Oat v.1 Language Specification

EECS 483

January 5, 2024

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
	|	$decl_1 \,..\, decl_i$	
decl	::=		global declarations
	|	*gdecl*	
	|	*fdecl*	
gdecl	::=		global variable declarations
	|	`global` *id* = *gexp* `;`	
arg	::=		arg
	|	*t id*	
args	::=		args
	|	$arg_1, \,..\,, arg_n$	
fdecl	::=		function declaration
	|	*retty id*(*args*) *block*	
block	::=		blocks
	|	`{`$stmt_1 \,..\, stmt_n$`}`	
t	::=		types
	|	`int`	
	|	`bool`	
	|	*ref*	

ref	::=		reference types
	|	`string`	
	|	*t*`[]`	

F	::=		function types
	|	$(t_0, .., t_n) \rightarrow$ *retty*	
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	::=		lhs expressions
	|	*id*	
	|	exp_1 `[`exp_2`]`	

exp	::=		expressions
	|	*id*	
	|	*integer*	64-bit integer literals
	|	*string*	C-style strings
	|	*ref* `null`	
	|	`true`	
	|	`false`	
	|	exp_1 `[`exp_2`]`	
	|	id`(`exp_1`, .. ,`exp_n`)`	
	|	`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

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

$$\frac{}{\vdash +, *, -, <<, >>, >>>, [\&], [|] : (\texttt{int}, \texttt{int}) \to \texttt{int}} \textsc{typ\_intOps}$$

$$\frac{}{\vdash ==, !=, <, <=, >, >= : (\texttt{int}, \texttt{int}) \to \texttt{bool}} \textsc{typ\_cmpOps}$$

$$\frac{}{\vdash \&, | : (\texttt{bool}, \texttt{bool}) \to \texttt{bool}} \textsc{typ\_boolOps}$$

$\boxed{\vdash uop : F}$

$$\frac{}{\vdash ! : (\texttt{bool}) \to \texttt{bool}} \textsc{typ\_lognot}$$

$$\frac{}{\vdash \sim : (\texttt{int}) \to \texttt{int}} \textsc{typ\_bitneg}$$

$$\frac{}{\vdash - : (\texttt{int}) \to \texttt{int}} \textsc{typ\_neg}$$

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

$$\frac{x : t \in L}{G; L \vdash x : t} \textsc{typ\_local}$$

$$\frac{}{G; L \vdash n : \texttt{int}} \textsc{typ\_int}$$

$$\frac{}{G; L \vdash s : \texttt{string}} \textsc{typ\_string}$$

$$\frac{}{G; L \vdash \texttt{string null} : \texttt{string}} \textsc{typ\_nullStr}$$

$$\frac{}{G; L \vdash t\texttt{[] null} : t\texttt{[]}} \textsc{typ\_nullArr}$$

$$\frac{}{G; L \vdash \texttt{true} : \texttt{bool}} \textsc{typ\_true}$$

$$\frac{}{G; L \vdash \texttt{false} : \texttt{bool}} \textsc{typ\_false}$$

$$\frac{G; L \vdash exp_1 : t\texttt{[]} \quad G; L \vdash exp_2 : \texttt{int}}{G; L \vdash exp_1\texttt{[}exp_2\texttt{]} : t} \textsc{typ\_index}$$

$$\frac{f : (t_1, .., t_i) \to t \in G \quad G; L \vdash exp_1 : t_1 \quad .. \quad G; L \vdash exp_i : t_i}{G; L \vdash f(exp_1, .., exp_i) : t} \textsc{typ\_call}$$

$$\frac{G; L \vdash exp_1 : t \quad .. \quad G; L \vdash exp_i : t}{G; L \vdash \texttt{new}\, t\texttt{[]\{}exp_1, .., exp_i\texttt{\}} \ : t\texttt{[]}} \textsc{typ\_arrlit}$$

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

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

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

$$\frac{\vdash uop : (t) \to t \quad G; L \vdash exp : t}{G; L \vdash uop\, exp : t} \textsc{typ\_uop}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{x \notin G \quad \cdot;\cdot \vdash gexp : t}{G \vdash \texttt{global}\ x = gexp; \Rightarrow G, x:t}\ \textsc{typ\_vdecl}$$

$$\frac{f \notin G}{G \vdash t\, f(t_1\, x_1, .., t_i\, x_i)\ block \Rightarrow G, f:(t_1, .., t_i) \to t}\ \textsc{typ\_fdecl}$$

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

$$\frac{G_0 \vdash decl_1 \Rightarrow G_1 \quad \ldots \quad G_{i-1} \vdash decl_i \Rightarrow G_i}{G_0 \vdash decl_1 .. decl_i \Rightarrow G_i}\ \textsc{typ\_global\_ctxt}$$

$\boxed{G \vdash decl}$

$$\frac{G; x_1:t_1, .., x_i:t_i; t \vdash block}{G \vdash t\, f(t_1\, x_1, .., t_i\, x_i)\ block}\ \textsc{typ\_gfun}$$

$$\frac{\cdot;\cdot \vdash gexp : t}{G \vdash \texttt{global}\ x = gexp;}\ \textsc{typ\_gvar}$$

$\boxed{\vdash prog}$

$$\frac{G_0 \vdash decl_1 .. decl_i \Rightarrow G \quad G \vdash decl_1 \quad .. \quad G \vdash decl_i}{\vdash decl_1 .. decl_i}\ \textsc{typ\_prog}$$