

# The Language clafer

BNF-converter

October 1, 2015

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 clafer

### Literals

PosInteger literals are recognized by the regular expression  $\langle digit \rangle +$

PosDouble literals are recognized by the regular expression  $\langle digit \rangle + \text{'.'} \langle digit \rangle + \text{'e'}$  or  $\text{'-'} \text{'?'}$   $\langle digit \rangle +$

PosReal literals are recognized by the regular expression  $\langle digit \rangle + \text{'.'} \langle digit \rangle +$

PosString literals are recognized by the regular expression  $\text{'\"}$  ( $\langle anychar \rangle - [\text{'\"'} \backslash]$  or  $\text{'\'} [\text{'\"'} \backslash \text{'nt'}]$ )  $\text{'\"}$

PosIdent literals are recognized by the regular expression  $\langle letter \rangle (\langle letter \rangle | \langle digit \rangle | \text{'_'} | \text{'\"'})^*$

PosLineComment literals are recognized by the regular expression  $\{ \text{'\"'} / \text{'\"'} \} (\langle anychar \rangle - \text{' '})^*$

PosBlockComment literals are recognized by the regular expression  $\{ \text{'\"'} / \text{'*'} \} (\langle anychar \rangle - \text{'*'} | \text{'*'} + (\langle anychar \rangle - [\text{'\"'} /]))^* \text{'*'} + \text{'/'}$

PosAlloy literals are recognized by the regular expression  $\{ \text{'[alloy]'} \} (\langle anychar \rangle - \text{'|'} | \text{'|'} + (\langle anychar \rangle - \text{'|'}))^* \{ \text{'[|]'} \}$

PosChoco literals are recognized by the regular expression  $\{ \text{'[chocol]'} \} (\langle anychar \rangle - \text{'|'} | \text{'|'} + (\langle anychar \rangle - \text{'|'}))^* \{ \text{'[|]'} \}$

### 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 clafer are the following:

abstract	all	assert
disj	else	enum
if	in	lone
max	maximize	min
minimize	mux	no
not	one	opt
or	product	some
sum	then	xor

The symbols used in clafer are the following:

=	[	]
<<	>>	{
}	'	:
->	->>	:=
?	+	*
..		<=>
=>		&&
!	<	>
<=	>=	!=
-	/	%
#	<:	:>
++	,	--
**	&	.
;	\	(
)		

## Comments

There are no single-line comments in the grammar.

There are no multiple-line comments in the grammar.

## The syntactic structure of clafer

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

$\langle \text{Module} \rangle ::= \langle \text{ListDeclaration} \rangle$

$\langle \text{Declaration} \rangle ::= \text{enum } \langle \text{PosIdent} \rangle = \langle \text{ListEnumId} \rangle$   
 $\quad \quad \quad | \quad \quad \langle \text{Element} \rangle$

$\langle \text{Clafer} \rangle ::= \langle \text{Abstract} \rangle \langle \text{GCard} \rangle \langle \text{PosIdent} \rangle \langle \text{Super} \rangle \langle \text{Reference} \rangle \langle \text{Card} \rangle \langle \text{Init} \rangle \langle \text{Elements} \rangle$

$\langle \text{Constraint} \rangle ::= [ \langle \text{ListExp} \rangle ]$

$\langle \text{Assertion} \rangle ::= \text{assert } [ \langle \text{ListExp} \rangle ]$

$\langle \text{Goal} \rangle ::= << \text{min } \langle \text{ListExp} \rangle >>$   
 $\quad \quad \quad | \quad << \text{max } \langle \text{ListExp} \rangle >>$   
 $\quad \quad \quad | \quad << \text{minimize } \langle \text{ListExp} \rangle >>$   
 $\quad \quad \quad | \quad << \text{maximize } \langle \text{ListExp} \rangle >>$

$\langle \text{Abstract} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad \text{abstract}$

$\langle \text{Elements} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad \{ \langle \text{ListElement} \rangle \}$

$\langle \text{Element} \rangle ::= \langle \text{Clafer} \rangle$   
 $\quad \quad \quad | \quad \langle \text{' } \langle \text{Name} \rangle \langle \text{Card} \rangle \langle \text{Elements} \rangle$   
 $\quad \quad \quad | \quad \langle \text{Constraint} \rangle$   
 $\quad \quad \quad | \quad \langle \text{Goal} \rangle$   
 $\quad \quad \quad | \quad \langle \text{Assertion} \rangle$

$\langle \text{Super} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad : \langle \text{Exp18} \rangle$

$\langle \text{Reference} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad -> \langle \text{Exp15} \rangle$   
 $\quad \quad \quad | \quad ->> \langle \text{Exp15} \rangle$

$\langle \text{Init} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad \langle \text{InitHow} \rangle \langle \text{Exp} \rangle$

$\langle \text{InitHow} \rangle ::= =$   
 $\quad \quad \quad | \quad :=$

$\langle \text{GCard} \rangle ::= \epsilon$   
 $\quad \quad \quad | \quad \text{xor}$   
 $\quad \quad \quad | \quad \text{or}$   
 $\quad \quad \quad | \quad \text{mux}$   
 $\quad \quad \quad | \quad \text{opt}$   
 $\quad \quad \quad | \quad \langle \text{NCard} \rangle$

$$\begin{aligned}
\langle Card \rangle &::= \epsilon \\
&| ? \\
&| + \\
&| * \\
&| \langle PosInteger \rangle \\
&| \langle NCard \rangle \\
\langle NCard \rangle &::= \langle PosInteger \rangle \dots \langle ExInteger \rangle \\
\langle ExInteger \rangle &::= * \\
&| \langle PosInteger \rangle \\
\langle Name \rangle &::= \langle ListModId \rangle \\
\langle Exp \rangle &::= \text{all disj } \langle Decl \rangle \mid \langle Exp \rangle \\
&| \text{all } \langle Decl \rangle \mid \langle Exp \rangle \\
&| \langle Quant \rangle \text{ disj } \langle Decl \rangle \mid \langle Exp \rangle \\
&| \langle Quant \rangle \langle Decl \rangle \mid \langle Exp \rangle \\
&| \text{if } \langle Exp \rangle \text{ then } \langle Exp \rangle \text{ else } \langle Exp \rangle \\
&| \langle Exp \rangle <=> \langle Exp1 \rangle \\
&| \langle Exp1 \rangle \\
\langle Exp2 \rangle &::= \langle Exp2 \rangle => \langle Exp3 \rangle \\
&| \langle Exp3 \rangle \\
\langle Exp3 \rangle &::= \langle Exp3 \rangle || \langle Exp4 \rangle \\
&| \langle Exp4 \rangle \\
\langle Exp4 \rangle &::= \langle Exp4 \rangle \text{ xor } \langle Exp5 \rangle \\
&| \langle Exp5 \rangle \\
\langle Exp5 \rangle &::= \langle Exp5 \rangle \&\& \langle Exp6 \rangle \\
&| \langle Exp6 \rangle \\
\langle Exp6 \rangle &::= ! \langle Exp7 \rangle \\
&| \langle Exp7 \rangle \\
\langle Exp7 \rangle &::= \langle Exp7 \rangle < \langle Exp8 \rangle \\
&| \langle Exp7 \rangle > \langle Exp8 \rangle \\
&| \langle Exp7 \rangle = \langle Exp8 \rangle \\
&| \langle Exp7 \rangle <= \langle Exp8 \rangle \\
&| \langle Exp7 \rangle >= \langle Exp8 \rangle \\
&| \langle Exp7 \rangle != \langle Exp8 \rangle \\
&| \langle Exp7 \rangle \text{ in } \langle Exp8 \rangle \\
&| \langle Exp7 \rangle \text{ not in } \langle Exp8 \rangle \\
&| \langle Exp8 \rangle
\end{aligned}$$

$\langle \text{Exp8} \rangle$	$::=$	$\langle \text{Quant} \rangle \langle \text{Exp12} \rangle$   $\langle \text{Exp9} \rangle$
$\langle \text{Exp9} \rangle$	$::=$	$\langle \text{Exp9} \rangle + \langle \text{Exp10} \rangle$   $\langle \text{Exp9} \rangle - \langle \text{Exp10} \rangle$   $\langle \text{Exp10} \rangle$
$\langle \text{Exp10} \rangle$	$::=$	$\langle \text{Exp10} \rangle * \langle \text{Exp11} \rangle$   $\langle \text{Exp10} \rangle / \langle \text{Exp11} \rangle$   $\langle \text{Exp10} \rangle \% \langle \text{Exp11} \rangle$   $\langle \text{Exp11} \rangle$
$\langle \text{Exp11} \rangle$	$::=$	$\max \langle \text{Exp12} \rangle$   $\min \langle \text{Exp12} \rangle$   $\langle \text{Exp12} \rangle$
$\langle \text{Exp12} \rangle$	$::=$	$\text{sum} \langle \text{Exp13} \rangle$   $\text{product} \langle \text{Exp13} \rangle$   $\# \langle \text{Exp13} \rangle$   $- \langle \text{Exp13} \rangle$   $\langle \text{Exp13} \rangle$
$\langle \text{Exp13} \rangle$	$::=$	$\langle \text{Exp13} \rangle <: \langle \text{Exp14} \rangle$   $\langle \text{Exp14} \rangle$
$\langle \text{Exp14} \rangle$	$::=$	$\langle \text{Exp14} \rangle :> \langle \text{Exp15} \rangle$   $\langle \text{Exp15} \rangle$
$\langle \text{Exp15} \rangle$	$::=$	$\langle \text{Exp15} \rangle ++ \langle \text{Exp16} \rangle$   $\langle \text{Exp15} \rangle , \langle \text{Exp16} \rangle$   $\langle \text{Exp16} \rangle$
$\langle \text{Exp16} \rangle$	$::=$	$\langle \text{Exp16} \rangle -- \langle \text{Exp17} \rangle$   $\langle \text{Exp17} \rangle$
$\langle \text{Exp17} \rangle$	$::=$	$\langle \text{Exp17} \rangle ** \langle \text{Exp18} \rangle$   $\langle \text{Exp17} \rangle \& \langle \text{Exp18} \rangle$   $\langle \text{Exp18} \rangle$
$\langle \text{Exp18} \rangle$	$::=$	$\langle \text{Exp18} \rangle . \langle \text{Exp19} \rangle$   $\langle \text{Exp19} \rangle$
$\langle \text{Exp19} \rangle$	$::=$	$\langle \text{Name} \rangle$   $\langle \text{PosInteger} \rangle$   $\langle \text{PosDouble} \rangle$   $\langle \text{PosReal} \rangle$   $\langle \text{PosString} \rangle$   $( \langle \text{Exp} \rangle )$

$$\begin{aligned}
\langle Decl \rangle & ::= \langle ListLocId \rangle : \langle Exp15 \rangle \\
\\
\langle Quant \rangle & ::= \text{no} \\
& \quad | \quad \text{not} \\
& \quad | \quad \text{lone} \\
& \quad | \quad \text{one} \\
& \quad | \quad \text{some} \\
\\
\langle EnumId \rangle & ::= \langle PosIdent \rangle \\
\\
\langle ModId \rangle & ::= \langle PosIdent \rangle \\
\\
\langle LocId \rangle & ::= \langle PosIdent \rangle \\
\\
\langle ListDeclaration \rangle & ::= \epsilon \\
& \quad | \quad \langle Declaration \rangle \langle ListDeclaration \rangle \\
\\
\langle ListEnumId \rangle & ::= \langle EnumId \rangle \\
& \quad | \quad \langle EnumId \rangle | \langle ListEnumId \rangle \\
\\
\langle ListElement \rangle & ::= \epsilon \\
& \quad | \quad \langle Element \rangle \langle ListElement \rangle \\
\\
\langle ListExp \rangle & ::= \epsilon \\
& \quad | \quad \langle Exp \rangle \langle ListExp \rangle \\
\\
\langle ListLocId \rangle & ::= \langle LocId \rangle \\
& \quad | \quad \langle LocId \rangle ; \langle ListLocId \rangle \\
\\
\langle ListModId \rangle & ::= \langle ModId \rangle \\
& \quad | \quad \langle ModId \rangle \setminus \langle ListModId \rangle \\
\\
\langle Exp1 \rangle & ::= \langle Exp2 \rangle
\end{aligned}$$