

1 Syntax of Concurrent Boolean Programs

Syntax	Description
$prog ::= decl^* proc^*$	A program is a list of global variable declarations followed by a list of procedure definitions
$decl ::= \mathbf{decl} id^+ ;$	Declaration of variables
$id ::= [a-zA-Z][a-zA-Z0-9]^*$	An identifier is a regular C-style identifier
$proc ::= type id(id^*) \mathbf{begin} enforce sseq \mathbf{end}$	Procedure definition
$type ::= \mathbf{void}$ \mathbf{bool} $\mathbf{bool} <id^+ >$	Procedures can return an arbitrary number of values
$enforce ::= \mathbf{enforce} (expr) ;$ 	Restriction on valuations of variables
$sseq ::= lstmt^+$	Sequence of statements
$lstmt ::= stmt$ $id:^+ stmt$	Labeled statement
$stmt ::= \mathbf{skip} ;$ $\mathbf{goto} id^+ ;$ $\mathbf{return} id^* ;$ $id^+ := expr^+ \mathbf{constrain} expr ;$ $\mathbf{if} (expr) \mathbf{then} sseq \mathbf{else} sseq \mathbf{fi}$ $\mathbf{assert} (expr) ;$ $\mathbf{assume} (expr) ;$ $id^* := id (expr) ;$ $\mathbf{start_thread} id ;$ $\mathbf{end_thread} ;$ $\mathbf{atomic_begin} ;$ $\mathbf{atomic_end} ;$	Nondeterministic goto Parallel assignment Conditional statement Assert statement Assume statement Procedure call Thread creation Thread termination Beginning of atomic section Ending of atomic section
$expr ::= expr binop expr$ $! expr$ $(expr)$ $const$ id $*$ $\mathbf{schoose} [expr, expr] ;$	Negation Variable Non-deterministic choice Non-deterministic choice
$binop ::= '!' '&' '^' '=' '!=' '==>'$	Logical connectives
$const ::= \mathbf{0} / \mathbf{1}$	False / True

Table 1: Syntax of concurrent Boolean Programs (adapted from [1] and extended to SATABS' syntax).

References

- [1] Ball, T., Rajamani, S.K.: Bebop: A symbolic model checker for Boolean programs. In: Model Checking and Software Verification (SPIN). Volume 1885 of LNCS., Springer (2000) 113–130