Lecture *Autamata, Languages, and Computability*

Finite State Automata

Dr. Pascal Bercher

Institute of Artificial Intelligence,
Ulm University, Germany

January 15th, 2019

ulm university universität
uulm

Prerequisites

Expected prior knowledge for *lecture*:

- Proof techniques (such as induction or contradiction)
- Set theory
- Recursive definitions
- Trees and graphs

Prerequisites

Expected prior knowledge for *lecture*:

- Proof techniques (such as induction or contradiction)
- Set theory
- Recursive definitions
- Trees and graphs

Teached in previous *classes*:

- Formal languages
- Formal grammars

Prerequisites

Expected prior knowledge for *lecture*:

- Proof techniques (such as induction or contradiction)
- Set theory
- Recursive definitions
- Trees and graphs

Teached in previous *classes*:

- Formal languages
- Formal grammars

} Short recap will be given

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet.

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet. More formally:

- Σ is the *alphabet*, i.e., a finite set of the available symbols (or letters)

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet. More formally:

- Σ is the *alphabet*, i.e., a finite set of the available symbols (or letters)
  - Σ^* denotes the set of all words over Σ, i.e.,
    $\Sigma^* = \{\omega_1 \dots \omega_n \mid \text{for all } 1 \leq i \leq n \text{ holds } \omega_i \in \Sigma\}$

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet. More formally:

- Σ is the *alphabet*, i.e., a finite set of the available symbols (or letters)
  - Σ^* denotes the set of all words over Σ, i.e., $\Sigma^* = \{\omega_1 \dots \omega_n \mid \text{for all } 1 \leq i \leq n \text{ holds } \omega_i \in \Sigma\}$
  - Σ^+ denotes the set of all words over Σ with at least one symbol, i.e., $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet. More formally:

- Σ is the *alphabet*, i.e., a finite set of the available symbols (or letters)
  - Σ^* denotes the set of all words over Σ, i.e., $\Sigma^* = \{\omega_1 \dots \omega_n \mid \text{for all } 1 \leq i \leq n \text{ holds } \omega_i \in \Sigma\}$
  - Σ^+ denotes the set of all words over Σ with at least one symbol, i.e., $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$
- $\omega \in \Sigma^*$ is a *word*, i.e., a finite sequence of symbols

Basic Vocabulary

Formal languages are (possibly infinite) sets of *words* over a given alphabet. More formally:

- Σ is the *alphabet*, i.e., a finite set of the available symbols (or letters)
  - Σ^* denotes the set of all words over Σ, i.e., $\Sigma^* = \{\omega_1 \dots \omega_n \mid \text{for all } 1 \leq i \leq n \text{ holds } \omega_i \in \Sigma\}$
  - Σ^+ denotes the set of all words over Σ with at least one symbol, i.e., $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$
- $\omega \in \Sigma^*$ is a *word*, i.e., a finite sequence of symbols
- $L \subseteq \Sigma^*$ is a *(formal) language* over the alphabet Σ, i.e., some possibly infinite set of words

Basic Vocabulary – Examples

- Σ – the *alphabet*:

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010
  - HELLO WORLD (if the blank is a symbol, otherwise these are two words)

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010
  - HELLO WORLD (if the blank is a symbol, otherwise these are two words)
- L – a *language*:

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010
  - HELLO WORLD (if the blank is a symbol, otherwise these are two words)
- L – a *language*:
  - $L = \{0, 1, 01, 10, 11\}$, defined over $\{0, 1\}$

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010
  - HELLO WORLD (if the blank is a symbol, otherwise these are two words)
- L – a *language*:
  - $L = \{0, 1, 01, 10, 11\}$, defined over $\{0, 1\}$
  - $L' = \{a^n b^n \mid n \in \mathbb{N}\}$, defined over $\{a, b\}$ (n indicates the number of as/bs)

Basic Vocabulary – Examples

- Σ – the *alphabet*:
  - $\{0, 1\}$
  - $\mathbb{N}$ – No! This set is not finite. Also, $42 \in \mathbb{N}$ consists of 4 and 2.
  - $\{a, \ldots, z\} \cup \{A, \ldots, Z\}$
- $\omega \in \Sigma^*$ – a *word*:
  - ε (empty word)
  - 101010
  - HELLO WORLD (if the blank is a symbol, otherwise these are two words)
- L – a *language*:
  - $L = \{0, 1, 01, 10, 11\}$, defined over $\{0, 1\}$
  - $L' = \{a^n b^n \mid n \in \mathbb{N}\}$, defined over $\{a, b\}$ (n indicates the number of as/bs)
  - $L'' = \{\omega \in \{0, 1, 2\}^* \mid \omega$ has at most two 2s and an odd number of 1s$\}$

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

Why is deciding the word problem relevant?

$\rightarrow$ Languages can formalize properties

$\rightarrow$ Deciding the word problem means checking this property

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

Why is deciding the word problem relevant?

$\rightarrow$ Languages can formalize properties

$\rightarrow$ Deciding the word problem means checking this property

Examples:

- $1012102112 \overset{?}{\in} \{\omega \in \{0,1,2\}^* \mid$ ω has at most two 2s and an odd number of 1s$\}$

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

Why is deciding the word problem relevant?

$\rightarrow$ Languages can formalize properties

$\rightarrow$ Deciding the word problem means checking this property

Examples:

- $1012102112 \overset{?}{\in} \{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$
- $23 \overset{?}{\in} \{\omega \in \{0,\ldots,9\}^+ \mid \omega \text{ is a prime number}\}$

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

Why is deciding the word problem relevant?

$\rightarrow$ Languages can formalize properties

$\rightarrow$ Deciding the word problem means checking this property

Examples:

- $1012102112 \overset{?}{\in} \{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$
- $23 \overset{?}{\in} \{\omega \in \{0,\ldots,9\}^+ \mid \omega \text{ is a prime number}\}$
- $\mathit{someCode} \overset{?}{\in} \{\omega \in [\mathit{CHAR}]^* \mid \omega \text{ is Java code without syntax errors}\}$

The *Word Problem*

Word Problem

Given a word $\omega \in \Sigma^*$ and a language L over Σ, does $\omega \in L$ hold?

Why is deciding the word problem relevant?

$\rightarrow$ Languages can formalize properties

$\rightarrow$ Deciding the word problem means checking this property

Examples:

- $1012102112 \overset{?}{\in} \{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$
- $23 \overset{?}{\in} \{\omega \in \{0,\ldots,9\}^+ \mid \omega \text{ is a prime number}\}$
- $\mathit{someCode} \overset{?}{\in} \{\omega \in [\mathit{CHAR}]^* \mid \omega \text{ is Java code without syntax errors}\}$
- $\mathit{moreCode} \overset{?}{\in} \{\omega \in [\mathit{CHAR}]^* \mid \omega \text{ is Java code that always halts}\}$

Introduction

- Automata can be used to decide the word problem for a given language L:

Introduction

- Automata can be used to decide the word problem for a given language L:

  INPUT a word $\omega \in \Sigma^*$

  OUTPUT *"yes"* if $\omega \in L$ and *"no"* otherwise

Introduction

- Automata can be used to decide the word problem for a given language L:

  INPUT a word $\omega \in \Sigma^*$

  OUTPUT *"yes"* if $\omega \in L$ and *"no"* otherwise

- There are different kinds of automata, which can decide the word problem for different kinds of languages, depending on their "difficulty"

Introduction

- Automata can be used to decide the word problem for a given language L:

  INPUT a word $\omega \in \Sigma^*$

  OUTPUT *"yes"* if $\omega \in L$ and *"no"* otherwise

- There are different kinds of automata, which can decide the word problem for different kinds of languages, depending on their "difficulty"
  - $\rightsquigarrow$ Classification of languages (Chomsky hierarchy, see last lecture)
  - $\rightsquigarrow$ Different automata for the different languages

Introduction

- Automata can be used to decide the word problem for a given language L:

  INPUT a word $\omega \in \Sigma^*$

  OUTPUT *"yes"* if $\omega \in L$ and *"no"* otherwise

- There are different kinds of automata, which can decide the word problem for different kinds of languages, depending on their "difficulty"
  - $\rightsquigarrow$ Classification of languages (Chomsky hierarchy, see last lecture)
  - $\rightsquigarrow$ Different automata for the different languages
- Today: Deterministic Finite Automata (DFA)

Introduction

- Automata can be used to decide the word problem for a given language L:

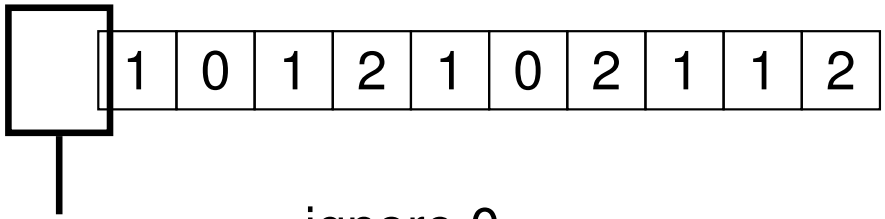
  INPUT a word $\omega \in \Sigma^*$

  OUTPUT *"yes"* if $\omega \in L$ and *"no"* otherwise

- There are different kinds of automata, which can decide the word problem for different kinds of languages, depending on their "difficulty"
  - $\rightsquigarrow$ Classification of languages (Chomsky hierarchy, see last lecture)
  - $\rightsquigarrow$ Different automata for the different languages
- Today: Deterministic Finite Automata (DFA)
  - They consist of finitely many states and state transitions
  - They can decide the word problem for regular languages

Deciding the Word Problem, Example I

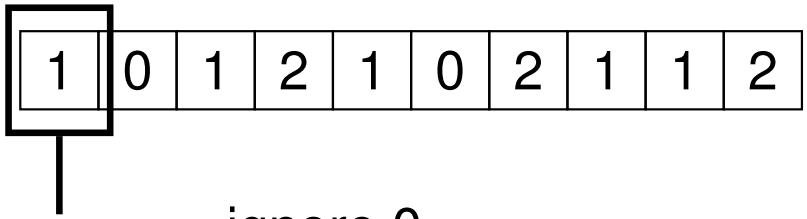
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	—
count 2	0

Deciding the Word Problem, Example I

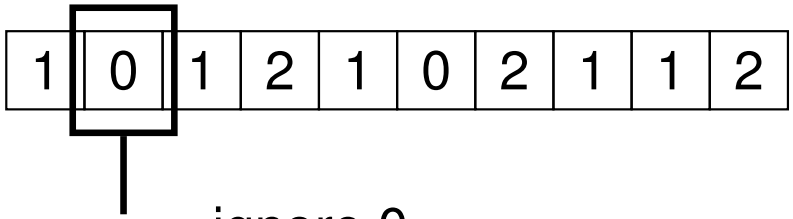
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	0

Deciding the Word Problem, Example I

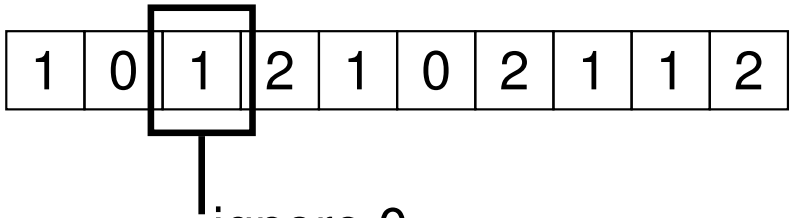
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	0

Deciding the Word Problem, Example I

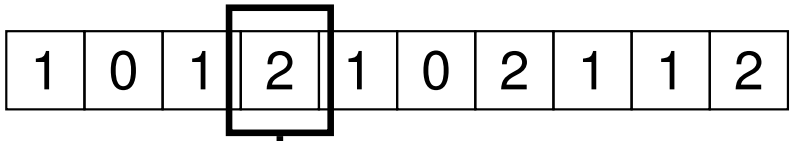
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*no*
count 2	0

Deciding the Word Problem, Example I

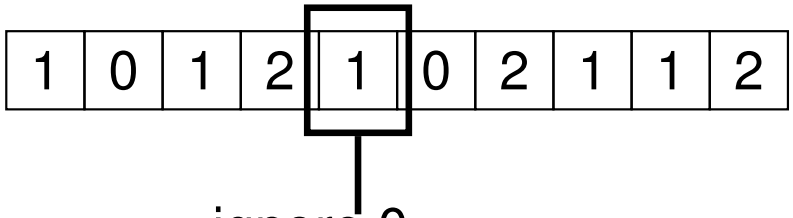
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*no*
count 2	1

Deciding the Word Problem, Example I

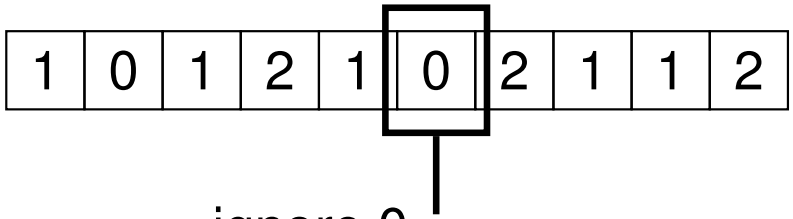
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	1

Deciding the Word Problem, Example I

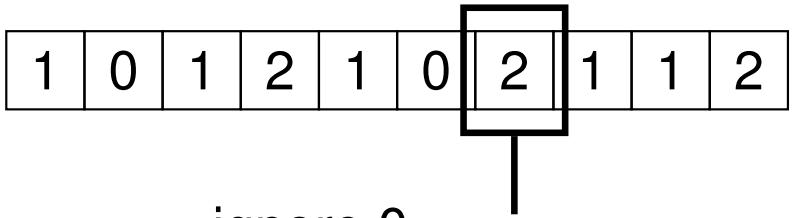
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	1

Deciding the Word Problem, Example I

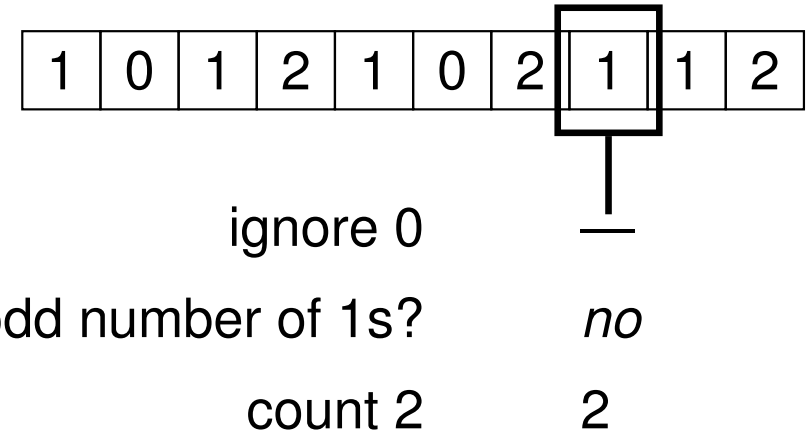
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	2

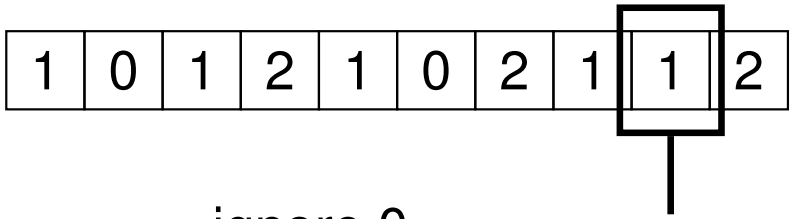
Deciding the Word Problem, Example I

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



Deciding the Word Problem, Example I

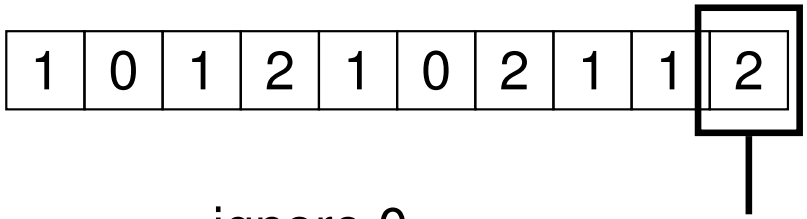
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	2

Deciding the Word Problem, Example I

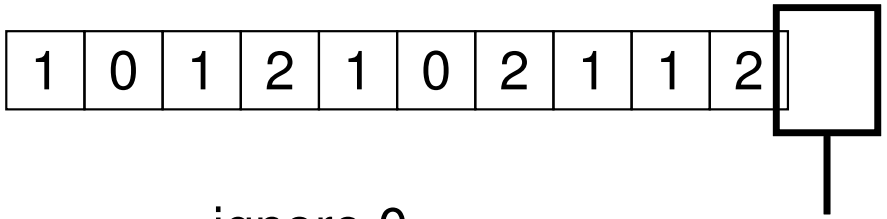
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes*
count 2	3

Deciding the Word Problem, Example I

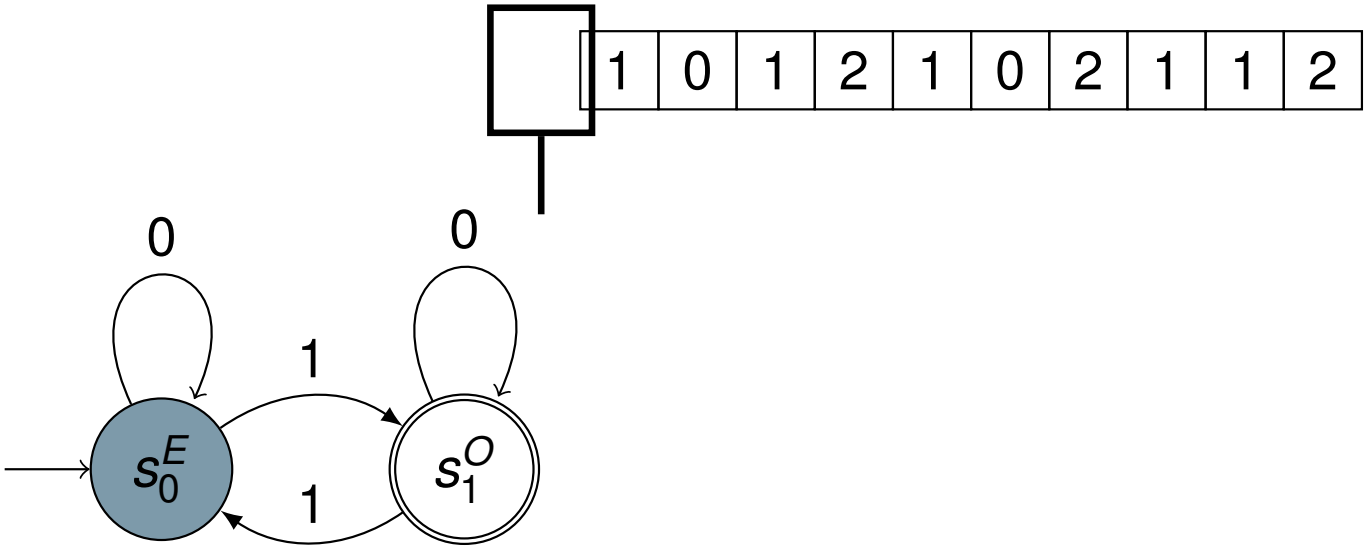
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



ignore 0	—
odd number of 1s?	*yes* ✓
count 2	3 ✗

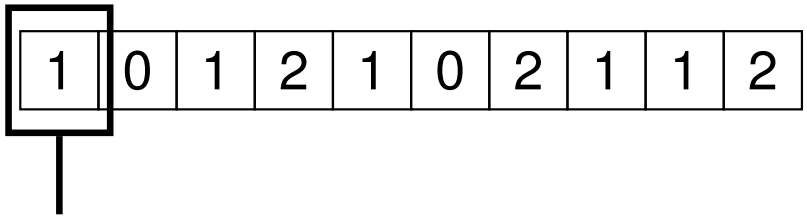
Deciding the Word Problem, Example II

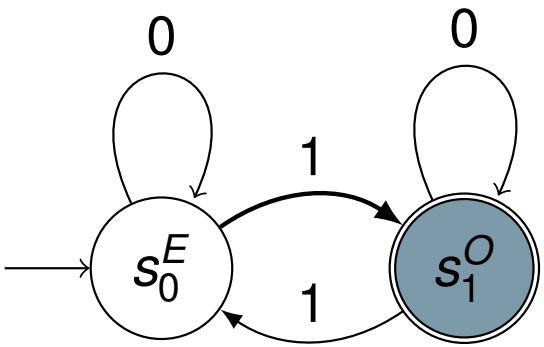
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



Deciding the Word Problem, Example II

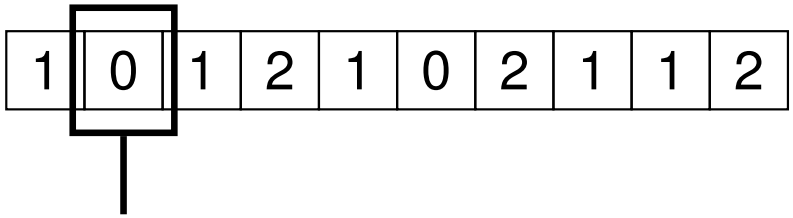
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$

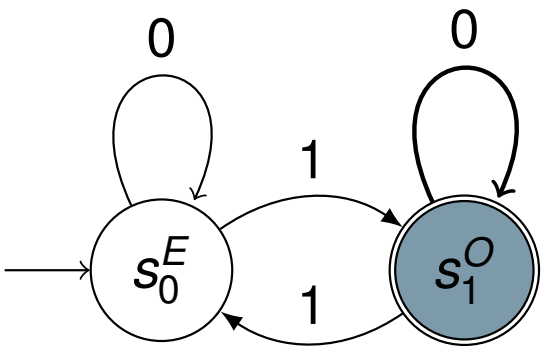




Deciding the Word Problem, Example II

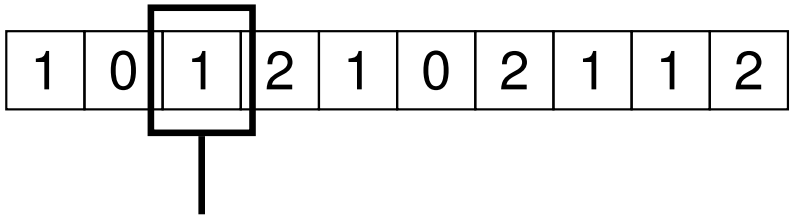
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$

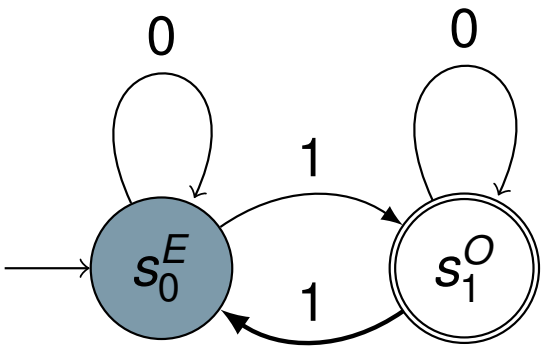




Deciding the Word Problem, Example II

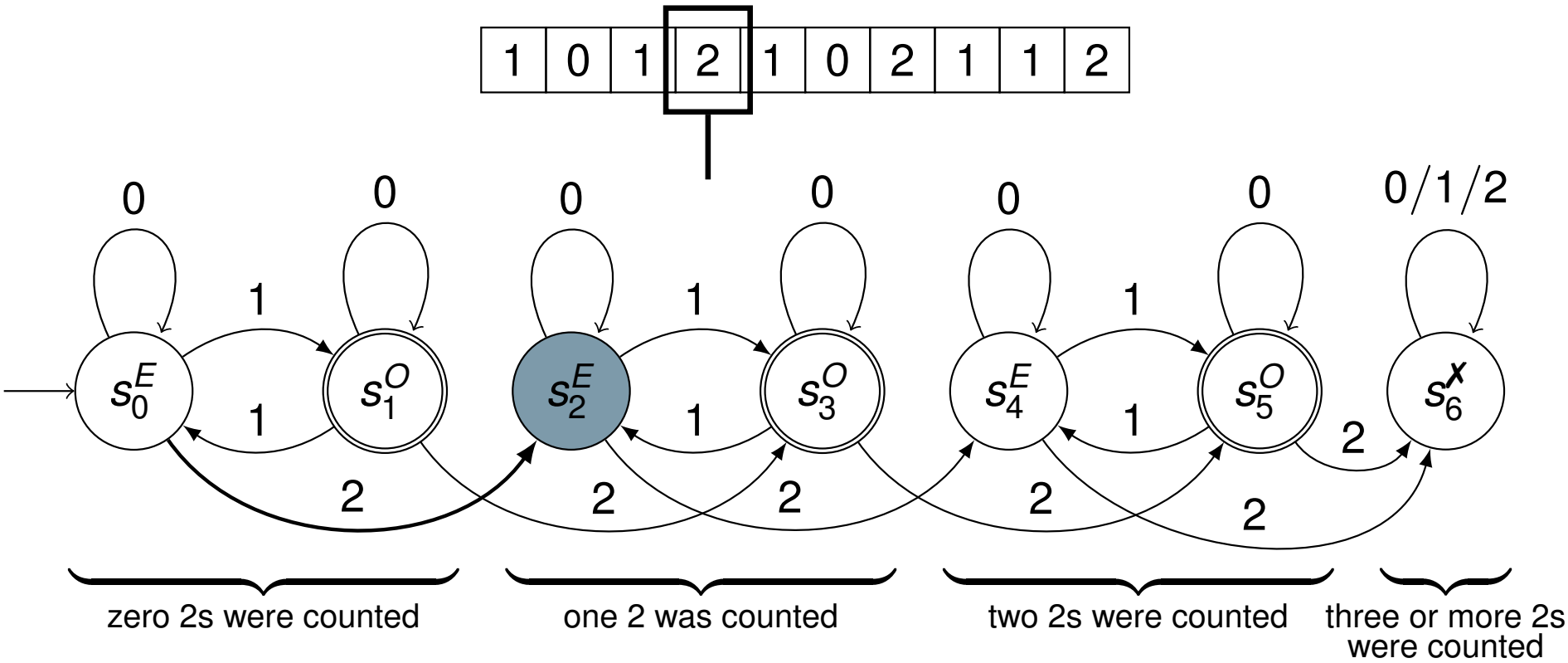
We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$





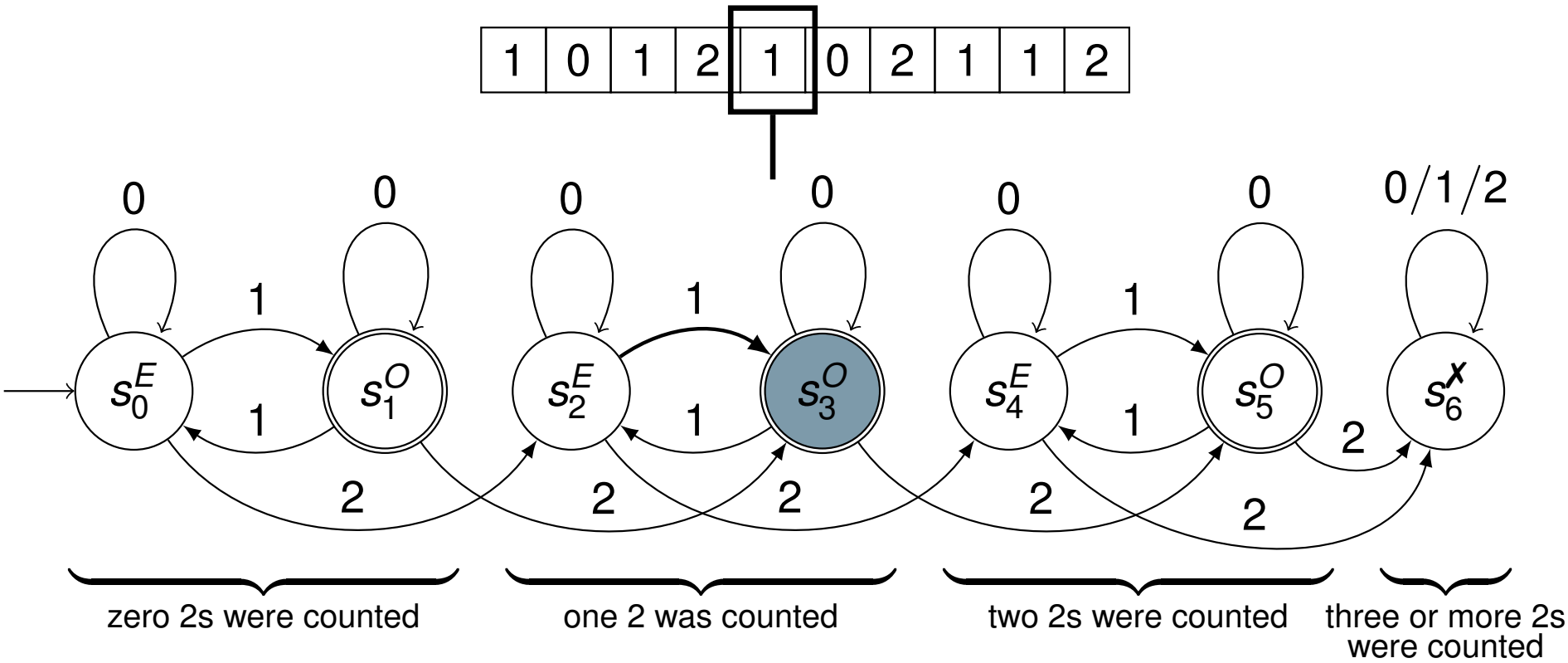
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



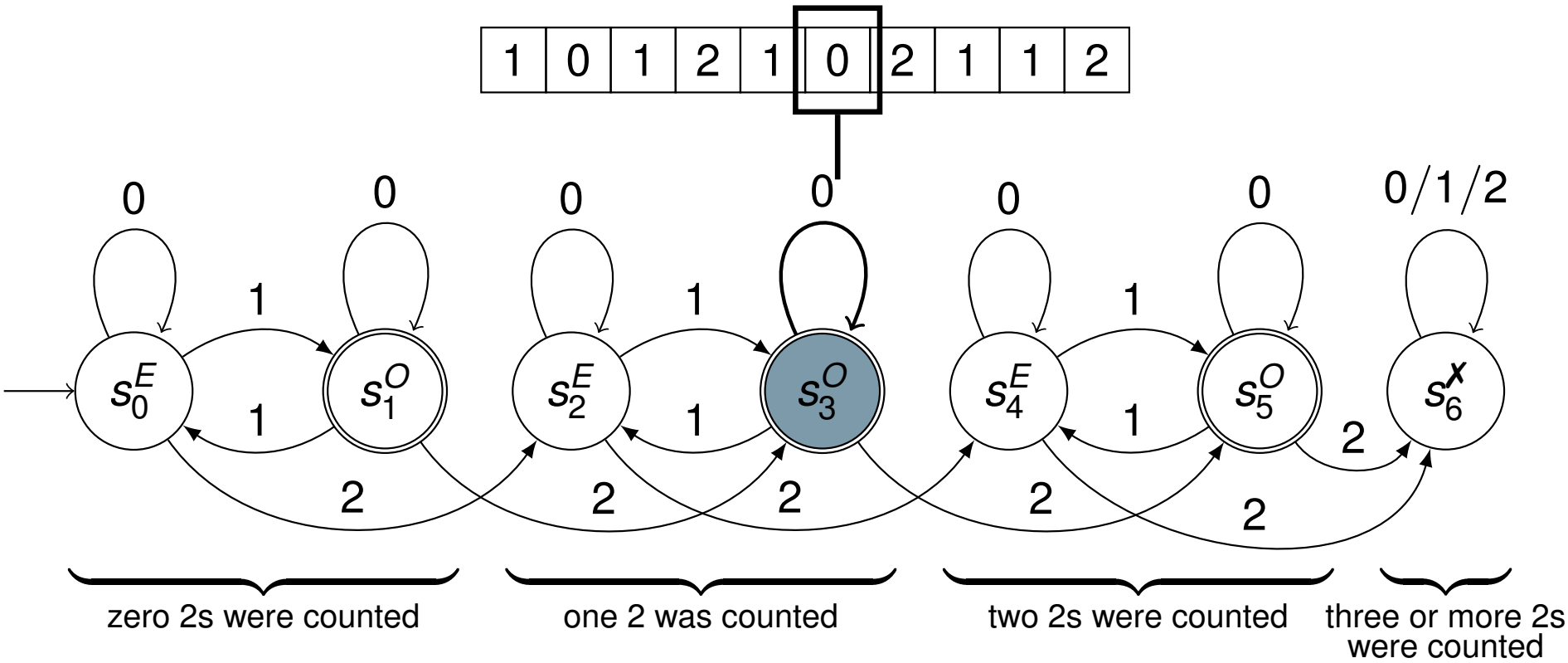
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



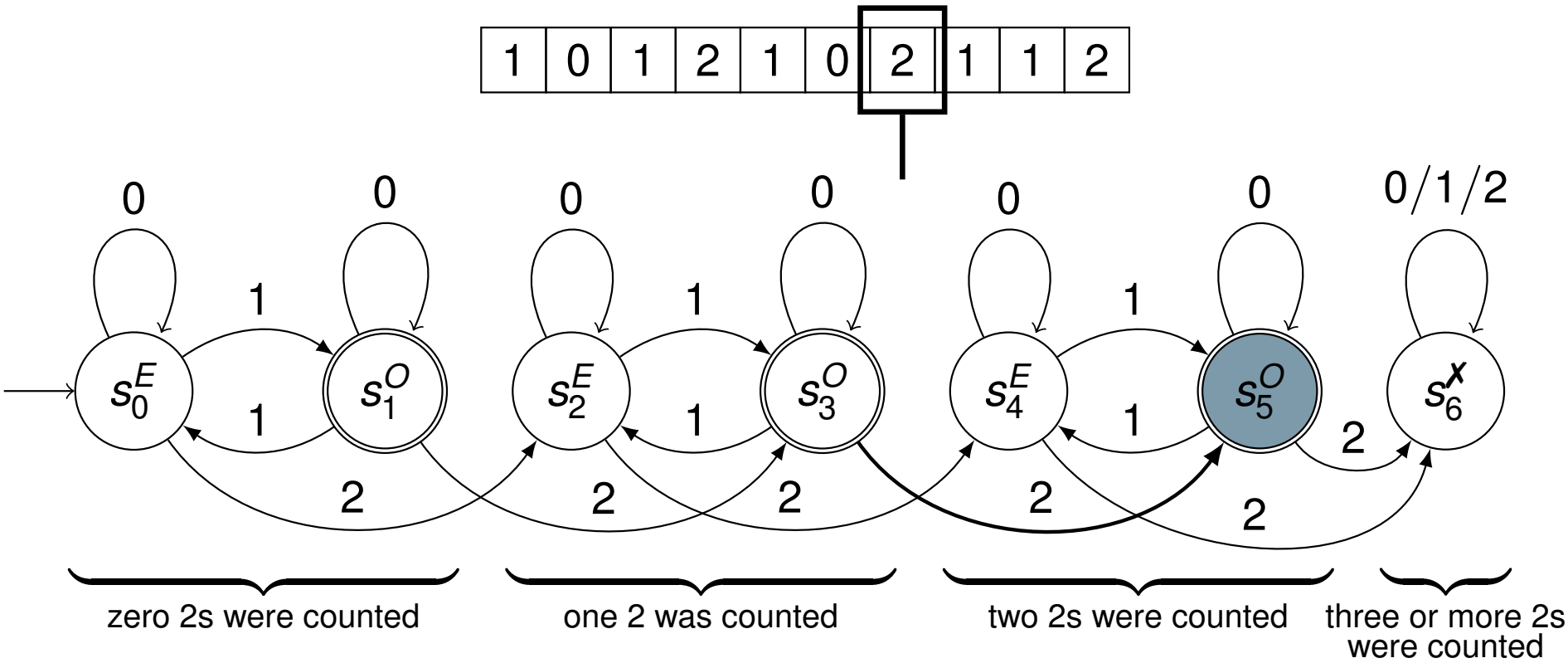
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



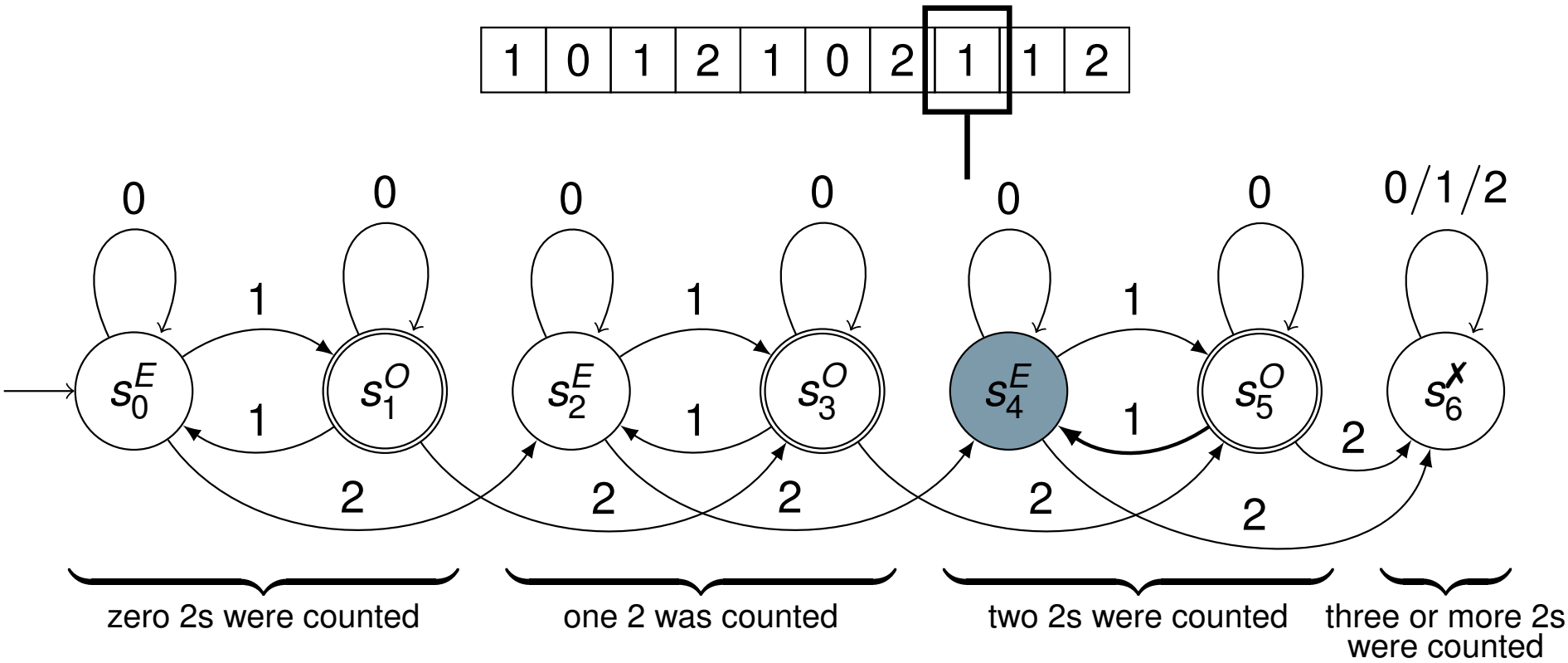
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



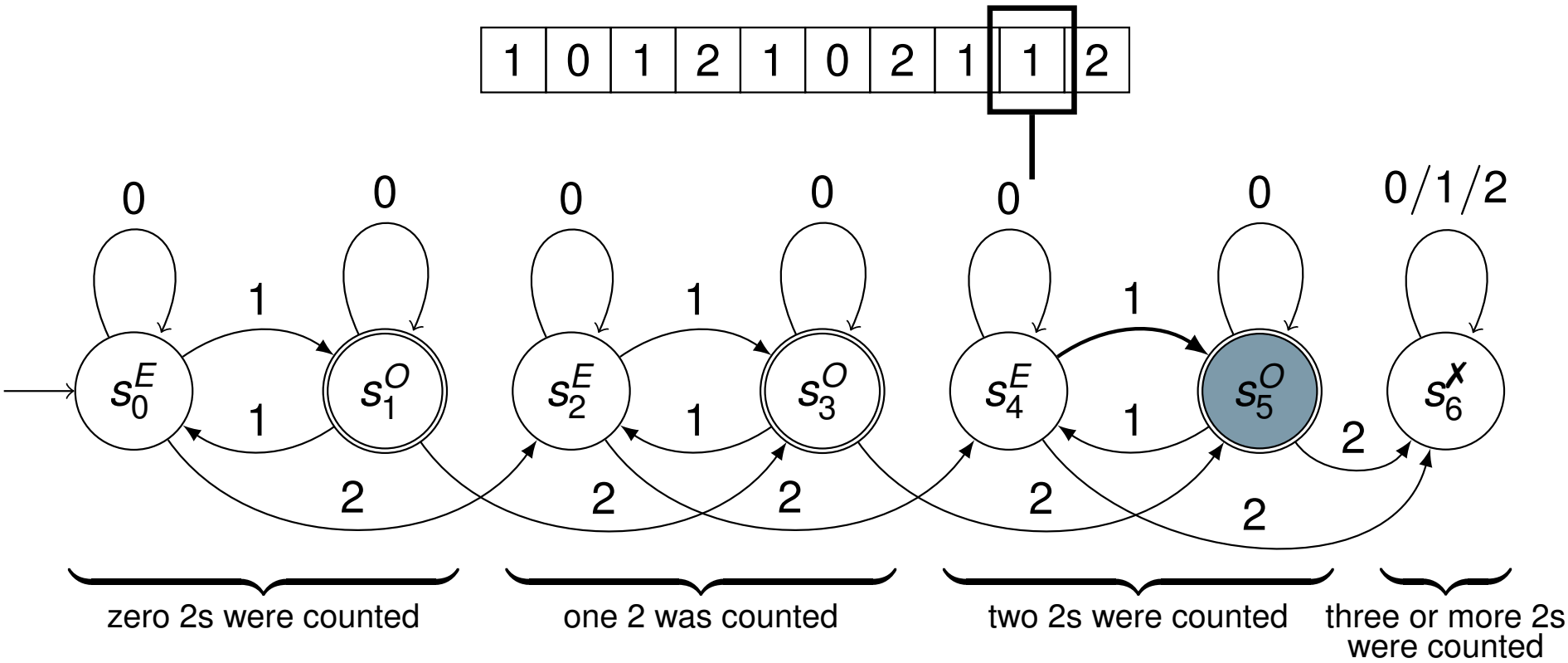
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



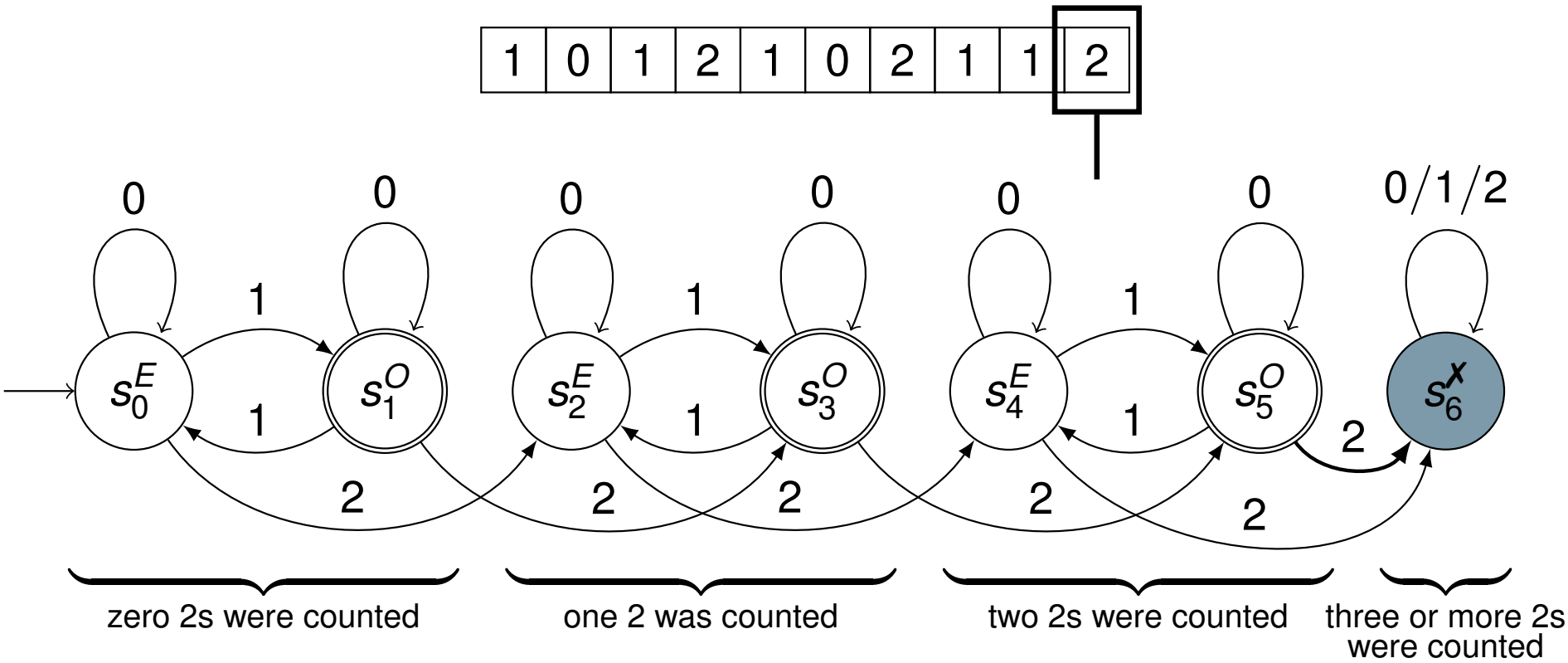
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$



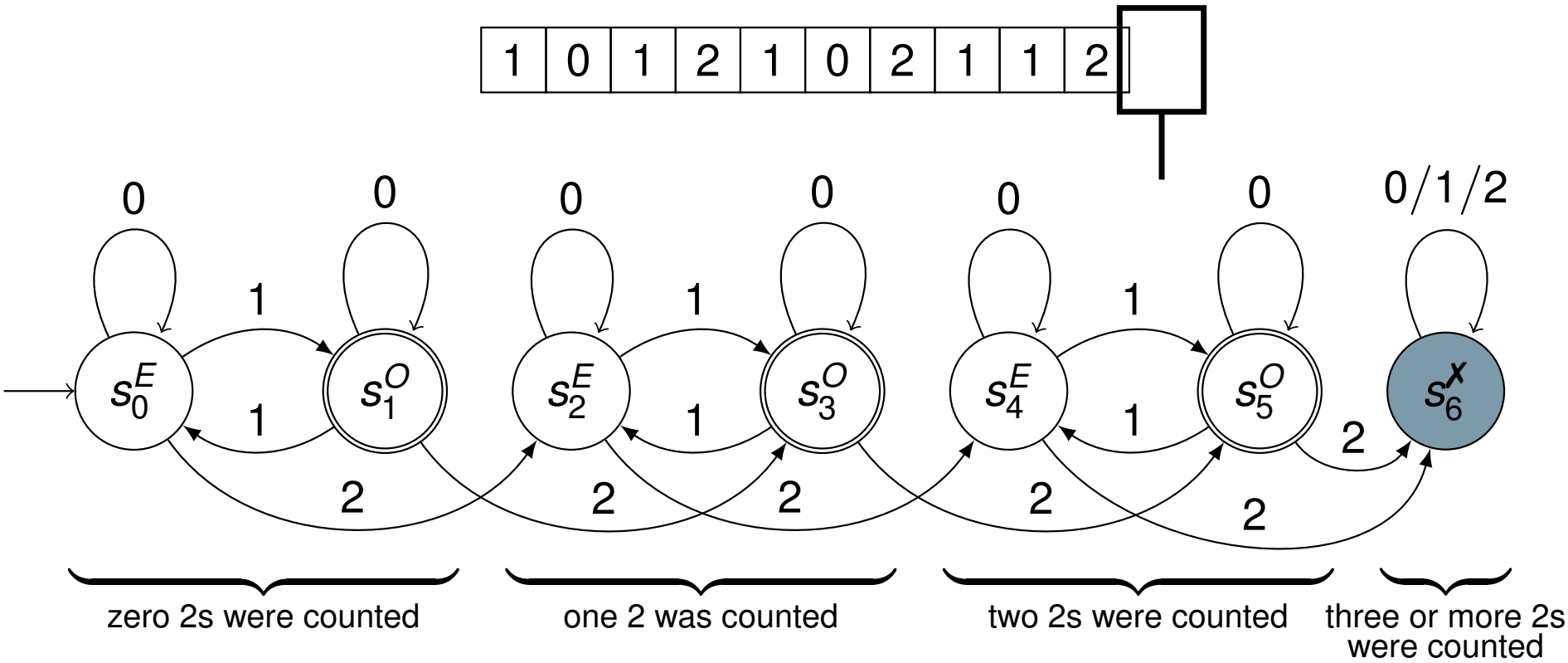
Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$

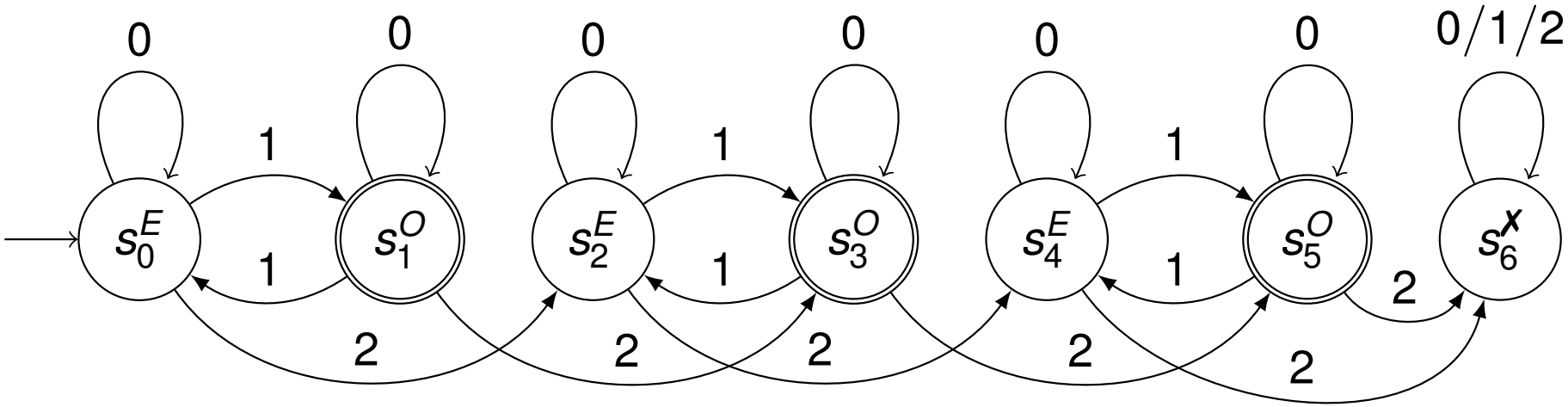


Deciding the Word Problem, Example II

We want to construct a finite automaton that decides the word problem for $\{\omega \in \{0,1,2\}^* \mid \omega \text{ has at most two 2s and an odd number of 1s}\}$

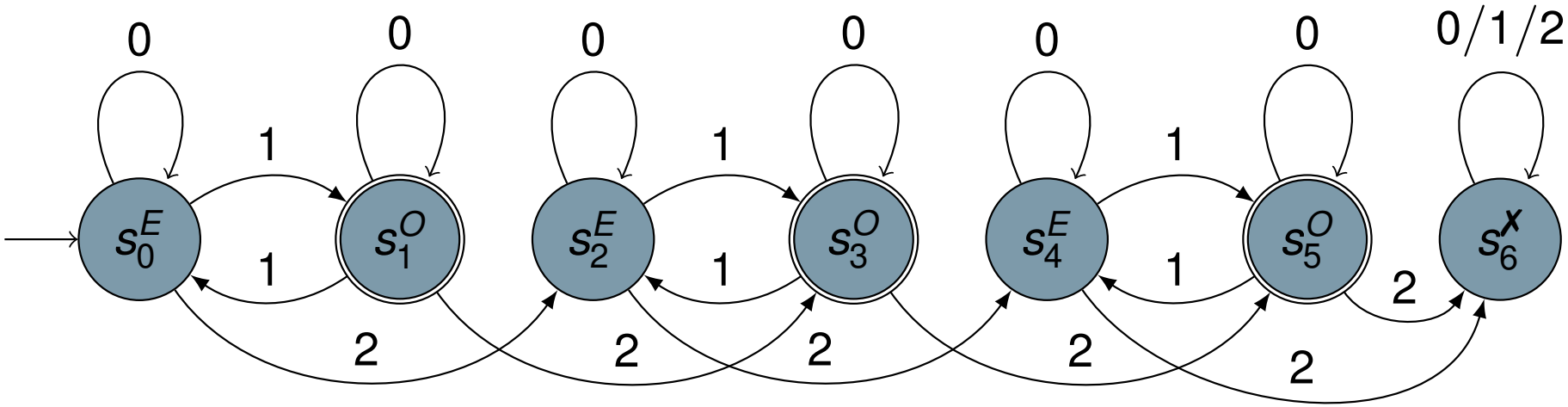


Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

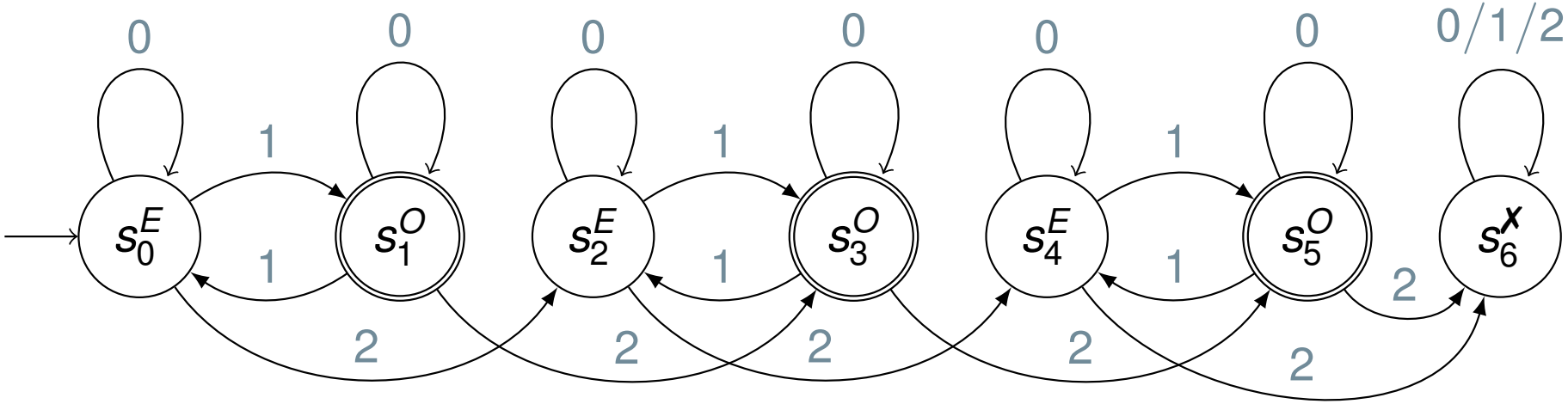
Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

- A finite set of *states Q*

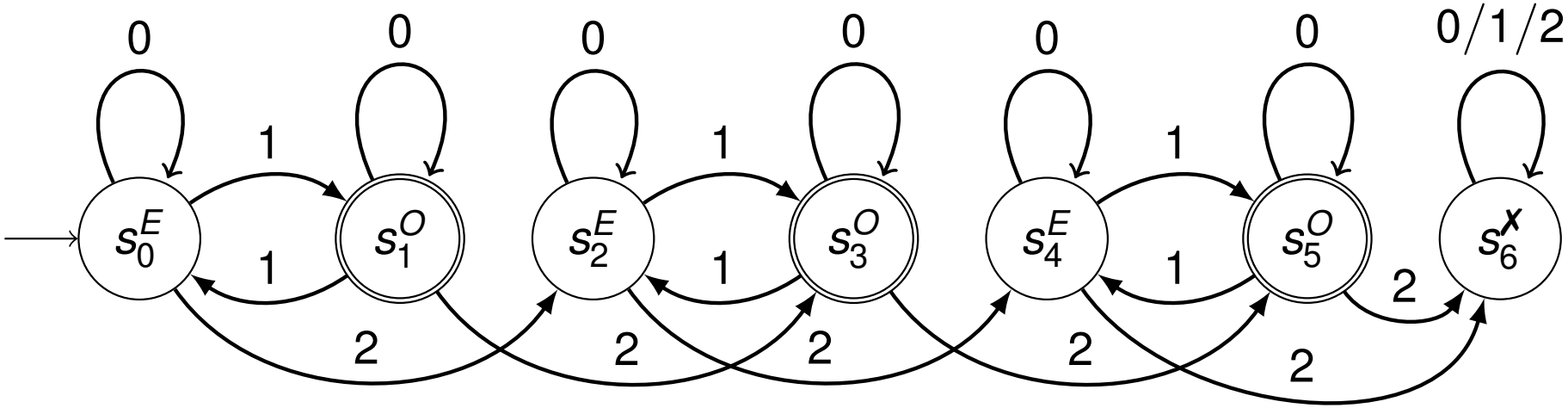
Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

- A finite set of *states* Q
- A finite *alphabet* Σ (the symbols on the edges)

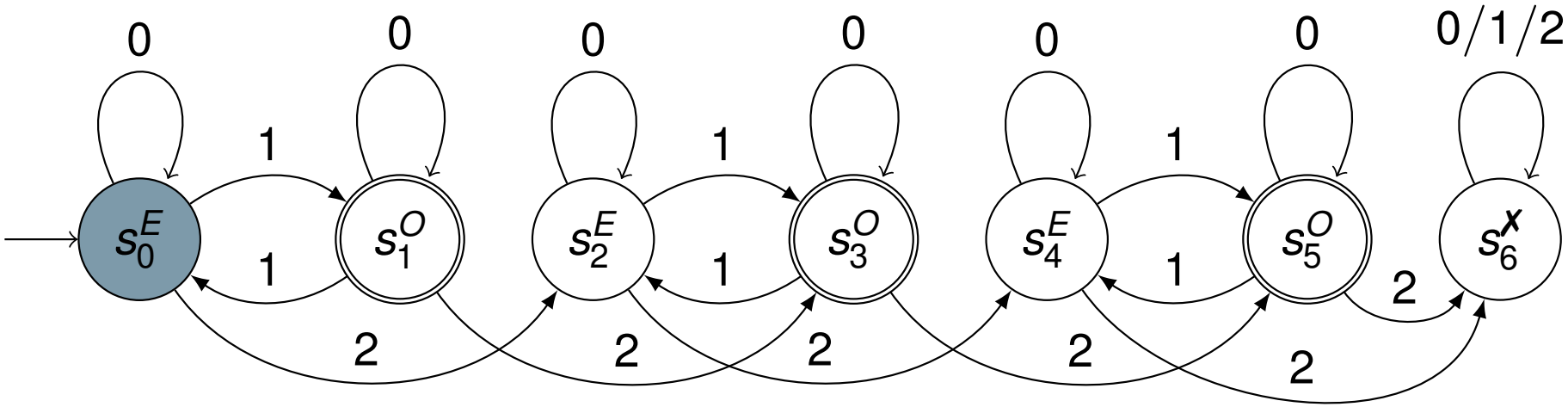
Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

- A finite set of *states* Q
- A finite *alphabet* Σ (the symbols on the edges)
- A *state transition function* $\delta : Q \times \Sigma \to Q$

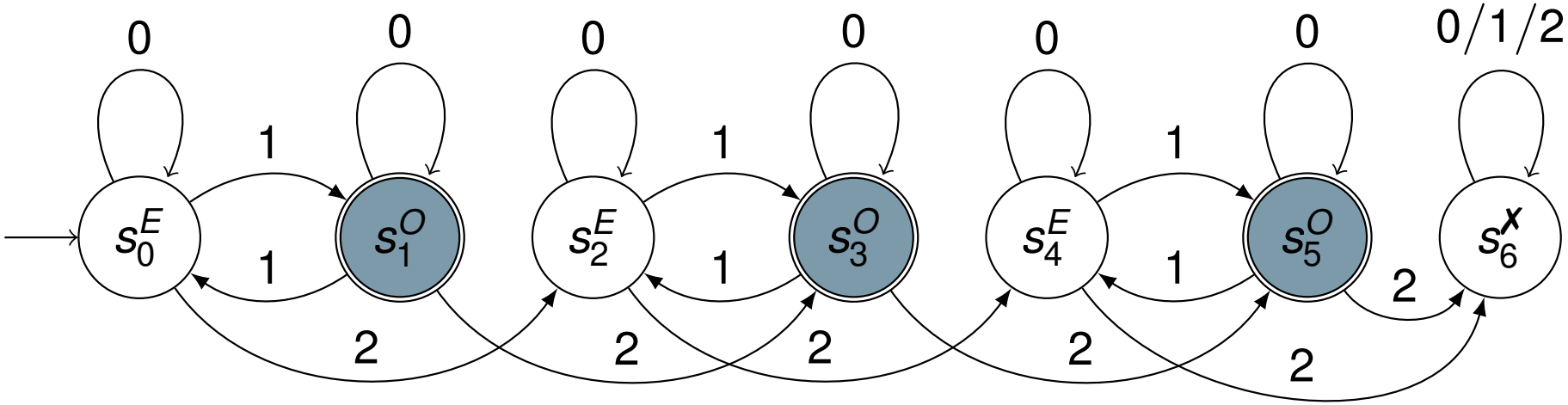
Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

- A finite set of *states* Q
- A finite *alphabet* Σ (the symbols on the edges)
- A *state transition function* $\delta : Q \times \Sigma \to Q$
- An *initial state* $q_o \in Q$ (here: s_0^E)

Finite Automata



(Deterministic) finite automata (or *finite state machines*) consist of:

- A finite set of *states* Q
- A finite *alphabet* Σ (the symbols on the edges)
- A *state transition function* $\delta : Q \times \Sigma \to Q$
- An *initial state* $q_o \in Q$ (here: s_0^E)
- A set of *accepting* (or *final*) states $F \subseteq Q$

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

- $\hat{\delta}(q, \sigma\omega) \mapsto \hat{\delta}(\delta(q, \sigma), \omega)$

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

- $\hat{\delta}(q, \sigma\omega) \mapsto \hat{\delta}(\delta(q, \sigma), \omega)$
- $\hat{\delta}(q, \sigma) \mapsto \delta(q, \sigma)$

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

- $\hat{\delta}(q, \sigma\omega) \mapsto \hat{\delta}(\delta(q, \sigma), \omega)$
- $\hat{\delta}(q, \sigma) \mapsto \delta(q, \sigma)$
- $\hat{\delta}(q, \varepsilon) \mapsto q$

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

- $\hat{\delta}(q, \sigma\omega) \mapsto \hat{\delta}(\delta(q, \sigma), \omega)$
- $\hat{\delta}(q, \sigma) \mapsto \delta(q, \sigma)$
- $\hat{\delta}(q, \varepsilon) \mapsto q$

We define $L(A) = \{\omega \in \Sigma^* \mid \hat{\delta}(q_0, \omega) \cap F \neq \emptyset\}$ as the *language of A*.

That is, $L(A)$ is the set of inputs *accepted* (or *recognized*) by A.

The Language Accepted by an Automaton

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton.

For $q \in Q$, $\sigma \in \Sigma$, and $\omega \in \Sigma^*$ we define the extension of δ to *words* by:

- $\hat{\delta}(q, \sigma\omega) \mapsto \hat{\delta}(\delta(q, \sigma), \omega)$
- $\hat{\delta}(q, \sigma) \mapsto \delta(q, \sigma)$
- $\hat{\delta}(q, \varepsilon) \mapsto q$

We define $L(A) = \{\omega \in \Sigma^* \mid \hat{\delta}(q_0, \omega) \cap F \neq \emptyset\}$ as the *language of A*.

That is, $L(A)$ is the set of inputs *accepted* (or *recognized*) by A.

Example:
1012102112 is *not* accepted by the previous automaton, since
$\hat{\delta}(s_0^E, 1012102112) = s_6^{\cancel{X}} \notin F = \{s_1^O, s_3^O, s_5^O\}$

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \rightarrow (\Sigma \cup \Gamma)^*$, a finite set of production rules

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \rightarrow (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:
  - $\gamma \to \varepsilon$,

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:
  - $\gamma \to \varepsilon$,
  - $\gamma \to \sigma$ with $\sigma \in \Sigma$, or

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:
  - $\gamma \to \varepsilon$,
  - $\gamma \to \sigma$ with $\sigma \in \Sigma$, or
  - $\gamma \to \sigma\gamma'$ with $\sigma \in \Sigma$ and $\gamma' \in \Gamma$
    Rules of this type are called *right-regular*

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:
  - $\gamma \to \varepsilon$,
  - $\gamma \to \sigma$ with $\sigma \in \Sigma$, or
  - $\gamma \to \sigma\gamma'$ with $\sigma \in \Sigma$ and $\gamma' \in \Gamma$
    Rules of this type are called *right-regular*
  - Left-regular rules (defined analogously) are also allowed, but left- and right-regular rules cannot be mixed

Recap on Regular Languages

- A *context-free formal grammar* is a tuple $G = (\Gamma, \Sigma, R, S)$ with:
  - Γ, a finite set of non-terminal symbols
  - Σ, a finite set of terminal symbols
  - $R : \Gamma \to (\Sigma \cup \Gamma)^*$, a finite set of production rules
  - $S \in \Gamma$, the start symbol
- In regular grammars, all production rules $r \in R$ have one of the following form:
  - $\gamma \to \varepsilon$,
  - $\gamma \to \sigma$ with $\sigma \in \Sigma$, or
  - $\gamma \to \sigma\gamma'$ with $\sigma \in \Sigma$ and $\gamma' \in \Gamma$
    Rules of this type are called *right-regular*
  - Left-regular rules (defined analogously) are also allowed, but left- and right-regular rules cannot be mixed
- *Regular languages* are those that can be generated by regular grammars.

Finite Automata and Regular Languages

Theorem

Finite Automata accept *exactly* the regular languages.

Finite Automata and Regular Languages

Theorem

Finite Automata accept *exactly* the regular languages.

More formally, this means:

"$\Rightarrow$" Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

Finite Automata and Regular Languages

Theorem

Finite Automata accept *exactly* the regular languages.

More formally, this means:

"$\Rightarrow$" Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

"$\Leftarrow$" Let L_{reg} be regular language. Then, there exists a finite automaton A, such that $L(A) = L_{reg}$.

Today, we only show "$\Rightarrow$".

Finite Automata and Regular Languages, Proof "$\Rightarrow$"

To show:

Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

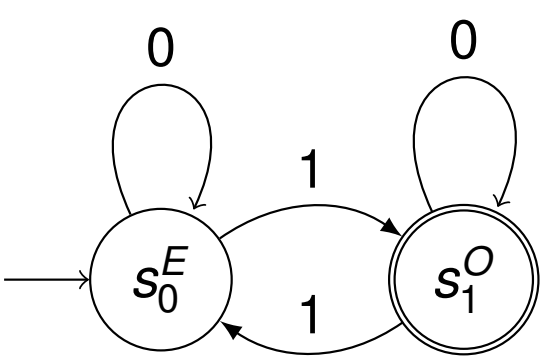
Finite Automata and Regular Languages, Proof "$\Rightarrow$"

To show:

Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

Proof sketch:

- The automaton's states are the grammar's non-terminals



$$G = (\overbrace{\Gamma}^{\{s_0^E, s_1^O, s_2^E, s_3^O, s_4^E, s_5^O, s_6^{\cancel{X}}\}}, \Sigma = \{0,1,2\}, R, S = s_O^E) \text{ with}$$

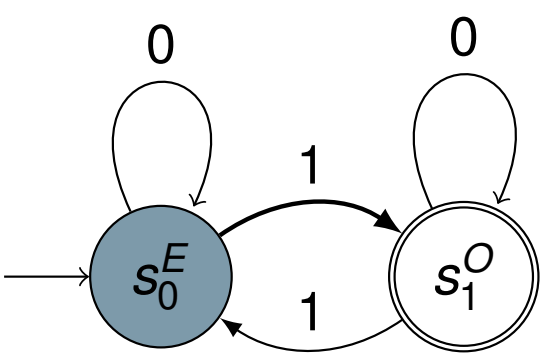
Finite Automata and Regular Languages, Proof "⇒"

To show:

Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

Proof sketch:

- The automaton's states are the grammar's non-terminals
- If $\delta(q_i, \sigma) = q_j$ with $q_i, q_j \in Q$ and $\sigma \in \Sigma$, then, add rule $q_i \to \sigma q_j$ to R.



$\{s_0^E, s_1^O, s_2^E, s_3^O, s_4^E, s_5^O, s_6^{\cancel{X}}\}$

$$G = (\overbrace{\Gamma}, \Sigma = \{0,1,2\}, R, S = s_O^E) \text{ with}$$

$$s_0^E \to 0s_0^E \in R,\ s_0^E \to 1s_1^O \in R,$$

$$s_1^O \to 0s_1^O \in R,\ s_1^O \to 1s_0^E \in R,$$

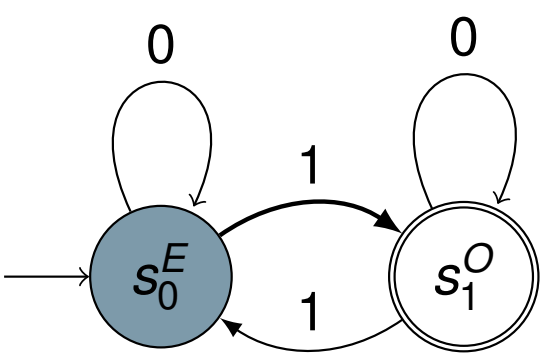
Finite Automata and Regular Languages, Proof "⇒"

To show:

Let A be a finite automaton and $L(A)$ its language. Then, $L(A)$ is regular, i.e., there exists a regular grammar G with $L(G) = L(A)$.

Proof sketch:

- The automaton's states are the grammar's non-terminals
- If $\delta(q_i, \sigma) = q_j$ with $q_i, q_j \in Q$ and $\sigma \in \Sigma$, then, add rule $q_i \to \sigma q_j$ to R.
- If $\delta(q_i, \sigma) = q_j$ with $q_i, q_j \in Q$ and $\sigma \in \Sigma$, and $q_j \in F$, then add rule $q_i \to \sigma$ to R (we can terminate here).



$$G = (\overbrace{\Gamma}^{\{s_0^E, s_1^O, s_2^E, s_3^O, s_4^E, s_5^O, s_6^{\cancel{X}}\}}, \Sigma = \{0,1,2\}, R, S = s_O^E) \text{ with}$$

$$s_0^E \to 0s_0^E \in R,\ s_0^E \to 1s_1^O \in R,$$

$$s_1^O \to 0s_1^O \in R,\ s_1^O \to 1s_0^E \in R,$$

$$s_0^E \to 1 \in R,\ s_1^O \to 0 \in R$$

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions
- They can decide the word problem for (exactly) the regular languages

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions
- They can decide the word problem for (exactly) the regular languages
- Since regular languages have many applications, so do finite automata, e.g.:

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions
- They can decide the word problem for (exactly) the regular languages
- Since regular languages have many applications, so do finite automata, e.g.:
  - Compiler construction / parsing program code

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions
- They can decide the word problem for (exactly) the regular languages
- Since regular languages have many applications, so do finite automata, e.g.:
  - Compiler construction / parsing program code
  - Checking regular expressions

Summary

- Finite automata describe simple “machines” with finitely many states and state transitions
- They can decide the word problem for (exactly) the regular languages
- Since regular languages have many applications, so do finite automata, e.g.:
  - Compiler construction / parsing program code
  - Checking regular expressions
  - Basis for many theoretical investigations, e.g., for Artificial Intelligence (AI) planning