Minimal automaton

EQUIVALENCE OF WORDS MODULO A LANGUAGE. Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ. Let $u, v \in \Sigma^*$ be two words (that may belong or not to the language L). We say that the language L SEPARATES u et v if

$$\begin{cases} u \in L \\ v \notin L \end{cases} \text{ ou } \begin{cases} u \notin L \\ v \in L \end{cases} \qquad (14)$$

We say that u et v are equivalent modulo L and we write $u \equiv v \bmod L$ if for every $w \in \Sigma^*$ the language L does **not separate** the words $u\,w$ and $v\,w$. In other words we have

$$u \equiv v \bmod L \iff (\forall w \in \Sigma^*)\ \left(u\,w \in L \iff v\,w \in L \right) \qquad (15)$$

The relation $u \equiv v \bmod L$ is an equivalence relation (since it is reflexive, symmetric and transitive) and thus defines equivalence classes called the *equivalence classes* (or *residue classes*) *associated with L*.

EQUIVALENCE OF WORDS MODULO AN AUTOMATON. Let $\mathcal{A} = (\Sigma, S, i, F, \delta)$ be a DFA. Let $u, v \in \Sigma^*$ be two words We say that u et v are equivalent modulo $\mathcal{A}$ and we write $u \equiv v \bmod \delta$ if after reading u the DFA is in the same state than after reading v. Let us denote by $\delta(i, u)$ and $\delta(i, v)$ the state of the automaton $\mathcal{A}$ after reading u and v respectively. Then we have

$$u \equiv v \bmod \delta \iff \delta(i, u) = \delta(i, v). \qquad (16)$$

Again the relation $u \equiv v \bmod \delta$ is an equivalence relation and thus defines residue classes associated with the automaton $\mathcal{A}$.

Example 6 The DFA over $\Sigma = \{a, b\}$ on Figure 20 has three residue classes:

- that of the words with no a,
- that of the words with no b after an a,
- that of the words with a factor ab.

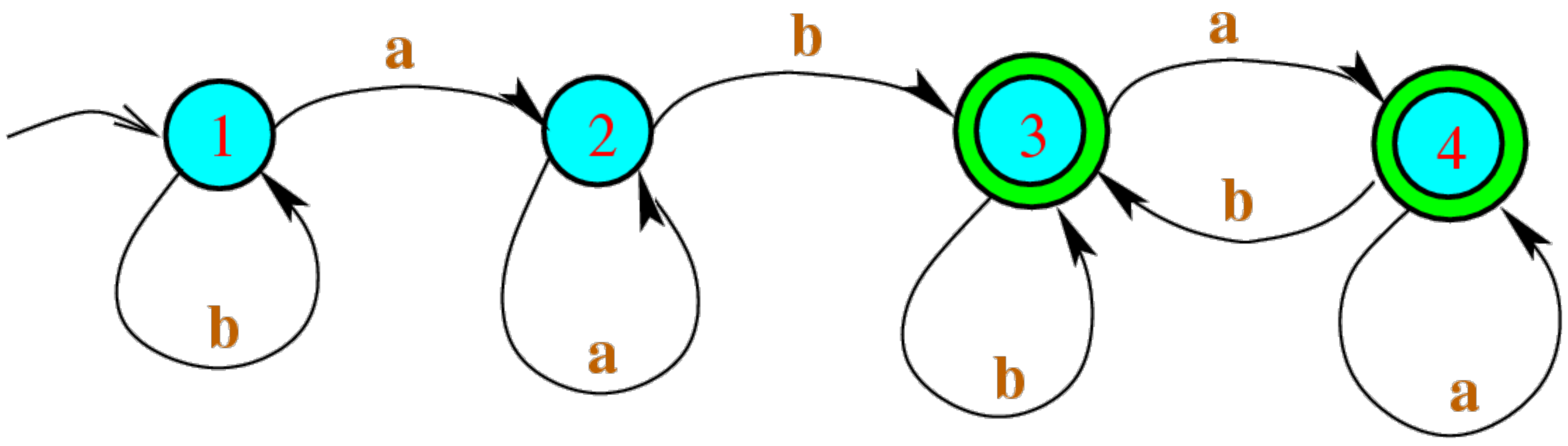


Figure 20: A DFA whith three classes of equivalence.

Proposition 8 Let $L \subseteq \Sigma^*$ be a language over Σ recognized by the DFA $\mathcal{A} = (\Sigma, S, i, F, \delta)$. For every words u and v we have

$$u \equiv v \bmod \delta \Longrightarrow u \equiv v \bmod L \qquad (17)$$

Remark 4 *Proposition 8 expresses the fact that every residue class modulo the automaton* $\mathcal{A}$ *is entirely contained in a residue class modulo its associated language L. Therefore the number of classes modulo L is upper bounded by the number of classes modulo* $\mathcal{A}$ *and thus by the number of states of* $\mathcal{A}$*. It follows from the following theorem that having a finite number of residue classes is a necessary and sufficient condition for a language to be recognized by FA.*

Theorem 4 Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ. Let n be a positive integer. If there exist exactly n residue classes modulo L then there exists a DFA with n states that recognizes L.

MINIMAL AUTOMATON. Let $L \subseteq \Sigma^*$ be a language over the alphabet Σ

- recognized by FA and
- with n residue classes.

It follows from Proposition 8 and Theorem 4 that

- every DFA recognizing L has at least n states,
- there exists a DFA with n states recognizing L.

Therefore we can call *minimal automaton* every DFA with n states and recognizing L.

Remark 5 *The number of residue classes of a language is rarely known (until one gets a minimal DFA accepting this language). So one needs a way to characterize minimal automata.*

Then one needs an algorithm to compute from a given DFA (minimal or not) recognizing L a minimal automaton recognizing L too. Such an algorithm will group states that do essentially the same job. This leads to the following notions.

EQUIVALENCE OF STATES IN A DFA. Let $\mathcal{A} = (\Sigma, S, i, F, \delta)$ be a DFA. We say that two states $p, q \in S$ are *equivalent* and we write $p \equiv q$ if for every word $w \in \Sigma^*$ we have

$$(\delta(p, w) \in F) \Longleftrightarrow (\delta(q, w) \in F). \tag{18}$$

Let k be a non-negative integer. We say that two states $p, q \in S$ are *equivalent at order k* and we write $p \equiv_k q$ if for every word $w \in \Sigma^*$ with length $|w|$

$$\left(|w| \leq k \right) \Longrightarrow \left((\delta(p, w) \in F) \Longleftrightarrow (\delta(q, w) \in F) \right) \tag{19}$$

CHARACTERIZATION OF A MINIMAL AUTOMATON. Among other properties, Theorem 5 states that for a minimal automaton, each residue class of states contains only one state.

Theorem 5 Let $L \subseteq \Sigma^*$ be a language recognized by FA and let n be its number of residue classes. Let $\mathcal{A} = (\Sigma, S, i, F, \delta)$ be a minimal DFA recognizing L. Then we have the following properties

- for every words $u, v \in \Sigma^*$ we have $u \equiv v \bmod \delta \Longleftrightarrow u \equiv v \bmod L$.
- for every $q \in F$ there exists a word $u \in \Sigma^*$ such that $\delta(i, u) = q$.
- for every states $p, q \in S$ we have $p \equiv q \Longleftrightarrow p = q$.

CONSTRUCTION OF THE EQUIVALENCE CLASSES FOR STATES. Proposition 9 suggests an algorithm for constructing the equivalence classes of states.

Proposition 9 Let $\mathcal{A} = (\Sigma, S, i, F, \delta)$ be a DFA and k be a non-negative integer. If the relations $\equiv_k$ and $\equiv_{k+1}$ have the same residue classes then $\equiv_k$ and $\equiv$ have the same residue classes too.

The method is follows.

- Compute $\equiv_0$ which consists of two classes:
  - the set of the final states,
  - the set of the non final states.

- Compute $\equiv_1$ and set k to 0
- Whhile $\equiv_k$ and $\equiv_{k+1}$ are different $\equiv_{k+2}$ and set k to $k + 1$.

In practice one construct a table that gives the transition from each state after reading any word of length 0, any word of length 1, any word of length 2, etc... In these tables we underline final states in order to detect classes more easily. Quite often computations can be simpler. For instance, in Example 7 computations can be stopped after computing the third column. Explain why!

THE CONSTRUCTION OF A MINIMAL AUTOMATON $\overline{\mathcal{A}} = (\Sigma, S', i', F', \delta')$ from a given DFA $\mathcal{A} = (\Sigma, S, i, F, \delta)$ can be down as follows (where the class of a state $s \in S$ is denoted by $\overline{s}$).

- Compute the equivalence classes of states. From Theorem 5 they provide S', also denoted by $\overline{S}$.
- Let $s, s' \in S$ such that $\overline{s} = \overline{s'}$. Let $x \in \Sigma$. We have $\overline{\delta(s,x)} = \overline{\delta(s',x)}$. Hence we can define $\delta'(\overline{s}, x)$ as

$$\delta'(\overline{s}, x) = \overline{\delta(s', x)} \tag{20}$$

- We choose $i' = \overline{i}$.
- We define $F' = \{\overline{s} \mid s \in F\}$.

Example 7

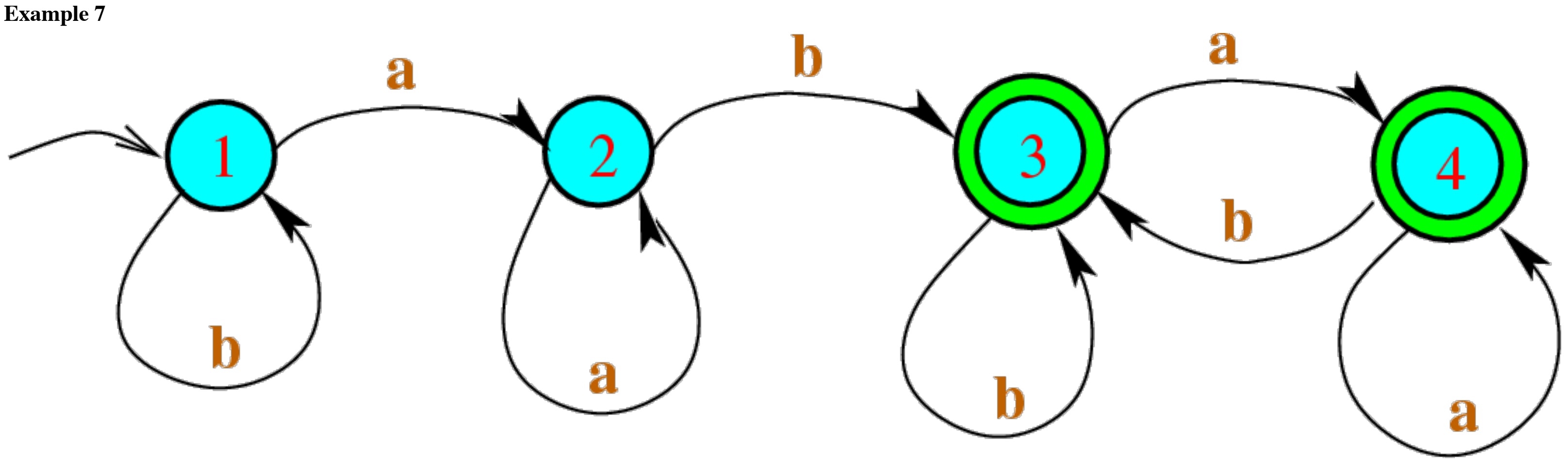


Figure 21: A DFA whith three classes of equivalence.

ε	a	b	aa	ab	ba	bb
1	2	1	2	1	2	1
2	2	$\underline{3}$	2	$\underline{3}$	$\underline{4}$	$\underline{3}$
$\underline{3}$	$\underline{4}$	$\underline{3}$	$\underline{4}$	$\underline{4}$	$\underline{4}$	$\underline{3}$
$\underline{4}$	$\underline{4}$	$\underline{3}$	$\underline{4}$	$\underline{4}$	$\underline{4}$	$\underline{3}$

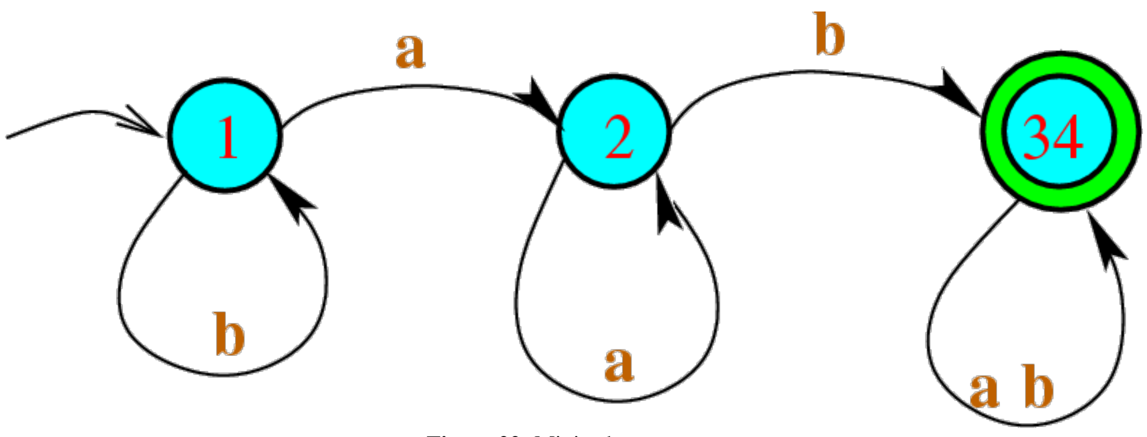


Figure 22: Minized automaton.

Example 8

ε	*a*	*b*	*aa*	*ab*	*ba*	*bb*
1	2	6	7	$\underline{3}$	$\underline{3}$	7
2	7	$\underline{3}$				
$\underline{3}$	1	$\underline{3}$	2	6	1	$\underline{3}$
5	8	6	7	$\underline{3}$	$\underline{3}$	7
6	$\underline{3}$	7	1	$\underline{3}$	7	$\underline{3}$
7	7	5	7	5	8	6
8	7	$\underline{3}$				

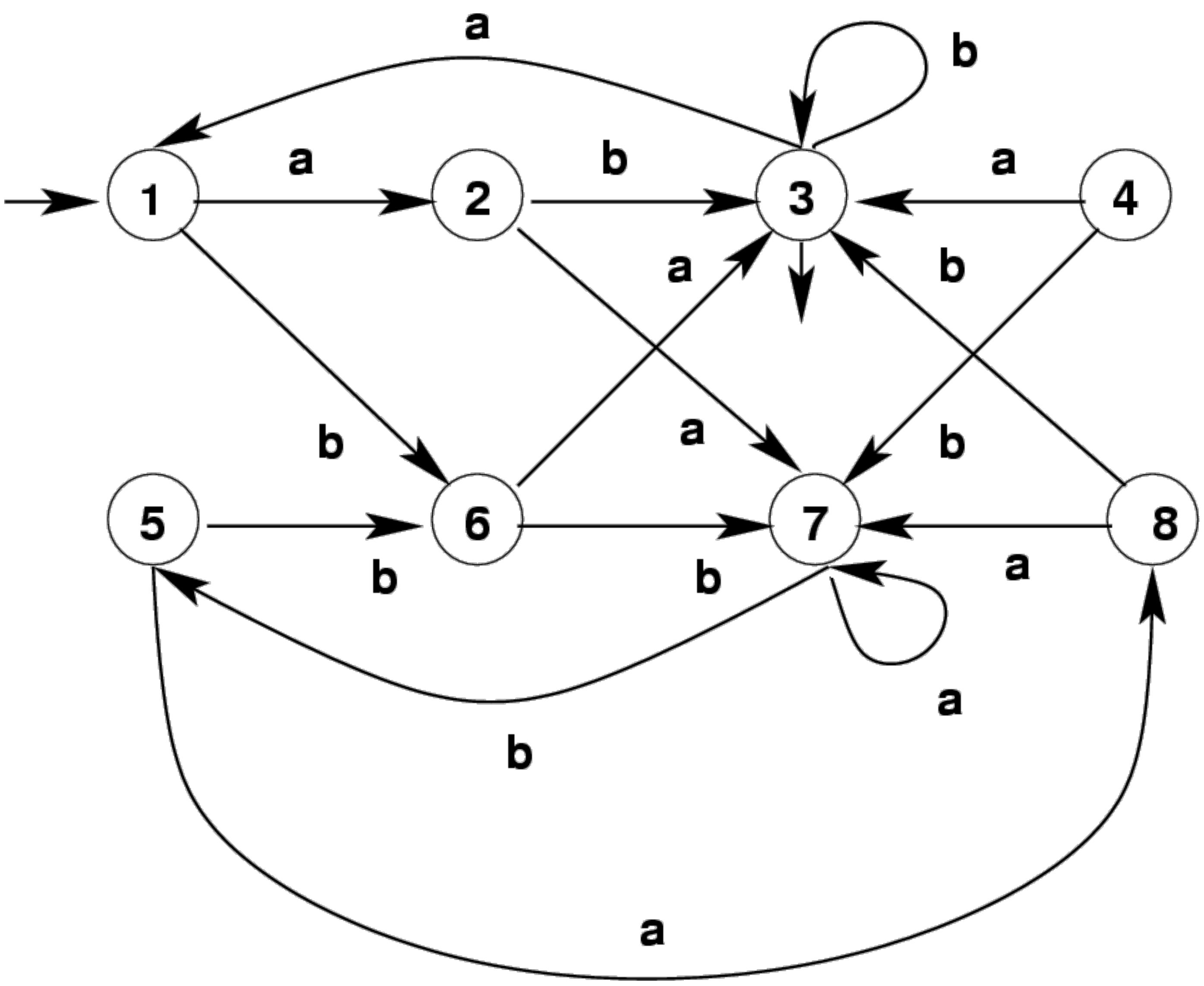
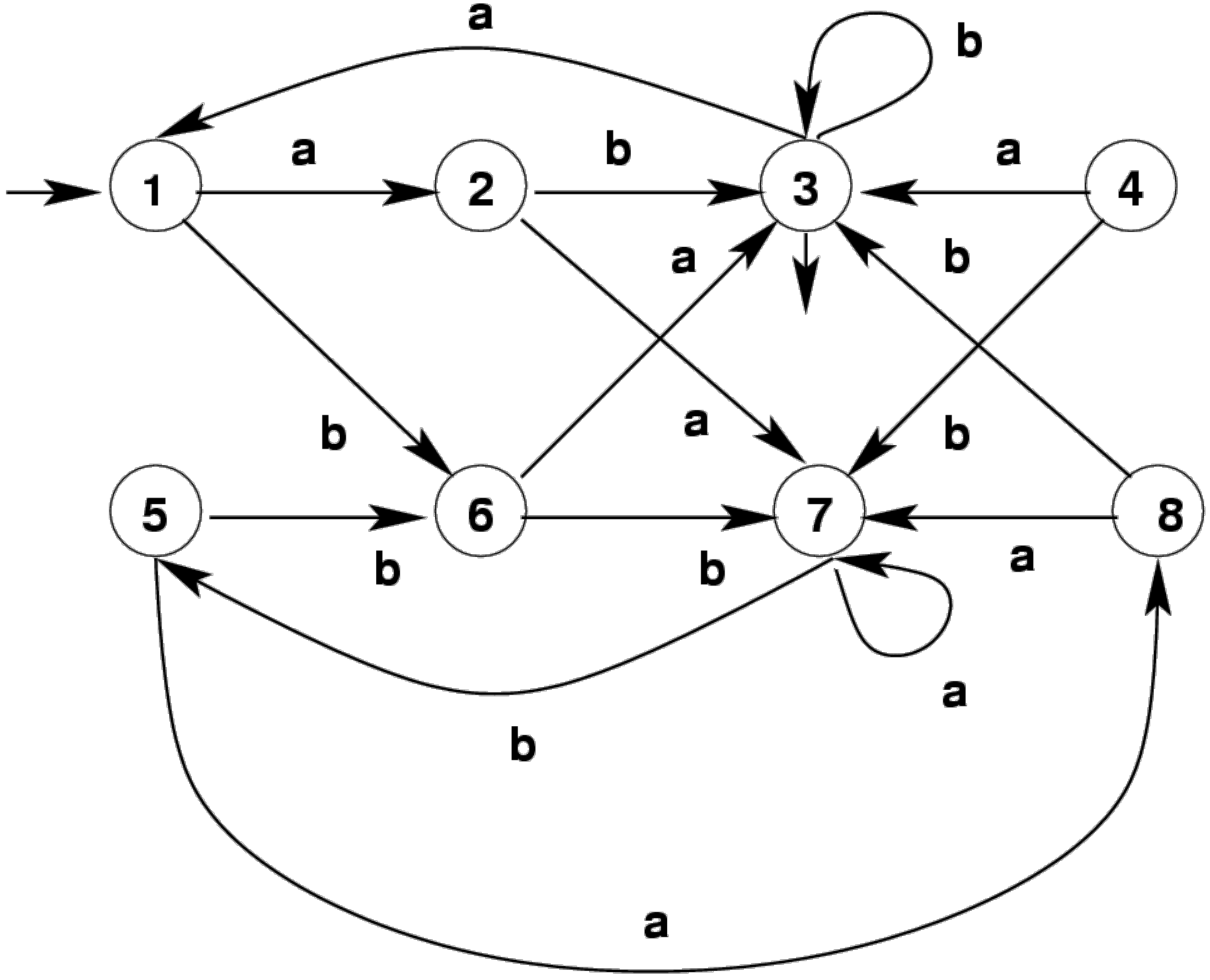


Figure 23: Another DFA to minize.



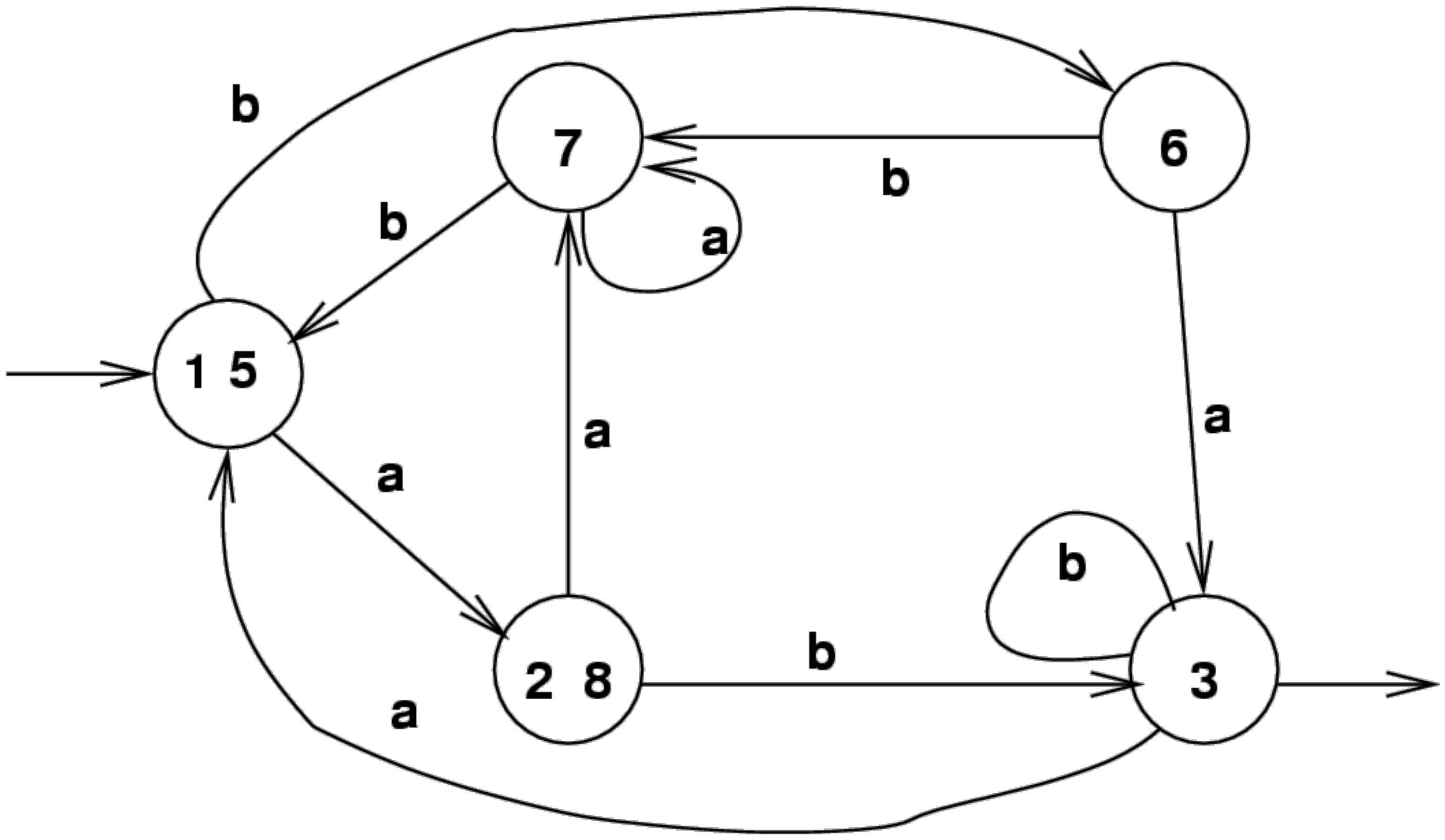


Figure 24: Minized Automaton.

Next Up Previous
Next: About this document ... **Up:** The design of an efficient **Previous:** The design of an efficient

Marc Moreno Maza
2004-12-02