The Static Semantics of C0

C0deine v24.01.1

1 Introduction

This document contains static semantics of C0, as implemented by the C0deine compiler. All the rules are auto-generated from their representation within the compiler[1]. Since everything is generated and the tool to do this is experimental, the entire static semantics is not listed here quite yet. More specifically, here is a brief list of what is not currently included in this document (that the compiler does verify):

- Some semantics about statements (all paths return, some annotations)
- Variables initialisation
- Restrictions on where `\result` and `\length` can be placed
- Semantics of function application

If you notice any bugs/errors beyond these elements please report them here.

2 Preface

2.1 Elaboration

Before reaching the typechecking phase C0deine does a number of elaborations. So, there will not be rules for anything that was elaborated away. Here is a summary of what is removed in elaboration:

- Compiler directives are elaborated and removed
- For-loops are elaborated into equivalent declaration and while-loop
- Post-ops (`++`, `--`) are elaborated to `asnops` (`+=`, `-=`)
- The arrow, `e->field`, operator is elaborated to `(*e).field`

[1] namely the Typed Syntax Tree

2.2 Utility Functions and Definitions

Some of the rules also use additional helper functions:

- `isEqualityOp` *op* checks whether a comparison operator is an equality operator (either `==` or `!=`).
- τ_1 `=` τ_2 checks whether two types are identical (structurally equal)
- `equiv` τ_1 τ_2 checks whether two types are simply equivalent (e.g. `any *` is equivalent to `int *`)
- `intersect` τ_1 τ_2 intersects two types (e.g. intersecting `any` and `int` is `int`)
- `UnOp.type` *op* gets is the type that *op* is operating on
- `FCtx.updateVar` Γ x τ updates the context Γ such that x has type τ.
- `type` *name* gets the type of the name.
- `is_var` *lv* checks if a LValue is a variable.

3 Expressions

The typing judgement for expressions is as follows:

$$\Delta;\ \Gamma \vdash e : \tau$$

where τ is `Typ` (the type), Γ is an `FCtx` (the function symbol context), and Δ is the `GCtx` (the global symbol context).

Again, the rule for function application is missing since it is not yet implemented into the rule-generating tool. Also missing the rule for `\result`.

$$\frac{}{\Delta;\ \Gamma\ \vdash\ n\ :\ \texttt{int}}\ \textsc{num} \qquad \frac{}{\Delta;\ \Gamma\ \vdash\ c\ :\ \texttt{char}}\ \texttt{CHAR}$$

$$\frac{}{\Delta;\ \Gamma\ \vdash\ s\ :\ \texttt{string}}\ \textsc{str}$$

$$\frac{\Gamma\ x\ =\ \texttt{some}\ (\texttt{Status.Symbol.var}\ \tau)}{\Delta;\ \Gamma\ \vdash\ x\ :\ \tau}\ \textsc{var}$$

$$\frac{}{\Delta;\ \Gamma\ \vdash\ \texttt{true}\ :\ \texttt{bool}}\ \texttt{TRUE} \qquad \frac{}{\Delta;\ \Gamma\ \vdash\ \texttt{false}\ :\ \texttt{bool}}\ \texttt{FALSE}$$

$$\frac{}{\Delta;\ \Gamma\ \vdash\ \texttt{null}\ :\ (\texttt{any})\texttt{*}}\ \textsc{null}$$

$$\frac{\texttt{equiv}\ \tau\ (\texttt{UnOp.type}\ op)\ =\ \texttt{true} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\ \tau}{\Delta;\ \Gamma\ \vdash\ op\ e0\ :\ \texttt{UnOp.type}\ op}\ \textsc{unop}$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\\ \tau_2\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow\tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ \texttt{int\_}op\ e0\ e1\ :\ \texttt{int}}\ \textsc{binop\_int}$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \}\\ \tau_2\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow\tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ \texttt{bool\_}op\ e0\ e1\ :\ \texttt{bool}}\ \textsc{binop\_bool}$$

$$\frac{\begin{array}{c}\texttt{isEqualityOp}\ op\ =\ \texttt{true}\\ \Delta;\ \Gamma\ \vdash\ e0\ :\ \tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\ \tau_2 \qquad \texttt{equiv}\ \tau_1\ \tau_2\ =\ \texttt{true}\end{array}}{\Delta;\ \Gamma\ \vdash\ op\ e0\ e1\ :\ \texttt{bool}}\ \textsc{binop\_eq}$$

//

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \} \qquad \tau_2\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\\ \neg\texttt{isEqualityOp}\ op\ =\ \texttt{true} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow\tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ op\ e0\ e1\ :\ \texttt{bool}}\ \textsc{binop\_rel}1$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{char})\ \} \qquad \tau_2\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{char})\ \}\\ \neg\texttt{isEqualityOp}\ op\ =\ \texttt{true} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow\tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ op\ e0\ e1\ :\ \texttt{bool}}\ \textsc{binop\_rel}2$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow\tau_1\\ \Delta;\ \Gamma\ \vdash\ e1\ :\ \tau_2 \qquad \Delta;\ \Gamma\ \vdash\ e2\ :\ \tau_3 \qquad \texttt{equiv}\ \tau_2\ \tau_3\ =\ \texttt{true}\end{array}}{\Delta;\ \Gamma\ \vdash\ (e0\ ?\ e1\ :\ e2)\ :\ \texttt{intersect}\ \tau_2\ \tau_3}\ \textsc{ternop}$$

$$\frac{}{\Delta;\ \Gamma\ \vdash\ \texttt{alloc}(\tau)\ :\ \tau\texttt{*}}\ \texttt{ALLOC}$$

$$\frac{\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau_1}{\Delta;\ \Gamma\ \vdash\ \texttt{alloc\_array}(\tau,\ e0)\ :\ \tau\texttt{[]}}\ \texttt{ALLOC\_ARRAY}$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{struct}s)\ \}\qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau_1\\ \texttt{GCtx.struct}\ \Delta\ s\ =\ \texttt{some}\,\{\ \mathit{fields}\ :=\ \mathit{struct\_fields},\ \mathit{defined}\ :=\ \texttt{true}\ \}\\ \mathit{struct\_fields}\ \mathit{field}\ =\ \texttt{some}\ \tau\end{array}}{\Delta;\ \Gamma\ \vdash\ e0.\mathit{field}\ :\ \tau}\ \text{DOT}$$

$$\frac{\tau_1\ \in\ \{\ \tau'\ //\ \tau'\ =\ (\tau\texttt{*})\ \}\qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau_1}{\Delta;\ \Gamma\ \vdash\ \texttt{*}e0\ :\ \tau}\ \text{DEREF}$$

$$\frac{\begin{array}{c}\tau_1\ \in\ \{\ \tau'\ //\ \tau'\ =\ (\tau\texttt{[]})\ \}\\ \tau_2\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau_1\qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow \tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ e0[e1]\ :\ \tau}\ \text{INDEX}$$

$$\frac{\tau_1\ \in\ \{\ \tau'\ //\ \tau'\ =\ (\tau\texttt{[]})\ \}\qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau_1}{\Delta;\ \Gamma\ \vdash\ \texttt{\textbackslash length}\,e0\ :\ \texttt{int}}\ \text{LENGTH}$$

4 LValues

The typing judgement for lvalues is as follows:

$$\Delta;\ \Gamma \vdash lv : \tau$$

where τ is `Typ` (the type), Γ is an `FCtx` (the function symbol context), and Δ is the `GCtx` (the global symbol context).

Currently doesn't express that the index expression cannot contain `\length` nor `\result`

$$\frac{\Gamma\ x\ =\ \texttt{some}\ (\texttt{Status.Symbol.var}\ \tau)}{\Delta;\ \Gamma\ \vdash\ x\ :\ \tau}\ \text{VAR}$$

$$\frac{\begin{array}{c}\tau_1\ :\ \{\ \tau\ //\ \tau\ =\ (\texttt{struct}s)\ \}\qquad \Delta;\ \Gamma\ \vdash\ lv0\ :\uparrow\tau_1\\ \texttt{GCtx.struct}\ \Delta\ s\ =\ \texttt{some}\ \{\ fields\ :=\ fields,\ defined\ :=\ \texttt{true}\ \}\\ fields\ field\ =\ \texttt{some}\ \tau\end{array}}{\Delta;\ \Gamma\ \vdash\ lv0.field\ :\ \tau}\ \text{DOT}$$

$$\frac{\tau_1\ :\ \{\ \tau'\ //\ \tau'\ =\ (\tau\texttt{*})\ \}\qquad \Delta;\ \Gamma\ \vdash\ lv0\ :\uparrow\tau_1}{\Delta;\ \Gamma\ \vdash\ \texttt{*}lv0\ :\ \tau}\ \text{DEREF}$$

$$\frac{\begin{array}{c}\tau_1\ :\ \{\ \tau'\ //\ \tau'\ =\ (\tau\texttt{[]})\ \}\\ \tau_2\ :\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\qquad \Delta;\ \Gamma\ \vdash\ lv0\ :\uparrow\tau_1\qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ lv0[e1]\ :\ \tau}\ \text{INDEX}$$

5 Annotations

The (typing) judgement for annoations is as follows:

$$\Delta;\ \Gamma \vdash \mathit{anno}\ \mathbf{valid}$$

where Γ is an `FCtx` (the function symbol context), and Δ is the `GCtx` (the global symbol context).

Currently missing enforcement that the expression doesn't contain `\result` outside of `//@ensures`

$$\frac{\tau\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau}{\Delta;\ \Gamma\ \vdash\ \texttt{//@requires}\ e0;\ \mathbf{valid}}\ \textsc{requires}$$

$$\frac{\tau\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau}{\Delta;\ \Gamma\ \vdash\ \texttt{//@ensures}\ e0;\ \mathbf{valid}}\ \textsc{ensures}$$

$$\frac{\tau\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau}{\Delta;\ \Gamma\ \vdash\ \texttt{//@loop\_invariant}\ e0;\ \mathbf{valid}}\ \textsc{loop\_invar}$$

$$\frac{\tau\ \in\ \{\ \tau\ //\ \tau\ =\ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma\ \vdash\ e0\ :\uparrow \tau}{\Delta;\ \Gamma\ \vdash\ \texttt{//@assert}\ e0;\ \mathbf{valid}}\ \textsc{assert}$$

6 Statements

The typing judgement for statements is as follows:

$$\Delta;\ \Gamma \vdash \mathit{stmt};\ :\ [\rho]$$

where $[\rho]$ is `Option Typ` (the function return type), Γ is an `FCtx` (the function symbol context), and Δ is the `GCtx` (the global symbol context).

Currently doesn't that the expressions cannot contain `\length` nor `\result`. Also doesn't include loop annotations.

$$\frac{}{\Delta;\ \Gamma\ \vdash\ \texttt{nop};\ :\ [\rho]}\ \textsc{nil}$$

$$\frac{\Delta;\ \Gamma\ \vdash\ \mathit{stmt}0\ :\ [\rho] \qquad \Delta;\ \Gamma\ \vdash\ \mathit{body}1\ :\ [\rho]}{\Delta;\ \Gamma\ \vdash\ \texttt{seq}(\mathit{stmt}0,\ \mathit{body}1);\ :\ [\rho]}\ \textsc{cons}$$

$$\frac{\begin{array}{c}\mathit{new\_ctx}\ :\ \Gamma'\ =\ \texttt{FCtx.updateVar}\,\Gamma\ (\ \mathit{name})\ (\texttt{type}\ \mathit{name})\\ \Delta;\ \Gamma'\ \vdash\ \mathit{body}0\ :\ [\rho]\end{array}}{\Delta;\ \Gamma\ \vdash\ \texttt{decl}(\mathit{name},\ \mathit{body}0);\ :\ [\rho]}\ \texttt{DECL}$$

$$\frac{\begin{array}{c}\Delta;\ \Gamma\ \vdash\ e0\ :\ \tau \qquad \mathit{ty\_equiv}\ :\ \texttt{equiv}\ (\texttt{type}\ \mathit{name})\ \tau\ =\ \texttt{true}\\ \mathit{new\_ctx}\ :\ \Gamma'\ =\ \texttt{FCtx.updateVar}\,\Gamma\ (\ \mathit{name})\ (\texttt{type}\ \mathit{name})\\ \Delta;\ \Gamma'\ \vdash\ \mathit{body}1\ :\ [\rho]\end{array}}{\Delta;\ \Gamma\ \vdash\ \texttt{decl\_init}(\mathit{name},\ e0,\ \mathit{body}1);\ :\ [\rho]}\ \texttt{DECL\_INIT}$$

$$\frac{\begin{array}{c}\Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ :\ \tau_1 \qquad \mathit{is\_var}\ :\ \texttt{is\_var}\ \mathit{lhs}\ =\ \texttt{true}\\ \Delta;\ \Gamma\ \vdash\ e1\ :\ \tau_2 \qquad \mathit{ty\_equiv}\ :\ \texttt{equiv}\ \tau_1\ \tau_2\ =\ \texttt{true}\end{array}}{\Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ =\ e1;\ :\ [\rho]}\ \textsc{assign\_var}$$

$$\frac{\begin{array}{c}\Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ :\ \tau_1 \qquad \mathit{is\_var}\ :\ \neg\texttt{is\_var}\ \mathit{lhs}\ =\ \texttt{true}\\ \Delta;\ \Gamma\ \vdash\ e1\ :\ \tau_2 \qquad \mathit{ty\_equiv}\ :\ \texttt{equiv}\ \tau_1\ \tau_2\ =\ \texttt{true}\end{array}}{\Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ =\ e1;\ :\ [\rho]}\ \textsc{assign}$$

$$\frac{\begin{array}{c}\tau_1\ :\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \}\\ \tau_2\ :\ \{\ \tau\ //\ \tau\ =\ (\texttt{int})\ \} \qquad \Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ :\uparrow\tau_1 \qquad \Delta;\ \Gamma\ \vdash\ e1\ :\uparrow\tau_2\end{array}}{\Delta;\ \Gamma\ \vdash\ \mathit{lv}0\ \texttt{int\_}\mathit{op}=\ e1;\ :\ [\rho]}\ \textsc{asnop}$$

$$\frac{\Delta;\ \Gamma\ \vdash\ e0\ :\ \tau}{\Delta;\ \Gamma\ \vdash\ e0;\ :\ [\rho]}\ \textsc{expr}$$

$$\frac{\begin{array}{c}\tau \ : \ \{\ \tau \,//\, \tau \ = \ (\texttt{bool})\ \} \\ \Delta;\ \Gamma \ \vdash \ e0 \ :\uparrow \tau \qquad \Delta;\ \Gamma \ \vdash \ body1 \ : \ [\rho] \qquad \Delta;\ \Gamma \ \vdash \ body2 \ : \ [\rho]\end{array}}{\Delta;\ \Gamma \ \vdash \ \texttt{if}(e0,\ body1,\ body2);\ : \ [\rho]} \ \textsc{ite}$$

$$\frac{\tau \ : \ \{\ \tau \,//\, \tau \ = \ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma \ \vdash \ e0 \ :\uparrow \tau \qquad \Delta;\ \Gamma \ \vdash \ body1 \ : \ [\rho]}{\Delta;\ \Gamma \ \vdash \ \texttt{while}(e0,\ body1);\ : \ [\rho]} \ \texttt{WHILE}$$

$$\frac{is\_void \ : \ \texttt{isNone}\ \rho \ = \ \texttt{true}}{\Delta;\ \Gamma \ \vdash \ \texttt{return} \ : \ [\rho]} \ \texttt{RETURN}$$

$$\frac{\tau_1 \ : \ \{\ \tau' \,//\, \tau' \ = \ \tau\ \} \qquad \Delta;\ \Gamma \ \vdash \ e0 \ :\uparrow \tau_1}{\Delta;\ \Gamma \ \vdash \ \texttt{return}\ e0;\ : \ [\texttt{some}\ \tau]} \ \texttt{RETURN}$$

$$\frac{\tau \ : \ \{\ \tau \,//\, \tau \ = \ (\texttt{bool})\ \} \qquad \Delta;\ \Gamma \ \vdash \ e0 \ :\uparrow \tau}{\Delta;\ \Gamma \ \vdash \ \texttt{assert}(e0);\ : \ [\rho]} \ \texttt{ASSERT}$$

$$\frac{\tau \ : \ \{\ \tau \,//\, \tau \ = \ (\texttt{string})\ \} \qquad \Delta;\ \Gamma \ \vdash \ e0 \ :\uparrow \tau}{\Delta;\ \Gamma \ \vdash \ \texttt{error}(e0);\ : \ [\rho]} \ \texttt{ERROR}$$

$$\frac{\Delta;\ \Gamma \ \vdash \ anno0\ \textbf{valid}}{\Delta;\ \Gamma \ \vdash \ anno0 \ : \ [\rho]} \ \textsc{anno}$$