עיבוד שפות טבעיות

שולי וינטנר

Finite-state automata

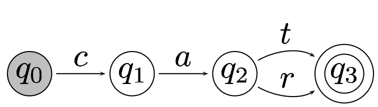
Automata are models of computation: they compute languages.

A finite-state automaton is a five-tuple $\langle Q, q_0, \Sigma, \delta, F\rangle$, where Σ is a finite set of **alphabet** symbols, Q is a finite set of **states**, $q_0 \in Q$ is the **initial state**, $F \subseteq Q$ is a set of **final** (accepting) states and $\delta : Q \times \Sigma \times Q$ is a relation from states and alphabet symbols to states.

Finite-state automata

Example: Finite-state automaton

- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{c, a, t, r\}$
- $F = \{q_3\}$
- $\delta = \{\langle q_0, c, q_1\rangle, \langle q_1, a, q_2\rangle, \langle q_2, t, q_3\rangle, \langle q_2, r, q_3\rangle\}$



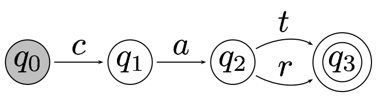
Finite-state automata

The reflexive transitive extension of the transition relation δ is a new relation, $\hat{\delta}$, defined as follows:

- for every state $q \in Q$, $(q, \epsilon, q) \in \hat{\delta}$
- for every string $w \in \Sigma^*$ and letter $a \in \Sigma$, if $(q, w, q') \in \hat{\delta}$ and $(q', a, q'') \in \delta$ then $(q, w \cdot a, q'') \in \hat{\delta}$.

Finite-state automata

Example: Paths
For the finite-state automaton:



$\hat{\delta}$ is the following set of triples:

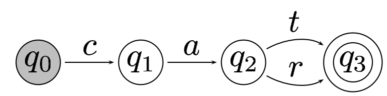
$$\langle q_0, \epsilon, q_0\rangle, \langle q_1, \epsilon, q_1\rangle, \langle q_2, \epsilon, q_2\rangle, \langle q_3, \epsilon, q_3\rangle,$$
$$\langle q_0, c, q_1\rangle, \langle q_1, a, q_2\rangle, \langle q_2, t, q_3\rangle, \langle q_2, r, q_3\rangle,$$
$$\langle q_0, ca, q_2\rangle, \langle q_1, at, q_3\rangle, \langle q_1, ar, q_3\rangle,$$
$$\langle q_0, cat, q_3\rangle, \langle q_0, car, q_3\rangle$$

Finite-state automata

A string w is accepted by the automaton $A = \langle Q, q_0, \Sigma, \delta, F\rangle$ if and only if there exists a state $q_f \in F$ such that $\hat{\delta}(q_0, w) = q_f$.

The *language accepted by a finite-state automaton* is the set of all string it accepts.

Example: Language
The language of the finite-state automaton:



is $\{$*cat, car*$\}$.

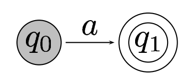
Finite-state automata

Example: Some finite-state automata

$\emptyset$

Finite-state automata

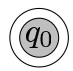
Example: Some finite-state automata



$\{a\}$

Finite-state automata

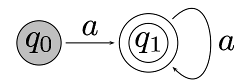
Example: Some finite-state automata



$\{\epsilon\}$

Finite-state automata

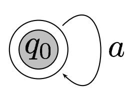
Example: Some finite-state automata



$\{a, aa, aaa, aaaa, \ldots\}$

Finite-state automata

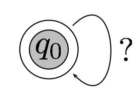
Example: Some finite-state automata



a^*

Finite-state automata

Example: Some finite-state automata



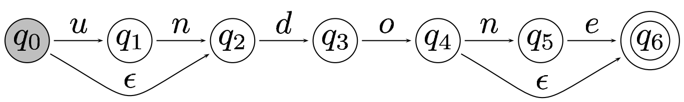
Σ^*

Finite-state automata

An extension: ϵ-moves.

The transition relation δ is extended to: $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$

Example: Automata with ϵ-moves
The language accepted by the following automaton is $\{$*do, undo, done, undone*$\}$:



Finite-state automata

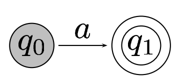
Theorem (Kleene, 1956): The class of languages recognized by finite-state automata is the class of regular languages.

Finite-state automata

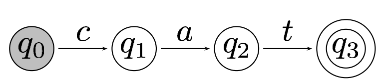
Example: Finite-state automata and regular expressions

$\emptyset$

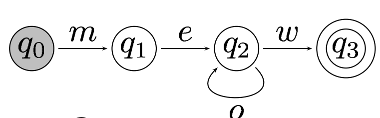
a



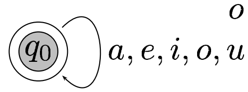
$((c \cdot a) \cdot t)$



$(((m \cdot e) \cdot (o)^*) \cdot w)$



$((a + (e + (i + (o + u)))))^*$



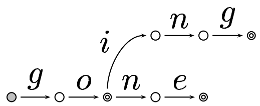
Operations on finite-state automata

- Concatenation
- Union
- Intersection
- Minimization
- Determinization

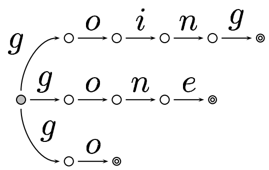
Minimization and determinization

Example: Equivalent automata

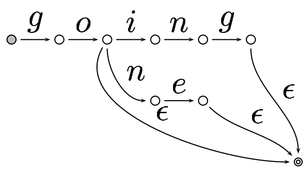
A_1



A_2



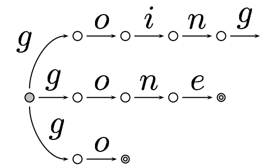
A_3



Applications of finite-state automata in language processing

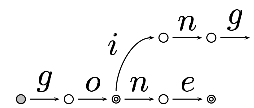
Lexicon:

go, gone, going :



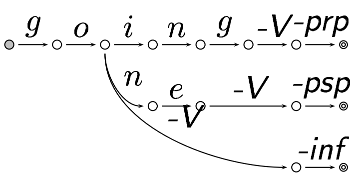
This automaton can then be determinized and minimized:

go, gone, going :



Applications of finite-state automata in language processing

A naïve morphological analyzer:



Regular relations

While regular expressions are sufficiently expressive for many natural language applications, it is sometimes useful to define relations over two sets of strings.

Regular relations

Part-of-speech tagging:

I	know	some	new	tricks	
PRON	V	DET	ADJ	N	
said	the	Cat	in	the	Hat
V	DET	N	P	DET	N

Regular relations

Morphological analysis:

I	know	some	new
I-PRON-1-sg	know-V-pres	some-DET-indef	new-ADJ
tricks	said	the	Cat
trick-N-pl	say-V-past	the-DET-def	cat-N-sg
in	the	Hat	
in-P	the-DET-def	hat-N-sg	

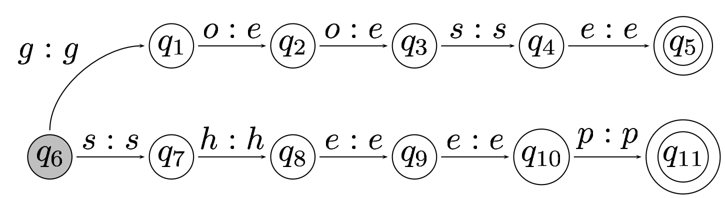
Regular relations

Singular-to-plural mapping:

cat	hat	ox	child	mouse	sheep	goose
cats	hats	oxen	children	mice	sheep	geese

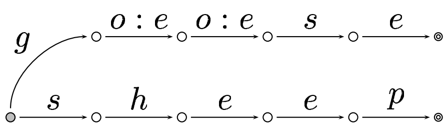
Finite-state transducers

A finite-state transducer is a six-tuple $\langle Q, q_0, \Sigma_1, \Sigma_2, \delta, F\rangle$. Similarly to automata, Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final (or accepting) states, Σ_1 and Σ_2 are alphabets: finite sets of symbols, not necessarily disjoint (or different).

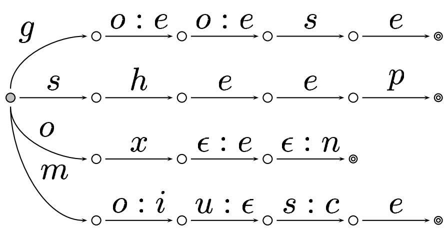


Finite-state transducers

Shorthand notation:



Adding ϵ-moves:

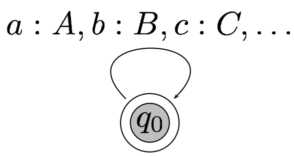


Finite-state transducers

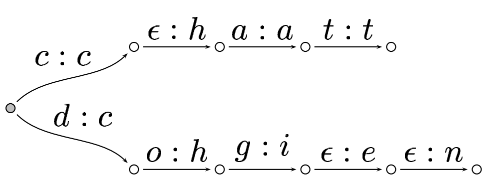
The language of a finite-state transducer is a language of pairs: a binary relation over $\Sigma_1^* \times \Sigma_2^*$. The language is defined analogously to how the language of an automaton is defined.

Finite-state transducers

Example: The uppercase transducer



Example: English-to-French



Properties of finite-state transducers

Given a transducer $\langle Q, q_0, \Sigma_1, \Sigma_2, \delta, F \rangle$,

- its underlying automaton is $\langle Q, q_0, \Sigma_1 \times \Sigma_2, \delta', F \rangle$, where $(q_1, (a, b), q_2) \in \delta'$ iff $(q_1, a, b, q_2) \in \delta$
- its upper automaton is $\langle Q, q_0, \Sigma_1, \delta_1, F \rangle$, where $(q_1, a, q_2) \in \delta_1$ iff for some $b \in \Sigma_2$, $(q_1, a, b, q_2) \in \delta$
- its lower automaton is $\langle Q, q_0, \Sigma_2, \delta_2, F \rangle$, where $(q_1, b, q_2) \in \delta_2$ iff for some $a \in \Sigma_a$, $(q_1, a, b, q_2) \in \delta$

Properties of finite-state transducers

A transducer T is *functional* if for every $w \in \Sigma_1^*$, $T(w)$ 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 $w \in \Sigma_1^*$, $T(w) = T_1(w) \cup T_2(w)$.

Transducers are closed under inversion: if T is a transducer, there exists a transducer T^{-1} such that for every $w \in \Sigma_1^*$, $T^{-1}(w) = \{u \in \Sigma_2^* \mid w \in T(u)\}$.

The inverse transducer is $\langle Q, q_0, \Sigma_2, \Sigma_1, \delta^{-1}, F\rangle$, where $(q_1, a, b, q_2) \in \delta^{-1}$ iff $(q_1, b, a, q_2) \in \delta$.

Properties of finite-state transducers

Transducers are closed under composition: if T_1 and T_2 are transducers, there exists a transducer T such that for every $w \in \Sigma_1^*$, $T(w) = T_1(T_2(w))$.

The number of states in the composition transducer might be $|Q_1 \times Q_2|$.

Properties of finite-state transducers

Transducers are not closed under intersection.

T_1 T_2

$$T_1(c^n) = \{a^n b^m \mid m \geq 0\}$$
$$T_2(c^n) = \{a^m b^n \mid m \geq 0\} \Rightarrow$$
$$T_1 \cap T_2(c^n) = \{a^n b^n\}$$

Transducers with no ϵ-moves are closed under intersection.

Properties of finite-state transducers

- Computationally efficient
- Denote regular relations
- Closed under concatenation, Kleene-star, union
- Not closed under intersection (and hence complementation)
- Closed under composition
- Weights