Nondeterministic finite automata

and pumping lemma

4 Nondeterministic finite automata

formal definition: nondeterministic finite automaton (nfa) M:

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

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

  $$\delta : Z \times A \to 2^Z$$

  recall

  $$2^Z = \{A \ : \ A \subseteq Z\}$$

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

1 Step:

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

4 Nondeterministic finite automata

formal definition: nondeterministic finite automaton (nfa) M:

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

have you seen this?

where?

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

  $$\delta : Z \times A \to 2^Z$$

  recall

  $$2^Z = \{A \ : \ A \subseteq Z\}$$

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

1 Step:

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

4 Nondeterministic finite automata

formal definition: nondeterministic finite automaton (nfa) M:

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

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

  $$\delta : Z \times A \to 2^Z$$

  recall

  $$2^Z = \{A \,:\, A \subseteq Z\}$$

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

1 Step:

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

have you seen this?

where?

hardware lab
OS support:

MIPS + disk ISA

I2OS:
C0 + disk

4 Nondeterministic finite automata

formal definition: nondeterministic finite automaton (nfa) M:

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

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

$$\delta : Z \times A \to 2^Z$$

  recall

$$2^Z = \{A \;:\; A \subseteq Z\}$$

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

1 Step:

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

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

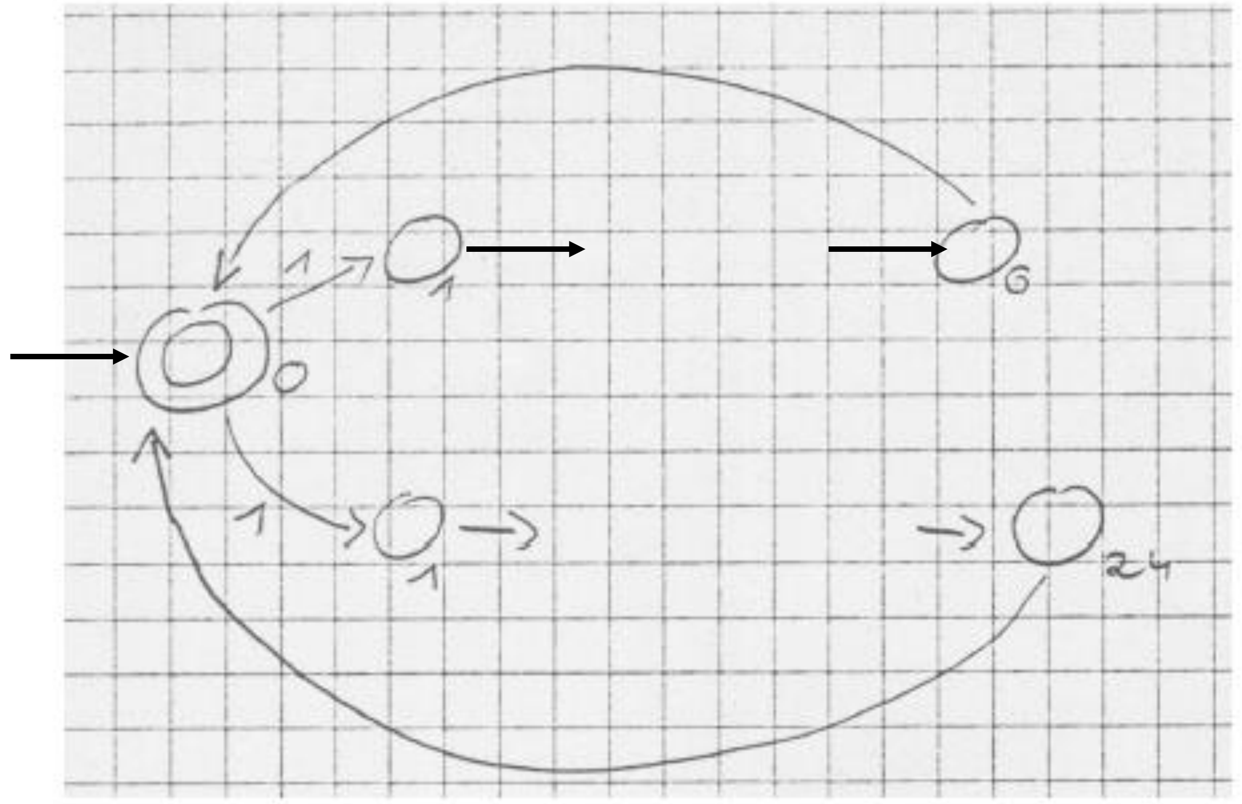


Figure 5: Example of an nda. With input 1^{14} it has accepting and rejecting computations. It accepts this input.

semantics

- set of configurations

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

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

$$s: \text{ current state} \quad , \quad w: \text{ remaining input}$$

- transition relation $\vdash \subseteq K \times K$ (in general not a function)

$$k \vdash k' : \; k' \text{ is a successor configuration of } k$$

  Let

$$k = (z, w_1 \ldots w_n) \quad , \quad k' = (z', w_2 \ldots w_n)$$

  then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, w_1)$$

semantics

- set of configurations
$$K = S \times A^*$$
where for $(s,w) \in K$
$$s: \text{ current state} \quad , \quad w: \text{ remaining input}$$

- transition relation $\vdash \subseteq K \times K$ (in general not a function)
$$k \vdash k' : \; k' \text{ is a successor configuration of } k$$
Let
$$k = (z, w_1 \dots w_n) \quad , \quad k' = (z', w_2 \dots w_n)$$
then
$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, w_1)$$

- a computation of M started with $w \in A^n$: this is a computation $(k^0, k^1, \dots, k^n)$ started with
$$k^0 = (z_0, w)$$
and ending with
$$k^n = (z', \varepsilon)$$
The computations is accepting if $z' \in Z_A$, otherwise it is rejecting.

semantics

- set of configurations

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

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

$$s: \text{ current state} \quad , \quad w: \text{ remaining input}$$

- transition relation $\vdash \subseteq K \times K$ (in general not a function)

$$k \vdash k': \; k' \text{ is a successor configuration of } k$$

Let

$$k = (z, w_1 \dots w_n) \quad , \quad k' = (z', w_2 \dots w_n)$$

then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, w_1)$$

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

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

and ending with

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

The computations is accepting if $z' \in Z_A$, otherwise it is rejecting.

- M accepts w if there exists an accepting computation of M with input w.
- language $L(M)$ accepted by M

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

semantics

- set of configurations
$$K = S \times A^*$$
where for $(s,w) \in K$
$$s: \text{ current state} \quad , \quad w: \text{ remaining input}$$

- transition relation $\vdash \subseteq K \times K$ (in general not a function)
$$k \vdash k': \; k' \text{ is a successor configuration of } k$$
Let
$$k = (z, w_1 \dots w_n) \quad , \quad k' = (z', w_2 \dots w_n)$$
then
$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, w_1)$$

- a computation of M started with $w \in A^n$: this is a computation $(k^0, k^1, \dots, k^n)$ started with
$$k^0 = (z_0, w)$$
and ending with
$$k^n = (z', \varepsilon)$$
The computations is accepting if $z' \in Z_A$, otherwise it is rejecting.

- M accepts w if there exists an accepting computation of M with input w.
- language $L(M)$ accepted by M
$$L(M) = \{w \; : \; M \text{ accepts } w\} \subseteq A^*$$

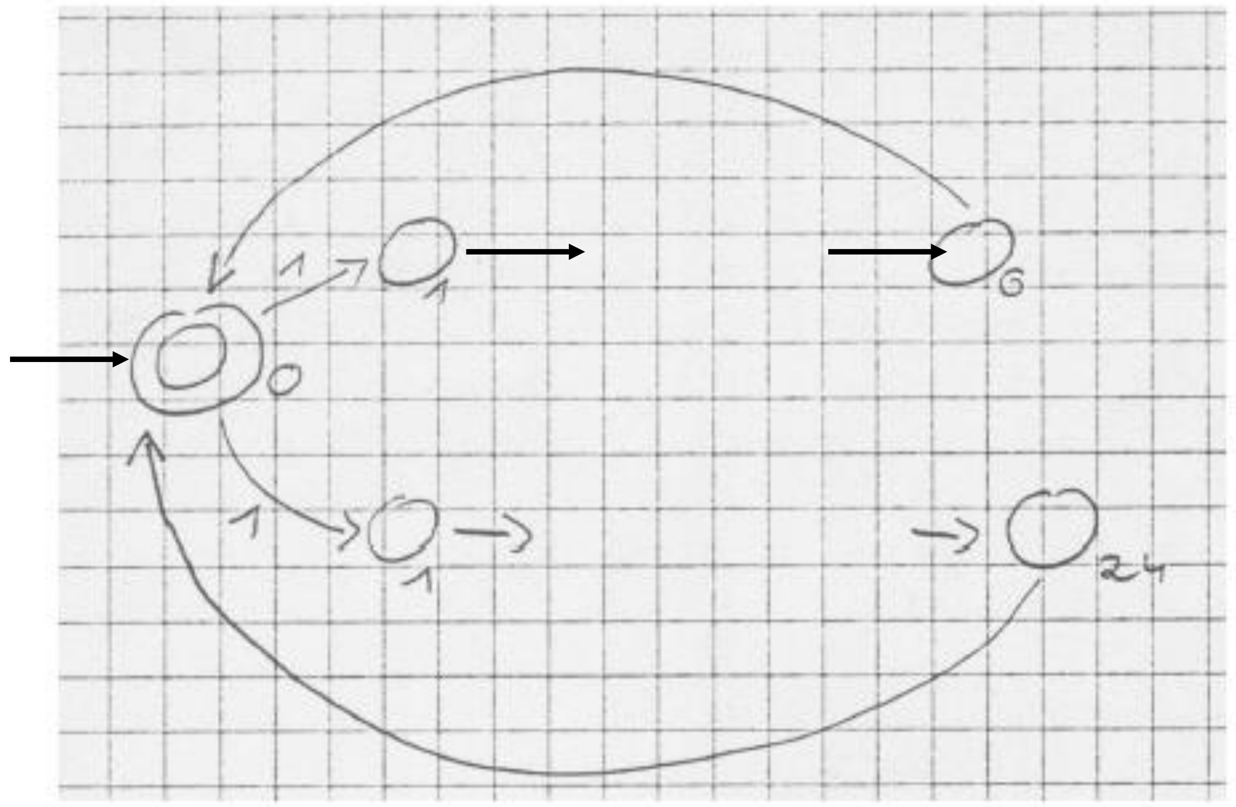


Figure 5: Example of an nda. With input 1^{14} it has accepting and rejecting computations. It accepts this input.

eliminating nondeterminism
(at a price)

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

eliminating nondeterminism
(at a price)

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

  we construct deterministic finite automaton

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

  such that

  $$L(M) = L(M')$$

eliminating nondeterminism
(at a price)

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

we construct deterministic finite automaton

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

such that

$$L(M) = L(M')$$

- 'power set construction'

$$\begin{aligned} Z' &= 2^Z \\ z_0' &= \{z_0\} \\ \delta(Y,a) &= \bigcup_{z \in Y} \delta(z,a) \\ Z_A' &= \{Y \; : \; Y \cap Z_a \neq \emptyset\} \end{aligned}$$

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

we construct deterministic finite automaton

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

such that

$$L(M) = L(M')$$

- 'power set construction'

$$\begin{aligned} Z' &= 2^Z \\ z_0' &= \{z_0\} \\ \delta(Y,a) &= \bigcup_{z \in Y} \delta(z,a) \\ Z_A' &= \{Y \;:\; Y \cap Z_a \neq \emptyset\} \end{aligned}$$

correctness:

- define $k \vdash^n k'$ meaning: k' is a possible successor configuration of k after n steps.

$$k \vdash^0 k' \leftrightarrow k = k'$$

$$k \vdash^{n+1} k' \quad \leftrightarrow \quad \exists k''.\ k \vdash^n k'' \vdash k'$$

also for all future models of computation

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

we construct deterministic finite automaton

$$M = (Z', A, \delta', z'_0, Z'_A)$$

such that

$$L(M) = L(M')$$

- 'power set construction'

$$\begin{aligned} Z' &= 2^Z \\ z'_0 &= \{z_0\} \\ \delta(Y,a) &= \bigcup_{z \in Y} \delta(z,a) \\ Z'_A &= \{Y \,:\, Y \cap Z_a \neq \emptyset\} \end{aligned}$$

correctness:

- define $k \vdash^n k'$ meaning: k' is a possible successor configuration of k after n steps.

$$k \vdash^0 k' \leftrightarrow k = k'$$

$$k \vdash^{n+1} k' \quad \leftrightarrow \quad \exists k''.\ k \vdash^n k'' \vdash k'$$

- Claim: if

$$(z'_0, w) \vdash^n_{M'} (Y, w')$$

then

$$Y = \{z \in Z \,:\, (z_0, w) \vdash^n_M (z, w')\}$$

The state Y after n steps of deterministic computation is the set of all states reachable by n steps of nondeterministic computation. Note: if $w = w_1 \ldots w_n w_{n+1} \ldots w_s$ then $w' = w_{n+1} \ldots w_s$. Proof by induction on n

correctness:

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

we construct deterministic finite automaton

$$M = (Z', A, \delta', z'_0, Z'_A)$$

such that

$$L(M) = L(M')$$

- 'power set construction'

$$\begin{aligned} Z' &= 2^Z \\ z'_0 &= \{z_0\} \\ \delta(Y,a) &= \bigcup_{z \in Y} \delta(z,a) \\ Z'_A &= \{Y \,:\, Y \cap Z_a \neq \emptyset\} \end{aligned}$$

- define $k \vdash^n k'$ meaning: k' is a possible successor configuration of k after n steps.

$$k \vdash^0 k' \leftrightarrow k = k'$$

$$k \vdash^{n+1} k' \quad \leftrightarrow \quad \exists k''.\, k \vdash^n k'' \vdash k'$$

- Claim: if

$$(z'_0, w) \vdash^n_{M'} (Y, w')$$

then

$$Y = \{z \in Z \,:\, (z_0, w) \vdash^n_M (z, w')\}$$

The state Y after n steps of deterministic computation is the set of all states reachable by n steps of nondeterministic computation. Note: if $w = w_1 \dots w_n w_{n+1} \dots w_s$ then $w' = w_{n+1} \dots w_s$. Proof by induction on n

- $n = 0$ trivial

correctness:

Lemma 3. *For every nfa M there exists a dfa M' which accepts the same language*

$$L(M) = L(M')$$

- Given nondeterministic finite automaton

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

we construct deterministic finite automaton

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

such that

$$L(M) = L(M')$$

- 'power set construction'

$$\begin{aligned} Z' &= 2^Z \\ z_0' &= \{z_0\} \\ \delta(Y,a) &= \bigcup_{z \in Y} \delta(z,a) \\ Z_A' &= \{Y \,:\, Y \cap Z_a \neq \emptyset\} \end{aligned}$$

- $n \to n+1$

$$(z_0', w) \vdash_{M'}^{n} (Y, w_{n+1} \ldots w_s) \vdash_{M'} (Y', w_{n+2} \ldots w_s)$$

$$IH: \quad Y = \{z \in Z \,:\, (z_0, w) \vdash_M^n (z, w_{n+1} \ldots w_s)$$

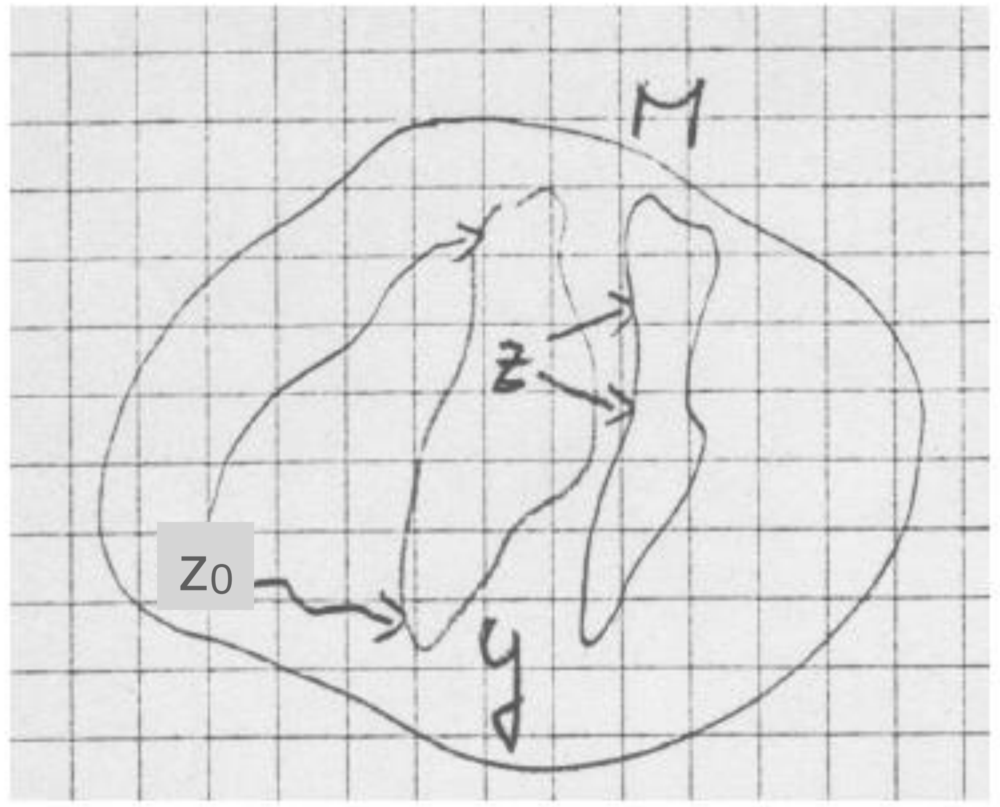


Figure 6: illustration of the power set construction

$$\begin{aligned} Y^* &= \bigcup_{z \in Y} \delta(z, w_{n+1}) \quad \text{(construction of } \delta'\text{)} \\ &= \{z \in Z \,:\, (z_0, w) \vdash_M^{n+1} (z, w_{n+2} \ldots w_s)\} \quad \text{(definition of } \vdash^{n+1}\text{)} \end{aligned}$$

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\} \ , \ \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\} \ , \ \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

  $$k = (z, w) \quad , \quad k' = (z', tail(w))$$

  then

  $$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

  $$k = (z, w) \quad , \quad k' = (z', w)$$

  then

  $$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, \varepsilon)$$

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \to 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\} \ , \ \delta : Z \times A_\varepsilon \to 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

  $$k = (z, w) \quad , \quad k' = (z', tail(w))$$

  then

  $$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

  $$k = (z, w) \quad , \quad k' = (z', w)$$

  then

  $$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, \varepsilon)$$

- simulation of nfa's by fa's becomes harder: in power set construction we had

  $$z_0' = \{z_0\} \quad , \quad \delta(Y, a) = \bigcup_{q \in Y} \delta(q, a)$$

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\}\ ,\ \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

$$k = (z, w) \quad , \quad k' = (z', tail(w))$$

  then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

$$k = (z, w) \quad , \quad k' = (z', w)$$

  then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, \varepsilon)$$

- simulation of nfa's by fa's becomes harder: in power set construction we had

$$z_0' = \{z_0\} \quad , \quad \delta(Y, a) = \bigcup_{q \in Y} \delta(q, a)$$

- must be augmented by states reachable by ε-moves. For $Q \subset Z$ define

$$E_0(Q) = Q \quad (1)$$

$$E_{i+1}(Q) = \bigcup_{q \in E_i(Q)} \delta(q, \varepsilon) \quad (2)$$

$$E(Q) = \bigcup_{i=0}^{|Z|-1} E_i(Q) \quad (3)$$

$$(4)$$

  new initial state and next state in deterministic simulation

$$z_0' = E(\{z_0\}) \quad , \quad \delta(Y, a) = E(\bigcup_{q \in Y} \delta(q, a))$$

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\} \;,\; \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z,\varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

$$k = (z,w) \quad , \quad k' = (z', tail(w))$$

then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

$$k = (z,w) \quad , \quad k' = (z',w)$$

then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z,\varepsilon)$$

- simulation of nfa's by fa's becomes harder: in power set construction we had

$$z_0' = \{z_0\} \quad , \quad \delta(Y,a) = \bigcup_{q \in Y} \delta(q,a)$$

- must be augmented by states reachable by ε-moves. For $Q \subset Z$ define

$$E_0(Q) = Q \quad (1)$$

$$E_{i+1}(Q) = \bigcup_{q \in E_i(Q)} \delta(q,\varepsilon) \quad (2)$$

$$E(Q) = \bigcup_{i=0}^{|Z|-1} E_i(Q) \quad (3)$$

(4)

why is finite union enough?

new initial state and next state in deterministic simulation

$$z_0' = E(\{z_0\}) \quad , \quad \delta(Y,a) = E(\bigcup_{q \in Y} \delta(q,a))$$

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\}\ ,\ \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

$$k = (z, w) \quad , \quad k' = (z', tail(w))$$

  then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

$$k = (z, w) \quad , \quad k' = (z', w)$$

  then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, \varepsilon)$$

- simulation of nfa's by fa's becomes harder: in power set construction we had

$$z_0' = \{z_0\} \quad , \quad \delta(Y, a) = \bigcup_{q \in Y} \delta(q, a)$$

- must be augmented by states reachable by ε-moves. For $Q \subset Z$ define

$$E_0(Q) = Q \tag{1}$$

$$E_{i+1}(Q) = \bigcup_{q \in E_i(Q)} \delta(q, \varepsilon) \tag{2}$$

$$E(Q) = \bigcup_{i=0}^{|Z|-1} E_i(Q) \tag{3}$$

(4)

  new initial state and next state in deterministic simulation

$$z_0' = E(\{z_0\}) \quad , \quad \delta(Y, a) = E(\bigcup_{q \in Y} \delta(q, a))$$

looks like great abstract nonsense...wait and see

5 allowing ε-moves of nfa's

so far: A finite input/tape alphabet, Z finite set of states, set valued transition fuction

$$\delta : Z \times A \rightarrow 2^Z$$

now:

$$A_\varepsilon = A \cup \{\varepsilon\} \;,\; \delta : Z \times A_\varepsilon \rightarrow 2^Z$$

ε-moves

$$z' \in \delta(z, \varepsilon)$$

z' can be reached from z by reading nothing, i.e. the head on the input tape is not advanced.

semantics:

- so far successor configurations consume an input symbol

$$k = (z, w) \quad , \quad k' = (z', tail(w))$$

then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, hd(w))$$

- now also:

$$k = (z, w) \quad , \quad k' = (z', w)$$

then

$$k \vdash k' \quad \leftrightarrow \quad z' \in \delta(z, \varepsilon)$$

- simulation of nfa's by fa's becomes harder: in power set construction we had

$$z_0' = \{z_0\} \quad , \quad \delta(Y, a) = \bigcup_{q \in Y} \delta(q, a)$$

- must be augmented by states reachable by ε-moves. For $Q \subset Z$ define

$$E_0(Q) = Q \tag{1}$$

$$E_{i+1}(Q) = \bigcup_{q \in E_i(Q)} \delta(q, \varepsilon) \tag{2}$$

$$E(Q) = \bigcup_{i=0}^{|Z|-1} E_i(Q) \tag{3}$$

$$\tag{4}$$

new initial state and next state in deterministic simulation

$$z_0' = E(\{z_0\}) \quad , \quad \delta(Y, a) = E(\bigcup_{q \in Y} \delta(q, a))$$

$\boxed{\textbf{looks like great abstract nonsense...wait and see}}$

OR: have you seen it already?

deterministic and nondeterministic models of computation

always

- K: set of configurations
- E: set of inputs

determinism:

- $k \in K$ and $e \in E$ uniquely determine next configuration k'
- model by transition function

$$\delta : K \times E \to K$$

- examples: hardware, MIPS, C0

graphical representation

$$k \to^{e} k' \leftrightarrow k' = \delta(k, e).$$

k' is **the** next configuration after k with input e

nondeterminism:

$$k \to^{e} k'$$

k' is **a possible** next configuration after k with input e

example: ISA + disk

- processor step: $d \to^{eev} \delta(d, eev)$
- disk step: $d \to \eta(d)$

consumes no external interrupt signal

ϵ-move!

6 Regular expressions and nfa's

Lemma 4. *For every regular language $L \in R(A)$ there is an nfa M which accepts L:*

$$L(M) = L$$

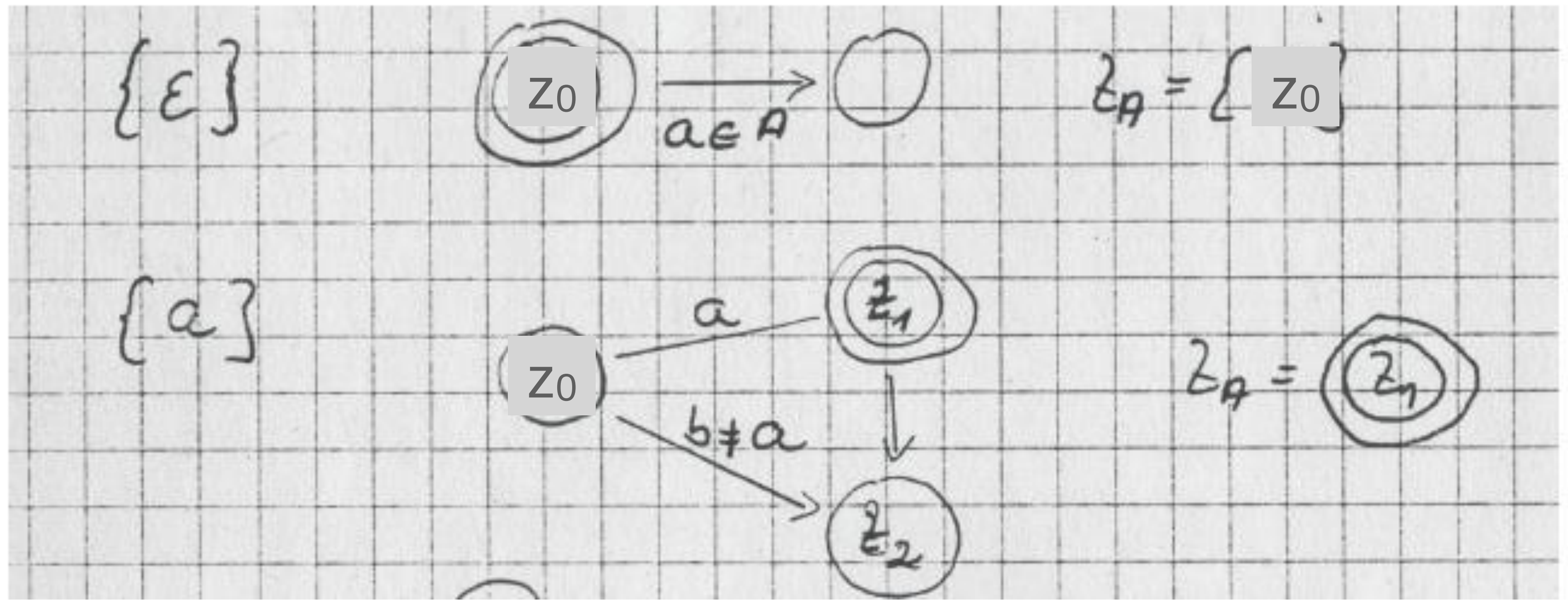
5 Regular expressions and nfa's

Lemma 4. *For every regular language $L \in R(A)$ there is an nfa M which accepts L:*

$$L(M) = L$$

Proof by induction over the structure of regular expressions:

- base case: $L = \{\varepsilon\}$ or $L = \{a\}$



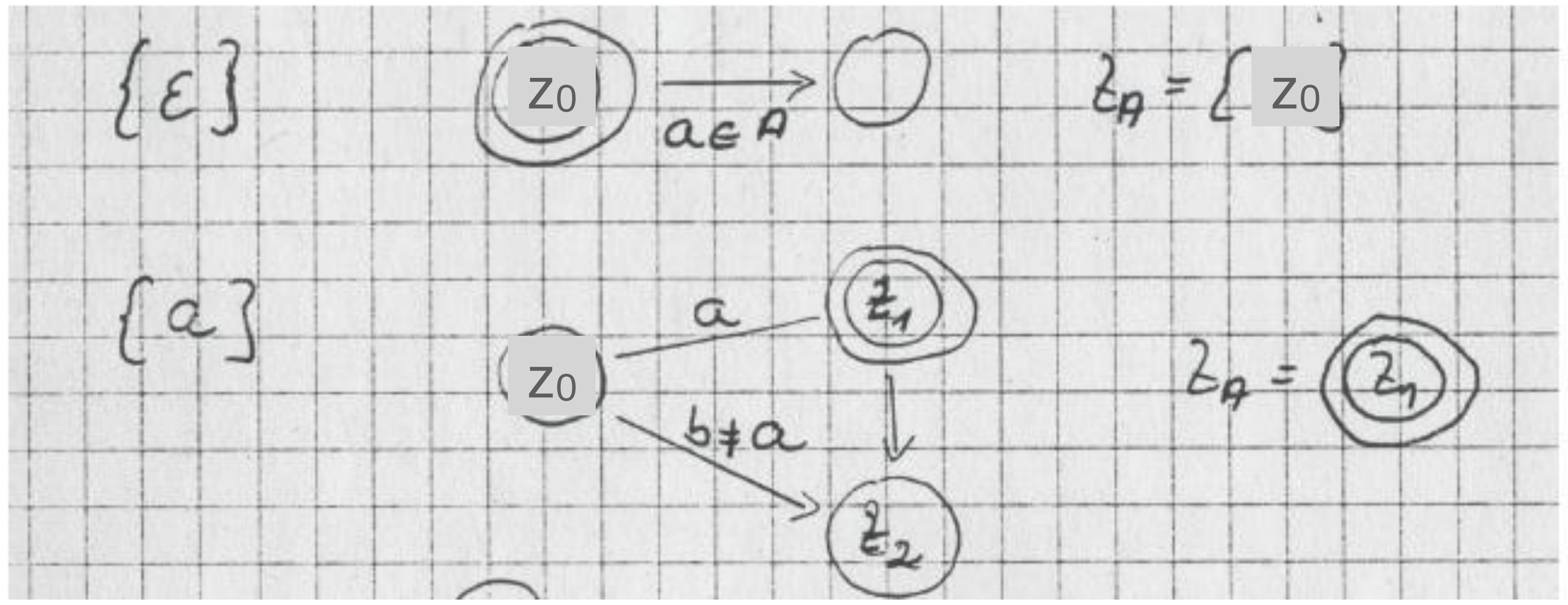
5 Regular expressions and nfa's

Lemma 4. *For every regular language expression $L \in R(A)$ there is an nfa M which accepts L:*

$$L(M) = L$$

Proof by induction over the structure of regular expressions:

- base case: $L = \{\varepsilon\}$ or $L = \{a\}$



- $L(M'') = L \circ L'$:

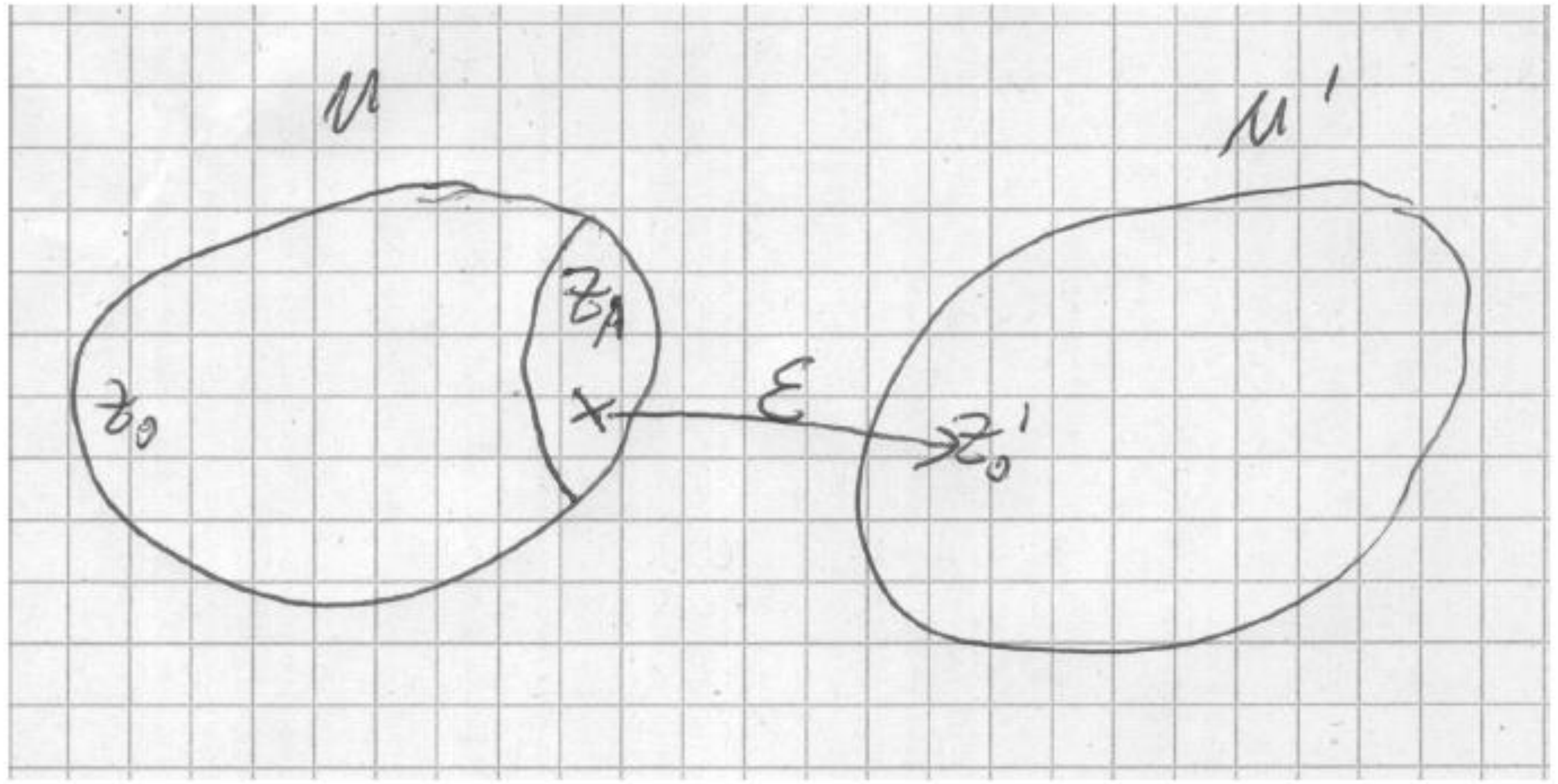


Figure 8: Automata for regular languages: $L \circ L'$

$$
\begin{aligned}
Z'' &= Z \cup Z' \\
z_0'' &= z_0 \\
Z_A'' &= Z_A' \\
\delta''(x,a) &= \begin{cases} \delta(x,a) & x \in Z \setminus Z_A \\ \delta(x,a) \cup \{z_0'\} & x \in Z_A \text{ , } a = \varepsilon \\ \delta'(x,a) & \text{otherwise} \end{cases}
\end{aligned}
$$

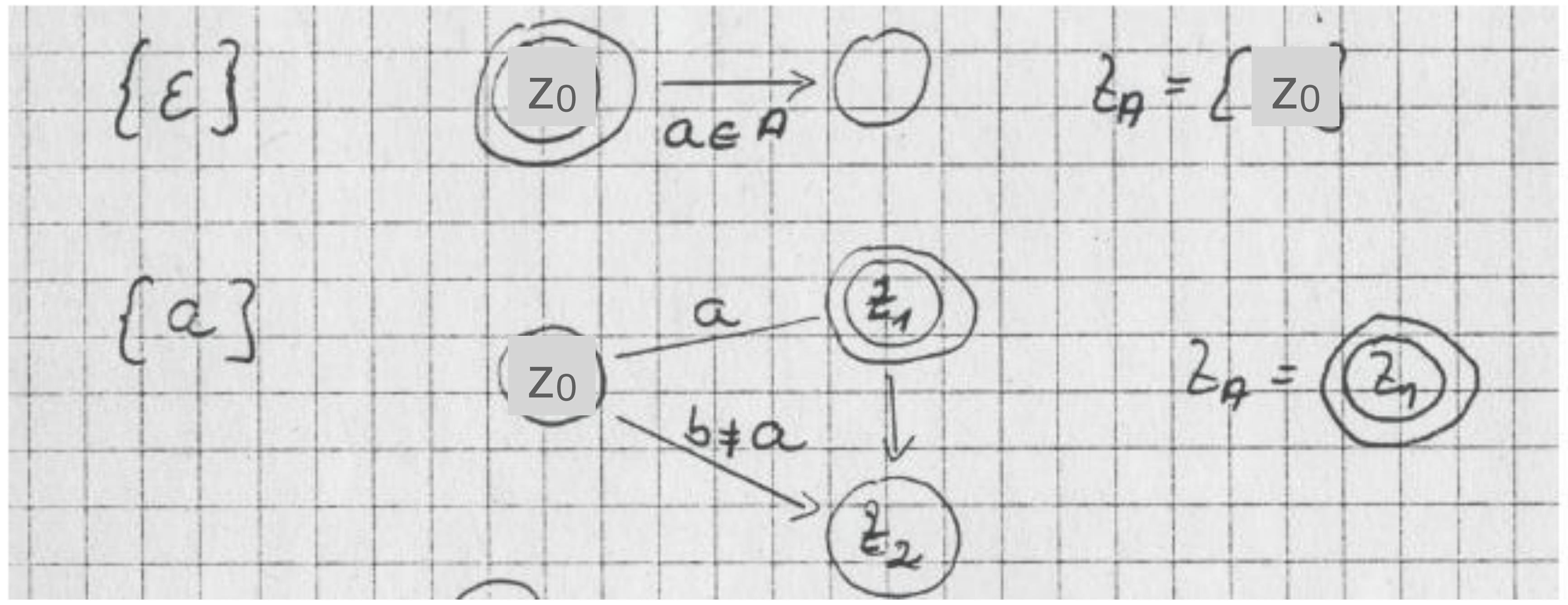
5 Regular expressions and nfa's

Lemma 4. *For every regular language expression $L \in R(A)$ there is an nfa M which accepts L:*

$$L(M) = L$$

Proof by induction over the structure of regular expressions:

- base case: $L = \{\varepsilon\}$ or $L = \{a\}$



- $L(M'') = L \cup L'$

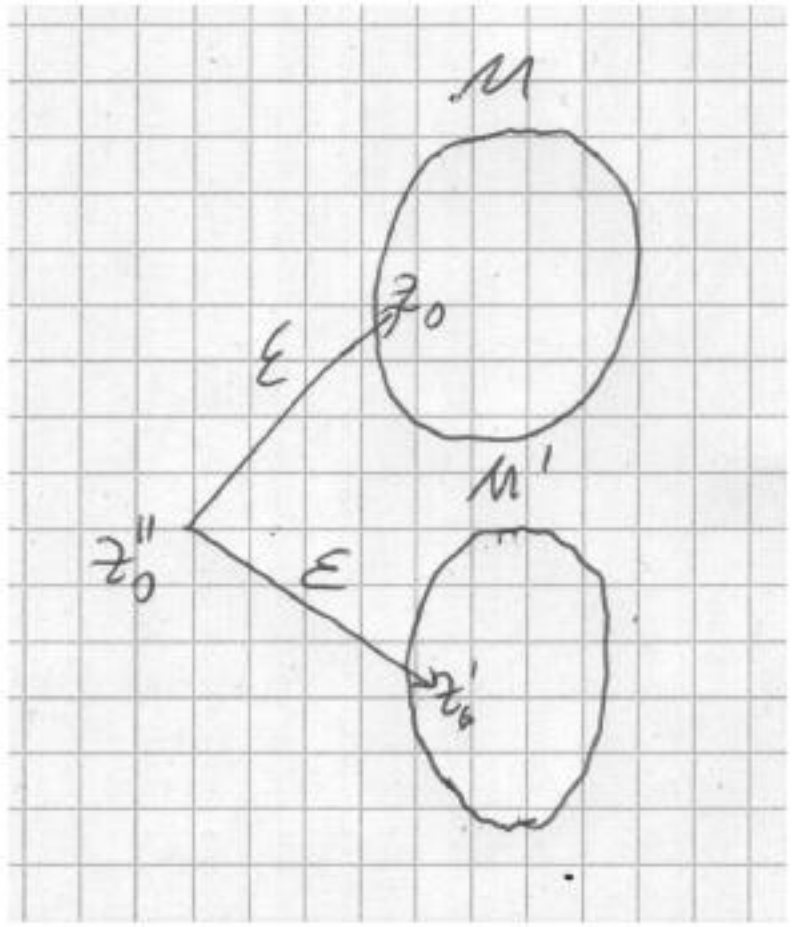


Figure 9: Automata for regular languages: $L \cup L'$

$$Z = Z \cup Z' \cup \{z_0''\} \quad \text{(new start state)}$$

$$\delta(x,a) = \begin{cases} \delta(x,a) & x \in Z \\ \delta'(x,a) & x \in Z' \\ \{z_0, z_0'\} & x = z_0'' \text{ , } a = \varepsilon \end{cases}$$

$$Z_a'' = Z_A \cup Z_A'$$

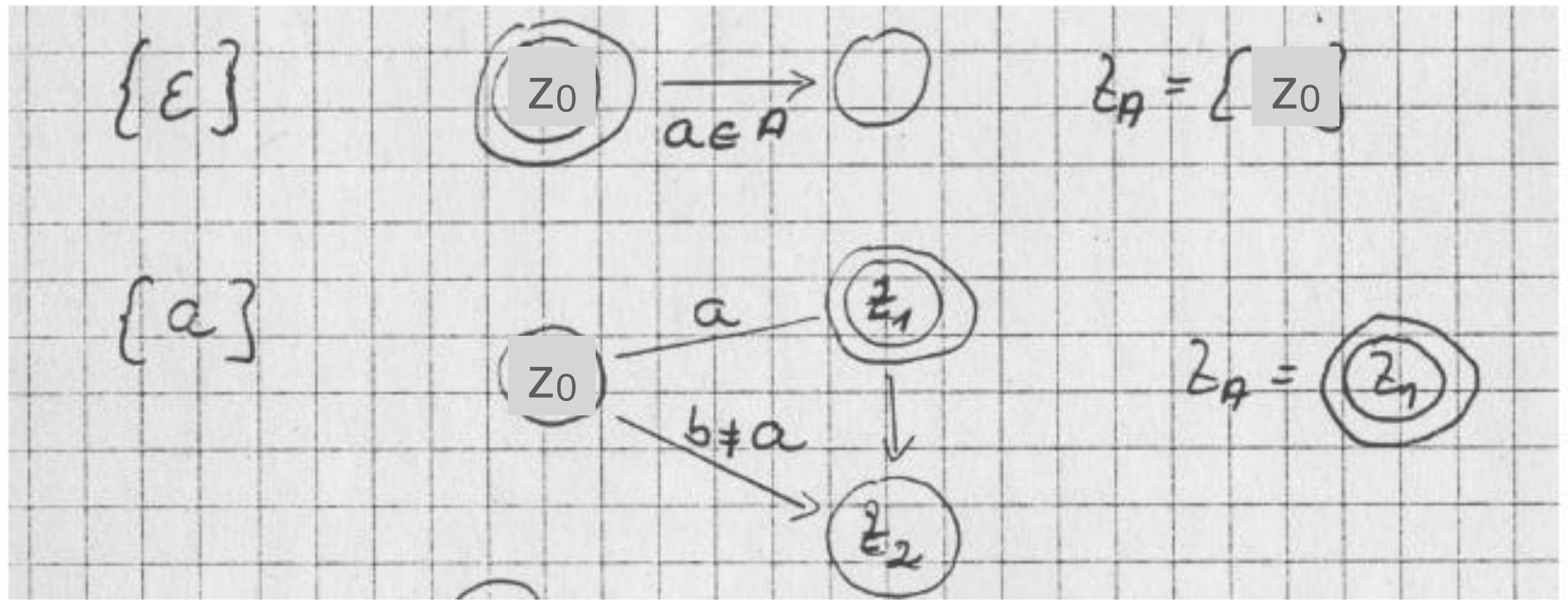
5 Regular expressions and nfa's

Lemma 4. *For every regular language expression $L \in R(A)$ there is an nfa M which accepts L:*

$$L(M) = L$$

Proof by induction over the structure of regular expressions:

- base case: $L = \{\varepsilon\}$ or $L = \{a\}$



- $L(M'') = L \cup L'$

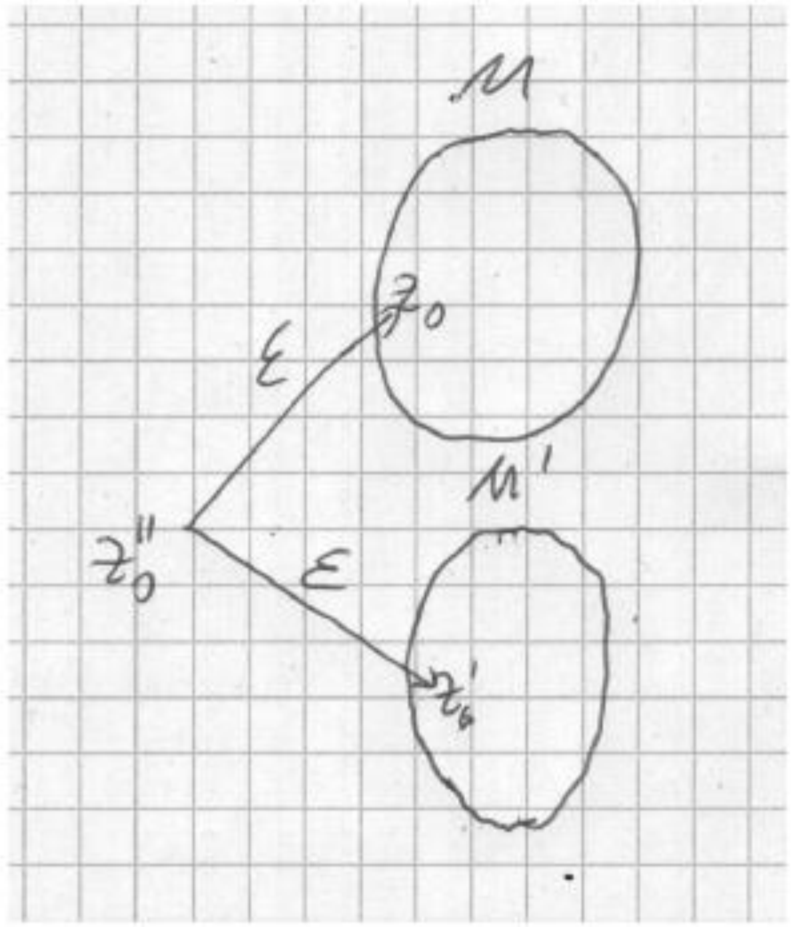


Figure 9: Automata for regular languages: $L \cup L'$

$$
\begin{aligned}
Z &= Z \cup Z' \cup \{z_0''\} \quad \text{(new start state)} \\
\delta(x,a) &= \begin{cases} \delta(x,a) & x \in Z \\ \delta'(x,a) & x \in Z' \\ \{z_0, z_0'\} & x = z_0'',\ a = \varepsilon \end{cases} \\
Z_a'' &= Z_A \cup Z_A'
\end{aligned}
$$

- $L(M'') = L^*$: exercise

7 Pumping lemma

Lemma 5. *Let $L \in R(A)$ be a regular language. Then there is a constant N such that all words $w \in L$ which are longer than N, i.e.*

$$|w| = n > N$$

can be decomposed as

$$w = uvx$$

such that

$$\begin{array}{rrcl} & v & \geq & 1 \\ & |uv| & \leq & N \\ \text{you can pump on v:} \quad \forall i \in \mathbb{N}_0. & uv^i w & \in & L \end{array}$$

7 Pumping lemma

Lemma 5. *Let $L \in R(A)$ be a regular language. Then there is a constant N such that all words $w \in L$ which are longer than N, i.e.*

$$|w| = n > N$$

can be decomposed as

$$w = uvx$$

such that

$$\begin{array}{rrcl} & v & \geq & 1 \\ & |uv| & \leq & N \\ \text{you can pump on v:} \quad \forall i \in \mathbb{N}_0. & uv^i w & \in & L \end{array}$$

Hence

Lemma 6.

$$\{a^n b^n \ : \ n \in \mathbb{N}\} \quad \textit{is not regular}$$

why?

7 Pumping lemma

Lemma 5. *Let $L \in R(A)$ be a regular language. Then there is a constant N such that all words $w \in L$ which are longer than N, i.e.*

$$|w| = n > N$$

can be decomposed as

$$w = uvx$$

such that

$$\begin{aligned} v &\geq 1 \\ |uv| &\leq N \end{aligned}$$

you can pump on v: $\forall i \in \mathbb{N}_0.\quad uv^i w \in L$

Hence

Lemma 6.

$$\{a^n b^n \;:\; n \in \mathbb{N}\} \quad \textit{is not regular}$$

proof of lemma 5:

Let $N =$ number of states of dfa M accepting L and let $n = |w| > N$. Consider computation of M started with w

$$(k_0, k_1, \ldots, k_n)$$

$$k_i = (z_i, w_{i+1} \ldots w_n)$$

ping lemma

Lemma 5. *Let $L \in R(A)$ be a regular language. Then there is a constant N such that all words $w \in L$ which are longer than N, i.e.*

$$|w| = n > N$$

can be decomposed as

$$w = uvx$$

such that

$$\begin{aligned} v &\geq 1 \\ |uv| &\leq N \\ \forall i \in \mathbb{N}_0. \quad uv^i w &\in L \end{aligned}$$

you can pump on v:

Hence

Lemma 6.

$$\{a^n b^n \ : \ n \in \mathbb{N}\} \quad \textit{is not regular}$$

proof of lemma 5:

Let $N =$ number of states of dfa M accepting L and let $n = |w| > N$. Consider computation of M started with w

$$(k_0, k_1, \ldots, k_n)$$

$$k_i = (z_i, w_{i+1} \ldots w_n)$$

Pidgeon hole argument:

$$n > N \ \rightarrow \exists i, j > i. \ z_i = z_j$$

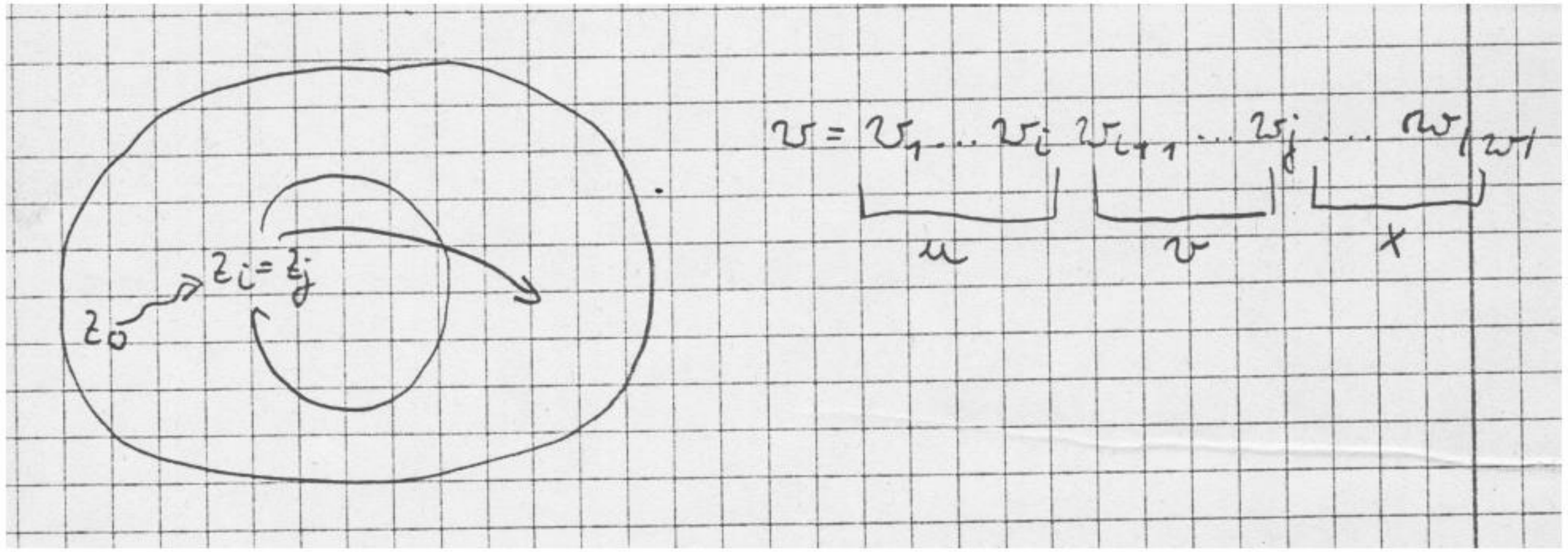


Figure 10: proof of puming lemma for regular languages