Oat v. 1 Language Specification

CIS341 – Steve Zdancewic

February 22, 2026

1 Grammar

The following grammar defines the Oat syntax. All binary operations are *left associative* with precedence levels indicated numerically. Higher precedence operators bind tighter than lower precedence ones.

prog	::=		prog
	|	$decl_1 \mathinner{..} decl_i$	
decl	::=		global declarations
	|	*gdecl*	
	|	*fdecl*	
gdecl	::=		global variable declarations
	|	`global` *id* `=` *gexp* `;`	
arg	::=		arg
	|	*t id*	
args	::=		args
	|	$arg_1, \mathinner{..}, arg_n$	
fdecl	::=		function declaration
	|	*retty id*`(`*args*`)` *block*	
block	::=		blocks
	|	`{`$stmt_1 \mathinner{..} stmt_n$`}`	
t	::=		types
	|	`int`	
	|	`bool`	
	|	*ref*	
ref	::=		reference types
	|	`string`	
	|	*t*`[]`	
	|	`(`$t_0, \mathinner{..}, t_n$`)` `->` *retty*	function pointer

retty	::=		return types
	|	`void`	
	|	*t*	
bop	::=		(left associative) binary operations
	|	`*`	multiplication (precedence 100)
	|	`+`	addition (precedence 90)
	|	`-`	subtraction (precedence 90)
	|	`<<`	shift left (precedence 80)
	|	`>>`	shift right logical (precedence 80)
	|	`>>>`	shift right arithmetic (precedence 80)
	|	`<`	less-than (precedence 70)
	|	`<=`	less-than or equal (precedence 70)
	|	`>`	greater-than (precedence 70)
	|	`>=`	greater-than or equal (precedence 70)
	|	`==`	equal (precedence 60)
	|	`!=`	not equal (precedence 60)
	|	`&`	logical and (precedence 50)
	|	`|`	logical or (precedence 40)
	|	`[&]`	bit-wise and (precedence 30)
	|	`[|]`	bit-wise or (precedence 20)
uop	::=		unary operations
	|	`-`	
	|	`!`	
	|	`~`	
gexp	::=		global initializers
	|	*integer*	64-bit integer literals
	|	*string*	C-style strings
	|	*ref* `null`	
	|	`true`	
	|	`false`	
	|	`new` *t* `[]{`$gexp_1$`,` .. `,`$gexp_n$`}`	
lhs	::=		left-hand-sides for assignment
	|	*id*	
	|	exp_1 `[`exp_2`]`	

exp	::=		expressions
	|	*integer*	64-bit integer literals
	|	*string*	C-style strings
	|	*ref* `null`	
	|	`true`	
	|	`false`	
	|	*lhs*	left-hand-side as an expression
	|	*id*`(`exp_1`, .. ,`exp_n`)`	function call
	|	`new` *t* `[]{`exp_1`, .. ,`exp_n`}`	Explicitly initialized array
	|	`new int [`exp_1`]`	Default-initialize int array
	|	`new bool [`exp_1`]`	Default-initialize bool array
	|	exp_1 *bop* exp_2	
	|	*uop exp*	
	|	`(`*exp*`)`	
vdecl	::=		local declarations
	|	`var` *id* `=` *exp*	
vdecls	::=		decl list
	|	$vdecl_1$`, .. ,`$vdecl_n$	
stmt	::=		statements
	|	*lhs* `=` *exp*`;`	
	|	*vdecl*`;`	
	|	`return` *exp*`;`	
	|	`return ;`	
	|	*id*`(`exp_1`, .. ,`exp_n`);`	
	|	*if_stmt*	
	|	`for(`*vdecls*`;` exp_{opt}`;` $stmt_{opt}$`)` *block*	
	|	`while(`*exp*`)` *block*	
if_stmt	::=		if statements
	|	`if(`*exp*`)` *block else_stmt*	
else_stmt	::=		else
	|	ϵ	
	|	`else` *block*	
	|	`else` *if_stmt*	

2 Typing Rules

$\boxed{\vdash bop_1, .., bop_i : retty}$

$$\frac{}{\vdash \texttt{+}, \texttt{*}, \texttt{-}, \texttt{<<}, \texttt{>>}, \texttt{>>>}, \texttt{[\&]}, \texttt{[|]} : (\texttt{int}, \texttt{int})\ \texttt{->}\ \texttt{int}} \quad \text{TYP\_INTOPS}$$

$$\frac{}{\vdash \texttt{==}, \texttt{!=}, \texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=} : (\texttt{int}, \texttt{int})\ \texttt{->}\ \texttt{bool}} \quad \text{TYP\_CMPOPS}$$

$$\frac{}{\vdash \texttt{\&}, \texttt{|} : (\texttt{bool}, \texttt{bool})\ \texttt{->}\ \texttt{bool}} \quad \text{TYP\_BOOLOPS}$$

$\boxed{\vdash uop : retty}$

$$\frac{}{\vdash \texttt{!} : (\texttt{bool})\ \texttt{->}\ \texttt{bool}} \quad \text{TYP\_LOGNOT}$$

$$\frac{}{\vdash \texttt{\~{}} : (\texttt{int})\ \texttt{->}\ \texttt{int}} \quad \text{TYP\_BITNEG}$$

$$\frac{}{\vdash \texttt{-} : (\texttt{int})\ \texttt{->}\ \texttt{int}} \quad \text{TYP\_NEG}$$

$\boxed{G; L \vdash_{lhs} lhs : t}$

$$\frac{x \notin L \quad x : t \in G \quad x \textit{ does not resolve to a global function}}{G; L \vdash_{lhs} x : t} \quad \text{TYP\_GLOBAL}$$

$$\frac{x : t \in L}{G; L \vdash_{lhs} x : t} \quad \text{TYP\_LOCAL}$$

$$\frac{G; L \vdash exp_1 : t\texttt{[]} \quad G; L \vdash exp_2 : \texttt{int}}{G; L \vdash_{lhs} exp_1\texttt{[}exp_2\texttt{]} : t} \quad \text{TYP\_INDEX}$$

$\boxed{G; L \vdash exp : t}$

$$\frac{}{G; L \vdash n : \texttt{int}} \quad \text{TYP\_INT}$$

$$\frac{}{G; L \vdash s : \texttt{string}} \quad \text{TYP\_STRING}$$

$$\frac{}{G; L \vdash \texttt{string}\,\texttt{null} : \texttt{string}} \quad \text{TYP\_NULLSTR}$$

$$\frac{}{G; L \vdash t\texttt{[]}\ \texttt{null} : t\texttt{[]}} \quad \text{TYP\_NULLARR}$$

$$\frac{}{G; L \vdash \texttt{true} : \texttt{bool}} \quad \text{TYP\_TRUE}$$

$$\frac{}{G; L \vdash \texttt{false} : \texttt{bool}} \quad \text{TYP\_FALSE}$$

$$\frac{G; L \vdash_{lhs} lhs : t}{G; L \vdash lhs : t} \quad \text{TYP\_LHS}$$

$$\frac{f : (t_1, .., t_i)\ \texttt{->}\ t \in G \quad G; L \vdash exp_1 : t_1 \quad .. \quad G; L \vdash exp_i : t_i}{G; L \vdash f(exp_1, .., exp_i) : t} \quad \text{TYP\_CALL}$$

$$\frac{G; L \vdash exp_1 : t \quad .. \quad G; L \vdash exp_i : t}{G; L \vdash \texttt{new}\ t\texttt{[]}\{exp_1, .., exp_i\} : t\texttt{[]}} \quad \text{TYP\_ARRLIT}$$

$$\frac{G; L \vdash exp_1 : \texttt{int}}{G; L \vdash \texttt{new}\ \texttt{int}\ [exp_1] : t\texttt{[]}} \quad \text{TYP\_ARRZEROINT}$$

$$\frac{G ; L \vdash exp_1 : \texttt{int}}{G ; L \vdash \texttt{new}\ \texttt{bool}\ [exp_1] : t[]} \quad \textsc{typ\_arrzerobool}$$

$$\frac{\vdash bop : (t_1 , t_2)\ \texttt{->}\ t \quad G ; L \vdash exp_1 : t_1 \quad G ; L \vdash exp_2 : t_2}{G ; L \vdash exp_1\ bop\ exp_2 : t} \quad \textsc{typ\_bop}$$

$$\frac{\vdash uop : (t)\ \texttt{->}\ t \quad G ; L \vdash exp : t}{G ; L \vdash uop\ exp : t} \quad \textsc{typ\_uop}$$

$\boxed{G;L_1 \vdash vdecl \Rightarrow L_2}$

$$\frac{G;L \vdash exp:t \quad x \notin L}{G;L \vdash \texttt{var}\ x = exp \Rightarrow L, x:t}\ \textsc{typ\_decl}$$

$\boxed{G;L_0 \vdash vdecls \Rightarrow L_i}$

$$\frac{G;L_0 \vdash vdecl_1 \Rightarrow L_1 \quad \ldots \quad G;L_{i-1} \vdash vdecl_i \Rightarrow L_i}{G;L_0 \vdash vdecl_1, .., vdecl_i \Rightarrow L_i}\ \textsc{typ\_vdecls}$$

$\boxed{G;L_1;retty \vdash stmt \Rightarrow L_2}$

$$\frac{G;L_1 \vdash vdecl \Rightarrow L_2}{G;L_1;t \vdash vdecl; \Rightarrow L_2}\ \textsc{typ\_sdecl}$$

$$\frac{G;L \vdash lhs:t \quad G;L \vdash exp_2:t}{G;L;t \vdash lhs = exp_2; \Rightarrow L}\ \textsc{typ\_assn}$$

$$\frac{f:(t_1, .., t_i)\ \texttt{->}\ \texttt{void} \in G \quad G;L \vdash exp_1:t_1 \quad .. \quad G;L \vdash exp_i:t_i}{G;L;t \vdash f(exp_1, .., exp_i); \Rightarrow L}\ \textsc{typ\_scall}$$

$$\frac{G;L \vdash exp:\texttt{bool} \quad G;L;t \vdash block_1 \quad G;L;t \vdash block_2}{G;L;t \vdash \texttt{if}(exp)\ block_1\ \texttt{else}\ block_2 \Rightarrow L}\ \textsc{typ\_if}$$

$$\frac{G;L \vdash exp:\texttt{bool} \quad G;L;t \vdash block}{G;L;t \vdash \texttt{while}(exp)\ block \Rightarrow L}\ \textsc{typ\_while}$$

$$\frac{G;L_1 \vdash vdecls \Rightarrow L_2 \quad G;L_2 \vdash exp:\texttt{bool} \quad G;L_2;t \vdash stmt \Rightarrow L_3 \quad G;L_2;t \vdash block}{G;L_1;t \vdash \texttt{for}(vdecls;\ exp_{opt};\ stmt_{opt})\ block \Rightarrow L_1}\ \textsc{typ\_for}$$

$$\frac{G;L \vdash exp:t}{G;L;t \vdash \texttt{return}\ exp; \Rightarrow L}\ \textsc{typ\_retT}$$

$$\frac{}{G;L;\texttt{void} \vdash \texttt{return}; \Rightarrow L}\ \textsc{typ\_retVoid}$$

$\boxed{G;L;t \vdash block}$

$$\frac{G;L_0;t \vdash stmt_1 .. stmt_i \Rightarrow L_i}{G;L_0;t \vdash \{stmt_1 .. stmt_i\}}\ \textsc{typ\_block}$$

$\boxed{G;L_0;t \vdash stmt_1 .. stmt_i \Rightarrow L_i}$

$$\frac{G;L_0;t \vdash stmt_1 \Rightarrow L_1 \quad \ldots \quad G;L_{i-1};t \vdash stmt_i \Rightarrow L_i}{G;L_0;t \vdash stmt_1 .. stmt_i \Rightarrow L_i}\ \textsc{typ\_stmts}$$

$\boxed{G_0 \vdash decl \Rightarrow G_1}$

$$\frac{x \notin G \quad \cdot\,;\cdot \vdash gexp : t}{G \vdash \texttt{global}\ x = gexp\,;\ \Rightarrow G, x : t}\ \text{TYP\_VDECL}$$

$$\frac{f \notin G}{G \vdash t\ f\,(t_1\,x_1, \mathinner{..}, t_i\,x_i)\ block \Rightarrow G, f : (t_1, \mathinner{..}, t_i)\ \texttt{->}\ t}\ \text{TYP\_FDECL}$$

$\boxed{G_0 \vdash decl_1 \mathinner{..} decl_i \Rightarrow G_i}$

$$\frac{G_0 \vdash decl_1 \Rightarrow G_1 \quad \ldots \quad G_{i-1} \vdash decl_i \Rightarrow G_i}{G_0 \vdash decl_1 \mathinner{..} decl_i \Rightarrow G_i}\ \text{TYP\_GLOBAL\_CTXT}$$

$\boxed{G \vdash decl}$

$$\frac{G\,; x_1 : t_1, \mathinner{..}, x_i : t_i\,; t \vdash block}{G \vdash t\ f\,(t_1\,x_1, \mathinner{..}, t_i\,x_i)\ block}\ \text{TYP\_GFUN}$$

$$\frac{\cdot\,;\cdot \vdash gexp : t}{G \vdash \texttt{global}\ x = gexp\,;}\ \text{TYP\_GVAR}$$

$\boxed{\vdash prog}$

$$\frac{G_0 \vdash decl_1 \mathinner{..} decl_i \Rightarrow G \quad G \vdash decl_1 \quad \mathinner{..} \quad G \vdash decl_i}{\vdash decl_1 \mathinner{..} decl_i}\ \text{TYP\_PROG}$$