

# The Princess Input Language (ApInput)

Philipp Rümmer

Department of Information Technology, Uppsala University, Sweden

July 18, 2013

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  $\langle Ident \rangle$  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  $\langle digit \rangle^+$

### 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 ApInput are the following:

```
false  inf   int
nat     true
```

The symbols used in ApInput are the following:

<code>\problem</code>	<code>{</code>	<code>}</code>
<code>\functions</code>	<code>\universalConstants</code>	<code>\predicates</code>
<code>\interpolant</code>	<code>;</code>	<code>\existentialConstants</code>
<code>\metaVariables</code>	<code>&lt;-&gt;</code>	<code>-&gt;</code>
<code> </code>	<code>&amp;</code>	<code>!</code>
<code>\eps</code>	<code>\part</code>	<code>[</code>
<code>]</code>	<code>+</code>	<code>-</code>
<code>*</code>	<code>\if</code>	<code>(</code>
<code>)</code>	<code>\then</code>	<code>\else</code>
<code>\forall</code>	<code>\exists</code>	<code>=</code>
<code>!=</code>	<code>&lt;=</code>	<code>&gt;=</code>
<code>&lt;</code>	<code>&gt;</code>	<code>,</code>
<code>\partial</code>	<code>\relational</code>	

## Comments

Single-line comments begin with `//`.

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

## The syntactic structure of ApInput

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.

$$\begin{aligned} \langle Entry \rangle & ::= \langle API \rangle \\ & | \langle Expression \rangle \end{aligned}$$

$$\langle API \rangle ::= \langle ListBlock \rangle$$

$$\begin{aligned} \langle ListBlock \rangle & ::= \epsilon \\ & | \langle Block \rangle \langle ListBlock \rangle \end{aligned}$$

$$\begin{aligned} \langle Block \rangle & ::= \backslash problem \{ \langle Expression \rangle \} \\ & | \backslash functions \{ \langle ListDeclFunC \rangle \} \\ & | \langle ExConstantsSec \rangle \{ \langle ListDeclConstantC \rangle \} \\ & | \backslash universalConstants \{ \langle ListDeclConstantC \rangle \} \\ & | \backslash predicates \{ \langle ListDeclPredC \rangle \} \\ & | \backslash interpolant \{ \langle ListIdent \rangle ; \langle ListIdent \rangle \} \end{aligned}$$

$$\begin{aligned}
\langle ExConstantsSec \rangle & ::= \quad \backslash \text{existentialConstants} \\
& \quad | \quad \backslash \text{metaVariables} \\
\langle Expression \rangle & ::= \quad \langle Expression \rangle <-> \langle Expression1 \rangle \\
& \quad | \quad \langle Expression1 \rangle \\
\langle Expression1 \rangle & ::= \quad \langle Expression2 \rangle -> \langle Expression1 \rangle \\
& \quad | \quad \langle Expression2 \rangle \\
\langle Expression2 \rangle & ::= \quad \langle Expression2 \rangle | \langle Expression3 \rangle \\
& \quad | \quad \langle Expression3 \rangle \\
\langle Expression3 \rangle & ::= \quad \langle Expression3 \rangle \& \langle Expression4 \rangle \\
& \quad | \quad \langle Expression4 \rangle \\
\langle Expression4 \rangle & ::= \quad ! \langle Expression4 \rangle \\
& \quad | \quad \langle Quant \rangle \langle DeclBinder \rangle \langle Expression4 \rangle \\
& \quad | \quad \backslash \text{eps} \langle DeclSingleVarC \rangle ; \langle Expression4 \rangle \\
& \quad | \quad \{ \langle ListArgC \rangle \} \langle Expression4 \rangle \\
& \quad | \quad \backslash \text{part} [ \langle Ident \rangle ] \langle Expression4 \rangle \\
& \quad | \quad \langle Expression5 \rangle \\
\langle Expression5 \rangle & ::= \quad \text{true} \\
& \quad | \quad \text{false} \\
& \quad | \quad \langle Expression6 \rangle \langle RelSym \rangle \langle Expression6 \rangle \\
& \quad | \quad \langle Expression6 \rangle \\
\langle Expression6 \rangle & ::= \quad \langle Expression6 \rangle + \langle Expression7 \rangle \\
& \quad | \quad \langle Expression6 \rangle - \langle Expression7 \rangle \\
& \quad | \quad \langle Expression7 \rangle \\
\langle Expression7 \rangle & ::= \quad \langle Expression7 \rangle * \langle Expression8 \rangle \\
& \quad | \quad \langle Expression8 \rangle \\
\langle Expression8 \rangle & ::= \quad + \langle Expression8 \rangle \\
& \quad | \quad - \langle Expression8 \rangle \\
& \quad | \quad \backslash \text{if} ( \langle Expression \rangle ) \backslash \text{then} ( \langle Expression \rangle ) \backslash \text{else} ( \langle Expression \rangle ) \\
& \quad | \quad \langle Expression9 \rangle \\
\langle Expression9 \rangle & ::= \quad \langle Ident \rangle \langle OptArgs \rangle \\
& \quad | \quad \langle IntLit \rangle \\
& \quad | \quad ( \langle Expression \rangle ) \\
\langle Quant \rangle & ::= \quad \backslash \text{forall} \\
& \quad | \quad \backslash \text{exists}
\end{aligned}$$

$$\begin{aligned}
\langle \text{RelSym} \rangle &::= = \\
&| \neq \\
&| \leq \\
&| \geq \\
&| < \\
&| > \\
\langle \text{OptArgs} \rangle &::= \epsilon \\
&| ( \langle \text{ListArgC} \rangle ) \\
\langle \text{ArgC} \rangle &::= \langle \text{Expression} \rangle \\
\langle \text{ListArgC} \rangle &::= \epsilon \\
&| \langle \text{ArgC} \rangle \\
&| \langle \text{ArgC} \rangle , \langle \text{ListArgC} \rangle \\
\langle \text{DeclConstC} \rangle &::= \langle \text{Type} \rangle \langle \text{ListIdent} \rangle \\
\langle \text{ListIdent} \rangle &::= \langle \text{Ident} \rangle \\
&| \langle \text{Ident} \rangle , \langle \text{ListIdent} \rangle \\
\langle \text{DeclSingleVarC} \rangle &::= \langle \text{BinderType} \rangle \langle \text{Ident} \rangle \\
\langle \text{DeclVarC} \rangle &::= \langle \text{BinderType} \rangle \langle \text{ListIdent} \rangle \\
\langle \text{DeclBinder} \rangle &::= \langle \text{DeclVarC} \rangle ; \\
&| ( \langle \text{ListDeclVarC} \rangle ) \\
\langle \text{ListDeclVarC} \rangle &::= \langle \text{DeclVarC} \rangle \\
&| \langle \text{DeclVarC} \rangle ; \langle \text{ListDeclVarC} \rangle \\
\langle \text{DeclFunC} \rangle &::= \langle \text{ListFunOption} \rangle \langle \text{DeclConstC} \rangle \\
&| \langle \text{ListFunOption} \rangle \langle \text{Type} \rangle \langle \text{Ident} \rangle \langle \text{FormalArgsC} \rangle \\
\langle \text{ListDeclFunC} \rangle &::= \epsilon \\
&| \langle \text{DeclFunC} \rangle ; \langle \text{ListDeclFunC} \rangle \\
\langle \text{FunOption} \rangle &::= \backslash \text{partial} \\
&| \backslash \text{relational} \\
\langle \text{ListFunOption} \rangle &::= \epsilon \\
&| \langle \text{FunOption} \rangle \langle \text{ListFunOption} \rangle \\
\langle \text{DeclConstantC} \rangle &::= \langle \text{DeclConstC} \rangle \\
\langle \text{ListDeclConstantC} \rangle &::= \epsilon \\
&| \langle \text{DeclConstantC} \rangle ; \langle \text{ListDeclConstantC} \rangle
\end{aligned}$$

$$\begin{aligned}
\langle \text{DeclPredC} \rangle &::= \langle \text{ListPredOption} \rangle \langle \text{Ident} \rangle \langle \text{OptFormalArgs} \rangle \\
\langle \text{ListDeclPredC} \rangle &::= \epsilon \\
&| \quad \langle \text{DeclPredC} \rangle ; \langle \text{ListDeclPredC} \rangle \\
\langle \text{OptFormalArgs} \rangle &::= \epsilon \\
&| \quad \langle \text{FormalArgsC} \rangle \\
\langle \text{FormalArgsC} \rangle &::= ( \langle \text{ListArgTypeC} \rangle ) \\
\langle \text{ArgTypeC} \rangle &::= \langle \text{Type} \rangle \\
\langle \text{ListArgTypeC} \rangle &::= \langle \text{ArgTypeC} \rangle \\
&| \quad \langle \text{ArgTypeC} \rangle , \langle \text{ListArgTypeC} \rangle \\
\langle \text{PredOption} \rangle &::= \backslash \text{negMatch} \\
&| \quad \backslash \text{noMatch} \\
\langle \text{ListPredOption} \rangle &::= \epsilon \\
&| \quad \langle \text{PredOption} \rangle \langle \text{ListPredOption} \rangle \\
\langle \text{Type} \rangle &::= \text{int} \\
\langle \text{BinderType} \rangle &::= \langle \text{Type} \rangle \\
&| \quad \text{nat} \\
&| \quad \text{int } [ \langle \text{IntervalLower} \rangle , \langle \text{IntervalUpper} \rangle ] \\
\langle \text{IntervalLower} \rangle &::= - \text{inf} \\
&| \quad \langle \text{IntLit} \rangle \\
&| \quad - \langle \text{IntLit} \rangle \\
\langle \text{IntervalUpper} \rangle &::= \text{inf} \\
&| \quad \langle \text{IntLit} \rangle \\
&| \quad - \langle \text{IntLit} \rangle
\end{aligned}$$