

The Language CFG

BNF-converter

April 16, 2017

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of CFG

Identifiers

Identifiers $\langle Ident \rangle$ are unquoted strings beginning with a letter, followed by any combination of letters, digits, and the characters `_` `'`, reserved words excluded.

Literals

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in CFG are the following:

`e`

The symbols used in CFG are the following:

<code>--></code>	<code>-></code>	<code>==></code>
<code>=></code>	<code>\n</code>	<code>{</code>
<code>}</code>	<code>(</code>	<code>)</code>
<code>+</code>	<code>*</code>	<code> </code>

Comments

Single-line comments begin with #.

There are no multiple-line comments in the grammar.

The syntactic structure of CFG

Non-terminals are enclosed between \langle and \rangle . The symbols $::=$ (production), $|$ (union) and ϵ (empty rule) belong to the BNF notation. All other symbols are terminals.

$$\langle Grammar \rangle ::= \langle ListRule \rangle$$
$$\begin{aligned} \langle Rule \rangle & ::= \langle LHS \rangle \langle ARROW \rangle \langle ListCRHS \rangle \\ & | \epsilon \end{aligned}$$
$$\begin{aligned} \langle ARROW \rangle & ::= \text{-->} \\ & | \text{->} \\ & | \text{==>} \\ & | \text{=>} \end{aligned}$$
$$\begin{aligned} \langle ListRule \rangle & ::= \epsilon \\ & | \langle Rule \rangle \backslash \mathbf{n} \langle ListRule \rangle \end{aligned}$$
$$\langle LHS \rangle ::= \langle Ident \rangle$$
$$\langle DISJSTART \rangle ::= \{$$
$$\langle DISJSTOP \rangle ::= \}$$
$$\langle BRSTART \rangle ::= ($$
$$\langle BRSTOP \rangle ::=)$$
$$\begin{aligned} \langle CRHS \rangle & ::= \langle DISJSTART \rangle \langle ListDRHS \rangle \langle DISJSTOP \rangle \\ & | \langle BRSTART \rangle \langle ListRHS \rangle \langle BRSTOP \rangle \\ & | \langle BRSTART \rangle \langle ListRHS \rangle \langle BRSTOP \rangle + \\ & | \langle BRSTART \rangle \langle ListRHS \rangle \langle BRSTOP \rangle * \\ & | \langle RHS \rangle \end{aligned}$$
$$\langle DRHS \rangle ::= \langle ListRHS \rangle$$

$$\begin{aligned}
\langle RHS \rangle & ::= \langle Ident \rangle \\
& \quad | \quad \langle Ident \rangle + \\
& \quad | \quad \langle Ident \rangle * \\
& \quad | \quad \langle String \rangle \\
& \quad | \quad e \\
\langle ListRHS \rangle & ::= \epsilon \\
& \quad | \quad \langle RHS \rangle \langle ListRHS \rangle \\
\langle ListCRHS \rangle & ::= \epsilon \\
& \quad | \quad \langle CRHS \rangle \langle ListCRHS \rangle \\
\langle ListDRHS \rangle & ::= \epsilon \\
& \quad | \quad \langle DRHS \rangle \\
& \quad | \quad \langle DRHS \rangle | \langle ListDRHS \rangle
\end{aligned}$$