Finite-Memory Automata
A program input is a sequence of atomic symbols over an infinite alphabet $\Sigma$, and a program itself consists of a specification of a finite set of variables $v_i$, $i = 1, 2, \ldots, r$, and a finite sequence of (labeled) commands of the following type.

- $v_i := \sigma$
- read ($v_i$)
- type ($v_i$)
- $v_i := v_j$
- if $v_i = v_j$, then go to $k$
- halt
Let $\Sigma$ be an infinite alphabet and let $\#$ be a symbol not belonging to $\Sigma$. An assignment is a word $w_1w_2 \cdots w_r \in (\Sigma \cup \{\#\})^*$ such that if $w_i = w_j$ and $i \neq j$, then $w_i = \#$. The set of all assignments of length $r$ is denoted by $\Sigma^r\neq$.

For a word $w = w_1w_2 \cdots w_r \in (\Sigma \cup \{\#\})^*$ we define the content of $w$, denoted $[w]$, by $[w] = \{w_i : i = 1, 2, \ldots, r\}$. 
A finite-memory automaton is a system $A = \langle S, s, u, \rho, \mu, F \rangle$, where

- $S$ is a finite set of states,
- $s \in S$ is the initial state,
- $u = u_1 u_2 \cdots u_r \in \Sigma^r$ is the initial assignment,
- $\rho : S \to \{1, 2, \ldots, r\}$ is a partial function called the reassignment,
- $\mu \subseteq S \times \{1, 2, \ldots, r\} \times S$ is the transition relation, and
- $F \subseteq S$ is the set of final states.

The automaton $A$ can be represented by its initial assignment and a directed graph whose vertices are states. There is an edge from $p$ to $q$, if there exists an index $k$ such that $(p, k, q) \in \mu$. Such edge is labeled $k$. Also, if for a vertex $p$ the value of $\rho$ is defined, then $p$ is labeled $\rho(p)$. 
Example  Let $A = \langle \{s, p, f\}, s, \#\#, \rho, \mu, \{f\} \rangle$, where

- $\rho(s) = 1, \rho(p) = \rho(f) = 2$; and
- $\mu = \{(s, 1, s), (s, 1, p), (p, 1, f), (p, 2, p), (f, 1, f), (f, 2, f)\}$.

$L(A) = \{\sigma_1 \sigma_2 \cdots \sigma_n : \text{there exist } 1 \leq i < j \leq n \text{ such that } \sigma_i = \sigma_j\}$.

An accepting run of $A$ on $abc\text{#d}$ is

$$(s, \#\#), (s, a\#), (p, b\#), (p, bc), (f, bc), (f, bd).$$
An *actual* state of $A$ is a state from $S$ together with the content of all its registers. Thus, $A$ has infinitely many states which are pairs $(p, w)$, where $p \in S$ and $w \in \Sigma^r \neq$. These are called the *configurations* of $A$. The set of all configurations of $A$ is denoted by $S^c$. The pair $s^c = (s, u)$ is called the *initial* configuration, and the configurations with the first component in $F$ are called *final* configurations. The set of final configurations is denoted by $F^c$. 
The transition relation $\mu$ induces the following relation $\mu^c$ on $S^c \times \Sigma \times S^c$.

Let $v, w \in \Sigma^r \setminus \{v\} = v_1 v_2 \cdots v_r$ and $w = w_1 w_2 \cdots w_r$. Then $((p, v), \sigma, (q, w)) \in \mu^c$ if the two following conditions are satisfied.

- If $\sigma = v_k \in [v]$, then $w = v$ and $(p, k, q) \in \mu$.

- If $\sigma \notin [v]$, then $\rho(p)$ is defined, $w_{\rho(p)} = \sigma$, for each $k \neq \rho(p)$, $w_k = v_k$, and $(p, \rho(p), q) \in \mu$.

Let $\sigma \in \Sigma^*$, $\sigma = \sigma_1 \sigma_2 \cdots \sigma_n$. A run of $A$ on $\sigma$ consists of a sequence of configurations $c_0, c_1, \ldots, c_n$ such that $c_0 = s^c$ and $(c_{i-1}, \sigma_i, c_i) \in \mu^c, i = 1, 2, \ldots, n$.

We say that $A$ accepts $\sigma$ if there exists a run $c_0, c_1, \ldots, c_n$ of $A$ on $\sigma$ such that $c_n \in F^c$. The set of all words accepted by $A$ is denoted by $L(A)$ and is referred to as a quasi-regular language.
Example Let $\Sigma' = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ be an $r$-element subset of $\Sigma$ and let $A' = \langle S, s, \mu', F \rangle$ be an ordinary finite automaton over $\Sigma'$. Consider a finite-memory automaton $A = \langle S, s, u, \rho, \mu, F \rangle$, where

- $u = \sigma_1 \sigma_2 \cdots \sigma_r$,
- the reassignment $\rho$ is nowhere defined, and
- $(p, k, q) \in \mu$ if and only if $(p, \sigma_k, q) \in \mu'$.

Then $L(A) = L(A')$. That is, every regular language is quasi-regular.
Example  Let $A$ be the following finite-memory automaton.

\begin{center}
\begin{tikzpicture}[node distance=2cm, thick, main/.style = {draw, circle}]
  \node[main] (s) {$s,1$};
  \node[main] (q1) [right of=s] {$q_{1,2}$};
  \node[main] (q2) [right of=q1] {$q_2$};
  \node[main] (f) [below of=q1] {$f$};
  \node[main] (q4) [right of=f] {$q_4$};
  \node[main] (q3) [below of=q4] {$q_{3,1}$};

  \path[->]
  (s) edge node [above] {1} (q1)
  (q1) edge node [above] {2} (q2)
  (q1) edge node [right] {1} (f)
  (q4) edge node [right] {1} (q3)
  (q4) edge node [right] {2} (q1)

\end{tikzpicture}
\end{center}

\begin{center}
\begin{tabular}{|c|c|}
\hline
# & # \\
\hline
\end{tabular}
\end{center}

\textit{initialization}
Let $n \geq 1$, and let $\tau_0, \tau_1, \ldots, \tau_{2n}$ be pairwise different elements of $\Sigma$. Consider a word $\sigma = \sigma_1\sigma_2 \cdots \sigma_{4n+2}$, where

- $\sigma_1 = \sigma_3 = \tau_0$,
- $\sigma_{4n} = \sigma_{4n+2} = \tau_{2n}$, and
- $\sigma_{2i} = \sigma_{2i+3} = \tau_i$ for $i = 1, 2, \ldots, 2n - 1$.

That is, $\sigma$ is of the form

```
  2 2 2
* * * * * * * * * * * * * * * *
1 1 1 1 1 1
```

Then $\sigma \in L(A)$, but $\sigma$ has no non-empty pattern that may be pumped.
Proposition  Let \( A = \langle S, s, u, \rho, \mu, F \rangle \) be an \( r \)-register finite-memory automaton and let \( \Sigma' \) be a finite subset of \( \Sigma \). Then \( L(A) \cap \Sigma'^* \) is a regular language (over \( \Sigma' \)).

Proof  Consider an ordinary finite automaton \( A' = \langle S', s', \mu', F' \rangle \) over \( \Sigma' \) that is defined as follows.

- \( S' = S^c \cap (S \times (\Sigma' \cup [u] \cup \{#\})^r) \). Since \( \Sigma' \) is finite, \( S' \) is finite as well.
- \( s' = (s, u) \).
- \( \mu' = \mu^c \cap (S' \times \Sigma' \times S') \).
- \( F' = F^c \cap S' \).

Let \( \sigma \) be a word over \( \Sigma' \). Then each accepting run of \( A \) on \( \sigma \) is an accepting run of \( A' \) on \( \sigma \), and vice versa. Thus, \( \sigma \in L(A) \cap \Sigma'^* \) if and only if \( \sigma \in L(A') \). \( \square \)
Lemma  Let $A = \langle S, s, u, \rho, \mu, F \rangle$ be a finite-memory automaton. Then for each automorphism $\iota : \Sigma \to \Sigma$, $\iota(L(A)) = L(\iota(A))$, where $\iota(A) = \langle S, s, \iota(u), \rho, \mu, F \rangle$.

Proof (sketch) We prove by induction on the length of $\sigma$ that

$$(s_0, w_0), (s_1, w_1), \ldots, (s_n, w_n)$$

is a run of $A$ on $\sigma$ if and only if

$$(s_0, \iota(w_0)), (s_1, \iota(w_1)), \ldots, (s_n, \iota(w_n))$$

is a run of $\iota(A)$ on $\iota(\sigma)$.

The induction step is based on the fact that if $((p, v), \sigma, (q, w)) \in \mu^c$, then $((p, \iota(v)), \iota(\sigma), (p, \iota(w))) \in \mu^c$. \hfill \Box

Corollary  (Closure under automorphisms) Let $A = \langle S, s, u, \rho, \mu, F \rangle$ be a finite-memory automaton. Then for each automorphism $\iota : \Sigma \to \Sigma$ that is an identity on $[u]$ and each $\sigma \in \Sigma^*$, $\sigma \in L(A)$ if and only if $\iota(\sigma) \in L(A)$. 

11
Proof The result immediately follows from the lemma, because, under the conditions of the corollary, \( \iota(A) = A \). \qed
Proposition (Indistinguishability property of finite-memory automata)

Let \( A = \langle S, s, \mathbf{u}, \rho, \mu, F \rangle \) be an \( r \)-register finite-memory automaton. If \( xy \in L(A) \), then there exists a subset \( \Sigma' \) of \( [x] \) such that the number of elements of \( \Sigma' \) does not exceed \( r \) and the following holds.

For any \( \sigma \notin \Sigma' \) and any \( \tau \notin [y] \cup \Sigma' \), the word \( x(y(\sigma/\tau)) \) obtained from \( xy \) by the substitution of \( \tau \) for each occurrence of \( \sigma \) in \( y \) is in \( L(A) \).

Proof Let \( x \) be a word of length \( i \) and let \((s_0, w_0), (s_1, w_1), \ldots, (s_n, w_n)\) be an accepting run of \( A \) on \( xy \). Let \( \Sigma' = [w_i], \sigma \notin [w_i], \) and \( \tau \notin [y] \cup \Sigma' \).

To prove that \( x(y(\sigma/\tau)) \in L(A) \), it suffices to show that \( y(\sigma/\tau) \in L(A_{(s_i, \#1_i)}) \), where \( A_{(s_i, \#1_i)} = \langle S, s_i, w_i, \rho, \mu, F \rangle \). Let \( \iota \) be the automorphism of \( \Sigma \) that permutes \( \sigma \) with \( \tau \) and leaves fixed all other symbols. Then \( y(\sigma/\tau) = \iota(y) \), and the result follows the above corollary, because neither \( \sigma \) nor \( \tau \) is in \([w_i]\). \( \square \)
Example Consider a language $L$ that consists of all words whose last symbol is different from all others. That is,

$$L = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_n, \ i = 1, 2, \ldots, n - 1 \}.$$

Assume to the contrary that $L$ is accepted by an $r$-register finite-memory automaton $A$.

Let $x = \sigma_1 \sigma_2 \cdots \sigma_r \sigma_{r+1}$ and $y = \sigma_{r+2}$, where all $\sigma_i$s are pairwise different. Then $xy \in L (= L(A))$.

Let $\Sigma'$ be a subset of $[x]$ provided by the *indistinguishability property* of finite-memory automata. Since the number of elements of $\Sigma'$ does not exceed $r$, there exists an $i \in \{1, 2, \ldots, r + 1\}$ such that $\sigma_i \notin \Sigma'$. Since $[x] \cap [y] = \emptyset$, $\sigma_i \notin [y] \cup \Sigma'$. Therefore, by the *indistinguishability property* of finite-memory automata, $x(y(\sigma_{r+2}/\sigma_i)) \in L(A)$. However in the last word $\sigma_i$ appears both in the $i$th and the last positions which contradicts the assumption $L(A) = L$. 
Proposition If an \( r \) register finite memory automaton \( A \) accepts a word of length \( n \), then it accepts a word of length \( n \) that contains at most \( r \) pairwise different symbols.

Proof (sketch) Let \( \sigma = \sigma_1 \sigma_2 \cdots \sigma_n \in L(A) \) contain more than \( r \) pairwise different symbols, and let

\[
r = (s_0, w_0), (s_1, w_1), \cdots, (s_n, w_n),
\]

\( w_i = w_i, 1 \cdots w_i, r, \ i = 0, 1, \ldots, n, \) be a run of \( A \) on \( \sigma \). Let \( i \) be the minimal integer such that \( \sigma_i \notin [w_{i-1}] \) and \( w_{i-1}, \rho(s_{i-1}) \neq \# \).

Let \( \iota \) be an automorphism of \( \Sigma \) such that interchanges \( \sigma_i \) with \( w_{i-1}, \rho(s_{i-1}) \) and leaves fixed all other symbols. Then

\[
r' = (s, u), (s_1, w_1), \cdots, (s_{i-1}, w_{i-1}), (s_i, \iota(w_i)), \cdots, (s_n, \iota(w_n))
\]

is an accepting run of \( A \) on \( \sigma' = \sigma_1 \cdots \sigma_{i-1} \iota(\sigma_i \cdots \sigma_n) \).
Example  Let

\[ L = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \text{there exist } 1 \leq i < j \leq n \text{ such that } \sigma_i = \sigma_j \}. \]

Then \( \bar{L} \) consists of all words in which each symbol appears at most one time. We contend that \( \bar{L} \) is nor quasi-regular.

Assume to the contrary that \( \bar{L} \) is accepted by an \( r \)-register finite-memory automaton \( A \). Since \( \Sigma \) is infinite, there exists a word \( \sigma \in L(A) \) of length \( r + 1 \). However, \( A \) must accept a word \( \sigma' \) of length \( r + 1 \) that contains at most \( r \) pairwise different symbols. Therefore, some symbol of \( \Sigma \) appears in \( \sigma' \) more than one time, in contradiction with the assumption \( L(A) = \bar{L} \).

Thus, quasi-regular sets are not closed under complementation.
**Theorem**  The emptiness problem for quasi-regular languages is decidable.

**Proof** Let $A = \langle S, s, u, \rho, \mu, F \rangle$, be an $r$-register finite-memory automaton and let $\Sigma' = [u] \cup \{\sigma_1, \ldots, \sigma_\ell\}$ be an $r$-element subset of $\Sigma$ such that $[u] \cap \{\sigma_1, \ldots, \sigma_\ell\} = \emptyset$. We contend that $L(A) \neq \emptyset$ if and only if $L(A) \cap \Sigma'^* \neq \emptyset$.

The "if" part is immediate. Let $L(A) \neq \emptyset$. There exists a subset $\Sigma'' = [u] \cup \{\tau_1, \ldots, \tau_\ell\}$ of $\Sigma$ such that $L(A) \cap \Sigma''^* \neq \emptyset$. Let $\iota$ be an automorphism of $\Sigma$ that interchanges $\sigma_i$ with $\tau_i$, $i = 1, 2, \ldots, \ell$, and leaves fixed all other symbols. Then,

$$L(A) \cap \Sigma'^* = L(A) \cap \iota(\Sigma''^*) = \iota(L(A) \cap \Sigma''^*).$$

Since $L(A) \cap \Sigma''^* \neq \emptyset$, $L(A) \cap \Sigma'^* \neq \emptyset$ as well. \hfill \Box

**Theorem**  For a two-register finite-memory automaton $A'$ and for a finite-memory automaton $A''$ it is decidable whether $L(A'') \subseteq L(A')$. 
Closure properties of quasi-regular languages

**Theorem**  The quasi-regular sets are closed under union, intersection, concatenation, and iteration (Kleene star).

**Example**  Let $\Sigma = \{\sigma_1, \sigma_2, \ldots\}$, $\Sigma' = \{\tau_1, \tau_2, \ldots\}$, and $\iota: \Sigma^* \rightarrow \Sigma'^*$ be a homomorphism defined by $\iota(\sigma_{3i}) = \iota(\sigma_{3i-1}) = \tau_{2i}$ and $\iota(\sigma_{3i-2}) = \tau_{2i-1}$, $i = 1, 2, \ldots$. Let $A$ be the following finite-memory automaton over $\Sigma$.

![Diagram of the automaton](image)

Initialization

18
Then

\[ L(A) = \{\sigma_i \sigma_j : i \neq j\} \]

and

\[ \iota(L(A)) = \{\tau_i \tau_j : i \neq j\} \cup \{\tau_{2i} \tau_{2i} : i = 1, 2, \ldots\}. \]

Assume that \( \iota(L(A)) \) is quasi-regular, and let \( A' \) be a finite-memory automaton over \( \Sigma' \) such that \( L(A') = \iota(L(A)) \). Let \( i \) be such that neither \( \tau_{2i} \) nor \( \tau_{2i+1} \) appear in the initial assignment of \( A' \), and let \( \iota' \) be an automorphism of \( \Sigma' \) that interchanges \( \tau_{2i} \) with \( \tau_{2i+1} \) and leaves fixed all other symbols. Since \( \tau_{2i} \tau_{2i} \in \iota(L(A)) \),

\[ \tau_{2i+1} \tau_{2i+1} \in \iota(L(A)) \left( = \{\tau_i \tau_j : i \neq j\} \cup \{\tau_{2i} \tau_{2i} : i = 1, 2, \ldots\} \right), \]

which is impossible.

Thus, quasi-regular languages are not closed under homomorphisms.
Example  Let $\Sigma = \{\sigma_1, \sigma_2, \ldots\}$, $\Sigma' = \{\tau_1, \tau_2, \ldots\}$, and $\iota: \Sigma \to \Sigma'^*$ be the homomorphism defined by $\iota(\sigma_{3i}) = \iota(\sigma_{3i-1}) = \tau_{2i}$ and $\iota(\sigma_{3i-2}) = \tau_{2i-1}$, $i = 1, 2, \ldots$. Let $A'$ be a finite-memory automaton over $\Sigma'$ defined by the following diagram.

![Diagram](image)

initialization
Then,
\[ L(A') = \{\tau_i \tau_i : i = 1, 2, \ldots\} \]
and
\[ \iota^{-1}(L(A')) = \bigcup_{i=1}^{\infty} \{\sigma_i \sigma_i, \sigma_{3i-1} \sigma_{3i}, \sigma_{3i} \sigma_{3i-1}\}. \]

Assume that \( \iota^{-1}(L(A')) \) is quasi-regular, and let \( A \) be a finite-memory automaton over \( \Sigma \) such that \( L(A) = \iota^{-1}(L(A')) \). Let \( i \) be such that neither \( \sigma_{3i-2} \) nor \( \sigma_{3i-1} \) appears in the initial assignment of \( A \) and let \( \iota' \) be an automorphism of \( \Sigma' \) that interchanges \( \tau_{3i-2} \) and \( \tau_{3i-1} \) and leaves fixed all other symbols. Since \( \sigma_{3i-1} \sigma_{3i} \in \iota^{-1}(L(A')) \),
\[ \sigma_{3i-2} \sigma_{3i} \in \iota^{-1}(L(A')) (= \bigcup_{i=1}^{\infty} \{\sigma_i \sigma_i, \sigma_{3i-1} \sigma_{3i}, \sigma_{3i} \sigma_{3i-1}\}), \]
which is impossible.

Thus, quasi-regular languages are not closed under inverse homomorphisms.
Remark Under a very weak assumption it can be shown that any class $L$ of languages over an infinite alphabet which is defined by a set of machines having a finite description is not closed under either homomorphisms or inverse homomorphisms.

First we observe that, since the set of machines having a finite description is countable, $L$ is countable.

We prove that $L$ is not closed under homomorphisms under the assumption that $\Sigma = \{\sigma_1, \sigma_2, \ldots\} \in L$.

Since $L$ is countable, there exists an infinite subset $L = \{\sigma_{j_1}, \sigma_{j_2}, \ldots\}$ of $\Sigma$ such that $L \notin L$. Let $\iota : \Sigma \rightarrow \Sigma$ be defined by $\iota(\sigma_i) = \sigma_{j_i}$, $i = 1, 2, \ldots$. Then $\iota(\Sigma) = L$, which shows that $L$ is not closed under homomorphisms.

We prove that $L$ is not closed under inverse homomorphisms under the assumption that $\{\sigma_1\} \in L$.

Let $\iota' : \Sigma \rightarrow \Sigma$ be defined by $\iota(\sigma) = \sigma_1$, if $\sigma \in L$; and $\iota(\sigma) = \sigma_2$, otherwise. Then $\iota^{-1}(\{\sigma_1\}) = L$, which shows that $L$ is not closed under inverse homomorphisms.
Example  Consider the language

\[ L = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_1, \ i = 2, 3, \ldots, n \}. \]

That is, \( L \) consists of all words whose first symbol is different from all other symbols, and is accepted by the following finite-memory automaton.

The reversal \( L^R \) of \( L \) language consists of all words whose last symbol is different from all others, which is not quasi-regular.
Deterministic finite-memory automata

An $r$-register finite-memory automaton $A = \langle S, s, u, \rho, \mu, F \rangle$ is called deterministic if $\rho$ is everywhere defined and for each $p \in S$ and each $k = 1, 2, \ldots, r$ there exists exactly one $q \in S$ such that $(p, k, q) \in \mu$. That is, $\rho$ is a function from $S$ into $\{1, 2, \ldots, r\}$ and $\mu$ can be thought of as a function from $S \times \{1, 2, \ldots, r\}$ into $S$.

**Theorem** The languages accepted by deterministic finite-memory automata are closed under complementation, union and intersection.
**Example**  Consider the following deterministic finite-memory automaton.

The language $L$ accepted by this automaton consists exactly of those words where the first symbol appears twice or more:

$$L = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \text{for some } i = 2, 3, \ldots, n, \ \sigma_i = \sigma_1 \}.$$
Therefore,

\[ L = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_1, i = 2, 3, \cdots, n \}, \]

implying

\[ L^R = \{ \sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_1, i = 2, 3, \cdots, n \}^R. \]

Were \( L^R \) be deterministic, its complement

\[ \{ \sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_1, i = 2, 3, \cdots, n \}^R. \]

would also be deterministic, in contradiction with the previous example.
Example  Consider the following deterministic finite-memory automaton.

This automaton accepts the language

$$L = \{\sigma_1 \sigma_2 \cdots \sigma_n : \sigma_1 = \sigma_n, n > 1\}.$$
Assume $L^* = L(A)$, where $A = \langle S, s, u, \rho, \mu, F \rangle$ is an $r$-register deterministic finite-memory automaton. Let $\sigma_1, \sigma_2, \cdots, \sigma_{r+1}$ be pairwise different elements of $\Sigma$. Then, for each $i = 1, 2, \cdots, r + 1$,

$$\sigma_1 \sigma_1 \sigma_2 \sigma_1 \sigma_3 \cdots \sigma_1 \sigma_r \sigma_1 \sigma_{r+1} \sigma_i \in L^*. $$

There is a unique configuration $(p, w)$ that $A$ can enter after reading

$$\sigma_1 \sigma_1 \sigma_2 \sigma_1 \sigma_3 \cdots \sigma_1 \sigma_r \sigma_1 \sigma_{r+1}. $$

Then, for each $i = 1, 2, \cdots, r + 1$, $A_{(p, \#1)}$ must accept $\sigma_i$. Since $A_{(p, \#1)}$ has $r$ registers, for some $i = 1, 2, \cdots, r + 1$, $\sigma_i \not\in [w]$. Let $\tau$ be a symbol different from any of the $\sigma_i$s and let $\iota$ be an automorphism of $\Sigma$ that interchanges $\tau$ and $\sigma_i$ and leaves fixed all other symbols. Then $A_{(p, \#1)}$ accepts $\tau$. Therefore $A$ accepts $\sigma_1 \sigma_1 \sigma_2 \sigma_1 \sigma_3 \cdots \sigma_1 \sigma_r \sigma_1 \sigma_{r+1} \tau$, which is impossible, because no suffix of that word belongs to $L$. 

28
Deterministic two-way finite-memory automata

A two-way deterministic finite-memory automaton is a system $A = \langle S, s, u, \rho, \mu, F \rangle$, where $S$, $s$, $u$, $\rho$, and $F$ are as in a deterministic finite-memory automaton. Inputs to $A$ are of the form $\sigma$, where $\not\in \Sigma$ and $\sigma \in \Sigma$, and the transition function $\mu$ maps $S \times \{1, 2, \ldots, r\}$ into $S \times \{-1, 1\}$.

The meaning of $\mu$ is as follows. If $\mu(p, k) = (q, -1)$, then in state $p$, scanning the input symbol stored in the $k$th register, $A$ enters the state $q$ and moves left.

Similarly, if $\mu(p, k) = (q, 1)$, then in state $p$, scanning the input symbol stored in the $k$th register, $A$ enters state $q$ and moves right.
Example  Let $L = \{\sigma_1 \sigma_2 \cdots \sigma_n : \sigma_i \neq \sigma_j \text{ for } i \neq j\}$.

Observe that $\sigma_1 \sigma_2 \cdots \sigma_n \in L$ if and only if for each $i = 2, 3, \ldots, n$, $\sigma_1 \sigma_2 \cdots \sigma_i \in L$.

Given an input $\sigma = \sigma_1 \sigma_2 \cdots \sigma_n$, our automaton first stores $\sigma_1$ in the first register and then for each $i = 2, 3, \ldots, n$ verifies whether $\sigma_1 \sigma_2 \cdots \sigma_i \in L$. For such verification the automaton performs the following sequence of moves.

After “accepting” $\sigma_1 \sigma_2 \cdots \sigma_{i-1}$, the automaton checks whether $\sigma_i = \sigma_1$. If the equality holds, then the automaton enters a “dead state”.

If $\sigma_i \neq \sigma_1$, the automaton stores $\sigma_i$ in the second register and starts moving left from $\sigma_i$ towards $\sigma_1$ trying to find out whether for some $j = 2, 3, \ldots, i-1$, $\sigma_j = \sigma_i$. If such a $j$ exists, then $\sigma \notin L$ and the automaton enters a dead state. Otherwise the automaton will eventually reach $\sigma_1$.

Since the automaton already “knows” from the previous verification that $\sigma_1 \sigma_2 \cdots \sigma_{i-1} \in L$, arriving to $\sigma_1$ indicates that it is at the left end of $\sigma$ and $\sigma_1 \sigma_2 \cdots \sigma_i \in L$.

After arriving at the left end of the input, the automaton turns right and moves to $\sigma_i$. From $\sigma_i$ it moves right, enters a final state, and repeats
the same procedure starting from $\sigma_{i+1}$, etc..