Module-Level Constructs

\textbf{MODULE} \textit{M}

\textbf{BEGINs} the module or submodule named \textit{M}.

\textbf{EXTENDS} \textit{M}_1, \ldots, \textit{M}_n

\textbf{INCORPORATES} the declarations, definitions, assumptions, and theorems from the modules named \textit{M}_1, \ldots, \textit{M}_n \textbf{INTO} the current module.

\textbf{CONSTANTS} \textit{C}_1, \ldots, \textit{C}_n \textbf{(1)}

\textbf{Declares} the \textit{C}_j \textbf{to be constant parameters (rigid variables)}. Each \textit{C}_j \textbf{is either an identifier or has the form} \textit{C}(_{-}, \ldots, _{-}) \textbf{, the latter form indicating that} \textit{C} \textbf{is an operator with the indicated number of arguments}.

\textbf{VARIABLES} \textit{x}_1, \ldots, \textit{x}_n \textbf{(1)}

\textbf{Declares} the \textit{x}_j \textbf{to be variables (parameters that are flexible variables)}.

\textbf{ASSUME} \textit{P}

\textbf{Asserts} \textit{P} \textbf{as an assumption}.

\textbf{\textit{F}(x}_1, \ldots, \textit{x}_n) \overset{\Delta}{=} \textit{exp}

\textbf{Deﬁnes} \textit{F} \textbf{to be the operator} \textbf{such that} \textit{F(e}_1, \ldots, \textit{e}_n) \textbf{equals} \textit{exp} \textbf{with each identifier} \textit{x}_k \textbf{replaced by} \textit{e}_k. \textbf{(For} \textit{n} = 0, \textbf{it is written} \textit{F} \overset{\Delta}{=} \textit{exp})

\textbf{\textit{f}[x \in S] \overset{\Delta}{=} \textit{exp} \textbf{(2)}}

\textbf{Deﬁnes} \textit{f} \textbf{to be the function} \textbf{with domain} \textit{S} \textbf{such that} \textit{f[x] = exp} \textbf{for all} \textbf{in} \textit{S}. \textbf{(The symbol} \textit{f} \textbf{may occur in} \textit{exp}, \textbf{allowing a recursive deﬁnition}.)

---

\textbf{(1)} \textbf{The terminal} \textit{s} \textbf{in the keyword is optional}.

\textbf{(2)} \textbf{\textit{x} \in S} \textbf{may be replaced by a comma-separated list of items} \textit{v} \textbf{\in S}, \textbf{where} \textit{v} \textbf{is either a comma-separated list or a tuple of identifiers}.
INSTANCE $M$ WITH $p_1 \leftarrow e_1, \ldots, p_m \leftarrow e_m$

For each defined operator $F$ of module $M$, this defines $F$ to be the operator whose definition is obtained from the definition of $F$ in $M$ by replacing each declared constant or variable $p_j$ of $M$ with $e_j$. (If $m = 0$, the WITH is omitted.)

$N(x_1, \ldots, x_n) \triangleq \text{INSTANCE } M \text{ WITH } p_1 \leftarrow e_1, \ldots, p_m \leftarrow e_m$

For each defined operator $F$ of module $M$, this defines $N(d_1, \ldots, d_n)!F$ to be the operator whose definition is obtained from the definition of $F$ by replacing each declared constant or variable $p_j$ of $M$ with $e_j$, and then replacing each identifier $x_k$ with $d_k$. (If $m = 0$, the WITH is omitted.)

THEOREM $P$

Asserts that $P$ can be proved from the definitions and assumptions of the current module.

LOCAL def

Makes the definition(s) of def (which may be a definition or an INSTANCE statement) local to the current module, thereby not obtained when extending or instantiating the module.

Ends the current module or submodule.
The Constant Operators

Logic
\[\land \lor \neg \Rightarrow \equiv\]
TRUE FALSE BOOLEAN [the set \{TRUE, FALSE\}]
\[\forall x : p \ \exists x : p \ \forall x \in S : p \quad (1) \quad \exists x \in S : p \quad (1)\]
CHOOSE \(x : p\) [An \(x\) satisfying \(p\)] CHOOSE \(x \in S : p\) [An \(x\) in \(S\) satisfying \(p\)]

Sets
\[\equiv \neq \in \notin \cup \cap \subseteq \setminus\] \{e_1, \ldots, e_n\} [Set consisting of elements \(e_i\)]
\{x \in S : p\} \quad (2) [Set of elements \(x\) in \(S\) satisfying \(p\)]
\{e : x \in S\} \quad (1) [Set of elements \(e\) such that \(x\) in \(S\)]
SUBSET \(S\) [Set of subsets of \(S\)]
UNION \(S\) [Union of all elements of \(S\)]

Functions
\(f[e]\) [Function application]
DOMAIN \(f\) [Domain of function \(f\)]
\[x \in S \mapsto e\] \quad (1) [Function \(f\) such that \(f[x] = e\) for \(x \in S\)]
\[S \rightarrow T\] [Set of functions \(f\) with \(f[x] \in T\) for \(x \in S\)]
\[f\ EXCEPT\ ![e_1] = e_2\] \quad (3) [Function \(\hat{f}\) equal to \(f\) except \(\hat{f}[e_1] = e_2\). An @ in \(e_2\) equals \(f[e_1]\).]

Records
\(e, h\) [The \(h\)-component of record \(e\)]
\[h_1 \mapsto e_1, \ldots, h_n \mapsto e_n\] [The record whose \(h_i\) component is \(e_i\)]
\[h_1 : S_1, \ldots, h_n : S_n\] [Set of all records with \(h_i\) component in \(S_i\)]
\[r \ EXCEPT\ !.h = e\] \quad (3) [Record \(\hat{r}\) equal to \(r\) except \(\hat{r}.h = e\). An @ in \(e\) equals \(r.h\).]

Tuples
\(e[i]\) [The \(i^{th}\) component of tuple \(e\)]
\[e_1, \ldots, e_n\] [The \(n\)-tuple whose \(i^{th}\) component is \(e_i\)]
\(S_1 \times \ldots \times S_n\) [The set of all \(n\)-tuples with \(i^{th}\) component in \(S_i\)]

Strings and Numbers
"c_1 \ldots c_n" [A literal string of \(n\) characters]
STRING [The set of all strings]
d_1 \ldots d_n \ d_1 \ldots d_n \cdot d_{n+1} \ldots d_m [Numbers (where the \(d_i\) are digits)]

(1) \(x \in S\) may be replaced by a comma-separated list of items \(v \in S\), where \(v\) is either a comma-separated list or a tuple of identifiers.
(2) \(x\) may be an identifier or tuple of identifiers.
(3) \![e_1]\) or \!\.h may be replaced by a comma separated list of items \!a_1 \ldots a_n, where each \(a_i\) is \![e_i]\) or \!.h.
Miscellaneous Constructs

- **IF** $p$ **THEN** $e_1$ **ELSE** $e_2$  
  \[e_1 \text{ if } p \text{ true, else } e_2\]
- **CASE** $p_1 \rightarrow e_1$ **□** ... **□** $p_n \rightarrow e_n$  
  \[\text{Some } e_i \text{ such that } p_i \text{ true}\]
- **CASE** $p_1 \rightarrow e_1$ **□** ... **□** $p_n \rightarrow e_n$ **□** OTHER $\rightarrow e$  
  \[\text{Some } e_i \text{ such that } p_i \text{ true, or } e \text{ if all } p_i \text{ are false}\]

**LET** $d_1 \triangleq e_1$ ... $d_n \triangleq e_n$ **IN** $e$  
\[e \text{ in the context of the definitions}\]

\[\land \ p_1 \ \text{[the conjunction } p_1 \land \ldots \land p_n] \quad \lor \ p_1 \ \text{[the disjunction } p_1 \lor \ldots \lor p_n] \]
\[; \quad ; \]
\[\land \ p_n \quad \lor \ p_n \]

### The Action Operators

- $e'$  
  \[\text{The value of } e \text{ in the final state of a step}\]
- $[A]_e$  
  \[A \lor (e' = e)\]
- $\langle A \rangle_e$  
  \[A \land (e' \neq e)\]
- **ENABLED** $A$  
  \[\text{An } A \text{ step is possible}\]
- **UNCHANGED** $e$  
  \[e' = e\]
- $A \cdot B$  
  \[\text{Composition of actions}\]

### The Temporal Operators

- $\Box F$  
  \[F \text{ is always true}\]
- $\Diamond F$  
  \[F \text{ is eventually true}\]
- $WF_e(A)$  
  \[\text{Weak fairness for action } A\]
- $SF_e(A)$  
  \[\text{Strong fairness for action } A\]
- $F \leadsto G$  
  \[F \text{ leads to } G\]
- $F \guarantees G$  
  \[F \text{ guarantees } G \text{ (an assumption/guarantee specification)}\]
- $\exists x : F$  
  \[\text{Temporal existential quantification (hiding)}\]
- $\forall x : F$  
  \[\text{Temporal universal quantification}\]
User-Definable Operator Symbols

Infix Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>-</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>*</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>/</td>
<td>(2) Defined by the Reals module.</td>
</tr>
<tr>
<td>◦</td>
<td>(3) Defined by the Sequences module.</td>
</tr>
<tr>
<td>⊕</td>
<td>(5) Defined by the Bags module.</td>
</tr>
<tr>
<td>⊖</td>
<td>(5) Defined by the Bags module.</td>
</tr>
<tr>
<td>⊗ ⊘ ⊙ ∗</td>
<td>(5) Defined by the Bags module.</td>
</tr>
<tr>
<td>&lt;</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>&gt;</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>≪ ≫</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>≦ ≧ ⪯ ⪰</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>≡ ≍ . . =</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>≃ ≈ ∼ =</td>
<td>(1) Defined by the Naturals, Integers, and Reals modules.</td>
</tr>
<tr>
<td>≊≀ ⊎ ⃝</td>
<td>(5) Defined by the Bags module.</td>
</tr>
<tr>
<td>! ! @@</td>
<td>(6) Defined by the TLC module.</td>
</tr>
</tbody>
</table>

Postfix Operators (7)

|^| ^*| ^#|

Prefix Operator

- (8)

(1) Defined by the Naturals, Integers, and Reals modules.
(2) Defined by the Reals module.
(3) Defined by the Sequences module.
(4) $x^y$ is printed as $x^y$.
(5) Defined by the Bags module.
(6) Defined by the TLC module.
(7) $e^+$ is printed as $e^+$, and similarly for $^*$ and $^#$.
(8) Defined by the Integers and Reals modules.
ASCII Representations of Symbols

∧ /\ or \land  ∨ \lor \lor \lor \lor ⇒ =⇒
¬ ~ \lnot or \neg ≡ <-> or \equiv \equiv
∈ \in \notin \notin \not \# or /=
〈 << \notgg \gg ◊ <>
≤ \leq \noteq \>= ~⇒ ~>
\ll \lll \llll \lllll
\prec \succ \succsim \succneqq
\preceq \succeq \div \\div
\subseteq \supseteq ∩ \cap \\cap \\cap \\cap
\subset \supset \cap \cap \cap \cap
\sqsubset \sqsupset \sqsubset \sqsupset \sqsubset \sqsupset
\implies \rightarrow \rightarrow \rightarrow
\bigcirc ∠ \angle \\angle \\angle
\models \models \models \models
\iff \iff \iff \iff
\forall \mathcal{E} \mathcal{E} \mathcal{E} \mathcal{E} \mathcal{E} \mathcal{E} \mathcal{E} \mathcal{E}
\\
(1) s is a sequence of characters.
(2) x and y are any expressions.
(3) a sequence of four or more - or = characters.
The Most Common Standard Modules

Modules Naturals, Integers, Reals

Define + − ∗ / ∨ ∧ . . Nat Real

÷ % ≤ ≥ < > Int Infinity

Prefix − is not defined in Naturals.

$a^b$ denotes $a^b$.

Nat, Int, and Real are the sets of naturals, integers, and real numbers.

$a \ldots b$ equals $\{n \in \text{Int} : a \leq n \leq b\}$.

$a \% b$ equals $a \mod b$, defined so $0 \leq a \% b < b$, if $b$ is a positive integer.

$\div$ is defined so $a = b \times (a \div b) + (a \% b)$ for $a$ and $b$ integers with $b > 0$.

/ (division) is defined only in Reals.

Infinity is defined in Reals so $-\text{Infinity} < r < \text{Infinity}$ for all $r \in \text{Real}$.

Module Sequences

Defines ◦ Head SelectSeq SubSeq

Append Len Seq Tail

The tuple/sequence $\langle e_1, \ldots, e_n \rangle$ equals the function $[i \in 1 \ldots n \mapsto e_i]$.

$s \circ t$ is the concatenation of sequences $s$ and $t$.

$\text{Append}(\langle e_1, \ldots, e_n \rangle, e_{n+1}) = \langle e_1, \ldots, e_{n+1} \rangle$

$\text{Head}(\langle e_1, \ldots, e_n \rangle) = e_1$

$\text{Tail}(\langle e_1, \ldots, e_n \rangle) = \langle e_2, \ldots, e_n \rangle$

$\text{Len}(\langle e_1, \ldots, e_n \rangle) = n$

$\text{Seq}(S)$ is the set of all finite sequences of elements of $S$.

$\text{SubSeq}(\langle e_1, \ldots, e_n \rangle, j, k) = \langle e_j, \ldots, e_k \rangle$

$\text{SelectSeq}(s, \text{Test})$ is the subsequence of elements $e$ of $s$ satisfying $\text{Test}(e)$.

Module FiniteSets

Defines IsFiniteSet Cardinality

IsFiniteSet($S$) is true iff $S$ is a finite set.

Cardinality($S$) is the number of elements in $S$, if $S$ is a finite set.