

Helena 2.3

Syntax summary

Sami Evangelista - (Sami [dot] Evangelista [at] lipn.univ-paris13 [dot] fr)

March 28, 2013

1 Net specification language

Nets

$$\begin{array}{ll}
\langle net \rangle & ::= \langle net\ name \rangle \\
& \quad [' (' \langle net\ parameter\ list \rangle ') '] \\
& \quad ' \{ ' (\langle definition \rangle)^* ' \} ' \\
\langle net\ name \rangle & ::= \langle name \rangle \\
\langle definition \rangle & ::= \langle type \rangle \\
& \quad | \langle constant \rangle \\
& \quad | \langle function \rangle \\
& \quad | \langle place \rangle \\
& \quad | \langle transition \rangle \\
& \quad | \langle state\ proposition \rangle
\end{array}$$

Net parameters

$$\begin{aligned} \langle \textit{net parameter list} \rangle &::= \langle \textit{net parameter} \rangle \\ &\quad | \langle \textit{net parameter} \rangle ', ' \langle \textit{net parameter list} \rangle \\ \langle \textit{net parameter} \rangle &::= \langle \textit{net parameter name} \rangle ' : = ' \langle \textit{number} \rangle \\ \langle \textit{net parameter name} \rangle &::= \langle \textit{name} \rangle \end{aligned}$$

Types and subtypes

$\langle type \rangle$::=	$\langle type\ name \rangle\ ' : '\ \langle type\ definition \rangle\ ' ; '$
		$\langle subtype \rangle$
$\langle type\ name \rangle$::=	$\langle name \rangle$
$\langle type\ definition \rangle$::=	$\langle range\ type \rangle$
		$\langle modulo\ type \rangle$
		$\langle enumeration\ type \rangle$
		$\langle vector\ type \rangle$
		$\langle struct\ type \rangle$
		$\langle list\ type \rangle$
		$\langle set\ type \rangle$

Range type

$$\begin{aligned}\langle range\ type \rangle &::= \langle range \rangle \\ \langle range \rangle &::= \text{'range'} \langle expression \rangle \dots \langle expression \rangle\end{aligned}$$

Modulo type

$$\langle modular\ type \rangle ::= \textbf{mod} \langle expression \rangle$$

Enumeration type

<i>enumeration type</i>	<code>::= 'enum' ' (' <i>enumeration constant</i> ' (',' <i>enumeration constant</i>)[*] ') '</code>
<i>enumeration constant</i>	<code>::= <i><name></i></code>

Vector type

$\langle \text{vector type} \rangle ::= \text{'vector' ' [' } \langle \text{index type list} \rangle \text{ '] ' 'of' } \langle \text{type name} \rangle$
 $\langle \text{index type list} \rangle ::= \langle \text{type name} \rangle (\text{' , ' } \langle \text{type name} \rangle)^*$

Structured type

$\langle \text{struct type} \rangle ::= \text{'struct' ' { ' } (\langle \text{component} \rangle)^+ \text{ ' } \text{' ; '}$
 $\langle \text{component} \rangle ::= \langle \text{type name} \rangle \langle \text{component name} \rangle \text{' ; '}$
 $\langle \text{component name} \rangle ::= \langle \text{name} \rangle$

List type

$\langle \text{list type} \rangle ::= \text{'list' ' [' } \langle \text{type name} \rangle \text{ '] ' 'of' } \langle \text{type name} \rangle \text{' with ' 'capacity' } \langle \text{expression} \rangle$

Set type

$\langle \text{set type} \rangle ::= \text{'set' 'of' } \langle \text{type name} \rangle \text{' with ' 'capacity' } \langle \text{expression} \rangle$

Subtype

$\langle \text{subtype} \rangle ::= \langle \text{subtype name} \rangle \text{' : ' } \langle \text{parent name} \rangle [\langle \text{constraint} \rangle]$
 $\langle \text{subtype name} \rangle ::= \langle \text{type name} \rangle$
 $\langle \text{parent name} \rangle ::= \langle \text{type name} \rangle$
 $\langle \text{constraint} \rangle ::= \langle \text{range} \rangle$

Constants

$\langle \text{constant} \rangle ::= \text{'constant' } \langle \text{type name} \rangle \langle \text{constant name} \rangle \text{' : = ' } \langle \text{expression} \rangle \text{' ; '}$
 $\langle \text{constant name} \rangle ::= \langle \text{name} \rangle$

Functions

$\langle \text{function} \rangle ::= \langle \text{function declaration} \rangle$
 $\quad \quad \quad | \quad \langle \text{function body} \rangle$
 $\langle \text{function prototype} \rangle ::= \text{'function' } \langle \text{function name} \rangle$
 $\quad \quad \quad \text{' (' } \langle \text{parameters specification} \rangle \text{ ') ' ' -> ' } \langle \text{type name} \rangle$
 $\langle \text{function declaration} \rangle ::= \langle \text{function prototype} \rangle \text{' ; '}$
 $\langle \text{function body} \rangle ::= \text{'import' } \langle \text{function prototype} \rangle \text{' ; '}$
 $\quad \quad \quad | \quad \langle \text{function prototype} \rangle \langle \text{statement} \rangle$
 $\langle \text{parameters specification} \rangle ::= [\langle \text{parameter specification} \rangle (\text{' , ' } \langle \text{parameter specification} \rangle)^*]$
 $\langle \text{parameter specification} \rangle ::= \langle \text{type name} \rangle \langle \text{parameter name} \rangle$
 $\langle \text{function name} \rangle ::= \langle \text{name} \rangle$
 $\langle \text{parameter name} \rangle ::= \langle \text{name} \rangle$

Statements

$\langle \text{statement} \rangle ::= \langle \text{assignment} \rangle$
 $\quad \quad \quad | \quad \langle \text{if statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{case statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{while statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{for statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{return statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{assert statement} \rangle$
 $\quad \quad \quad | \quad \langle \text{block} \rangle$

Assignment

$\langle \text{assignment} \rangle ::= \langle \text{variable} \rangle \text{' : = ' } \langle \text{expression} \rangle \text{' ; '}$

If-then-else

$\langle \text{if statement} \rangle ::= \text{'if' ' (' } \langle \text{expression} \rangle \text{ ') ' } \langle \text{true statement} \rangle [\text{' else ' } \langle \text{false statement} \rangle]$
 $\langle \text{true statement} \rangle ::= \langle \text{statement} \rangle$
 $\langle \text{false statement} \rangle ::= \langle \text{statement} \rangle$

Case

$\langle \text{case statement} \rangle ::= \text{'case' ' (' } \langle \text{expression} \rangle \text{') ' ' \{ ' (} \langle \text{case alternative} \rangle \text{') ' }^* [\langle \text{default alternative} \rangle] \text{' ' ' }$
 $\langle \text{case alternative} \rangle ::= \langle \text{expression} \rangle \text{' : ' } \langle \text{statement} \rangle$
 $\langle \text{default alternative} \rangle ::= \text{'default' ' : ' } \langle \text{statement} \rangle$

While

$\langle \text{while statement} \rangle ::= \text{'while' ' (' } \langle \text{expression} \rangle \text{') ' } \langle \text{statement} \rangle$

For loop

$\langle \text{for statement} \rangle ::= \text{'for' ' (' } \langle \text{iteration scheme} \rangle \text{') ' } \langle \text{statement} \rangle$
 $\langle \text{iteration scheme} \rangle ::= \langle \text{iteration variable} \rangle [\text{' , ' } \langle \text{iteration variable} \rangle]$
 $\langle \text{iteration variable} \rangle ::= \langle \text{variable name} \rangle \text{' in' } \langle \text{type name} \rangle [\langle \text{range} \rangle]$
| $\langle \text{variable name} \rangle \text{' in' } \langle \text{place name} \rangle$
| $\langle \text{variable name} \rangle \text{' in' } \langle \text{expression} \rangle$

Return

$\langle \text{return statement} \rangle ::= \text{'return' } \langle \text{expression} \rangle \text{' ;' }$

Assertion

$\langle \text{assert statement} \rangle ::= \langle \text{assert} \rangle$

Block

$\langle \text{block} \rangle ::= \text{'\{ ' (} \langle \text{declaration} \rangle \text{') ' }^* (\langle \text{statement} \rangle \text{') ' }^+ \text{' \}' }$
 $\langle \text{declaration} \rangle ::= \langle \text{constant declaration} \rangle$
| $\langle \text{variable declaration} \rangle$
 $\langle \text{variable declaration} \rangle ::= \langle \text{type name} \rangle \langle \text{variable name} \rangle [\text{' : ' } \langle \text{expression} \rangle] \text{' ;' }$
 $\langle \text{variable name} \rangle ::= \langle \text{name} \rangle$

Places

$\langle \text{place} \rangle ::= \text{'place' } \langle \text{place name} \rangle \text{' ' } \{ \langle \text{place domain} \rangle (\langle \text{place attribute} \rangle)^* \text{' ' } \}$
 $\langle \text{place attribute} \rangle ::= \langle \text{initial marking} \rangle$
| $\langle \text{capacity} \rangle$
| $\langle \text{place type} \rangle$

Domain

$\langle \text{domain} \rangle ::= \text{'dom' ' : ' } \langle \text{domain definition} \rangle \text{' ;' }$
 $\langle \text{domain definition} \rangle ::= \text{'epsilon' }$
| $\langle \text{types product} \rangle$
 $\langle \text{types product} \rangle ::= \langle \text{type name} \rangle (\text{' * ' } \langle \text{type name} \rangle)^*$

Initial marking

$\langle \text{initial marking} \rangle ::= \text{'init' ' : ' } \langle \text{marking} \rangle \text{' ;' }$
 $\langle \text{marking} \rangle ::= \langle \text{arc label} \rangle$

Capacity

$\langle \text{capacity} \rangle ::= \text{'capacity' ' : ' } \langle \text{expression} \rangle \text{' ;' }$

Type

$\langle \text{place type} \rangle ::= \text{'type' ' : ' } \langle \text{place type name} \rangle \text{' ;' }$
 $\langle \text{place type name} \rangle ::= \text{'process' }$
| 'local'
| 'shared'
| 'protected'
| 'buffer'
| 'ack'

Transitions

$\langle transition \rangle$	$::=$	$\langle transition \rangle$ $\langle transition\ name \rangle$ $\{ \langle transition\ inputs \rangle$ $\langle transition\ outputs \rangle$ $[\langle transition\ inhibitors \rangle]$ $[\langle transition\ free\ variables \rangle]$ $[\langle transition\ bound\ variables \rangle]$ $(\langle transition\ attribute \rangle)^* \}$
$\langle transition\ name \rangle$	$::=$	$\langle name \rangle$
$\langle transition\ inputs \rangle$	$::=$	in $\{ \langle arc \rangle^* \}$
$\langle transition\ outputs \rangle$	$::=$	out $\{ \langle arc \rangle^* \}$
$\langle transition\ inhibitors \rangle$	$::=$	inhibit $\{ \langle arc \rangle^* \}$
$\langle transition\ attribute \rangle$	$::=$	$\langle transition\ guard \rangle$ $ \langle transition\ priority \rangle$ $ \langle transition\ description \rangle$ $ \langle safe \rangle$

Arcs

$$\langle arc \rangle ::= \langle place\ name \rangle ' : ' \langle arc\ label \rangle ' ; '$$

Free variables

$\langle \text{transition free variables} \rangle$	$::=$	<code>'pick' '{' ($\langle \text{free variable} \rangle$)* '}'</code>
$\langle \text{free variable} \rangle$	$::=$	$\langle \text{free variable name} \rangle$ <code>'in'</code> $\langle \text{free variable domain} \rangle$
$\langle \text{free variable domain} \rangle$	$::=$	$\langle \text{type name} \rangle$ [$\langle \text{range} \rangle$]
		$\langle \text{expression} \rangle$

Bound variables

$$\begin{aligned} \langle \text{transition bound variables} \rangle &::= \text{'let' '{' } (\langle \text{transition bound variable} \rangle)^* \text{'}} \\ \langle \text{transition bound variable} \rangle &::= \langle \text{type name} \rangle \langle \text{variable name} \rangle \text{' := ' } \langle \text{expression} \rangle \text{' ; '} \end{aligned}$$

Guard

$$\begin{aligned} \langle \text{transition guard} \rangle &::= \text{'guard' ':' } \langle \text{guard definition} \rangle \text{' ; ' } \\ \langle \text{guard definition} \rangle &::= \langle \text{expression} \rangle \end{aligned}$$

Priority

$$\langle transition\ priority \rangle ::= \textbf{priority} \textbf{' : ' } \langle expression \rangle \textbf{ ; '}$$

Safe attribute

$$\langle safe \rangle ::= \text{'safe' ; '}$$

Description

$$\langle \textit{transition description} \rangle ::= \textbf{description} \textbf{' : ' } \langle \textit{string} \rangle \textbf{[';' } \langle \textit{non empty expression list} \rangle \textbf{] ';' }$$

Expressions

$\langle \text{expression} \rangle$	$::=$	$\text{'('} \langle \text{expression} \rangle \text{'}$ $\langle \text{enumeration constant} \rangle$ $\langle \text{predecessor-successor operation} \rangle$ $\langle \text{comparison operation} \rangle$ $\langle \text{function call} \rangle$ $\langle \text{if-then-else} \rangle$ $\langle \text{structure component} \rangle$ $\langle \text{vector} \rangle$ $\langle \text{vector assignment} \rangle$ $\langle \text{list} \rangle$ $\langle \text{list assignment} \rangle$ $\langle \text{list concatenation} \rangle$ $\langle \text{empty set} \rangle$ $\langle \text{set membership} \rangle$ $\langle \text{token component} \rangle$ $\langle \text{iterator} \rangle$	$\langle \text{numerical constant} \rangle$ $\langle \text{variable} \rangle$ $\langle \text{integer operation} \rangle$ $\langle \text{boolean operation} \rangle$ $\langle \text{cast} \rangle$ $\langle \text{structure} \rangle$ $\langle \text{structure assignment} \rangle$ $\langle \text{vector component} \rangle$ $\langle \text{empty list} \rangle$ $\langle \text{list component} \rangle$ $\langle \text{list slice} \rangle$ $\langle \text{list membership} \rangle$ $\langle \text{set} \rangle$ $\langle \text{set operation} \rangle$ $\langle \text{attribute} \rangle$
$\langle \text{expression list} \rangle$	$::=$	ϵ	
		$\langle \text{non empty expression list} \rangle$	
$\langle \text{non empty expression list} \rangle$	$::=$	$\langle \text{expression} \rangle \text{' , ' } \langle \text{expression} \rangle^*$	

Numerical and enumeration constants

$\langle \text{numerical constant} \rangle$	$::=$	$\langle \text{number} \rangle$
$\langle \text{enumeration constant} \rangle$	$::=$	$\langle \text{name} \rangle$

Predecessor and successor operators

$\langle \text{predecessor-successor operation} \rangle$	$::=$	'pred' $\langle \text{expression} \rangle$
		'succ' $\langle \text{expression} \rangle$

Integer arithmetic

$\langle \text{integer operation} \rangle$	$::=$	$\langle \text{expression} \rangle \text{' + ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' - ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' * ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' / ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' \% ' } \langle \text{expression} \rangle$
		$\text{' + ' } \langle \text{expression} \rangle$
		$\text{' - ' } \langle \text{expression} \rangle$

Comparison operators

$\langle \text{comparison operation} \rangle$	$::=$	$\langle \text{expression} \rangle \text{' = ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' != ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' > ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' >= ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' < ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' <= ' } \langle \text{expression} \rangle$

Boolean logic

$\langle \text{boolean operation} \rangle$	$::=$	$\langle \text{expression} \rangle \text{' or ' } \langle \text{expression} \rangle$
		$\langle \text{expression} \rangle \text{' and ' } \langle \text{expression} \rangle$
		$\text{' not ' } \langle \text{expression} \rangle$

Variables

$\langle \text{variable} \rangle$	$::=$	$\langle \text{variable name} \rangle$
		$\langle \text{structure component} \rangle$
		$\langle \text{vector component} \rangle$
		$\langle \text{list component} \rangle$

Structures

$\langle \text{structure} \rangle ::= \text{'{' } \langle \text{non empty expression list} \rangle \text{'}}$
 $\langle \text{structure component} \rangle ::= \langle \text{variable} \rangle \text{'.' } \langle \text{component name} \rangle$
 $\langle \text{structure assignment} \rangle ::= \langle \text{expression} \rangle \text{':::' } (\langle \text{component name} \rangle \text{'::=' } \langle \text{expression} \rangle \text{'})'$

Vectors

$\langle \text{vector} \rangle ::= \text{'[' } \langle \text{non empty expression list} \rangle \text{'}'$
 $\langle \text{vector component} \rangle ::= \langle \text{variable} \rangle \text{'[' } \langle \text{non empty expression list} \rangle \text{'}'$
 $\langle \text{vector assignment} \rangle ::= \langle \text{expression} \rangle \text{':::' } (\text{'[' } \langle \text{non empty expression list} \rangle \text{'}'::=' } \langle \text{expression} \rangle \text{'})'$

Lists

$\langle \text{empty list} \rangle ::= \text{'empty'}$
 $\langle \text{list} \rangle ::= \text{'|'} \langle \text{non empty expression list} \rangle \text{'|'}$
 $\langle \text{list component} \rangle ::= \langle \text{variable} \rangle \text{'[' } \langle \text{expression} \rangle \text{'}'$
 $\langle \text{list assignment} \rangle ::= \langle \text{expression} \rangle \text{':::' } (\text{'[' } \langle \text{expression} \rangle \text{'}'::=' } \langle \text{expression} \rangle \text{'})'$
 $\langle \text{list slice} \rangle ::= \langle \text{expression} \rangle \text{'[' } \langle \text{expression} \rangle \text{'..' } \langle \text{expression} \rangle \text{'}'$
 $\langle \text{list concatenation} \rangle ::= \langle \text{expression} \rangle \text{'\&' } \langle \text{expression} \rangle$
 $\langle \text{list membership} \rangle ::= \langle \text{expression} \rangle \text{'in' } \langle \text{expression} \rangle$

Sets

$\langle \text{empty set} \rangle ::= \text{'empty'}$
 $\langle \text{set} \rangle ::= \text{'|'} \langle \text{non empty expression list} \rangle \text{'|'}$
 $\langle \text{set membership} \rangle ::= \langle \text{expression} \rangle \text{'in' } \langle \text{expression} \rangle$
 $\langle \text{set operation} \rangle ::= \langle \text{expression} \rangle \text{'or' } \langle \text{expression} \rangle$
 $\quad \quad \quad \quad \quad \quad \quad \quad \langle \text{expression} \rangle \text{'and' } \langle \text{expression} \rangle$
 $\quad \quad \quad \quad \quad \quad \quad \quad \langle \text{expression} \rangle \text{'-'} \langle \text{expression} \rangle$

Function call

$\langle \text{function call} \rangle ::= \langle \text{function name} \rangle \text{'(' } \langle \text{expression list} \rangle \text{'})'$

Cast

$\langle \text{cast} \rangle ::= \langle \text{type name} \rangle \text{'(' } \langle \text{expression} \rangle \text{'})'$

If-then-else

$\langle \text{if-then-else} \rangle ::= \langle \text{condition} \rangle \text{'?' } \langle \text{true expression} \rangle \text{':::' } \langle \text{false expression} \rangle$
 $\langle \text{condition} \rangle ::= \langle \text{expression} \rangle$
 $\langle \text{true expression} \rangle ::= \langle \text{expression} \rangle$
 $\langle \text{false expression} \rangle ::= \langle \text{expression} \rangle$

Token component

$\langle \text{token component} \rangle ::= \langle \text{token} \rangle \text{'->' } \langle \text{component number} \rangle$
 $\langle \text{token} \rangle ::= \langle \text{variable name} \rangle$
 $\langle \text{component number} \rangle ::= \langle \text{number} \rangle$

Attributes

$\langle \text{attribute} \rangle ::= \langle \text{type name} \rangle \text{' ' } \langle \text{type attribute} \rangle$
 $\quad \quad \quad \quad \quad \quad \quad \quad \langle \text{place name} \rangle \text{' ' } \langle \text{place attribute} \rangle$
 $\quad \quad \quad \quad \quad \quad \quad \quad \langle \text{expression} \rangle \text{' ' } \langle \text{container attribute} \rangle$
 $\quad \quad \quad \quad \quad \quad \quad \quad \langle \text{expression} \rangle \text{' ' } \langle \text{list attribute} \rangle$
 $\langle \text{type attribute} \rangle ::= \text{'first' } \quad \quad \quad \text{'last' } \quad \quad \quad \text{'card'}$
 $\langle \text{place attribute} \rangle ::= \text{'card' } \quad \quad \quad \text{'mult'}$
 $\langle \text{container attribute} \rangle ::= \text{'full' } \quad \quad \quad \text{'empty' } \quad \quad \quad \text{'capacity'}$
 $\quad \quad \quad \quad \quad \quad \quad \quad \text{'size' } \quad \quad \quad \text{'space'}$
 $\langle \text{list attribute} \rangle ::= \text{'first' } \quad \quad \quad \text{'first_index' } \quad \quad \quad \text{'prefix'}$
 $\quad \quad \quad \quad \quad \quad \quad \quad \text{'last' } \quad \quad \quad \text{'last_index' } \quad \quad \quad \text{'suffix'}$

Iterator

$\langle \text{iterator} \rangle ::= \langle \text{iterator type} \rangle ' (' \langle \text{iteration scheme} \rangle [\langle \text{iterator condition} \rangle] [\langle \text{iterator expression} \rangle] ') '$
 $\langle \text{iterator type} \rangle ::= \text{'forall' | 'exists' | 'card' | 'mult' | 'min' | 'max' | 'sum' | 'product'}$
 $\langle \text{iterator condition} \rangle ::= ' | ' \langle \text{expression} \rangle$
 $\langle \text{iterator expression} \rangle ::= ' : ' \langle \text{expression} \rangle$

Arc labels

$\langle \text{arc label} \rangle ::= \langle \text{complex tuple} \rangle ' + ' \langle \text{complex tuple} \rangle ^*$

Tuples

$\langle \text{complex tuple} \rangle ::= [\langle \text{tuple for} \rangle] [\langle \text{tuple guard} \rangle] [\langle \text{tuple factor} \rangle] \langle \text{tuple} \rangle$
 $\langle \text{tuple for} \rangle ::= \text{'for' } ' (' \langle \text{iteration scheme} \rangle ') '$
 $\langle \text{tuple guard} \rangle ::= \text{'if' } ' (' \langle \text{expression} \rangle ') '$
 $\langle \text{tuple factor} \rangle ::= \langle \text{expression} \rangle ' * '$
 $\langle \text{tuple} \rangle ::= \text{'< (} \langle \text{non empty expression list} \rangle ') >'$
| 'epsilon'

State propositions

$\langle \text{state proposition} \rangle ::= \text{'proposition' } \langle \text{state proposition name} \rangle ' : ' \langle \text{expression} \rangle ' ; '$
 $\langle \text{state proposition name} \rangle ::= \langle \text{name} \rangle$

2 Property specification language

Properties

$\langle \text{property specification} \rangle ::= (\langle \text{property} \rangle) ^*$
 $\langle \text{property} \rangle ::= \langle \text{state property} \rangle$
| $\langle \text{temporal property} \rangle$

State properties

$\langle \text{state property} \rangle ::= \text{'state' } \text{'property' } \langle \text{property name} \rangle ' : ' \langle \text{state property definition} \rangle$
 $\langle \text{property name} \rangle ::= \langle \text{name} \rangle$
 $\langle \text{state property definition} \rangle ::= \langle \text{reject clause} \rangle (\langle \text{accept clause} \rangle) ^*$
 $\langle \text{reject clause} \rangle ::= \text{'reject' } \langle \text{predicate} \rangle ' ; '$
 $\langle \text{accept clause} \rangle ::= \text{'accept' } \langle \text{predicate} \rangle ' ; '$
 $\langle \text{predicate} \rangle ::= \text{'deadlock'}$
| $\langle \text{state proposition name} \rangle$

Temporal properties

$\langle \text{temporal property} \rangle ::= \text{'ltl' } \text{'property' } \langle \text{property name} \rangle ' : ' \langle \text{temporal expression} \rangle ' ; '$
 $\langle \text{temporal expression} \rangle ::= \text{' (} \langle \text{temporal expression} \rangle ') '$
| 'true'
| 'false'
| $\langle \text{state proposition name} \rangle$
| $\text{'not' } \langle \text{temporal expression} \rangle$
| $\langle \text{temporal expression} \rangle \text{'or' } \langle \text{temporal expression} \rangle$
| $\langle \text{temporal expression} \rangle \text{'and' } \langle \text{temporal expression} \rangle$
| $\text{'[]' } \langle \text{temporal expression} \rangle$
| $\text{'<>' } \langle \text{temporal expression} \rangle$
| $\langle \text{temporal expression} \rangle \text{'until' } \langle \text{temporal expression} \rangle$