The Princess Input Language (ApInput)

Philipp Rümmer

Department of Information Technology, Uppsala University, Sweden

December 1, 2017

This document was automatically generated by the *BNF-Converter*, with some manual modifications. 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.

The lexical structure of ApInput

Identifiers

Identifiers ⟨*Ident*⟩ are unquoted strings beginning with a letter, followed by any combination of letters, digits, and the characters _ ', reserved words excluded.

Literals

IntLit literals are recognized by the regular expression ⟨*digit*⟩+

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 ApInput are the following:

```
bool   bv      false
inf    int     mod
nat    signed  true
```

The symbols used in ApInput are the following:

```
\problem              {                     }
\functions            \universalConstants   \predicates
\interpolant          ;                     \existentialConstants
\metaVariables        <->                   ->
<-                    |                     &
!                     \abs                  \max
\min                  \distinct             \as
\eps                  \part
[                     ]                     +
-                     *                     /
%                     ^                     \if
(                     )                     \then
\else
\forall               \exists               =
!=                    <=                    >=
<                     >                     ,
\partial              \relational           \sorts
\size                 \negMatch             \noMatch
.
```

Comments

Single-line comments begin with `//`.
Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of ApInput

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

⟨*Entry*⟩   ::=   ⟨*API*⟩
         | ⟨*Expression*⟩

⟨*API*⟩   ::=   ⟨*ListBlock*⟩

⟨*ListBlock*⟩   ::= ϵ
              | ⟨*Block*⟩ ⟨*ListBlock*⟩

$$
\begin{array}{lll}
\langle Block\rangle & ::= & \texttt{\textbackslash problem} \ \{ \ \langle Expression\rangle \ \} \\
& | & \texttt{\textbackslash sorts} \ \{ \ \langle ListDeclSortC\rangle \ \} \\
& | & \texttt{\textbackslash functions} \ \{ \ \langle ListDeclFunC\rangle \ \} \\
& | & \langle ExConstantsSec\rangle \ \{ \ \langle ListDeclConstantC\rangle \ \} \\
& | & \texttt{\textbackslash universalConstants} \ \{ \ \langle ListDeclConstantC\rangle \ \} \\
& | & \texttt{\textbackslash predicates} \ \{ \ \langle ListDeclPredC\rangle \ \} \\
& | & \texttt{\textbackslash interpolant} \ \{ \ \langle ListIdent\rangle \ ; \ \langle ListIdent\rangle \ \} \\[6pt]
\langle ExConstantsSec\rangle & ::= & \texttt{\textbackslash existentialConstants} \\
& | & \texttt{\textbackslash metaVariables} \\[6pt]
\langle Expression\rangle & ::= & \langle Expression\rangle \ \texttt{<->} \ \langle Expression1\rangle \\
& | & \langle Expression1\rangle \\[6pt]
\langle Expression1\rangle & ::= & \langle Expression2\rangle \ \texttt{->} \ \langle Expression1\rangle \\
& | & \langle Expression1\rangle \ \texttt{<-} \ \langle Expression2\rangle \\
& | & \langle Expression2\rangle \\[6pt]
\langle Expression2\rangle & ::= & \langle Expression2\rangle \ \texttt{|} \ \langle Expression3\rangle \\
& | & \langle Expression3\rangle \\[6pt]
\langle Expression3\rangle & ::= & \langle Expression3\rangle \ \texttt{\&} \ \langle Expression4\rangle \\
& | & \langle Expression4\rangle \\[6pt]
\langle Expression4\rangle & ::= & \texttt{!} \ \langle Expression4\rangle \\
& | & \langle Quant\rangle \ \langle DeclBinder\rangle \ \langle Expression4\rangle \\
& | & \texttt{\textbackslash eps} \ \langle DeclSingleVarC\rangle \ ; \ \langle Expression4\rangle \\
& | & \{ \ \langle ListArgC\rangle \ \} \ \langle Expression4\rangle \\
& | & \texttt{\textbackslash part} \ \texttt{[} \ \langle Ident\rangle \ \texttt{]} \ \langle Expression4\rangle \\
& | & \langle Expression5\rangle \\[6pt]
\langle Expression5\rangle & ::= & \langle Expression6\rangle \ \langle RelSym\rangle \ \langle Expression6\rangle \\
& | & \langle Expression6\rangle \\[6pt]
\langle Expression6\rangle & ::= & \langle Expression6\rangle \ \texttt{+} \ \langle Expression7\rangle \\
& | & \langle Expression6\rangle \ \texttt{-} \ \langle Expression7\rangle \\
& | & \langle Expression7\rangle \\[6pt]
\langle Expression7\rangle & ::= & \langle Expression7\rangle \ \texttt{*} \ \langle Expression8\rangle \\
& | & \langle Expression7\rangle \ \texttt{/} \ \langle Expression8\rangle \\
& | & \langle Expression7\rangle \ \texttt{\%} \ \langle Expression8\rangle \\
& | & \langle Expression8\rangle \\[6pt]
\langle Expression8\rangle & ::= & \texttt{\textbackslash as} \ \texttt{[} \ \langle Type\rangle \ \texttt{]} \ \langle Expression8\rangle \\
\langle Expression8\rangle & ::= & \texttt{+} \ \langle Expression9\rangle \\
& | & \texttt{-} \ \langle Expression9\rangle \\
& | & \langle Expression10\rangle \ \texttt{.} \ \texttt{\textbackslash as} \ \texttt{[} \ \langle Type\rangle \ \texttt{]} \\
& | & \langle Expression9\rangle \\
\end{array}
$$

$\langle Expression9 \rangle$ ::= $\langle Expression9 \rangle$ `^` $\langle Expression10 \rangle$
         | $\langle Expression10 \rangle$

$\langle Expression10 \rangle$ ::=   `\if (` $\langle Expression \rangle$ `)` `\then (` $\langle Expression \rangle$ `)` `\else (` $\langle Expression \rangle$ `)`
         | `\abs (` $\langle Expression \rangle$ `)`
         | `\max` $\langle OptArgs \rangle$
         | `\min` $\langle OptArgs \rangle$
         | `\distinct` $\langle OptArgs \rangle$
         | `\size (` $\langle Expression \rangle$ `)`
         | $\langle Ident \rangle$ $\langle OptArgs \rangle$
         | $\langle Expression10 \rangle$ `.` $\langle Ident \rangle$
         | `true`
         | `false`
         | $\langle IntLit \rangle$
         | `(` $\langle Expression \rangle$ `)`

$\langle Quant \rangle$ ::=   `\forall`
        | `\exists`

$\langle RelSym \rangle$ ::=   `=`
        | `!=`
        | `<=`
        | `>=`
        | `<`
        | `>`

$\langle OptArgs \rangle$ ::= ϵ
        | `(` $\langle ListArgC \rangle$ `)`

$\langle ArgC \rangle$ ::= $\langle Expression \rangle$

$\langle ListArgC \rangle$ ::= ϵ
        | $\langle ArgC \rangle$
        | $\langle ArgC \rangle$ `,` $\langle ListArgC \rangle$

$\langle DeclConstC \rangle$ ::= $\langle Type \rangle$ $\langle ListIdent \rangle$

$\langle ListIdent \rangle$ ::= $\langle Ident \rangle$
        | $\langle Ident \rangle$ `,` $\langle ListIdent \rangle$

$\langle DeclSingleVarC \rangle$ ::= $\langle Type \rangle$ $\langle Ident \rangle$

$\langle DeclVarC \rangle$ ::= $\langle Type \rangle$ $\langle ListIdent \rangle$

$\langle DeclBinder \rangle$ ::= $\langle DeclVarC \rangle$ `;`
        | `(` $\langle ListDeclVarC \rangle$ `)`

$$\langle ListDeclVarC \rangle \quad ::= \quad \langle DeclVarC \rangle$$
$$| \quad \langle DeclVarC \rangle \; ; \; \langle ListDeclVarC \rangle$$

$$\langle DeclFunC \rangle \quad ::= \quad \langle ListFunOption \rangle \; \langle DeclConstC \rangle$$
$$| \quad \langle ListFunOption \rangle \; \langle Type \rangle \; \langle Ident \rangle \; \langle FormalArgsC \rangle \; \langle OptBody \rangle$$

$$\langle ListDeclFunC \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle DeclFunC \rangle \; ; \; \langle ListDeclFunC \rangle$$

$$\langle FunOption \rangle \quad ::= \quad \texttt{\textbackslash partial}$$
$$| \quad \texttt{\textbackslash relational}$$

$$\langle ListFunOption \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle FunOption \rangle \; \langle ListFunOption \rangle$$

$$\langle DeclSortC \rangle \quad ::= \quad \langle Ident \rangle \; \{ \; \langle ListDeclCtorC \rangle \; \}$$
$$| \quad \langle Ident \rangle$$

$$\langle ListDeclSortC \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle DeclSortC \rangle \; ; \; \langle ListDeclSortC \rangle$$

$$\langle DeclCtorC \rangle \quad ::= \quad \langle Ident \rangle \; \langle OptFormalArgs \rangle$$

$$\langle ListDeclCtorC \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle DeclCtorC \rangle \; ; \; \langle ListDeclCtorC \rangle$$

$$\langle DeclConstantC \rangle \quad ::= \quad \langle DeclConstC \rangle$$

$$\langle ListDeclConstantC \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle DeclConstantC \rangle \; ; \; \langle ListDeclConstantC \rangle$$

$$\langle DeclPredC \rangle \quad ::= \quad \langle ListPredOption \rangle \; \langle Ident \rangle \; \langle OptFormalArgs \rangle \; \langle OptBody \rangle$$

$$\langle ListDeclPredC \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle DeclPredC \rangle \; ; \; \langle ListDeclPredC \rangle$$

$$\langle OptFormalArgs \rangle \quad ::= \quad \epsilon$$
$$| \quad \langle FormalArgsC \rangle$$

$$\langle FormalArgsC \rangle \quad ::= \quad (\; \langle ListArgTypeC \rangle \;)$$

$$\langle ArgTypeC \rangle \quad ::= \quad \langle Type \rangle$$
$$| \quad \langle Type \rangle \; \langle Ident \rangle$$

$$\langle ListArgTypeC \rangle \quad ::= \quad \langle ArgTypeC \rangle$$
$$| \quad \langle ArgTypeC \rangle \; , \; \langle ListArgTypeC \rangle$$

$$\langle PredOption \rangle \quad ::= \quad \texttt{\textbackslash negMatch}$$
$$| \quad \texttt{\textbackslash noMatch}$$

$\langle ListPredOption \rangle \quad ::= \quad \epsilon$
$\qquad\qquad\qquad\quad | \qquad \langle PredOption \rangle \, \langle ListPredOption \rangle$

$\langle OptBody \rangle \quad ::= \quad \{ \; \langle Expression \rangle \; \}$
$\qquad\qquad\qquad | \qquad \epsilon$

$\langle Type \rangle \quad ::= \quad$ `int`
$\qquad\qquad\quad | \qquad$ `nat`
$\qquad\qquad\quad | \qquad$ `int [` $\langle IntervalLower \rangle$ `,` $\langle IntervalUpper \rangle$ `]`
$\qquad\qquad\quad | \qquad$ `bool`
$\qquad\qquad\quad | \qquad$ `mod [` $\langle IntervalLower \rangle$ `,` $\langle IntervalUpper \rangle$ `]`
$\qquad\qquad\quad | \qquad$ `bv [` $\langle IntLit \rangle$ `]`
$\qquad\qquad\quad | \qquad$ `signed bv [` $\langle IntLit \rangle$ `]`
$\qquad\qquad\quad | \qquad \langle Ident \rangle$

$\langle IntervalLower \rangle \quad ::= \quad -$ `inf`
$\qquad\qquad\qquad\quad | \qquad \langle IntLit \rangle$
$\qquad\qquad\qquad\quad | \qquad - \langle IntLit \rangle$

$\langle IntervalUpper \rangle \quad ::= \quad$ `inf`
$\qquad\qquad\qquad\quad | \qquad \langle IntLit \rangle$
$\qquad\qquad\qquad\quad | \qquad - \langle IntLit \rangle$