The Language PML

BNF-converter

March 13, 2017

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 PML

Literals

STRING literals are recognized by the regular expression '"'(⟨*anychar*⟩ − ["""]) ∗ '"'

ID literals are recognized by the regular expression (⟨*letter*⟩ | '_')(⟨*letter*⟩ | ⟨*digit*⟩ | '_')∗

NUMBER literals are recognized by the regular expression (⟨*digit*⟩+('.'⟨*digit*⟩+)? | '.'⟨*digit*⟩+)(('e' | 'E')('+' | '−')?⟨*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 PML are the following:

```
action      agent      branch
executable  iteration  manual
process     provides   requires
script      selection  sequence
task        tool
```

The symbols used in PML are the following:

```
{     }     ||
&&    ==    !=
<     >     <=
>=    !     (
)     .
```

Comments

There are no single-line comments in the grammar.
Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of PML

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

⟨*PROCESS*⟩ ::= `process` ⟨*ID*⟩ { ⟨*ListPRIM*⟩ }

⟨*ListPRIM*⟩ ::= ϵ
| ⟨*PRIM*⟩ ⟨*ListPRIM*⟩

⟨*PRIM*⟩ ::= `branch` ⟨*OPTNM*⟩ { ⟨*ListPRIM*⟩ }
| `selection` ⟨*OPTNM*⟩ { ⟨*ListPRIM*⟩ }
| `iteration` ⟨*OPTNM*⟩ { ⟨*ListPRIM*⟩ }
| `sequence` ⟨*OPTNM*⟩ { ⟨*ListPRIM*⟩ }
| `task` ⟨*OPTNM*⟩ { ⟨*ListPRIM*⟩ }
| `action` ⟨*ID*⟩ ⟨*OPTYP*⟩ { ⟨*ListSPEC*⟩ }

⟨*OPTNM*⟩ ::= ϵ
| ⟨*ID*⟩

⟨*OPTYP*⟩ ::= ϵ
| `manual`
| `executable`

⟨*ListSPEC*⟩ ::= ϵ
| ⟨*SPEC*⟩ ⟨*ListSPEC*⟩

⟨*SPEC*⟩ ::= `provides` { ⟨*EXPR*⟩ }
| `requires` { ⟨*EXPR*⟩ }
| `agent` { ⟨*EXPR*⟩ }
| `script` { ⟨*STRING*⟩ }
| `tool` { ⟨*STRING*⟩ }

⟨*EXPR*⟩ ::= ⟨*EXPR2*⟩

⟨*EXPR2*⟩ ::= ⟨*EXPR3*⟩
| ⟨*EXPR2*⟩ || ⟨*EXPR3*⟩

⟨*EXPR3*⟩ ::= ⟨*EXPR4*⟩
| ⟨*EXPR3*⟩ `&&` ⟨*EXPR4*⟩

⟨*EXPR4*⟩ ::= ⟨*STRING*⟩
| ⟨*EXPR5*⟩
| ⟨*VALEXPR*⟩ == ⟨*VALEXPR*⟩
| ⟨*VALEXPR*⟩ `!=` ⟨*VALEXPR*⟩
| ⟨*VALEXPR*⟩ < ⟨*VALEXPR*⟩
| ⟨*VALEXPR*⟩ > ⟨*VALEXPR*⟩
| ⟨*VALEXPR*⟩ <= ⟨*VALEXPR*⟩
| ⟨*VALEXPR*⟩ >= ⟨*VALEXPR*⟩
| ⟨*VAREXPR*⟩ == ⟨*VAREXPR*⟩
| ⟨*VAREXPR*⟩ `!=` ⟨*VAREXPR*⟩

⟨*EXPR5*⟩ ::= ⟨*VAREXPR*⟩
| ⟨*ATTREXPR*⟩
| `!` ⟨*EXPR5*⟩
| `(` ⟨*EXPR*⟩ `)`

⟨*VAREXPR*⟩ ::= ⟨*ID*⟩
| `(` ⟨*ID*⟩ `)`
| `(` ⟨*ID*⟩ `)` ⟨*VAREXPR*⟩

⟨*ATTREXPR*⟩ ::= ⟨*VAREXPR*⟩ `.` ⟨*ID*⟩

⟨*VALEXPR*⟩ ::= ⟨*ATTREXPR*⟩
| ⟨*STRING*⟩
| ⟨*NUMBER*⟩