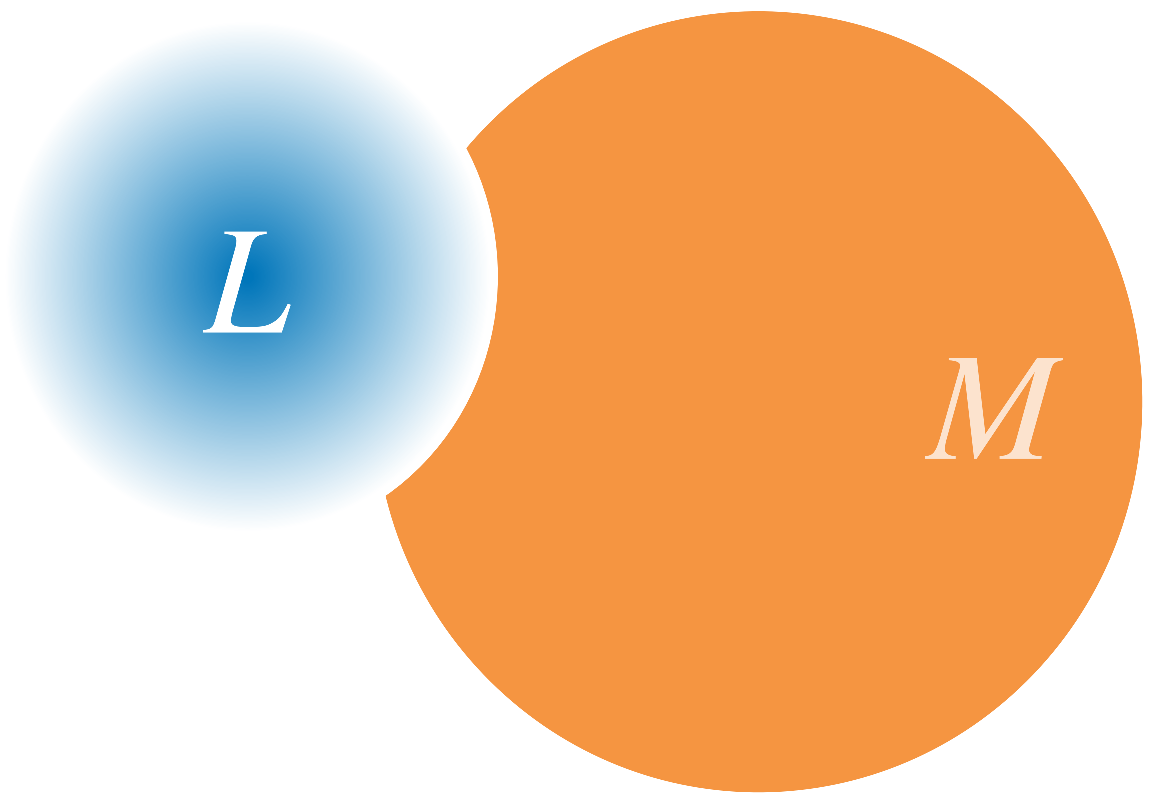


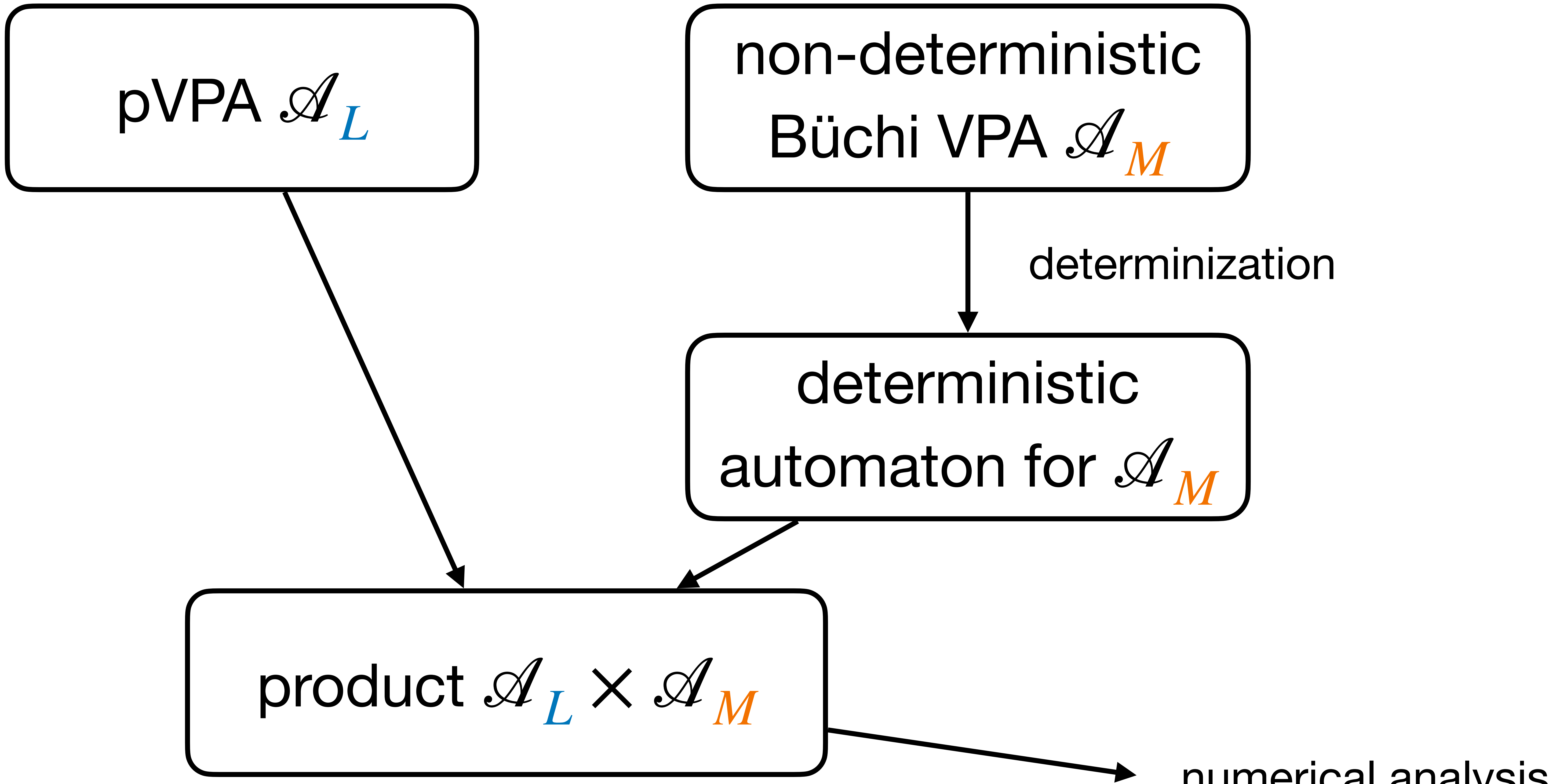
Probabilistic Visibly Pushdown Automata (pVPA)

- "Generative" probabilistic pushdown automaton $\mathscr{A}_L$ with trivial Büchi acceptance

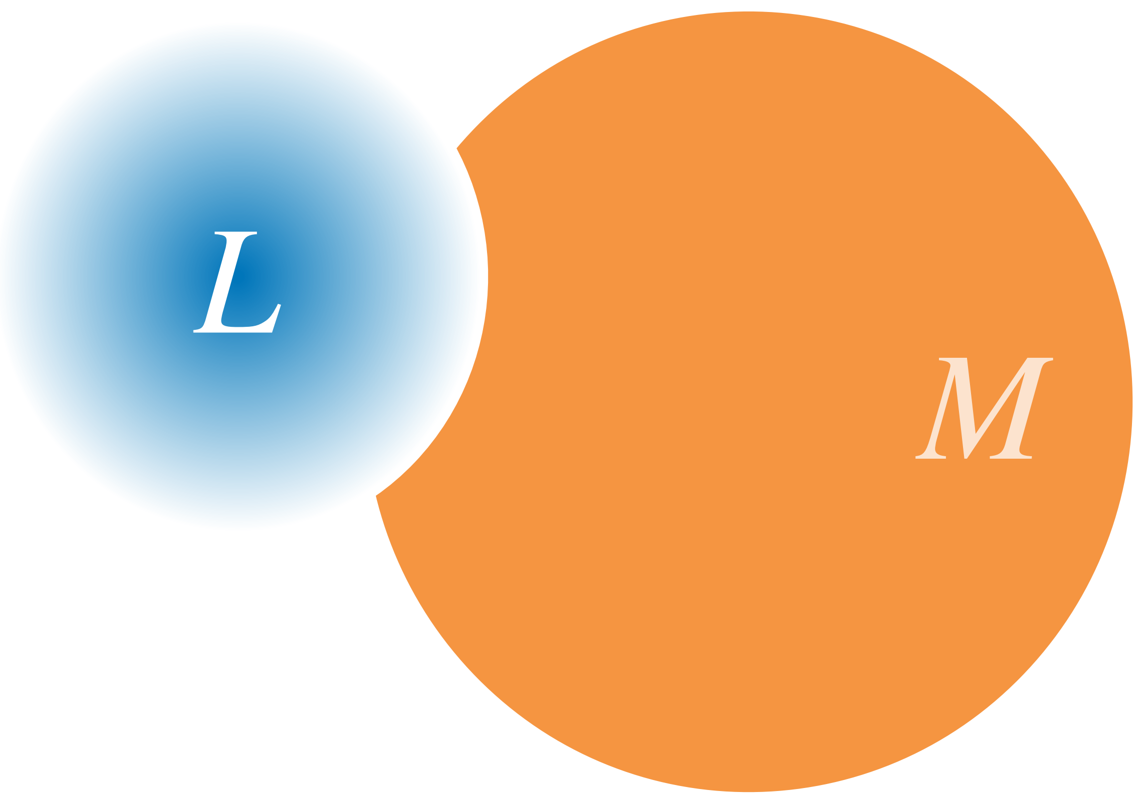


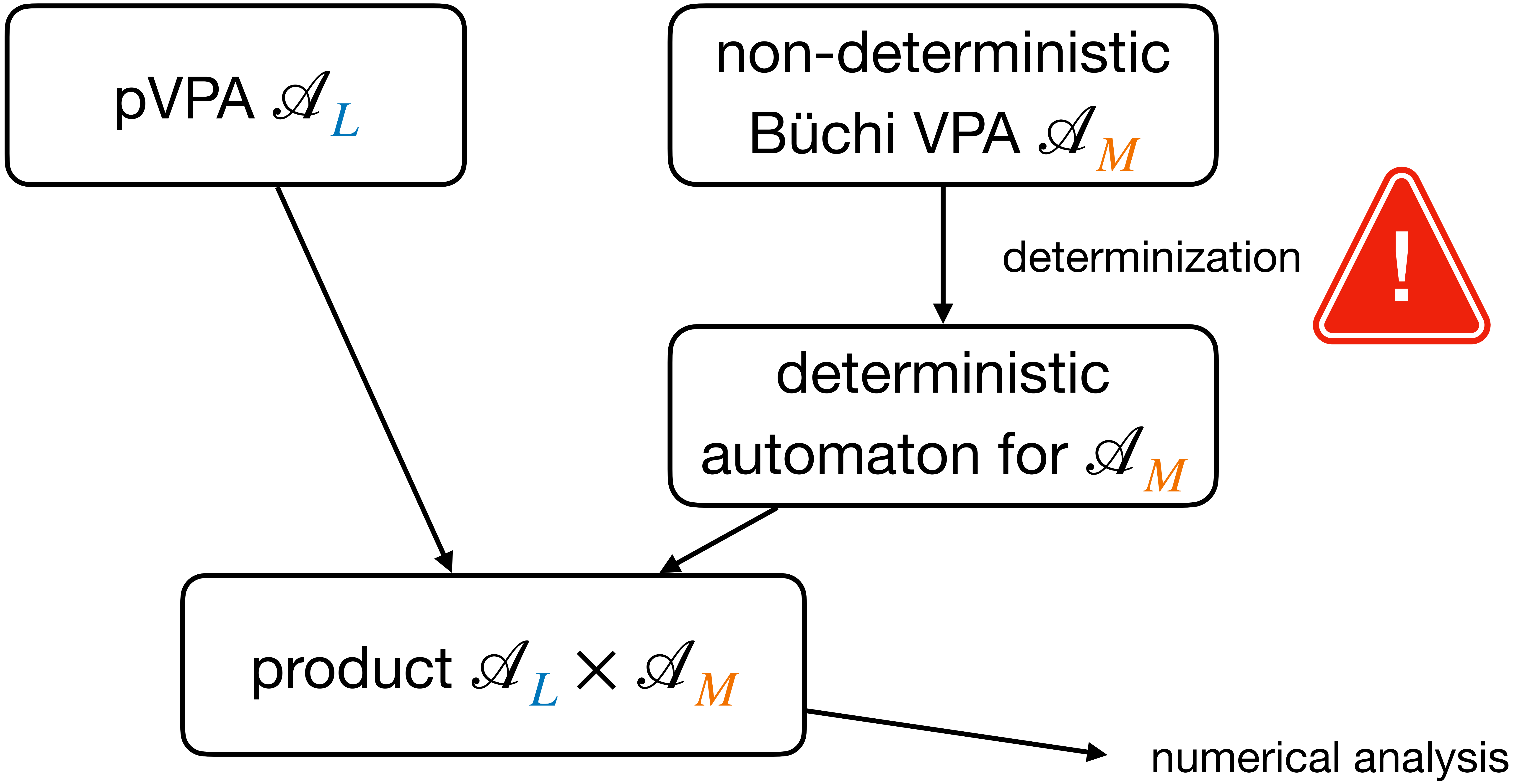
Probabilistic ωVPL Inclusion





Probabilistic ωVPL Inclusion





Determinizing VPA

[Löding, Madhusudan, Serre '04]

Every VPA $\mathscr{A}$ with Büchi acceptance can be transformed into an equivalent deterministic VPA $\mathscr{D}$ with $|\mathscr{D}| \in O(2^{|\mathscr{A}|^2})$.

$\mathscr{D}$ has a stair-parity acceptance condition.

Stair-Parity Acceptance

Stair-Parity Acceptance

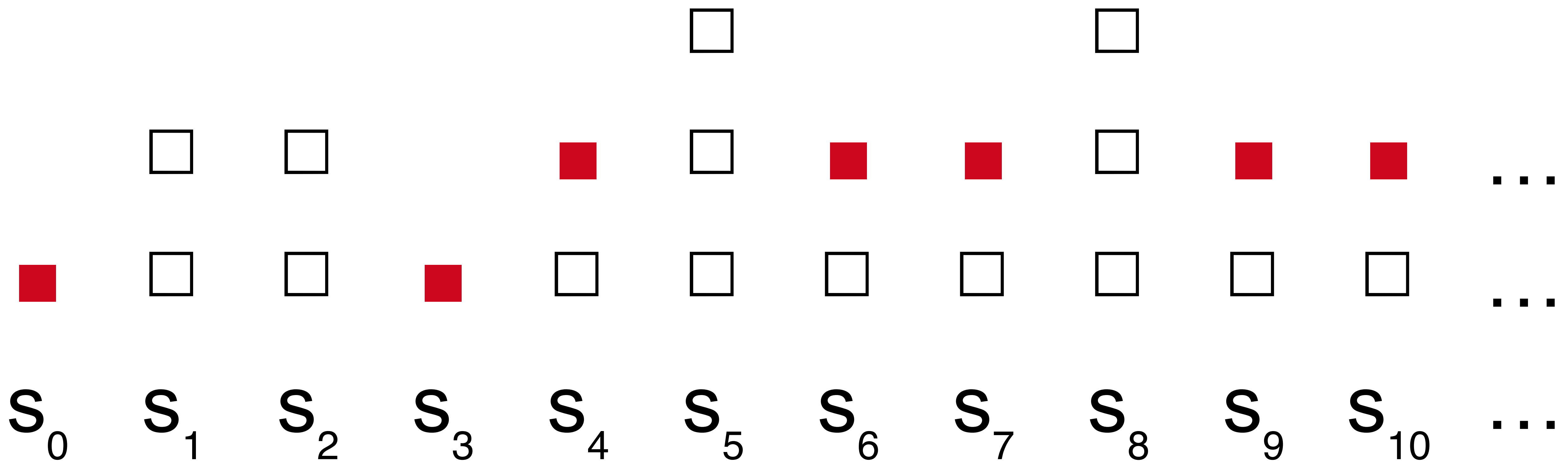
- Priority function $\Omega\colon \mathit{States} \to \mathbb{N}$ (like standard parity)

Stair-Parity Acceptance

- Priority function $\Omega : States \to \mathbb{N}$ (like standard parity)
- Position i is a step of an ω-run of a PDA $\iff \forall j \geq i : stackHeight(j) \geq stackHeight(i)$

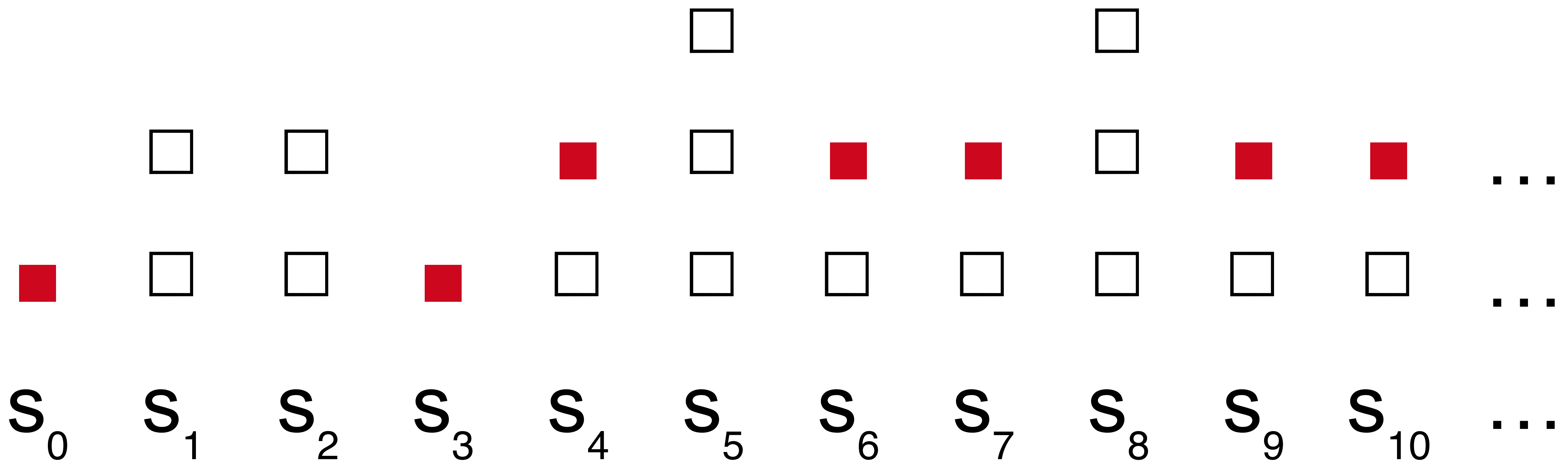
Stair-Parity Acceptance

- Priority function $\Omega : States \to \mathbb{N}$ (like standard parity)
- Position i is a step of an ω-run of a PDA $\iff$ $\forall j \geq i : stackHeight(j) \geq stackHeight(i)$



Stair-Parity Acceptance

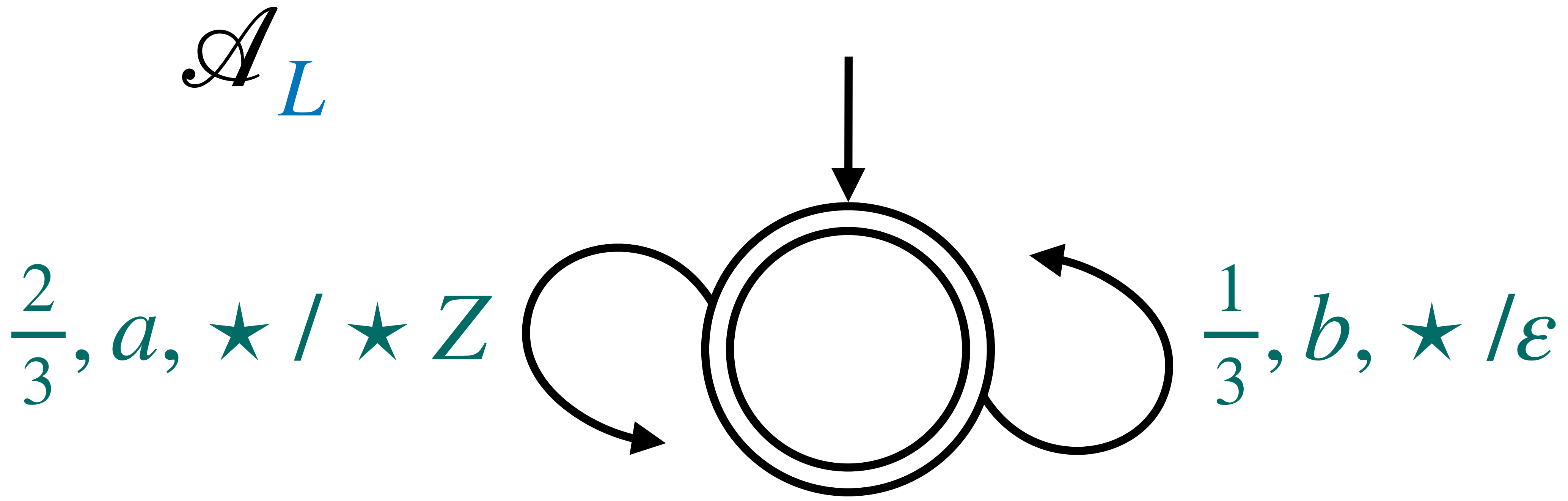
- Priority function $\Omega: States \to \mathbb{N}$ (like standard parity)
- Position i is a step of an ω-run of a PDA $\iff$ $\forall j \geq i:\ stackHeight(j) \geq stackHeight(i)$



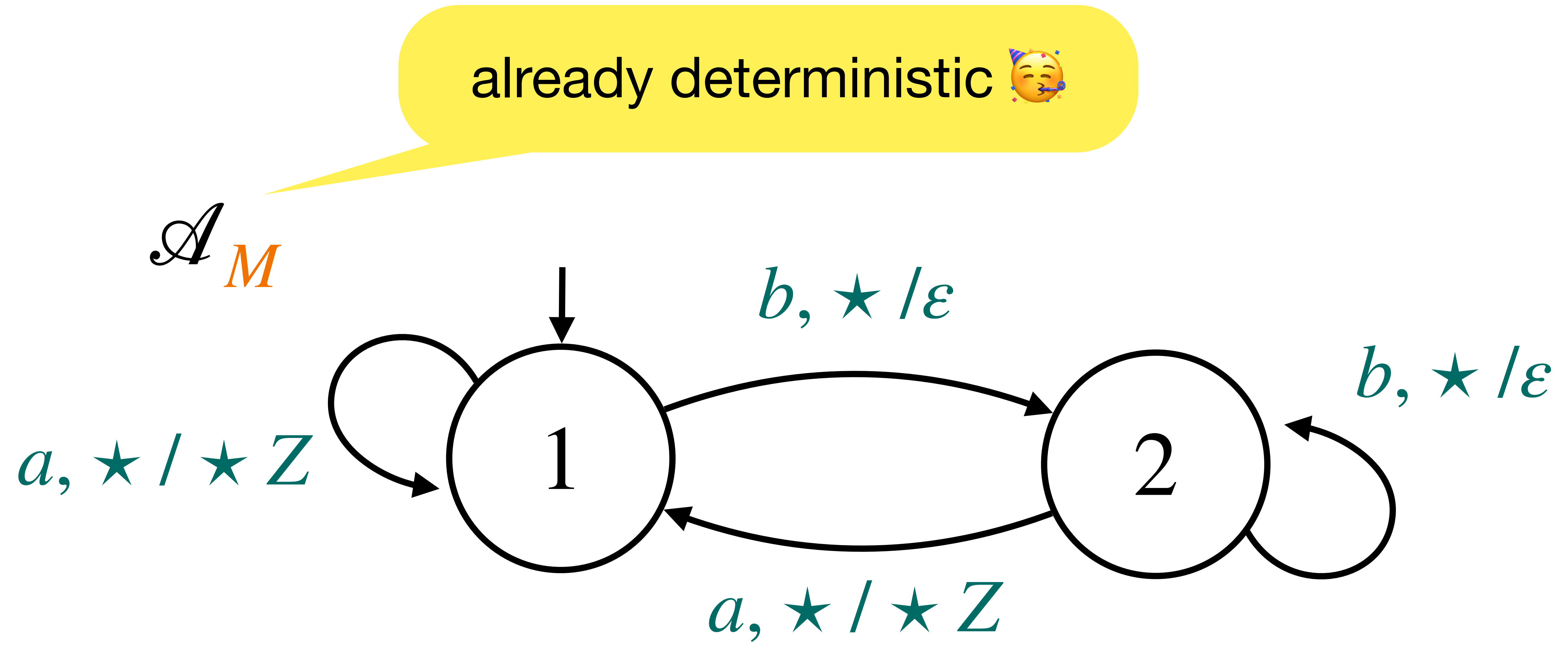
- Stair-parity = standard parity evaluated on sequence of steps

  (ω-run is accepting iff the minimum priority seen ∞-often at steps is even)

Example



$\mathscr{A}_L$
$\frac{2}{3}, a, \star / \star Z$
$\frac{1}{3}, b, \star / \varepsilon$

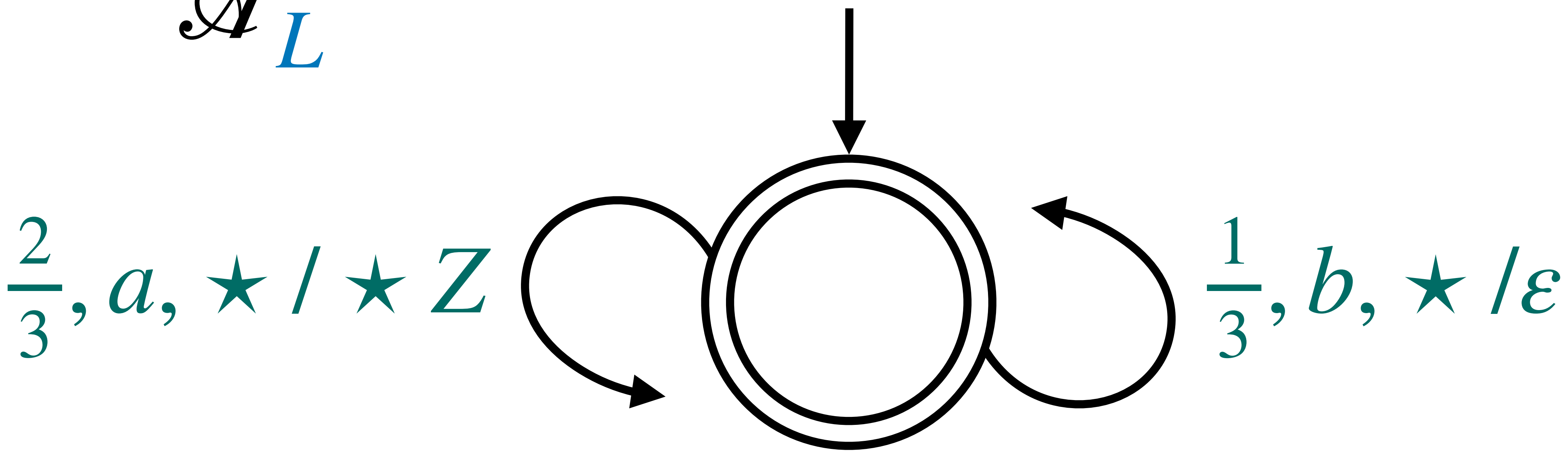


already deterministic 🥳
$\mathscr{A}_M$
$b, \star / \varepsilon$
$a, \star / \star Z$
1
2
$b, \star / \varepsilon$
$a, \star / \star Z$

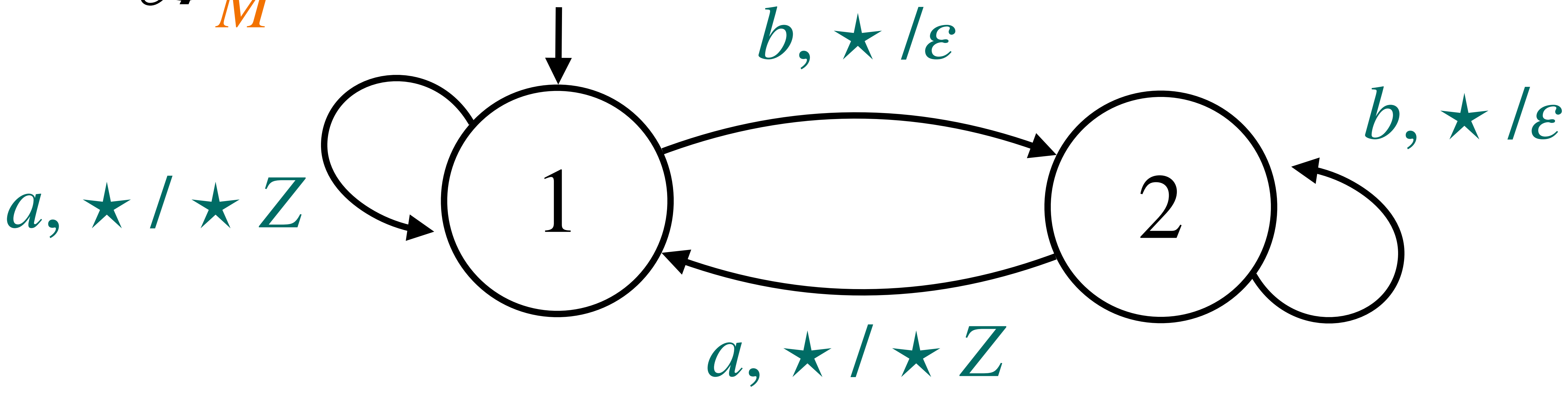
Example

already deterministic 🥳

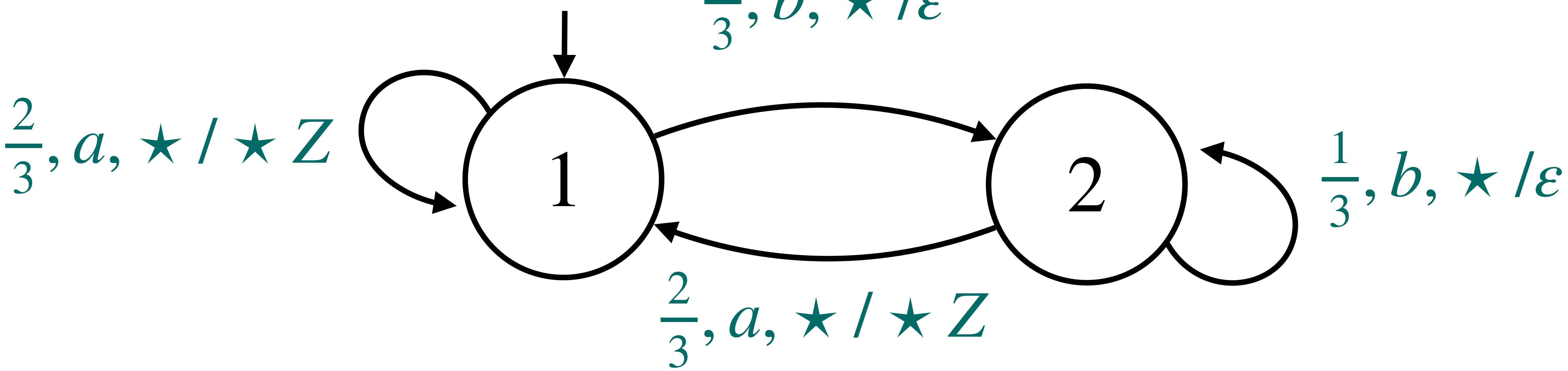
$\mathscr{A}_L$



$\mathscr{A}_M$

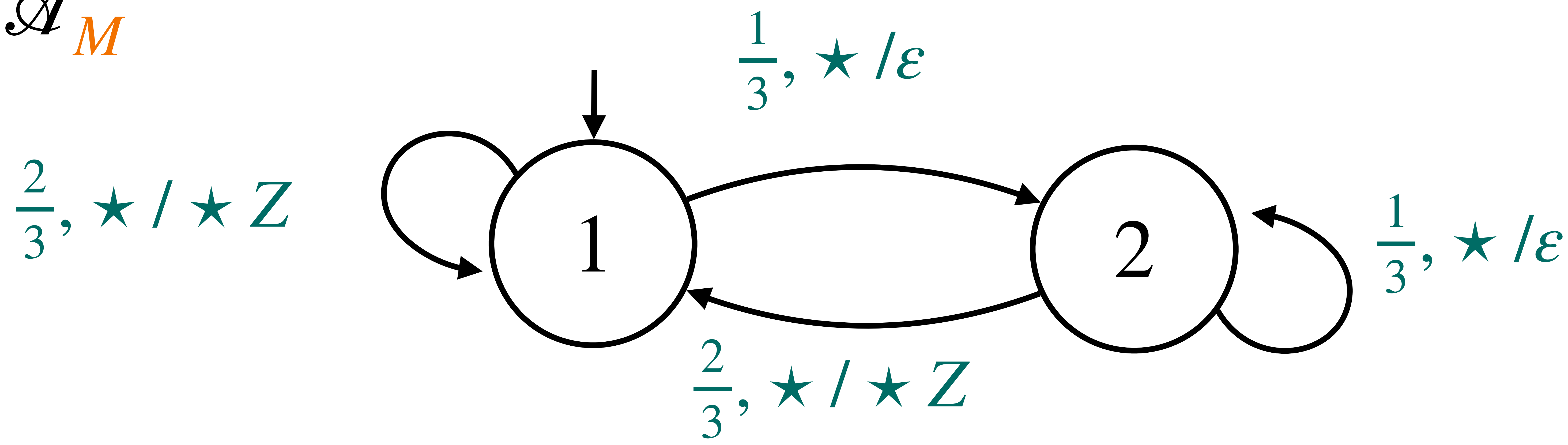


$\mathscr{A}_L \times \mathscr{A}_M$



Example cont'd

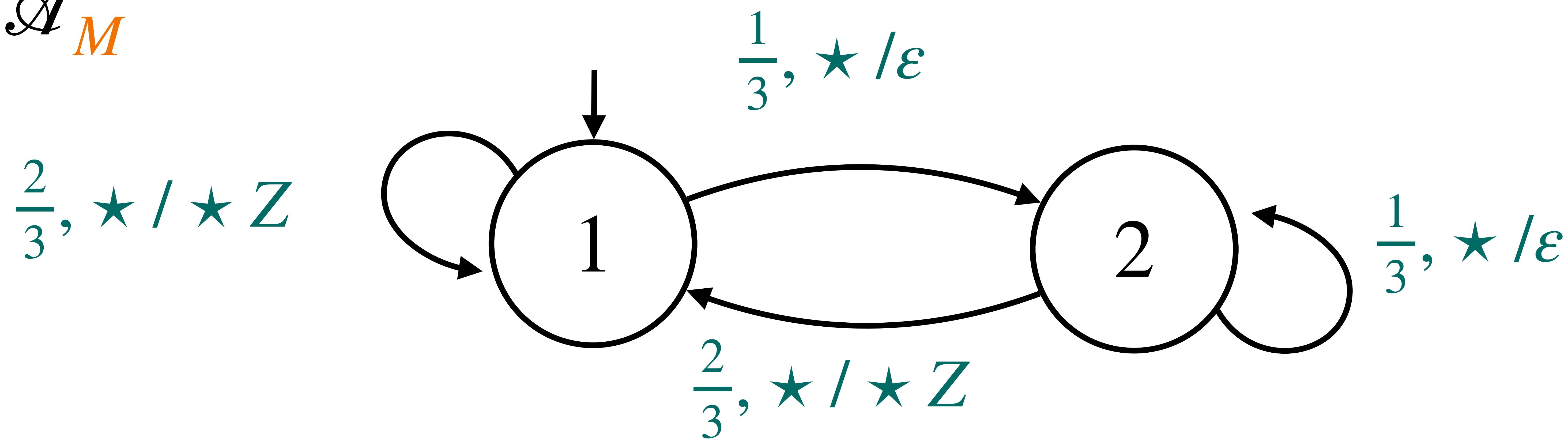
$\mathscr{A}_L \times \mathscr{A}_M$



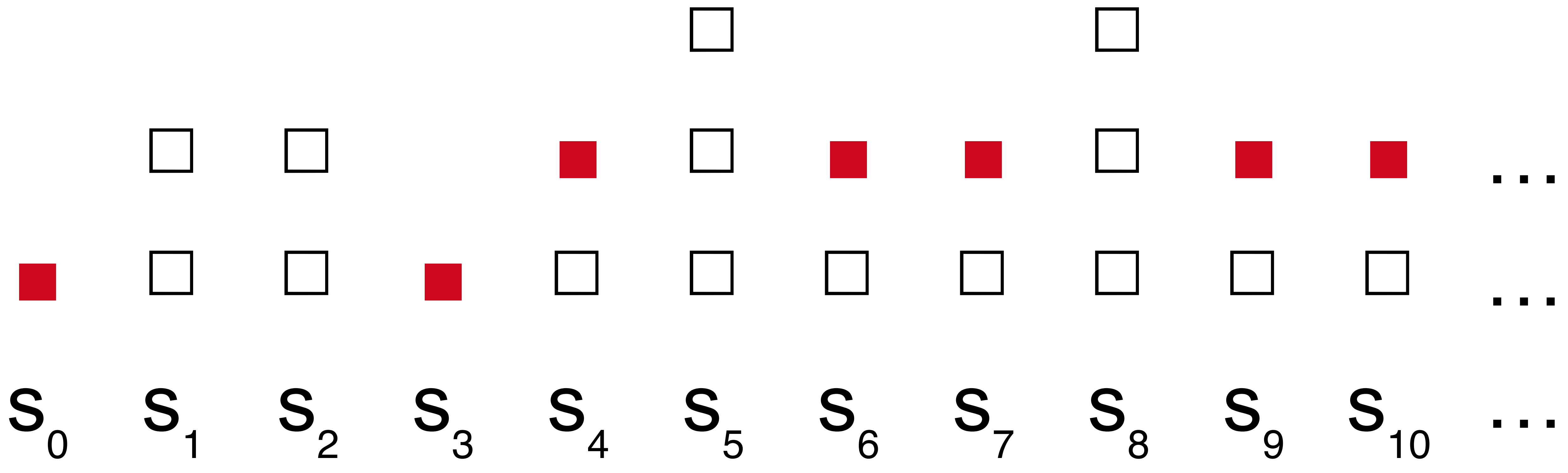
Example cont'd

- Probability that $\mathscr{A}_L \times \mathscr{A}_M$ generates a run it accepts (with stair-parity)?

$\mathscr{A}_L \times \mathscr{A}_M$

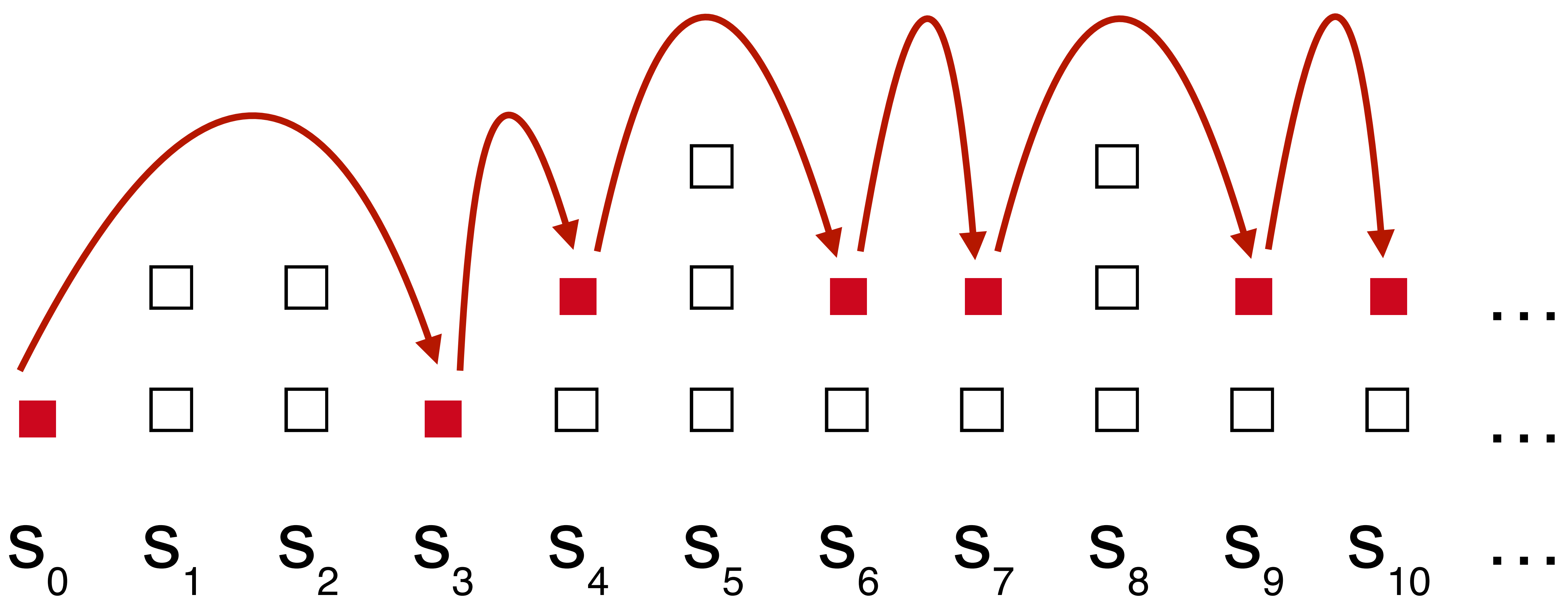


Remember: Stair-Parity Acceptance



- Stair-parity = standard parity evaluated on sequence of steps
  (ω-run is accepting iff the minimum priority seen ∞-often at steps is even)

Remember: Stair-Parity Acceptance

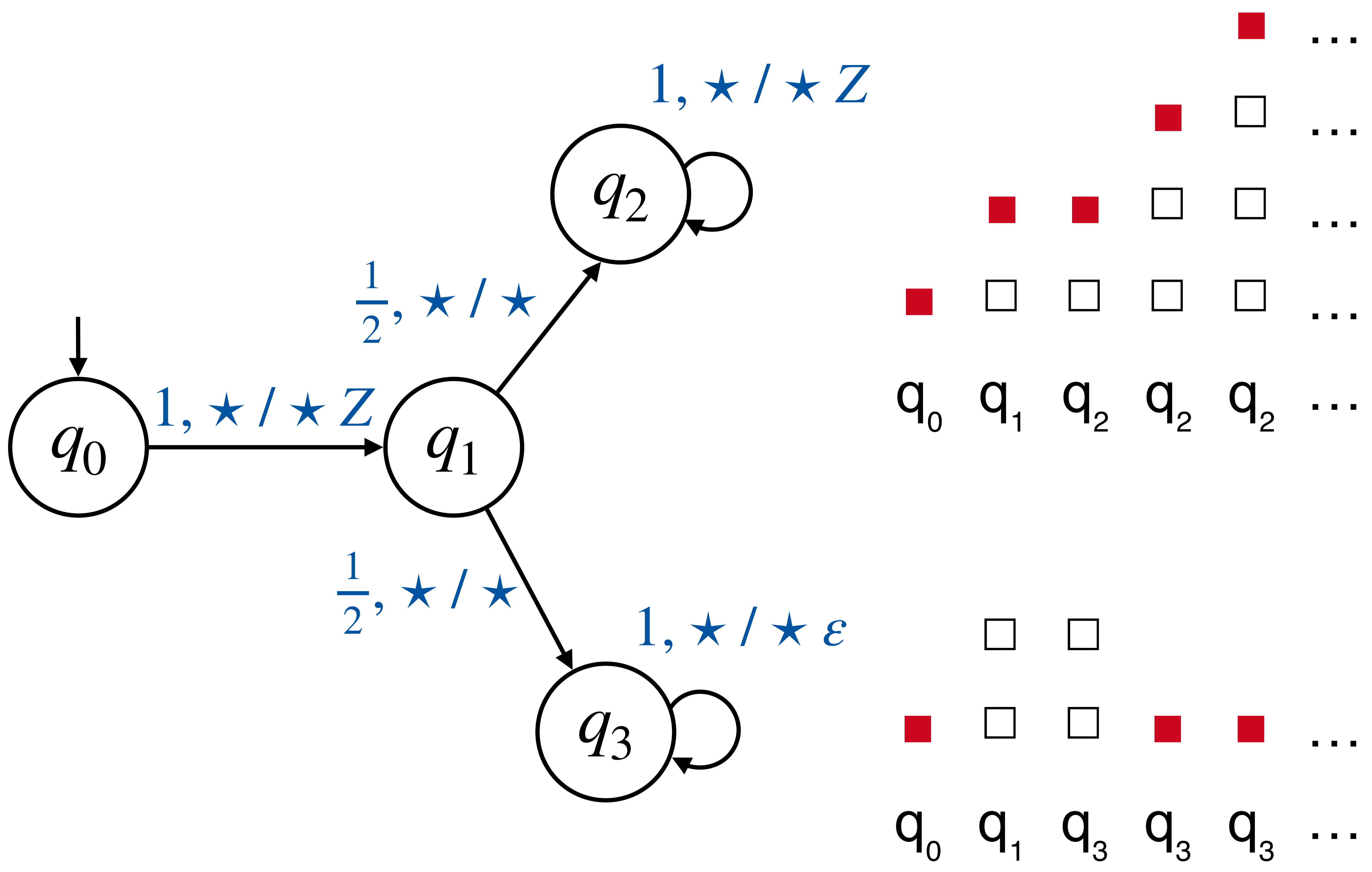


- Stair-parity = standard parity evaluated on sequence of steps

  (ω-run is accepting iff the minimum priority seen ∞-often at steps is even)

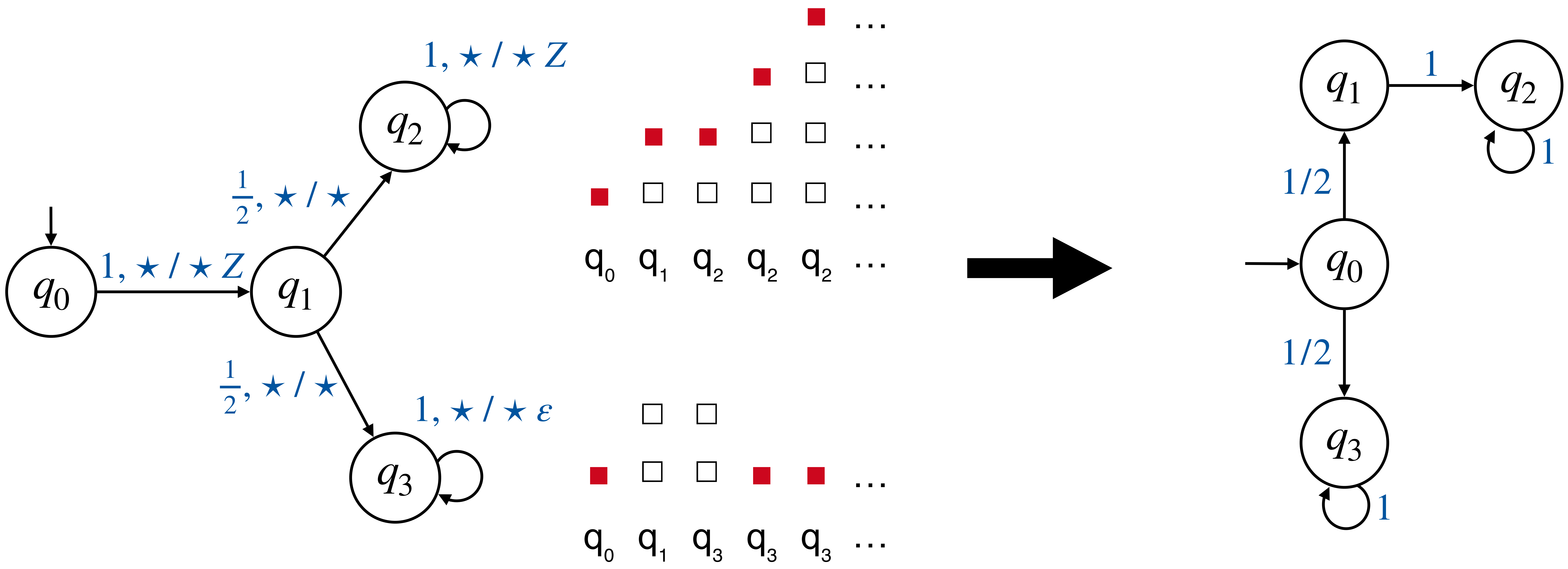
The Markov Chain of Steps: Easy Example

[Esparza, Kucera, Mayr '04]



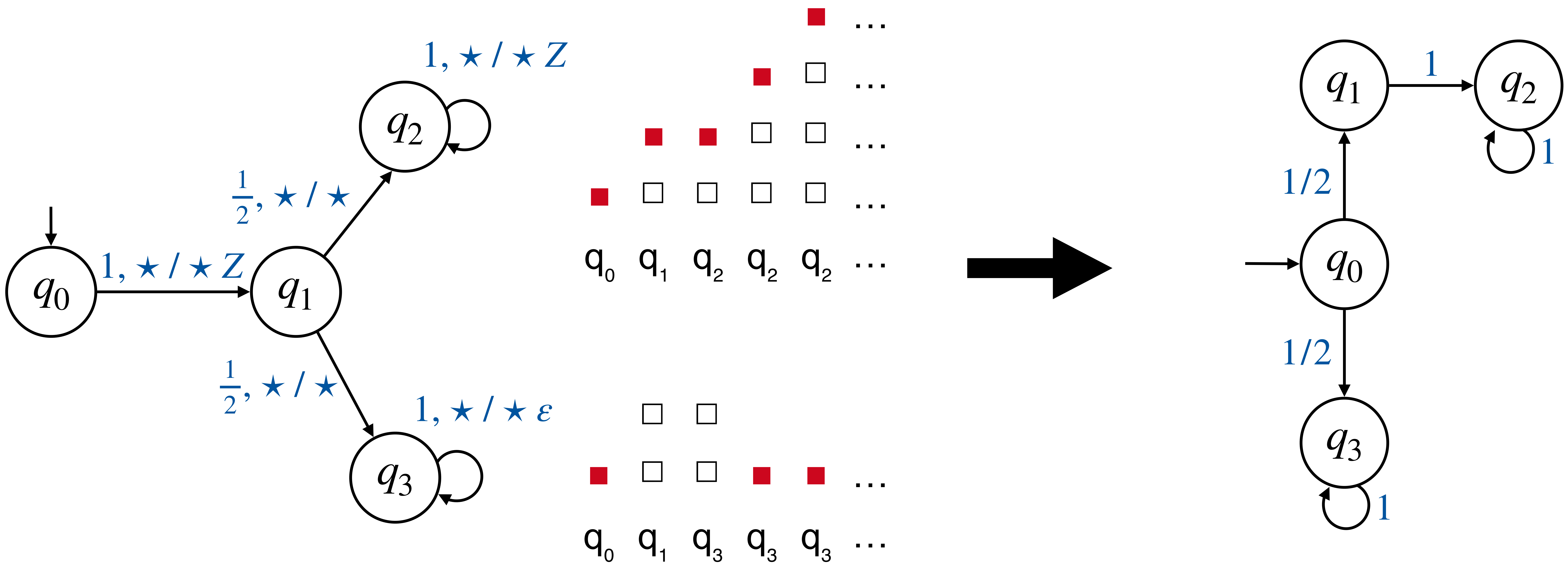
The Markov Chain of Steps: Easy Example

[Esparza, Kucera, Mayr '04]



The Markov Chain of Steps: Easy Example

[Esparza, Kucera, Mayr '04]



- Transition probs of step Markov chain may be irrational, but are expressible in $\exists\mathbb{R}$

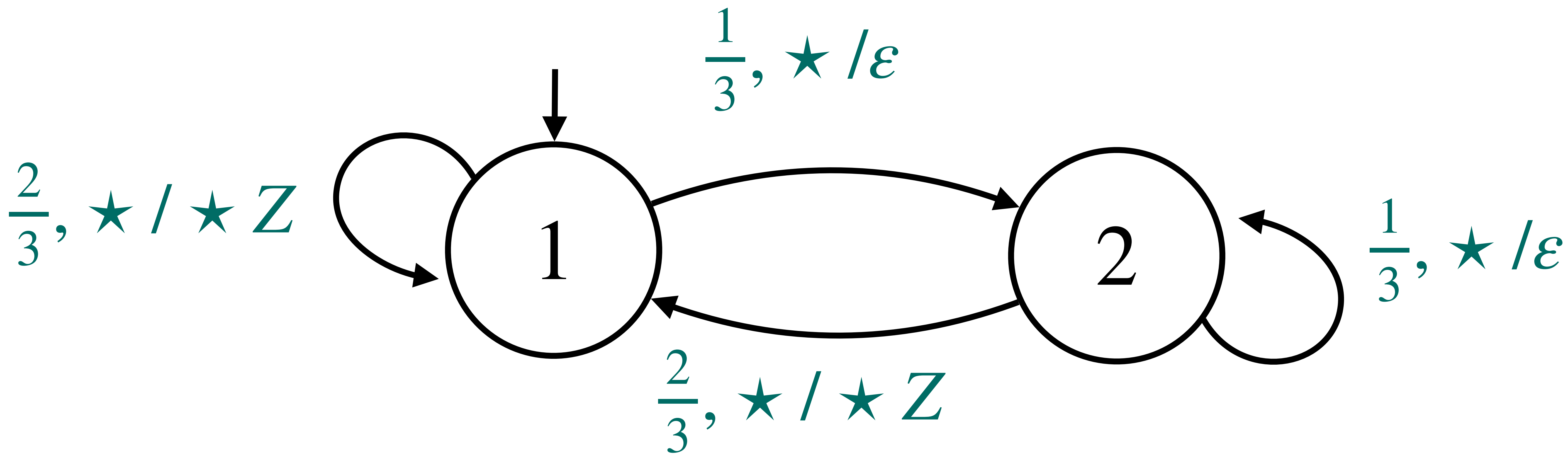
General Formulas for Step Markov Chain

Don't read this

	$q \to r$	$q\bot \to r$	$q\bot \to r\bot$	$q \to r\bot$
$q \in Q_{\text{call}}$	$\frac{[r\uparrow]}{[q\uparrow]}\left(\sum_{r',Z} P_{\text{call}}(q, r'Z)[r'Z{\downarrow}r] + \sum_{Z} P_{\text{call}}(q, rZ)\right)$	$\sum_{Z} P_{\text{call}}(q, rZ)[r\uparrow]$	$\sum_{r',Z} P_{\text{call}}(q, r'Z)[r'Z{\downarrow}r]$	0
$q \in Q_{\text{int}}$	$\frac{[r\uparrow]}{[q\uparrow]} P_{\text{int}}(q, r)$	0	$P_{\text{int}}(q, r)$	0
$q \in Q_{\text{ret}}$	undef.	0	$P_{\text{ret}}(q\bot, r)$	undef.

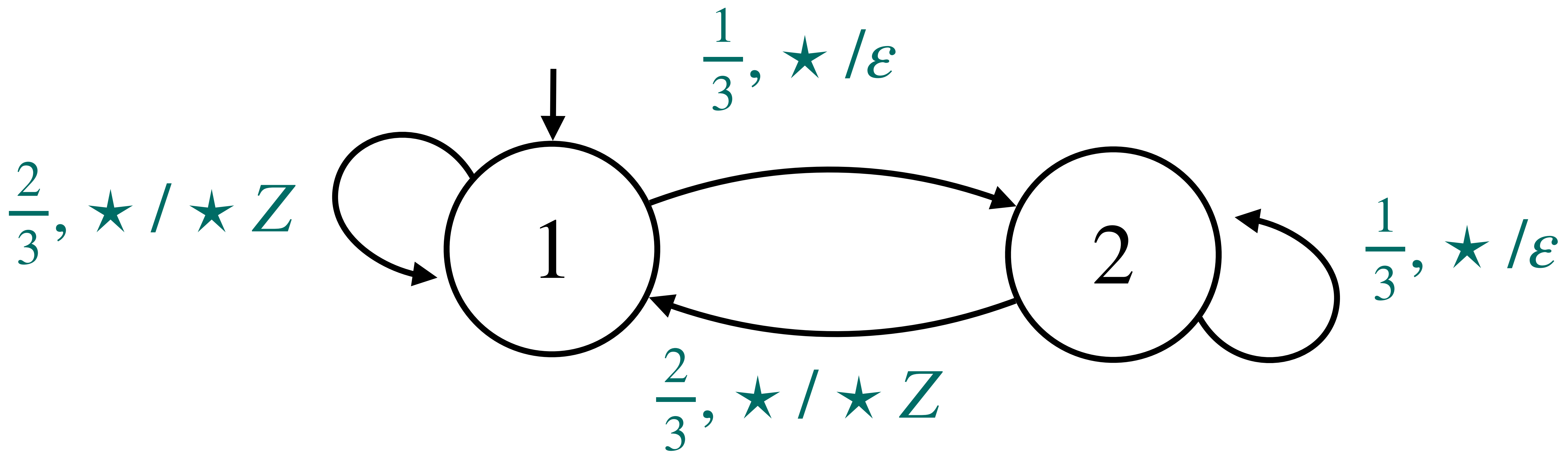
Example cont'd

$\mathscr{A}_L \times \mathscr{A}_M$

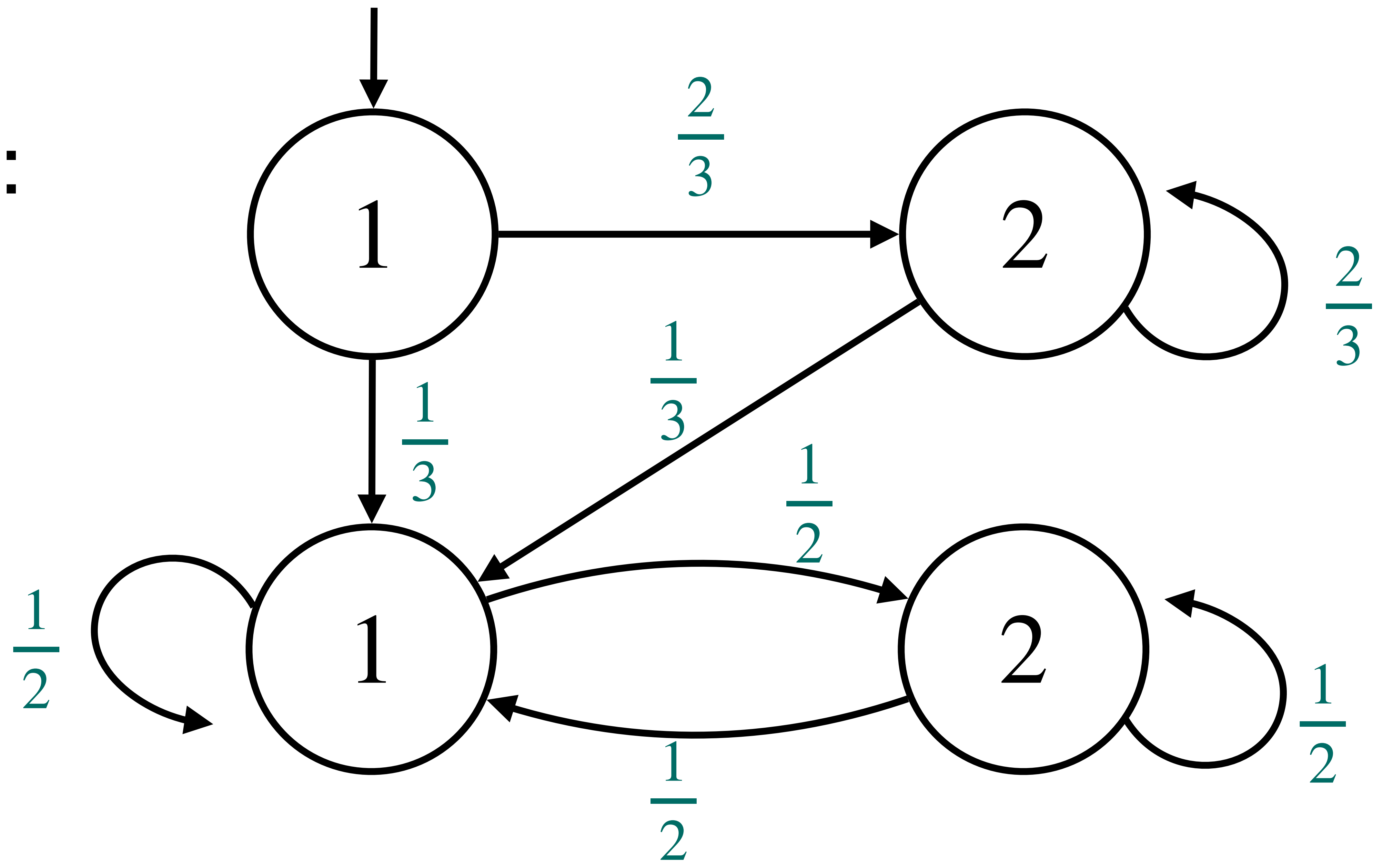


Example cont’d

$\mathscr{A}_L \times \mathscr{A}_M$

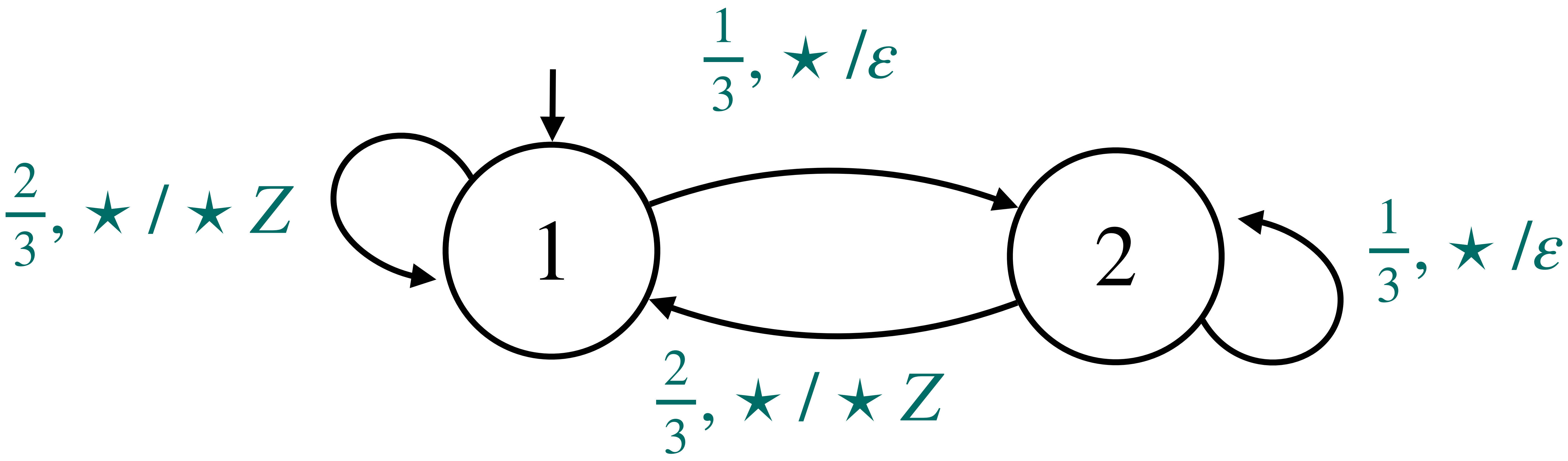


Markov chain of steps:

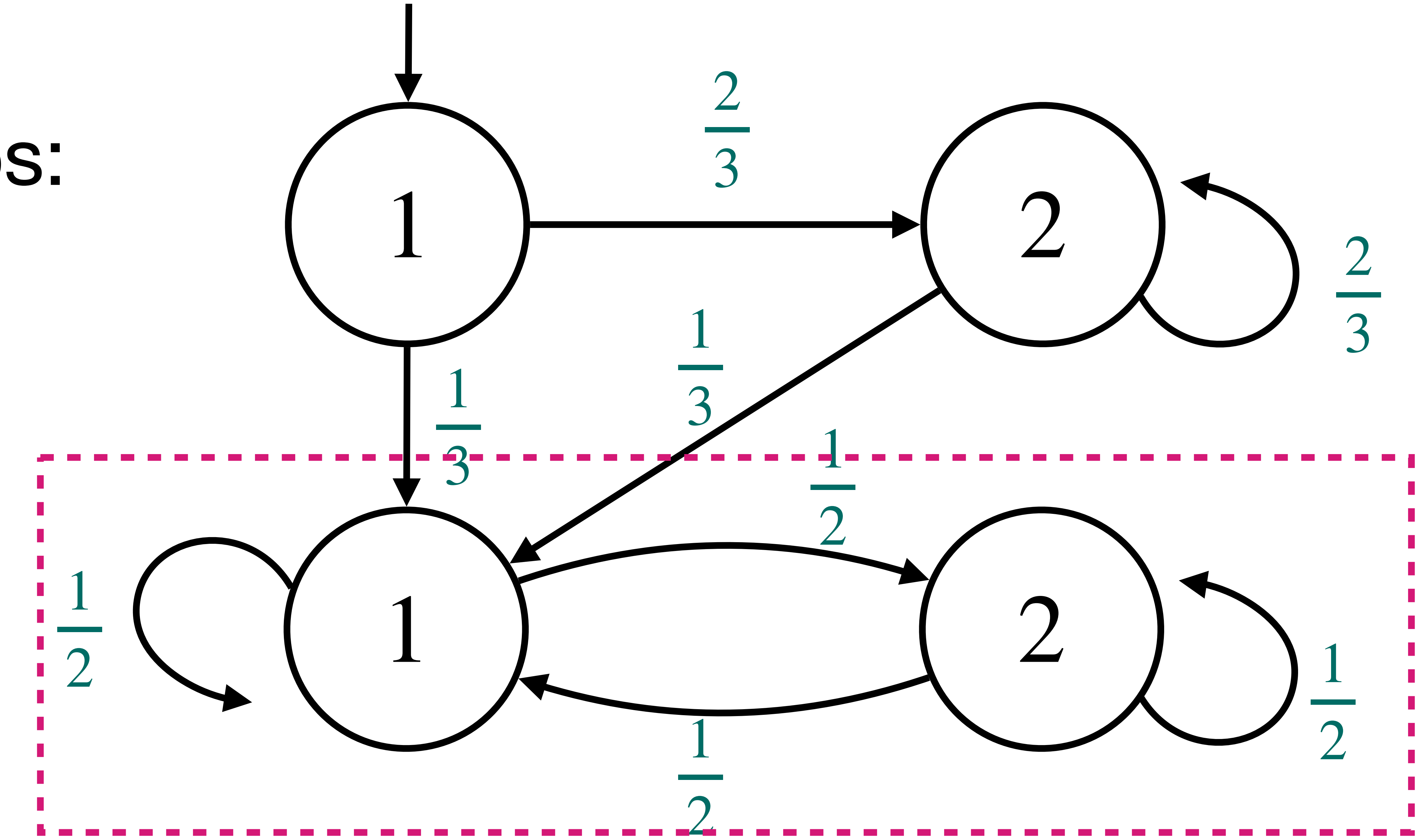


Example cont'd

$\mathscr{A}_L \times \mathscr{A}_M$



Markov chain of steps:



the only BSCC violates standard parity

$\Longrightarrow Pr_{w \sim L}(w \in M) = 0$

Two Birds, One Stone

With the step Markov chain construction we

- … got rid of the stack
- … reduced stair-parity to standard parity

Two Birds, One Stone

With the step Markov chain construction we

- … got rid of the stack
- … reduced stair-parity to standard parity

It follows: For a pVPA $\mathscr{A}_L$ and a Büchi VPA $\mathscr{A}_M$

Two Birds, One Stone

With the step Markov chain construction we

- … got rid of the stack
- … reduced stair-parity to standard parity

It follows: For a pVPA $\mathscr{A}_L$ and a Büchi VPA $\mathscr{A}_M$

- … deciding $Pr_{w \sim L}(w \in M) = 1$ is $EXPTIME$-complete

Two Birds, One Stone

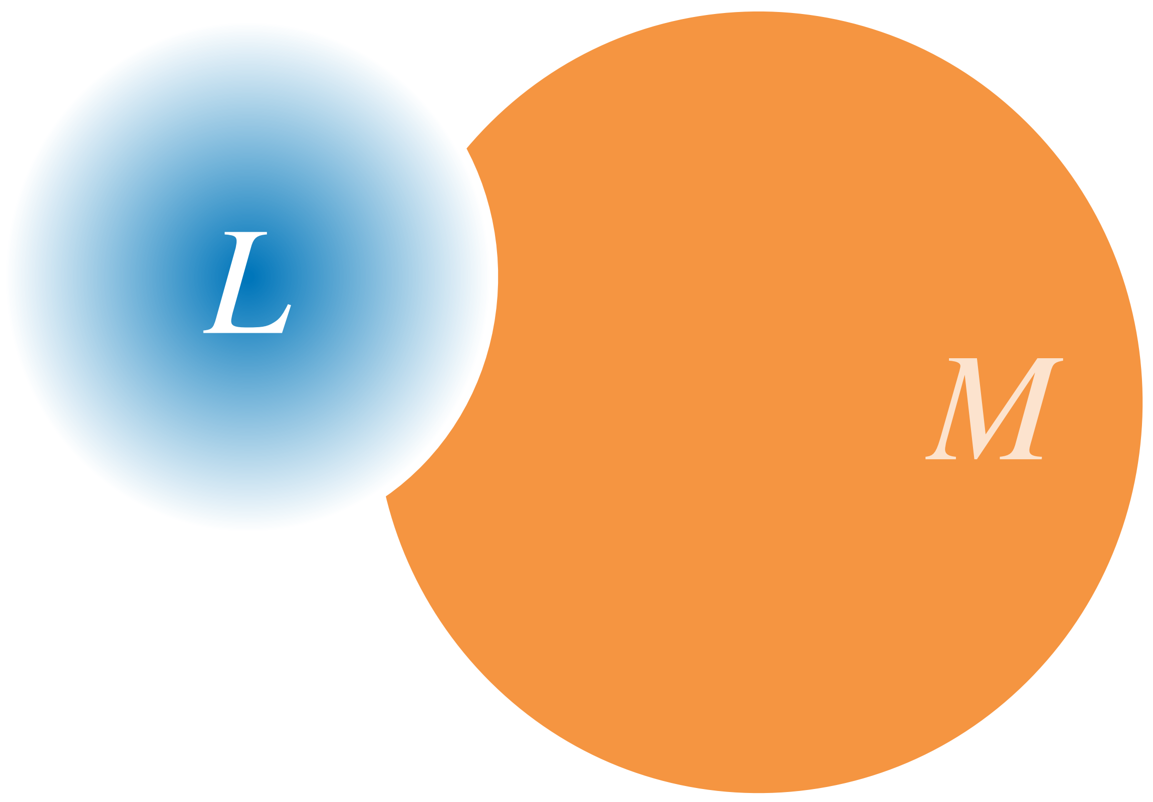
With the step Markov chain construction we

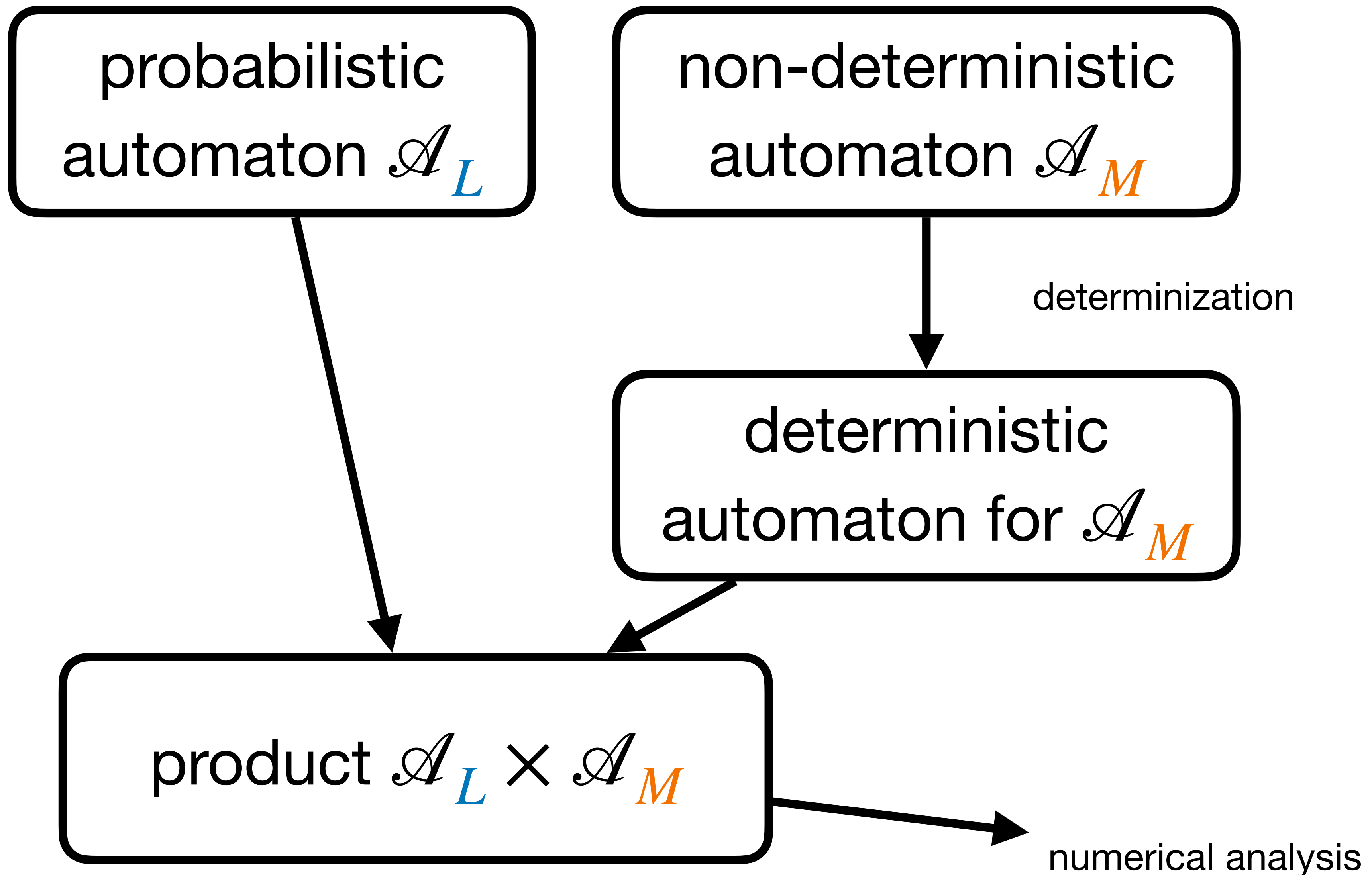
- … got rid of the stack
- … reduced stair-parity to standard parity

It follows: For a pVPA $\mathscr{A}_L$ and a Büchi VPA $\mathscr{A}_M$

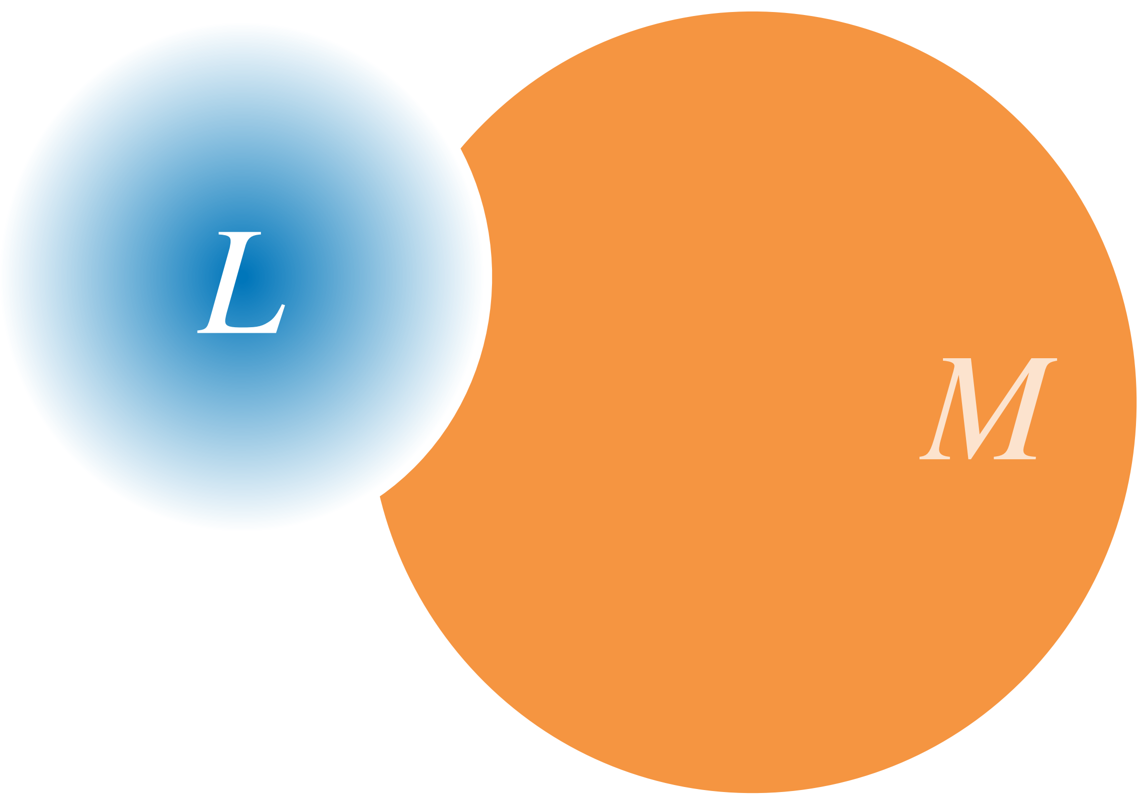
- … deciding $Pr_{w \sim L}(w \in M) = 1$ is $EXPTIME$-complete
- … deciding $Pr_{w \sim L}(w \in M) \gtreqless \lambda$ is in $EXPSPACE$

Summary & Outlook

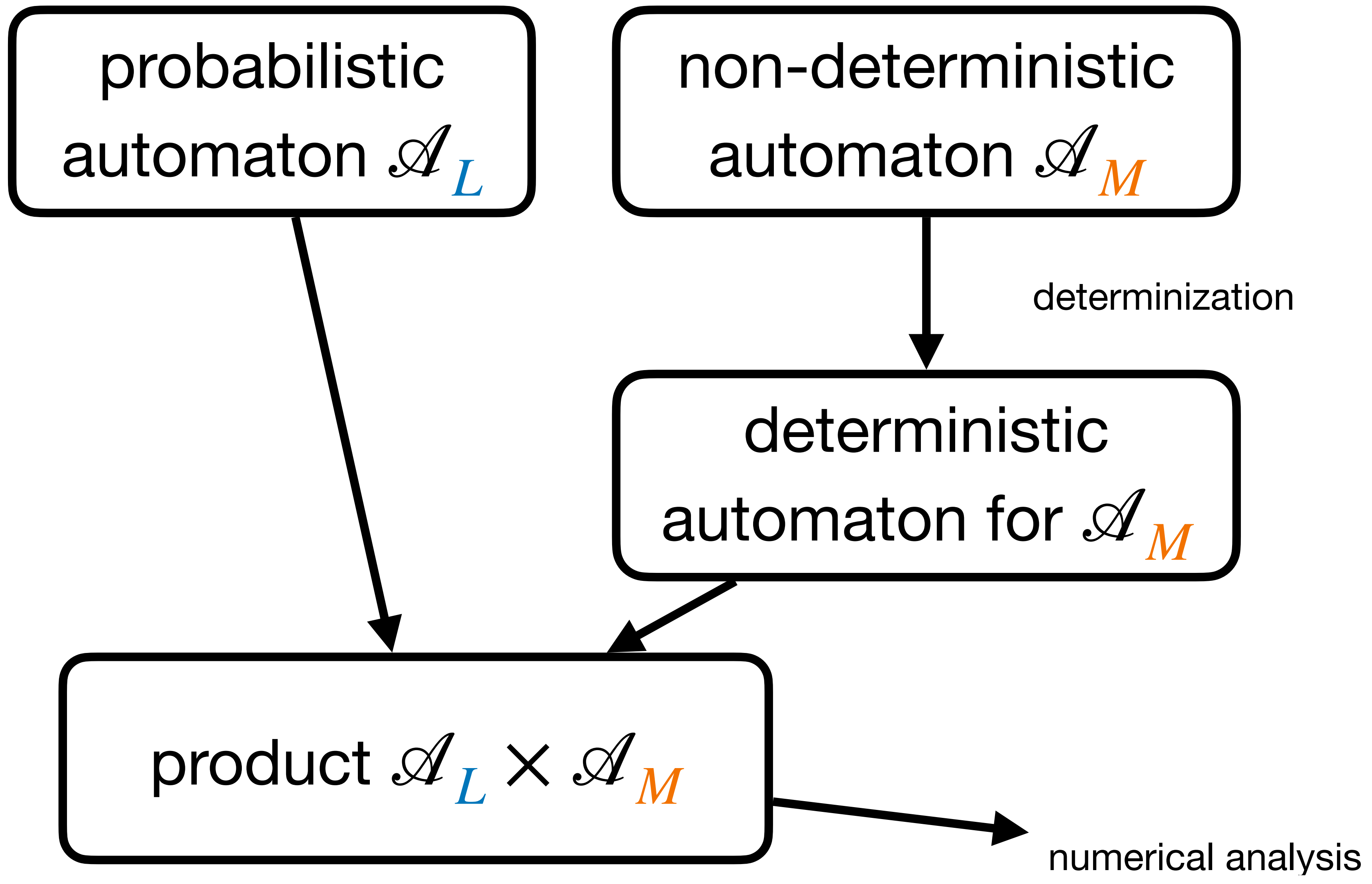




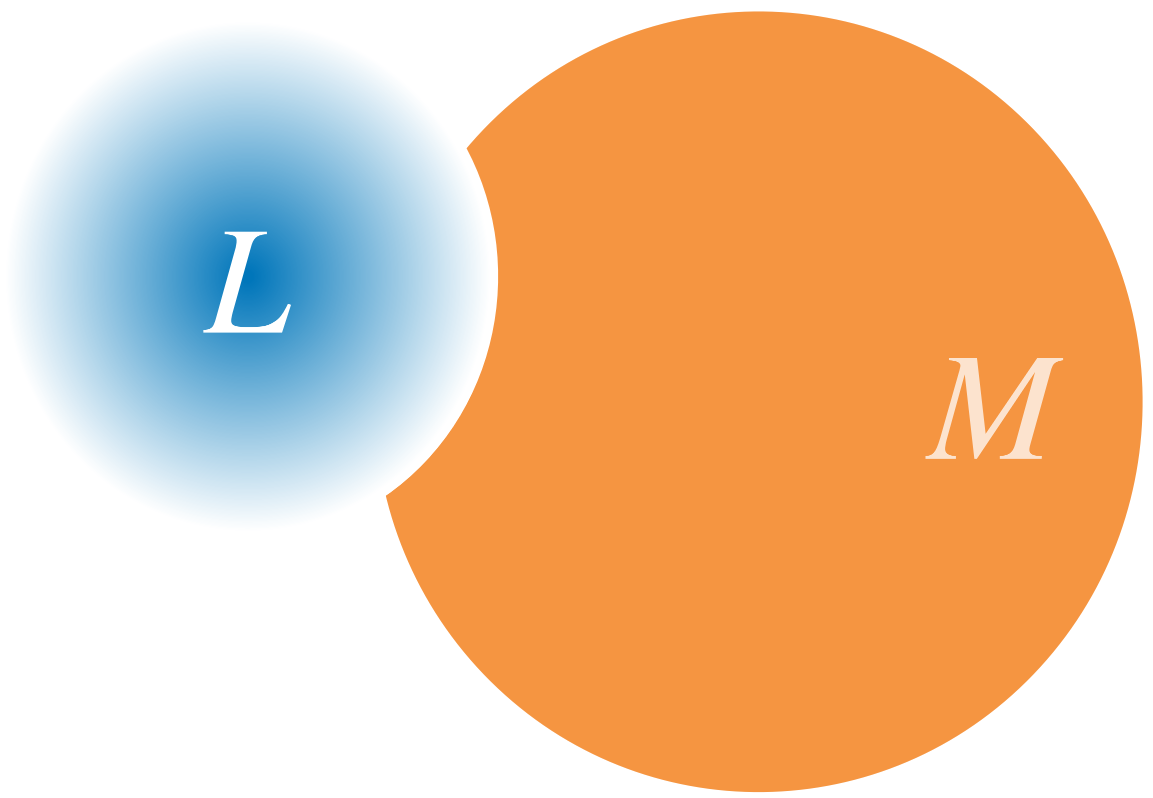
Summary & Outlook



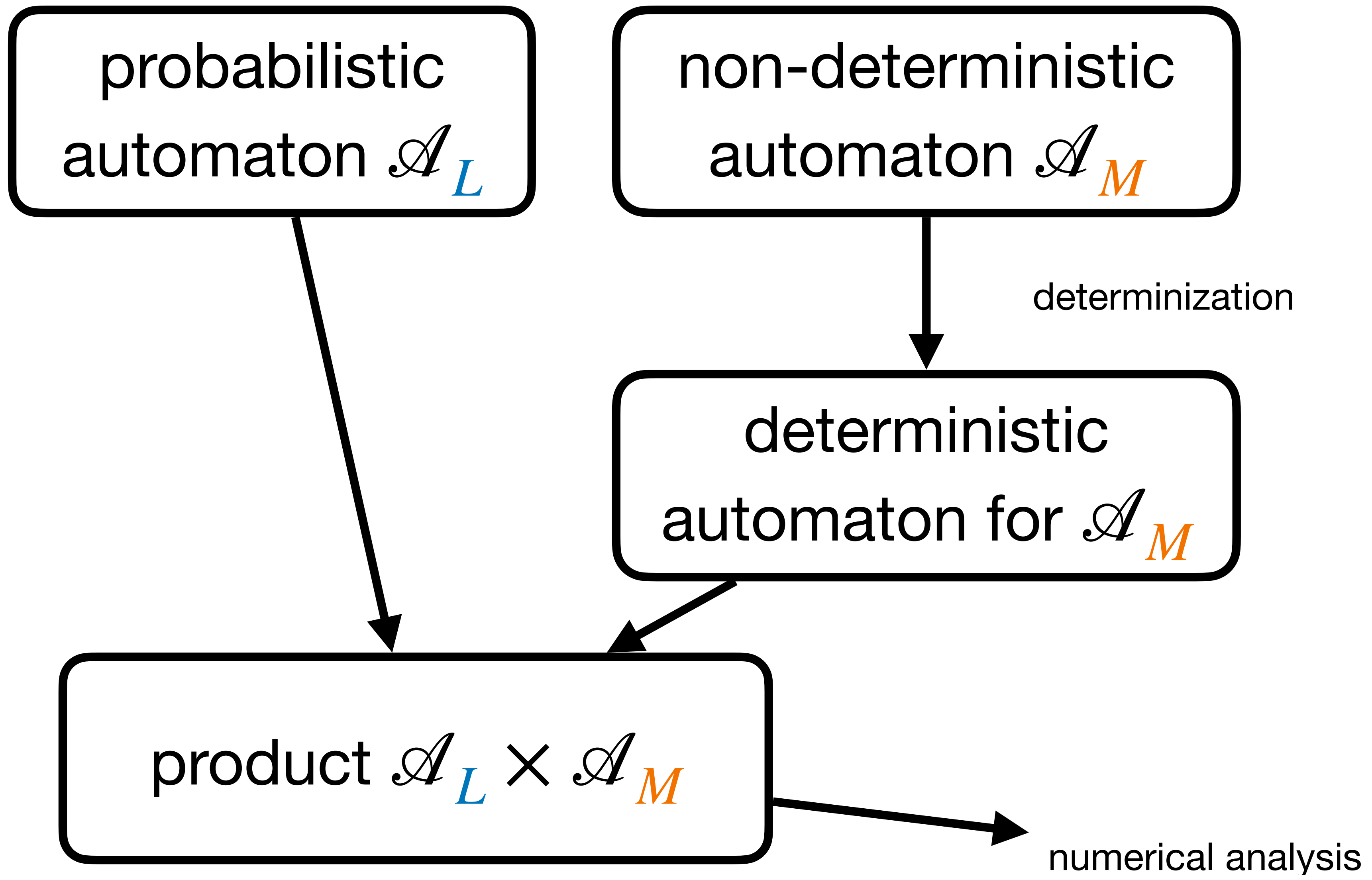
- **In this talk:** General approach for probabilistic language inclusion + concrete case of ωVPL



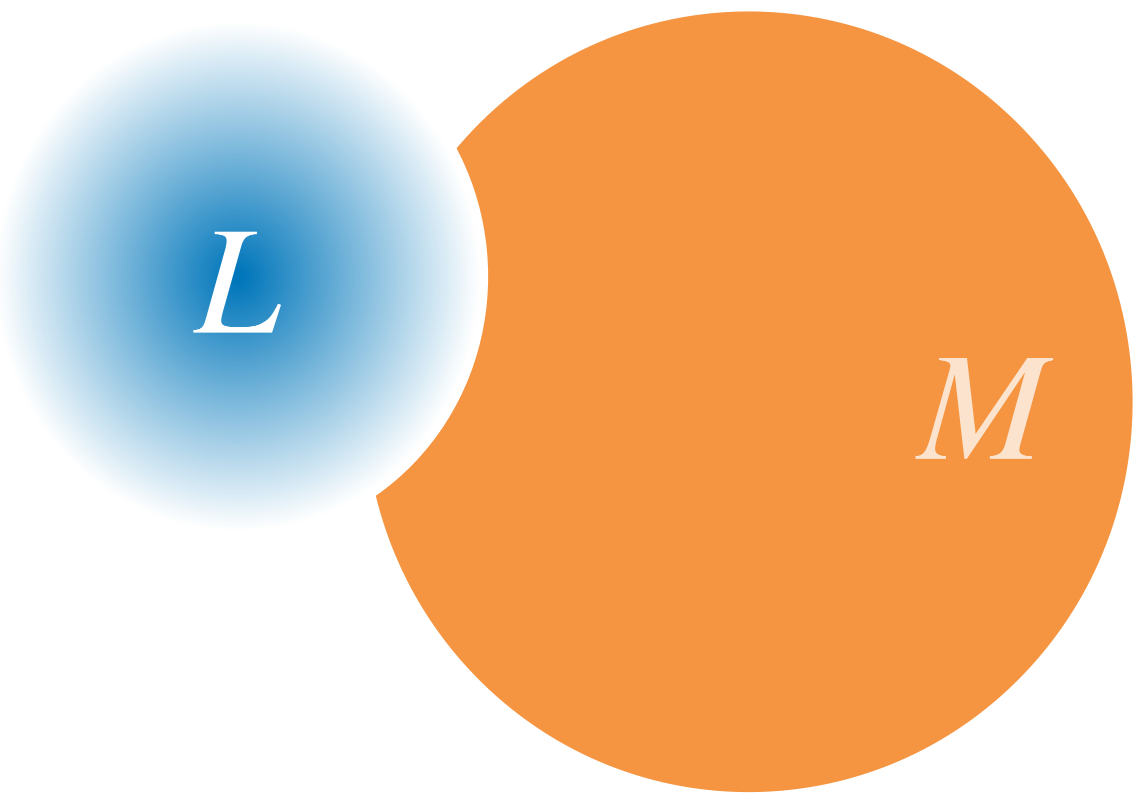
Summary & Outlook



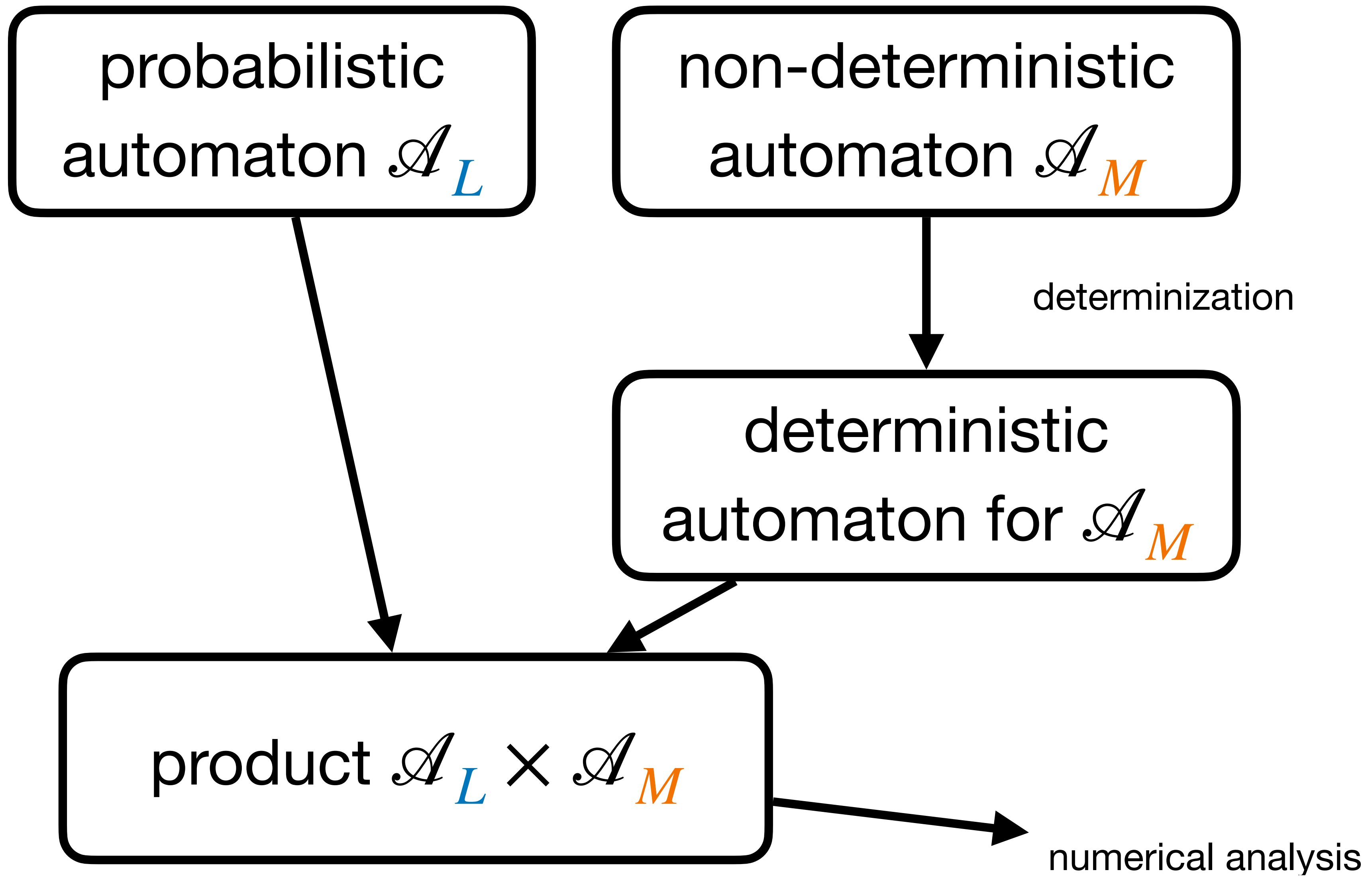
- **In this talk:** General approach for probabilistic language inclusion + concrete case of ωVPL
- **Main technique:** Reduce problems to limiting behaviour of finite Markov chains



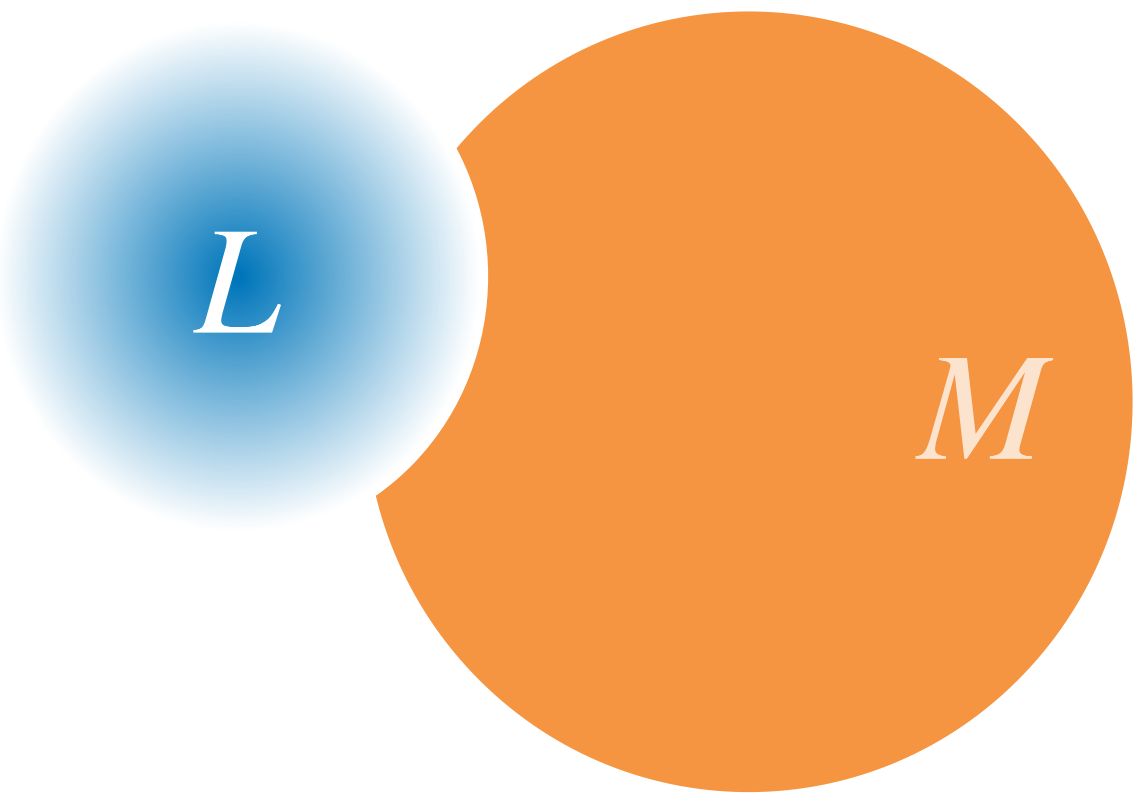
Summary & Outlook



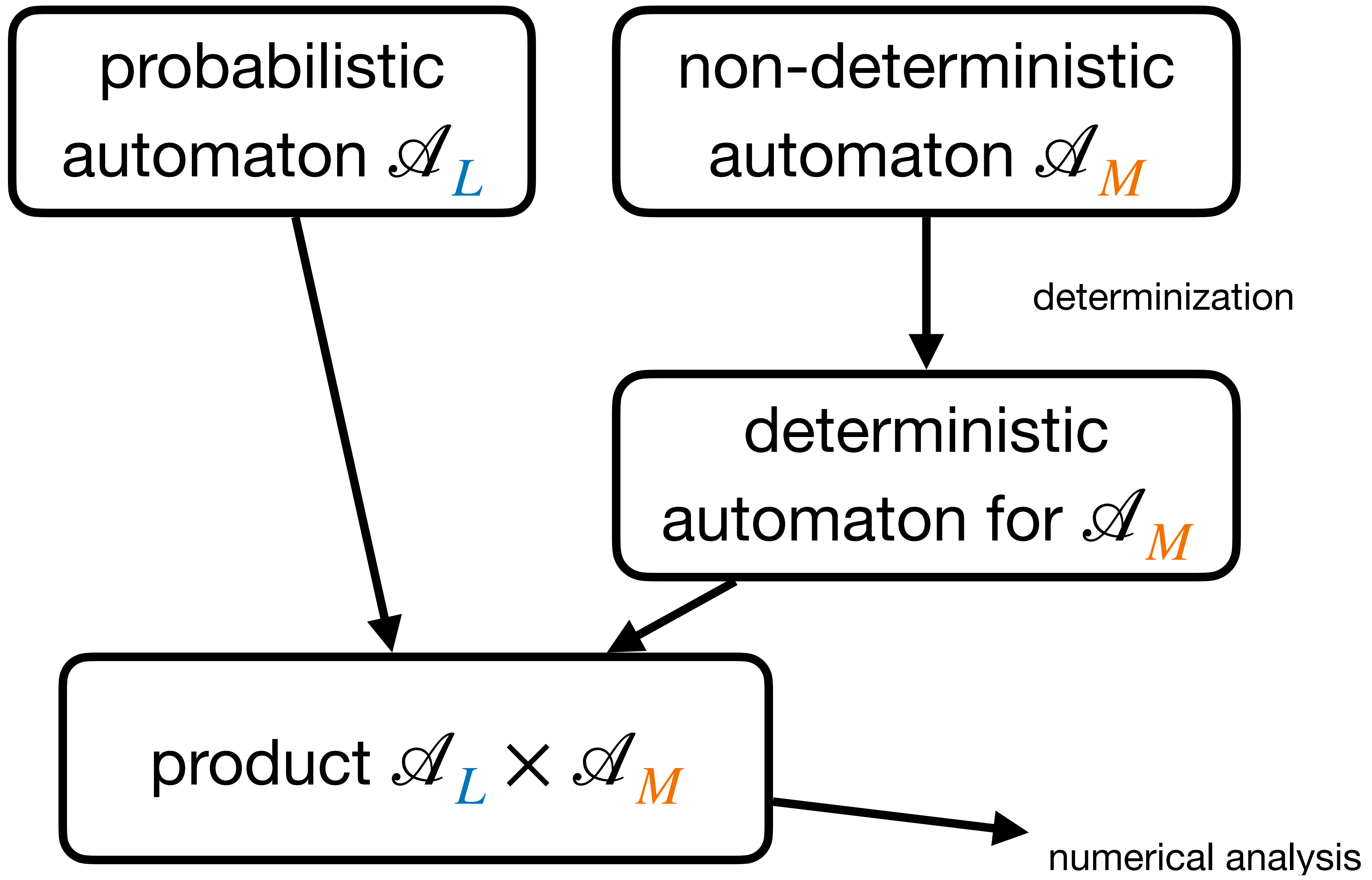
- **In this talk:** General approach for probabilistic language inclusion + concrete case of ωVPL
- **Main technique:** Reduce problems to limiting behaviour of finite Markov chains
- **Complexity bottleneck:** Determinization



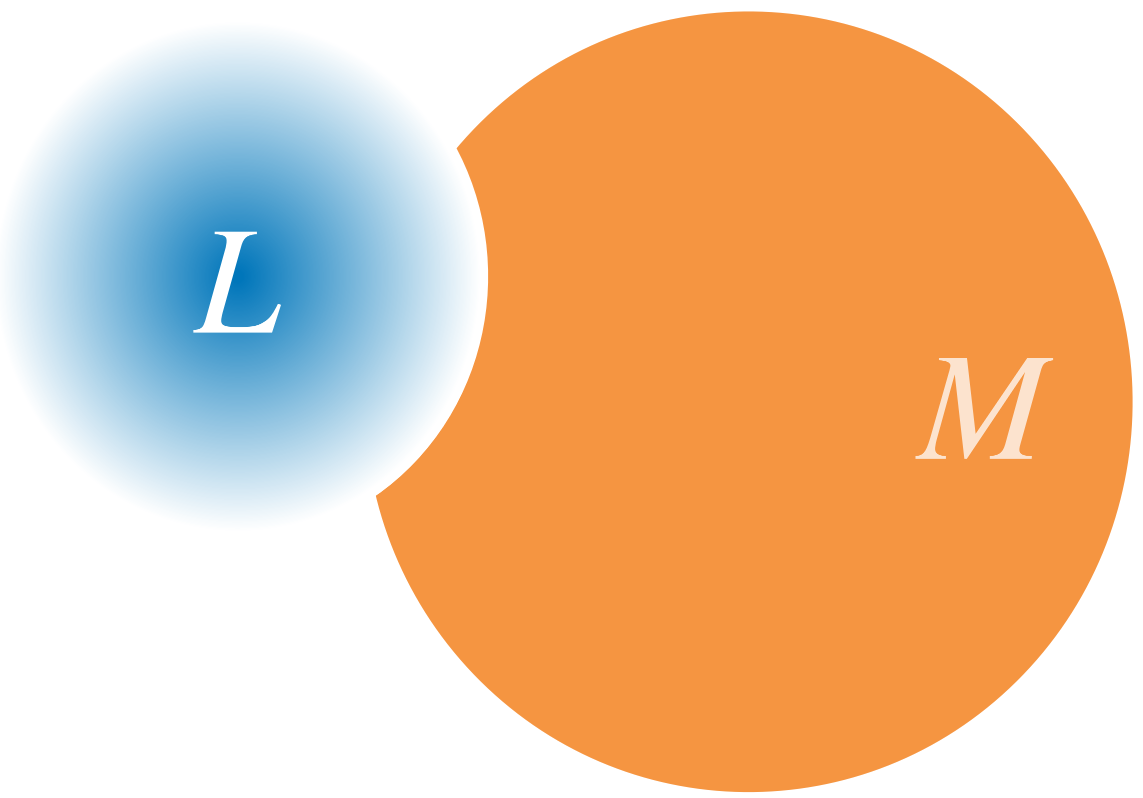
Summary & Outlook



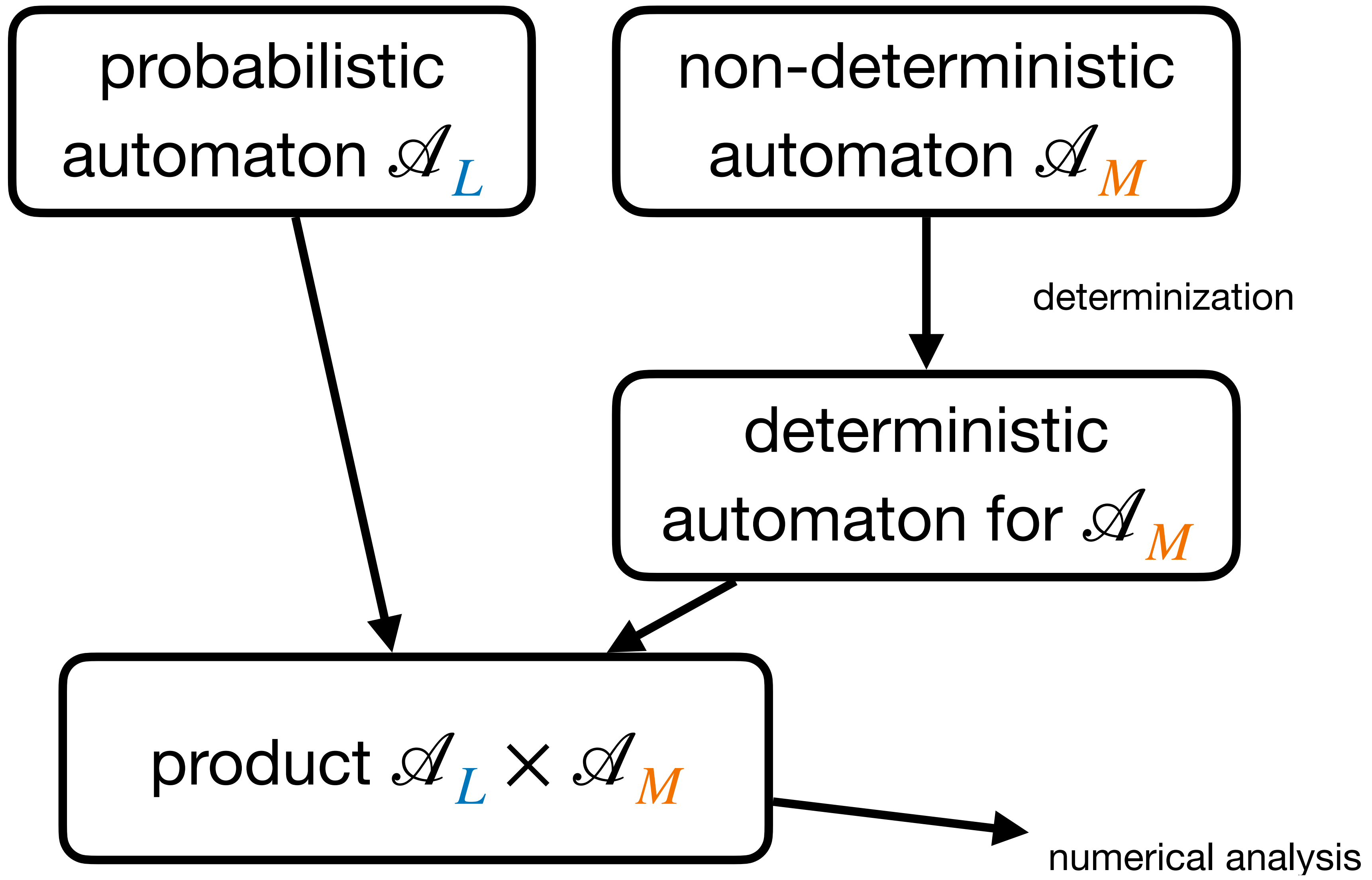
- **In this talk:** General approach for probabilistic language inclusion + concrete case of ωVPL
- **Main technique:** Reduce problems to limiting behaviour of finite Markov chains
- **Complexity bottleneck:** Determinization
- **What's next?** Unambiguous instead of deterministic, probabilistic automaton for $\mathscr{A}_M$



Summary & Outlook



- **In this talk:** General approach for probabilistic language inclusion + concrete case of ωVPL
- **Main technique:** Reduce problems to limiting behaviour of finite Markov chains
- **Complexity bottleneck:** Determinization
- **What's next?** Unambiguous instead of deterministic, probabilistic automaton for $\mathscr{A}_M$



Thank you for listening!

Code for probabilistic robot

```
dtmc

const int N = 4;

module probot
  x : [0..N] init 2;
  y : [0..N] init 2;

  [] x=0 & y=0 -> 0.5 : (x'=x+1) + 0.5 : (y'=y+1);
  [] x=0 & y=N -> 0.5 : (x'=x+1) + 0.5 : (y'=y-1);
  [] x=N & y=0 -> 0.5 : (x'=x-1) + 0.5 : (y'=y+1);
  [] x=N & y=N -> 0.5 : (x'=x-1) + 0.5 : (y'=y-1);

  [] x=0 & y>0 & y<N -> 1/3: (x'=x+1) + 1/3 : (y'=y-1) + 1/3 : (y'=y+1);
  [] x=N & y>0 & y<N -> 1/3: (x'=x-1) + 1/3 : (y'=y-1) + 1/3 : (y'=y+1);
  [] y=0 & x>0 & x<N -> 1/3: (y'=y+1) + 1/3 : (x'=x-1) + 1/3 : (x'=x+1);
  [] y=N & x>0 & x<N -> 1/3: (y'=y-1) + 1/3 : (x'=x-1) + 1/3 : (x'=x+1);

  [] x>0 & x<N & y>0 & y<N -> 0.25 : (x'=x+1) + 0.25 : (x'=x-1) + 0.25 : (y'=y+1) + 0.25 : (y'=y-1);

endmodule

label "treasure" = x=4 & y=1;
label "safe" = !(x=0 & y=N);
```