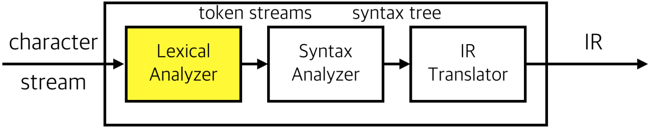
COSE312: Compilers

Lecture 2 — Lexical Analysis (1)

Hakjoo Oh

2015 Fall

Lexical Analysis



ex) Given a C program

```
float match0 (char *s) /* find a zero */
{if (!strncmp(s, "0.0", 3))
  return 0.0;
}
```

the lexical analyzer returns the stream of tokens:

FLOAT ID(match0) LPAREN CHAR STAR ID(s) RPAREN
LBRACE IF LPAREN BANG ID(strncmp) LPAREN ID(s)
COMMA STRING(0.0) COMMA NUM(3) RPAREN RPAREN
RETURN REAL(0.0) SEMI RBRACE EOF

Specification, Recognition, and Automation

1. **Specification**: how to specify lexical patterns?
   - `x`, `y`, `match0`, `_abc` are identifiers (ID)
   - `float`, `return` are keywords (FLOAT, RETURN)
   - 3, 12, 512 are numbers (NUM)

   ⇒ *regular expressions*

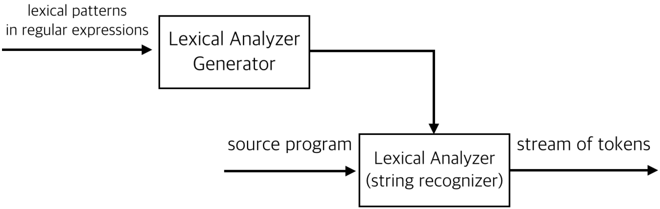
2. **Recognition**: how to *recognize* the lexical patterns?
   - Recognize `match0` as an identifier.
   - Recognize `float` as a keyword.
   - Recognize 512 as a number.

   ⇒ *deterministic finite automata*.

3. **Automation**: how to automatically generate string recognizers from specifications?

   ⇒ *Thompson's construction* and *subset construction*

cf) Lexical Analyzer Generator



- `lex`: a lexical analyzer generator for C
- `jlex`: a lexical analyzer generator for Java
- `ocamllex`: a lexical analyzer generator for OCaml

Part 1: Specification

- Preliminaries: alphabets, strings, languages
- Syntax and semantics of regular expressions
- Extensions of regular expressions

Alphabet

An alphabet $\boldsymbol{\Sigma}$ is a finite, non-empty set of symbols. E.g,

- $\boldsymbol{\Sigma = \{0, 1\}}$
- $\boldsymbol{\Sigma = \{a, b, \ldots, z\}}$

Strings

A string is a finite sequence of symbols chosen from an alphabet, e.g., 1, 01, 10110 are strings over $\Sigma = \{0, 1\}$. Notations:

- ϵ: the empty string.
- wv: the concatenation of w and v.
- w^R: the reverse of w.
- $|w|$: the length of string w:

$$\begin{aligned} |\epsilon| &= 0 \\ |va| &= |v| + 1 \end{aligned}$$

- If $w = vu$, then v is a *prefix* of w, and u is a *suffix* of w.
- Σ^k: the set of strings over Σ of length k
- Σ^*: the set of all strings over alphabet Σ:

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{i \in \mathbb{N}} \Sigma^i$$

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \cdots = \Sigma^* \setminus \{\epsilon\}$

Languages

A language L is a subset of Σ^*: $L \subseteq \Sigma^*$.

- $L_1 \cup L_2, \quad L_1 \cap L_2, \quad L_1 - L_2$
- $L^R = \{w^R \mid w \in L\}$
- $\overline{L} = \Sigma^* - L$
- $L_1 L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$
- The *power* of a language, L^n:

$$\begin{array}{rcl} L^0 & = & \{\epsilon\} \\ L^n & = & L^{n-1}L \end{array}$$

- The *star-closure* (or *Kleene closure*) of a language, L^*:

$$L^* = L^0 \cup L^1 \cup L^2 \cup \cdots = \bigcup_{i \geq 0} L^i$$

- The *positive closure* of a language, L^+:

$$L^+ = L^1 \cup L^2 \cup L^3 \cup \cdots = \bigcup_{i \geq 1} L^i$$

Regular Expressions

A regular expression is a notation to denote a language.

- Syntax

$$
\begin{array}{rcl}
R & \rightarrow & \emptyset \\
  & | & \epsilon \\
  & | & a \in \Sigma \\
  & | & R_1 \mid R_2 \\
  & | & R_1 \cdot R_2 \\
  & | & R_1^* \\
  & | & (R)
\end{array}
$$

- Semantics

$$
\begin{array}{rcl}
L(\emptyset) & = & \emptyset \\
L(\epsilon) & = & \{\epsilon\} \\
L(a) & = & \{a\} \\
L(R_1 \mid R_2) & = & L(R_1) \cup L(R_2) \\
L(R_1 \cdot R_2) & = & L(R_1)L(R_2) \\
L(R^*) & = & (L(R))^* \\
L((R)) & = & L(R)
\end{array}
$$

Example

$$
\begin{aligned}
L(a^* \cdot (a \mid b)) &= L(a^*)L(a \mid b) \\
&= (L(a))^*(L(a) \cup L(b)) \\
&= (\{a\})^*(\{a\} \cup \{b\}) \\
&= \{\epsilon, a, aa, aaa, \ldots\}(\{a, b\}) \\
&= \{a, aa, aaa, \ldots, b, ab, aab, \ldots\}
\end{aligned}
$$

Exercises

Write regular expressions for the following languages:

- The set of all strings over $\Sigma = \{a, b\}$.
- The set of strings of a's and b's, terminated by ab.
- The set of strings with an even number of a's followed by an odd number of b's.
- The set of C identifiers.

Regular Definitions

Give names to regular expressions and use the names in subsequent expressions, e.g., the set of C identifiers:

$$\begin{array}{rcl} letter & \rightarrow & \mathtt{A} \mid \mathtt{B} \mid \cdots \mid \mathtt{Z} \mid \mathtt{a} \mid \mathtt{b} \mid \cdots \mid \mathtt{z} \mid \_ \\ digit & \rightarrow & \mathtt{0} \mid \mathtt{1} \mid \cdots \mid \mathtt{9} \\ id & \rightarrow & letter(letter \mid digit)^* \end{array}$$

Formally, a *regular definition* is a sequence of definitions of the form:

$$\begin{array}{rcl} d_1 & \rightarrow & r_1 \\ d_2 & \rightarrow & r_2 \\ & \cdots & \\ d_n & \rightarrow & r_n \end{array}$$

1. Each d_i is a new name such that $d_i \not\in \Sigma$.
2. Each r_i is a regular expression over $\Sigma \cup \{d_1, d_2, \ldots, d_{i-1}\}$.

Example

Unsigned numbers (integers or floating point), e.g., `5280`, `0.01234`, `6.336E4`, or `1.89E-4`:

$$
\begin{array}{rcl}
digit & \rightarrow & \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{9} \\
digits & \rightarrow & digit\ digit^* \\
optionalFraction & \rightarrow & .\ digits \mid \epsilon \\
optionalExponent & \rightarrow & (\texttt{E}\ (\texttt{+} \mid \texttt{-} \mid \epsilon)\ digits) \mid \epsilon \\
number & \rightarrow & digits\ optionalFraction\ optionalExponent
\end{array}
$$

Extensions of Regular Expressions

1. R^+: the positive closure of R, i.e., $L(R^+) = L(R)^+$.
2. $R?$: zero or one instance of R, i.e., $L(R?) = L(R) \cup \{\epsilon\}$.
3. $[a_1 a_2 \cdots a_n]$: the shorthand for $a_1 \mid a_2 \mid \cdots \mid a_n$.
4. $[a_1\text{-}a_n]$: the shorthand for $[a_1 a_2 \cdots a_n]$, where $a_1, \ldots, a_n$ are consecutive symbols.
   - $[abc] = a \mid b \mid c$
   - $[a\text{-}z] = a \mid b \mid \cdots \mid z$.

Examples

- C identifiers:

$$\begin{array}{rcl} letter & \rightarrow & \texttt{[A-Za-z\_]} \\ digit & \rightarrow & \texttt{[0-9]} \\ id & \rightarrow & letter\ (letter|digit)^* \end{array}$$

- Unsigned numbers:

$$\begin{array}{rcl} digit & \rightarrow & \texttt{[0-9]} \\ digits & \rightarrow & digit^+ \\ number & \rightarrow & digits\ (.\ digits)?\ (\texttt{E}\ [+-]?\ digits)? \end{array}$$