AUTOMATA
AND
GRAMMARS

(Linguistic machinery)
Whenever I learn a new skill, I concoct elaborate fantasy scenarios where it lets me save the day.

Oh no! The killer must have followed her on vacation!

But to find them, we'd have to search through 200 MB of emails looking for something formatted like an address!

It's hopeless!

Everybody stand back.

I know regular expressions.
Finite-state automaton

• Finite set of states (initial, final)
• Finite set of symbols (labels, alphabet)
• Finite set of transitions (state x symbol)

• Fairly easily implemented
• Representable as network, table
Sample FSA

- a
- big
- dog
- yawned
Acceptors

- aka recognizers, sequence detectors
- Given: fsa, input string
- Traverse automaton
- Report success/failure
- Limitations:
  - Recognition only
FSA’s can encode regexes (& vice-versa)

- \([c^*c | [b[a|b|c]^c]]\)
Other properties of regular languages

- Closed under union
- Closed under concatenation
- Closed under Kleene star
- Closed under complementation
- Closed under intersection
- Useful next lecture
Swahili morphology FSA

Subj: ni, u, a, tu, wa
Tense: ta, na, me, li
Object: ni, ku, m, tu, wa
Root: penda, piga, sumbua, lipa
Determinism

• Deterministic: no choice at any transition(s)
• Nondeterministic: choice at some transition(s)
  • >1 arc from given state for given symbol
  • Empty arc: labelled $\lambda$ or $\varepsilon$
• Any NFA: can construct equivalent DFA
• Regular languages: representable
Sample NDFA
Equivalence

- Regular expressions $\rightarrow$ NFA’s with $\lambda$-transitions $\rightarrow$ NFA’s $\rightarrow$ DFA’s
- Same class of languages (called regular sets)
- Transition tables
Dealing with non-determinism

- Backup
  - Save search-state, search
  - Stacks, queues
- Look-ahead
- Parallelism
Transducers

- Given: fsa, input string
- Traverse automaton
- Report (mapped) output string
- Applications:
  - Sublanguage translation
  - TTS transduction for some languages
  - Morphological processing
- Limitation: non-recursive
Transducer

WH+: where| où
BV: is| est
DetF: the| la
DetM: the| le
NF: exit| sortie, shop| boutique, toilet| toilette
NM: bar | bar, policeman | gendarme
Language is recursive

- John said that Mary is a genius.
- Fred thinks that …
- I doubt Fred thinks that …

- The man fell.
- The woman who saw the man fell.
- The man who the woman I met saw fell.
Recursive transition networks

- Not just categories, but name of another network on arc
- 4 types of arcs: category, empty, seek, exit
- Uses a pushdown stack
- Pros/cons:
  - Modularity, more powerful than FSA’s
  - Linearity constraint, computationally costly
Sample RTN

N: dog, cat, boy, girl, ...
Det: a, the, my, your, seven, every, ...
Adj: happy, tall, big, uninteresting, ...
V: sneezed, saw, died, barks, ...
Compz: that, because, whether, ...
Pushdown transducers

- RTN + paired transitions
- Output string via traversal
Problems

• No memory: straight input, output
• Language has gaps:
  • What movie did you ___ see ___ ?
  • Where did you go ___?
  • I saw the dog that ___ barked.
  • Who will Bill date ___?
  • What is Fred so very upset about ___?
ATN’s

- Add register(s)
- Add procedures to arcs
- Procedural in nature
- Pros/cons:
  - Relaxes linearity constraint
  - Nonlocal dependencies (gapping, particles)
  - Contradicts prevalence of declarative formalisms
Building automata

- Specialized languages (lex, yacc)
- Generalized toolkits, utilities
  - [http://odur.let.rug.nl/~vannoord/Fsa/](http://odur.let.rug.nl/~vannoord/Fsa/)
  - [GraphViz](http://graphviz.org/)
  - [Foma](http://foma.org/) (follow the forward link to BitBucket for the code)
- Generalized applications (PC-Kimmo)
- Specific toolkits
Checking dates

- Years 1-9999: 3.7 million days
- Leap years, calendar changes
- Accept valid ones, reject invalid ones
- Xerox: 1346 states, 21,006 arcs

\begin{verbatim}
A | B  Union.
A B  Concatenation.
(A)  Optionality; union with the empty string.
A+  Iteration; one or more concatenations of A.
A*  Kleene star; equivalent to (A+).

A .x. B  Crossproduct
A .o. B  Composition

1To9 = [ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 ]
0To9 = [ %0 | 1To9 ]
SP   = [ "", " ]
Day  = [ Monday | Tuesday | ...... | Saturday | Sunday ]
Month = [ January | February | ...... | November | December ]
Date  = [ 1To9 | [ 1 | 2 ] 0To9 | 3 [%0 | 1 ] ]
Year  = 1To9 (0To9 (0To9 (0To9)))
DateExpression = Day | (Day SP) Month " " Date (SP Year)

Even    = [ %0 | 2 | 4 | 6 | 8 ]
Odd     = [ 1 | 3 | 5 | 7 | 9 ]
N       = 1To9 0To9*
Div4    = [[(N) Even] [%0 | 4 | 8]] | [[(N) Odd [2 | 6]]
LeapYear = Div4 - [[N - Div4] %0 %0]

MaxDaysInMonth =
   "$[ February " " 3 %0 |
     [February | April | June | September | November] " " 31 ]
\end{verbatim}
Applications

• Phonology: speech applications
• Morphology: word formation
• Syntax: phrase structure
• Discourse: dialogue structure
• Text mining: word spotting, threat detection
• Text tools: grammar checking, correction
ATT’s FSM toolkit

digraph finite_state_machine {
    rankdir=LR;
    size="8,5"
    orientation=land;
    node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;
    node [shape = circle];
    LR_0 -> LR_2 [ label = "SS(B)" ];
    LR_0 -> LR_1 [ label = "SS(S)" ];
    LR_1 -> LR_3 [ label = "S($end)" ];
    LR_2 -> LR_6 [ label = "SS(b)" ];
    LR_2 -> LR_5 [ label = "SS(a)" ];
    LR_2 -> LR_4 [ label = "S(A)" ];
    LR_5 -> LR_7 [ label = "S(b)" ];
    LR_5 -> LR_5 [ label = "S(a)" ];
    LR_6 -> LR_6 [ label = "S(b)" ];
    LR_6 -> LR_5 [ label = "S(a)" ];
    LR_7 -> LR_8 [ label = "S(b)" ];
    LR_7 -> LR_5 [ label = "S(a)" ];
    LR_8 -> LR_6 [ label = "S(b)" ];
    LR_8 -> LR_5 [ label = "S(a)" ];
}
Graphing via FSM
What is a grammar?

• A type of knowledge representation
• Description of combinations of constituents
• Provides no procedural information (how)
• Provides implicit structural description (strings => structure)
• System of rules and categories that underlies some level of language
Formal language theory

- Alphabets
- String operations: concatenation, reversal
- Set operations: intersection, difference, union, complementation
- Closure