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
	|	**'** IDENT
⟨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⟩$^+$ $\rightarrow$ ⟨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** ⟨match_case⟩$^{+}_{\text{BAR}}$
⟨simple_labeled_expr⟩	::=	IDENT **:** ⟨simple_expr⟩
	|	⟨simple_expr⟩
	|	˜ IDENT
⟨match_case⟩	::=	⟨pattern⟩ → ⟨expr⟩
⟨simple_expr⟩	::=	IDENT
	|	**(** ⟨expr⟩ **)**
	|	**( )**
	|	⟨const_expr⟩
	|	[⟨expr⟩$^{+}_{;}$]
	|	[ ]
	|	**'** ⟨basic_expr⟩ **'**
⟨const_expr⟩	::=	INT
	|	**true**
	|	**false**
⟨expr_comma_list⟩	::=	⟨expr_comma_list⟩ **,** ⟨expr⟩
	|	⟨expr⟩ **,** ⟨expr⟩
⟨pattern⟩	::=	⟨simple_pattern⟩
	|	⟨pattern_comma_list⟩
	|	⟨pattern⟩ **::** ⟨pattern⟩
	|	[⟨simple_pattern⟩$^{+}_{;}$]
⟨fun_pattern⟩	::=	IDENT
⟨simple_pattern⟩	::=	IDENT
	|	_
	|	**(** ⟨pattern⟩ **)**
	|	**( )**
	|	[ ]
⟨pattern_comma_list⟩	::=	⟨pattern_comma_list⟩ **,** ⟨pattern⟩
	|	⟨pattern⟩ **,** ⟨pattern⟩
⟨struct_graph_desc⟩	::=	⟨struct_decl⟩*
⟨struct_decl⟩	::=	⟨wire_decl⟩
	|	⟨box_decl⟩
⟨wire_decl⟩	::=	**wire** IDENT$^{*}_{,}$ **:** ⟨type_expr⟩
⟨box_decl⟩	::=	**box** IDENT **:** IDENT ⟨box_inps⟩ ⟨box_outps⟩

⟨box_inps⟩	::=	**(** $\langle\text{box\_inp}\rangle^{*}_{,}$ **)**
⟨box_outps⟩	::=	**(** $\langle\text{box\_outp}\rangle^{*}_{,}$ **)**
⟨box_inp⟩	::=	IDENT
	$\mid$	**'** ⟨basic_expr⟩ **'**
⟨box_outp⟩	::=	IDENT
⟨basic_expr⟩	::=	IDENT
	$\mid$	⟨const_expr⟩
	$\mid$	⟨basic_expr⟩ INFIX1 ⟨basic_expr⟩
	$\mid$	⟨basic_expr⟩ INFIX2 ⟨basic_expr⟩
	$\mid$	⟨basic_expr⟩ INFIX3 ⟨basic_expr⟩
	$\mid$	**(** ⟨basic_expr⟩ **)**
⟨toplevel_phrase⟩	::=	⟨type_decl⟩ **;**
	$\mid$	⟨toplevel_node_decl⟩ **;**
	$\mid$	⟨val_decl⟩ **;**
	$\mid$	⟨toplevel_inp_decl⟩ **;**
	$\mid$	⟨toplevel_outp_decl⟩ **;**
	$\mid$	HASH ⟨toplevel_directive⟩ **;**
	$\mid$	EOF
⟨toplevel_directive⟩	::=	IDENT
	$\mid$	IDENT STRING
	$\mid$	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⟩ $\mid$ ⟨digit⟩)*
⟨letter⟩	::=	'a', ..., 'z', 'A', ..., 'Z'
INT	::=	[-]⟨digit⟩$^+$
⟨digit⟩	::=	'0', ..., '9'
INFIX1	::=	'`=`' $\mid$ '`!=`' $\mid$ '`<`' $\mid$ '`>`'
INFIX2	::=	'`+`' $\mid$ '`-`'
INFIX3	::=	'`*`' $\mid$ '`/`' $\mid$ '`%`'