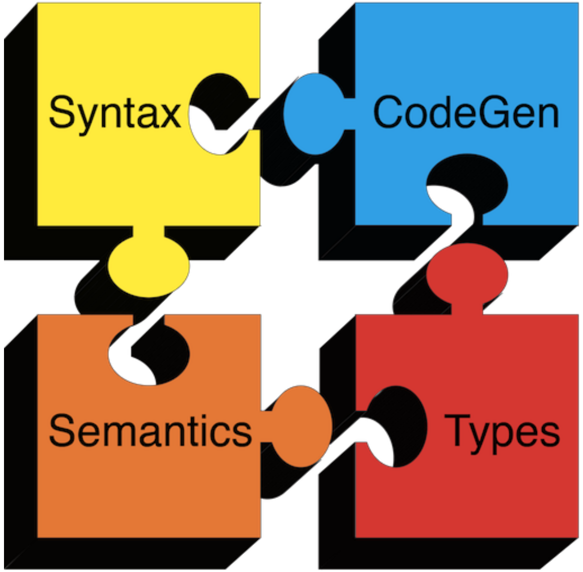

Compilers

Lexical Specification

Alex Aiken

Lexical Specification

- At least one: A^+ $\equiv$ AA^*
- Union: $A \mid B$ $\equiv$ $A + B$
- Option: $A?$ $\equiv A + \varepsilon$
- Range: ‘a’ +’ b’ +…+’ z’ $\equiv$ [a-z]
- Excluded range:

complement of [a-z] $\equiv$ [^a-z]

Lexical Specification

- Last lecture: a specification for the predicate

$$\underline{s} \in \underbrace{\underline{L}(\underline{R})}_{\text{sets of strings}}$$

- Not enough!

$$\underline{c_1\, c_2\, c_3}\;\Big|\;\underline{c_4\, c_5\, c_6\, c_7}\;\Big|\;\ldots\;.$$

Lexical Specification

1. Write a rexp for the lexemes of each token class
    - Number = digit^+
    - Keyword = ‘if’ + ‘else’ + …
    - Identifier = letter (letter + digit)*
    - OpenPar = ‘(‘
    - …

Lexical Specification

2. Construct R, matching all lexemes for all tokens

$$R \underline{= \text{Keyword} + \text{Identifier} + \text{Number} + \ldots}$$

$$= \underline{R}_1 + \underline{R}_2 + \ldots$$

Lexical Specification

3. Let input be $\underline{x_1 \ldots x_n}$

   For $1 \leq i \leq n$ check

   $\underline{x_1 \ldots x_i} \in \underline{L(R)}$?

4. If success, then we know that

   $\underline{x_1 \ldots x_i} \in L(\underline{R_j})$ for some j

   $R = R_1 + R_2 + R_3 + \cdots$

5. Remove $\underline{x_1 \ldots x_i}$ from input and go to (3)

Lexical Specification

- How much input is used?

$$\underline{x_1 \ldots x_i} \in L(R)$$

$$\underline{x_1 \ldots x_j} \in L(R)$$

$$i \neq j$$

"Maximal Munch"

Lexical Specification

Choose the one listed first

- Which token is used?

$x_1 \ldots x_i \in L(R) \qquad R = R_1 + \ldots + R_N$

$x_1 \ldots x_i \in L(R_j)$

$x_1 \ldots x_i \in L(R_k)$

$\text{if} \in \begin{matrix} L(\text{Keywords}) \\ L(\text{Identifiers}) \end{matrix}$

$\text{Keywords} = \text{'if'} + \text{'else'} + \ldots$

$\text{Identifiers} = \text{letter}(\text{letter} + \text{digit})^*$

Lexical Specification

- What if no rule matches?

$x_1 \ldots x_i \notin L(R)$] No!

ERROR = [all strings not in the lexical spec]

→ put it last in priority

Lexical Specification

- Regular expressions are a concise notation for string patterns

- Use in lexical analysis requires small extensions
  - To resolve ambiguities - [matches as long as possible
    highest priority match]
  - To handle errors

- Good algorithms known
  - Require only single pass over the input
  - Few operations per character (table lookup)