The Princess Input Language (ApInput)

Philipp Rümmer
Department of Information Technology, Uppsala University, Sweden

October 16, 2011

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:

```
false  int  true
```

The symbols used in ApInput are the following:

`\problem`	{	}
`\functions`	`\universalConstants`	`\predicates`
`\interpolant`	;	`\existentialConstants`
`\metaVariables`	<−>	−>
|	&	!
`\eps`	`\part`	[
]	+	−
*	`\if`	(
)	`\then`	`\else`
`\forall`	`\exists`	=
!=	<=	>=
<	>	,
`\partial`	`\relational`	

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*⟩

⟨*Block*⟩ ::= `\problem` { ⟨*Expression*⟩ }
| `\functions` { ⟨*ListDeclFunC*⟩ }
| ⟨*ExConstantsSec*⟩ { ⟨*ListDeclConstantC*⟩ }
| `\universalConstants` { ⟨*ListDeclConstantC*⟩ }
| `\predicates` { ⟨*ListDeclPredC*⟩ }
| `\interpolant` { ⟨*ListIdent*⟩ ; ⟨*ListIdent*⟩ }

⟨*ExConstantsSec*⟩ ::= `\existentialConstants`
| `\metaVariables`

⟨*Expression*⟩ ::= ⟨*Expression*⟩ `<->` ⟨*Expression1*⟩
| ⟨*Expression1*⟩

⟨*Expression1*⟩ ::= ⟨*Expression2*⟩ `->` ⟨*Expression1*⟩
| ⟨*Expression2*⟩

⟨*Expression2*⟩ ::= ⟨*Expression2*⟩ `|` ⟨*Expression3*⟩
| ⟨*Expression3*⟩

⟨*Expression3*⟩ ::= ⟨*Expression3*⟩ `&` ⟨*Expression4*⟩
| ⟨*Expression4*⟩

⟨*Expression4*⟩ ::= `!` ⟨*Expression4*⟩
| ⟨*Quant*⟩ ⟨*DeclBinder*⟩ ⟨*Expression4*⟩
| `\eps` ⟨*DeclSingleVarC*⟩ `;` ⟨*Expression4*⟩
| `{` ⟨*ListArgC*⟩ `}` ⟨*Expression4*⟩
| `\part [` ⟨*Ident*⟩ `]` ⟨*Expression4*⟩
| ⟨*Expression5*⟩

⟨*Expression5*⟩ ::= `true`
| `false`
| ⟨*Expression6*⟩ ⟨*RelSym*⟩ ⟨*Expression6*⟩
| ⟨*Expression6*⟩

⟨*Expression6*⟩ ::= ⟨*Expression6*⟩ `+` ⟨*Expression7*⟩
| ⟨*Expression6*⟩ `-` ⟨*Expression7*⟩
| ⟨*Expression7*⟩

⟨*Expression7*⟩ ::= ⟨*Expression7*⟩ `*` ⟨*Expression8*⟩
| ⟨*Expression8*⟩

⟨*Expression8*⟩ ::= `+` ⟨*Expression8*⟩
| `-` ⟨*Expression8*⟩
| `\if (` ⟨*Expression*⟩ `) \then (` ⟨*Expression*⟩ `) \else (` ⟨*Expression*⟩ `)`
| ⟨*Expression9*⟩

⟨*Expression9*⟩ ::= ⟨*Ident*⟩ ⟨*OptArgs*⟩
| ⟨*IntLit*⟩
| `(` ⟨*Expression*⟩ `)`

⟨*Quant*⟩ ::= `\forall`
| `\exists`

⟨*RelSym*⟩ ::= =
| `!=`
| `<=`
| `>=`
| `<`
| `>`

⟨*OptArgs*⟩ ::= ϵ
| `(` ⟨*ListArgC*⟩ `)`

⟨*ArgC*⟩ ::= ⟨*Expression*⟩

⟨*ListArgC*⟩ ::= ϵ
| ⟨*ArgC*⟩
| ⟨*ArgC*⟩ `,` ⟨*ListArgC*⟩

⟨*DeclSingleVarC*⟩ ::= ⟨*Type*⟩ ⟨*Ident*⟩

⟨*DeclVarConstC*⟩ ::= ⟨*Type*⟩ ⟨*ListIdent*⟩

⟨*ListIdent*⟩ ::= ⟨*Ident*⟩
| ⟨*Ident*⟩ `,` ⟨*ListIdent*⟩

⟨*DeclBinder*⟩ ::= ⟨*DeclVarConstC*⟩ `;`
| `(` ⟨*ListDeclVarConstC*⟩ `)`

⟨*ListDeclVarConstC*⟩ ::= ⟨*DeclVarConstC*⟩
| ⟨*DeclVarConstC*⟩ `;` ⟨*ListDeclVarConstC*⟩

⟨*DeclFunC*⟩ ::= ⟨*ListFunOption*⟩ ⟨*DeclVarConstC*⟩
| ⟨*ListFunOption*⟩ ⟨*Type*⟩ ⟨*Ident*⟩ ⟨*FormalArgsC*⟩

⟨*ListDeclFunC*⟩ ::= ϵ
| ⟨*DeclFunC*⟩ `;` ⟨*ListDeclFunC*⟩

⟨*FunOption*⟩ ::= `\partial`
| `\relational`

⟨*ListFunOption*⟩ ::= ϵ
| ⟨*FunOption*⟩ ⟨*ListFunOption*⟩

⟨*DeclConstantC*⟩ ::= ⟨*DeclVarConstC*⟩

⟨*ListDeclConstantC*⟩ ::= ϵ
| ⟨*DeclConstantC*⟩ `;` ⟨*ListDeclConstantC*⟩

⟨*DeclPredC*⟩ ::= ⟨*Ident*⟩ ⟨*OptFormalArgs*⟩

⟨*ListDeclPredC*⟩ ::= ϵ
| ⟨*DeclPredC*⟩ `;` ⟨*ListDeclPredC*⟩

⟨*OptFormalArgs*⟩ ::= ϵ
| ⟨*FormalArgsC*⟩

⟨*FormalArgsC*⟩ ::= `(` ⟨*ListArgTypeC*⟩ `)`

⟨*ArgTypeC*⟩ ::= ⟨*Type*⟩

⟨*ListArgTypeC*⟩ ::= ⟨*ArgTypeC*⟩
| ⟨*ArgTypeC*⟩ `,` ⟨*ListArgTypeC*⟩

⟨*Type*⟩ ::= `int`