A LOTOS *Vade Mecum*

K. J. Turner, University of Stirling

16th April 1996

Syntax classes are given in *italics*, keywords in **bold**. '...' denotes a repeated element (usually optional). The precedence of behaviour operators is given as 1 (loosest binding) to 8 (tightest binding).

action	**i**	(internal)
	gate_id ! *value* or ? *var_decl* ...	(observable)
	gate_id ! *value* or ? *var_decl* ... [*guard*]	(sel. pred.)
behaviour	**let** *var_decl = value, ...* **in** *behaviour*	*1*
	choice *var_decl* or *gate_decl, ...* [] *behaviour*	*1*
	par *gate_decl, ...* ||| or |[*gate_id, ...*]| or || *behaviour*	*1*
	hide *gate_id, ...* **in** *behaviour*	*1*
	behaviour ≫ *behaviour*	*2* (enables)
	behaviour ≫ **accept** *var_decl, ...* **in** *behaviour*	
	behaviour ▷ *behaviour*	*3* (disabled by)
	behaviour ||| or |[*gate_id, ...*]| or || *behaviour*	*4* (parallel)
	behaviour [] *behaviour*	*5* (choice)
	[*guard*] ⇒ *behaviour*	*6*
	action; *behaviour*	*7*
	stop	*8*
	exit	*8*
	exit (*value* or **any** *sort_id, ...*)	
	proc_id [*gate_id, ...*] (*value, ...*)	*8*
	(*behaviour*)	*8* (grouping)
definition	*behaviour* **where** *type* or *process ...*	
digit	0..9	
equation	*value = value*	(unconditional)
	[*guard*] ⇒ *value = value*	(conditional)

equations	**forall** *var_decl, ...*
	  ofsort *sort_id*
	  forall *var_decl, ...*
	  equation; *...*
	  ...	

formal_pars	[*gate_id, ...*] (*var_decl, ...*) : **noexit**	
	[*gate_id, ...*] (*var_decl, ...*) : **exit**	
	[*gate_id, ...*] (*var_decl, ...*) : **exit** (*sort_id, ...*)	
gate_decl	*gate_id, ...* **in** [*gate_id, ...*]	
guard	*value = value*	
	bool_value	(Boolean)
id	*letter*, *letter* or *digit* or *underscore ...*	(normal)
	digit, *letter* or *digit* or *underscore ...*	(operation)
	special_char ...	(operation)
letter	a..z A..Z	

lib_ type_ id	***type_ id***	***sort_ id***	***opn_ ids***
	Bit	*Bit*	*0..1 eq ge gt le lt NatNum ne*
	BitString	*BitString*	*+ ++ BitString eq Length NatNum ne Reverse*
	Boolean	*Bool*	*and eq false iff implies ne not or true xor*
	DecDigit	*DecDigit*	*0..9 eq ge gt le lt NatNum ne*
	DecString	*DecString*	*+ ++ DecString eq Length NatNum ne Reverse*
	Element	*Element*	*eq ne*
	FBoolean	*FBool*	*not true*
	HexDigit	*HexDigit*	*0..F eq ge gt le lt NatNum ne*
	HexString	*HexString*	*+ ++ eq HexString Length NatNum ne Reverse*
	NaturalNumber	*Nat*	*0 + * ** eq ge gt le lt ne Succ*
	NonEmptyString	*NonEmptyString*	*+ ++ eq Length ne Reverse String*
	OctDigit	*OctDigit*	*0..7 eq ge gt le lt NatNum ne*
	OctString	*OctString*	*+ ++ eq Length NatNum ne OctString Reverse*
	Octet	*Octet*	*Bit1..Bit8 eq ne Octet*
	OctetString	*OctetString*	*<> + ++ eq Length ne OctetString Reverse*
	Set	*Set*	*{} Card eq Includes Insert Ints IsIn IsSubsetOf Minus ne NotIn Remove Union*
	String	*String*	*<> + ++ eq Length ne Reverse String*

operation *pref_ opn_ id* or *_ inf_ opn_ id _ , ... : sort_ id, ... → sort_ id*

process
process *proc_ id formal_ pars* :=
definition
endproc

special_ char #%&*+−./<=>@\^~{}

specification
specification *spec_ id formal_ pars*
type ...
behaviour or **behavior** *definition*
endspec

type
library
lib_ type_ id, ...
endlib

type *type_ id* **is** *type_ id, ...*
formalsorts *sort_ id, ...*
formalopns *operation ...*
formaleqns *equations*
sorts *sort_ id, ...*
opns *operation ...*
eqns *equations*
endtype

type *type_ id* **is** *type_ id*
renamedby *type_ repl*
endtype

type *type_ id* **is** *type_ id*
actualizedby *type_ id, ...* **using** *type_ repl*
endtype

type_ repl
sortnames *sort_ id* **for** *sort_ id ...*
opnnames *opn_ id* **for** *opn_ id ...*

value

var_ id	
pref_ opn_ id (*value, ...*)	(prefix)
value inf_ opn_ id value	(infix)
value **of** *sort_ id*	(coercion)
(*value*)	(grouping)

var_ decl *var_ id, ... : sort_ id*