

Regular Sets

and automata

1 Review: Notation

- alphabet: finite set

$$A = \{a_1, \dots, a_n\}$$

- cardinality

$$\#A = n$$

- string, word with letters in A

$$b = b_1 \dots b_\ell \quad \text{where} \quad \forall i. b_i \in A$$

- length of string b

$$|b| = \ell$$

- concatenation. $b = b_1 \dots b_r, c = c_1 \dots c_s$

$$bc = b \circ c = b_1 \dots b_r c_1 \dots c_s$$

- strings of length i with letters in A :

$$A^i = \{x : x \text{ word with letters in } A, |x| = i\}$$

- all nonempty finite words with letters in A

$$A^+ = \bigcup_{i=1}^{\infty} A^i$$

1 Review: Notation

- alphabet: finite set

$$A = \{a_1, \dots, a_n\}$$

- cardinality

$$\#A = n$$

- string, word with letters in A

$$b = b_1 \dots b_\ell \quad \text{where} \quad \forall i. b_i \in A$$

- length of string b

$$|b| = \ell$$

- concatenation. $b = b_1 \dots b_r, c = c_1 \dots c_s$

$$bc = b \circ c = b_1 \dots b_r c_1 \dots c_s$$

- strings of length i with letters in A :

$$A^i = \{x : x \text{ word with letters in } A, |x| = i\}$$

- all nonempty finite words with letters in A

$$A^+ = \bigcup_{i=1}^{\infty} A^i$$

- empty word ε :

$$a \circ \varepsilon = \varepsilon \circ a = a$$

- all finite words with letters in A

$$A^* = A^+ \cup \{\varepsilon\} = \bigcup_{i=0}^{\infty} A^i$$

- language L

$$L \subseteq A^*$$

Note that historically this was a *radically abstract* definition.

- concatenation of languages $L_1, L_2 \subseteq A^*$:

$$L_1 \circ L_2 = \{bc : b \in L_1, c \in L_2\}$$

- complement of $L \subseteq A^*$:

$$\bar{L} = A^* \setminus L$$

- concatenating strings in L on gets L^* :

$$L^* = \{w^{(1)} \circ \dots \circ w^{(s)} : w^{(i)} \in L \text{ for all } i \in [1 : s]\}$$

2 Regular sets and Expressions

def: regular sets over alphabet A

1.

$$\{\varepsilon\} \in R(A)$$

2.

$$a \in A \rightarrow \{a\} \in R(A)$$

3. $L_1, L_2 \in R(A) \rightarrow$

$$L_1 \circ L_2, L_1 \cup L_2, L_1^* \in R(A)$$

4. these are all

2 Regular sets and Expressions

def: regular sets over alphabet A

1.

$$\{\varepsilon\} \in R(A)$$

2.

$$a \in A \rightarrow \{a\} \in R(A)$$

3. $L_1, L_2 \in R(A) \rightarrow$

$$L_1 \circ L_2, L_1 \cup L_2, L_1^* \in R(A)$$

4. these are all

how to specify them?

regular expressions over alphabet A

1. Empty expression is regular:

$$\varepsilon \in RE(A)$$

2.

$$a \in A \rightarrow a \in RE(A)$$

3. $c, d \in RE(A) \rightarrow$

$$c \circ d, c \cup d, c^* \in RE(A)$$

4. these are all

2 Regular sets and Expressions

def: regular sets over alphabet A

1.

$$\{\varepsilon\} \in R(A)$$

2.

$$a \in A \rightarrow \{a\} \in R(A)$$

3. $L_1, L_2 \in R(A) \rightarrow$

$$L_1 \circ L_2, L_1 \cup L_2, L_1^* \in R(A)$$

4. these are all

how to specify them?

regular expressions over alphabet A

1.

$$\varepsilon \in RE(A)$$

2.

$$a \in A \rightarrow a \in RE(A)$$

3. $c, d \in RE(A) \rightarrow$

$$c \circ d, c \cup d, c^* \in RE(A)$$

4. these are all

: semantics language $L(e)$ generated by regular expression e

1.

$$L(\varepsilon) = \{\varepsilon\}$$

2.

$$a \in A \rightarrow L(a) = \{a\}$$

3.

$$L(c \circ d) = L(c) \circ L(d), L(c \cup d) = L(c) \cup L(d), L(c^*) = (L(c))^*$$

def: regular language/regular set

$$L \subseteq A^* \text{ regular} \iff \exists c \in RE(A). L = L(c)$$

2 Regular sets and Expressions

def: regular sets over alphabet A

1.

$$\{\varepsilon\} \in R(A)$$

2.

$$a \in A \rightarrow \{a\} \in R(A)$$

3. $L_1, L_2 \in R(A) \rightarrow$

$$L_1 \circ L_2, L_1 \cup L_2, L_1^* \in R(A)$$

4. these are all

how to specify them?

regular expressions over alphabet A

1.

$$\varepsilon \in RE(A)$$

2.

$$a \in A \rightarrow a \in RE(A)$$

3. $c, d \in RE(A) \rightarrow$

$$c \circ d, c \cup d, c^* \in RE(A)$$

4. these are all

: semantics language $L(e)$ generated by regular expression e

1.

$$L(\varepsilon) = \{\varepsilon\}$$

2.

$$a \in A \rightarrow L(a) = \{a\}$$

3.

$$L(c \circ d) = L(c) \circ L(d), L(c \cup d) = L(c) \cup L(d), L(c^*) = (L(c))^*$$

def: regular language/regular set

$$L \subseteq A^* \text{ regular} \leftrightarrow \exists c \in RE(A). L = L(c)$$

examples of regular languages

- all sets of symbols $L \subset A$ are regular.

$$L = \{a_1, \dots, a_s\} = L(a_1 \cup \dots \cup a_s)$$

- natural numbers (in decimal representation without leading zeros)

$$\mathbb{N}_0 = \{1, \dots, 9\} \circ \{0, \dots, 9\}^* \cup \{0\}$$

- variable names V (with capital letters)

$$V = \{A, \dots, Z\} \circ \{A, \dots, Z, 0, \dots, 9\}^*$$

2 Regular sets and Expressions

def: regular sets over alphabet A

1.

$$\{\varepsilon\} \in R(A)$$

2.

$$a \in A \rightarrow \{a\} \in R(A)$$

3. $L_1, L_2 \in R(A) \rightarrow$

$$L_1 \circ L_2, L_1 \cup L_2, L_1^* \in R(A)$$

4. these are all

how to specify them?

regular expressions over alphabet A

1.

$$\varepsilon \in RE(A)$$

2.

$$a \in A \rightarrow a \in RE(A)$$

3. $c, d \in RE(A) \rightarrow$

$$c \circ d, c \cup d, c^* \in RE(A)$$

4. these are all

: semantics language $L(e)$ generated by regular expression e

1.

$$L(\varepsilon) = \{\varepsilon\}$$

2.

$$a \in A \rightarrow L(a) = \{a\}$$

3.

$$L(c \circ d) = L(c) \circ L(d), L(c \cup d) = L(c) \cup L(d), L(c^*) = (L(c))^*$$

def: regular language/regular set

$$L \subseteq A^* \text{ regular} \leftrightarrow \exists c \in RE(A). L = L(c)$$

question

$$L \text{ regular} \rightarrow \bar{L} \text{ regular?}$$

If true then

$$L_1, L_2 \text{ regular} \rightarrow L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}} \text{ regular}$$

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

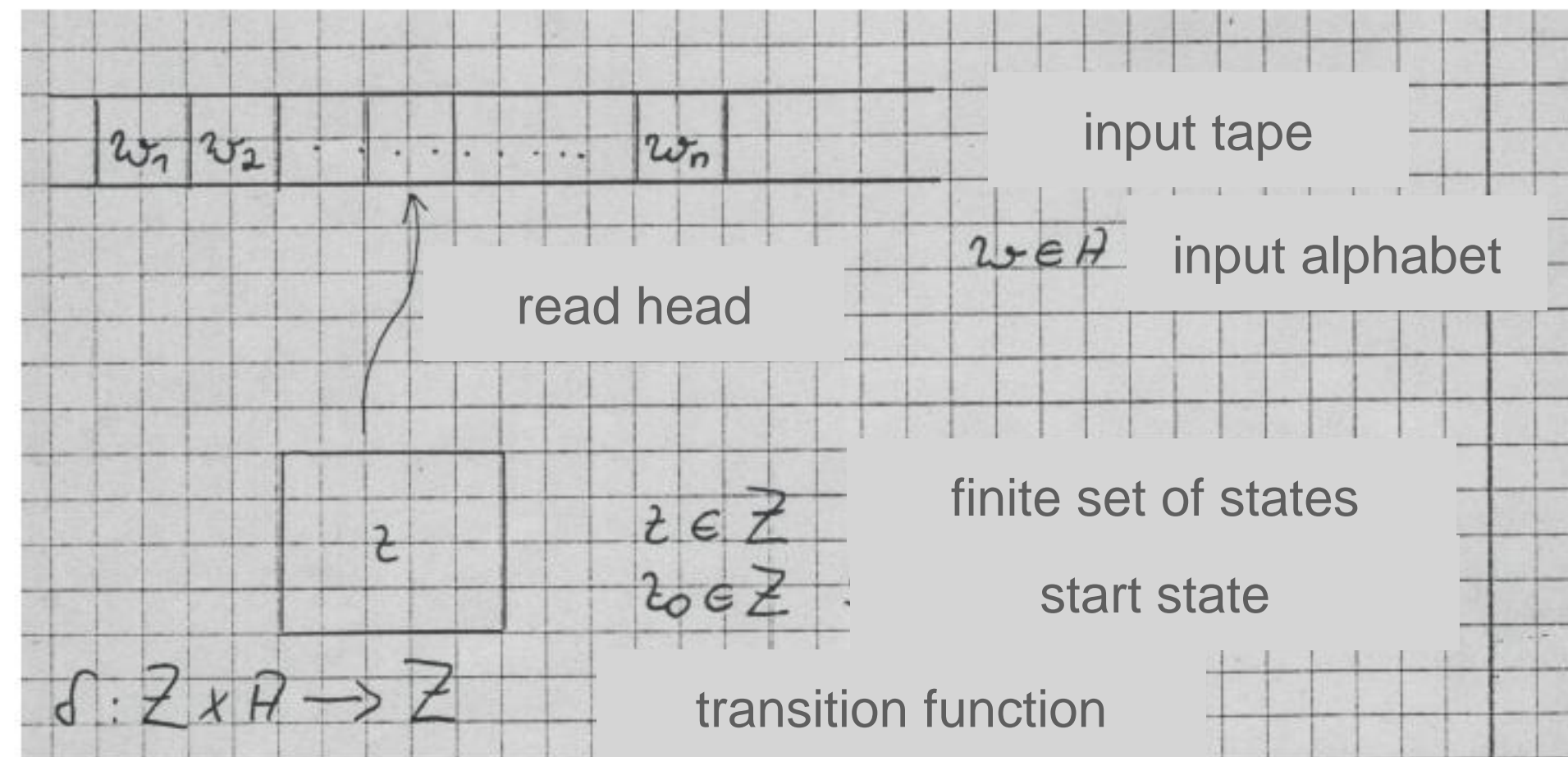


Figure 1: finite automaton

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

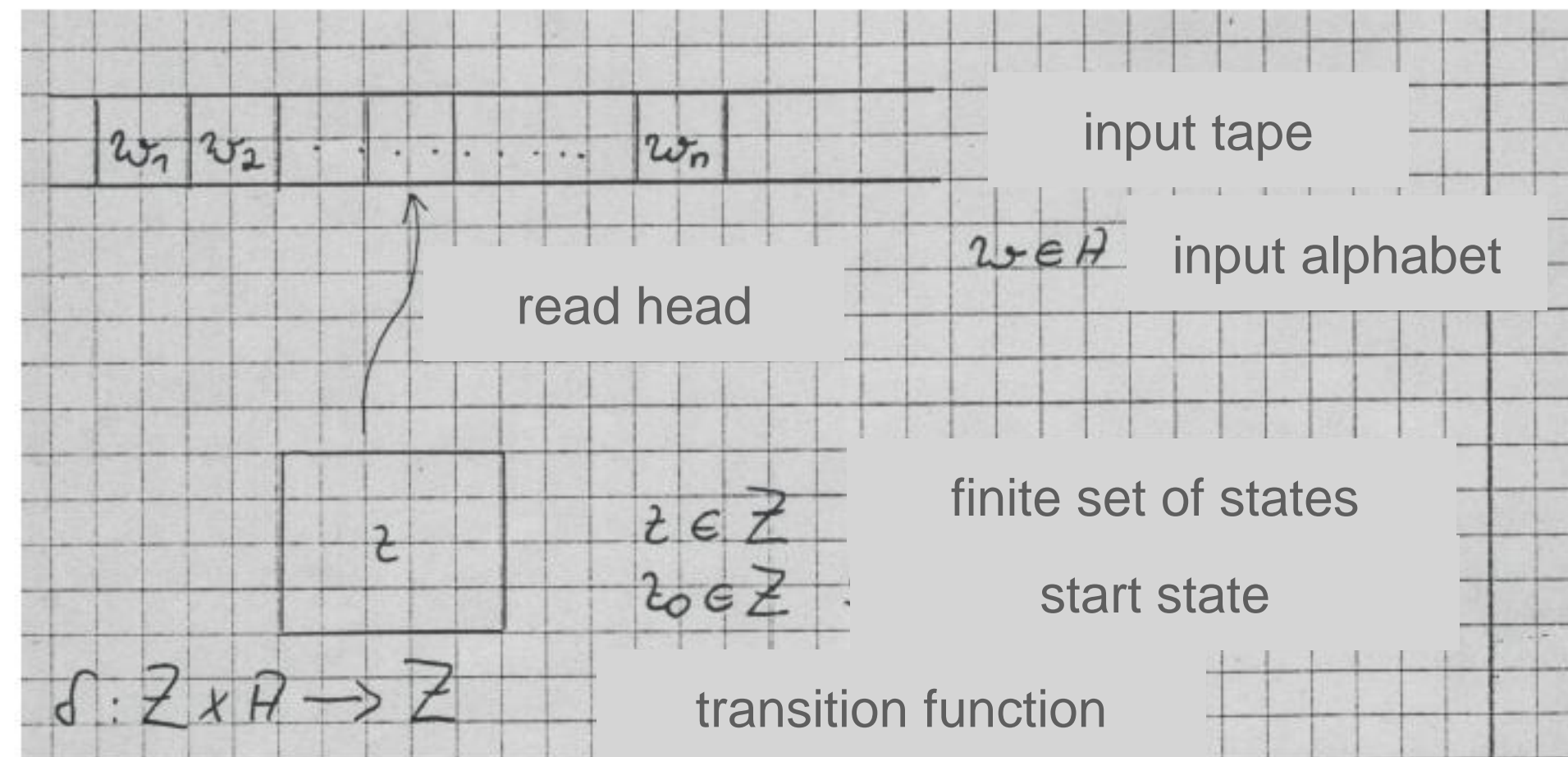


Figure 1: finite automaton

components of model

- input tape (THE stone age/tape age input device); subdivided in fields
- each field i holds a symbol w_i from an input alphabet A
- finite input $w = w_1 \dots w_n \in A^*$

I used $w_i = \text{input in cycle } i$, possibly $i = 0, 1, \dots$

- finite control with set of states Z
- transition function

$$\delta : Z \times A \rightarrow Z$$

- initial state $z_0 \in Z$ (after reset)
- set of accepting states Z_A

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

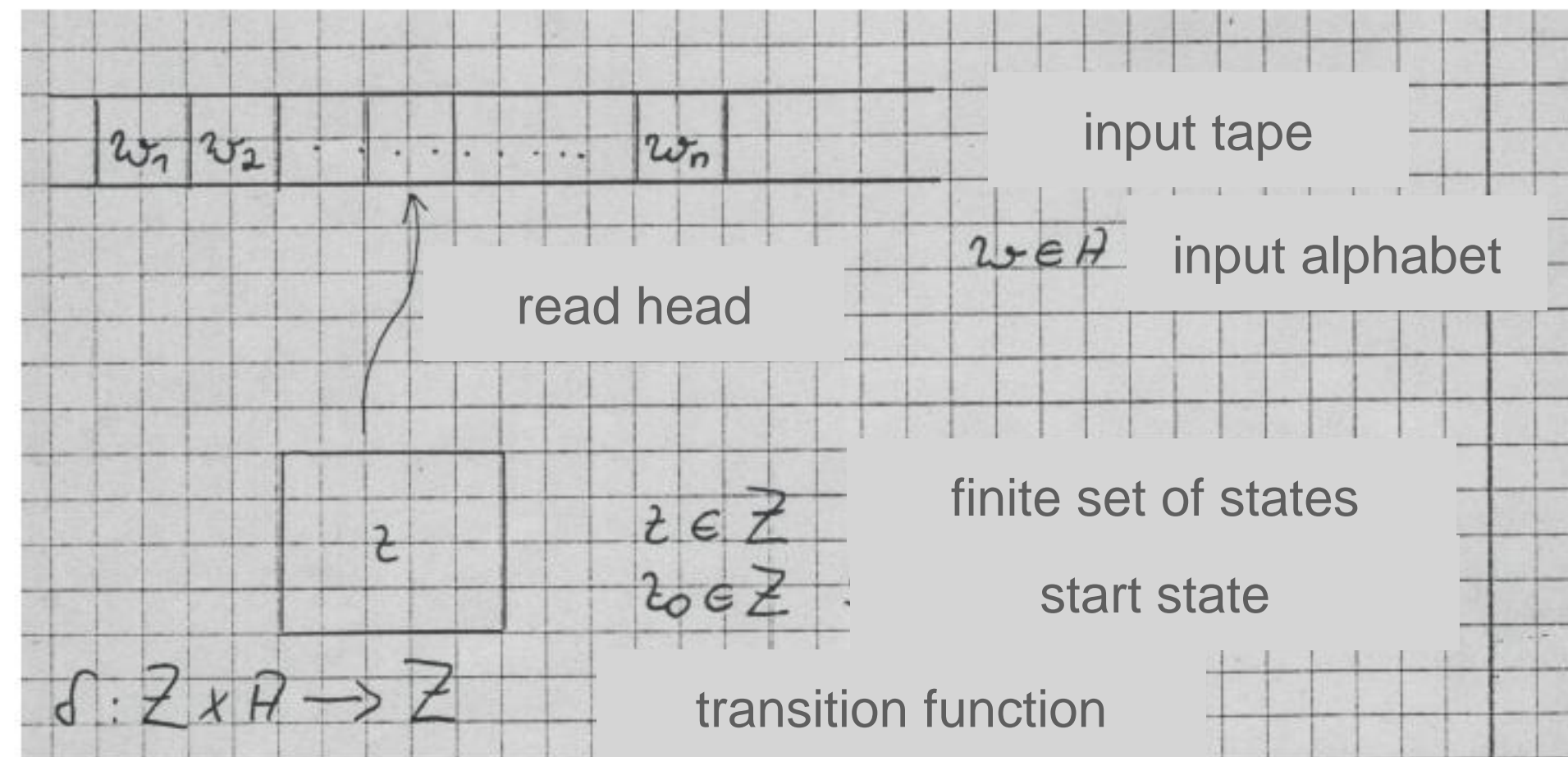


Figure 1: finite automaton

components of model

- input tape (THE stone age/tape age input device); subdivided in fields
- each field i holds a symbol w_i from an input alphabet A
- finite input $w = w_1 \dots w_n \in A^*$

I used $w_i = \text{input in cycle } i$, possibly $i = 0, 1, \dots$

- finite control with set of states Z
- transition function

$$\delta : Z \times A \rightarrow Z$$

- initial state $z_0 \in Z$ (after reset)
- set of accepting states Z_A

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

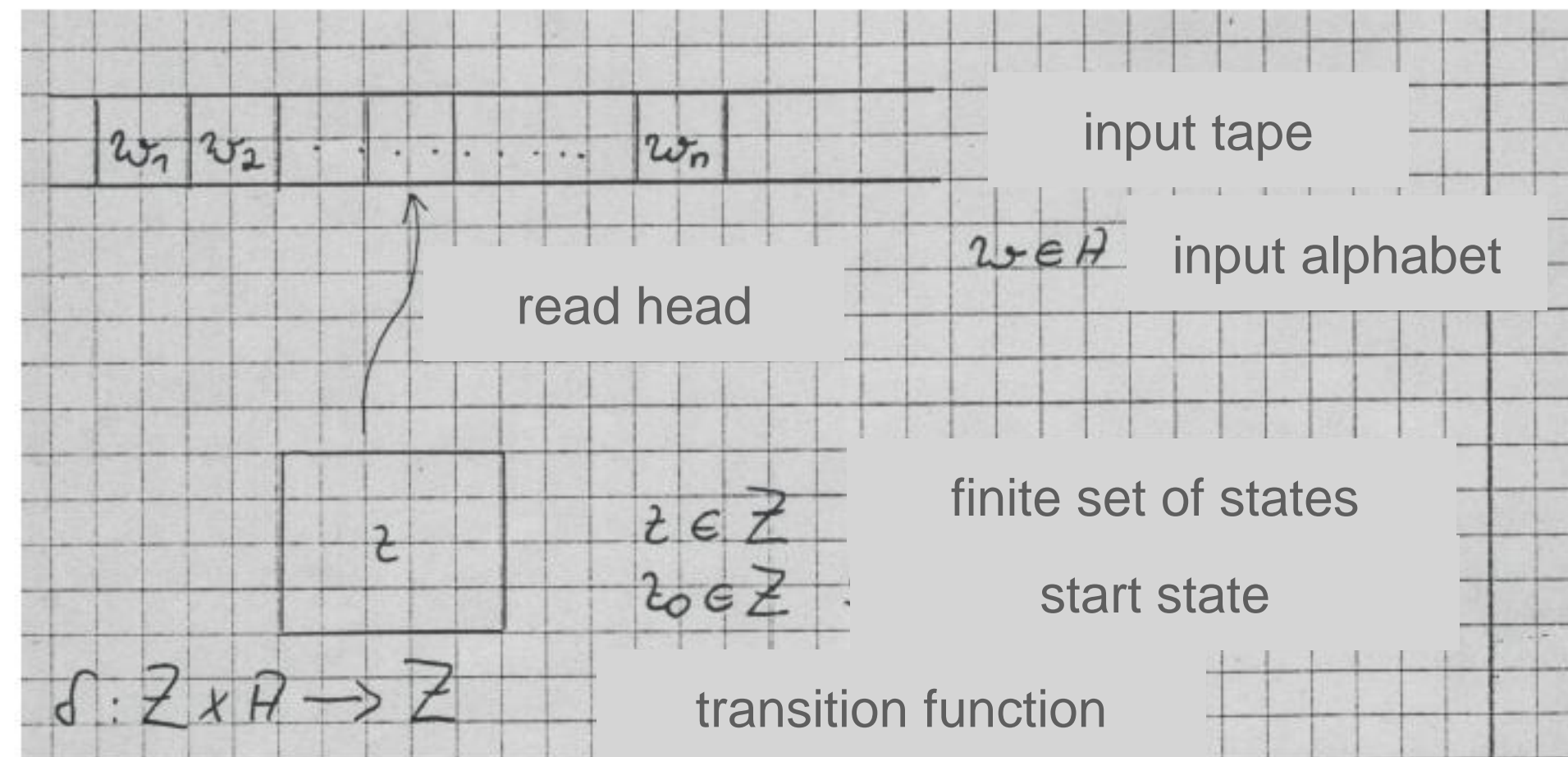


Figure 1: finite automaton

components of model

- input tape (THE stone age/tape age input device); subdivided in fields
- each field i holds a symbol w_i from an input alphabet A
- finite input $w = w_1 \dots w_n \in A^*$

I used $w_i = \text{input in cycle } i$, possibly $i = 0, 1, \dots$

- finite control with set of states Z
- transition function
- initial state $z_0 \in Z$ (after reset)
- set of accepting states Z_A

$$\delta : Z \times A \rightarrow Z$$

machine will accept or reject an input w by ending in a state $s \in Z_A$ or in a state $s \in Z \setminus Z_A$. So essentially we are computing predicates resp. 0/1 valued functions. The general idea is to study the complexity of computations and we do not want to deal with the time space to 'print' (from the print age) the answer of the computation

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

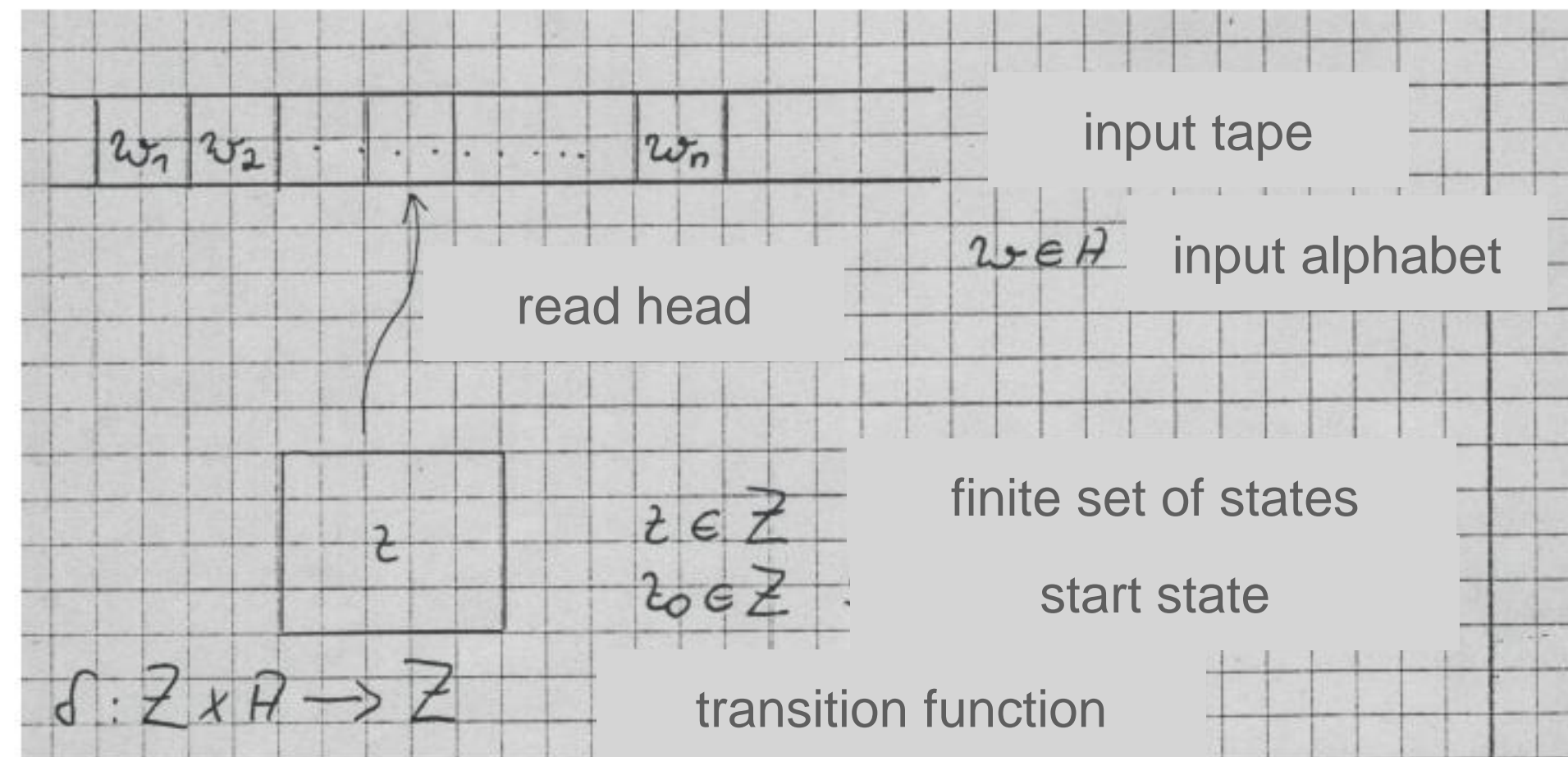


Figure 1: finite automaton

components of model

- input tape (THE stone age/tape age input device); subdivided in fields
- each field i holds a symbol w_i from an input alphabet A
- finite input $w = w_1 \dots w_n \in A^*$

I used $w_i = \text{input in cycle } i$, possibly $i = 0, 1, \dots$

- finite control with set of states Z
- transition function
- initial state $z_0 \in Z$ (after reset)
- set of accepting states Z_A

$$\delta : Z \times A \rightarrow Z$$

1 Step:

- if automaton is in state s and reads input $a \in A$, then it goes to state $\delta(s, a)$ and moves the head 1 field to the right.

machine will accept or reject an input w by ending in a state $s \in Z_A$ or in a state $s \in Z \setminus Z_A$. So essentially we are computing predicates resp. 0/1 valued functions. The general idea is to study the complexity of computations and we do not want to deal with the time space to 'print' (from the print age) the answer of the computation

3 Finite Automata

a historical model - from the time when your grand parents were young

3.1 Intuition

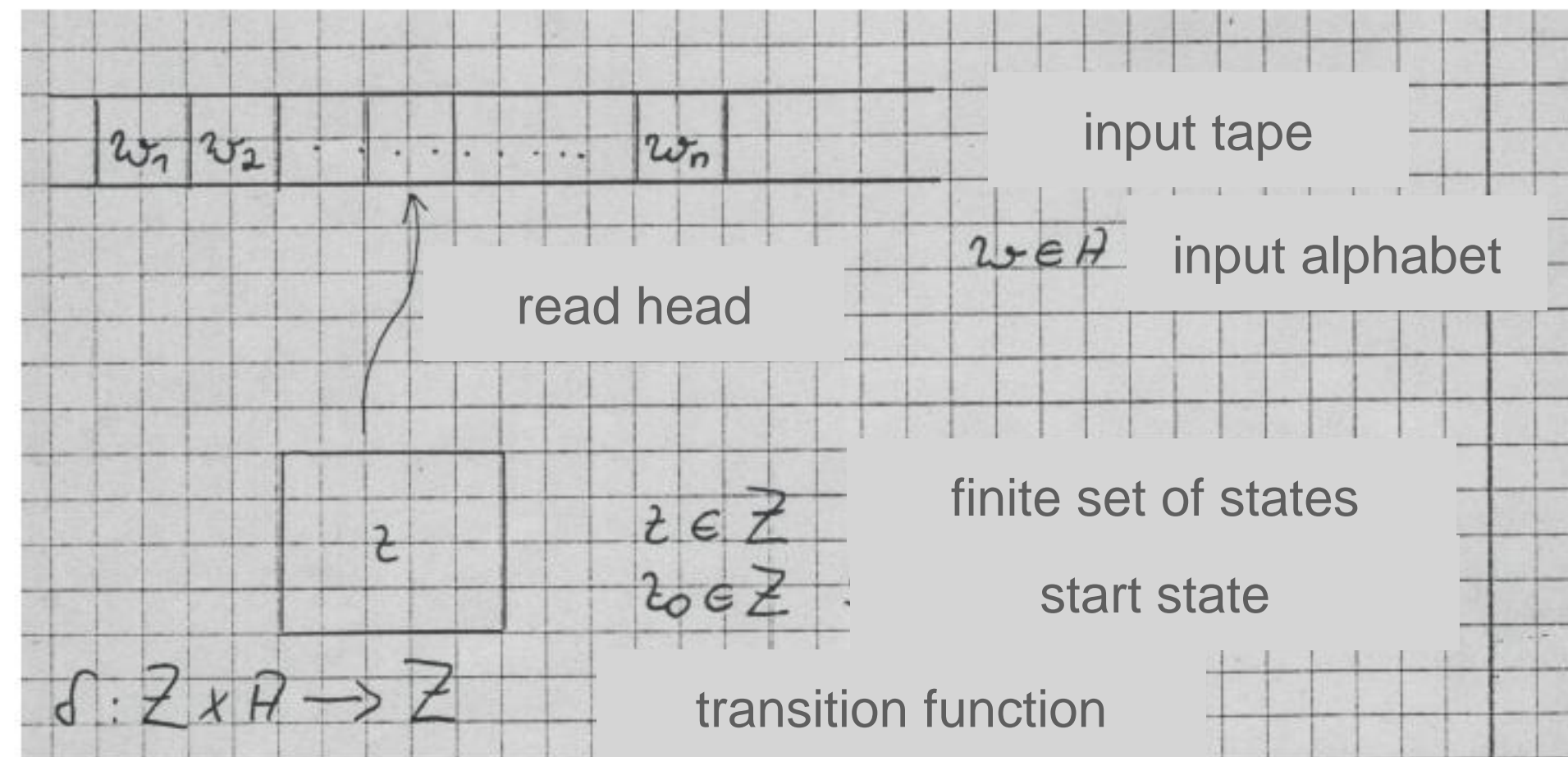


Figure 1: finite automaton

examples, that you know already

- clocked circuits, our hardware model
- MIPS machine
- various control automata, e.g. in constructions for the hardware design lab
- pattern matching automation in I2EA
- **not** the abstract C machine: stack and heap both writeable and unbounded

1 Step:

- if automaton is in state s and reads input $a \in A$, then it goes to state $\delta(s, a)$ and moves the head 1 field to the right.

3.1 Intuition

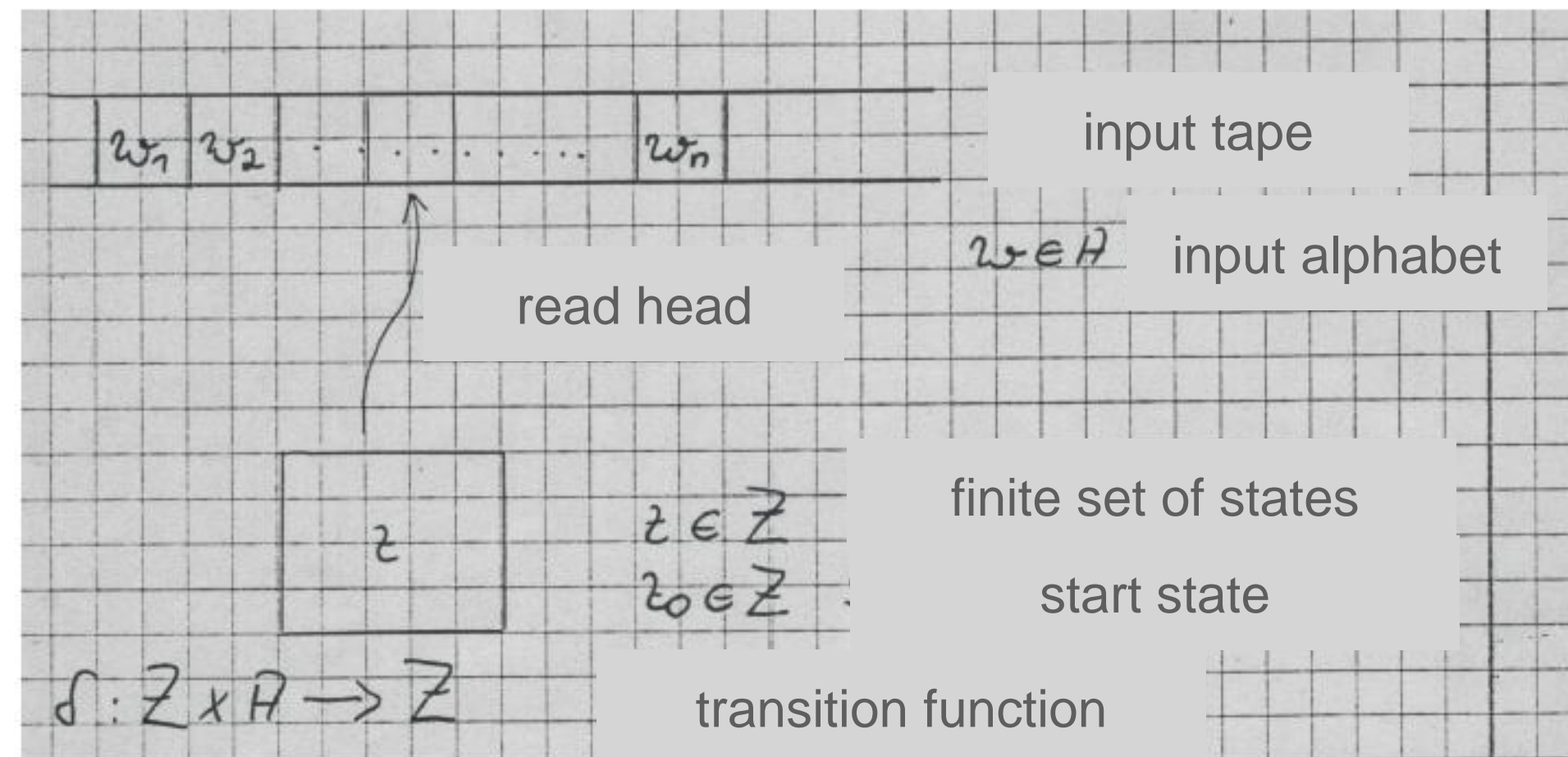


Figure 1: finite automaton

formal definition: (deterministic) finite automaton (fa/dfa) M :

$$M = (Z, A, \delta, z_0, Z_A)$$

- A finite input/tape alphabet
- Z finite set of states
- transition function
- $z_0 \in Z$ initial state
- $Z_A \subseteq Z$ set of accepting states

$$\delta: Z \times A \rightarrow Z$$

1 Step:

- if automaton is in state s and reads input $a \in A$, then it goes to state $\delta(s, a)$ and moves the head 1 field to the right.

3.1 Intuition

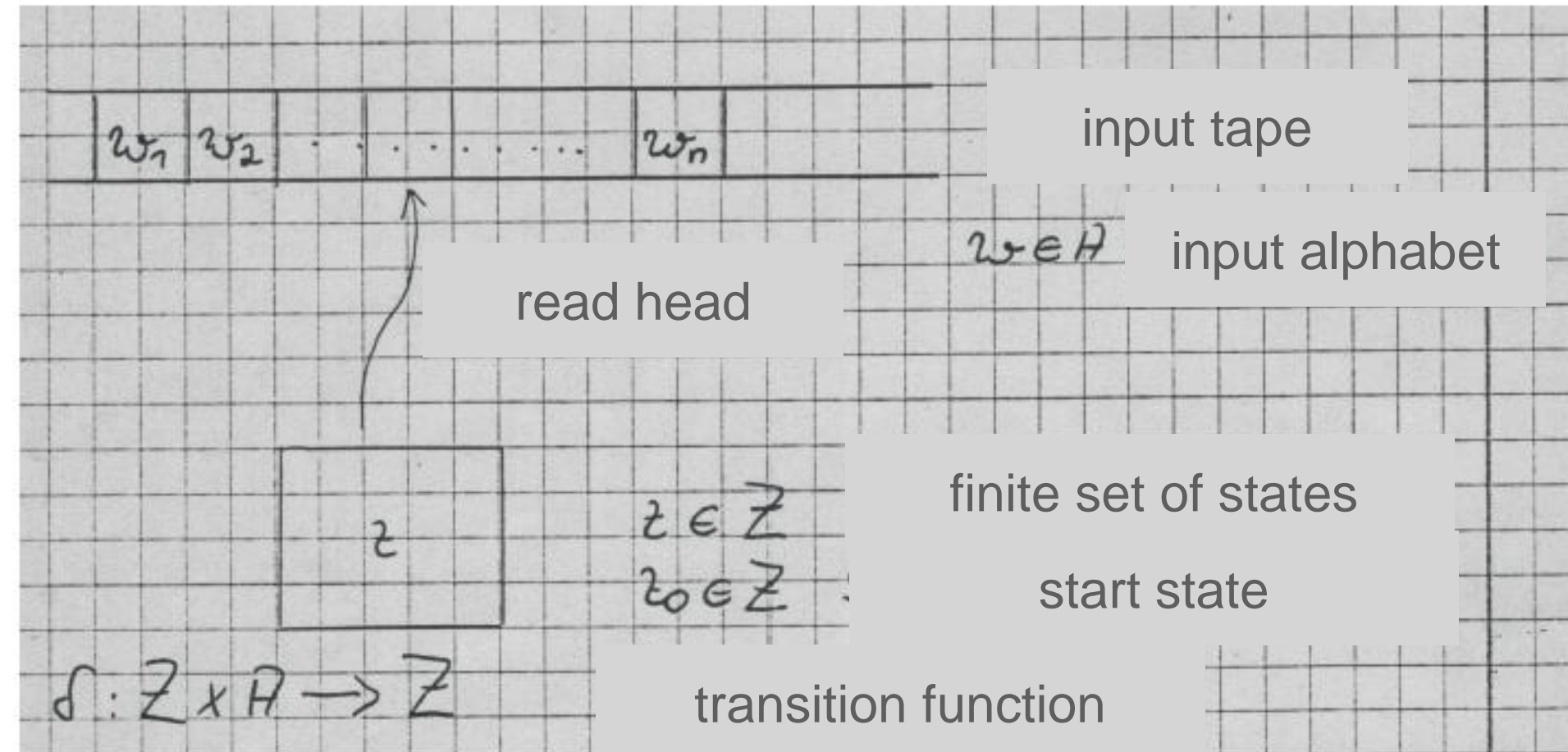


Figure 1: finite automaton

formal definition: (deterministic) finite automaton (fa/dfa) M :

$$M = (Z, A, \delta, z_0, Z_A)$$

- A finite input/tape alphabet
- Z finite set of states
- transition function

$$\delta : Z \times A \rightarrow Z$$

- $z_0 \in Z$ initial state
- $Z_A \subseteq Z$ set of accepting states

semantics

- set of configurations

$$K = Z \times A^*$$

where for $(s, w) \in K$

s : current state , w : remaining input

- transition relation $\vdash \subseteq K \times K$ (here a transition function $\vdash : K \rightarrow K$)

$k \vdash k'$: k' is THE successor configuration of k

If

$$k = (z, w_1 \dots w_n) = (z, w)$$

then

$$\begin{aligned} k' &= (\delta(z, w_1), w_2 \dots w_n) \\ &= (\delta(z, hd(w)), tail(w)) \end{aligned}$$

question: does this allow the head to move left or stay in place?

why?

3.1 Intuition

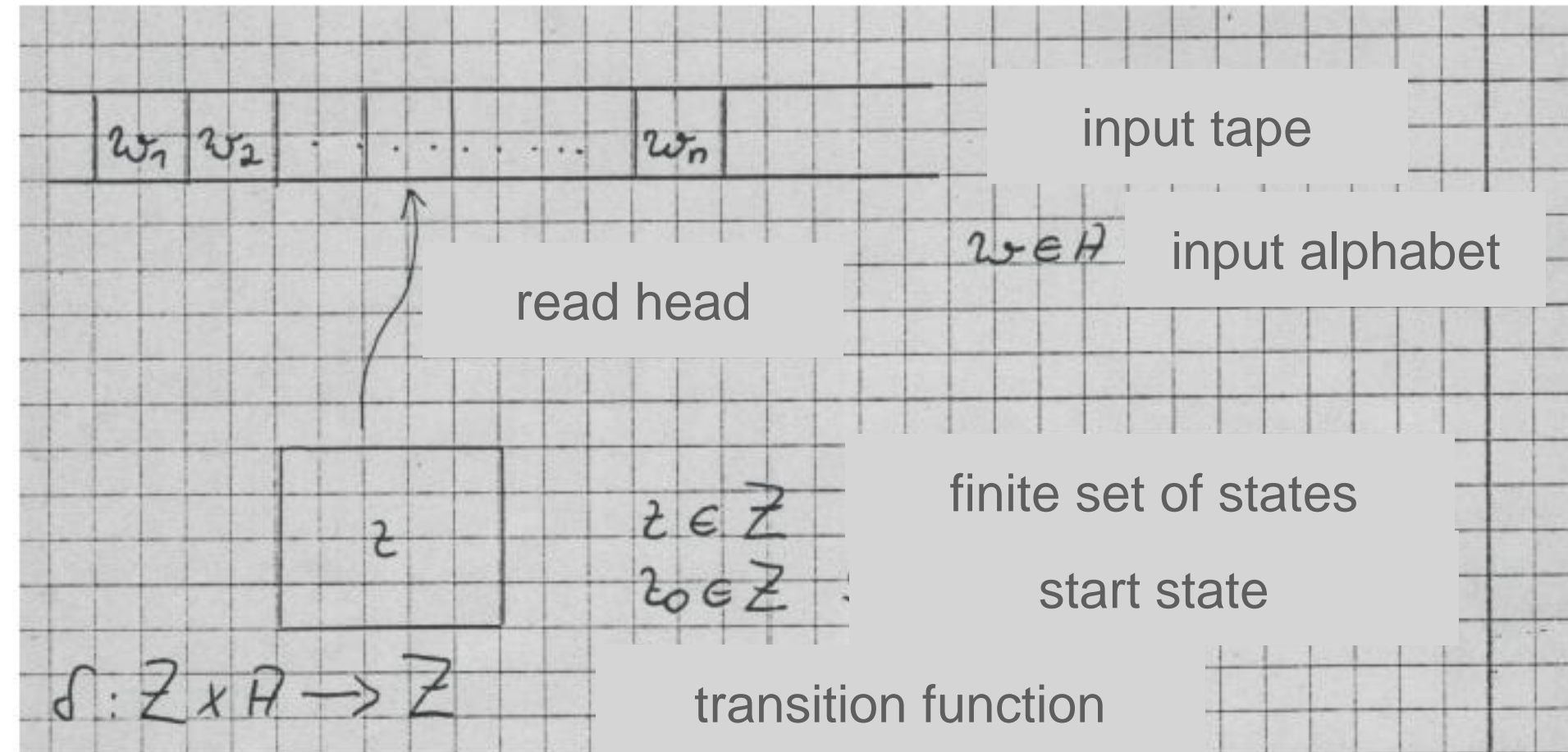


Figure 1: finite automaton

formal definition: (deterministic) finite automaton (fa/dfa) M :

$$M = (Z, A, \delta, z_0, Z_A)$$

- A finite input/tape alphabet
- Z finite set of states
- transition function
- $z_0 \in Z$ initial state
- $Z_A \subseteq Z$ set of accepting states

$$\delta : Z \times A \rightarrow Z$$

semantics

- set of configurations

$$K = Z \times A^*$$

where for $(s, w) \in K$

s : current state , w : remaining input

- transition relation $\vdash \subseteq K \times K$ (here a transition function $\vdash : K \rightarrow K$)

$k \vdash k'$: k' is THE successor configuration of k

If

$$k = (z, w_1 \dots w_n) = (z, w)$$

then

$$\begin{aligned} k' &= (\delta(z, w_1), w_2 \dots w_n) \\ &= (\delta(z, hd(w)), tail(w)) \end{aligned}$$

- computation: sequence of configurations (k^0, k^1, \dots, k^t) with

$$k^i \vdash k^{i+1} \text{ for all } i \geq 0$$

- the computation of M started with $w \in A^n$: this is the computation (k^0, k^1, \dots, k^n) started with

$$k^0 = (z_0, w)$$

and ending with

$$k^n = (z', \epsilon)$$

3.1 Intuition

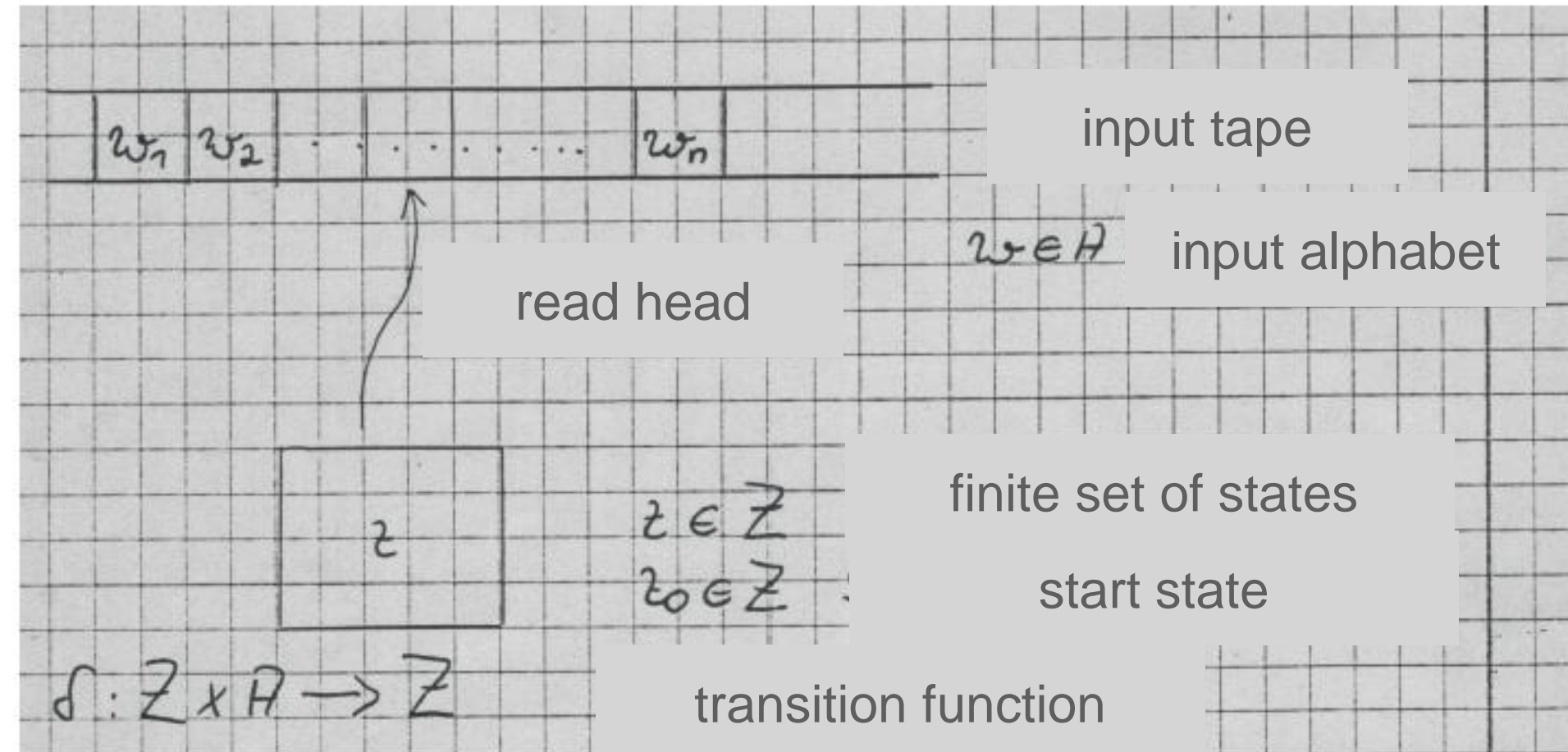


Figure 1: finite automaton

- $M \text{ accepts } w \iff z' \in Z_A$

- language $L(M)$ accepted by M

$$L(M) = \{w : M \text{ accepts } w\} \subseteq A^*$$

semantics

- set of configurations

$$K = Z \times A^*$$

where for $(s, w) \in K$

s : current state , w : remaining input

- transition relation $\vdash \subseteq K \times K$ (here a transition function $\vdash : K \rightarrow K$)

$k \vdash k'$: k' is THE successor configuration of k

If

$$k = (z, w_1 \dots w_n) = (z, w)$$

then

$$\begin{aligned} k' &= (\delta(z, w_1), w_2 \dots w_n) \\ &= (\delta(z, \text{hd}(w)), \text{tail}(w)) \end{aligned}$$

- computation: sequence of configurations (k^0, k^1, \dots, k^t) with

$$k^i \vdash k^{i+1} \text{ for all } i \geq 0$$


- the computation of M started with $w \in A^n$: this is the computation (k^0, k^1, \dots, k^n) started with

$$k^0 = (z_0, w)$$


and ending with

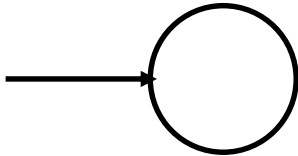
$$k^n = (z', \epsilon)$$

3.2 Visualizing finite automata by labelled graphs


- states $z \in Z$ 

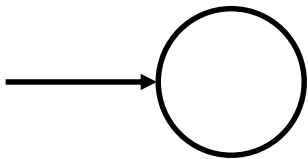
3.2 Visualizing finite automata by labelled graphs

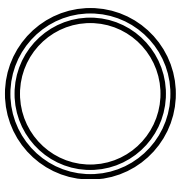
- states $z \in Z$ 

- initial state: 


3.2 Visualizing finite automata by labelled graphs

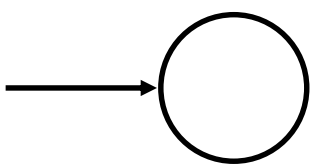
- states $z \in Z$ 

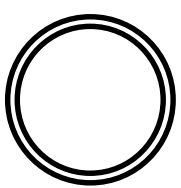
- initial state: 

- final state: 

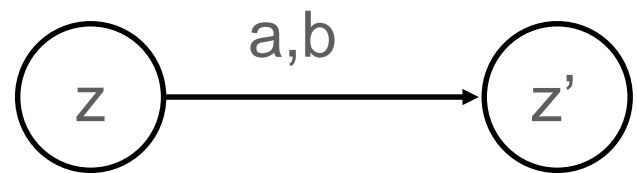
3.2 Visualizing finite automata by labelled graphs

- states $z \in Z$ 


- initial state: 

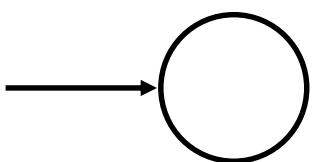
- final state: 

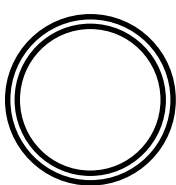
- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)



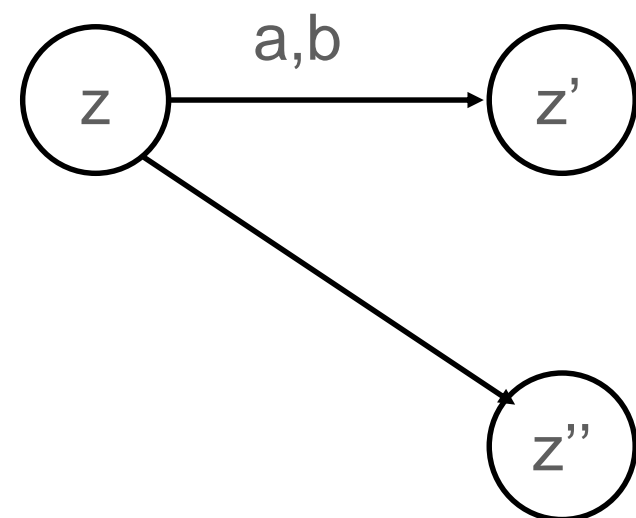
3.2 Visualizing finite automata by labelled graphs

- states $z \in Z$ 

- initial state: 

- final state: 

- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)




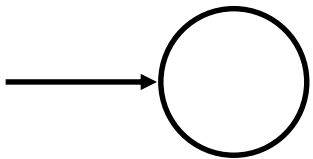
- $\delta(z, c) = z''$ for all other c : (else)

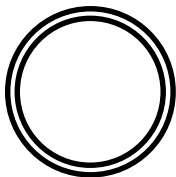
have you seen this?

if yes, where?

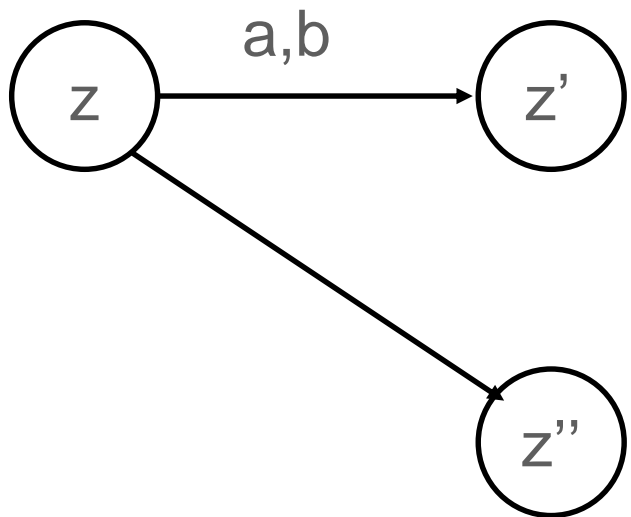
3.2 Visualizing finite automata by labelled graphs

- states $z \in Z$ 

- initial state: 

- final state: 

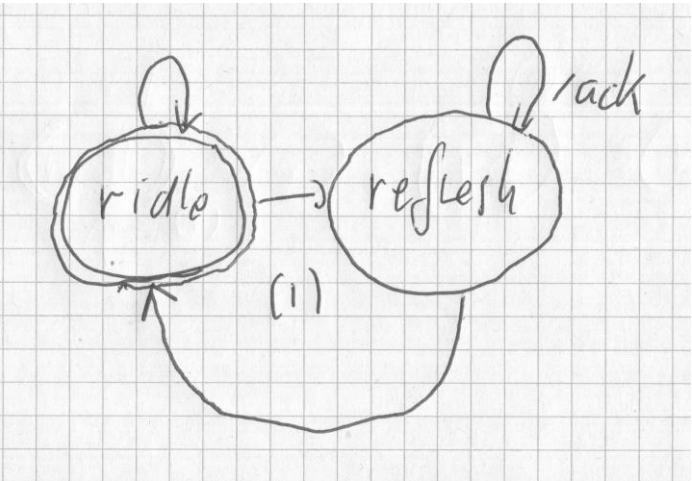
- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)



- $\delta(z, c) = z''$ for all other c : (else)

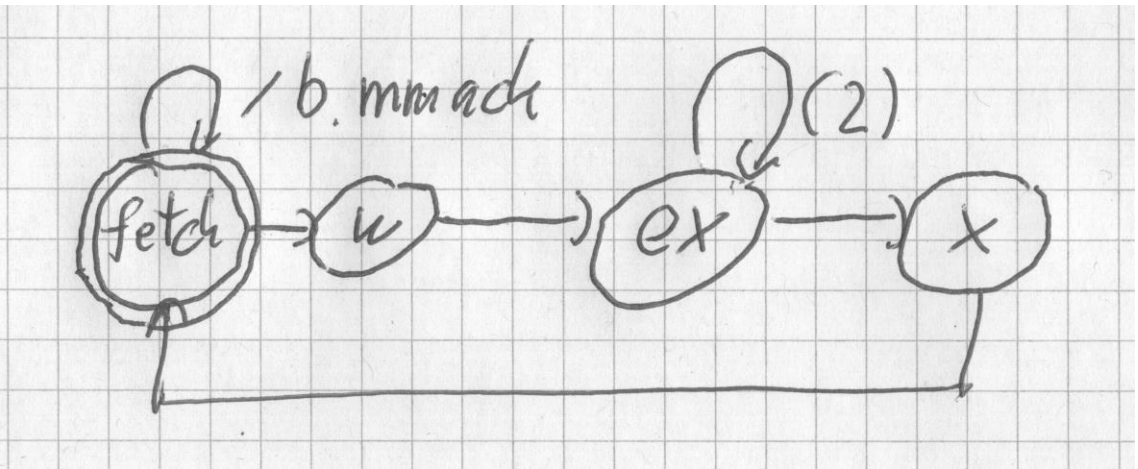
hardware lab

DRAM controller




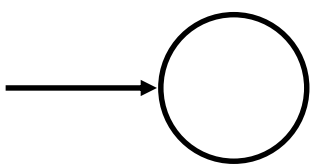
have you seen this?

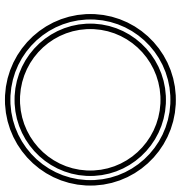
if yes, where?



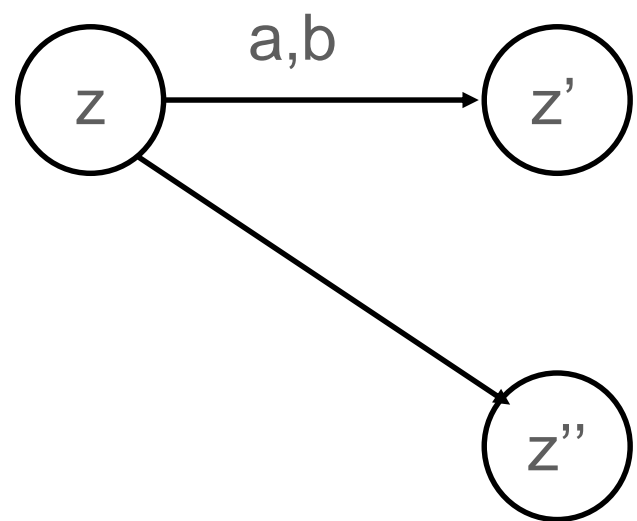
example

- states $z \in Z$ 

- initial state: 

- final state: 

- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)



- $\delta(z, c) = z''$ for all other c : (else)

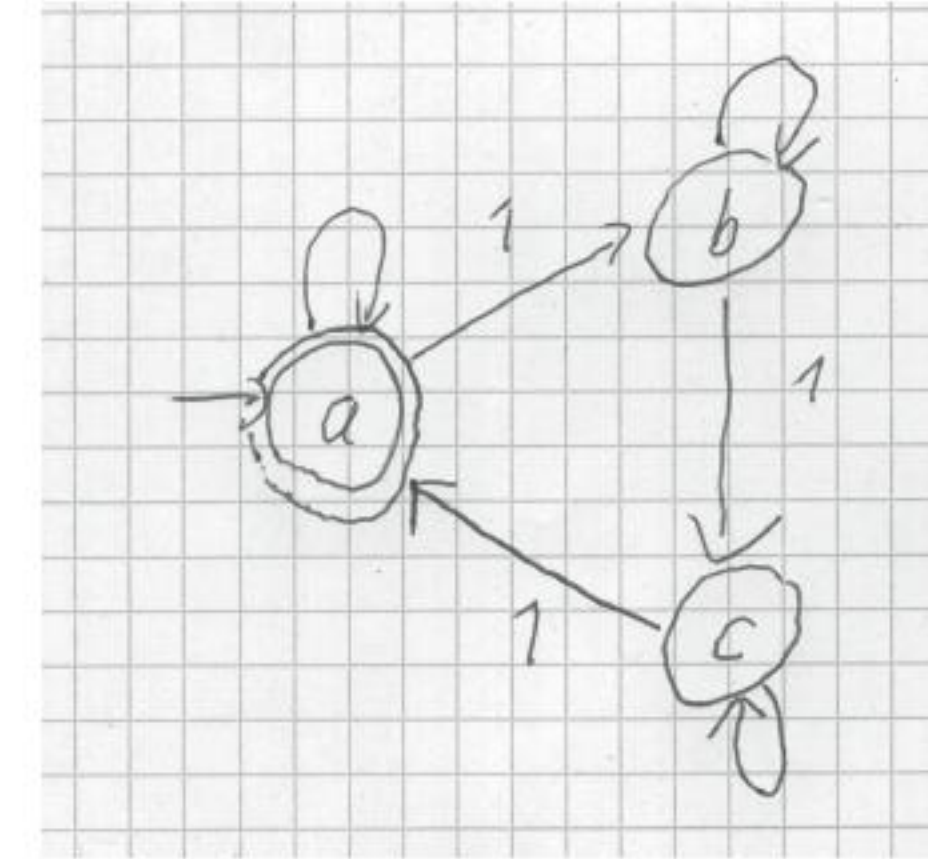


Figure 2: counting ones mod 3 with a finite automaton

$$Z = \{a, b, c\}$$

$$A = \mathbb{B}$$


$$z_0 = a$$

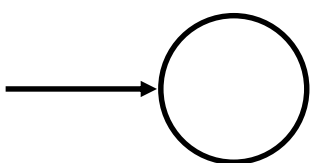
$$Z_A = \{a\}$$

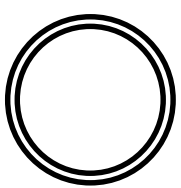
$$\delta(a, x) = \begin{cases} b & x = 1 \\ a & x = 0 \end{cases}$$

$$\delta(b, x) = \begin{cases} c & x = 1 \\ b & x = 0 \end{cases}$$

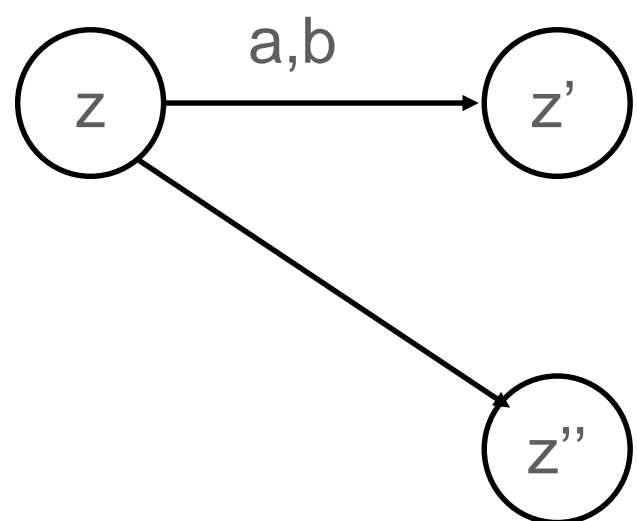
$$\delta(c, x) = \begin{cases} a & x = 1 \\ c & x = 0 \end{cases}$$

- states $z \in Z$ 

- initial state: 

- final state: 

- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)



- $\delta(z, c) = z''$ for all other c : (else)

example

$$A = \{1\} \quad , \quad L = \{1^i : 7i\}$$

reminder

$$a^i = a \circ \dots \circ a \quad (i \text{ times})$$

- L is regular

$$L = \{1^7\}^*$$

- automaton accepting L

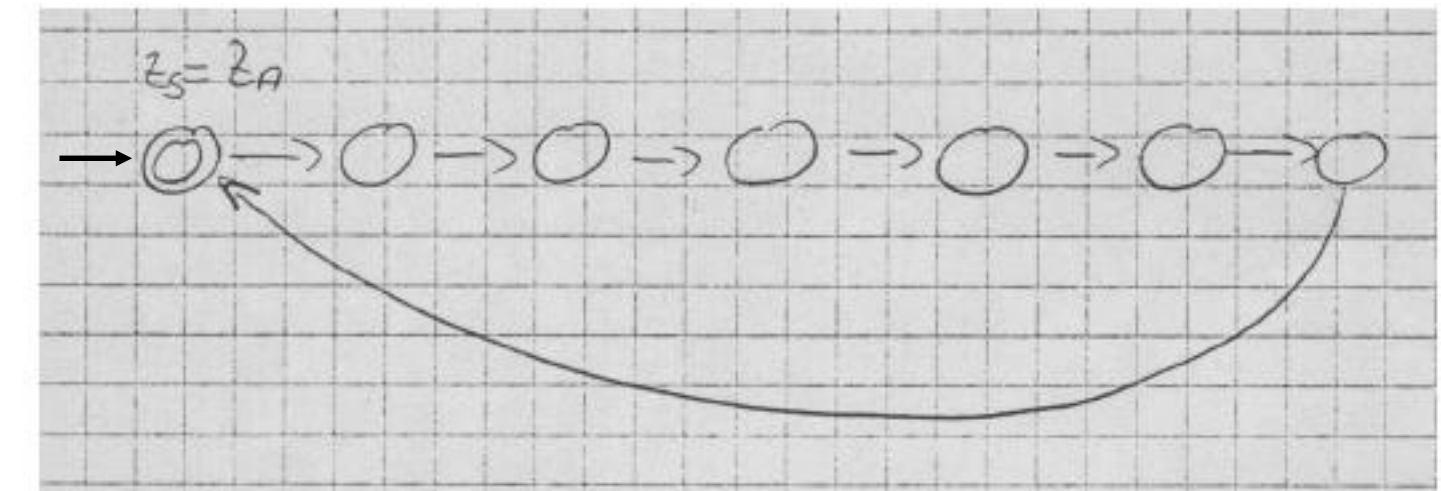

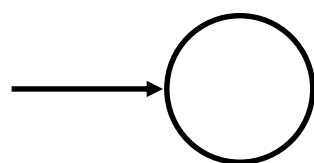
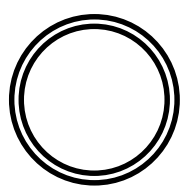


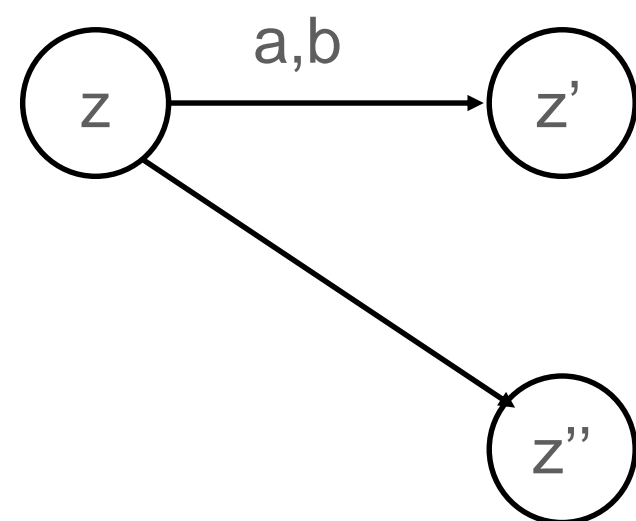
Figure 3: counting mod 7 with a finite automaton

- states $z \in Z$ 

- initial state: 

- final state: 

- $\delta(z, a) = \delta(z, b) = z'$: (if input symbol =)



- $\delta(z, c) = z''$ for all other c : (else)

example

$$A = \{1\} \quad , \quad L = \{1^i : 7i\}$$

reminder

$$a^i = a \circ \dots \circ a \quad (i \text{ times})$$

- L is regular

$$L = \{1^7\}^*$$

- automaton accepting L

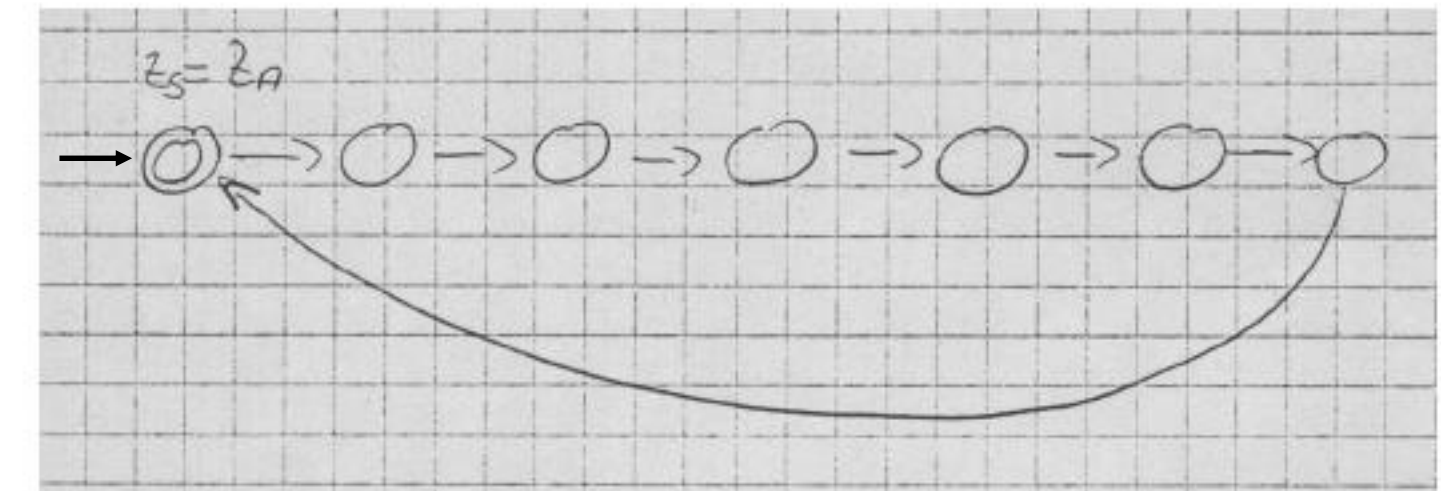


Figure 3: counting mod 7 with a finite automaton

-

$$\begin{aligned}
 L' &= \{1^{7i+25j} : i, j \in \mathbb{N}_0\} \\
 &= (\{1^7\} \cup \{1^{25}\})^* \quad (\text{thus it is regular}) \\
 M &= ??
 \end{aligned}$$

Lemma 1. *Regular languages are exactly the languages accepted by finite automata*

$$L \in R(A) \quad \leftrightarrow \quad \exists \text{ finite automaton } M. L(M) = L$$

this one is NOT obvious

Lemma 1. *Regular languages are exactly the languages accepted by finite automata*

$$L \in R(A) \quad \leftrightarrow \quad \exists \text{ finite automaton } M. L(M) = L$$

Lemma 2. *if M is finite automaton, then $L(M)$ is regular*

Lemma 1. *Regular languages are exactly the languages accepted by finite automata*

$$L \in R(A) \iff \exists \text{ finite automaton } M. L(M) = L$$

Lemma 2. *if M is finite automaton, then $L(M)$ is regular*

-

$$M = (Z, A, \delta, z_s, Z_A)$$

w.l.o.g. (without loss of generality)

$$Z = \{1, \dots, n\}$$

otherwise rename states

-

$$R(i, j, k) = \{w : \exists \text{ computation } (i, w) \vdash \dots \vdash (j, \epsilon) \\ \text{and all configurations between } (i, w) \text{ and } (j, \epsilon) \\ \text{have states } \leq k\}$$

Lemma 1. *Regular languages are exactly the languages accepted by finite automata*

$$L \in R(A) \iff \exists \text{ finite automaton } M. L(M) = L$$

Lemma 2. *if M is finite automaton, then $L(M)$ is regular*

-

$$M = (Z, A, \delta, z_s, Z_A)$$

w.l.o.g. (without loss of generality)

$$Z = \{1, \dots, n\}$$

otherwise rename states

-

$$R(i, j, k) = \{w : \exists \text{ computation } (i, w) \vdash \dots \vdash (j, \epsilon) \\ \text{and all configurations between } (i, w) \text{ and } (j, \epsilon) \\ \text{have states } \leq k\}$$

Claim: all sets $R(i, j, k)$ are regular. Proof by induction on k

- $k = 0$

$$R(i, j, 0) = \begin{cases} \{a : \delta(i, a) = j\} & i \neq j \\ \{a : \delta(i, a) = i\} \cup \{\epsilon\} & i = j \end{cases}$$

Lemma 1. Regular languages are exactly the languages accepted by finite automata

$$L \in R(A) \iff \exists \text{ finite automaton } M. L(M) = L$$

Lemma 2. if M is finite automaton, then $L(M)$ is regular

• $k - 1 \rightarrow k$

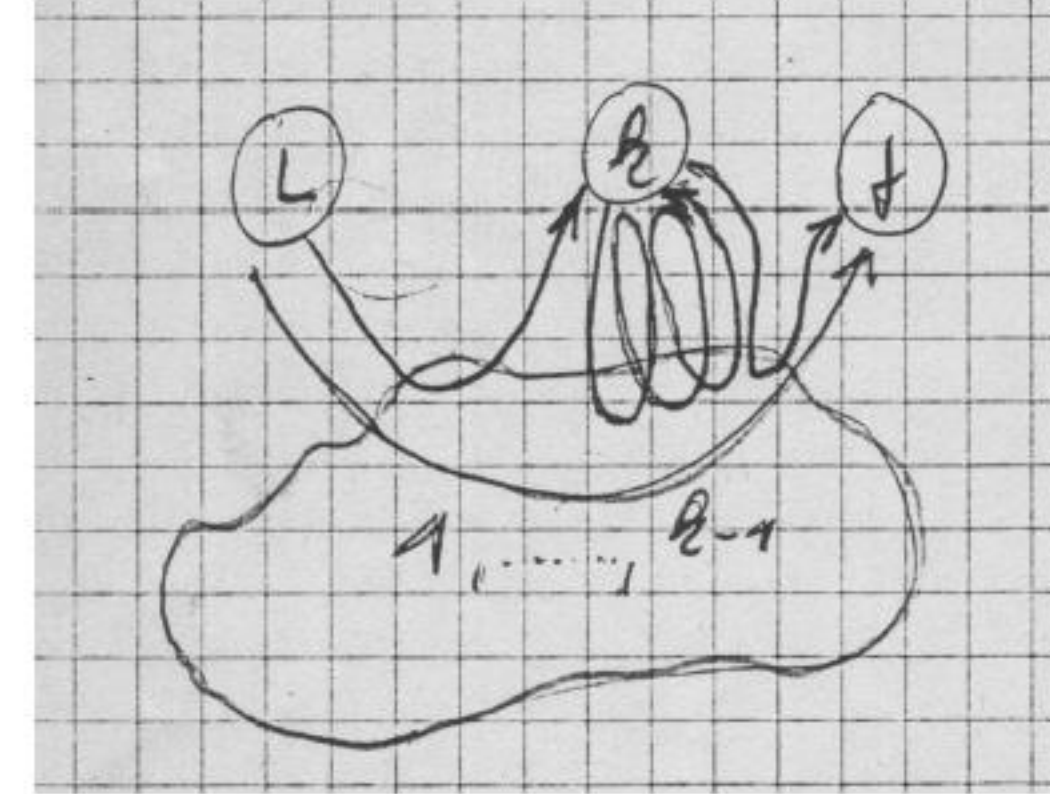


Figure 4: Illustrating the recursive definition of regular sets $R(i, j, k)$

$$R(i, j, k) = R(i, j, k - 1) \cup (R(i, k, k - 1) \circ (R(k, k, k - 1))^* \circ R(k, j, k - 1))$$

$$R(i, j, k) = \{w : \exists \text{ computation } (i, w) \vdash \dots \vdash (j, \epsilon) \\ \text{and all configurations between } (i, w) \text{ and } (j, \epsilon) \\ \text{have states } \leq k\}$$

Claim: all sets $R(i, j, k)$ are regular. Proof by induction on k

• $k = 0$

$$R(i, j, 0) = \begin{cases} \{a : \delta(i, a) = j\} & i \neq j \\ \{a : \delta(i, a) = i\} \cup \{\epsilon\} & i = j \end{cases}$$

Lemma 1. *Regular languages are exactly the languages accepted by finite automata*

$$L \in R(A) \iff \exists \text{ finite automaton } M. L(M) = L$$

Lemma 2. *if M is finite automaton, then $L(M)$ is regular*

• $k - 1 \rightarrow k$

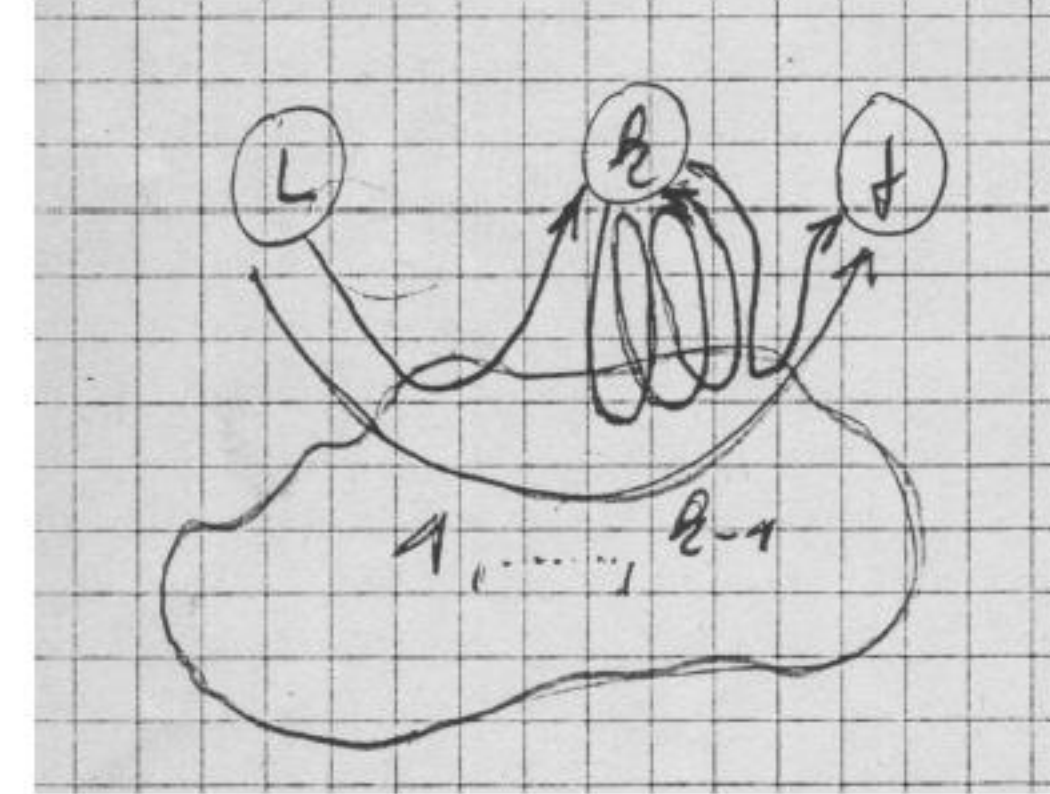


Figure 4: Illustrating the recursive definition of regular sets $R(i, j, k)$

$$R(i, j, k) = R(i, j, k-1) \cup (R(i, k, k-1) \circ (R(k, k, k-1))^* \circ R(k, j, k-1))$$

$$R(i, j, k) = \{w : \exists \text{ computation } (i, w) \vdash \dots \vdash (j, \varepsilon) \text{ and all configurations between } (i, w) \text{ and } (j, \varepsilon) \text{ have states } \leq k\}$$

Claim: all sets $R(i, j, k)$ are regular. Proof by induction on k

• $k = 0$

$$R(i, j, 0) = \begin{cases} \{a : \delta(i, a) = j\} & i \neq j \\ \{a : \delta(i, a) = i\} \cup \{\varepsilon\} & i = j \end{cases}$$

$$L(M) = \bigcup_{z_a \in Z_A} R(z_s, z_a, n)$$