# Mechanized semantics for Clight

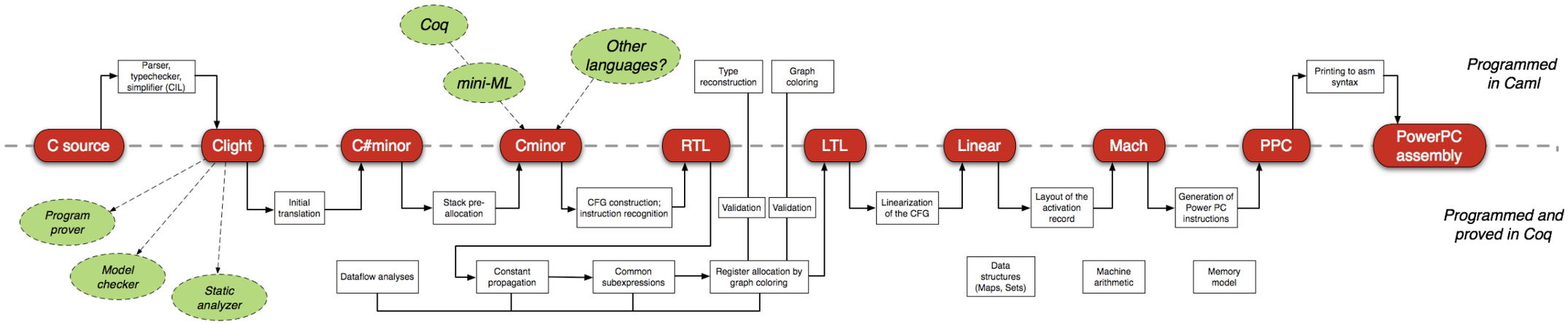
Sandrine Blaxy, Xavier Leroy

Pim Jager - Type Theory and Coq

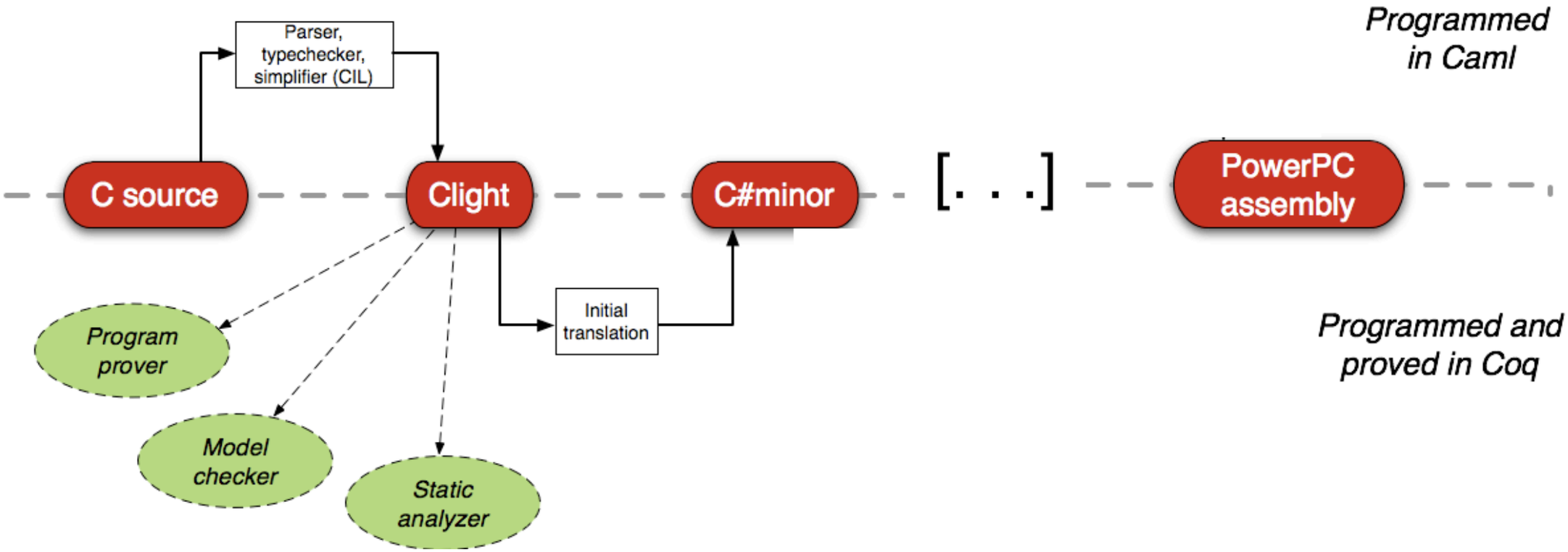
Radboud University

# The CompCert C project – Formally proving a compiler

- ” The CompCert project investigates the formal verification of realistic compilers usable for critical embedded software”
- Goal: Formally verify the transformation of C programs to machine executable assembly.



# Today we focus on Clight, a large subset of C



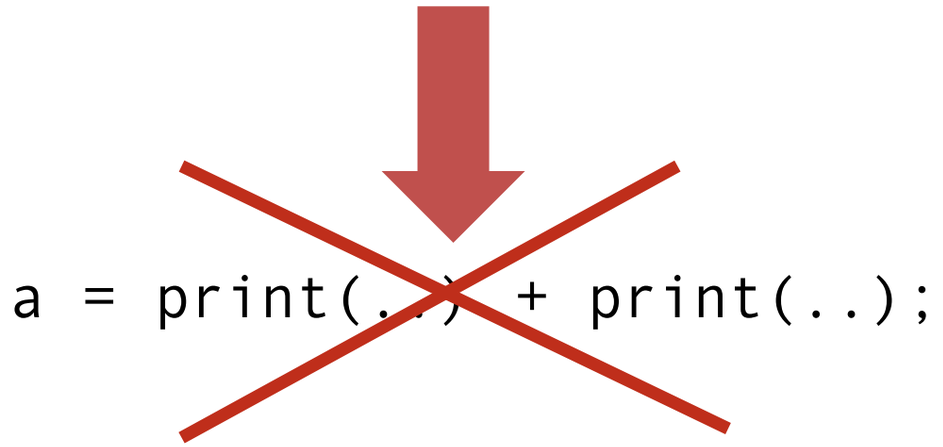
# Clight is a (very large) subset of C

**Including:**

- Most of C types
- Most C operators
- Pointers (and pointer arithmetic)
- Function pointers
- Structs
- Unions
- Almost all control structures

**Differences:**

- No GOTO statement
- Expressions must be pure
  - Ensures termination
  - Ensures deterministic evaluation



# Clight Syntax: Types

Signedness: $signedness ::= \texttt{Signed} \mid \texttt{Unsigned}$

Integer sizes: $intsize ::= \texttt{I8} \mid \texttt{I16} \mid \texttt{I32}$

Float sizes: $floatsize ::= \texttt{F32} \mid \texttt{F64}$

Types:
$$\begin{aligned}\tau ::= {} & \texttt{int}(intsize, signedness) \\ \mid {} & \texttt{float}(floatsize) \\ \mid {} & \texttt{void} \\ \mid {} & \texttt{array}(\tau, n) \\ \mid {} & \texttt{pointer}(\tau) \\ \mid {} & \texttt{function}(\tau^*, \tau) \\ \mid {} & \texttt{struct}(id, \varphi) \\ \mid {} & \texttt{union}(id, \varphi) \\ \mid {} & \texttt{comp\_pointer}(id)\end{aligned}$$

Field lists: $\varphi ::= (id, \tau)^*$

```
Inductive type : Type :=
   | Tvoid: type
   | Tint: intsize -> signedness -> attr -> type
   | Tlong: signedness -> attr -> type
   | Tfloat: floatsize -> attr -> type
   | Tpointer: type -> attr -> type
   | Tarray: type -> Z -> attr -> type
   | Tfunction: typelist -> type -> calling_convention -> type
   | Tstruct: ident -> attr -> type
   | Tunion: ident -> attr -> type
with typelist : Type :=
   | Tnil: typelist
   | Tcons: type -> typelist -> typelist.
```

# Clight Syntax : Expressions

Expressions: $a ::= id$
$\mid n$
$\mid f$
$\mid \texttt{sizeof}(\tau)$
$\mid op_1\ a$
$\mid a_1\ op_2\ a_2$
$\mid *a$
$\mid a.id$
$\mid \&a$
$\mid (\tau)a$
$\mid a_1\ ?\ a_2\ :\ a_3$

Unary operators: $op_1 ::= - \mid \sim \mid !$

Binary operators: $op_2 ::= + \mid - \mid * \mid / \mid \%$
$\mid << \mid >> \mid \& \mid \mid \mid \hat{}$
$\mid < \mid <= \mid > \mid >= \mid == \mid !=$

```
Inductive expr : Type :=
  | Econst_int: int -> type -> expr
  | Econst_float: float -> type -> expr
  | Econst_single: float32 -> type -> expr
  | Econst_long: int64 -> type -> expr
  | Evar: ident -> type -> expr
  | Etempvar: ident -> type -> expr
  | Ederef: expr -> type -> expr
  | Eaddrof: expr -> type -> expr
  | Eunop: unary_operation -> expr -> type -> expr
  | Ebinop: binary_operation -> expr -> expr -> type -> expr
  | Ecast: expr -> type -> expr
  | Efield: expr -> ident -> type -> expr
  | Esizeof: type -> type -> expr
  | Ealignof: type -> type -> expr.
```

# Clight syntax: Statements

Statements: $s ::=$ **skip**
$\mid a_1 = a_2$
$\mid a_1 = a_2(a^*)$
$\mid a(a^*)$
$\mid s_1; s_2$
$\mid$ **if**$(a)$ $s_1$ **else** $s_2$
$\mid$ **switch**$(a)$ $sw$
$\mid$ **while**$(a)$ $s$
$\mid$ **do** $s$ **while**$(a)$
$\mid$ **for**$(s_1, a_2, s_3)$ $s$
$\mid$ **break**
$\mid$ **continue**
$\mid$ **return** $a^?$

Switch cases: $sw ::=$ **default** : $s$
$\mid$ **case** $n$ : $s; sw$

```
Inductive statement : Type :=
  | Sskip : statement                                                                    (*
  | Sassign : expr -> expr -> statement                                                  (*
  | Sset : ident -> expr -> statement                                                    (*
  | Scall: option ident -> expr -> list expr -> statement                                (*
  | Sbuiltin: option ident -> external_function -> typelist -> list expr -> statemen
  | Ssequence : statement -> statement -> statement                                      (*
  | Sifthenelse : expr -> statement -> statement -> statement                            (*
  | Sloop: statement -> statement -> statement                                           (*
  | Sbreak : statement                                                                   (*
  | Scontinue : statement                                                                (*
  | Sreturn : option expr -> statement                                                   (*
  | Sswitch : expr -> labeled_statements -> statement                                    (*
  | Slabel : label -> statement -> statement
  | Sgoto : label -> statement

with labeled_statements : Type :=                                                       (*
  | LSnil: labeled_statements
  | LScons: option Z -> statement -> labeled_statements -> labeled_statements.
```

# C: calculating a least common multiple

```
int n1, n2;

int main() {
      int lcm;
      n1 = 42;
      n2 = 34;
      lcm = (n1>n2) ? n1 : n2;
      while(1) {
            if(lcm%n1==0 && lcm%n2==0){
                  break;
            }
            lcm++;
      }
      return lcm;
}
```

# Clight: calculating a least common multiple

```
int(I32, Signed) n1;
int(I32, Signed) n2;
int f() {
     int(I32, Signed) lcm;
     n1 = 42;
     n2 = 34;
     lcm = (n1>n2) ? n1 : n2;
     while(1) {
          if(lcm%n1==0 ? (lcm%n2==0 ? 1 : 0) : 0){
               break;
          } else { skip; }
          lcm = lcm + 1;
     } return lcm;
}
main = f
```

# Semantics of Clight

- Big step semantics: $(c, s) \Longrightarrow s'$

$G, E \vdash a, M \Leftarrow \ell$ → **Expression in left position**

$G, E \vdash a, M \Rightarrow v$ → **Expression in right position**

$G, E \vdash a^*, M \Rightarrow v^*$

$G, E \vdash s, M \overset{t}{\Rightarrow} out, M'$ → **Execution of terminating statement**

$G, E \vdash sw, M \overset{t}{\Rightarrow} out, M'$

$G, E \vdash s, M \overset{T}{\Rightarrow} \infty$ → **Execution of diverging statement**

$G, E \vdash sw, M \overset{T}{\Rightarrow} \infty$

$\vdash P \Rightarrow B$

# Judgment of terminating statements

$$G, E \vdash s, M \overset{t}{\Rightarrow} out, M'$$

- **G:** *Global environment*
- **E:** *Local environment*
- **s:** *statement to be executed*
- **M:** *Current state of memory*
- **t:** *trace of IO events*
- **out:** *statement outcomes:*
  - `Normal`
  - `Continue`
  - `Break`
  - `Return`
  - `Return(v)`
- **M':** *new state of memory*

# Semantics of statements (except loops)

$$G,E \vdash \texttt{skip},M \overset{\varepsilon}{\Rightarrow} \texttt{Normal},M \quad (15) \qquad G,E \vdash \texttt{break},M \overset{\varepsilon}{\Rightarrow} \texttt{Break},M \quad (16)$$

$$G,E \vdash \texttt{continue},M \overset{\varepsilon}{\Rightarrow} \texttt{Continue},M \quad (17) \qquad G,E \vdash (\texttt{return}\ \emptyset),M \overset{\varepsilon}{\Rightarrow} \texttt{Return},M \quad (18)$$

$$\frac{G,E \vdash a,M \Rightarrow v}{G,E \vdash (\texttt{return}\ \lfloor a \rfloor),M \overset{\varepsilon}{\Rightarrow} \texttt{Return}(v),M} \quad (19)$$

$$\frac{G,E \vdash a_1,M \Leftarrow \ell \qquad G,E \vdash a_2,M \Rightarrow v \qquad \texttt{storeval}(\texttt{type}(a_1),M,\ell,v) = \lfloor M' \rfloor}{G,E \vdash (a_1 \ = \ a_2),M \overset{\varepsilon}{\Rightarrow} \texttt{Normal},M'} \quad (20)$$

$$\frac{G,E \vdash s_1,M \overset{t_1}{\Rightarrow} \texttt{Normal},M_1 \qquad G,E \vdash s_2,M_1 \overset{t_2}{\Rightarrow} out,M_2}{G,E \vdash (s_1;s_2),M \overset{t_1 \cdot t_2}{\Rightarrow} out,M_2} \quad (21)$$

$$\frac{G,E \vdash s_1,M \overset{t}{\Rightarrow} out,M' \qquad out \neq \texttt{Normal}}{G,E \vdash (s_1;s_2),M \overset{t}{\Rightarrow} out,M'} \quad (22)$$

Coq

# Recall: Semantics for IMP statements

- Assignment:

$$x := e, s \Rightarrow s[x \leftarrow [\![e]\!]\ s]$$

$$\frac{G,E \vdash a_1,M \Leftarrow \ell \qquad G,E \vdash a_2,M \Rightarrow v \qquad \texttt{storeval}(\texttt{type}(a_1),M,\ell,v) = \lfloor M' \rfloor}{G,E \vdash (a_1 \ \texttt{=}\ a_2),M \overset{\varepsilon}{\Rightarrow} \texttt{Normal},M'} \quad (20)$$

- Sequencing:

$$\frac{c_1, s \Rightarrow s_1 \qquad c_2, s_1 \Rightarrow s_2}{c_1; c_2, s \Rightarrow s_2}$$

$$\frac{G,E \vdash s_1,M \overset{t}{\Rightarrow} out,M' \qquad out \neq \texttt{Normal}}{G,E \vdash (s_1;s_2),M \overset{t}{\Rightarrow} out,M'}$$

$$\frac{G,E \vdash s_1,M \overset{t_1}{\Rightarrow} \texttt{Normal},M_1 \qquad G,E \vdash s_2,M_1 \overset{t_2}{\Rightarrow} out,M_2}{G,E \vdash (s_1;s_2),M \overset{t_1.t_2}{\Rightarrow} out,M_2}$$

# Semantics of while loops

$$\frac{G,E \vdash a,M \Rightarrow v \qquad \texttt{is\_false}(v,\texttt{type}(a))}{G,E \vdash (\texttt{while}(a)\ s),M \overset{\varepsilon}{\Rightarrow} \texttt{Normal},M} \quad (23)$$

$$\frac{G,E \vdash a,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a)) \qquad G,E \vdash s,M \overset{t}{\Rightarrow} out,M' \qquad out \overset{loop}{\rightsquigarrow} out'}{G,E \vdash (\texttt{while}(a)\ s),M \overset{t}{\Rightarrow} out',M'} \quad (24)$$

$$\frac{G,E \vdash a,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a)) \qquad G,E \vdash s,M \overset{t_1}{\Rightarrow} (\texttt{Normal} \mid \texttt{Continue}),M_1 \qquad G,E \vdash (\texttt{while}(a)\ s),M_1 \overset{t_2}{\Rightarrow} out',M_2}{G,E \vdash (\texttt{while}(a)\ s),M \overset{t_1.t_2}{\Rightarrow} out',M_2} \quad (25)$$

$$\texttt{Break} \overset{loop}{\rightsquigarrow} \texttt{Normal} \qquad \texttt{Return} \overset{loop}{\rightsquigarrow} \texttt{Return} \qquad \texttt{Return}(v) \overset{loop}{\rightsquigarrow} \texttt{Return}(v)$$

Coq

# Recall: while in IMP

$$\frac{[\![b]\!]\ s = \texttt{false}}{\texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}, s \Rightarrow s}\ \text{[exec\_while\_stop]}$$

$$\frac{[\![b]\!]\ s = \texttt{true} \quad c, s \Rightarrow s_1 \quad \texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}, s_1 \Rightarrow s_2}{\texttt{while}\ b\ \texttt{do}\ c\ \texttt{done}, s \Rightarrow s_2}\ \text{[exec\_while\_loop]}$$

# Judgment of diverging statements

$$G, E \vdash s, M \overset{T}{\Rightarrow} \infty$$

- **G:** *Global environment*
- **E:** *Local environment*
- **s:** *statement to be executed*
- **M:** *Current state of memory*
- **T:** *(infinite) trace of IO events*

## A statement diverges if any of its components diverges.

$$\frac{G,E \vdash s_1,M \overset{T}{\Rightarrow} \infty}{G,E \vdash s_1;s_2,M \overset{T}{\Rightarrow} \infty} \quad (34) \qquad \frac{G,E \vdash s_1,M \overset{t}{\Rightarrow} \texttt{Normal},M_1 \qquad G,E \vdash s_2,M_1 \overset{T}{\Rightarrow} \infty}{G,E \vdash s_1;s_2,M \overset{t.T}{\Rightarrow} \infty} \quad (35)$$

$$\frac{G,E \vdash a,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a)) \qquad G,E \vdash s,M \overset{T}{\Rightarrow} \infty}{G,E \vdash (\texttt{while}(a)\ s),M \overset{T}{\Rightarrow} \infty} \quad (36)$$

$$\frac{G,E \vdash a,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a)) \qquad G,E \vdash s,M \overset{t}{\Rightarrow} (\texttt{Normal} \mid \texttt{Continue}),M_1 \qquad G,E \vdash (\texttt{while}(a)\ s),M_1 \overset{T}{\Rightarrow} \infty}{G,E \vdash (\texttt{while}(a)\ s),M \overset{t.T}{\Rightarrow} \infty} \quad (37)$$

Coq

# Just like commands in IMP diverge when any of their components diverge

$$\frac{c_1, s \Rightarrow \infty}{c_1; c_2, s \Rightarrow \infty}\ \text{[execinf\_seq\_left]} \qquad \frac{c_1, s \Rightarrow s_1 \qquad c_2, s_1 \Rightarrow \infty}{c_1; c_2, s \Rightarrow \infty}\ \text{[execinf\_seq\_right]}$$

$$\frac{\begin{array}{c} c_1, s \Rightarrow \infty \text{ if } [\![b]\!]\ s = \texttt{true} \\ c_2, s \Rightarrow \infty \text{ if } [\![b]\!]\ s = \texttt{false} \end{array}}{\texttt{if } b \texttt{ then } c_1 \texttt{ else } c_2, s \Rightarrow \infty}\ \text{[execinf\_if]}$$

$$\frac{[\![b]\!]\ s = \texttt{true} \qquad c, s \Rightarrow \infty}{\texttt{while } b \texttt{ do } c \texttt{ done}, s \Rightarrow \infty}\ \text{[execinf\_while\_body]}$$

$$\frac{[\![b]\!]\ s = \texttt{true} \qquad c, s \Rightarrow s_1 \qquad \texttt{while } b \texttt{ do } c \texttt{ done}, s_1 \Rightarrow \infty}{\texttt{while } b \texttt{ do } c \texttt{ done}, s \Rightarrow \infty}\ \text{[execinf\_while\_loop]}$$

## Semantics are defined by ~~8~~ 10 judgments

$$G, E \vdash a, M \Leftarrow \ell$$

$$G, E \vdash a, M \Rightarrow v$$

$$G, E \vdash a^*, M \Rightarrow v^*$$

$$G, E \vdash s, M \overset{t}{\Rightarrow} out, M'$$

$$G, E \vdash sw, M \overset{t}{\Rightarrow} out, M'$$

$$G \vdash Fd(v^*), M \overset{t}{\Rightarrow} v, M' \rightarrow$$ **Calling of terminating functions**

$$G, E \vdash s, M \overset{T}{\Rightarrow} \infty$$

$$G, E \vdash sw, M \overset{T}{\Rightarrow} \infty$$

$$G \vdash Fd(v^*), M \overset{T}{\Rightarrow} \infty \longrightarrow$$ **Calling of diverging functions**

$$\vdash P \Rightarrow B$$

# Function calls

$$\frac{G,E \vdash a_{fun},M \Rightarrow \mathtt{ptr}(b,0) \qquad G,E \vdash a_{args},M \Rightarrow v_{args} \qquad \mathtt{funct}(G,b) = \lfloor Fd \rfloor \qquad \mathtt{type\_of\_fundef}(Fd) = \mathtt{type}(a_{fun}) \qquad G \vdash Fd(v_{args}),M \overset{t}{\Rightarrow} v_{res},M'}{G,E \vdash a_{fun}(a_{args}),M \overset{t}{\Rightarrow} v_{res},M'} \quad (30)$$

$$\frac{G,E \vdash a,M \Leftarrow \ell \qquad G,E \vdash a_{fun},M \Rightarrow \mathtt{ptr}(b,0) \qquad G,E \vdash a_{args},M \Rightarrow v_{args} \qquad \mathtt{funct}(G,b) = \lfloor Fd \rfloor \qquad \mathtt{type\_of\_fundef}(Fd) = \mathtt{type}(a_{fun}) \qquad G \vdash Fd(v_{args}),M \overset{t}{\Rightarrow} v_{res},M_1 \qquad \mathtt{storeval}(\mathtt{type}(a),M_1,\mathtt{ptr}(\ell),v_{res}) = \lfloor M_2 \rfloor}{G,E \vdash a = a_{fun}(a_{args}),M \overset{t}{\Rightarrow} v_{res},M_2} \quad (31)$$

Coq

# Function invocation

$$\frac{\begin{array}{c} F = \tau\ id(dcl_1)\{dcl_2; s\} \\ \texttt{alloc\_vars}(M, dcl_1 + dcl_2, E) = (M_1, b^*) \qquad \texttt{bind\_params}(E, M_1, dcl_1, v_{args}) = M_2 \\ G, E \vdash s, M_2 \stackrel{t}{\Rightarrow} out, M_3 \qquad out,\ \tau \# v_{res} \end{array}}{G \vdash F(v_{args}), M \stackrel{t}{\Rightarrow} v_{res}, \texttt{free}(M_3, b^*)} \quad (32)$$

$$\frac{Fe = \texttt{extern}\ \tau\ id(dcl) \qquad \nu = id(v_{args}, v_{res})}{G \vdash Fe(v_{args}), M \stackrel{\nu}{\Rightarrow} v_{res}, M} \quad (33)$$

$\texttt{Normal}, \texttt{void} \# \texttt{undef}$ $\texttt{Return}, \texttt{void} \# \texttt{undef}$ $\texttt{Return}(v), \tau \# v$ when $\tau \neq \texttt{void}$

## 2009 to 2016
## Big step to Small step

- CompCert changed semantics model to small step: $(s, m) \rightarrow (s', m')$
- Small step based on continuations

- Coq

# Questions?

## Judgment for expressions (in right position)

$$G, E \vdash a, M \Rightarrow v$$

- **G:** *Global environment*
- **E:** *Local environment*
- **a:** *expression to be evaluated*
- **M:** *Current state of memory*
- **v:** *result value, either:*
  - `int(n)`
  - `float(f)`
  - `ptr(l)`
  - `undef`

# Semantics for expressions in right position

$$G,E \vdash n,M \Rightarrow \texttt{int}(n) \quad (5) \qquad G,E \vdash f,M \Rightarrow \texttt{float}(f) \quad (6)$$

$$G,E \vdash \texttt{sizeof}(\tau),M \Rightarrow \texttt{int}(\texttt{sizeof}(\tau)) \quad (7) \qquad \frac{G,E \vdash a,M \Leftarrow \ell \qquad \texttt{loadval}(\texttt{type}(a),M',\ell) = \lfloor v \rfloor}{G,E \vdash a,M \Rightarrow v} \quad (8)$$

$$\frac{G,E \vdash a,M \Leftarrow \ell}{G,E \vdash \&a,M \Rightarrow \texttt{ptr}(\ell)} \quad (9) \qquad \frac{G,E \vdash a_1,M \Rightarrow v_1 \qquad \texttt{eval\_unop}(op_1,v_1,\texttt{type}(a_1)) = \lfloor v \rfloor}{G,E \vdash op_1\ a_1,M \Rightarrow v} \quad (10)$$

$$\frac{G,E \vdash a_1,M \Rightarrow v_1 \qquad G,E \vdash a_2,M_1 \Rightarrow v_2 \qquad \texttt{eval\_binop}(op_2,v_1,\texttt{type}(a_1),v_2,\texttt{type}(a_2)) = \lfloor v \rfloor}{G,E \vdash a_1\ op_2\ a_2,M \Rightarrow v} \quad (11)$$

$$\frac{G,E \vdash a_1,M \Rightarrow v_1 \qquad \texttt{is\_true}(v_1,\texttt{type}(a_1)) \qquad G,E \vdash a_2,M \Rightarrow v_2}{G,E \vdash a_1\ \texttt{?}\ a_2\ \texttt{:}\ a_3,M \Rightarrow v_2} \quad (12)$$

$$\frac{G,E \vdash a_1,M \Rightarrow v_1 \qquad \texttt{is\_false}(v_1,\texttt{type}(a_1)) \qquad G,E \vdash a_3,M \Rightarrow v_3}{G,E \vdash a_1\ \texttt{?}\ a_2\ \texttt{:}\ a_3,M \Rightarrow v_3} \quad (13)$$

$$\frac{G,E \vdash a,M \Rightarrow v_1 \qquad \texttt{cast}(v_1,\texttt{type}(a),\tau) = \lfloor v \rfloor}{G,E \vdash (\tau)a,M \Rightarrow v} \quad (14)$$

# Semantics for (a1 op a2)

$$\frac{s_1 \neq \texttt{skip} \qquad G,E \vdash s_1,M \overset{t_1}{\Rightarrow} \texttt{Normal},M_1 \qquad G,E \vdash (\texttt{for}(\texttt{skip},a_2,s_3)\ s),M_1 \overset{t_2}{\Rightarrow} out,M_2}{G,E \vdash (\texttt{for}(s_1,a_2,s_3)\ s),M \overset{t_1.t_2}{\Rightarrow} out,M_2} \quad (26)$$

$$\frac{G,E \vdash a_2,M \Rightarrow v \qquad \texttt{is\_false}(v,\texttt{type}(a_2))}{G,E \vdash (\texttt{for}(\texttt{skip},a_2,s_3)\ s),M \overset{\varepsilon}{\Rightarrow} \texttt{Normal},M} \quad (27)$$

$$\frac{G,E \vdash a_2,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a_2)) \qquad G,E \vdash s,M \overset{t_1}{\Rightarrow} out_1,M_1 \qquad out_1 \overset{loop}{\rightsquigarrow} out}{G,E \vdash (\texttt{for}(\texttt{skip},a_2,s_3)\ s),M \overset{t}{\Rightarrow} out,M_1} \quad (28)$$

$$\frac{\begin{array}{c} G,E \vdash a_2,M \Rightarrow v \qquad \texttt{is\_true}(v,\texttt{type}(a_2)) \\ G,E \vdash s,M \overset{t_1}{\Rightarrow} (\texttt{Normal} \mid \texttt{Continue}),M_1 \qquad G,E \vdash s_3,M_1 \overset{t_2}{\Rightarrow} \texttt{Normal},M_2 \\ G,E \vdash (\texttt{for}(\texttt{skip},a_2,s_3)\ s),M_2 \overset{t_3}{\Rightarrow} out,M_3 \end{array}}{G,E \vdash (\texttt{for}(\texttt{skip},a_2,s_3)\ ),M \overset{t_1.t_2.t_3}{\Rightarrow} out,M_3} \quad (29)$$