Regular Expressions

CSE 311 Autumn 2023
Lecture 17

Announcements

Everyone will get credit for yesterday's section

HW 4 part 1 grades will be out today

HW 5 Is due tonight and HW 6 releases today

Quick Set Theory Proof:

Prove that if $A = B$, then $P(A) = P(B)$.

Suppose A = B.

Quick Set Theory Proof:

Prove that if $A = B$, then $P(A) = P(B)$.

Suppose A = B.

Let $X \in P(A)$ be arbitrary. This means that $X \subseteq A$ by definition of powerset. Since $A = B$, this means that $X \subseteq B$ or $X \in P(B)$. Therefore, $P(A) \subseteq P(B)$

Let $X \in P(B)$ be arbitrary. This means that $X \subseteq \mathrm{B}$ by definition of powerset. Since $A = B$, this means that $X \subseteq A$ or $X \in P(A)$. Therefore, $P(B) \subseteq P(A)$.

Since $P(A) \subseteq P(B)$ and $P(B) \subseteq P(A)$, we have $P(A) = P(B)$

Therefore, if $A = B$, then $P(A) = P(B)$.

Theoretical Computer Science

Recall: Course Goals

1. Learn to make & clearly communicate rigorous formal arguments
   - Mathematical Proofs
2. Understand mathematical objects that are widely used in CS
   - Number Theory, Set Theory, Recursively-Defined Functions
3. Explore and analyze models of computation
   - Regular Expressions, Context-Free Grammars, Finite Automata

Languages

Definition:
A language is a set of strings.

For Example:

- "The set of all valid English sentences"
- "The set of all binary strings of even length"
- "The set of all syntactically correct Java programs"

Languages in Theoretical Computer Science

- We want to study different models of computation, and the strengths & limitations of each.
- A computer is said to recognize a language if it can distinguish which strings are in a language vs. which are not.
- One way to evaluate how powerful a model of computation is is to determine which languages it can recognize.

Regular Expressions In Practice

EXTREMELY useful. Used to define valid “tokens” (like legal variable names or all known keywords when writing compilers/languages)

Used in `grep` to actually search through documents.

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

- **^** start of string
- **$** end of string
- **[01]** a 0 or a 1
- **[0-9]** any single digit
- **\\.** period **\\,** comma **\\-** minus
- . any single character
- ab a followed by b (**AB**)
- (a|b) a or b (**A** ∪ **B**)
  a**?** zero or one of a (**A** ∪ ε)
  a***** zero or more of a **A***
- a**+** one or more of a **AA***

e.g. **^[\-+]?[0-9]*(\.|\,)?[0-9]+$**

General form of decimal number e.g. 9.12 or -9,8 (Europe)

Regular Languages

One class of languages

Regular Expressions

Basis:

ε is a regular expression. The empty string itself matches the pattern (and nothing else does).

$\emptyset$ is a regular expression. No strings match this pattern.

a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive

If A, B are regular expressions then $(A \cup B)$ is a regular expression
matched by any string that matches A or that matches B [or both]).

If A, B are regular expressions then AB is a regular expression.
matched by any string x such that $x = yz$, y matches A and z matches B.

If A is a regular expression, then A^* is a regular expression.
matched by any string that can be divided into $\mathbf{0}$ or more strings that match A.

Regular Expressions

Each regular expression **matches** a set of strings (a language).

$A \cup B$ matches all strings that either A matches or B matches.

AB matches all strings $x = yz$ where A matches y and B matches z.

A^* matches all strings with any number of strings that A matches, i.e. $\varepsilon \cup A \cup AA \cup \cdots$

Regular Expressions

Each regular expression **matches** a set of strings (a language).

$A \cup B$ matches all strings that either A matches or B matches.

$(0 \cup 1)$ matches the strings in the set $\{0,1\}$

AB matches all strings $x = yz$ where A matches y and B matches z.

$0(0 \cup 1)1$ matches the strings in the set $\{001,011\}$

A^* matches all strings with any number of strings that A matches, i.e. $\varepsilon \cup A \cup AA \cup \cdots$

0^* matches the strings in the set $\{\varepsilon, 0, 00, 000, 0000 \ldots\}$

Examples

a^*b^*

$(0 \cup 1)0(0 \cup 1)0$

$(00 \cup 11)^*$

Examples

$a^* b^*$

Matches strings with any number of as followed by any number of bs.

$(0 \cup 1)0(0 \cup 1)0$

Matches strings in the set $\{0000, 1000, 0010, 1010\}$.

$(00 \cup 11)^*$

Matches all binary strings where 0s and 1s come in pairs

Examples

- *Construct a regular expression that matches the given set of strings.*

All binary strings.

- All binary strings that contain 0110.

Examples

- *Construct a regular expression that matches the given set of strings.*

All binary strings.

$(0 \cup 1)^*$

- All binary strings that contain $\mathbf{0110}$.

$(0 \cup 1)^* 0110 (0 \cup 1)^*$

Examples

Construct a regular expression that matches the given set of strings.

All binary strings that have an even number of 1s.

All binary strings that don't contain 00.

Examples

- *Construct a regular expression that matches the given set of strings.*

All binary strings that have an even number of 1s.

$(10^*1 \cup 0)^*$

- All binary strings that don't contain 00.

$(01 \cup 1)^*(0 \cup \varepsilon)$

Practical Advice

- Check ε and single character strings. Those are often edge cases.
- List 5 strings that should be matched, and 5 strings that shouldn't be.

  Test your RegEx against those strings.
- Remember $*$ allows for 0 copies! To say "at least one copy", use AA^*.

Exercises

- *Construct a regular expression that matches the given set of strings.*

The set of all binary strings of odd length.

- The set of all binary strings with at most two ones.

- The set of all binary strings with equal number of 0s and 1s.

Exercises

- *Construct a regular expression that matches the given set of strings.*

The set of all binary strings of odd length.

$(0 \cup 1)(00 \cup 01 \cup 10 \cup 11)^*$

- The set of all binary strings with at most two ones.

$0^*(1 \cup \varepsilon)0^*(1 \cup \varepsilon)0^*$

- The set of all binary strings with equal number of 0s and 1s.

Not possible! 😈

Note:

- Many implementations of RegExs are more powerful than our theoretical Regular Expression

Finite Languages vs Regular Expressions

- All Finite Languages have a regular expression

- Why?

- Could make this formal by induction

Finite Languages vs Regular Expressions

- Every Regular Expression generates a finite language if it does not use *

Why?

- You can prove this by structural induction on the syntax of a regular expression

Star-free implies finite

Let A be a regular expression that does not use *. Then L(A) is finite.

Proof: We proceed by *informal (don't do this on HW)* structural induction on A.

Case ε:

L(ε) = {ε}, which is finite

Case a:

L(a) = {a}, which is finite

Case A ∪ B:

L(A ∪ B) = L(A) ∪ L(B)

By the IH, each is finite, so their union is finite.

Star-free implies finite

Let A be a regular expression that does not use *. Then L(A) is finite.

Proof: We proceed by structural induction on A.

Case AB:

$$\mathrm{L(AB)} = \{x \ : \ \exists y \in L(A), \exists z \in L(B)\ (x = y \bullet z)\}$$

By the IH, $\mathrm{L(A)}$ and $\mathrm{L(B)}$ are finite.

Every element of $\mathrm{L(AB)}$ is covered by a pair (y, z) where $y \in L(A)$ and $z \in L(B)$, so $\mathrm{L(AB)}$ is finite.

(No case for A*!)


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.
PERL!
TAP TAP

Regular Languages

- Definitions:
- Regular Languages are languages that can be specified by a regular expression.
- Irregular Languages are languages that are not regular.

Irregular Languages

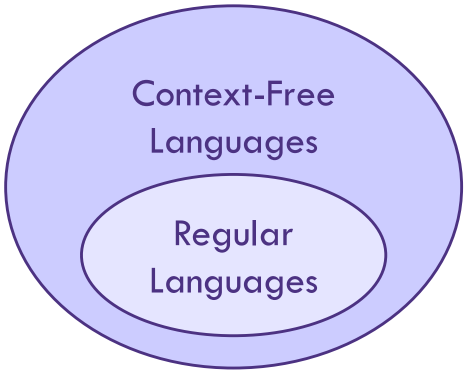
- It turns out a lot of useful languages are irregular.
- Binary strings with an equal number of 0s and 1s
- Palindromes (strings that read the same forwards and backwards)
- Matched parentheses, e.g. ((())())
- Properly formed arithmetic expressions

Context Free Languages

Another class of languages

Context-Free Languages

- We just saw some limitations of Regular Languages
- Context-Free Languages are a strictly larger class of languages



- Context-Free Languages are generated by Context-Free Grammars
  (just like Regular Languages are specified by Regular Expressions)

Context-Free Grammars (CFGs)

- A Context-Free Grammar is a finite set of production rules, involving:
  - Alphabet of *terminal* symbols (e.g. $0, 1, \mathrm{a}, \mathrm{b}, \varepsilon$)
  - A finite set of *nonterminal* symbols (e.g. $\mathrm{A}, \mathrm{B}, \mathrm{S}, \mathrm{T}, \mathrm{R}$)
  - One special nonterminal called the start symbol, usually S

- A production rule for a nonterminal A takes the form:

$$\mathrm{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

  where each w_i is a string of terminals and nonterminals

Context-Free Grammars (CFGs)

- For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \varepsilon$

Context-Free Grammars

For Example:
S → Ab | c
A → Aa | ε

- We think of Context-Free Grammars as **generating** strings.

1. Start from the start symbol S.
2. Choose a nonterminal, e.g. S, in the string, and replace it by one of the w's in the rules for S

$$S \to w_1 \mid w_2 \mid \ldots \mid w_k$$

3. Repeat step 2 until there are no nonterminals left.

- The language that the CFG describes is the set of all strings that it generates.

Example Context-Free Grammars

Example: $\mathbf{S} \rightarrow 0\mathbf{S}0 \mid 1\mathbf{S}1 \mid 0 \mid 1 \mid \varepsilon$

How does this grammar generate 0110?

$\mathbf{S} \rightarrow 0\mathbf{S}0 \rightarrow 01\mathbf{S}10 \rightarrow 01\varepsilon10 = 0110$

Example

$S \rightarrow 0S \mid S1 \mid \varepsilon$

- The set of all binary strings with any number of 0s followed by any number of 1s

Example

$$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$$

- The set of all binary palindromes.

Example

CFG for the language $\{0^n 1^n : n \geq 0\}$

Example

CFG for the language $\{0^n 1^n : n \geq 0\}$

$S \rightarrow 0S1 \mid \varepsilon$

Example

CFG for the language $\{0^n 1^n 23 : n \geq 0\}$

Example

CFG for the language $\{0^n 1^n 23 : n \geq 0\}$

- $S \rightarrow A23$
- $A \rightarrow 0A1 \mid \varepsilon$

Exercises

- CFG for the set of binary strings with the same number of 0s as 1s.

- CFG for the set of balanced parentheses. E.g. ((())())

Exercises

- CFG for the set of binary strings with the same number of 0s as 1s.
- $S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$

- CFG for the set of balanced parentheses. E.g. ((())())
- $S \rightarrow SS \mid (S) \mid \varepsilon$