HoCL v1.2
Syntax

J. Sérot

⟨program⟩ ::= ⟨declaration⟩* EOF

⟨declaration⟩ ::= ⟨type_decl⟩ **;**
| ⟨val_decl⟩ **;**
| ⟨node_decl⟩ **;**

⟨type_decl⟩ ::= **type** IDENT

⟨node_decl⟩ ::= **node** IDENT **in** ⟨io_decls⟩ **out** ⟨io_decls⟩ [⟨node_impl⟩]
| **graph** IDENT **in** ⟨io_decls⟩ **out** ⟨io_decls⟩ ⟨node_impl⟩

⟨node_impl⟩ ::= **fun** ⟨val_decl⟩* **end**
| **struct** ⟨struct_graph_desc⟩ **end**
| **actor** ⟨actor_desc⟩* **end**

⟨actor_desc⟩ ::= IDENT **(** ⟨impl_attr⟩*, **)**

⟨impl_attr⟩ ::= IDENT = STRING
| IDENT

⟨io_decls⟩ ::= **(** ⟨io_decl⟩*, **)**

⟨io_decl⟩ ::= IDENT **:** ⟨type_expr⟩ [⟨io_expr⟩] ⟨io_annots⟩

⟨io_expr⟩ ::= = ⟨simple_expr⟩

⟨io_annots⟩ ::= ϵ
| [⟨basic_expr⟩]
| **{** ⟨io_annot⟩*, **}**

⟨io_annot⟩ ::= IDENT = STRING

⟨simple_type_expr⟩ ::= IDENT
| TYVAR

⟨type_expr⟩ ::= ⟨simple_type_expr⟩
| ⟨simple_type_expr⟩ IDENT

⟨val_decl⟩ ::= **val** [**rec**] ⟨binding⟩$^{+}_{\textbf{and}}$

⟨binding⟩ ::= ⟨pattern⟩ = ⟨expr⟩
| ⟨binding_name⟩ ⟨fun_pattern⟩$^{+}$ = ⟨expr⟩

⟨binding_name⟩ ::= IDENT
| **(** INFIX0 **)**

⟨expr⟩ ::= ⟨simple_expr⟩
| ⟨simple_expr⟩ ⟨simple_labeled_expr⟩$^{+}$
| ⟨expr_comma_list⟩
| **fun** ⟨fun_pattern⟩$^{+}$ → ⟨expr⟩
| **let** [**rec**] ⟨binding⟩$^{+}_{\textbf{and}}$ **in** ⟨expr⟩
| **if** ⟨expr⟩ **then** ⟨expr⟩ **else** ⟨expr⟩
| ⟨expr⟩ INFIX0 ⟨expr⟩

| | ⟨expr⟩ INFIX1 ⟨expr⟩
| | ⟨expr⟩ INFIX2 ⟨expr⟩
| | ⟨expr⟩ INFIX3 ⟨expr⟩
| | ⟨expr⟩ **=** ⟨expr⟩
| | ⟨expr⟩ **::** ⟨expr⟩
| | ⟨simple_expr⟩ [⟨simple_expr⟩]
| | **match** ⟨expr⟩ **with** $\langle \text{match\_case} \rangle^{+}_{\text{BAR}}$

⟨simple_labeled_expr⟩ ::= IDENT **:** ⟨simple_expr⟩
| | ⟨simple_expr⟩
| | ~ IDENT

⟨match_case⟩ ::= ⟨pattern⟩ → ⟨expr⟩

⟨simple_expr⟩ ::= IDENT
| | **(** ⟨expr⟩ **)**
| | **( )**
| | ⟨const_expr⟩
| | [$\langle \text{expr} \rangle^{+}_{;}$]
| | [ ]
| | **'** ⟨basic_expr⟩ **'**

⟨const_expr⟩ ::= INT
| | **true**
| | **false**

⟨expr_comma_list⟩ ::= ⟨expr_comma_list⟩ **,** ⟨expr⟩
| | ⟨expr⟩ **,** ⟨expr⟩

⟨pattern⟩ ::= ⟨simple_pattern⟩
| | ⟨pattern_comma_list⟩
| | ⟨pattern⟩ **::** ⟨pattern⟩
| | [$\langle \text{simple\_pattern} \rangle^{+}_{;}$]

⟨fun_pattern⟩ ::= IDENT

⟨simple_pattern⟩ ::= IDENT
| | _
| | **(** ⟨pattern⟩ **)**
| | **( )**
| | [ ]

⟨pattern_comma_list⟩ ::= ⟨pattern_comma_list⟩ **,** ⟨pattern⟩
| | ⟨pattern⟩ **,** ⟨pattern⟩

⟨struct_graph_desc⟩ ::= $\langle \text{struct\_decl} \rangle^{*}$

⟨struct_decl⟩ ::= ⟨wire_decl⟩
| | ⟨box_decl⟩

⟨wire_decl⟩ ::= **wire** $\text{IDENT}^{*}_{,}$ **:** ⟨type_expr⟩

⟨box_decl⟩ ::= **box** IDENT **:** IDENT ⟨box_inps⟩ ⟨box_outps⟩

⟨box_inps⟩ ::= (⟨box_inp⟩$^*_,$)

⟨box_outps⟩ ::= (⟨box_outp⟩$^*_,$)

⟨box_inp⟩ ::= IDENT
| ' ⟨basic_expr⟩ '

⟨box_outp⟩ ::= IDENT

⟨basic_expr⟩ ::= IDENT
| ⟨const_expr⟩
| ⟨basic_expr⟩ INFIX1 ⟨basic_expr⟩
| ⟨basic_expr⟩ INFIX2 ⟨basic_expr⟩
| ⟨basic_expr⟩ INFIX3 ⟨basic_expr⟩
| (⟨basic_expr⟩)

⟨toplevel_phrase⟩ ::= ⟨type_decl⟩ ;
| ⟨toplevel_node_decl⟩ ;
| ⟨val_decl⟩ ;
| ⟨toplevel_inp_decl⟩ ;
| ⟨toplevel_outp_decl⟩ ;
| HASH ⟨toplevel_directive⟩ ;
| EOF

⟨toplevel_directive⟩ ::= IDENT
| IDENT STRING
| IDENT INT

⟨toplevel_node_decl⟩ ::= **node** IDENT **in** ⟨io_decls⟩ **out** ⟨io_decls⟩

⟨toplevel_inp_decl⟩ ::= INPUT IDENT **:** ⟨type_expr⟩

⟨toplevel_outp_decl⟩ ::= OUTPUT IDENT **:** ⟨type_expr⟩

Lexical Syntax

IDENT ::= ⟨letter⟩ (⟨letter⟩ | ⟨digit⟩)*
⟨letter⟩ ::= 'a', ..., 'z', 'A', ..., 'Z'
INT ::= [-]⟨digit⟩$^+$
⟨digit⟩ ::= '0', ..., '9'
INFIX1 ::= '=' | '!=' | '<' | '>'
INFIX2 ::= '+' | '-'
INFIX3 ::= '*' | '/' | '%'
TYVAR ::= '$' ⟨letter⟩