# Equivalences of Finite Automata

*Antonina Nepeivoda*
`a_nevod@mail.ru`

# Machines Comparison
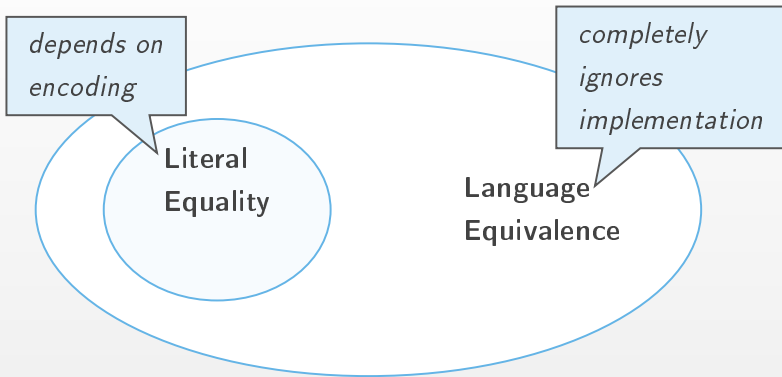


**Problem:** *how to find an equivalence that is sustainable to irrelevant implementation details (such as node naming) but tracks parsing-relevant properties?*

# Equality of DFA and NFA

Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.
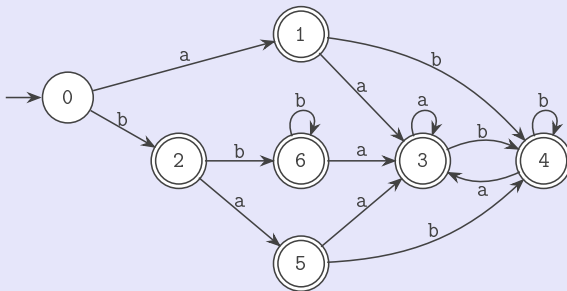


*Above is the state numeration with respect to the military (length- lexicographic) order, given* $a \prec b$:
$\varepsilon \prec a \prec b \prec aa \prec ab \prec ba \prec bb$.

## Equality of DFA and NFA

Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.
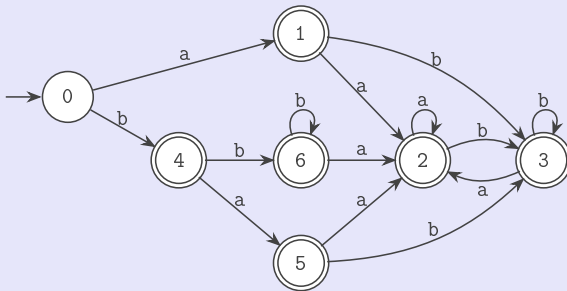


*Another numeration is induced by the "vanilla" lexicographic order, given $a \prec b$: now $\varepsilon \prec a \prec aa \prec ab \prec b \prec ba \prec bb$.*

# Equality of DFA and NFA

Given a DFA, its states can be *canonically named*: i.e. state number is determined by the string marking the shortest path from the starting state to the state considered. Then the numeration depends only on the chosen linear order on strings.

However, the canonical numeration does not work for NFA, provided that the string sets marking paths to the states can coincide.

Hence, DFA equality up to the state renaming can be thought as the literal coincidence of the canonically ordered DFA; but the canonical order does not work for recognising NFA equality.

# Behavioral Equivalence and Language Equivalence

Language equivalence tracks only admissible actions, but not a way the actions are performed. The following example is well-known.

- N is «put a note into the machine»;
- C is «order a coffee»;
- T is «order a tea».
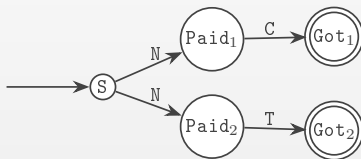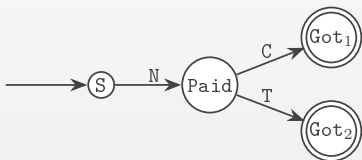


The given two machines have the same language, but are distinct from the point of view of a user:

- the first requires a note, and then asks what drink is ordered;
- the second asks for a choice when taking money, then requires to press the button that prepares it.

# Bisimulation of Labelled Transition Systems

*Bisimulation is a relation $\sim$ between states of the systems $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfying the following property:*

- *If $q_1 \sim q_2$ ($q_1 \in \mathcal{T}_1$, $q_2 \in \mathcal{T}_2$), then for every transition $q_1 \xrightarrow{\gamma} q_1'$ in $\mathcal{T}_1$ there is a transition $q_2 \xrightarrow{\gamma} q_2'$ in $\mathcal{T}_2$ such that $q_1' \sim q_2'$, and vice versa.*

*Starting and final (if any) states must be bisimilar.*

Every state machine $\mathscr{A}$ can be represented as a labelled transition system.

- $\mathscr{A}_1$ and $\mathscr{A}_2$ are bisimilar $\Leftrightarrow$ their LTS $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar.
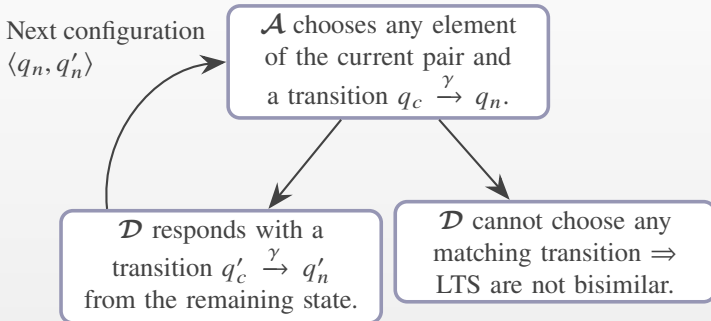
Labelled transition systems are not necessarily finite; moreover, LTS contain no final states.

Existence of final states can be modelled via introducing *endmarkers* (usually denoted $\$$). Then every final state of an NFA has a transition by the endmarker to the unique «bottom» state.

# Bisimulation Game

$\mathcal{T}_1$ and $\mathcal{T}_2$ bisimilarity checking technique can be formulated as a two-player game with an initial configuration $\langle q_S, q'_S \rangle$:

Next configuration $\langle q_n, q'_n \rangle$

$\mathcal{A}$ chooses any element of the current pair and a transition $q_c \xrightarrow{\gamma} q_n$.

$\mathcal{D}$ responds with a transition $q'_c \xrightarrow{\gamma} q'_n$ from the remaining state.

$\mathcal{D}$ cannot choose any matching transition $\Rightarrow$ LTS are not bisimilar.

- Attacker's winning strategy always leads to the fact that any possible play is finite.

- In presence of final states, $\mathcal{A}$ can additionally declare «game-over» in any final state (and $\mathcal{D}$ must respond with the «game-over» as well).

## Equivalent Trim DFA are Bisimilar

Given two non-bisimilar trim DFA $\mathscr{A}_1$ and $\mathscr{A}_2$, we assume by the contradiction that the player $\mathcal{A}$ has a winning strategy. The strategy is completely determined by a finite input string $\omega$.

- If after reading the string $\omega$ one DFA ends up in a final state, while the other ends up in a non-final state, then $\omega$ witnesses that their languages do not coincide.
- If after reading the string $\omega$ one DFA (say $\mathscr{A}_1$) ends up in a state with an outgoing transition by some $\gamma \in \Sigma$, while the other does not has such a transition, then no string in $\mathscr{L}(\mathscr{A}_1)$ with the prefix $\omega\gamma$ can belong to $\mathscr{L}(\mathscr{A}_2)$. The DFA are trim $\Rightarrow$ at least one string prefixed by $\omega\gamma$ is in $\mathscr{L}(\mathscr{A}_1)$.

## Bisimulation and Equality

Bisimilar DFA are not necessarily equal, even if cardinalities of their state sets are also equal.



In this example, the distinction between the states $c_1$ and $c_3$ is redundant: they are indistinguishable with respect to the languages that can be recognised from them, namely, $\mathscr{L}(c_1) = \mathscr{L}(c_3) = \{\varepsilon\}$. Hence, $c_1 \sim c_3$.

We could *merge* the bisimilar states with no impact to the recognised DFA language; conversely, if we know that the DFA states coincide wrt their languages, then we know they are behaviorally equivalent.

## *k*-Bisimulation: Playing Backwards

> Given an NFA $\mathscr{A}$, how do we know that its states $q_i$, $q_j$ are bisimilar?

- if $q_i$ is final, while $q_j$ is non-final, then $\mathcal{A}$ can choose $q_i$ and declare game-over, winning the game. Hence, $\mathcal{A}$ can win doing no move at all. If $q_i$, $q_j$ are both final or both non-final, at least one move is required for $\mathcal{A}$ to win, and we say that $q_i \sim_0 q_j$.

- if exists $\gamma$ and $q_i'$ s.t. $q_i \xrightarrow{\gamma} q_i'$, and for all $q_j'$ s.t. $q_j \xrightarrow{\gamma} q_j'$ $q_i' \nsim_k q_j'$, then $\mathcal{A}$ wins in $k + 1$ moves starting from the position $\langle q_i, q_j \rangle$. Otherwise, we say that $q_i \sim_{k+1} q_j$.

A closer look on the $\sim_{k+1}$-condition:

$$q_i \sim_{k+1} q_j \Leftrightarrow \forall q_i', \gamma(q_i \xrightarrow{\gamma} q_i' \Rightarrow \exists q_j'(q_j \xrightarrow{\gamma} q_j'))$$
$$\& \ \forall q_j', \gamma(q_j \xrightarrow{\gamma} q_j' \Rightarrow \exists q_i'(q_i \xrightarrow{\gamma} q_i'))$$

When we reach the fixpoint of $\sim_k$ (i.e. $\sim_k = \sim_{k+1}$), then we know that $\mathcal{A}$ never can win in position $\langle q_i, q_j \rangle$ given $q_i \sim_k q_j$, hence $q_i \sim q_j$.

### ◔◔ **Definition**

*Given a function $f : \mathcal{D} \to \mathcal{D}$, its fixpoint is a value $\tau \in \mathcal{D}$ such that $f(\tau) = \tau$.*

If $f : \mathbb{R} \to \mathbb{R}$, then its fixpoints are located at the intersection of $f$-graph with the diagonal.

We have met the fixpoint construction before in the Arden lemma. Namely, the expression $\Phi^* \Psi$ is the fixpoint of the function $f(X) = \Phi X \mid \Psi$.

Moreover, the «Kleene star» construction is a fixpoint itself: $\Phi^*$ is the fixpoint language for the function $s(X) = \Phi X \mid \varepsilon$.

The fixpoint construction is particularly useful in computer science when some set is saturated wrt a well-founded relation.

# Bisimilarity in DFA and Minimization

Given a DFA with state set $Q$, and set of final states $F$, we know that $q_i \sim q_j$ if and only if the sets of words recognised starting from $q_i$ and starting from $q_j$ are equal.
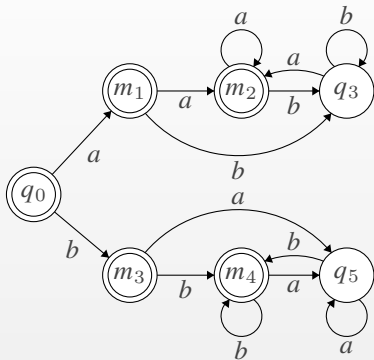Hence, we can minimize the DFA by merging bisimilar states.

---

### *DFA Minimization Algorithm*

---

1. Given $q_i, q_j \in Q$, mark the pairs $\{q_i, q_j\}$ s.t. $q_i \not\sim_0 q_j$ (i.e. $q_i \in F$, while $q_j \notin F$, or vice versa).
2. Mark all the pairs $\{q_i, q_j\}$ s.t. $\exists \gamma (q_i \xrightarrow{\gamma} q_i' \,\&\, q_j \xrightarrow{\gamma} q_j' \,\&\, \{q_i', q_j'\}$ is marked).
3. Repeat Step 2 until no new marked pair appear (i.e. fixpoint of $\sim_k$ is reached).
4. Merge the states in the all unmarked pairs: they are bisimilar.
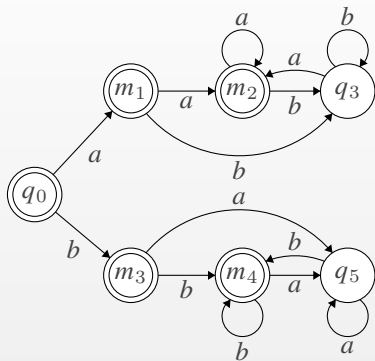
# Minimisation Example

The initial DFA is:



First, we construct the table in order to track marked pairs. We are required to track the pair only once, hence we can consider only the table part below its diagonal.

## Minimisation Example



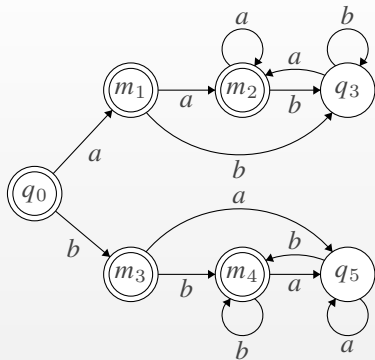|       | $q_0$ | $m_1$ | $m_2$ | $q_3$ | $m_3$ | $m_4$ |
|-------|-------|-------|-------|-------|-------|-------|
| $m_1$ |       |       |       |       |       |       |
| $m_2$ |       |       |       |       |       |       |
| $q_3$ | ✓     | ✓     | ✓     |       |       |       |
| $m_3$ |       |       |       | ✓     |       |       |
| $m_4$ |       |       |       | ✓     |       |       |
| $q_5$ | ✓     | ✓     | ✓     |       | ✓     | ✓     |

We have marked the states being not 0-bisimilar so far.

# Minimisation Example



| | | | | | | |
|---|---|---|---|---|---|---|
| $m_1$ | | | | | | |
| $m_2$ | | | | | | |
| $q_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | | |
| $m_3$ | | | | $\checkmark$ | | |
| $m_4$ | | | | $\checkmark$ | | |
| $q_5$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | $\checkmark$ | $\checkmark$ |
| | $q_0$ | $m_1$ | $m_2$ | $q_3$ | $m_3$ | $m_4$ |

Now we are checking 1-bisimilarity:

$q_0 \overset{a}{\to} m_1, m_1 \overset{a}{\to} m_2$    $\textcolor{red}{q_0 \overset{b}{\to} m_3, m_1 \overset{b}{\to} q_3}$    $\textcolor{blue}{\{m_1, m_2\}} \overset{a}{\to} m_2$    $\textcolor{blue}{\{m_1, m_2\}} \overset{b}{\to} q_3$

$q_0 \overset{a}{\to} m_1, m_2 \overset{a}{\to} m_2$    $\textcolor{red}{q_0 \overset{b}{\to} m_3, m_2 \overset{b}{\to} q_3}$

The states $q_0$ and $m_1$ are distinguishable by $b$, as well as $q_0$ and $m_2$. The states $m_1$ and $m_2$ behave equally, i.e. we can say in advance that $m_1 \sim m_2$.

# Minimisation Example



We continue checking 1-bisimilarity:

$q_0 \xrightarrow{a} m_1, m_3 \xrightarrow{a} q_5 \qquad q_0 \xrightarrow{a} m_1, m_4 \xrightarrow{a} q_5 \qquad m_1 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5$

$m_2 \xrightarrow{a} m_2, m_3 \xrightarrow{a} q_5 \qquad m_1 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5 \qquad m_2 \xrightarrow{a} m_2, m_4 \xrightarrow{a} q_5$

$q_3 \xrightarrow{a} m_2, q_5 \xrightarrow{a} q_5$

We find out that $a$-transitions distinguish almost all remaining pairs.

# Minimisation Example



| $m_1$ | $\checkmark$ | | | | | |
|---|---|---|---|---|---|---|
| $m_2$ | $\checkmark$ | | | | | |
| $q_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | | |
| $m_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $m_4$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $q_5$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| | $q_0$ | $m_1$ | $m_2$ | $q_3$ | $m_3$ | $m_4$ |

The one more exception are the states $m_3$, $m_4$, behaving equally on both terminal letters.

$\{m_3, m_4\} \xrightarrow{a} q_5 \qquad \{m_3, m_4\} \xrightarrow{b} m_4$

We cannot distinguish $m_3$ and $m_4$, as well as $m_1$ and $m_2$, by 2-bisimilarity relation, hence, the relation fixpoint is reached.

# Minimisation Example



| $m_1$ | $\checkmark$ | | | | | |
|---|---|---|---|---|---|---|
| $m_2$ | $\checkmark$ | | | | | |
| $q_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | | |
| $m_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $m_4$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $q_5$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| | $q_0$ | $m_1$ | $m_2$ | $q_3$ | $m_3$ | $m_4$ |

Now we can merge the pairs $m_1$, $m_2$ and $m_3$, $m_4$, constructing a DFA with less number of states.

## Minimisation Example

The resulting DFA is:



| $m_1$ | $\checkmark$ | | | | | |
|---|---|---|---|---|---|---|
| $m_2$ | $\checkmark$ | | | | | |
| $q_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | | |
| $m_3$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $m_4$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | | |
| $q_5$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ | $\checkmark$ |
| | $q_0$ | $m_1$ | $m_2$ | $q_3$ | $m_3$ | $m_4$ |

Nice, we have reduced the DFA. But how we can guarantee that the resulted DFA is *the least one*, i.e. there is no DFA with less number of states recognising the same language?

We can guarantee so far only the fact that the states in the given DFA cannot be merged any more.

## Suffix Congruence

Every state $q_i$ of an NFA $\mathscr{A}$ determines a language of words
$\mathscr{L}_p(q_i) = \{\omega \mid q_0 \stackrel{\omega}{\to} q_i\}$. If $\mathscr{A}$ a DFA, given $q_i \neq q_j$,
$\mathscr{L}_p(q_i) \cap \mathscr{L}_p(q_j) = \varnothing$.

If every two states in the DFA $\mathscr{A}$ are not bisimilar, then

$$\forall q_i, q_j \big(i \neq j \Rightarrow$$
$$\exists \omega_{ij} \Big( \big( \mathscr{L}_p(q_i)\{\omega_{ij}\} \subseteq \mathscr{L}(\mathscr{A}) \; \& \; \mathscr{L}_p(q_j)\{\omega_{ij}\} \nsubseteq \mathscr{L}(\mathscr{A}) \big)$$
$$\vee \big( \mathscr{L}_p(q_j)\{\omega_{ij}\} \subseteq \mathscr{L}(\mathscr{A}) \; \& \; \mathscr{L}_p(q_i)\{\omega_{ij}\} \nsubseteq \mathscr{L}(\mathscr{A}) \big) \Big) \big)$$

Namely, suffix $\omega_{ij}$ distinguishes states $q_i$ and $q_j$.

Meanwhile, all the words belonging to the same $\mathscr{L}_p(q_i)$ are congruent
(undistinguishable) with respect to suffixes.

### 👀 **Definition**

*A word $\omega$ is said to be a forbidden factor in a language $\mathscr{L}$ is no word in $\mathscr{L}$ contains $\omega$ as a subword.*

*Languages with forbidden factors are said to be thin.*

E.g. the word $ba$ is forbidden in $a^*b^*$, hence the expression $a^*b^*$ defines a thin language. On the other hand, language given by the expression $a(a|b)^*a$ has no forbidden factors.

All the words with forbidden factors in $\mathscr{L}$ are undistinguishable with respect to suffixes, i.e. lead to the trap state in any machine recognising it. However, even if $\mathscr{L}$ is not thin, its recognising machine can still contain trap states, e.g. the language of balanced parentheses has no forbidden factors, but requires the parentheses balance to be non-negative in any prefix of its words.

# Language IS THE Minimal DFA

> 👀 **Theorem**
> - *Every two equivalent DFA containing no bisimilar states are equal (up to the canonical numeration).*
> - *Hence, the minimisation procedure is confluent for DFA, and the minimal DFA is unique for any regular language.*

***Proof:*** Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be DFA both recognising the same language $\mathscr{L}$, s.t. neither $\mathscr{A}_1$ nor $\mathscr{A}_2$ contains bisimilar states. Equivalent trim DFA are bisimilar, thus non-trap parts of $\mathscr{A}_1$ and $\mathscr{A}_2$ are bisimilar, as well as the traps (if any).

For any $q_i$ in $\mathscr{A}_1$, there is a non-empty set of states $q_{i,j}$ in $\mathscr{A}_2$ that are bisimilar to $q_i$ (and vice versa). If this set is not a singleton, then all the states $q_{i,j}$ are mutually bisimilar, which contradicts the choice of $\mathscr{A}_1$ and $\mathscr{A}_2$. Therefore, $\mathscr{A}_1$ and $\mathscr{A}_2$ have equal state set cardinality.

Assume that the states are canonically numbered, and there is at least one prefix $\upsilon$ s.t. $q_0(\mathscr{A}_1) \xrightarrow{\upsilon} q_i$, $q_0(\mathscr{A}_2) \xrightarrow{\upsilon} q_j$, and $i \neq j$, i.e. $q_i \not\sim q_j$. Hence, there exists a suffix $\omega_{ij}$ distinguishing $q_i$ and $q_j$ (i.e. the tests $\upsilon\omega_{ij} \in \mathscr{L}(\mathscr{A}_1)$ and $\upsilon\omega_{ij} \in \mathscr{L}(\mathscr{A}_2)$ have distinct values), which contradicts the choice of $\mathscr{A}_1$ and $\mathscr{A}_2$. $\qquad\square$

## Suffix Congruence and Minimal DFA

Now we know that partition to $\mathscr{L}_p(q_i)$-sets wrt the minimal DFA can be considered as a relation wrt its language $\mathscr{L}$. Namely, we say that $\upsilon_1 \equiv_{\mathscr{L}} \upsilon_2$, iff $\forall \omega(\upsilon_1 \omega \in \mathscr{L} \Leftrightarrow \upsilon_2 \omega \in \mathscr{L})$, i.e. no suffix can distinguish $\upsilon_1$ and $\upsilon_2$.

### Myhill–Nerode Theorem

Language $\mathscr{L}$ in an alphabet $\Sigma$ is regular $\Leftrightarrow$ the quotient set $\Sigma^* / \equiv_{\mathscr{L}}$ is of a finite cardinality.

**Proof** ($\Leftarrow$): Let $\Sigma^* / \equiv_{\mathscr{L}}$ be of a finite cardinality. We are going to construct a DFA $\mathscr{A}$ recognising $\mathscr{L}$.

- For every equivalence class $S_i$ wrt $\equiv_{\mathscr{L}}$, we introduce a state of $\mathscr{A}$. The initial state is $S_0$ s.t. $\varepsilon \in S_0$. Final states are $S_k$ s.t. $\forall \omega \in S_k (\omega \in \mathscr{L})$.

- Given state $S_i$ and $\gamma \in \Sigma$, add transition $S_i \xrightarrow{\gamma} S_j$ s.t. $\forall \omega \in S_i (\omega \gamma \in S_j)$.

($\Rightarrow$):         Was proven already. Elements of $\Sigma^* / \equiv_{\mathscr{L}}$ are $\mathscr{L}_p(q_i)$-sets of the minimal DFA.       $\square$

The suffix congruence $\equiv_{\mathscr{L}}$ is also called the Myhill–Nerode equivalence.

# Matrix Method: Non-Regularity

## *Language Regularity Criterion, Negated*

Language $\mathscr{L}$ in an alphabet $\Sigma$ is non-regular $\Leftrightarrow$ the quotient set $\Sigma^*/\equiv_{\mathscr{L}}$ is infinite.

In order to prove that $\Sigma^*/\equiv_{\mathscr{L}}$ is infinite:

- construct an infinite series $v_1,...,v_k,...$ of prefixes of words in $\mathscr{L}$;
- construct an infinite series of <u>distinguishing suffixes</u> $\omega_1,..., \omega_k$ such that $\forall i, j \exists k \left( v_i \omega_k \in \mathscr{L} \ \& \ v_j \omega_k \notin \mathscr{L} \right)$.

If $\mathscr{L}$ is "scarse", it is convenient to choose suffixes s.t. $\forall i \left( v_i \omega_i \in \mathscr{L} \right)$, and for most $k \neq i$, $v_k \omega_i \notin \mathscr{L}$.

Then the classes can be visualised as an infinite matrix with distinct rows containing 1's on its leading diagonal.

NB: if $\mathscr{L}$ is dense, then it is better to choose an opposite strategy, constructing a matrix containing 0's on its leading diagonal.

## Matrix Method: Non-Regularity

If $\mathscr{L}$ is "scarse", it is convenient to choose suffixes s.t. $\forall i\big(v_i\omega_i \in \mathscr{L}\big)$, and for most $k \neq i$, $v_k\omega_i \notin \mathscr{L}$.

Let us show non-regularity of the language
$$\left\{ w_1w_2 \mid w_1 \neq \varepsilon \,\&\, |w_1| = 2 \cdot k \,\&\, w_1 = w_1^R \,\&\, w_i \in \{a, b\}^* \right\}$$

Given $\omega_i = v_i = ab^{2 \cdot i - 1}a$,
$$v_i\omega_j = \begin{cases} ab^{2 \cdot i - 1}aab^{2 \cdot i - 1}a, \text{ palindrome, if } i = j; \\ ab^{2 \cdot i - 1}aab^{2 \cdot j - 1}a, \text{never starts with an even palindrome, if } i \neq j. \end{cases}$$

The classes matrix:

|            | $aba$ | $ab^3a$ | $\ldots$ | $ab^{2 \cdot k - 1}a$ | $\ldots$ |
|------------|-------|---------|----------|------------------------|----------|
| $aba$      | +     | −       |          | −                      |          |
| $ab^3a$    | −     | +       |          | −                      |          |
| $\ldots$   |       | $\ldots$ |         |                        |          |
| $ab^{2 \cdot k - 1}a$ | − | −  |          | +                      |          |
| $\ldots$   |       |         |          |                        |          |

There $v_i$ name rows, $\omega_i$ name columns, **+** at position $(i, j)$ denotes the fact that $v_i\omega_j \in \mathscr{L}$; and **−** at position $(i, j)$ denotes the fact that $v_i\omega_j \notin \mathscr{L}$.

## Bounded Languages

### ◑◑ __Definition__

*Language $\mathcal{L}$ is said to be bounded if $\mathcal{L} \subseteq \omega_1^* ... \omega_n^*$, where $\omega_1,...,\omega_n$ are fixed words.*

The bounded languages are of extreme usefulness further in the course (when investigating about the context-freeness property), but they are still of interest even in the regular case.

When discussing "scarse" and "dense" languages, we assume that the notions of scarcity and density are given in the context of the enclosing bounded language.

E.g. the language $\mathcal{L}_{\neq} = \{a^m b^n \mid n \neq m\}$ is thin in $\{a, b\}^*$ (since its word contain no subword $ba$); but in the bounded context of $a^* b^*$ it is dense (for almost all pairs $n, m$, $a^n b^m \in \mathcal{L}_{\neq}$).

Indeed, the trap state equivalence class corresponding to the words with forbidden factors is useless in constructing infinite classes matrix, so it is natural to omit it.

## Pumping Lemma vs. Classes Matrix Method

**Pumping Lemma:**

- requires <u>a unique</u> parameterized word.
- usually requires preliminary intersection with a bounded language, in order to narrow a set of pumped subwords.

**Matrix Method:**

- requires constructing <u>series</u> of words, not necessarily parameterized.
- normally does not require any intersection tricks, since we are free to restrict combinations of subwords by the given series.

The matrix method is far more universal and flexible. However, in a few cases the pumping lemma is more convenient.

*Let us prove that $\mathscr{L} = \{\omega \mid \omega$ does not contain a factor of power $4\}$ in $\{a, b\}^*$ is non-regular.*

- *We know that $\mathscr{L}$ is infinite, since Thue words (given by iterative application of Thue morphism $h(a) = ab$, $h(b) = ba$) belong to $\mathscr{L}$.*

- *Then, if $\mathscr{L}$ is regular and its pumping length is $N$, there is some Thue word $\omega$ s.t. $|\omega| > N$, and $\omega = \omega_1 \omega_2 \omega_3$, $\forall i (\omega_1 \omega_2^i \omega_3 \in \mathscr{L})$. But that can never happen for $i \geq 4$, by definition of $\mathscr{L}$.*