1 Collected syntax

The following sections list the sire syntax, with each section related to a particular feature.

1.1 Program

$$
\begin{aligned}
\textit{program} &= \langle \text{specification} \rangle \ \texttt{:} \ \langle \text{program} \rangle \\
&\mid \langle \text{sequence} \rangle
\end{aligned}
$$

1.2 Specifications

$$
\begin{aligned}
\textit{specification} &= \langle \text{declaration} \rangle \\
&\mid \langle \text{abbreviation} \rangle \\
&\mid \langle \text{definition} \rangle \\
&\mid \langle \text{simultaneous-specification} \rangle \\
\textit{simultaneous-specification} &= \{_2 \ \texttt{\&} \ \langle \text{specification} \rangle \ \}
\end{aligned}
$$

1.2.1 Declarations

$$
\begin{aligned}
\textit{declaration} &= \langle \text{specifier} \rangle \ \{_1 \ \texttt{,} \ \langle \text{name} \rangle \ \} \\
&\mid \langle \text{server-declaration} \rangle \\
&\mid \langle \text{hiding-declaration} \rangle \\
\textit{specifier} &= \langle \text{type} \rangle \\
&\mid \langle \text{type} \rangle \ \langle \text{name} \rangle \\
&\mid \langle \text{type} \rangle \ \langle \text{interface} \rangle \\
&\mid \langle \text{specifier} \rangle \ \texttt{[ ]} \\
&\mid \langle \text{specifier} \rangle \ \texttt{[} \ \langle \text{expression} \rangle \ \texttt{]} \\
\textit{type} &= \texttt{var} \\
&\mid \texttt{chan} \\
&\mid \texttt{call} \\
&\mid \texttt{function} \\
&\mid \texttt{process} \\
&\mid \texttt{server}
\end{aligned}
$$

1.2.2 Abbreviations

$$
\begin{aligned}
\textit{abbreviation} &= \langle \text{specifier} \rangle\ \langle \text{name} \rangle\ \texttt{is}\ \langle \text{element} \rangle \\
&\mid \langle \text{specifier} \rangle\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)}\ \texttt{is}\ \langle \text{element} \rangle \\
&\mid \texttt{val}\ \langle \text{name} \rangle\ \texttt{is}\ \langle \text{expression} \rangle
\end{aligned}
$$

1.3 Sequence

$$
\begin{aligned}
\textit{sequence} &= \{_0\ \texttt{;}\ \langle \text{command} \rangle\ \} \\
\textit{command} &= \langle \text{primitive-command} \rangle \\
&\mid \langle \text{structured-command} \rangle \\
&\mid \langle \text{specification} \rangle\ \texttt{:}\ \langle \text{command} \rangle
\end{aligned}
$$

1.3.1 Primitive commands

$$
\begin{aligned}
\textit{primitive-command} &= \langle \text{skip} \rangle \\
&\mid \langle \text{stop} \rangle \\
&\mid \langle \text{assignment} \rangle \\
&\mid \langle \text{input} \rangle \\
&\mid \langle \text{output} \rangle \\
&\mid \langle \text{connect} \rangle \\
\textit{skip} &= \texttt{skip} \\
\textit{stop} &= \texttt{stop} \\
\textit{assignment} &= \langle \text{element} \rangle\ \texttt{:=}\ \langle \text{expression} \rangle \\
\textit{input} &= \langle \text{element} \rangle\ \texttt{?}\ \langle \text{element} \rangle \\
\textit{output} &= \langle \text{element} \rangle\ \texttt{!}\ \langle \text{expression} \rangle \\
\textit{connect} &= \texttt{connect}\ \langle \text{element} \rangle\ \texttt{to}\ \langle \text{element} \rangle
\end{aligned}
$$

1.3.2 Structured commands

$$
\begin{aligned}
\textit{structured-command} &= \langle \text{alternation} \rangle \\
&\mid \langle \text{conditional} \rangle \\
&\mid \langle \text{case} \rangle \\
&\mid \langle \text{loop} \rangle \\
&\mid \texttt{\{}\ \langle \text{sequence} \rangle\ \texttt{\}} \\
&\mid \texttt{\{}\ \langle \text{parallel} \rangle\ \texttt{\}} \\
&\mid \texttt{seq}\ \{_1\ \texttt{,}\ \langle \text{range} \rangle\ \}\ \texttt{do}\ \langle \text{command} \rangle
\end{aligned}
$$

$$
\begin{aligned}
&| \ \textbf{par}\ \{_1\ \textbf{,}\ \langle \text{range} \rangle\ \}\ \textbf{do}\ \langle \text{command} \rangle \\
\textit{conditional} &= \textbf{test}\ \textbf{\{}\ \{_0\ \textbf{|}\ \langle \text{choice} \rangle\ \textbf{\}} \\
&| \ \textbf{if}\ \langle \text{expression} \rangle\ \textbf{do}\ \langle \text{command} \rangle \\
&| \ \textbf{if}\ \langle \text{expression} \rangle\ \textbf{then}\ \langle \text{command} \rangle\ \textbf{else}\ \langle \text{command} \rangle \\
\textit{choice} &= \langle \text{guarded-choice} \rangle \\
&| \ \langle \text{conditional} \rangle \\
&| \ \langle \text{specification} \rangle\ \textbf{:}\ \langle \text{choice} \rangle \\
\textit{guarded-choice} &= \langle \text{expression} \rangle\ \textbf{:}\ \langle \text{command} \rangle \\
\textit{case} &= \textbf{case}\ \textbf{(}\ \langle \text{expression} \rangle\ \textbf{)}\ \textbf{\{}\ \{_0\ \textbf{|}\ \langle \text{selection} \rangle\ \}\ \textbf{\}} \\
\textit{selection} &= \langle \text{expression} \rangle\ \textbf{:}\ \langle \text{command} \rangle \\
&| \ \textbf{else}\ \langle \text{process} \rangle \\
\textit{alternation} &= \textbf{alt}\ \textbf{\{}\ \{_0\ \textbf{|}\ \langle \text{alternative} \rangle\ \}\ \textbf{\}} \\
\textit{alternative} &= \langle \text{guarded-alternative} \rangle \\
&| \ \langle \text{alternation} \rangle \\
&| \ \langle \text{specification} \rangle\ \textbf{:}\ \langle \text{alternative} \rangle \\
\textit{guarded-alternative} &= \langle \text{guard} \rangle\ \textbf{:}\ \langle \text{command} \rangle \\
\textit{guard} &= \langle \text{input} \rangle \\
&| \ \langle \text{expression} \rangle\ \textbf{\&}\ \langle \text{input} \rangle \\
&| \ \langle \text{expression} \rangle\ \textbf{\&}\ \langle \text{skip} \rangle \\
\textit{loop} &= \textbf{while}\ \langle \text{expression} \rangle\ \textbf{do}\ \langle \text{command} \rangle \\
&| \ \textbf{until}\ \langle \text{expression} \rangle\ \textbf{do}\ \langle \text{command} \rangle \\
&| \ \textbf{do}\ \langle \text{command} \rangle\ \textbf{while}\ \langle \text{expr} \rangle \\
&| \ \textbf{do}\ \langle \text{command} \rangle\ \textbf{until}\ \langle \text{expr} \rangle \\
\textit{parallel} &= \{_0\ \textbf{\&}\ \langle \text{component} \rangle\ \} \\
\textit{component} &= \langle \text{label} \rangle\ \langle \text{process} \rangle \\
&| \ \langle \text{process} \rangle \\
\textit{label} &= \langle \text{name} \rangle\ \textbf{is}
\end{aligned}
$$

1.4 Process

$$
\begin{aligned}
\textit{process} &= \langle \text{interface} \rangle\ \textbf{to}\ \langle \text{command} \rangle \\
&| \ \langle \text{command} \rangle \\
\textit{interface} &= \textbf{interface}\ \textbf{(}\ \{_0\ \textbf{,}\ \langle \text{declaration} \rangle\ \}\ \textbf{)}
\end{aligned}
$$

1.5 Server

1.5.1 Specification

$$
\begin{aligned}
\textit{server} &= \langle \text{interface} \rangle \ \texttt{to} \ \langle \text{server-specification} \rangle \\
\textit{primitive-type} &= \langle \text{call-type} \rangle \\
\textit{call-type} &= \texttt{call} \\
\textit{declaration} &= \langle \text{type} \rangle \ \{_1 \ \texttt{,} \ \langle \text{name} \rangle \ \texttt{(} \ \{_0 \ \texttt{,} \ \langle \text{formal} \rangle \ \} \ \texttt{)} \ \} \\
\textit{server-specification} &= \langle \text{declaration} \rangle \\
&\mid \ \texttt{\{} \ \{_1 \ \texttt{:} \ \langle \text{declaration} \rangle \ \} \ \texttt{\}} \\
\textit{declaration} &= \texttt{initial} \ \langle \text{command} \rangle \\
&\mid \ \texttt{final} \ \langle \text{command} \rangle \\
&\mid \ \langle \text{alternation} \rangle \\
\textit{guard} &= \texttt{accept} \ \langle \text{name} \rangle \ \texttt{(} \ \{_0 \ \texttt{,} \ \langle \text{formal} \rangle \ \} \ \texttt{)} \\
&\mid \ \langle \text{expression} \rangle \ \texttt{\&} \ \texttt{accept} \ \langle \text{name} \rangle \ \texttt{(} \ \{_0 \ \texttt{,} \ \langle \text{formal} \rangle \ \} \ \texttt{)}
\end{aligned}
$$

1.5.2 Declarations

$$
\begin{aligned}
\textit{server-declaration} &= \texttt{server} \ \langle \text{name} \rangle \ \texttt{is} \ \langle \text{server} \rangle \\
\textit{server} &= \langle \text{server-array} \rangle \\
\textit{server-array} &= \texttt{[} \ \{_1 \ \texttt{,} \ \langle \text{server} \rangle \ \} \ \texttt{]} \\
\textit{hiding-declaration} &= \texttt{from} \ \texttt{\{} \ \{_1 \ \texttt{:} \ \langle \text{declaration} \rangle \ \} \ \texttt{\}} \ \texttt{interface} \ \langle \text{name} \rangle
\end{aligned}
$$

1.5.3 Call

$$
\begin{aligned}
\textit{command} &= \langle \text{server-call} \rangle \\
\textit{server-call} &= \langle \text{element} \rangle \ \texttt{(} \ \{_0 \ \texttt{,} \ \langle \text{actual} \rangle \ \} \ \texttt{)}
\end{aligned}
$$

1.6 Replication

$$
\begin{aligned}
\textit{label} &= \langle \text{name} \rangle \ \texttt{is} \ \langle \text{replicator} \rangle \\
\textit{conditional} &= \texttt{if} \ \langle \text{replicator} \rangle \ \langle \text{choice} \rangle \\
\textit{case} &= \texttt{case} \ \langle \text{replicator} \rangle \ \langle \text{selection} \rangle \\
\textit{alternation} &= \texttt{alt} \ \langle \text{replicator} \rangle \ \langle \text{alternative} \rangle \\
\textit{declaration} &= \texttt{server} \ \langle \text{name} \rangle \ \texttt{is} \ \langle \text{replicator} \rangle \ \langle \text{server} \rangle \\
\textit{replicator} &= \texttt{[} \ \{_1 \ \texttt{,} \ \langle \text{range} \rangle \ \} \ \texttt{]} \\
\textit{range} &= \langle \text{name} \rangle \ \texttt{=} \ \langle \text{expression} \rangle \ \texttt{for} \ \langle \text{expression} \rangle
\end{aligned}
$$

| ⟨name⟩ **=** ⟨expression⟩ **for** ⟨expression⟩ **step** ⟨expression⟩

1.7 Expressions

expression = ⟨unary-operator⟩ ⟨operand⟩
| ⟨operand⟩ ⟨binary-operator⟩ ⟨operand⟩
| ⟨operand⟩
operand = ⟨element⟩
| ⟨literal⟩
| **(** ⟨expression⟩ **)**

1.7.1 Valof

valof = **valof** ⟨cmd⟩ **result** ⟨expression⟩
| ⟨specification⟩ **:** ⟨valof⟩
expression = **(** ⟨valof⟩ **)**

1.8 Elements

element = ⟨element⟩ **[** ⟨expression⟩ **]**
| ⟨field⟩
| ⟨name⟩
field = ⟨element⟩ **.** ⟨name⟩
literal = ⟨decimal-integer⟩
| **#** ⟨hexdecimal-integer⟩
| ⟨byte⟩
| **true**
| **false**
hexdecimal-integer = ⟨hexdecimal-digit⟩
| ⟨hexdecimal-digit⟩ ⟨hexdecimal-integer⟩
decimal-integer = ⟨digit⟩
| ⟨digit⟩ ⟨decimal-integer⟩
byte = **'** ⟨character⟩ **'**

1.9 Abstraction

1.9.1 Definitions

$$
\begin{aligned}
\textit{definition} &= \langle \text{procedure} \rangle \\
&\mid \langle \text{function} \rangle \\
\textit{procedure} &= \texttt{process}\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)}\ \texttt{is}\ \langle \text{process} \rangle \\
&\mid \texttt{server}\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)}\ \texttt{is}\ \langle \text{server} \rangle \\
&\mid \texttt{server}\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)}\ \texttt{inherits}\ \langle \text{hiding-declaration} \rangle \\
\textit{function} &= \texttt{function}\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)}\ \texttt{is}\ \langle \text{valof} \rangle \\
\textit{formal} &= \langle \text{specifier} \rangle\ \langle \text{name} \rangle \\
&\mid \langle \text{specifier} \rangle\ \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{formal} \rangle\ \}\ \texttt{)} \\
&\mid \texttt{val}\ \langle \text{name} \rangle
\end{aligned}
$$

1.9.2 Instances

$$
\begin{aligned}
\textit{process} &= \langle \text{instance} \rangle \\
\textit{server} &= \langle \text{instance} \rangle \\
\textit{command} &= \langle \text{instance} \rangle \\
\textit{expression} &= \langle \text{instance} \rangle \\
\textit{instance} &= \langle \text{name} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{actual} \rangle\ \}\ \texttt{)} \\
\textit{actual} &= \langle \text{element} \rangle \\
&\mid \langle \text{expression} \rangle
\end{aligned}
$$

1.10 Compiler transformations

1.10.1 Process distribution

$$
\textit{process} = \texttt{on}\ \langle \text{expression} \rangle\ \texttt{do}\ \langle \text{process} \rangle
$$

1.10.2 Channel end references

$$
\begin{aligned}
\textit{absolute-reference} &= \texttt{(}\ \langle \text{expression} \rangle\ \texttt{:}\ \langle \text{expression} \rangle\ \texttt{:}\ \langle \text{expression} \rangle\ \texttt{)} \\
\textit{interface} &= \texttt{interface}\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{declaration} \rangle\ \}\ \texttt{)}\ \texttt{@}\ \langle \text{absolute-reference} \rangle \\
\textit{server-call} &= \langle \text{local-chanend} \rangle\ \texttt{!}\ \langle \text{element} \rangle\ \texttt{(}\ \{_0\ \texttt{,}\ \langle \text{actual} \rangle\ \}\ \texttt{)} \\
\textit{local-chanend} &= \langle \text{element} \rangle\ \texttt{@}\ \langle \text{absolute-reference} \rangle \\
\textit{chanend} &= \langle \text{remote-chanend} \rangle
\end{aligned}
$$

remote-chanend = **@** ⟨absolute-reference⟩ **.** ⟨expression⟩

2 Ordered syntax

The following lists each element of the sire syntax in alphabetical order.

abbreviation = **val** ⟨name⟩ **is** ⟨expression⟩
| ⟨specifier⟩ ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** **is** ⟨element⟩
| ⟨specifier⟩ ⟨name⟩ **is** ⟨element⟩

absolute-reference = **(** ⟨expression⟩ **:** ⟨expression⟩ **:** ⟨expression⟩ **)**

actual = ⟨element⟩
| ⟨expression⟩

alternation = **alt {** {$_0$ **|** ⟨alternative⟩ } **}**
| **alt** ⟨replicator⟩ ⟨alternative⟩

alternative = ⟨alternation⟩
| ⟨guarded-alternative⟩
| ⟨specification⟩ **:** ⟨alternative⟩

assignment = ⟨element⟩ **:=** ⟨expression⟩

byte = **'** ⟨character⟩ **'**

call-type = **call**

case = **case (** ⟨expression⟩ **) {** {$_0$ **|** ⟨selection⟩ } **}**
| **case** ⟨replicator⟩ ⟨selection⟩

chanend = ⟨remote-chanend⟩

choice = ⟨conditional⟩
| ⟨guarded-choice⟩
| ⟨specification⟩ **:** ⟨choice⟩

command = ⟨instance⟩
| ⟨primitive-command⟩
| ⟨server-call⟩
| ⟨specification⟩ **:** ⟨command⟩
| ⟨structured-command⟩

component = ⟨label⟩ ⟨process⟩
| ⟨process⟩

conditional = **if** ⟨expression⟩ **do** ⟨command⟩
| **if** ⟨expression⟩ **then** ⟨command⟩ **else** ⟨command⟩
| **if** ⟨replicator⟩ ⟨choice⟩
| **test {** {$_0$ **|** ⟨choice⟩ **}**

connect = **connect** ⟨element⟩ **to** ⟨element⟩

decimal-integer = ⟨digit⟩
| ⟨digit⟩ ⟨decimal-integer⟩

declaration = **final** ⟨command⟩

| **initial** ⟨command⟩
| **server** ⟨name⟩ **is** ⟨replicator⟩ ⟨server⟩
| ⟨alternation⟩
| ⟨hiding-declaration⟩
| ⟨server-declaration⟩
| ⟨specifier⟩ {$_1$ **,** ⟨name⟩ }
| ⟨type⟩ {$_1$ **,** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** }

definition = ⟨function⟩
| ⟨procedure⟩

element = ⟨element⟩ **[** ⟨expression⟩ **]**
| ⟨field⟩
| ⟨name⟩

expression = **(** ⟨valof⟩ **)**
| ⟨instance⟩
| ⟨operand⟩
| ⟨operand⟩ ⟨binary-operator⟩ ⟨operand⟩
| ⟨unary-operator⟩ ⟨operand⟩

field = ⟨element⟩ **.** ⟨name⟩

formal = **val** ⟨name⟩
| ⟨specifier⟩ ⟨name⟩
| ⟨specifier⟩ ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)**

function = **function** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** **is** ⟨valof⟩

guard = **accept** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)**
| ⟨expression⟩ **&** **accept** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)**
| ⟨expression⟩ **&** ⟨input⟩
| ⟨expression⟩ **&** ⟨skip⟩
| ⟨input⟩

guarded-alternative = ⟨guard⟩ **:** ⟨command⟩

guarded-choice = ⟨expression⟩ **:** ⟨command⟩

hexdecimal-integer = ⟨hexdecimal-digit⟩
| ⟨hexdecimal-digit⟩ ⟨hexdecimal-integer⟩

hiding-declaration = **from** **{** {$_1$ **:** ⟨declaration⟩ } **}** **interface** ⟨name⟩

input = ⟨element⟩ **?** ⟨element⟩

instance = ⟨name⟩ **(** {$_0$ **,** ⟨actual⟩ } **)**

interface = **interface** **(** {$_0$ **,** ⟨declaration⟩ } **)**
| **interface** **(** {$_0$ **,** ⟨declaration⟩ } **)** **@** ⟨absolute-reference⟩

label = ⟨name⟩ **is**
| ⟨name⟩ **is** ⟨replicator⟩

literal = **#** ⟨hexdecimal-integer⟩

| **false**
| **true**
| ⟨byte⟩
| ⟨decimal-integer⟩

local-chanend = ⟨element⟩ **@** ⟨absolute-reference⟩

loop = **do** ⟨command⟩ **until** ⟨expr⟩
| **do** ⟨command⟩ **while** ⟨expr⟩
| **until** ⟨expression⟩ **do** ⟨command⟩
| **while** ⟨expression⟩ **do** ⟨command⟩

operand = **(** ⟨expression⟩ **)**
| ⟨element⟩
| ⟨literal⟩

output = ⟨element⟩ **!** ⟨expression⟩

parallel = {$_0$ **&** ⟨component⟩ }

primitive-command = ⟨assignment⟩
| ⟨connect⟩
| ⟨input⟩
| ⟨output⟩
| ⟨skip⟩
| ⟨stop⟩

primitive-type = ⟨call-type⟩

procedure = **process** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** **is** ⟨process⟩
| **server** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** **inherits** ⟨hiding-declaration⟩
| **server** ⟨name⟩ **(** {$_0$ **,** ⟨formal⟩ } **)** **is** ⟨server⟩

process = **on** ⟨expression⟩ **do** ⟨process⟩
| ⟨command⟩
| ⟨instance⟩
| ⟨interface⟩ **to** ⟨command⟩

program = ⟨sequence⟩
| ⟨specification⟩ **:** ⟨program⟩

range = ⟨name⟩ **=** ⟨expression⟩ **for** ⟨expression⟩
| ⟨name⟩ **=** ⟨expression⟩ **for** ⟨expression⟩ **step** ⟨expression⟩

remote-chanend = **@** ⟨absolute-reference⟩ **.** ⟨expression⟩

replicator = **[** {$_1$ **,** ⟨range⟩ } **]**

selection = **else** ⟨process⟩
| ⟨expression⟩ **:** ⟨command⟩

sequence = {$_0$ **;** ⟨command⟩ }

server = ⟨instance⟩
| ⟨interface⟩ **to** ⟨server-specification⟩

| ⟨server-array⟩
server-array = **[** {$_1$ **,** ⟨server⟩ } **]**
server-call = ⟨element⟩ **(** {$_0$ **,** ⟨actual⟩ } **)**
| ⟨local-chanend⟩ **!** ⟨element⟩ **(** {$_0$ **,** ⟨actual⟩ } **)**
server-declaration = **server** ⟨name⟩ **is** ⟨server⟩
server-specification = **{** {$_1$ **:** ⟨declaration⟩ } **}**
| ⟨declaration⟩
simultaneous-specification = {$_2$ **&** ⟨specification⟩ }
skip = **skip**
specification = ⟨abbreviation⟩
| ⟨declaration⟩
| ⟨definition⟩
| ⟨simultaneous-specification⟩
specifier = ⟨specifier⟩ **[ ]**
| ⟨specifier⟩ **[** ⟨expression⟩ **]**
| ⟨type⟩
| ⟨type⟩ ⟨interface⟩
| ⟨type⟩ ⟨name⟩
stop = **stop**
structured-command = **par** {$_1$ **,** ⟨range⟩ } **do** ⟨command⟩
| **seq** {$_1$ **,** ⟨range⟩ } **do** ⟨command⟩
| **{** ⟨parallel⟩ **}**
| **{** ⟨sequence⟩ **}**
| ⟨alternation⟩
| ⟨case⟩
| ⟨conditional⟩
| ⟨loop⟩
type = **call**
| **chan**
| **function**
| **process**
| **server**
| **var**
valof = **valof** ⟨cmd⟩ **result** ⟨expression⟩
| ⟨specification⟩ **:** ⟨valof⟩

3 Operators

A ⟨binary-operator⟩ $\oplus$ takes two operands a and b and produces a value $a \oplus b$. A binary *arithmetic operators* takes signed integer operands and produces a signed integer result.

Symbol	Meaning	Definition
+	sum	$a + b$
-	difference	$a - b$
*	produce	$a \times b$
/	quotient	a/b
`rem`	remainder	$a \mod b$

A binary *relational operator* takes two signed integer values and produces the value `true` or `false`.

Symbol	Meaning	Definition
=	equality	$a = b$
~=	inequality	$a \neq b$
<	less than	$a < b$
<=	less than or equal	$a \leq b$
>	greater than	$a > b$
>=	greater than or equal	$a \geq b$

A binary *logical operator* takes two operands and produces a bitwise result $b_i = a_i \oplus b_i$ for $0 \leq i < n$.

Symbol	Meaning	Definition
`or`	bitwise or	b `or` $0 = b$, b `or` $1 = 1$
`and`	bitwise and	b `and` $0 = 0$, b `and` $1 = b$
`xor`	bitwise exclusive or	b `xor` $0 = b$, b `xor` $1 = 1$, 1 `xor` $1 = 0$,
<<	left bitwise shift	$b_i = b_{i+1}$, $b_0 = 0$
>>	right bitwise shift	$b_i = b_{i-1}$, $b_n - 1 = 0$

A ⟨unary-operator⟩ $\oplus$ takes a single operand a and produces the value $\oplus a$.

Symbol	Meaning	Definition
-	negation	$0 - a$
~	bitwise not	$\sim 0 = 1, \sim 1 = 0$

4 Representation of values

Signed values are represented as a two's complement bit-pattern. With a word size of n, values in the range $-2^{n-1} \leq n < 2^{n-1}$ can be represented.

The literal value `true` represents an all-1 bit pattern (the value -1) and the literal value `false` represents an all-0 bit pattern (the value 0). The effect of this is consistent with the logical not operation: `not false` $=$ `true` and `not true` = `false`.

5 Character set

A ⟨character⟩ can be any alphabetical character

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

or any special character

```
_ + - = , .  :  ; ?  { } [ ] ( ) # & !  * @ | " '
```

A ⟨digit⟩ can be any numeric character

```
0 1 2 3 4 5 6 7 8 9
```

A ⟨hex-digit⟩ can be any numeric character or

```
A B C D E F a b c d e f
```

A ⟨name⟩ consists of a sequence of alphanumeric characters and underscores.

6 Comments

A comment is introduced with a '%' symbol and continues until the end of the line.

7 Keywords

The following are keywords in sire and cannot be used for names.

accept	**function**	**skip**
alt	**if**	**step**
call	**inherits**	**stop**
case	**initial**	**test**
chan	**interface**	**then**
connect	**is**	**to**
do	**on**	**true**
else	**par**	**until**
false	**process**	**val**
final	**result**	**valof**
for	**seq**	**var**
from	**server**	**while**