Hoon used to have two syntaxes and one mode for runes which expected *model* hoons, *value* hoons, or both. Now Hoon uses one consistent syntax for constructing hoons and infers based on the rune whether to use the hoon as a model or value.

bar - core		**description**	**irregular form**
|%	*<arms>*	form a core with subject as the payload	
|~	[model value]	form an iron gate	
|=	[model value]	form a gate, a dry one-armed core with sample	
|.	hoon	form a trap, a one-armed core with one arm $	
|-	hoon	form a trap and kick (“call”) it	
|_	model (map term foot)	form a door, a many-armed core with sample	
|*	[model value]	form a gill, a wet one-armed core with sample	
|^	hoon (map term foot)	form a core with battery and anonymous arm $ and kick it	
|:	[hoon hoon]	form a core with burnt sample	
|?	hoon	form a lead trap	
|$	(lest term) body	form a mold builder wet gate	
|*	[spec hoon]	form a wet gate	
|@	[(unit term) (map term tome)]	form a wet core	

buc - mold			
$:	(list model)	form a mold to recognize a tuple	[a=foo b=bar c=baz]
$=	[@tas model]	mold that wraps a face around another mold	foo=bar
$%	(list [[aura @] model])	mold recognizing a union tagged by head atom	
$@	[model model]	mold that normalizes a union tagged by depth	
$^	[model model]	mold that normalizes a union tagged by head depth	
$?	(list model)	mold that normalizes a generic union	?($foo $bar $baz)
$-	[model model]	mold that normalizes to an example gate	
$_	value	mold that normalizes to an example	_foo

cen - call			
%=	[wing (list (pair wing hoon))]	take a wing with changes	foo(x 1, y 2, z 3)
%_	[wing (list (pair wing hoon))]	take a wing with changes, preserving type	
%~	[wing hoon hoon]	call with multi-armed door	~(arm core arg)
%-	[hoon hoon]	call a gate (function)	(fun arg)
%.	[hoon hoon]	call a gate, reversed	
%+	[hoon hoon hoon]	call a gate with pair sample	
%^	[hoon hoon hoon hoon]	call a gate with triple sample	
%*	[wing hoon (list (pair wing hoon))]	make with arbitrary hoon	
%:	[hoon (list hoon)]	call a gate with many arguments	

col - cell			
:-	[hoon hoon]	construct a cell (2-tuple)	[a b], a^b
:+	[hoon hoon hoon]	construct a triple (3-tuple)	[a b c]
:^	[hoon hoon hoon hoon]	construct a quadruple (4-tuple)	[a b c d]
:*	(list hoon)	construct an n-tuple	[a b c d e …]
:~	(list hoon)	construct a null-terminated list	~[a b c]
:_	[hoon hoon]	construct a cell, inverted	

dot - nock			
.*	[hoon hoon]	evaluate with nock 2	
.?	hoon	check for cell or atom with nock 3	
.+	atom	increment an atom with nock 4	+(42)
.=	[hoon hoon]	test for equality with nock 5	=(a b)
.^	[model value]	load from the arvo namespace with nock 11	

ket - cast			
^+	[value value]	typecast by example (value)	
^-	[model value]	typecast by mold	`foo`bar
^=	[toga value]	name a value	foo=bar
^?	hoon	convert any core to a lead core (bivariant)	
^|	hoon	convert a gold core to an iron core (contravariant)	
^~	hoon	fold constant at compile time	
^.	[p=hoon q=hoon]	typecast on value produced by passing q to p	
^&	hoon	convert a core to a zinc core (covariant)	
^*	p=spec	produce the bunt of p	
^:	p=spec	mold gate for type p	

mic - misc		**description**	**irregular form**
;=	marl	make a list of XML nodes (Sail)	
;:	[hoon (list hoon)]	call a binary function as an n-ary	
;/	hoon	tape as XML element	
;<	spec hoon hoon hoon	monadic bind	
;~	hoon (list hoon)	glue pipeline together with a product-sample adapter	
;;	spec hoon	normalize with a mold, asserting fixpoint.	

sem - make			
;;	[model value]	normalize with a mold, asserting fixpoint	
;~	[hoon (list hoon)	glue a pipeline together with a product-sample adapter	
;:	[hoon (list hoon)	call a binary function as an n-ary function	:(fun a b c d)
;/	hoon	tape as XML element	

sig - hint			
~&	[hoon hoon]	debugging printf	
~%	[term wing (list [term hoon]) hoon]	jet registration	
~/	[term hoon]	jet registration for gate with registered context	
~$	[term hoon]	profiling hit counter	
~|	[hoon hoon]	tracing printf	
~_	[hoon hoon]	user-formatted tracing printf	
~?	[hoon hoon hoon]	conditional debug printf	
~>	$@(term [term hoon]) hoon]	raw hint, applied to computation	
~<	$@(term [term hoon]) hoon]	raw hint, applied to product	
~+	hoon	cache a computation	
~=	[hoon hoon]	detect duplicate	
~!	[hoon hoon]	print type on compilation fail	

tis - flow			
=>	[hoon hoon]	compose two hoons	
=^	[taco wing hoon hoon]	pin the head of a pair; change a leg with the tail	
=*	[term hoon hoon]	define an alias	
=~	(list hoon)	compose many hoons	
=<	[hoon hoon]	compose two hoons, inverted	foo:bar
=+	[hoon hoon]	combine a new noun with the subject	
=-	[hoon hoon]	combine a new noun with the subject, inverted	
=|	[model value]	combine a defaulted mold with the subject	
=/	[taco value hoon]	combine a named and/or typed noun with the subject	
=;	[taco value hoon]	combine a named and/or typed noun with the subject, inverted	
=.	[wing hoon hoon]	change one leg in the subject	
=:	[(list (pair wing hoon)) hoon]	change multiple legs in the subject	
=?	[wing hoon hoon hoon]	conditionally change one leg in the subject	
=,	[hoon hoon]	expose namespace	

wut - test			
?:	[hoon hoon hoon]	branch on a boolean test	
?<	[hoon hoon]	negative assertion	
?>	[hoon hoon]	positive assertion	
?-	[wing (list (pair model value))]	switch against a union, with no default	
?+	[wing value (list (pair model value))]	switch against a union, with a default	
?.	[hoon hoon hoon]	branch on a boolean test, inverted	
?~	[wing hoon hoon]	branch on whether a wing of the subject is null	
?@	[wing hoon hoon]	branch on whether a wing of the subject is an atom	
?^	[wing hoon hoon]	branch on whether a wing of the subject is a cell	
?=	[model wing]	test model match	
?!	hoon	logical not	!foo
?&	(list hoon)	logical and	&(foo bar baz)
?|	(list hoon)	logical or	|(foo bar baz)

zap - wild		
!!	$~	crash
!>	hoon	wrap a noun in its type (create a vase)
!=	hoon	make the nock formula for a hoon
!?	[@ hoon]	restrict the hoon version
!,	[hoon hoon]	ast quote
!<	[spec hoon]	check that the type in a vase matches a mold

zap - wild	description	irregular form
!; [hoon hoon]	type quote	
!@ [(list wing) hoon hoon]	conditional compilation	

other syntax

+1:[a b]	[a b]
+2:[a b]	a
+3:[a b]	b
+6:[a [b c]]	b
+7:[a [b c]]	c
.:[a b]	[a b]
-:[a b]	a
+:[a b]	b
+<:[a [b c]]	b
+>:[a [b c]]	c
-.core	battery
+.core	payload
+>.core	context (outer core)
+<.core	sample
^face	face in outer core
.	current subject
+	+:.
-	-:.
+>	+>:.
..arm	core in which ++arm is defined

~	0 (nil)
%.y &	yes (true)
%.n |	no (false)
`a	[~ a]
~[a b c]	[a b c ~]
[a b c]~	[[a b c] ~]
~2017.8.26	`@da`date
~marzod-taglux	`@p`pronounceable base-256 number
12.345.567	`@ud`decimal
--12.345.567	`@sd`signed decimal
0xdeadbeef	`@ux`hexadecimal
.1.23e4	`@rs`floating-point decimal
“hoon”	tape (text as list of characters)
‘hoon’	cord (text as atom)
%hoon	term (text as ASCII symbol, kabob-case)
?=($hoon %hoon)	%.y
?=($hoon %loon)	%.n
foo/bar	[%foo bar]
/foo/bar	[%foo %bar ~] wire (path)
,%hoon	manually switch into model mode (term)
,[a=foo b=bar]	manually switch into model mode (cell)