

# The Language TplSpec

BNF-converter

September 13, 2014

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 TplSpec

### 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 `\`.

Numeral literals are recognized by the regular expression `'0' | ["123456789"]["0123456789"]*`

Rational literals are recognized by the regular expression `('0' | ["123456789"]["0123456789"]*)'.'["0123456789"]*`

Hexadecimal literals are recognized by the regular expression `{"#h"}["0123456789ABCDEF"]+`

Binary literals are recognized by the regular expression `{"#b"}["01"]+`

NormalSymbolT literals are recognized by the regular expression `(["+ - / * = ~"] |  $\langle letter \rangle$ )(["+ - / * = ~"] |  $\langle letter \rangle$  |  $\langle digit \rangle$ )*`

QuotedSymbolT literals are recognized by the regular expression `'|'["^ ~"]*'|'`

AnnotAttribute literals are recognized by the regular expression `':'(["+ - / * = ~"] |  $\langle letter \rangle$ )(["+ - / * = ~"] |  $\langle letter \rangle$  |  $\langle digit \rangle$ )*`

### 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 TplSpec are the following:

```
as      exists    forall
let     predicate  templates
term
```

The symbols used in TplSpec are the following:

```
(                )                predicate-2
inequality-term  inequality-term-2  !
-
```

## Comments

Single-line comments begin with ;.

There are no multiple-line comments in the grammar.

## The syntactic structure of TplSpec

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.

$$\langle SpecC \rangle ::= \langle ListTemplatesC \rangle$$

$$\langle TemplatesC \rangle ::= ( \text{templates} ( \langle SymbolRef \rangle \langle ListSymbol \rangle ) \langle ListTemplateC \rangle )$$

$$\langle ListTemplatesC \rangle ::= \epsilon$$

$$| \quad \langle TemplatesC \rangle \langle ListTemplatesC \rangle$$

$$\langle TemplateC \rangle ::= ( \langle TemplateType \rangle \langle Term \rangle \langle Integer \rangle )$$

$$\langle ListTemplateC \rangle ::= \epsilon$$

$$| \quad \langle TemplateC \rangle \langle ListTemplateC \rangle$$

$$\langle TemplateType \rangle ::= \text{predicate}$$

$$| \quad \text{predicate-2}$$

$$| \quad \text{term}$$

$$| \quad \text{inequality-term}$$

$$| \quad \text{inequality-term-2}$$

$$\langle Sort \rangle ::= \langle Identifier \rangle$$

$$| \quad ( \langle Identifier \rangle \langle ListSort \rangle )$$

$$\begin{aligned}
\langle \text{ListSort} \rangle & ::= \langle \text{Sort} \rangle \\
& \quad | \quad \langle \text{Sort} \rangle \langle \text{ListSort} \rangle \\
\langle \text{Term} \rangle & ::= \langle \text{SpecConstant} \rangle \\
& \quad | \quad \langle \text{SymbolRef} \rangle \\
& \quad | \quad ( \langle \text{SymbolRef} \rangle \langle \text{ListTerm} \rangle ) \\
& \quad | \quad ( \text{let} ( \langle \text{ListBindingC} \rangle ) \langle \text{Term} \rangle ) \\
& \quad | \quad ( \langle \text{Quantifier} \rangle ( \langle \text{ListSortedVariableC} \rangle ) \langle \text{Term} \rangle ) \\
& \quad | \quad ( ! \langle \text{Term} \rangle \langle \text{ListAnnotation} \rangle ) \\
\langle \text{ListTerm} \rangle & ::= \langle \text{Term} \rangle \\
& \quad | \quad \langle \text{Term} \rangle \langle \text{ListTerm} \rangle \\
\langle \text{BindingC} \rangle & ::= ( \langle \text{Symbol} \rangle \langle \text{Term} \rangle ) \\
\langle \text{ListBindingC} \rangle & ::= \langle \text{BindingC} \rangle \\
& \quad | \quad \langle \text{BindingC} \rangle \langle \text{ListBindingC} \rangle \\
\langle \text{Quantifier} \rangle & ::= \text{forall} \\
& \quad | \quad \text{exists} \\
\langle \text{SymbolRef} \rangle & ::= \langle \text{Identifier} \rangle \\
& \quad | \quad ( \text{as} \langle \text{Identifier} \rangle \langle \text{Sort} \rangle ) \\
\langle \text{SortedVariableC} \rangle & ::= ( \langle \text{Symbol} \rangle \langle \text{Sort} \rangle ) \\
\langle \text{ListSortedVariableC} \rangle & ::= \langle \text{SortedVariableC} \rangle \\
& \quad | \quad \langle \text{SortedVariableC} \rangle \langle \text{ListSortedVariableC} \rangle \\
\langle \text{SpecConstant} \rangle & ::= \langle \text{Numeral} \rangle \\
& \quad | \quad \langle \text{Rational} \rangle \\
& \quad | \quad \langle \text{Hexadecimal} \rangle \\
& \quad | \quad \langle \text{Binary} \rangle \\
& \quad | \quad \langle \text{String} \rangle \\
\langle \text{Identifier} \rangle & ::= \langle \text{Symbol} \rangle \\
& \quad | \quad ( - \langle \text{Symbol} \rangle \langle \text{ListIndexC} \rangle ) \\
\langle \text{IndexC} \rangle & ::= \langle \text{Numeral} \rangle \\
\langle \text{ListIndexC} \rangle & ::= \langle \text{IndexC} \rangle \\
& \quad | \quad \langle \text{IndexC} \rangle \langle \text{ListIndexC} \rangle \\
\langle \text{Symbol} \rangle & ::= \langle \text{NormalSymbolT} \rangle \\
& \quad | \quad \langle \text{QuotedSymbolT} \rangle \\
\langle \text{ListSymbol} \rangle & ::= \epsilon \\
& \quad | \quad \langle \text{Symbol} \rangle \langle \text{ListSymbol} \rangle
\end{aligned}$$

$$\begin{aligned}
\langle Annotation \rangle & ::= \langle AnnotAttribute \rangle \langle AttrParam \rangle \\
\langle ListAnnotation \rangle & ::= \langle Annotation \rangle \\
& \quad | \quad \langle Annotation \rangle \langle ListAnnotation \rangle \\
\langle AttrParam \rangle & ::= \langle SExpr \rangle \\
& \quad | \quad \epsilon \\
\langle SExpr \rangle & ::= \langle SpecConstant \rangle \\
& \quad | \quad \langle Symbol \rangle \\
& \quad | \quad ( \langle ListSExpr \rangle ) \\
\langle ListSExpr \rangle & ::= \epsilon \\
& \quad | \quad \langle SExpr \rangle \langle ListSExpr \rangle
\end{aligned}$$