

The Language Parser

BNF-converter

May 3, 2010

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 Parser

Literals

Integer literals $\langle Int \rangle$ are nonempty sequences of digits.

String literals $\langle String \rangle$ have the form " x ", where x is any sequence of any characters except " unless preceded by \.

CapIdent literals are recognized by the regular expression $\langle upper \rangle (\langle upper \rangle | \langle digit \rangle | ' _ ') * \langle lower \rangle (\langle letter \rangle | \langle digit \rangle | ' _ ') *$

LowIdent literals are recognized by the regular expression $\langle lower \rangle (\langle letter \rangle | \langle digit \rangle | ' _ ') *$

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 Parser are the following:

CHAN	DeSchedule	FOR
Ms	Ns	PAR
PROC	Print	S
SEQ	TO	Us
VAR	Work	

The symbols used in Parser are the following:

$\begin{array}{l} : \quad ; \quad \{ \\ \} \quad [\quad] \\ = \quad ? \quad ! \\ + \quad - \quad (\\) \end{array}$

Comments

There are no single-line comments in the grammar.

There are no multiple-line comments in the grammar.

The syntactic structure of Parser

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.

$$\langle ProgramSpec \rangle ::= \langle ListGlobDecl \rangle$$

$$\begin{array}{l} \langle GlobDecl \rangle ::= \text{PROC } \langle CapIdent \rangle : \langle DeadLine \rangle : \langle SingleBlock \rangle \\ | \quad \langle Chan \rangle ; \end{array}$$

$$\begin{array}{l} \langle ListGlobDecl \rangle ::= \epsilon \\ | \quad \langle GlobDecl \rangle \langle ListGlobDecl \rangle \end{array}$$

$$\langle SingleBlock \rangle ::= \{ \langle ListVar \rangle \langle ListStmtCompound \rangle \}$$

$$\begin{array}{l} \langle StmtCompound \rangle ::= \langle Order \rangle \text{FOR } \langle LowIdent \rangle \langle AssignOp \rangle \langle Integer \rangle \text{TO } \langle Integer \rangle : \langle StmtBlock \rangle \\ | \quad \langle Order \rangle \langle StmtBlock \rangle \end{array}$$

$$\begin{array}{l} \langle ListStmtCompound \rangle ::= \epsilon \\ | \quad \langle StmtCompound \rangle \langle ListStmtCompound \rangle \end{array}$$

$$\langle StmtBlock \rangle ::= \{ \langle ListStmt \rangle \}$$

$$\begin{array}{l} \langle Order \rangle ::= \text{PAR} \\ | \quad \text{SEQ} \end{array}$$

$$\begin{array}{l} \langle ListOrder \rangle ::= \epsilon \\ | \quad \langle Order \rangle \langle ListOrder \rangle \end{array}$$

$$\begin{aligned}
\langle \text{DeadLine} \rangle &::= \langle \text{Integer} \rangle \text{S} \\
&| \langle \text{Integer} \rangle \text{Ms} \\
&| \langle \text{Integer} \rangle \text{Us} \\
&| \langle \text{Integer} \rangle \text{Ns} \\
\langle \text{ListDeadLine} \rangle &::= \epsilon \\
&| \langle \text{DeadLine} \rangle \langle \text{ListDeadLine} \rangle \\
\langle \text{Stmt} \rangle &::= \langle \text{LowIdent} \rangle \langle \text{AssignOp} \rangle \langle \text{Expr} \rangle ; \\
&| \langle \text{LowIdent} \rangle \langle \text{Array} \rangle \langle \text{AssignOp} \rangle \langle \text{Expr} \rangle ; \\
&| \text{DeSchedule} ; \\
&| \langle \text{CapIdent} \rangle ; \\
&| \langle \text{Expr} \rangle ; \\
&| \text{Work} ; \\
&| \langle \text{StmtCompound} \rangle \\
&| \text{Print} \langle \text{String} \rangle \langle \text{ListExpr} \rangle \\
&| \langle \text{ChanCom} \rangle ; \\
&| \langle \text{Stmt} \rangle ; \\
\langle \text{ListStmt} \rangle &::= \epsilon \\
&| \langle \text{Stmt} \rangle \langle \text{ListStmt} \rangle \\
\langle \text{Chan} \rangle &::= \text{CHAN} \langle \text{LowIdent} \rangle \\
&| \text{CHAN} \langle \text{LowIdent} \rangle \langle \text{Array} \rangle \\
\langle \text{ListChan} \rangle &::= \epsilon \\
&| \langle \text{Chan} \rangle ; \langle \text{ListChan} \rangle \\
\langle \text{Var} \rangle &::= \text{VAR} \langle \text{LowIdent} \rangle \\
&| \text{VAR} \langle \text{LowIdent} \rangle \langle \text{Array} \rangle \\
\langle \text{ListVar} \rangle &::= \epsilon \\
&| \langle \text{Var} \rangle ; \langle \text{ListVar} \rangle \\
\langle \text{Array} \rangle &::= [\langle \text{Expr} \rangle] \\
\langle \text{ListArray} \rangle &::= \epsilon \\
&| \langle \text{Array} \rangle \langle \text{ListArray} \rangle \\
\langle \text{AssignOp} \rangle &::= = \\
\langle \text{ChanCom} \rangle &::= \langle \text{ChanExpr} \rangle ? \langle \text{Expr} \rangle \\
&| \langle \text{ChanExpr} \rangle ! \langle \text{Expr} \rangle \\
\langle \text{ListChanCom} \rangle &::= \epsilon \\
&| \langle \text{ChanCom} \rangle ; \langle \text{ListChanCom} \rangle \\
\langle \text{ChanExpr} \rangle &::= \langle \text{LowIdent} \rangle \\
&| \langle \text{LowIdent} \rangle [\langle \text{Expr} \rangle]
\end{aligned}$$

$$\begin{aligned}
\langle ListChanExpr \rangle & ::= \epsilon \\
& \quad | \quad \langle ChanExpr \rangle ; \langle ListChanExpr \rangle \\
\langle Expr1 \rangle & ::= \langle Expr1 \rangle + \langle Expr2 \rangle \\
& \quad | \quad \langle Expr1 \rangle - \langle Expr2 \rangle \\
& \quad | \quad \langle Expr2 \rangle [\langle Expr2 \rangle] \\
& \quad | \quad \langle Expr2 \rangle \\
\langle Expr4 \rangle & ::= \langle Integer \rangle \\
& \quad | \quad \langle Expr5 \rangle \\
\langle Expr5 \rangle & ::= \langle LowIdent \rangle \\
& \quad | \quad (\langle Expr \rangle) \\
\langle ListExpr \rangle & ::= \epsilon \\
& \quad | \quad \langle Expr \rangle ; \langle ListExpr \rangle \\
\langle Expr \rangle & ::= \langle Expr1 \rangle \\
\langle Expr2 \rangle & ::= \langle Expr3 \rangle \\
\langle Expr3 \rangle & ::= \langle Expr4 \rangle
\end{aligned}$$