Finite-state Automata

Basic Mathematics

Martin Kay and Ron Kaplan

Regular Languages

$\Sigma^\epsilon = \Sigma \cup \epsilon$

- **The empty set and {a} for all a in Σ^ε are regular languages**
- **If L_1, L_2, and L are regular languages, then so are**

$L_1 L_2 = \{xy | x \in L_1 \text{ and } y \in L_2\}$ **(concatenation)**

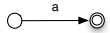
$L_1 \cup L_2 = \{x | x \in L_1 \text{ or } x \in L_2\}$ **(union)**

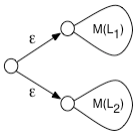
$L^* = \cup_{i=0}^{\infty} L^i$ **(Kleene closure)**

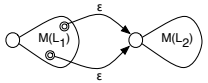
- **There are no other regular languages**

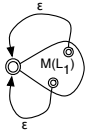
Correspondence Theorem (Kleene)

Every regular language L is accepted by some FSM M(L)

$\{a\}$ 

$L_1 \cup L_2$ 

$L_1 L_2$ 

$\cup_{i=0}^{\infty} L^i$ 

Types of FSMs

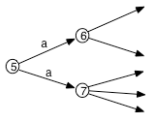
- **+ - Deterministic**
- **+ - ε-free**
- **+ - Minimal**
- **+ - Complete**

... can all characterize any regular language

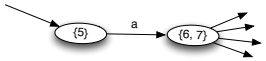
Determinizing

- **Nondeterministic**

$\delta: Q \times \Sigma \rightarrow 2^Q$



- **Deterministic**

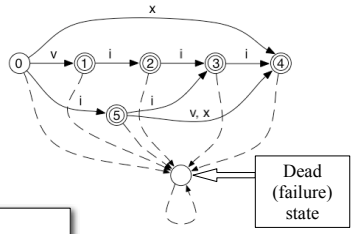


$\delta: Q' \times \Sigma \rightarrow Q'$
$Q'=2^Q$

A State is final if any member of the set is final

Search time is linear for a deterministic, but exponential for a nondeterministic FSM

Complete vs. Pruned

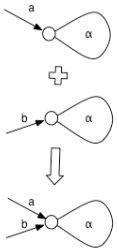


Pruned:
- Smaller
- δ is a partial function.
- No dead states
- Lookup is faster

Follow a dashed line iff there is no solid one for the current character

Minimization

Minimal machine is unique (up to renaming of states and arc ordering)



Minimized: No two states have congruent suffix graphs

Proof Strategy for Language Properties

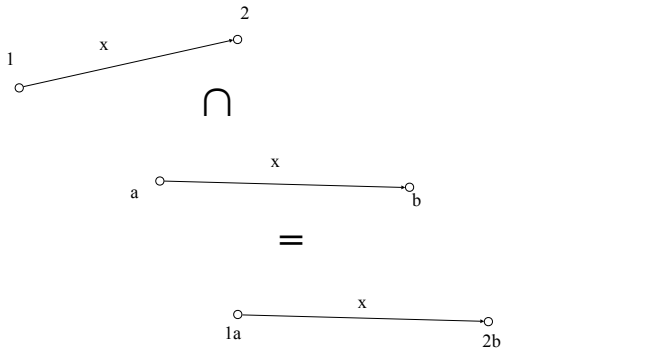
To prove $\phi(L_1, \ldots L_n)$

- **Get machines $M(L_1), \ldots M(L_n)$**
- **Transform $M(L_1), \ldots M(L_n) \Rightarrow M$**
- **Show $L(M) = \phi(L_1, \ldots L_n)$**

L(M) is the language accepted by M

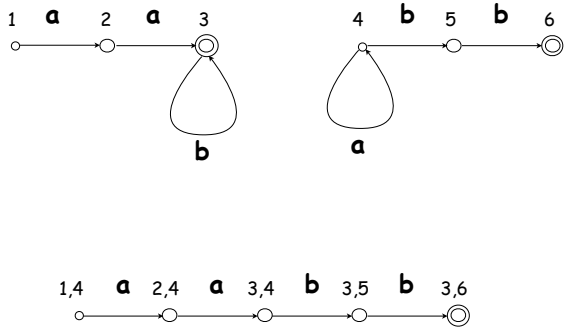
Compute with machine — They are finite!

L1 ∩ L2 is Regular



2b is final if 2 and b are final

Intersection



$\overline{L}$ is Regular

- **Get deterministic, complete M(L) = <Σ, Q, q, F, δ >**
- **M = < Σ , Q, q, F', δ> where F'=Q-F**
- **L(M)=$\overline{L}$?**

Suppose x not in L
x takes M(L) into r∈Q-F.
∴ x takes M into r ∈ F'
∴ x ∈ L(M).

If M(L) were not deterministic, a string x could take M(L) into r ∈ F and s∉F. It would therefore take L(M) into r ∉ F' and s ∈ F'

Suppose x in L
x takes M(L) into r ∈ Q.
∴ x takes M into r ∈ Q-F'
∴ x ∉ L(M).

Properties of Regular Languages

- **Closure**
  - **Intersection**
  - **Union**
  - **Complementation**
  - **Concatenation**
  - **Iteration**
  - **Reversal**
- **Decidable Predicates**
  - **Emptiness**
  - **Equality**
  - **Finiteness**

String Relations

n-way concatenation

$X = \langle x_1, x_2 \ldots x_n \rangle$

$Y = \langle y_1, y_2 \ldots y_n \rangle$

$\Big\} XY = \langle x_1 y_1, x_2 y_2 \ldots x_n y_n \rangle$

The n-way concatenation of two string-tuples is the tuple of strings formed by string concatenation of the corresponding elements.

One can construct families of string relations that parallel the usual classes of formal languages

A Context-free Relation

S → <s, ε> <(, ε > NP VP <), ε >

NP → <np, ε> <(, ε > DET N <), ε >

VP → <vp, ε > <(, ε > V NP <), ε >

DET → <det, the>

N → <n, dog>

N → <n, cat>

V → <v, chased>

```
s ( np ( det  n ) vp (   v    np ( det  n  )))
        the dog       chased      the cat
```


Regular Relations

- The empty set and $\{a\}$ for all a in $\Sigma^\varepsilon \times \Sigma^\varepsilon \ldots$ are regular relations
- If R_1, R_2, and R are regular n-relations, then so are

  $R_1R_2 = \{xy \mid x \in R_1 \text{ and } y \in R_2\}$ (concatenation)

  $R_1 \cup R_2$ (union)

  $R^* = \cup R^i$ (n-way Kleene closure)

- There are no other regular relations

All Correspondences

- Every n-way regular expression describes a regular n-relation.
- Every regular n-relation is described by an n-way regular expression.
- Every n-tape finite-state transducer accepts a regular n-relation.
- Every regular n-relation is accepted by an n-tape finite-state transducer.

Regular—

- If L_1, L_2, and L are regular languages and R_1, R_2, and R are regular relations, the following relations are regular:

  $R_1 \cup R_2$

  $R_1 R_2$

  R^*

  R^{-1}

  $R_1 \circ R_2$

  Id(L)

  $L_1 \times L_2$

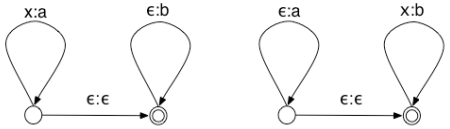
  Rev(R)

... and the following languages
Dom(R)
Range(R)
L/R
R/L
x/R
R/x

Not Regular

- $R_1 \cap R_2$
- $\overline{R}$

Intersection is not Regular



$\overline{R}$ is not regular because ...

if it were, you could use it, together with union to construct intersection!

$$A \cap B = \overline{\overline{A} \cup B}$$

n-way Automata—Transducers

An n-way automaton is defined by a quintuple similar to the ones that define ordinary finite-state machines

$$(\Sigma, Q, q, F, \delta)$$

where Σ is a finite alphabet

Q is a finite set of states

$q \in Q$ is the initial state

$F \subset Q$ is the set of final states

δ maps $Q \times \Sigma^\epsilon \times ...\Sigma^\epsilon$ to 2^Q

From now on, we limit our discussion to binary relations and transducers

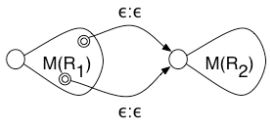
Union and Iteration

$R_1 \cup R_2$

$R^* = \bigcup_{i=0}^{\infty} R^i$

Concatenation

$R = R_1 R_2$



Range and Domain

Dom(R) = R/Σ*

Range(R) = Σ*/R

Accepting FSMs derived from T(R) by replacing all transition labels a:b by

a (domain) or

b (range)

⇒ Regular languages.

R^{-1}

$$R^{-1} = \{< y, x > | < x, y > \in R\}$$

$$\delta_{r^{-1}}(p, a, b) = \delta_{r^{-1}}(p, b, a)$$

Id(L)

$$Id(l) = \{< x, x > | x \in L\})$$

$$\delta_{Id}(r, a, a) = \delta_L(r, a)$$

$$\delta_{Id}(r, a, b) = \quad \text{FAIL!} \quad a \neq b$$

Extending δ

To state sets

Parallel extensions for transducers

For all $P \subseteq Q$ and $a \in \Sigma^{\varepsilon}$

$$\delta(P,a) = \bigcup_{p \in P} \delta(p,a)$$

To strings

For all $r \in Q$, $\delta^*(r,\varepsilon) = \delta(r,\varepsilon)$

For all $u \in \Sigma^*$ and $a \in \Sigma^{\varepsilon}$

$$\delta^*(r,ua) = \delta(\delta^*(r,u),a)$$

$$\delta^*(q,x) \cap F$$

The machine accepts a string x just in case F is not empty.

Cartesian Product

N.B. $L \times L \neq ID(L)$

Let $L_1 \times L_2 = \{< x,y > | x \in L_1 \ \& \ y \in L_2\}$

$$T = (\Sigma, Q_1 \times Q_2, < q_1, q_2 >, F_1 \times F_2, \delta)$$

where $\delta(< s_1, s_2 >, a, b) = \delta(s_1, a) \times \delta(s_2, b)$

Claim: T accepts $L_1 \times L_2$

T accepts $L_1 \times L_2$

Proof:

by induction.

$$\delta^*(< q_1, q_2 >, x, y) = \delta_1^*(q_1, x) \times \delta_2^*(q_2, y)$$

Thus, T enters a final state on <x,y> iff M(L_1) enters a final state on x and M(L_2) enters a final state on y.

$R_1 \circ R_2$

$$R_1 \circ R_2 = \{< x, y > \mid \text{for some } z, < x, z > \in R_1 \,\&\, < z, y > \in R_2\}$$

$$T = (\Sigma, Q_1 \times Q_2, < q_1, q_2 >, F_1 \times F_2, \delta)$$

$$\delta(< s_1, s_2 >, a, b) = \{< t1, t2 > \mid \text{for some } c \in \Sigma^\epsilon,\ t_1 \in \delta(s_1, a, c) \text{ and } t_2 \in \delta(s_2, c, b)$$

Images

xRy ⇔ <x,y> ∈ R

x/R = {y | <x,y> ∈ R}

R/y = {x | <x,y> ∈ R}

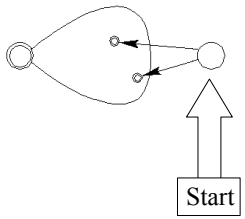
$$X / R = \bigcup_{x \in X} x / R$$

R/intractable = {intractable, iNtractable}
iNtractable/R = {intractable}

Images are Regular

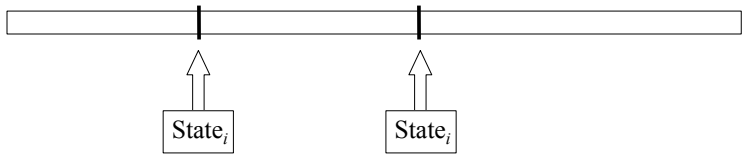
$$L / R = \text{Range}(\text{Id}(L))) / R$$
$$= \text{Range}(\text{Id}(L) \circ R)$$

Rev(R)



Pumping Lemma

It is possible to delete a part of any sufficiently long substring of a regular language and leave a string the is a member of the language



Text