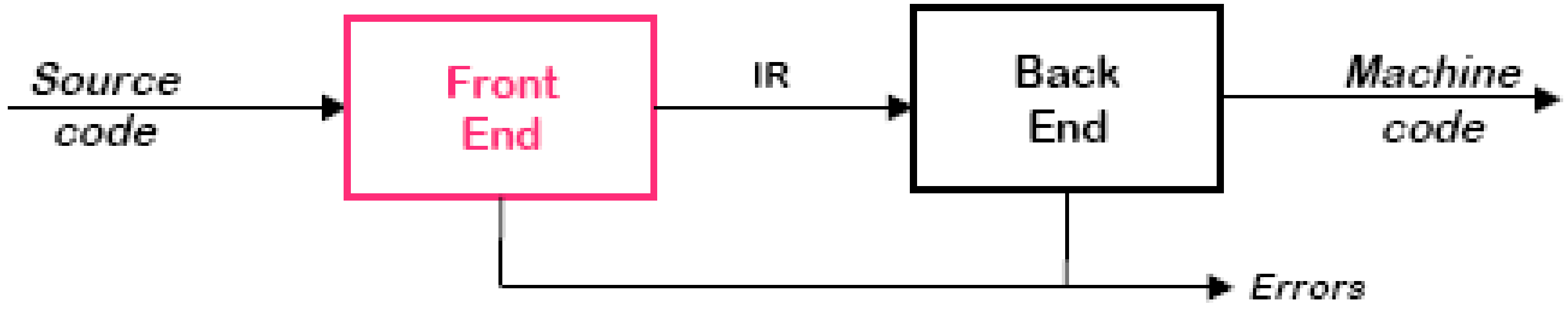
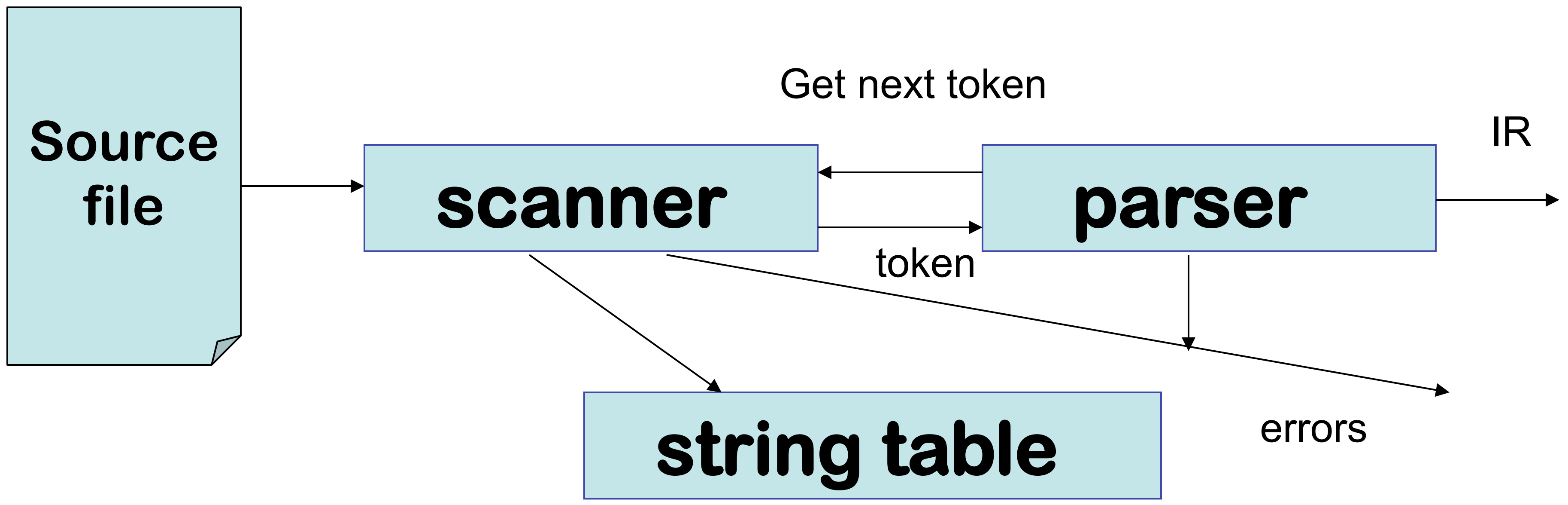
Class 4

The Front End: Scanning and Parsing



How they work together…



Since the scanner is the only phase to touch the input source file, what else does it need to do?

What is a token? A lexeme?

- English?
- Programming Languages?

- Lexeme
- Token
- Examples?

lexemes	tokens

Designing a Scanner

Step 1: define a finite set of tokens

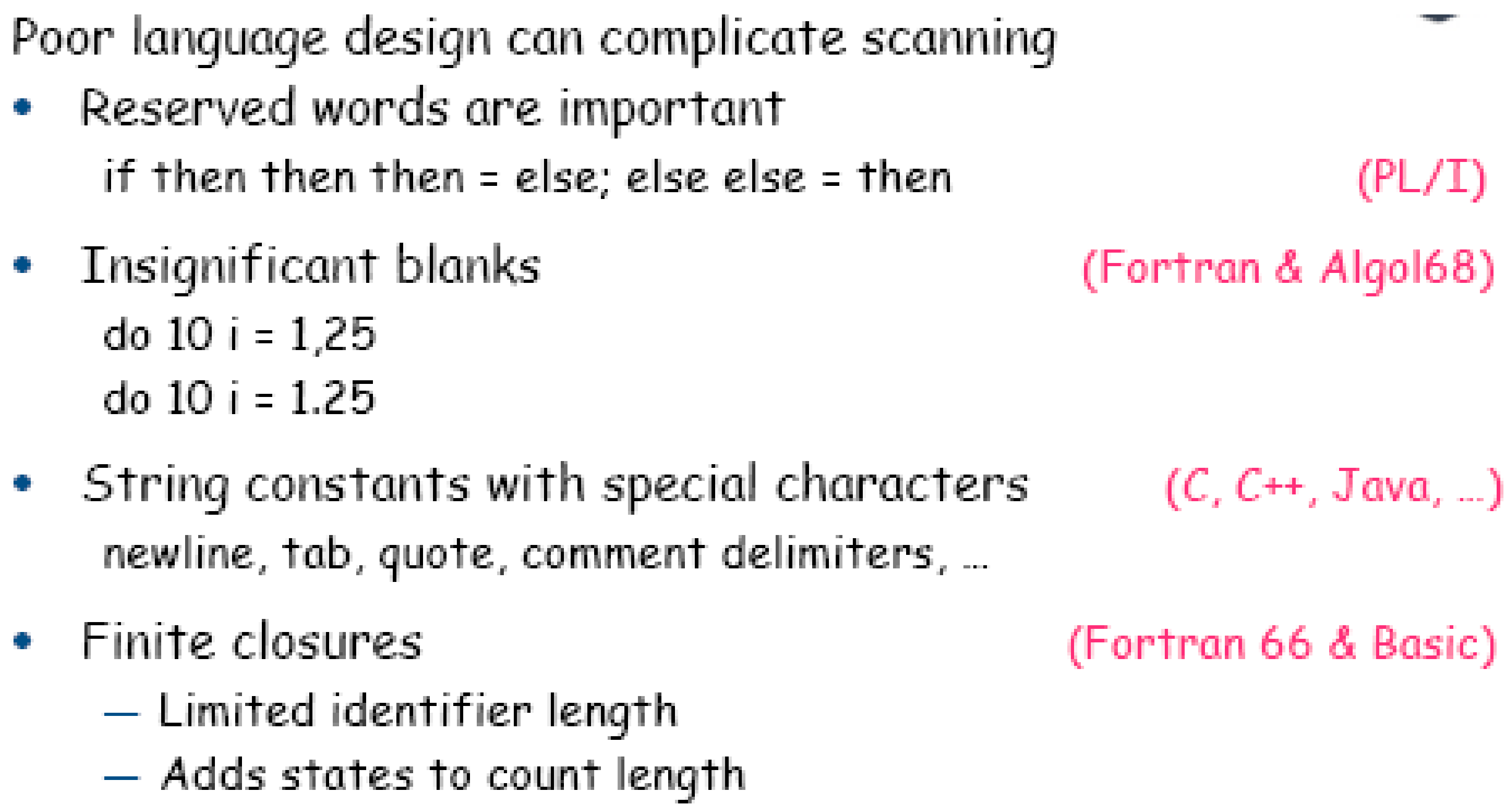
How?

Step 2: describe the strings (lexemes) for each token

How?

So, a simple scanner design?

Then, why did they invent lex?



Poor language design can complicate scanning

- Reserved words are important

  if then then then = else; else else = then (PL/I)

- Insignificant blanks (Fortran & Algol68)

  do 10 i = 1,25

  do 10 i = 1.25

- String constants with special characters (C, C++, Java, ...)

  newline, tab, quote, comment delimiters, ...

- Finite closures (Fortran 66 & Basic)
  - Limited identifier length
  - *Adds states to count length*

Even, simple examples: i vs if ; = vs ==

It is not so straightforward...

Specifying lexemes with Regular Expressions

Let ∑ be an alphabet.
Rules for Defining regular expressions over ∑ :

Help me out here, those from theory class!

Specifying lexemes with Regular Expressions

Let ∑ be an alphabet.
Rules for Defining regular expressions over ∑ :

- ε Denotes the set containing the empty string.
- For each a in ∑ , a is the reg expr denoting {a}

- If r and s are reg expr's, then

r s	= set of strings consisting of strings from r followed by strings from s
r | s	= set of strings for either r or s
r *	= 0 or more strings from r (closure)
(r)	used to indicate precedence

Reading Regular Expressions

- **Identifiers:**
  - **Letter -> (a|b|c|d|..|z|A|B|C…|Z)**
  - **Digit -> (0|1|2|…|9)**
  - **Identifier -> Letter (Letter | Digit)***
- **Numbers:**

  Integer -> (+|-|Ɛ) (0|1|2|3|..|9) (Digit*)
  Decimal -> Integer.Digit*
  Real -> (Integer | Decimal) E (+|-|Ɛ) Digit*

What strings/lexemes are represented by these regular expressions?

Practice with writing regular expressions

1. Binary numbers of at least one digit
2. Capitalized words
3. Legal identifiers that must start with a letter, can contain either upper or lower case letters, digits, or _.
4. white space including tabs, newlines, spaces

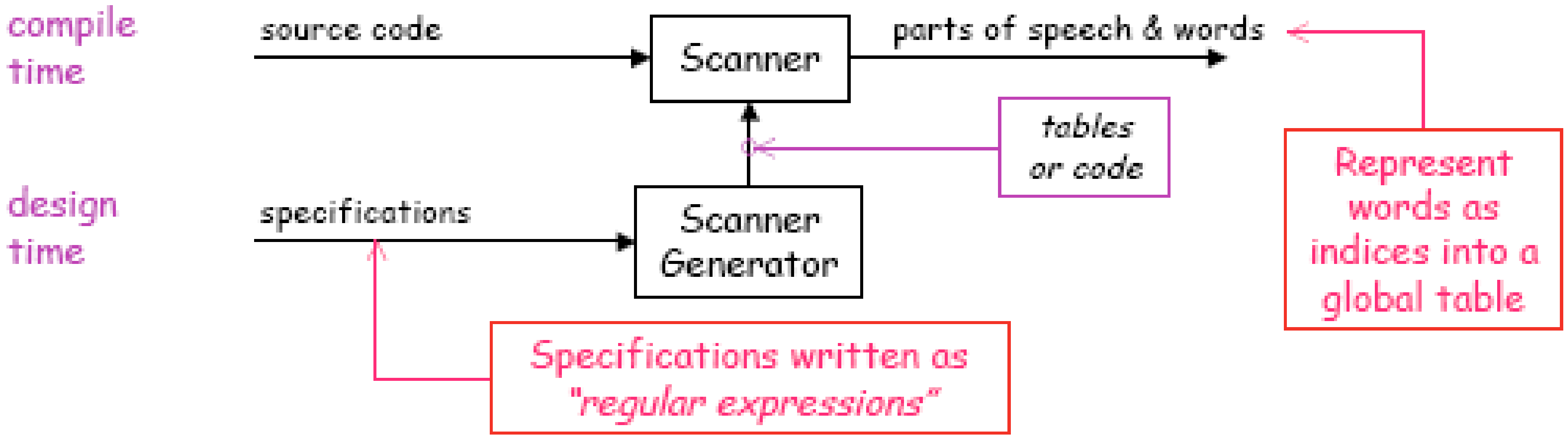
Shorthand for regular expressions?

What strings are accepted here?

- Numerical literals in Pascal may be generated by the following:

$$
\begin{array}{lcl}
\textit{digit} & \longrightarrow & \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9} \\
\textit{unsigned\_integer} & \longrightarrow & \textit{digit}\ \textit{digit}\ ^{*} \\
\textit{unsigned\_number} & \longrightarrow & \textit{unsigned\_integer}\ (\ (\ \texttt{.}\ \ \textit{unsigned\_integer}\ )\mid \epsilon\ ) \\
 & & (\ (\ (\ \texttt{e} \mid \texttt{E}\ )\ (\ \texttt{+} \mid \texttt{-} \mid \epsilon\ )\ \textit{unsigned\_integer}\ ) \mid \epsilon\ )
\end{array}
$$

The Scanner Generator



Form of a Lex/Flex Spec File

```
Definitions/declarations used for re clarity
%%
Reg exp0   {action0}  // translation rules to be
Reg exp1   {action1}   // converted to scanner
…                    …
%%
Auxiliary functions to be copied directly
```

Lex Spec Example

```
delim           [ \t\n]
ws              {delim}+
letter          [A-Za-z]
digit           [0-9]
id              {letter}({letter}|{digit})*
number          {digit}+(\.{digit}+)?(E[+-]?{digit}+)?
%%
{ws}            {/*no action and no return*?}
if              {return(IF);}
then            {return(THEN);}
{id}            {yylval=(int) installID(); return(ID);}
{number}        {yylval=(int) installNum(); return(NUMBER);}
%%

Int installID() {/* code to put id lexeme into string table*/}

Int installNum() {/* code to put number constants into constant table*/}
```