Reference Grammar of LCL (Version 2.4x)
(as used in LCLint Version 1.3)

Yang Meng Tan
(ymtan@larch.lcs.mit.edu)

July 20, 1994

Interfaces

interface	::=	{*import* | *use*} * {*export* | *private* | *claim* } *
import	::=	**imports** *(* *id* | " *id* " | < *id* > *)*$^+$, ;
use	::=	**uses** *traitRef*$^+$, ;
export	::=	*constDeclaration* | *varDeclaration* | *type* | *fcn* | *claim*
private	::=	**spec** { *constDeclaration* |*varDeclaration* | *type* | *fcn* }
constDeclaration	::=	**constant** *typeSpecifier* { *varId* [= *term*] }$^+$, ;
varDeclaration	::=	{ [**const** | **volatile**] } *lclTypeSpec* { *declarator* [= *term*] }$^+$, ;
traitRef	::=	*id* [(*renaming*)]
renaming	::=	*replace*$^+$, | *typeName*$^+$, *replace**,
replace	::=	*typeName* **for** { *opId* [: *sortId**, *mapSym* *sortId*] | *CType* }
typeName	::=	[**obj**] *lclTypeSpec* [*abstDeclarator*]

Functions

fcn	::=	*lclTypeSpec* *declarator* { *global* } * { *fcnBody* }
global	::=	*lclTypeSpec* *declarator*$^+$, ;
fcnBody	::=	[*letDecl*] [*checks*] [*requires*] [*modify*]
		[*ensures*] [*claims*]
letDecl	::=	**let** { *varId* [: *sortSpec*] **be** *term* }$^+$, ;
sortSpec	::=	*lclTypeSpec*
requires	::=	**requires** *lclPredicate* ;
checks	::=	**checks** *lclPredicate* ;
modify	::=	**modifies** { **nothing** | *storeRef*$^+$, } ;
storeRef	::=	*term* | [**obj**] *lclTypeSpec* * *
ensures	::=	**ensures** *lclPredicate* ;
claims	::=	**claims** *lclPredicate* ;

Types

type ::= *abstract* | *exposed*
abstract ::= [**mutable** | **immutable**] **type** *id* ;
exposed ::= **typedef** *lclTypeSpec* { *declarator* [{ *constraint* }] }$^+$, ;
  | { **struct** | **union** } *id* ;
constraint ::= **constraint** *quantifierSym* *id* : *id* (*lclPredicate*) ;
lclTypeSpec ::= *typeSpecifier* | *structSpec* | *enumSpec*
structSpec ::= [**struct** | **union**] [*id*] { *structDecl*$^+$ }
  | [**struct** | **union**] *id*
structDecl ::= *lclTypeSpec* *declarator*$^+$, ;
enumSpec ::= **enum** [*id*] { *id*$^+$, } | **enum** *id*
typeSpecifier ::= *id* | *CType*$^+$
CType ::= **void** | **char** | **double** | **float** | **int**
  | **long** | **short** | **signed** | **unsigned**
abstDeclarator ::= (*abstDeclarator*)
  | * [*abstDeclarator*]
  | [*abstDeclarator*] *arrayQual*
  | *abstDeclarator* ( )
  | [*abstDeclarator*] (*param**,)
param ::= [**out**] *lclTypeSpec* *parameterDecl*
  | [**out**] *lclTypeSpec* *declarator*
  | [**out**] *lclTypeSpec* [*abstDeclarator*]
declarator ::= *varId* | * *declarator*
  | (*declarator*)
  | *declarator* *arrayQual* | *declarator* (*param**,)
parameterDecl ::= *varId* | * *parameterDecl*
  | *parameterDecl* *arrayQual* | *parameterDecl* (*param**,)
arrayQual ::= [[*term*]]

Predicates

lclPredicate ::= *term*
term ::= **if** *term* **then** *term* **else** *term* | *equalityTerm*
  | *term* *logicalOp* *term*
equalityTerm ::= *simpleOpTerm* [{ *eqOp* | = } *simpleOpTerm*]
  | *quantifier+* (*term*)
simpleOpTerm ::= *simpleOp2*$^+$ *secondary* | *secondary* *simpleOp2*$^+$
  | *secondary* { *simpleOp2* *secondary* }*
simpleOp2 ::= *simpleOp* | *
secondary ::= *primary* | [*primary*] *bracketed* [: *sortId*] [*primary*]
  | *sqBracketed* [: *sortId*] [*primary*]
bracketed ::= *open* [*term* { { *sepSym* | , } *term* }*] *close*

$$
\begin{array}{lll}
\mathit{sqBracketed} & ::= & [\ \mathit{[}\ \mathit{term}\ \{\ \{\ \mathit{sepSym}\ |\ ,\ \}\ \mathit{term}\ \}^{*}\ \mathit{]}\ ] \\
\mathit{open} & ::= & \{\ |\ \mathit{openSym} \\
\mathit{close} & ::= & \}\ |\ \mathit{closeSym} \\
\mathit{primary} & ::= & (\ \mathit{term}\ )\ |\ \mathit{varId}\ |\ \mathit{opId}\ (\ \mathit{term}^{+},\ )\ |\ \mathit{lclPrimary} \\
 & & |\ \mathit{primary}\ \{\ \mathit{preSym}\ |\ \mathit{postSym}\ |\ \mathit{anySym}\ \} \\
 & & |\ \mathit{primary}\ \{\ \mathit{selectSym}\ |\ \mathit{mapSym}\ \}\ \mathit{id} \\
 & & |\ \mathit{primary}\ [\ \mathit{term}^{+},\ ] \\
 & & |\ \mathit{primary}:\ \mathit{sortId} \\
\mathit{lclPrimary} & ::= & \mathit{cLiteral}\ |\ \mathbf{result}\ |\ \mathbf{fresh}\ (\ \mathit{term}\ ) \\
 & & |\ \mathbf{trashed}\ (\ \mathit{storeRef}\ ) \\
 & & |\ \mathbf{unchanged}\ (\ \{\ \mathbf{all}\ |\ \mathit{storeRef}^{+},\ \}\ ) \\
 & & |\ \mathbf{sizeof}\ (\ \{\ \mathit{lclTypeSpec}\ |\ \mathit{term}\ \}\ ) \\
 & & |\ \mathbf{minIndex}\ (\ \mathit{term}\ ) \\
 & & |\ \mathbf{maxIndex}\ (\ \mathit{term}\ ) \\
 & & |\ \mathbf{isSub}\ (\ \mathit{term}\ ,\ \mathit{term}\ ) \\
\mathit{cLiteral} & ::= & \mathit{intLiteral}\ |\ \mathit{stringLiteral}\ |\ \mathit{singleQuoteLiteral}\ |\ \mathit{floatLiteral} \\
\mathit{quantifier} & ::= & \mathit{quantifierSym}\ \{\ \mathit{varId}:\ \mathit{[}\ \mathbf{obj}\ \mathit{]}\ \mathit{sortSpec}\ \}^{+}, \\
\mathit{varId} & ::= & \mathit{id} \\
\mathit{fcnId} & ::= & \mathit{id} \\
\mathit{sortId} & ::= & \mathit{id} \\
\mathit{opId} & ::= & \mathit{id}
\end{array}
$$

Claims

$$
\begin{array}{lll}
\mathit{claim} & ::= & \mathbf{claims}\ \mathit{id}\ (\ \mathit{param}^{*},\ )\ \{\ \mathit{global}\ \}^{*} \\
 & & \{\ \mathit{[\ letDecl\ ]}\ \mathit{[\ requires\ ]}\ \mathit{[\ body\ ]}\ \mathit{ensures}\ \} \\
 & & |\ \mathbf{claims}\ \mathit{fcnId}\ \mathit{id}\ ; \\
\mathit{body} & ::= & \mathbf{body}\ \{\ \mathit{fcnId}\ (\ \mathit{value}^{*},\ )\ ;\ \} \\
\mathit{value} & ::= & \mathit{cLiteral}\ |\ \mathit{varId}\ |\ (\ \mathit{value}\ ) \\
 & & |\ \mathit{[\ value\ ]}\ \mathit{simpleOp}\ \mathit{[\ value\ ]}\ |\ \mathit{fcnId}\ (\ \mathit{value}^{*},\ )
\end{array}
$$