FSP Syntax reference

Identifiers

Identifiers starting with a lower case letter are used for action label names and range variables. Identifiers starting with an upper case letter are used for process names, range names, constant names and process parameter names.

upper_identifier ::= uppercase_letter { letter | digit }

lower_identifier ::= lowercase_letter { letter | digit }

Reserved words: **const, property, range, if, then, else, forall, when**

Predefined local process identifiers: **ERROR, STOP**

Constant Declaration

constant_declaration ::= **const** upper_identifier "**=**" integer_value

Example: `const TRUE = 1`

Range Declaration

range_declaration ::= **range** upper_identifier "**=**" *lower_bound_expression* "**..**" *upper_bound_expression*

Example: `range BIT = 0..1, range BOOL = FALSE..TRUE`

Index & Index range

index ::= "**[**" expression "**]**"

range ::= range_declaration_identifier | *lower_bound_expression* "**..**" *upper_bound_expression*

range_declaration_identifier ::= upper_identifier

index_range ::= "**[**" [lower_identifier "**:**"] range "**]**"

index_label ::= index | index_range

index_labels::= index_label { index_label }

Examples: `[1], [i * 2], [1..2], [i:0..5]`

Action Label

simple_action_label ::= lower_identifier | index_labels

action_label ::= simple_action_label { "**.**" simple_action_label }

action_label_set ::= "**{**" action_label { "**,**" action_label } "**}**"

Examples: `tick, coord[1][2], buff.out, buff.in[x:0..1], {a,b,c,d}`

A label of the form `a[0..2]` is exactly equivalent to the label set `{a[0], a[1], a[2]}`

Primitive Process

primitive_process :: = upper_identifier ["**(**" parameter_list "**)**"] "**=**" primitive_process_body

primitive_process_body ::= process_body { "**,**" local_process_defn } [alphabet_extension] [relabels] [label_visibility]"**.**"

local_process_name ::= upper_identifier [index] | **STOP** | **ERROR**

process_body ::= "**(**" choices "**)**" | local_process_name | conditional

choices ::= choice { "**|**" choice }

choice ::= [**when** *boolean_expression*] action_label_part "**->**" process_body

action_label_part ::= action_label | action_label_set

conditional ::= **if** *boolean_expression* **then** process_body [**else** process_body]

local_process_defn::= upper_identifier ["@"][index | index_range] "**=**" process_body

parameter_list ::= parameter {"," parameter }

parameter ::= upper_identifier "**=**" integer_value

Label operations

alphabet_extension ::= "**+**" action_label_set

label_visibility::= hide_label | expose_label

hide_label ::= "****" action_label_set

relabels::="**/**" relabel_set

relabel_set ::= "**{**" relabel { "**,**" relabel } "**}**"

relabel::= simple_relabel | **forall** index_range relabel_set

simple_relabel ::= action_label "**/**" action_label

Composite Process

composite_process ::=

"**||**" upper_identifier ["**(**" parameter_list "**)**"] "**=**" composite_body [relabels] [label_visibility] "**.**"

composite_body ::= process_instance | parallel_list | composite_conditional | composite_replicator

composite_replicator ::= **forall** index_range composite_body

composite_conditional ::= **if** *boolean_expression* **then** composite_body [**else** composite_body]

parallel_list ::= "**(**" composite_body { "**||**" composite_body } "**)**"

process_instance ::= [action_label_set "**::**"][action_label "**:**"] upper_identifier ["**(**" actual_parameter_list "**)**"] [relabels]

actual_parameter_list ::= expression { "**,**" expression }

Property automata

property_automata ::= **property** primitive_process

Specification

specification ::= { constant_declaration | primitive_process | composite_process |

property_automata }