Regular Expressions

Regular Expression notation can be used to represent certain languages.

Regular languages are the languages represented by regular expressions.

Let $\Sigma$ be an alphabet.

- $\emptyset$ is a regular expression representing the empty language;
- $\epsilon$ is a regular expression representing the language $\{\epsilon\}$;
- $a \in \Sigma$ is a regular expression representing the language $\{a\}$;
if $r$ and $s$ are regular expressions representing languages $L(r)$ and $L(s)$, respectively, then

- $rs$ is a regular expression representing $L(r)L(s)$;
- $r + s$ or $r|s$ is a regular expression representing $L(r) \cup L(s)$;
- $r^*$ is a regular expression representing $L(r)^*$; and
- nothing else is a regular expression.

Operator precedence, from high to low $\ast, \cdot, +$ (or $|$), can be altered by parentheses.
Examples:

\[(a + b)^*\] all strings of a’s and b’s

\[(a + b)^*(a + bb)\] all strings of a’s and b’s

ending with a or bb

\[(aa)^*(bb)^*b\] even number of a’s followed by an odd number of b’s
More Examples:

- strings made by concatenating copies of $ab$’s and/or $aba$’s
- strings of 0’s and 1’s that end in 001
- strings of 0’s and 1’s that contain three consecutive 0’s
Applications

- compilers (scanning)

```
reg-expr --------> DFA

--------> |lex/flex | --------> (C-code)
```

- ls *

- grep
Problems: write regular expressions for each of the following:

1. an identifier, which is a letter followed by zero or more letters or digits

2. an integer, which is a sequence of one or more digits preceded by an optional sign

3. A floating point number, which is an optionally signed integer followed by a period, optionally followed by an unsigned integer and/or the letter E followed by an optionally signed integer.

4. A sequence of zero or more identifiers separated by one or more blanks with optional leading or trailing blanks