אלכ"א ישאר

שתל ישלות
and alphabet symbols to states. A finite-state automaton is a five-tuple \( \langle A, \Sigma, \delta, q_0, \mathcal{F} \rangle \) where \( A \) is a finite set of states, \( \mathcal{F} \) is the initial state, \( q_0 \in A \) is an initial state, \( \delta : O \times \Sigma \times O \) is a relation from states and alphabet symbols to states. Languages are models of computation: they compute
Example: Finite-state automaton

Finite-state automata
\[
\begin{align*}
\varphi \in (\varphi^*, \gamma) \text{ then } \exists \varphi^* \in (\varphi, \gamma, \varphi)
\end{align*}
\]

and
\[
\exists \varphi^* \in (\varphi, \gamma, \varphi)
\]

for every string \( m \in \varphi^* \) and letter \( \alpha \in \varphi \).

\[
\exists \varphi^* \in (\varphi, \gamma, \varphi)
\]

for every state \( \gamma \in \varphi^* \).

is a new relation, \( \varphi^* \) defined as follows:

The reflexive transitive extension of the transition relation \( \varphi^* \)

\textbf{Finite-state automata}
\[ \langle h_0, c\alpha, h_3 \rangle, \langle h_0, c\alpha, h_2 \rangle, \langle h_0, c\alpha, h_1 \rangle, \langle h_0, c\alpha, h_3 \rangle, \langle h_0, c\alpha, h_2 \rangle, \langle h_0, c\alpha, h_1 \rangle, \langle h_0, c\alpha, h_0 \rangle \]

\( \sigma \) is the following set of triples:

For the finite-state automaton:

Example: Paths

Finite-state automata
Finite-state automata

The language of the finite-state automaton:

Example: Language

if all string it accepts:

A string $w$ is accepted by the automaton $A$ if and only if there exists a state $q' \in fB$ such that $\langle m, q', 0b, q' \rangle = (m, 0b, q)$.
Example: Some finite-state automata

Finite-state automata
Example: Some finite-state automata

Finite-state automata

\{ \nu \} \quad \text{or}

\begin{array}{c}
\text{\nu} \\
\text{b}
\end{array}

\begin{array}{c}
\text{\nu} \\
\text{0}
\end{array}
Example: Some finite-state automata

Finite-state automata
Example: Some finite-state automata

Finite-state automata
Example: Some finite-state automata

Finite-state automata
Example: Some finite-state automata

Finite-state automata
The language accepted by the following automaton is:

\{ \text{do, undo, done, undone} \}

Example: Automata with \( e \)-moves

\[ \delta \times (\{ e \} \cup \mathcal{Z}) \times \delta \supseteq \delta \]

An extension: \( e \)-moves.

Finite-state automata
Finite-state automata is the class of regular languages.

Theorem (Kleene, 1956): The class of languages recognized by finite-state automata
Example: Finite-state automata and regular expressions

Finite-state automata
Operations on Finite-State Automata

- Determinization
- Minimization
- Intersection
- Union
- Concatenation
Example: Equivalent automata

Minimization and Determinization
Lexicon:

Language processing

Applications of finite-state automata in
A naive morphological analyzer:

Language Processing

Applications of Finite-State Automata in
While regular expressions are sufficiently expressive for many natural language applications, it is sometimes useful to define relations over two sets of strings.
said the cat in the hat

I know some new tricks

Part-of-speech tagging:

Regular relations
Morphological analysis:

Regular relations
Regular Relations

Singular-to-plural mapping:
cat hat ox child
cows hats oxen children mice shear sheep goose
necessarily disjoint (or different).

states, and are alphabets: finite sets of symbols, not is the initial state, is the set of final (or accepting) Similarly to automata, is a finite set of states, A finite-state transducer is a six-tuple

Finite-state transducers
Finite-state transducers

Adding e-moves:

Shorthand notation:
The language of a finite-state transducer is a language of pairs: a binary relation over \( \mathbb{Z} \times \mathbb{Z} \). The language of a finite-state transducer is defined analogously to how the language of an automaton is defined.
Example: English-to-French

\[ a : A, b : B, c : C, \ldots \]

The uppercase transducer

Finite-state transducers
\[ \mathcal{Q}_? \text{ if for some } a \in \mathcal{A}, \langle q_1, a, q_2, q_0', \delta, q_0 \rangle \text{ is lower automaton is } \bullet \]

\[ \mathcal{Q}_? \text{ if for some } q \in \mathcal{Q}_?, \langle q_1, a, q_2, q_0', \delta, q_0 \rangle \text{ is upper automaton is } \bullet \]

\[ \mathcal{Q}_? \text{ if for some } \langle q_1, a, q_2, q_0', \delta, q_0 \rangle \text{ is underlying automaton is } \bullet \]

Given a transducer of finite-state transducers

Properties of Finite-State Transducers
The inverse transducer is $(Q, q_0, \Sigma, \delta, q_f, \epsilon)$. If $(q_1, a, b, q_2) \in \delta^{-1}$ iff $(q_1, b, a, q_2) \in \delta$.

If $T$ is a transducer, there exists a transducer $T^{-1}$ such that for every $u \in \Sigma^*$, $T^{-1}(w) = \{ u \in \Sigma^* \mid w \in T(u) \}$.

A transducer $T$ is functional if for every $w \in \Sigma$, $T(w)$ is either empty or a singleton.

Transducers are closed under union: if $T_1$ and $T_2$ are transducers, there exists a transducer $T$ such that for every $w \in \Sigma^*$, $T(w) = T_1(w) \cup T_2(w)$.
The number of states in the composition transducer might be

\[(\mathcal{L})^{\dag} \mathcal{L} = (\mathcal{L})^{\dag} \mathcal{L} \quad \text{I.e.} \quad \mathcal{L} \in \mathcal{L}

for every transducer, there exists a transducer such that for every transducers are closed under composition: if \(I\) and \(J\) are

Properties of finite-state transducers
Transducers with no e-moves are closed under intersection.

\[ \{uq_u \alpha \} = (u \alpha)^* L \cup \dagger L \]

\[ \left\{ 0 \preceq w \mid uq_w \alpha \right\} = (u \alpha)^* L \]

\[ \left\{ 0 \preceq w \mid uq_w \alpha \right\} = (u \alpha)\dagger L \]

Transducers are not closed under intersection.

Properties of finite-state transducers
Weights

Closed under composition

Not closed under intersection (and hence complementation)

Closed under concatenation, Kleene-star, union

Denote regular relations

Computationally efficient

Properties of finite-state transducers