The Syntactic Side of Languages (Again)

Natural Languages

- stream of phonemes via lexical analysis → stream of words via parsing → sentences

Artificial Languages

- stream of characters via lexical analysis → stream of tokens via parsing → abstract syntax

Tokens: Variable names, numerals, operators, key-words, ...

```
int main(void) {
    printf("hello, world\n");
    return 0;
}
```

Context Free Grammars, 1

Grammars rules for organizing
- word-streams into sentences
- token-streams into abstract syntax (parse trees)

Context Free Grammars (CFGs)
- **Terminals**: concrete syntax (e.g., `printf([ ... ])
- **Nonterminals**: syntactic categories: (e.g., Noun-Phrase, key-word, ...)

Example (Palandromes over \{ a, b, c \})

```
A ::= e | a | b | c | aAa | bAb | cAc
```
CFGs, 2

- CFGs recursively specify a finite collection of sets of strings, syntactic categories.
- Each syntactic category is named by a nonterminal symbol. E.g.: \langle object \rangle, \langle verb1 \rangle, and \langle noun \rangle.
- One of the nonterminals is chosen to be the start symbol; its syntactic category is the language given by the grammar. E.g.: \langle sentence \rangle.
- A syntactic category (named by nonterminal N) is described by a set of productions of the form:

  \[ N ::= X_1 \ldots X_n \]

where each \( X_i \) is a terminal or nonterminal (and \( n \) could be 0). E.g.:

- \langle sentence \rangle ::= \langle subject \rangle \langle verb1 \rangle
- \langle sentence \rangle ::= \langle subject \rangle \langle verb2 \rangle \langle object \rangle
- \langle object \rangle ::= that \langle sentence \rangle

Example: Translating a regular expression to CFG

**Notation:** \( X_e \) = the nonterminal for reg. exp. \( e \)

For:

- \( e = a \) \( X_e ::= a \)
- \( e = \epsilon \) \( X_e ::= \epsilon \)
- \( e = (e_1|e_2) \) \( X_e ::= X_{e_1} X_{e_2} \mid X_{e_2} \)
- \( e = (e)\star \) \( X_e ::= X_e X_e \mid \epsilon \)

For \( e = (01|10)^\star \):

- \( X_{(01|10)^\star} ::= X_{01|10} X_{(01|10)^\star} \mid \epsilon \)
- \( X_{01|10} ::= X_{01} \mid X_{10} \)
- \( X_{01} ::= X_0 X_1 \)
- \( X_{00} ::= 0 \)
- \( X_{10} ::= X_1 X_0 \)
- \( X_{11} ::= 1 \)
A Big-Step Semantics for CFG

Notation: $N \Downarrow w$ means $w$ is in $N$’s syntactic category.

$$N_1 \Downarrow w_1 \cdots N_k \Downarrow w_k \quad (N := u_0 N_1 u_1 N_2 \cdots N_k u_k) \quad w = u_0 w_1 u_1 \cdots w_k u_k$$

$(	ext{exp}) ::= (\text{exp}) + (\text{exp})$
$| (\text{exp}) - (\text{exp})$
$| (\text{exp}) \times (\text{exp})$
$| (\text{exp}) \div (\text{exp})$
$| (\text{num})$
$| ((\text{exp}))$

A dodgy grammar

$(\text{exp}) := (\text{exp}) + (\text{exp})$
$\text{num} \Downarrow 3$
$\text{num} \Downarrow 4$
$\text{exp} \Downarrow 2$
$\text{exp} \Downarrow 3$
$\text{exp} \Downarrow 4$

($\star$) “$3 \times 4 = 3'' + '+' + '+' + '+' + 4''$
($\star\star$) “$2 + 3 \times 4 = 2'' + '+' + '+' + '+' + 3 \times 4''$

Parse Trees

Definition (Ambiguity)

A CFG is ambiguous when some string in the language has two possible parses. (Great for lawyers, not-so-great in computing.)

[From a newspaper discussion of a documentary on Merle Haggard.]

“Among those interviewed were his two ex-wives, Kris Kristofferson and Robert Duvall.”

Grammar Repair, 1 (§3.4 in Mogensen)

Definition

Suppose $\oplus$ is an operator (e.g., $+$, $\ast$, $\div$).

(a) $\oplus$ is left-associative when $a \oplus b \oplus c = (a \oplus b) \oplus c$. (E.g., $-\div$)

(b) $\oplus$ is right-associative when $a \oplus b \oplus c = a \oplus (b \oplus c)$. (E.g., $\ast\div =$ in C)

(c) $\oplus$ is non-associative when $a \oplus b \oplus c$ is illegal. (E.g., $<$)

- $+\ast$ can be either left- or right-associative.
- To be consistent with $-\div$, we treat them as left-assoc.

<table>
<thead>
<tr>
<th>For</th>
<th>rewrite</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>left-assoc. $\oplus$</td>
<td>$E := E \oplus E \mid \text{num}$</td>
<td>$E := E \oplus E' \mid E'$</td>
</tr>
<tr>
<td>right-assoc. $\oplus$</td>
<td>$E := E \oplus E \mid \text{num}$</td>
<td>$E := E' \oplus E' \mid E'$</td>
</tr>
</tbody>
</table>

[What is the parse of $1 \oplus 2 \oplus 3$ under these two grammars?]
Grammar Repair, 3  (§3.4 in Mogensen)

\[
\langle \text{exp} \rangle ::= \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle - \langle \text{exp} \rangle \quad \text{(level 1 precedence)}
\mid \langle \text{exp} \rangle \ast \langle \text{exp} \rangle \mid \langle \text{exp} \rangle / \langle \text{exp} \rangle \quad \text{(level 2 precedence)}
\mid \langle \text{num} \rangle \mid (\langle \text{exp} \rangle) \quad \text{(level 3 precedence)}
\]

▶ Handle left- and right-associativity as before.
▶ Each level gets its own nonterminal.
▶ Go from lowest to highest precedence levels.

\[
\langle \text{exp} \rangle_1 ::= \langle \text{exp} \rangle_1 + \langle \text{exp} \rangle_2 \mid \langle \text{exp} \rangle_1 - \langle \text{exp} \rangle_2 \mid \langle \text{exp} \rangle_2
\]

\[
\langle \text{exp} \rangle_2 ::= \langle \text{exp} \rangle_2 \ast \langle \text{exp} \rangle_3 \mid \langle \text{exp} \rangle_2 / \langle \text{exp} \rangle_3 \mid \langle \text{exp} \rangle_3
\]

\[
\langle \text{exp} \rangle_2 :: = \langle \text{num} \rangle \mid (\langle \text{exp} \rangle_1)
\]

[More problems and repairs in the next homework.]

Digression

See Graham Hutton’s slides for Chapter 8 of his “Programming in Haskell” text

http://www.cs.nott.ac.uk/~gmh/chapter8.ppt

Also:
▶ Hutton’s “Programming in Haskell, 2/e” homepage:
▶ Hutton’s Example Parsing Library
(From the 1st edition — Not GHC 8.0.1 compliant):
http://www.cs.nott.ac.uk/~gmh/Parsing.lhs
▶ Erik Meijer’s video lecture based on the Hutton’s Chapter 8
C9-Lectures-Dr-Erik-Meijer-Functional-Programming-Fundamentals-Chapter-8-of-13
(Skip to time 6:05 for the beginning for the discussion of parsers.)