GhostML : a mini-ML with global references and ghost terms

prog	::=	*typedecl** *vardecl** *t*	program
typedecl	::=	`type` *id* $\alpha, ..., \alpha$ $= \tau$	type declaration
vardecl	::=	`val` *id* : `ref` τ	global reference declaration

GhostML Programs

τ	::=	α	type variable
	|	$\varepsilon\ (\tau, \ldots, \tau)$	datatype constructor
	|	$\tau \to \tau$	function type
	|	`int` | `bool` | `Prop` | ...	build-in types
σ	::=	$\forall\overline{\alpha}.\tau$	type scheme

GhostML Types and Schemes

v	::=	x	variable
	|	op	build-in constants and operands (*1, true,* +, $\vee$, ...)
	|	$C(v, \ldots, v)$	constructor application
	|	`fun` $x \to t$	function

GhostML Values

t	::=	v	value
	|	$t(t)$	application
	|	$C(t, \ldots, t)$	constructor application
	|	`let` $x = t$ `in` t	local binding
	|	`letrec` $f\ x = t$	recursive function
	|	$!\ x$	global reference access
	|	$x := t$	global reference assignment
	|	if t then t else t	conditional expression
	|	match t with $p \to t,\ \ldots,\ p \to t$ end	pattern-matching

GhostML Terms

p	::=	x	variable pattern
	|	$C(p, \ldots, p)$	constructor pattern

GhostML Patterns

Operational Semantics

$$\frac{t_1|_\sigma \rightsquigarrow t_1'|_{\sigma'}}{t_1\ t_2|_\sigma \rightsquigarrow t_1'\ t_2|_{\sigma'}} \qquad \text{(E-AppLeft)}$$

$$\frac{t_2|_\sigma \rightsquigarrow t_2'|_{\sigma'}}{v\ t_2|_\sigma \rightsquigarrow v\ t_2'|_{\sigma'}} \qquad \text{(E-AppRight)}$$

$$(\textbf{fun}\ \varepsilon x_\tau \to t)v|_\sigma \rightsquigarrow t[x \leftarrow\!\shortmid v|_\sigma \qquad \textbf{(E-AppRedex)}$$