

Oat v. 1 Language Specification

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
		$(t_0, .., t_n) \rightarrow retty$

<i>retty</i>	::= void <i>t</i>	return types
<i>bop</i>	::= * + - << >> >>> < <= > >= == != & [&] []	(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)
<i>uop</i>	::= - ! ~	unary operations
<i>gexp</i>	::= <i>integer</i> <i>string</i> <i>ref null</i> true false new <i>t</i> [] { <i>gexp</i> ₁ , ..., <i>gexp</i> _{<i>n</i>} }	global initializers 64-bit integer literals C-style strings
<i>lhs</i>	::= <i>id</i> <i>exp</i> ₁ [<i>exp</i> ₂]	lhs expressions

<i>exp</i>	<pre> ::= <i>id</i> <i>integer</i> <i>string</i> <i>ref null</i> <i>true</i> <i>false</i> <i>exp</i>₁ [<i>exp</i>₂] <i>id</i>(<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}) <i>new t</i> [] {<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}} <i>new int</i> [<i>exp</i>₁] <i>new bool</i> [<i>exp</i>₁] <i>exp</i>₁ <i>bop</i> <i>exp</i>₂ <i>uop exp</i> (<i>exp</i>) </pre>	<p>expressions</p> <p>64-bit integer literals C-style strings</p> <p>Explicitly initialized array Default-initialize int array Default-initialize bool array</p>
<i>vdecl</i>	<pre> ::= <i>var id = exp</i> </pre>	<p>local declarations</p>
<i>vdecls</i>	<pre> ::= <i>vdecl</i>₁, .., <i>vdecl</i>_{<i>n</i>} </pre>	<p>decl list</p>
<i>stmt</i>	<pre> ::= <i>lhs = exp</i>; <i>vdecl</i>; <i>return exp</i>; <i>return</i> ; <i>id</i>(<i>exp</i>₁, .., <i>exp</i>_{<i>n</i>}); <i>if_stmt</i> <i>for</i>(<i>vdecls</i>; <i>exp</i>_{opt}; <i>stmt</i>_{opt}) <i>block</i> <i>while</i>(<i>exp</i>) <i>block</i> </pre>	<p>statements</p>
<i>if_stmt</i>	<pre> ::= <i>if</i>(<i>exp</i>) <i>block</i> <i>else_stmt</i> </pre>	<p>if statements</p>
<i>else_stmt</i>	<pre> ::= ϵ <i>else block</i> <i>else if_stmt</i> </pre>	<p>else</p>

2 Typing Rules

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

$\frac{}{\vdash +, *, -, \ll, \gg, \ggg, [\&], [!]: (\text{int}, \text{int}) \rightarrow \text{int}}$ TYP_INTOPS

$\frac{}{\vdash ==, !=, <, <=, >, >= : (\text{int}, \text{int}) \rightarrow \text{bool}}$ TYP_CMPOPS

$\frac{}{\vdash \&, | : (\text{bool}, \text{bool}) \rightarrow \text{bool}}$ TYP_BOOLOPS

$\boxed{\vdash uop : F}$

$\frac{}{\vdash ! : (\text{bool}) \rightarrow \text{bool}}$ TYP_LOGNOT

$\frac{}{\vdash \sim : (\text{int}) \rightarrow \text{int}}$ TYP_BITNEG

$\frac{}{\vdash - : (\text{int}) \rightarrow \text{int}}$ TYP_NEG

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

$\frac{x:t \in L}{G; L \vdash x : t}$ TYP_LOCAL

$\frac{}{G; L \vdash n : \text{int}}$ TYP_INT

$\frac{}{G; L \vdash s : \text{string}}$ TYP_STRING

$\frac{}{G; L \vdash \text{stringnull} : \text{string}}$ TYP_NULLSTR

$\frac{}{G; L \vdash t[] \text{ null} : t[]}$ TYP_NULLARR

$\frac{}{G; L \vdash \text{true} : \text{bool}}$ TYP_TRUE

$\frac{}{G; L \vdash \text{false} : \text{bool}}$ TYP_FALSE

$\frac{G; L \vdash exp_1 : t[] \quad G; L \vdash exp_2 : \text{int}}{G; L \vdash exp_1[exp_2] : t}$ TYP_INDEX

$\frac{f:(t_1, \dots, t_i) \rightarrow t \in G \quad G; L \vdash exp_1 : t_1 \quad \dots \quad G; L \vdash exp_i : t_i}{G; L \vdash f(exp_1, \dots, exp_i) : t}$ TYP_CALL

$\frac{G; L \vdash exp_1 : t \quad \dots \quad G; L \vdash exp_i : t}{G; L \vdash \text{new } t[] \{exp_1, \dots, exp_i\} : t[]}$ TYP_ARRLIT

$\frac{G; L \vdash exp_1 : \text{int}}{G; L \vdash \text{new int } [exp_1] : t[]}$ TYP_ARRZEROINT

$\frac{G; L \vdash exp_1 : \text{int}}{G; L \vdash \text{new bool } [exp_1] : t[]}$ TYP_ARRZERBOOL

$\frac{\vdash bop : (t_1, t_2) \rightarrow t \quad G; L \vdash exp_1 : t_1 \quad G; L \vdash exp_2 : t_2}{G; L \vdash exp_1 \text{ bop } exp_2 : t}$ TYP_BOP

$\frac{\vdash uop : (t) \rightarrow t \quad G; L \vdash exp : t}{G; L \vdash uop \text{ exp} : t}$ TYP_UOP

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

$$\frac{G;L \vdash exp : t \quad x \notin L}{G;L \vdash \text{var } x = exp \Rightarrow L, x:t} \text{ TYP_DECL}$$

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

$$\frac{G;L_0 \vdash vdecl_1 \Rightarrow L_1 \quad \dots \quad G;L_{i-1} \vdash vdecl_i \Rightarrow L_i}{G;L_0 \vdash vdecl_1, \dots, vdecl_i \Rightarrow L_i} \text{ TYP_VDECLS}$$

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

$$\frac{G;L_1 \vdash vdecl \Rightarrow L_2}{G;L_1;t \vdash vdecl; \Rightarrow L_2} \text{ TYP_SDECL}$$

$$\frac{G;L \vdash lhs : t \quad G;L \vdash exp_2 : t}{G;L;t \vdash lhs = exp_2; \Rightarrow L} \text{ TYP_ASSN}$$

$$\frac{f:(t_1, \dots, t_i) \rightarrow \text{void} \in G \quad G;L \vdash exp_1 : t_1 \quad \dots \quad G;L \vdash exp_i : t_i}{G;L;t \vdash f(exp_1, \dots, exp_i); \Rightarrow L} \text{ TYP_SCALL}$$

$$\frac{G;L \vdash exp : \text{bool} \quad G;L;t \vdash block_1 \quad G;L;t \vdash block_2}{G;L;t \vdash \text{if}(exp) block_1 \text{ else } block_2 \Rightarrow L} \text{ TYP_IF}$$

$$\frac{G;L \vdash exp : \text{bool} \quad G;L;t \vdash block}{G;L;t \vdash \text{while}(exp) block \Rightarrow L} \text{ TYP_WHILE}$$

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

$$\frac{G;L \vdash exp : t}{G;L;t \vdash \text{return } exp; \Rightarrow L} \text{ TYP_RETT}$$

$$\frac{}{G;L;\text{void} \vdash \text{return}; \Rightarrow L} \text{ TYP_RETVoid}$$

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

$$\frac{G;L_0;t \vdash stmt_1 \dots stmt_i \Rightarrow L_i}{G;L_0;t \vdash \{stmt_1 \dots stmt_i\}} \text{ TYP_BLOCK}$$

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

$$\frac{G;L_0;t \vdash stmt_1 \Rightarrow L_1 \quad \dots \quad G;L_{i-1};t \vdash stmt_i \Rightarrow L_i}{G;L_0;t \vdash stmt_1 \dots stmt_i \Rightarrow L_i} \text{ TYP_STMTS}$$

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

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

$$\frac{f \notin G}{G \vdash t f(t_1 x_1, \dots, t_i x_i) \text{ block} \Rightarrow G, f : (t_1, \dots, t_i) \rightarrow t} \quad \text{TYP_FDECL}$$

$$\boxed{G_0 \vdash \text{decl}_1 \dots \text{decl}_i \Rightarrow G_i}$$

$$\frac{G_0 \vdash \text{decl}_1 \Rightarrow G_1 \quad \dots \quad G_{i-1} \vdash \text{decl}_i \Rightarrow G_i}{G_0 \vdash \text{decl}_1 \dots \text{decl}_i \Rightarrow G_i} \quad \text{TYP_GLOBAL_CTXT}$$

$$\boxed{G \vdash \text{decl}}$$

$$\frac{G; x_1 : t_1, \dots, x_i : t_i; t \vdash \text{block}}{G \vdash t f(t_1 x_1, \dots, t_i x_i) \text{ block}} \quad \text{TYP_GFUN}$$

$$\frac{\cdot; \cdot \vdash \text{gexp} : t}{G \vdash \text{global } x = \text{gexp};} \quad \text{TYP_GVAR}$$

$$\boxed{\vdash \text{prog}}$$

$$\frac{G_0 \vdash \text{decl}_1 \dots \text{decl}_i \Rightarrow G \quad G \vdash \text{decl}_1 \quad \dots \quad G \vdash \text{decl}_i}{\vdash \text{decl}_1 \dots \text{decl}_i} \quad \text{TYP_PROG}$$