The Language SWL

BNF-converter

March 14, 2022

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 SWL

Literals

Integer literals ⟨*Int*⟩ are nonempty sequences of digits.

String literals ⟨*String*⟩ have the form "x", where x is any sequence of any characters except " unless preceded by \.

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

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

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

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

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

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

ValidImport literals are recognized by the regular expression '{'(⟨*letter*⟩ | ⟨*digit*⟩ | '_' | '.' | '/') ∗ '}'

Date literals are recognized by the regular expression ⟨*digit*⟩⟨*digit*⟩⟨*digit*⟩⟨*digit*⟩('−'⟨*digit*⟩⟨*digit*⟩)∗ 'T'(':' | ⟨*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 SWL are the following:

```
INCLUDE  PREFIX  include
prefix
```

The symbols used in SWL are the following:

```
;            {     }
<            -     >
=            ,     _
(            )     [
]            .     /
StreamWork:  ---   :
```

Comments

Single-line comments begin with `#`.
Multiple-line comments are enclosed with `{#` and `#}`.

The syntactic structure of SWL

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

⟨*Valide*⟩ ::= ⟨*ValidConfig*⟩
| ⟨*ValidSW*⟩

⟨*ValidSW*⟩ ::= ⟨*ListStm*⟩

⟨*Stm*⟩ ::= ⟨*Prefix*⟩ ⟨*Stringval*⟩
| ⟨*Include*⟩ ⟨*Stringval*⟩
| ⟨*DataFlow*⟩
| ⟨*Numassgn*⟩
| ⟨*Strassgn*⟩
| ⟨*SymAssgn*⟩
| ⟨*Hermt*⟩
| ⟨*Subdef*⟩

⟨*ListStm*⟩ ::= ϵ
| ⟨*Stm*⟩ ; ⟨*ListStm*⟩

⟨*Subdef*⟩ ::= ⟨*SubId*⟩ { ⟨*ListSubnet*⟩ }

⟨*Subnet*⟩ ::= ⟨*Hermt*⟩
| ⟨*DataFlow*⟩
| ⟨*ExtPortIn*⟩
| ⟨*ExtPortOut*⟩

⟨*ListSubnet*⟩ ::= ϵ
| ⟨*Subnet*⟩ ; ⟨*ListSubnet*⟩

⟨*ExtPortIn*⟩ ::= ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*Larrow*⟩ ⟨*Tab*⟩
| ⟨*Tab*⟩ ⟨*Rarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩

⟨*ExtPortOut*⟩ ::= ⟨*Tab*⟩ ⟨*Larrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*Rarrow*⟩ ⟨*Tab*⟩

⟨*Tab*⟩ ::= ⟨*Numval*⟩
| ⟨*Symval*⟩

⟨*DataFlow*⟩ ::= ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*Larrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*Rarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*LSarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*Proc*⟩ ⟨*Prt*⟩ ⟨*RSarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*DataFlow*⟩ ⟨*Prt*⟩ ⟨*Larrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*DataFlow*⟩ ⟨*Prt*⟩ ⟨*Rarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*DataFlow*⟩ ⟨*Prt*⟩ ⟨*LSarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩
| ⟨*DataFlow*⟩ ⟨*Prt*⟩ ⟨*RSarrow*⟩ ⟨*Prt*⟩ ⟨*Proc*⟩

⟨*Larrow*⟩ ::= < ⟨*TypeDef*⟩ ⟨*Buffsize*⟩ −

⟨*Rarrow*⟩ ::= − ⟨*TypeDef*⟩ ⟨*Buffsize*⟩ >

⟨*LSarrow*⟩ ::= < ⟨*TypeDef*⟩ ⟨*Buffsize*⟩ =

⟨*RSarrow*⟩ ::= = ⟨*TypeDef*⟩ ⟨*Buffsize*⟩ >

⟨*TypeDef*⟩ ::= ⟨*Symvalu*⟩
| ϵ
| ⟨*TypeDef*⟩ , ⟨*TypeDef*⟩

⟨*Buffsize*⟩ ::= ⟨*Numval*⟩
| ϵ

⟨*Hermt*⟩ ::= ⟨*Symvalu*⟩ ⟨*Comp*⟩ ⟨*ListArgument*⟩
| ⟨*Symvalu*⟩ ⟨*ListArgument*⟩

⟨*Symvalu*⟩ ::= ⟨*Symval*⟩
| _

⟨*Proc*⟩ ::= (⟨*Symvalu*⟩ ⟨*Comp*⟩ ⟨*ListArgument*⟩ ⟨*Attributes*⟩)
| (⟨*Symvalu*⟩ ⟨*Attributes*⟩)

⟨*Attributes*⟩ ::= [⟨*ListAttr*⟩]
| ϵ

⟨*Attr*⟩ ::= ⟨*Symval*⟩ = ⟨*Stringval*⟩
| ⟨*Symval*⟩ = ⟨*Numval*⟩

⟨*ListAttr*⟩ ::= ϵ
| ⟨*Attr*⟩
| ⟨*Attr*⟩ , ⟨*ListAttr*⟩

⟨*Prt*⟩ ::= ⟨*Numval*⟩
| ⟨*Numval*⟩ . ⟨*Symval*⟩
| ⟨*Symval*⟩ . ⟨*Numval*⟩
| ⟨*Symval*⟩
| ϵ

⟨*Comp*⟩ ::= ⟨*Symval*⟩
| ⟨*SubId*⟩
| ⟨*ModPath*⟩ ⟨*Symval*⟩
| ⟨*RemPath*⟩

⟨*ModPath*⟩ ::= / ⟨*Symval*⟩ /
| ⟨*Symval*⟩ /
| ⟨*ModPath*⟩ ⟨*Symval*⟩ /
| ⟨*Stringvar*⟩ /

⟨*RemPath*⟩ ::= ⟨*ValidImport*⟩ . ⟨*Symval*⟩

⟨*Argument*⟩ ::= ⟨*Stringval*⟩

⟨*ListArgument*⟩ ::= ϵ
| ⟨*Argument*⟩ ⟨*ListArgument*⟩

⟨*Numassgn*⟩ ::= ⟨*Numvar*⟩ = ⟨*Numval*⟩

⟨*Strassgn*⟩ ::= ⟨*Stringvar*⟩ = ⟨*Stringval*⟩

⟨*SymAssgn*⟩ ::= ⟨*Symvar*⟩ = ⟨*Symval*⟩

⟨*Numval*⟩ ::= ⟨*Integer*⟩
| ⟨*Numvar*⟩

⟨*Stringval*⟩ ::= ⟨*String*⟩
| ⟨*Stringvar*⟩
| ⟨*Envar*⟩

⟨*Symval*⟩ ::= ⟨*Symvar*⟩
| ⟨*Id*⟩
| ⟨*Envar*⟩

⟨*Include*⟩ ::= `include`
| `INCLUDE`

⟨*Prefix*⟩ ::= `PREFIX`
| `prefix`

⟨*ValidConfig*⟩ ::= `StreamWork:` ⟨*ListEntry*⟩
| `---` `StreamWork: {` ⟨*ListCentry*⟩ `}`

⟨*Centry*⟩ ::= ⟨*KeyVal*⟩
| ⟨*KeyName*⟩

⟨*ListCentry*⟩ ::= ϵ
| ⟨*Centry*⟩ `,` ⟨*ListCentry*⟩

⟨*Entry*⟩ ::= ⟨*KeyVal*⟩
| ⟨*KeyName*⟩

⟨*ListEntry*⟩ ::= ϵ
| ⟨*Entry*⟩ ⟨*ListEntry*⟩

⟨*KeyVal*⟩ ::= ⟨*KeyName*⟩ ⟨*Integer*⟩
| ⟨*KeyName*⟩ ⟨*String*⟩
| ⟨*KeyName*⟩ ⟨*Date*⟩

⟨*KeyName*⟩ ::= ⟨*Symval*⟩ `:`
| ⟨*ModPath*⟩ ⟨*Symval*⟩ `:`