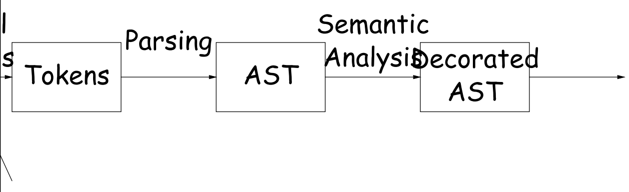
Lecture 2: Lexical Analysis

iew: Front End Compiler Structure



re here

of translations that each:

errors

r put aside extraneous information

a more conveniently accessible.

nd tools that partially automate this procedure.

analysis: convert description that uses patterns (ex-
ar expressions) into program.

Tokens

ts of *syntactic category* (like “noun” or “adjective”) plus
ormation (like a particular name).

“customer”) only needs syntactic category:

to the store” and “Harry went to the beach” have same
cal structure.

ming, semantic information might be text of identifier

n Notes:

```
/* No work needed */     ⟹    IF, LPAR, ID("i"), EQUALS,
                               ID("j"), RPAR, ID("z"),
                               ASSIGN, INTLIT("0"), SEMI,
                               ELSE, ID("z"), ASSIGN,
                               INTLIT("1"), SEMI
```

Classical Regular Expressions

essions denote formal languages, which are sets of strings
from some alphabet).

since internal structure not all that complex yet.

R denotes language $L(R)$:

$") = \{""\}$.

haracter, $L(c) = \{"c"\}$.

are r.e.s, $L(R_1R_2) = \{x_1x_2 | x_1 \in L(R_1), x_2 \in L(R_2)\}$.

$= L(R_1) \cup L(R_2)$.

$L(\epsilon) \cup L(R) \cup L(R\ R) \cup \cdots$.

$L(R)$.

s ‘*’ (highest), concatenation, union (lowest). Parenthe-
ide grouping.

Abbreviations

sts, such as `[abcf-mxy]` in Java, Perl, or Python.

aracter lists, such as `[^aeiou]`.

asses such as `.` (dot), `\d`, `\s` in Java, Perl, Python.

$R*$).

R).

Extensions

renthesized expressions:

`re.match(r'\s*(\d+)\s*,\s*(\d+)\s*', '12,34')`, have
`1) == '12', m.group(2) == '34'`.

edy quantifiers:

`(r'(\d+).*', '1234ab')` makes `group(1)` match '1234'.
`(r'(\d+?).*', '1234ab')` makes `group(1)` match '1'.

`h(r'(^abc|qef)', L)` matches abc only at beginning of
d qef anywhere.

`h(r'(?m)(^abc|qef)', L)` matches abc only at begin-
ring or of any line.

`h(r'rowr(?=baz)', L)` matches an instance of 'rowr',
f 'baz' follows (does not match baz).

`h(r'(?<=rowr)baz', L)` matches an instance of 'baz',
f immediately preceded by 'rowr' (does not match rowr).

atterns: `re.search(r'(\S+),\1', L)` matches a word
the same word after a comma.

An Example

:

```
   /    =       ;    ,    (    )    <    >
->
lse fi   while
```

rals

rt with # and go to end of line.
ograms in Chapter 2 of Course Notes.)

Problems

erals in *C*, Java.

in *C*, Java.

t numerals.

n *C*, Java.

n Ada.

C++, Java.

kups.

keting.

Some Problem Solutions

erals in C, Java: `0|[1-9][0-9]*`

in C, Java: `[1-9][0-9]+|0[xX][0-9a-fA-F]+|0[0-7]*`

nt numerals: `(\d+\.\d*|\d*\.\d+)([eE][-+]?\d+)?|[0-9]+[eE][-`

n C, Java. (*ASCII only, no dollar signs*):
`-zA-Z_0-9]*`

n Ada: `[a-zA-Z]([a-zA-Z_0-9]|_[a-zA-Z0-9])*`

C++, Java: `//.*|/\*([^*]|\*[^/])*\*+/`
ne extended features: `//.*|/\*(.|\n)*?\*/`

keting: *Nothing much you can do here, except to note*
e beginnings of lines and to do some programming in the