CS109B Notes for Lecture 4/24/95

Regular Expressions in UNIX

1. *Character Class*: $[a_1 a_2 \cdots a_n]$ is shorthand for $a_1 \mid a_2 \mid \cdots \mid a_n$.

   □ Also, $\alpha - \beta$ stands for the set of characters with ASCII codes from the code for character α to the code for β.

Example: `[a-zA-Z]` denotes any of the 52 upper or lower case letters. `[-+*/]` denotes the four arithmetic operators.

- Note that `-` must come first to avoid it having a special meaning. `[+-*/]` denotes `/` and all the characters between `+` and `*`.

2. Additional operators:

   □ $R?$ stands for $\epsilon \mid R$.

   □ R^+ stands for $R \mid RR \mid RRR \mid \cdots$ (one or more occurrences of R).

3. Special symbols:

   □ Dot stands for "any ASCII character except the newline."

   □ `^` stands for the beginning of a line.

   □ `$` stands for the end of a line.

Example: The file `/usr/dict/words` contains common English words, one to a line. To find all 5-letter words beginning with `a` and with `b` as the fourth letter, issue the command

```
grep '^a..b.$' /usr/dict/words
```

The two words `adobe` and `alibi` are identified.

Example: Words with at least three `t`'s can be found by

```
grep 't.*t.*t' /usr/dict/words
```

- Note that `grep` scans for a pattern anywhere in the word. There is no need here to "anchor" the pattern at beginning or end.

- 153 words are found. `Afterthought` is the first and `uttermost` the last.

Class Problem

How would you search for words that have three `t`'s separated by at most one letter between each consecutive pair?

- E.g., `attitude`, `destitute`, `tattle`.
- Hint: you need the `?` operator and the command `egrep` (because `grep` doesn't allow `?`).

Class Problem

How would you search for all words beginning with 4 or more consonants (excluding `y`)?

- Only examples: `phthalate`, `schlieren`, `schnapps`.

Operator Precedence

- The unary, postfix operators, `*`, `+`, and `?` have highest precedence.
- Then comes concatentation.
- Union ($|$) is of lowest precedence.

Example: $a \mid bc?$ is grouped $a \mid (b(c?))$ and denotes the language $\{a, b, bc\}$.

Algebra of RE's

Like the set operators $\cup$ etc., there are many algebraic laws that apply to the regular expression operators.

- One approach: manipulate expressions to show equivalence:
  - ☐ Substitute RE's for variables in known equivalences.
  - ☐ Substitute an equivalent RE for another.
  - ☐ Use transitivity and commutativity of equivalence.

Example: Suppose $R(S \mid T) \equiv RS \mid RT$ is known. Substitute $R \Rightarrow R$, $S \Rightarrow \emptyset$, $T \Rightarrow \epsilon$, yields $R(\emptyset \mid \epsilon) \equiv R\emptyset \mid R\epsilon$.

Substitute $R\emptyset \equiv \emptyset$; $R\epsilon \equiv R$, yields $R(\emptyset \mid \epsilon) \equiv \emptyset \mid R$.

Substitute $R \mid \emptyset \equiv R$, yields $R(\emptyset \mid \epsilon) \equiv R$.

- Another approach: show containment in both directions.
  - ☐ Remember that the "meaning" of an RE is a language, i.e., a set of strings, so containment of sets makes sense.
- Read catalog of laws, pp. 569ff, FCS.

Example: Let us use a containment of sets argument to prove the following distributive law: $R(S \mid T) \equiv RS \mid RT$.

$\subseteq$.

- Let w be in $L(R(S \mid T)) = L(R)L(S \mid T)$.
- Then $w = rx$; r is in $L(R)$ and x is in $L(S \mid T) = L(S) \cup L(T)$.
  - ☐ Case 1: x in $L(S)$. Then $rx = w$ is in $L(RS)$. Therefore, w is in $L(RS \mid RT)$.
  - ☐ Case 2: x is in $L(T)$. Similarly, $rx = w$ is in $L(RT)$ and in $L(RS \mid RT)$.

$\supseteq$.

- Let w be in $L(RS \mid RT) = L(RS) \cup L(RT)$.
  - ☐ Case 1: w is in $L(RS) = L(R)L(S)$. Then $w = rs$, r is in $L(R)$ and s is in $L(S)$. Thus, s is in $L(S \mid T) = L(S) \cup L(T)$ and $rs = w$ is in $L(R(S \mid T))$.
  - ☐ Case 2: w is in $L(RT)$. Similar.