CSE431 – Translation of Computer Languages

- Deterministic Finite Automata
- Regular Expressions

Doug Shook

A Puzzle







Solution

- First: what is the actual solution (in English)?
- How can we represent this solution?

Solution

- Let's take a look at a finite state machine for this problem
  - Also called deterministic finite automata

Solution

- What are the components of this DFA?
  - There are five total

Alphabets, Strings, Languages

- An alphabet is a finite set of symbols
- A string is a finite sequence of symbols from some alphabet
- A language is a possibly infinite set of strings

Alphabets, Strings, Languages

- The empty string is represented by lambda
- The language Σ* is the set of all possible strings, including the empty string

Regular Languages

- A regular language is a language recognized by a DFA
- Given a DFA d and a string s, how can we determine if s is in the language defined by d?

Exercises

- Create a DFA for each of the following:
  - Alphabet = {0, 1}

    Every possible binary string (i.e. Σ*)

    All binary strings that include a 1

    All binary strings that contain two consecutive 1's

    All binary strings that end with a 1

    All binary strings that contain exactly two 0's

    All binary strings that have an even number of 0's and an even number of 1's

Regular Expressions

- DFA are great for visualizing and processing a language
  - Drawbacks?
- Enter regular expressions:
  - More concise format
- Operators available with regular expressions:
  - Kleene closure (*), alternation, concatenation

Regular Expressions

- Kleene closure (*): we've already seen this
- Alternation
  - Similar to OR
    - A | B = {A, B}
- Concatenation
  - Similar to AND
    - AB = {AB}
- In practice, more operators are added for convenience

Regular expressions

- What is represented by the following regular expressions (all use alphabet = {a, b, c})
  - Write a few example strings
  - Describe the language in english

a | bc*

(a | b) c*

(ab)*| b

Regular Expressions

- What are the regular expressions for the following (alphabet = {0, 1})?

  All binary strings that include a 1

  All binary strings that contain two consecutive 1's

  All binary strings that end with a 1

  All binary strings that contain exactly two 0's

Limitations

- Matching given strings to regex is difficult:

  (a(bc)*(cd | (de | ef))*)*

- Which of the following match?

  aaaaaaaaa abc

  adeefdeabc abcefcdcdade

- What is an easier tool for this problem?

Grammars

- Like DFA, regular expressions work with regular languages
  - Next class we will show the equivalency
- Grammars can also be used to represent regular languages
  - Set of terminals (Σ)
  - Set of nonterminals (V)
  - Start nonterminal (S)
  - Set of productions (P)

Right Linear Grammars

- The type of grammar that recognizes a regular language is known as a right linear grammar
  - A -> α, where α contains at most one nonterminal
  - Non-terminal must be on the right

- Example:

  S -> aB
  | cD
  | B
  B -> bD
  | b
  D -> d
  | λ

Right Linear Grammars

- What languages are defined by the following RLGs?
  - Alphabet = {a, b}

```
S -> aS
   |  bS
   |  λ
------------
S -> aT
   |  bT
T -> aS
   |  bS
   |  λ
```

Right Linear Grammars

- What languages are defined by the following RLGs?
  - Alphabet = {a, b}

```
S -> aT
   |  bT
   |  λ
T -> aS
   |  bS
-----------------------
S -> aA
A -> aA
   |  bB
B -> bB
   |  b
```

Right Linear Grammars

- Construct a RLG for the following:
  - Alphabet = {a, b}

    All binary strings that include a ‘b’

    All binary strings that contain two consecutive b’s

    All binary strings that end with a b

    All binary strings that contain exactly two a’s

- Construct a right linear grammar for the tiger, goat, cabbage problem
  - Alphabet = {s, t, g, c}

Equivalencies

- DFAs, RLGs, and Regular expressions are all capable of specifying regular languages

- It is possible to go from one to the other
  - Topic for next class