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

GhostML Types and Schemes

v	::=	x	variable
	|	c	build-in constant (*1, 2, ..., true, false, ... True, ...*)
	|	op	build-in operand $(+, \times, \wedge, \vee, ...)$
	|	$C(v, \ldots, v)$	constructor application
	|	`fun` $x \to t$	function

GhostML Values

t	::=	v	value
	|	$t(t)$	application
	|	$t; t$	sequence
	|	`let`$x = t$ `in` t	local binding
	|	`letrec`$f\,x = t$	recursive function
	|	`ghost` t	ghost term
	|	$!\,x$	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