Introduction to Computational Linguistics

PD Dr. Frank Richter

fr@sfs.uni-tuebingen.de.

Seminar für Sprachwissenschaft

Eberhard-Karls-Universität Tübingen

Germany

Example of String Acceptance

Let M = $(\{a, b\}, \{q_0, q_1, q_2\}, q_0, \{q_1\}, \{((q_0, a), q_1), ((q_0, b), q_1), ((q_1, a), q_2), ((q_1, b), q_2), ((q_2, a), q_2), ((q_2, b), q_2), \})$.

Example of String Acceptance

Let M = $(\{a, b\}, \{q_0, q_1, q_2\}, q_0, \{q_1\}, \{((q_0, a), q_1), ((q_0, b), q_1), ((q_1, a), q_2), ((q_1, b), q_2), ((q_2, a), q_2), ((q_2, b), q_2), \})$.

M accepts a and b and nothing else, i.e. $L(M) = \{a, b\}$, since

$(0, q_0, a) \vdash (a, q_1, 0)$ and
$(0, q_0, b) \vdash (b, q_1, 0)$

are the only derivations from a start state to a final state for M.

More Properties of FSAs

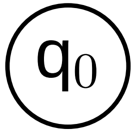
Given the FSAs A, A_1, and A_2 and the string w, the following properties are decidable:

Membership:	$w \stackrel{?}{\in} L(A)$
Emptiness:	$L(A) \stackrel{?}{=} \varnothing$
Totality:	$L(A) \stackrel{?}{=} \Sigma^*$
Subset:	$L(A_1) \stackrel{?}{\subseteq} L(A_2)$
Equality:	$L(A_1) \stackrel{?}{=} L(A_2)$

Regular Expressions and Automata (1)

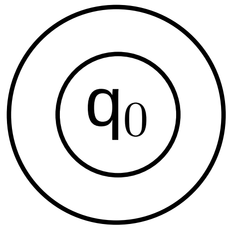
Regular Expression: ∅

Automaton:



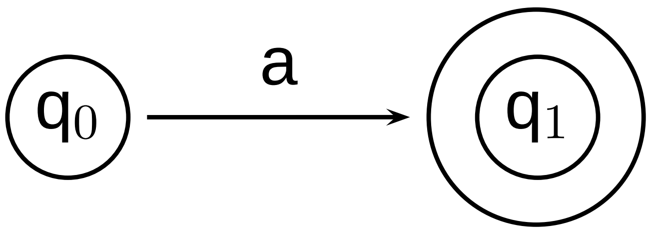
Regular Expression: O

Automaton:



Regular Expression: a

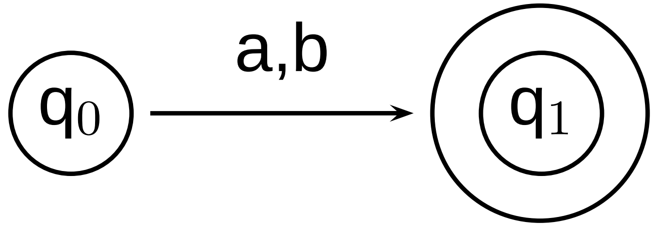
Automaton:



Regular Expressions and Automata (2)

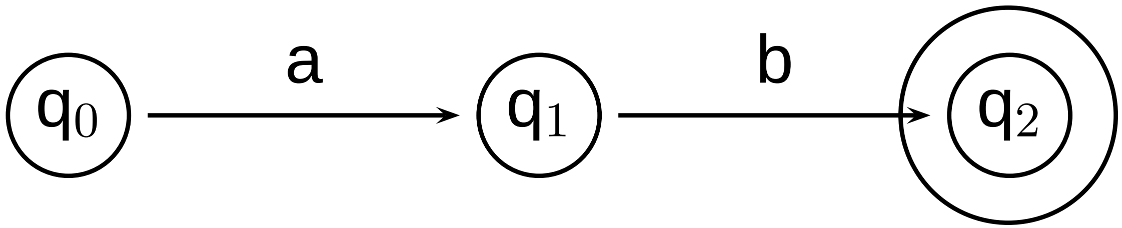
Regular Expression: [a | b]

Automaton:



Regular Expression: [a b]

Automaton:



The Finite State Utilities

The FSA Utilities toolbox:

- a collection of utilities to manipulate regular expressions, finite-state automata (and finite-state transducers).
- inplemented in Prolog by Gertjan van Noord, University of Groningen
- Home Page: `http://odur.let.rug.nl/~vannoord/Fsa/`
- command in the SfS network (penthesilea): `fsa -tk`

Reg. Expressions: Syntactic Extensions

$A	*contains*
	$A $=_{def}$ [?* A ?*]
	for example: $[a | b] denotes all strings
	that contain at least one a or b somewhere.
A & B	Intersection
A - B	Relative complement (minus)
$\sim$ A	Complement (negation)

The Bigger Picture

Definition 9 (Regular Languages)

A language L is said to be *regular or recognizable* if the set of strings s such that $s \in L$ are accepted by a DFA.

Theorem (Kleene, 1956)

The family of regular languages over Σ^* is equal to the smallest family of languages over Σ^* that contains the empty set, the singleton sets, and that is closed under Kleene star, concatenation, and union.

$\Rightarrow$ The family of regular languages over Σ^* is equal to the family of languages denoted by the set of regular expressions.

Regular Relations

- Regular expressions can contain two kinds of symbols: unary symbols and symbol pairs.
  - Unary symbols (a, b, etc) denote strings.
  - Symbol pairs (a:b, a:0, 0:b, etc.) denote pairs of strings.
- The simplest kind of regular expression contains a single symbol. E.g., “a” denotes the set $\{$a$\}$.
- Similarly, the regular expression “a:b” denotes the singleton relation $\{\langle \mathrm{a}, \mathrm{b} \rangle\}$.
- A regular relation can be viewed as a mapping between two regular languages. The a:b relation is simply the crossproduct of the languages denoted by the expressions a and b.

Finite-State Transducer

Definition 10 (FST) A finite-state transducer is a 6-tuple $(\Sigma_1, \Sigma_2, Q, i, F, E)$ where

Σ_1 is a finite alphabet,
(called the *input alphabet*)

Σ_2 is a finite alphabet,
(called the *output alphabet*)

Q is a finite set of *states*,

$i \in Q$ is the *initial state*,

$F \subseteq Q$ the set of *final states*, and

$E \subseteq Q \times (\Sigma_1{}^* \times \Sigma_2{}^*) \times Q$
is the set of edges.

Constructing Regular Relations

- Crossproduct: A .x. B
  - The crossproduct operator, .x., is used only with expressions that denote a regular language; it constructs a relation between them.
  - [A .x. B] designates the relation that maps every string of A to every string of B. If A contains x and B contains y, the pair $\langle x, y \rangle$ is included in the crossproduct.

Constructing Regular Relations

- Composition: A .o. B
  - Composition is an operation on relations that yields a new relation. [A .o. B] maps strings that are in the upper language of A to strings that are in the lower language of B.
  - If A contains the pair $\langle x, y\rangle$ and B contains the pair $\langle y, z\rangle$, the pair $\langle x, z\rangle$ is in the composite relation.

Properties of Regular Relations

Regular relations in general are *not closed* under

- complementation,
- intersection, and
- subtraction.

Properties of Transducers

- A transducer is functional iff for any input there is at most one output.
- A transducer is sequential iff no state has more than one arc with the same symbol on the input side.

On-Line Literature, Demos and Tools

Finte State Technology, Lauri Karttunen's Web Page at Xerox

Lauri Karttunen et. al.:

Regular Expressions for Language Engineering

Link: `http://www.xrce.xerox.com/competencies/content-analysis/fst/home.en.html`

Replace Operators

- Unconditional obligatory replacement:

  $A \rightarrow B =_{def}$ [[$\sim$$[A - [ ]] [A .x. B]]* $\sim$$[A - [ ]]]

- Unconditional optional replacement:

  A ($\rightarrow$) B $=_{def}$ [[$\sim$$[A - [ ]] [A .x. A | A .x. B]]*
  $\sim$$[A - [ ]]]

- Contextual obligatory replacement:

  $A \rightarrow B \parallel L \_ R$

  meaning: “Replace A by B in the context L __ R.”