CLE Type System

Benjamin Flin

August 2021

1 Introduction

2 CLE Types Grammar

The following is a small grammar for CLE types τ. l and r represent enclaves and can be thought of as arbitrary identifiers. $\alpha, \beta, \theta, \sigma$ are sets of remote enclaves, while δ is a set of μ.

$$\tau, \pi, \gamma ::= l\ \sigma \mid l\ \delta$$
$$\mu ::= r\ (\alpha_1, \ldots, \alpha_n) \rightarrow_\phi \theta$$

$l\ \sigma$ describes a variable in enclave l which can be shared with enclaves inside σ. The inhabitants of $l\ \delta$ are functions in the enclave l. For every $r\ (\alpha_1, \ldots, \alpha_n) \rightarrow_\phi \theta \in \delta$ every function whose type is $l\ \delta$ can be called from an enclave r with arguments whose shareability is $\alpha_1, \ldots, \alpha_n$, whose function body can share with ϕ and whose return is able to be shared with θ.

The terms of the inhabitants of these types are an idealized subset of llvm, where the only control flow instructions are breaks and function calls.

3 Type rules

Here we assume all functions and global variables have cle types associated with them. We will focus on how to infer such types in the next section. There are several types of judgements, each of which is enumerated below:

3.1 Judgements

1. $\Gamma \vdash e : \tau$. Top-level entity or instruction e has type τ.
2. $\Gamma \vdash b :_\pi \gamma$. Basic block or set of basic blocks b has type π for all variables bound in instructions, type γ for the terminator.
3. $\Gamma \vdash t :_\pi \gamma$. Terminator t has type γ and all referenced basic blocks, b are given type ${}_\pi\gamma$.

3.2 Rules for top-level entities

$$\frac{\tau = l\ \sigma}{\Gamma \vdash @x : \tau}\ \text{global-decl} \qquad \frac{\tau = l\ \sigma}{\Gamma \vdash @x = c : \tau}\ \text{global-def}$$

$$\frac{\forall (r\ (\alpha_1, \ldots, \alpha_n) \to_\phi \theta) \in \delta.\ \Gamma[@f \mapsto \tau, \%1 \mapsto l\ \alpha_1, \ldots, \%n \mapsto l\ \alpha_n] \vdash body :_{(l\ \phi)} l\ \theta}{\Gamma \vdash @f(\%1, \ldots, \%n)\ \{body\} : l\ \delta}\ \text{fn-def}$$

$$\frac{\Gamma(@f) = \tau = l\ \delta}{\Gamma \vdash @f(\%1, \ldots, \%n)\ \{body\} : \tau}\ \text{fn-def-known}$$

3.3 Rules for basic blocks and instruction lists

$$\frac{\Gamma \vdash b :_\pi \gamma \qquad \Gamma, \%b :_\pi \gamma \vdash bbs :_\pi \gamma}{\Gamma \vdash b\ bbs :_\pi \gamma}\ \text{fn-body}$$

$$\frac{\Gamma \vdash instrs : \pi \qquad \Gamma \vdash term :_\pi \gamma}{\Gamma \vdash \%b : instrs\ term :_\pi \gamma}\ \text{bb-unknown}$$

$$\frac{\Gamma(\%b) =_\pi \gamma}{\Gamma \vdash \%b : instrs\ term :_\pi \gamma}\ \text{bb-known}$$

$$\frac{\Gamma \vdash instr : \pi \qquad \Gamma[\%a \mapsto \pi] \vdash instrs : \pi}{\Gamma \vdash \%a = instr; instrs : \pi}\ \text{instrs}$$

3.4 Rules for special instructions and terminators

$$\frac{\Gamma(\%1) = l\ \alpha_1, \ldots, \Gamma(\%n) = l\ \alpha_n \qquad l\ (\alpha_1, \ldots, \alpha_n) \to_\phi \theta \in \delta \qquad \Gamma \vdash @f : r\ \delta}{\Gamma \vdash \text{call}\ @f(\%1, \ldots, \%n) : l\ \theta}\ \text{call}$$

$$\frac{\Gamma(\%a) = \gamma \qquad \Gamma \vdash \%b_1 :_\pi \gamma \qquad \Gamma \vdash \%b_2 :_\pi \gamma}{\Gamma \vdash \text{br}\ \%a, \%b_1, \%b_2 :_\pi \gamma}\ \text{break}$$

$$\frac{\Gamma(\%a) = \gamma}{\Gamma \vdash \text{ret}\ \%a :_\pi \gamma}\ \text{ret}$$

3.5 General rule for instructions

The rules for all other instructions can be derived from general instruction form which takes in a number of arguments:

$$\text{instr}\ \%a_1, \ldots, \%a_n$$

Thus,

$$\frac{\Gamma(\%a_1) = \cdots = \Gamma(\%a_n) = \pi}{\Gamma \vdash \text{instr}\ \%a_1, \ldots \%a_n : \pi}\ \text{ret}$$

4 Conversion from CLE annotations