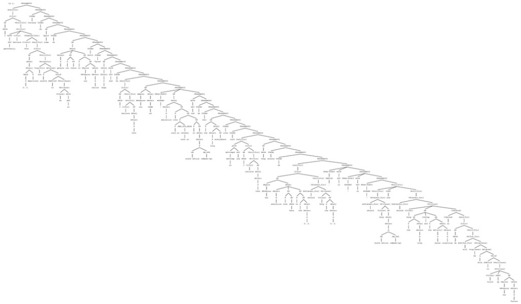
CIS 352

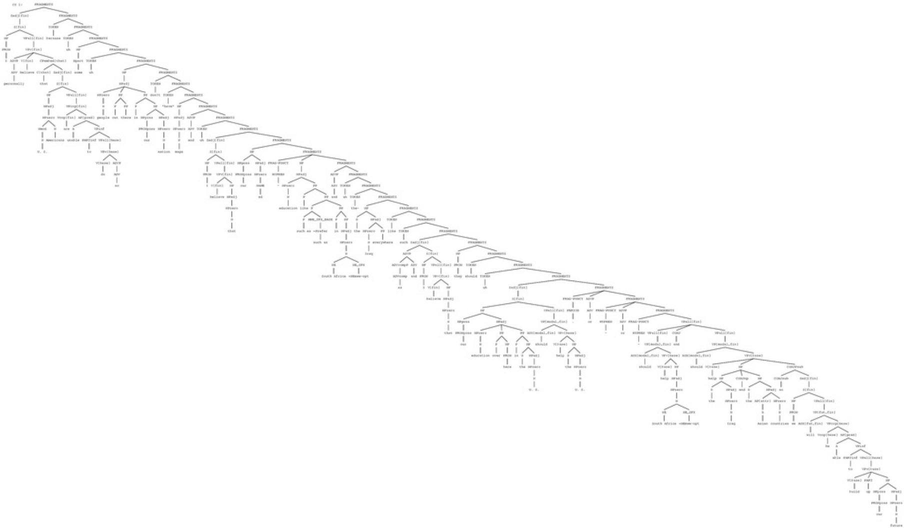
Parsing, Part I

Jim Royer

April 2, 2019



Miss Teen South Carolina's Famous Answer



https://kellblog.com/2007/09/01/parsing-the-unparseable-miss-teen-south-carolinas-answer/

The Syntactic Side of Languages (Again)

Natural Languages

stream of phonemes $\xrightarrow[\text{analysis}]{\text{via lexical}}$ stream of words $\xrightarrow{\text{via parsing}}$ sentences

Artificial Languages

stream of characters $\xrightarrow[\text{analysis}]{\text{via lexical}}$ stream of tokens $\xrightarrow{\text{via parsing}}$ abstract syntax

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

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

`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 Grammmars (CFGs)

- *Terminals:* concrete syntax (e.g., $\boxed{\text{printf}}$ $\boxed{(}$ ...)
- *Nonterminals:* syntactic categories: (e.g., Noun-Phrase, key-word, ...)

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

$$A \ ::= \ \epsilon \mid a \mid b \mid c \mid aAa \mid bAb \mid cAc$$

CFGs Examples: LC

$$
\begin{array}{rlcl}
\text{Phases} & P & ::= & C \mid E \mid B \\
\text{Commands} & C & ::= & \mathbf{skip} \mid \ell := E \mid C;\, C \\
 & & \mid & \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C \mid \mathbf{while}\ B\ \mathbf{do}\ C \\
\text{Integer Expressons} & E & ::= & n \mid\ !\ell \mid E \circledast E \qquad (\circledast \in \{+, -, \times, \ldots\}) \\
\text{Boolean Expressons} & B & ::= & b \mid E \circledast E \qquad (\circledast \in \{=, <, \geq, \ldots\})
\end{array}
$$

$$
\begin{array}{rrl}
\text{Integers} & n \in \mathbb{Z} & = \{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\} \\
\text{Booleans} & b \in \mathbb{B} & = \{\text{true}, \text{false}\} \\
\text{Locations} & \ell \in \mathbb{L} & = \{x_0, x_1, x_2, \ldots\} \\
 & & !\ell \equiv \text{the integer currently stored in } \ell
\end{array}
$$

```
x1 := 1; x2 := !x0;    // Computes factorial of !x0
while (!x2>0) do
   x1 := (!x1*!x2);
   x2 := (!x2-1)
```

CFGs Examples: A Fragment of English

$$
\begin{aligned}
\langle\text{sentence}\rangle &::= \langle\text{subject}\rangle\langle\text{verb1}\rangle \mid \langle\text{subject}\rangle\langle\text{verb2}\rangle\langle\text{object}\rangle \\
\langle\text{subject}\rangle &::= \langle\text{article}\rangle\langle\text{noun}\rangle \mid \langle\text{pronoun}\rangle \\
\langle\text{object}\rangle &::= \text{that } \langle\text{sentence}\rangle \\
\langle\text{verb1}\rangle &::= \text{swims} \mid \text{pauses} \mid \text{exists} \\
\langle\text{verb2}\rangle &::= \text{believes} \mid \text{hopes} \mid \text{imagines} \\
\langle\text{article}\rangle &::= \text{a} \mid \text{some} \mid \text{the} \\
\langle\text{noun}\rangle &::= \text{lizard} \mid \text{truth} \mid \text{man} \\
\langle\text{pronoun}\rangle &::= \text{he} \mid \text{she} \mid \text{it}
\end{aligned}
$$

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.: ⟨object⟩, ⟨verb1⟩, and ⟨noun⟩.
- One of the nonterminals is chosen to be the *start symbol*;
  its syntactic category is the language given by the grammar.
  E.g.: ⟨sentence⟩.
- 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_1 is a terminal or nonterminal (and n could be 0). E.g.:

  $$\langle\text{sentence}\rangle ::= \langle\text{subject}\rangle\langle\text{verb1}\rangle$$
  $$\langle\text{sentence}\rangle ::= \langle\text{subject}\rangle\langle\text{verb2}\rangle\langle\text{object}\rangle$$
  $$\langle\text{object}\rangle ::= \text{that } \langle\text{sentence}\rangle$$

Example: Translating a regular expression to CFG

Notation: X_e = the nonterminal for reg. exp. e

For:	Add:
$e = a$	$X_e ::= a$
$e = \epsilon$	$X_e ::= \epsilon$
$e = (e_1 \| e_2)$	$X_e ::= X_{e_1} \mid X_{e_2}$
$e = (e_1 e_2)$	$X_e ::= X_{e_1} X_{e_2}$
$e = (e')^*$	$X_e ::= X_{e'} X_e \mid \epsilon$

For $e = (01|10)^*$:

$$X_{(01|10)^*} ::= X_{01|10} X_{(01|10)^*} \mid \epsilon \qquad X_{01|10} ::= X_{01} \mid X_{10}$$

$$X_{01} ::= X_0 X_1 \qquad X_{10} ::= X_1 X_0$$

$$X_0 ::= 0 \qquad X_1 ::= 1$$

A Big-Step Semantics for CFG

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

$$\frac{N_1 \Downarrow w_1 \quad \cdots N_k \Downarrow w_k}{N \Downarrow w} \left(\begin{array}{c} N ::= u_0 N_1 u_1 N_2 \ldots N_k u_k \\ w = u_0 w_1 u_1 \ldots w_k u_k \end{array} \right)$$

$$\begin{array}{rcl} \langle \text{exp} \rangle & ::= & \langle \text{exp} \rangle + \langle \text{exp} \rangle \\ & | & \langle \text{exp} \rangle - \langle \text{exp} \rangle \\ & | & \langle \text{exp} \rangle * \langle \text{exp} \rangle \\ & | & \langle \text{exp} \rangle / \langle \text{exp} \rangle \\ & | & \langle \text{num} \rangle \\ & | & (\langle \text{exp} \rangle) \end{array}$$

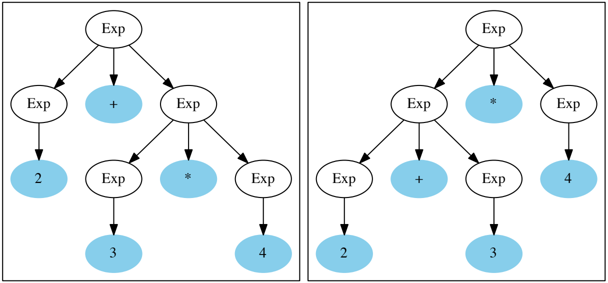
A dodgy grammar

$$\dfrac{\dfrac{\dfrac{}{\langle \text{num} \rangle \Downarrow 2}}{\langle \text{exp} \rangle \Downarrow 2} \qquad \dfrac{\dfrac{\dfrac{}{\langle \text{num} \rangle \Downarrow 3}}{\langle \text{exp} \rangle \Downarrow 3} \qquad \dfrac{\dfrac{}{\langle \text{num} \rangle \Downarrow 4}}{\langle \text{exp} \rangle \Downarrow 4}}{\langle \text{exp} \rangle \Downarrow 3 * 4}\,(\star)}{\langle \text{exp} \rangle \Downarrow 2 + 3 * 4}\,(\dagger)$$

($\star$) "3*4" = "3"++"*"++"4"
($\dagger$) "2+3*4" = "2"++"+"++"3*4"

Parse Trees

$$
\begin{aligned}
\langle \text{exp} \rangle ::= \; & \langle \text{exp} \rangle + \langle \text{exp} \rangle \\
\mid \; & \langle \text{exp} \rangle - \langle \text{exp} \rangle \\
\mid \; & \langle \text{exp} \rangle * \langle \text{exp} \rangle \\
\mid \; & \langle \text{exp} \rangle \, / \, \langle \text{exp} \rangle \\
\mid \; & \langle \text{num} \rangle \\
\mid \; & (\langle \text{exp} \rangle)
\end{aligned}
$$



Two parses of $2 + 3 * 4$

Definition (Ambiguity)

A CFG is *abmiguous* when some 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., $+$, $*$, $<$).

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

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

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

- $+$ and $*$ can be either left- or right-associative.
- To be consistent with $-$ and $/$, we treat them as left-assoc.

For	rewrite	to
left-assoc. $\oplus$	$E ::= E \oplus E \mid \langle\text{num}\rangle$	$E ::= E \oplus E' \mid E'$ $E' ::= \langle\text{num}\rangle$
right-assoc. $\oplus$	$E ::= E \oplus E \mid \langle\text{num}\rangle$	$E ::= E' \oplus E \mid E'$ $E' ::= \langle\text{num}\rangle$

[What is the parse of $1 \oplus 2 \oplus 3$ under these two grammars?]

Definition

Operators have an ordering called *precedence*.
In an expression $a \oplus b \odot c$:

- if $\mathit{precedence}(\oplus) > \mathit{precedence}(\odot)$, then: $a \oplus b \odot c = (a \oplus b) \odot c$.
- if $\mathit{precedence}(\oplus) < \mathit{precedence}(\odot)$, then: $a \oplus b \odot c = a \oplus (b \odot c)$.
- if $\mathit{precedence}(\oplus) = \mathit{precedence}(\odot)$, then:
  - ⇨ if $\oplus$ and $\otimes$ are both left-assoc., then: $a \oplus b \odot c = (a \oplus b) \odot c$.
  - ⇨ if $\oplus$ and $\odot$ are both right-assoc., then: $a \oplus b \odot c = a \oplus (b \odot c)$.
  - ⇨ Otherwise, no standard answer.

Grammar Repair, 3 (§3.4 in Mogensen)

$$
\begin{array}{rcll}
\langle \text{exp} \rangle & ::= & \langle \text{exp} \rangle + \langle \text{exp} \rangle \mid \langle \text{exp} \rangle - \langle \text{exp} \rangle & \textit{(level 1 precedence)} \\
& \mid & \langle \text{exp} \rangle * \langle \text{exp} \rangle \mid \langle \text{exp} \rangle / \langle \text{exp} \rangle & \textit{(level 2 precedence)} \\
& \mid & \langle \text{num} \rangle \mid (\langle \text{exp} \rangle) & \textit{(level 3 precedence)}
\end{array}
$$

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

$$
\begin{array}{rcl}
\langle \text{exp} \rangle_1 & ::= & \langle \text{exp} \rangle_1 + \langle \text{exp} \rangle_2 \mid \langle \text{exp} \rangle_1 - \langle \text{exp} \rangle_2 \mid \langle \text{exp} \rangle_2 \\
\langle \text{exp} \rangle_2 & ::= & \langle \text{exp} \rangle_2 * \langle \text{exp} \rangle_3 \mid \langle \text{exp} \rangle_2 / \langle \text{exp} \rangle_3 \mid \langle \text{exp} \rangle_3 \\
\langle \text{exp} \rangle_3 & ::= & \langle \text{num} \rangle \mid (\langle \text{exp} \rangle_1)
\end{array}
$$

[More problems and repairs in the next homework.]

Warning: Greek letters!

Notation:

(a) $\alpha N \beta \Rightarrow \alpha\gamma\beta$ means $\alpha N \beta$ rewrites to $\alpha\gamma\beta$ by applying the production $N ::= \gamma$.

$$\frac{}{G \vdash \alpha N \beta \Rightarrow \alpha\gamma\beta} \left(\begin{array}{c} N ::= \gamma \\ \text{is in } G \end{array}\right)$$

(b) $\Rightarrow^*$ = the reflexive-transitive closure of $\Rightarrow$.

$$\begin{array}{l}
\underline{\langle\text{sentence}\rangle} \\
\quad \Rightarrow \underline{\langle\text{subject}\rangle}\ \langle\text{verb2}\rangle\ \langle\text{object}\rangle \\
\quad \Rightarrow \underline{\langle\text{article}\rangle}\ \langle\text{noun}\rangle\ \langle\text{verb2}\rangle\ \langle\text{object}\rangle \\
\quad \Rightarrow \text{the}\ \underline{\langle\text{noun}\rangle}\ \langle\text{verb2}\rangle\ \langle\text{object}\rangle \\
\quad \Rightarrow \text{the man}\ \underline{\langle\text{verb2}\rangle}\ \langle\text{object}\rangle \\
\quad \Rightarrow \text{the man believes}\ \underline{\langle\text{object}\rangle} \\
\quad \Rightarrow \text{the man believes that}\ \underline{\langle\text{sentence}\rangle} \\
\quad \Rightarrow \text{the man believes that}\ \underline{\langle\text{subject}\rangle}\ \langle\text{verb1}\rangle \\
\quad \Rightarrow \text{the man believes that}\ \underline{\langle\text{article}\rangle}\ \langle\text{noun}\rangle\ \langle\text{verb1}\rangle \\
\quad \Rightarrow \text{the man believes that some}\ \underline{\langle\text{noun}\rangle}\ \langle\text{verb1}\rangle \\
\quad \Rightarrow \text{the man believes that some lizard}\ \underline{\langle\text{verb1}\rangle} \\
\quad \Rightarrow \text{the man believes that some lizard exists}
\end{array}$$

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:
  `http://www.cs.nott.ac.uk/~gmh/book.html`
- 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
  `http://channel9.msdn.com/Series/`
  `C9-Lectures-Erik-Meijer-Functional-Programming-Fundamentals/`
  `C9-Lectures-Dr-Erik-Meijer-Functional-Programming-Fundamentals-Chapt`
  (Skip to time 6:05 for the beginning for the discussion of parsers.)

...