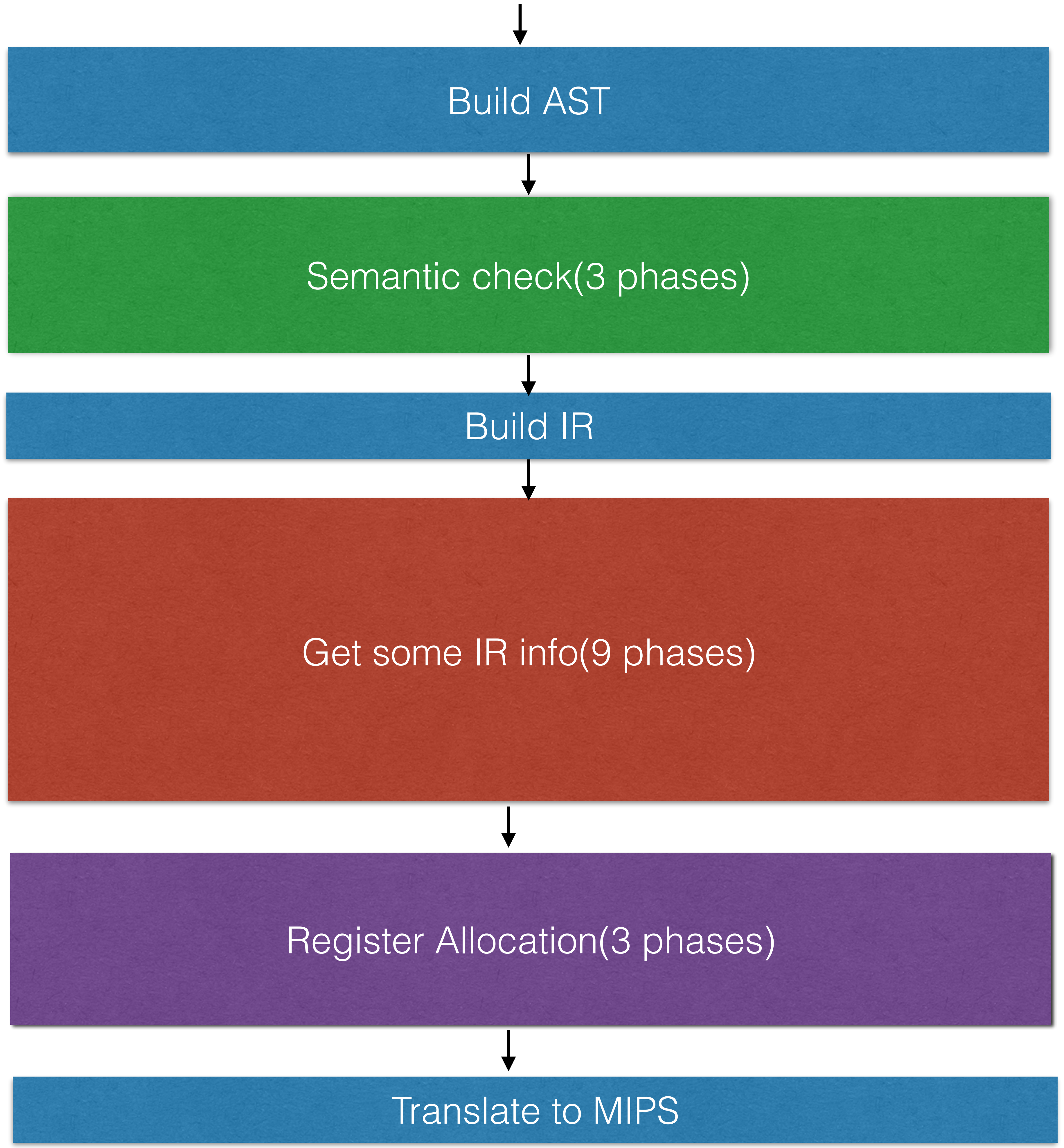
MeowCompiler

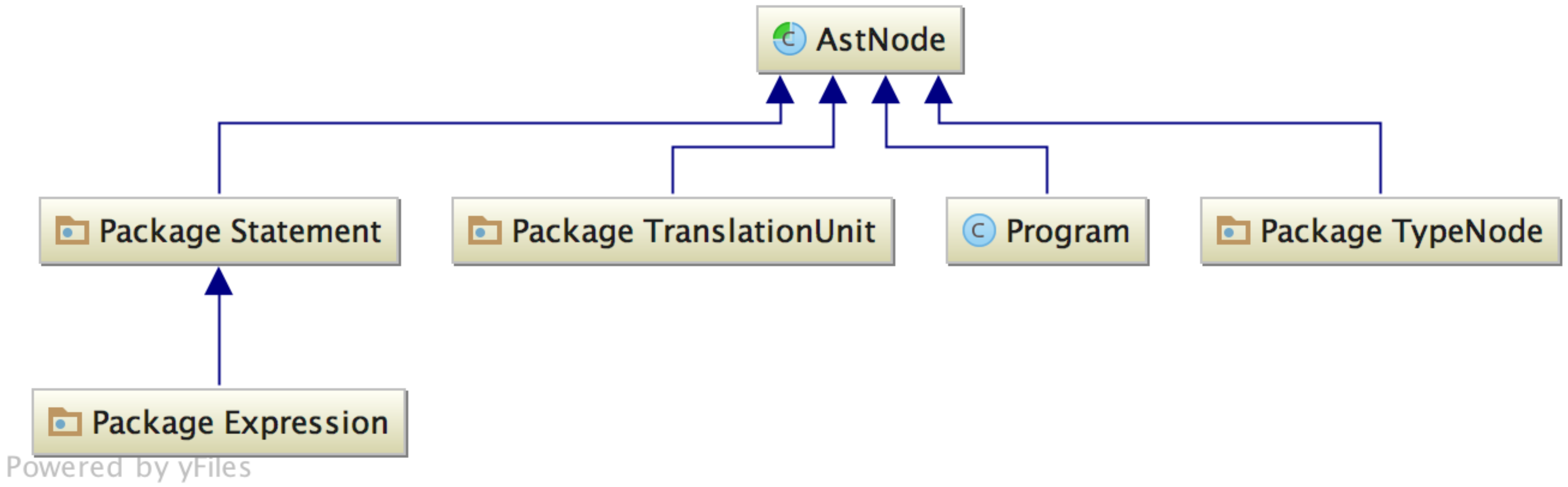
A simple and naive compiler

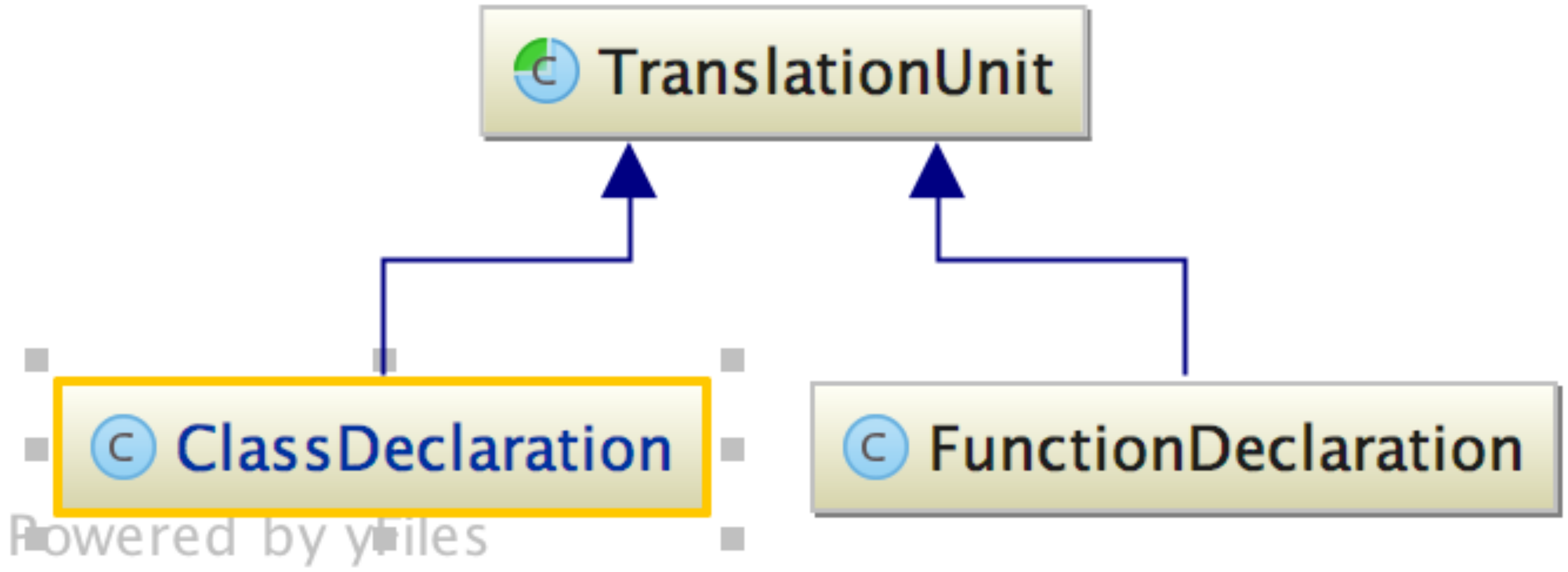
Overview

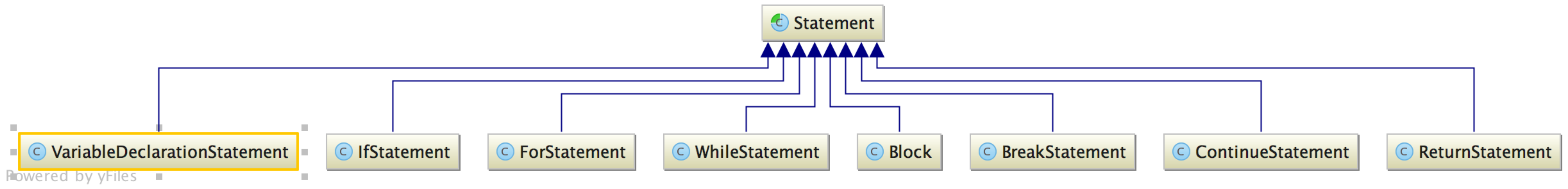

Build AST
Semantic check(3 phases)
Build IR
Get some IR info(9 phases)
Register Allocation(3 phases)
Translate to MIPS

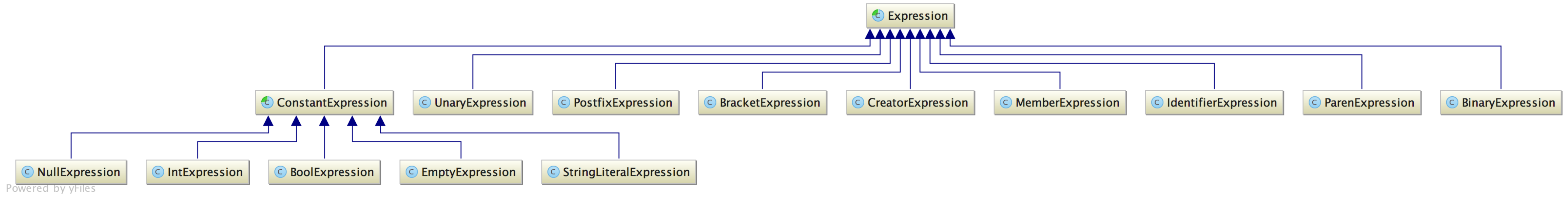
ANTLR 4

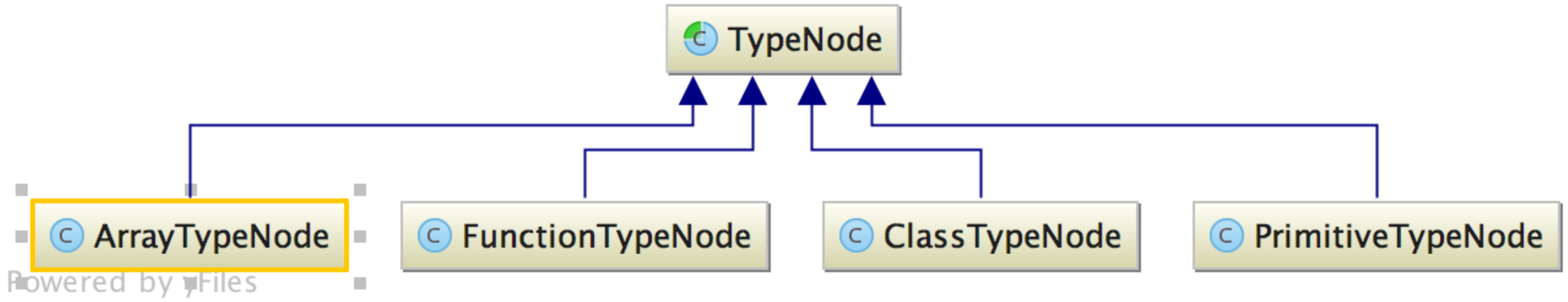
AST


AstNode
Package Statement
Package TranslationUnit
Program
Package TypeNode
Package Expression
Powered by yFiles


TranslationUnit
ClassDeclaration
FunctionDeclaration
Powered by yFiles


Statement
VariableDeclarationStatement
IfStatement
ForStatement
WhileStatement
Block
BreakStatement
ContinueStatement
ReturnStatement
Powered by yFiles

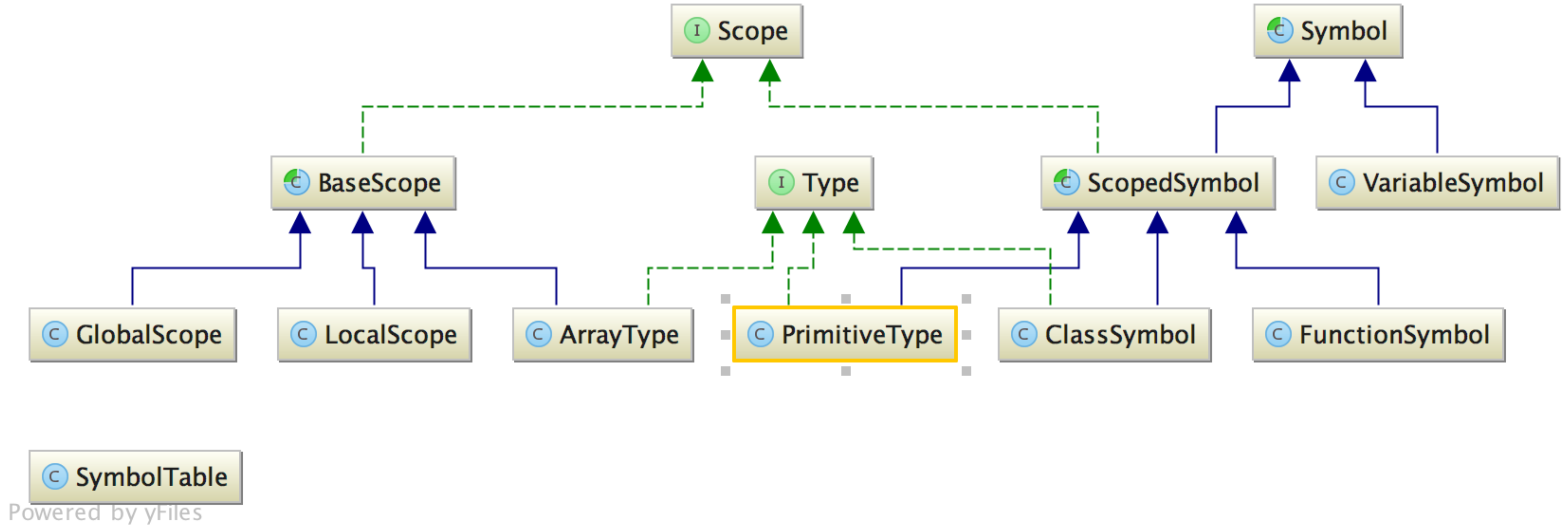

Expression
ConstantExpression
UnaryExpression
PostfixExpression
BracketExpression
CreatorExpression
MemberExpression
IdentifierExpression
ParenExpression
BinaryExpression
NullExpression
IntExpression
BoolExpression
EmptyExpression
StringLiteralExpression
Powered by yFiles


TypeNode
ArrayTypeNode
FunctionTypeNode
ClassTypeNode
PrimitiveTypeNode
Powered by yFiles

Semantic Check

- Phase 1: get all class type name
- Phase 2: resolve all class body and function args
- Phase 3: resolve all all function bodies

Symbol Table


Scope
Symbol
BaseScope
Type
ScopedSymbol
VariableSymbol
GlobalScope
LocalScope
ArrayType
PrimitiveType
ClassSymbol
FunctionSymbol
SymbolTable
Powered by yFiles

Built in function

IR

```
Jump Instruction:
    ret $src
    jump %target
    br $cond %ifTrue %ifFalse

Memory Access Instruction:
    store size $addr $src offset    // M[$addr+offset : $addr+offset+size-1] <- $src
    $dest = load size $addr offset  // $dest <- M[$addr+offset : $addr+offset+size-1]
    $dest = alloc $size

Function Call Instruction:
    call funcname $op1 $op2 $op3 ...
    $dest = call funcname $op1 $op2 $op3 ...

Register Transfer Instruction:
    $dest = move $src

Phi Instruction:
    $dest = phi %block1 $reg1 %block2 $reg2 ...

Arithmetic Instruction:
    $dest = neg $src
    $dest = add $src1 $src2
    $dest = sub $src1 $src2
    $dest = mul $src1 $src2
    $dest = div $src1 $src2
    $dest = rem $src1 $src2

Bitwise Instruction:
    $dest = shl $src1 $src2
    $dest = shr $src1 $src2
    $dest = and $src1 $src2
    $dest = xor $src1 $src2
    $dest = or $src1 $src2
    $dest = not $src

Condition Set Instruction:
    $dest = slt $src1 $src2
    $dest = sgt $src1 $src2
    $dest = sle $src1 $src2
    $dest = sge $src1 $src2
    $dest = seq $src1 $src2
    $dest = sne $src1 $src2
```

IR Info

- Calling Graph
- Static Data
- Liveness
- interference graph

Register Alloc

- Graph Colouring
- \$t0,\$t1 is used as temp reg
- \$a0-\$a3 is pre-alloc to the args
- \$v0 is used as return value
- \$t2-\$t9,\$s0-\$s7,\$v0-\$v1,\$a0-\$a3,\$fp,are available
- move a,b is suggested to be the same physical-reg

Optimisations

remove useless register

	horse	horse2	horse3	tak	heapsort
limit	25000000	15000000	25000000	3500000	20000000
original	0.57	0.99	0.68	0.71	FAILD
optimise	0.34	0.34	0.43	0.41	0.59
delta	0.23	0.65	0.25	0.3	0.59

print

	burgaria	hanoi	hashmap	magic	spill2
limit	1500000	450000	550000	2000000	100000
original	1.02	0.86	1.2	0.93	FAILD
optimise	0.64	0.14	0.53	0.72	0.16
delta	0.38	0.72	0.67	0.21	0.16

jump

- 10%

array

- 10%

static

	burgarian	expr	maxflow	prime
limit	1500000	40000	15000000	6000000
original	1.02	0.95	1.1	0.85
optimised	0.50	0.30	0.51	0.51
delta	0.52	0.65	0.59	0.34

recursive expand

	hashmap	horse2	horse3	prime
limit	550000	15000000	25000000	6000000
original	1.2	0.99	0.69	0.85
optimised	0.35	0.27	0.18	0.38
delta	0.85	0.72	0.51	0.47

THX