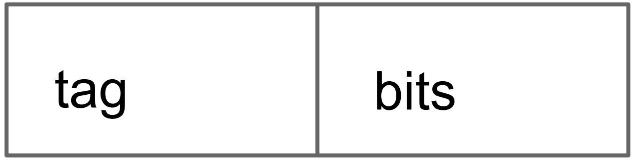
```
############################################################################################
##
###                                                                                      ###
###                                                _                                     ###
###                                   _       _ _(_)_                                    ###
###                                  (_)     | (_) (_)                                   ###
###                                   _ _   _| |_  __ _                                  ###
###                                  | | | | | | |/ _` |                                 ###
###                                  | | |_| | | | (_| |                                 ###
###                                 _/ |\__'_|_|_|\__'_|                                 ###
###                                |__/                                                  ###
###                                                                                      ###
###                              _     _                     _                           ###
###                             (_)_ _| |_ ___ _ _ _ _  __ _| |___                       ###
###                             | | ' \  _/ -_) '_| ' \/ _` | (_-<                       ###
###                             |_|_||_\__\___|_| |_||_\__,_|_/__/                       ###
###                                                                                      ###
###                                                                                      ###
###                                Jeff Bezanson, 6/27/2014                              ###
###                                                                                      ###
############################################################################################
##
```

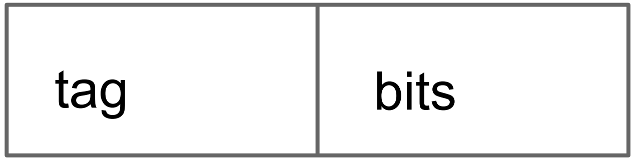
What exists?

- Data model
  - bits, struct, tuple
  - *structured* type tags
  - abstract types



- Code model
  - object-level intrinsics (27)
  - bit-level intrinsics (82)
  - lambda, var, assign, goto-if, call, return, try, new
- Glue
  - Method tables match data with code

Cute oversimplification



C only has the right part

Symbolic systems only have the left part

Popular HLLs have both parts…

...but what can go in them is limited.

We go nuts with it.

types (77) - boot.jl, jltypes.c

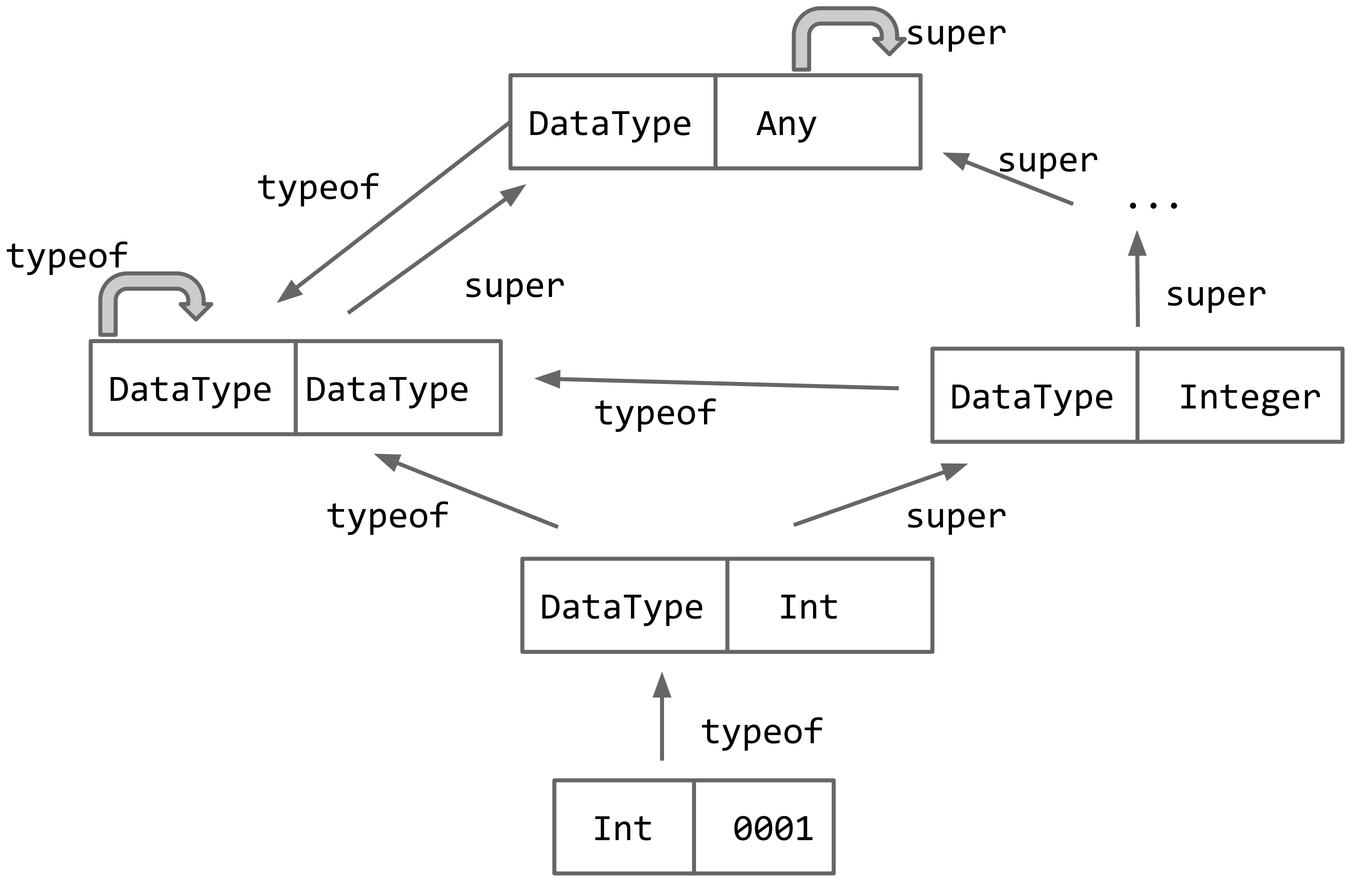
```
Any, DataType, Vararg, NTuple, None, Tuple, Type, TypeConstructor,
TypeName, TypeVar, Union, UnionType, AbstractArray, DenseArray, Box,
Function, IntrinsicFunction, LambdaStaticData, Method, MethodTable,
Module, Nothing, Symbol, Task, Array, Bool, FloatingPoint, Float16,
Float32, Float64, Number, Integer, Int, Int8, Int16, Int32, Int64,
Int128, Ptr, Real, Signed, Uint, Uint8, Uint16, Uint32, Uint64,
Uint128, Unsigned, Char, ASCIIString, ByteString, DirectIndexString,
String, UTF8String, BoundsError, DivideError, DomainError,
Exception, InexactError, InterruptException, MemoryError,
OverflowError, StackOverflowError, UndefRefError, UndefVarError,
Expr, GotoNode, LabelNode, LineNumberNode, QuoteNode, SymbolNode,
TopNode, GetfieldNode, NewvarNode
```

...plus their constructors

types (77)

- Closed under `typeof`, `super` via circular references
- Represents all code and data

Object graph



object primitives (27) - builtins.c

types & predicates

```
is  issubtype  isa
typeassert  convert_default
```

constructors

```
tuple  apply_type  Union
```

calls and methods

```
apply    invoke
yieldto  throw
kwcall   applicable
method_exists
```

property access

```
fieldtype   getfield
setfield!   isdefined
arraylen    arrayref
arrayset    arraysize
tuplelen    tupleref
typeof
```

eval

```
eval
```

intrinsics (82) - intrinsics.cpp

ccall, cglobal, abs_float, add_float, add_int, and_int, ashr_int, box, bswap_int, checked_fptosi, checked_fptoui, checked_sadd, checked_smul, checked_ssub, checked_uadd, checked_umul, checked_usub, nan_dom_err, copysign_float, ctlz_int, ctpop_int, cttz_int, div_float, eq_float, eq_int, eqfsi64, eqfui64, flipsign_int, select_value, sqrt_llvm, powi_llvm, fpext64, fpiseq, fpislt, fpsiround, fpuiround, fptosi, fptoui, fptrunc32, le_float, lefsi64, lefui64, lesif64, leuif64, lshr_int, lt_float, ltfsi64, ltfui64, ltsif64, ltuif64, mul_float, mul_int, ne_float, ne_int, neg_float, neg_int, not_int, or_int, rem_float, sdiv_int, shl_int, sitofp, sle_int, slt_int, smod_int, srem_int, sub_float, sub_int, trunc_int, udiv_int, uitofp, ule_int, ult_int, unbox, urem_int, xor_int, sext_int, zext_int, jl_alloca, pointerref, pointerset, pointertoref

Translation stages

1. parse (text → Expr) - `parse`
2. expand macros - `macroexpand`
3. syntax de-sugaring
4. statementize control flow
5. resolve scopes
6. generate IR (“goto” form) - `expand`, `code_lowered`
7. top-level eval, method sorting - `methods`
8. type inference
9. inlining, high-level optimizations - `code_typed`
10. LLVM IR generation - `code_llvm`
11. LLVM optimizer, native code gen - `code_native`

Test subject

```
# edited from version.jl
function check_new_version(existing::Vector{VersionNumber},
                           ver::VersionNumber)
    @assert issorted(existing)
    for v in [v"0", v"0.0.1", v"0.1", v"1"]
        lowerbound(v) <= ver <= v && return
    end
    error("$ver is not a valid initial version (try 0.0.0, 0.0.1,
0.1 or 1.0)")
end
```

http://goo.
gl/JNE2Y0

1. parse (julia-parser.scm)

```
(function
 (call check_new_version (:: existing (curly Vector VersionNumber))
                         (:: ver VersionNumber))
 (block (line 2 none)
  (macrocall @assert (call issorted existing))
  (line 3)
  (for (= v (vcat (macrocall @v_str "0")   (macrocall @v_str "0.0.1")
                  (macrocall @v_str "0.1") (macrocall @v_str "1")))
       (block (line 4)
              (&& (comparison (call lowerbound v) <= ver <= v)
                  (return (null)))))
  (line 6)
  (call error (string ver " is not a valid initial version (try 0.0.0,
0.0.1, 0.1 or 1.0)"))))
```

2. expand macros (julia-syntax.scm, ast.c)

```
...
 (block (line 2 none)
  (if (call issorted existing) (null)
      (call (|.| #<julia: Base> 'error)
            #<julia: "assertion failed: issorted(existing)">))
  (line 3)
  (for (= v (vcat #<julia: VersionNumber> #<julia: VersionNumber>
                  #<julia: VersionNumber> #<julia: VersionNumber>))
       (block (line 4)
              (&& (comparison (call lowerbound v) <= ver <= v)
                  (return (null)))))
  (line 6)
  (call error (string ver " is not a valid initial version (try 0.0.0,
0.0.1, 0.1 or 1.0)"))))
```

3. syntax de-sugaring (julia-syntax.scm)

```
(method check_new_version
 (call (top tuple)
      (call (top tuple) (call (top apply_type) Vector VersionNumber)
                        VersionNumber) (call (top tuple)))
 (lambda (existing ver)
  (scope-block
   (block (line 2 none)
          (if (call issorted existing) (null)
              (call (call (top getfield) #<julia: Base> 'error)
                    #<julia: "assertion failed: issorted(existing)">))
   ;; LOOP - next slide
   (line 6)
   (call error (call (top string) ver " is not a valid initial version
(try 0.0.0, 0.0.1, 0.1 or 1.0)"))))))
```

3. syntax de-sugaring (cont.)

```
(scope-block
 (block (= |#s118| (call vcat #<VersionNumber> #<VersionNumber>
                              #<VersionNumber> #<VersionNumber>))
        (= |#s117| (call (top start) |#s118|))
        (scope-block
         (break-block loop-exit
          (_while (call (top !) (call (top done) |#s118| |#s117|))
           (break-block loop-cont
            (scope-block
             (block (block (= |#s116| (call (top next) |#s118||#s117|))
                           (= v (call (top tupleref) |#s116| 1))
                           (= |#s117| (call (top tupleref) |#s116| 2))
                           (null) |#s116|)
   (block (line 4) (&& (&& (call <= (call lowerbound v) ver)
                           (call <= ver v))
                       (return (null))))))))))))
```

4. statementize control flow

```
(&& (&& (call <= (call lowerbound v) ver)
        (call <= ver v))
    (return (null)))
```

$$\downarrow$$

```
(if (call <= (call lowerbound v) ver)
    (if (call <= ver v)
        (return (null))
        false)
    false))
```

5-6. goto form (julia-syntax.scm)

```
(lambda (existing ver) ((|#s118| |#s117| |#s116| v) ((existing Any 0) (ver Any 0) (|#s118|
Any 18) (|#s117| Any 2) (|#s116| Any 18) (v Any 18)) ())
 (body (line 2 none)
  (gotoifnot (call issorted existing) 0)  (goto 1)
  (label 0)
  (call (call (top getfield) #<julia: Base> 'error) #<"assertion failed: …">)
  (label 1) (line 3)
  (= |#s118| (call vcat #<VersionNum> #<VersionNum> #<VersionNum> #<VersionNum>))
  (= |#s117| (call (top start) |#s118|))
  (gotoifnot (call (top !) (call (top done) |#s118| |#s117|)) 3)  (label 4)
  (= |#s116| (call (top next) |#s118| |#s117|))
  (= v (call (top tupleref) |#s116| 1))
  (= |#s117| (call (top tupleref) |#s116| 2)) (line 4)
  (gotoifnot (call <= (call lowerbound v) ver) 7)
  (gotoifnot (call <= ver v) 6)
  (return (null))
  (label 6) (goto 7) (label 7) (label 5)
  (gotoifnot (call (top !) (call (top !) (call (top done) |#s118| |#s117|))) 4)
  (label 3) (label 2) (line 6)
  (return (call error (call (top string) ver " is not a valid initial version (try 0.0.0,
0.0.1, 0.1 or 1.0)")))))))
```

5-6. goto form - code_lowered

```
    unless issorted(existing) goto 0
    goto 1
    0:
    top(getfield)(Base,:error)("assertion failed: issorted(existing)")
    1:  # line 3:
    #s120 = vcat(v"0.0.0",v"0.0.1",v"0.1.0",v"1.0.0")
    #s119 = top(start)(#s120)
    unless top(!)(top(done)(#s120,#s119)) goto 3
    4: #s118 = top(next)(#s120,#s119)
    v = top(tupleref)(#s118,1)
    #s119 = top(tupleref)(#s118,2) # line 4:
    unless lowerbound(v) <= ver goto 7
    unless ver <= v goto 6
    return
    6: goto 7
    7: 5:
    unless top(!)(top(!)(top(done)(#s120,#s119))) goto 4
    3: 2:  # line 6:
    return error(top(string)(ver," is not a valid initial version (try 0.0.0, 0.0.1, 0.1 or 
1.0)"))
```

7. eval, method sorting

toplevel.c

interpreter.c

gf.c

Method Cache (gf.c)

- Two problems:
    - Slow method lookup
    - Too many specializations
- Solution
    - Supports a subset of the type system
    - “widens” signatures before adding to cache

```
function max_fixed_point(P::Vector, a₁::AbstractValue, eval)
    bot = (Symbol=>LatticeElement)[ v => ⊥ for v in keys(a₁) ]
    n = length(P);  s = [ a₁; [ bot for i = 2:n ] ];  W = IntSet(1)
    while !isempty(W)
        pc = first(W)
        while pc != n+1
            delete!(W, pc)
            I = P[pc]; new = s[pc]
            if isa(I, Assign)
                new = copy(new); new[I.lhs.name] = eval(I.rhs, new)
            if isa(I, Goto)
                pc´ = I.label
            else
                pc´ = pc+1
                if isa(I, GotoIf)
                    l = I.label
                    if !(new <= s[l])
                        push!(W, l)
                        s[l] = s[l] ⊔ new
            if pc´<=n && !(new <= s[pc´])
                s[pc´] = s[pc´] ⊔ new
                pc = pc´
            else
                pc = n+1
```

Compile-time method lookup

```
Base._methods(copy!, (Vector{Int},Vector), 4)
2-element Array{Any,1}:

 ((Array{Int64,1},Array{Int64,1}),
  (T,Int64),copy!{T}(dest::Array{T,N},src::Array{T,N}) at array.jl:57)

 ((Array{Int64,1},Array{T,1}),
  (),copy!(dest::AbstractArray{T,N},src) at abstractarray.jl:147)
```

8. type inference (inference.jl)

```
unless issorted(existing::Array{VersionNumber,1})::Bool goto 0
goto 1
0:  top(getfield)(Base,:error)("assertion failed: issorted(existing)")::None
1:  # line 3:
#s120 = vcat(v"0.0.0",v"0.0.1",v"0.1.0",v"1.0.0")::Array{VersionNumber,1}
#s119 = top(start)(#s120::Array{VersionNumber,1})::Int64
unless top(!)(top(done)(#s120::Array{VersionNumber,1},#s119::Int64)::Bool)::Bool goto 3
4: #s118 = top(next)(#s120::Array{VersionNumber,1},#s119::Int64)::(VersionNumber,Int64)
v = top(tupleref)(#s118::(VersionNumber,Int64),1)::VersionNumber
#s119 = top(tupleref)(#s118::(VersionNumber,Int64),2)::Int64 # line 4:
unless lowerbound(v::VersionNumber) <= ver::VersionNumber::Bool goto 7
unless ver::VersionNumber <= v::VersionNumber::Bool goto 6
return
6: goto 7
7: 5:
unless top(!)(top(!)(top(done)(#s120::Array{VersionNumber,1},#s119::Int64)::Bool)::Bool)::
Bool goto 4
3: 2:  # line 6:
return error(top(string)(ver::VersionNumber," is not a valid initial version (try 0.0.0,
0.0.1, 0.1 or 1.0)")::Union(UTF8String,ASCIIString))::None
    end::Nothing))))
```

9. inlining, high-level opt

```
unless issorted(existing::Array{VersionNumber,1},GetfieldNode(Base.Sort,:ord,Any) (isless::
F,identity::F,false,GetfieldNode(Base.Sort,:Forward,ForwardOrdering)))::Bool goto 0
goto 1
0: throw($(Expr(:new, ErrorException, "assertion failed: ..."))::ErrorException)::None
1:  # line 3:
#s120 = vcat(v"0.0.0",v"0.0.1",v"0.1.0",v"1.0.0")::Array{VersionNumber,1}
#s119 = 1
_var1 = arraylen(#s120::Array{VersionNumber,1})::Int64
unless box(Bool,top(not_int)(top(slt_int)(_var1::Int64,#s119::Int64)::Bool))::Bool goto 3
4: _var6 = arrayref(#s120::Array{VersionNumber,1},#s119::Int64)::VersionNumber
_var7 = box(Int64,top(add_int)(#s119::Int64,1))::Int64
v = _var6::VersionNumber
#s119 = _var7::Int64 # line 4:
_var3 = lowerbound(v::VersionNumber)
_var2 = isless(ver::VersionNumber,_var3)::Bool
unless box(Bool,top(not_int)(_var2::Bool))::Bool goto 7
_var4 = isless(v::VersionNumber,ver::VersionNumber)::Bool
unless box(Bool,top(not_int)(_var4::Bool))::Bool goto 6
return
6: goto 7
7: 5:
```

9. inlining, high-level opt (cont.)

```
_var5 = arraylen(#s120::Array{VersionNumber,1})::Int64
unless box(Bool,   not_int(box(Bool,not_int(slt_int(_var5::Int64,#s119::Int64)::Bool))::
Bool))::Bool goto 4
3: 2:  # line 6:
return error(print_to_string(ver::VersionNumber," ...")::Union(UTF8String,ASCIIString))::
None
    end::Nothing))))
```

10. LLVM IR generation

codegen.cpp, cgutils.cpp, ccall.cpp, intrinsics.
cpp

Code gen key concepts

- boxed vs. unboxed
- `jl_varinfo_t`
- `jl_codectx_t`
- object primitives: `emit_known_call`
- `emit_expr`, `emit_function` (18 steps)
- gc frame and gc roots
- julia ↔ llvm type translation