Variables $x, y \in \mathsf{Var}$.

Type names $T \in \mathsf{Name}$. Both type variables and names of things like headers are represented by type names.

Field names $f \in \mathsf{Name}$.

Integer literals $m, n \in \mathbb{Z}$.

Unary operators $\ominus \in \{\dots\}$.

Binary operators $\odot \in \{\dots\}$.

Directions for function arguments.

$$
\begin{array}{rcl}
p & ::= & \mathsf{in} \\
 & | & \mathsf{out} \\
 & | & \mathsf{inout}
\end{array}
$$

Expressions.

$$
\begin{array}{rcl}
e & ::= & \mathsf{true} \\
 & | & \mathsf{false} \\
 & | & n \\
 & | & m \mathsf{s} n \\
 & | & m \mathsf{u} n \\
 & | & x \\
 & | & .f \\
 & | & e_1[e_2] \\
 & | & e_1[e_2 : e_3] \\
 & | & \{\overline{e}\} \\
 & | & \ominus e \\
 & | & e_1 \odot e_2 \\
 & | & (\tau) e \\
 & | & \tau.f \\
 & | & \mathsf{error}.f \\
 & | & e_1 \,?\, e_2 : e_3 \\
 & | & e\langle\overline{\tau}\rangle(\overline{e}) \\
 & | & e(\overline{e}) \\
 & | & e_1 \,\&\&\&\, e_2 \\
 & | & e_1 .. e_2
\end{array}
$$

Blocks and statements.

$$
\begin{array}{rcl}
B & ::= & \{\overline{s}\} \\
s & ::= & B \\
 & | & D \\
 & | & e\langle\overline{\tau}\rangle(\overline{x:\tau\ d}) \\
 & | & x = e \\
 & | & \tau(\overline{x:\tau\ d}) \\
 & | & \mathsf{if}\ (e)\ s\ \mathsf{else}\ s \\
 & | & \mathsf{exit} \\
 & | & \mathsf{nop} \\
 & | & \mathsf{return}\ e \\
 & | & \mathsf{switch}\ e\ \{\mathsf{TODO}\}
\end{array}
$$

Declarations.

$$
\begin{array}{rcl}
D & ::= & \mathsf{const}\ x:\tau = e \\
 & | & x:\tau = e \\
 & | & \tau(\overline{e})\ x \\
 & | & \mathsf{header}\ T\ \{\overline{f:\tau}\} \\
 & | & \mathsf{header\_union}\ T\ \{\overline{f:\tau}\} \\
 & | & \mathsf{struct}\ T\ \{\overline{f:\tau}\} \\
 & | & \mathsf{error}.f : \mathsf{error} \\
 & | & \mathsf{enum}\ T\ \{\overline{f}\} \\
 & | & \mathsf{enum}\ T\ \{\overline{f:e}\} \\
 & | & \mathsf{newtype}\ T\ \tau \\
 & | & \mathsf{typedef}\ T\ \tau \\
 & | & \mathsf{package}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p}) \\
 & | & \mathsf{control}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p}) \\
 & | & \mathsf{control}\ T(\overline{x:\tau\ p})(\overline{x:\tau})\ \{\overline{D}\ \mathsf{apply}\ B\} \\
 & | & \mathsf{action}\ f(\overline{x:\tau\ p}, \overline{x:\tau})B \\
 & | & \mathsf{parser}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p}) \\
 & | & \mathsf{parser}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p})(\overline{x:\tau})\ \{\mathsf{TODO}\} \\
 & | & \mathsf{value\_set}\langle\tau\rangle(e)\ T \\
 & | & \mathsf{function}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p})\ \{\mathsf{TODO}\} \\
 & | & \mathsf{extern\ function}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p})\ \{\mathsf{TODO}\} \\
 & | & \mathsf{extern}\ T\langle\overline{T}\rangle\ \{\mathsf{TODO}\} \\
 & | & \mathsf{table}\ T\ \mathsf{TODO}
\end{array}
$$

Types.

$$
\begin{array}{rcl}
\tau & ::= & \mathsf{header}\ T\ \{\overline{f : \tau}\} \\
& | & \tau_1 \times \cdots \times \tau_n \\
& | & (\overline{x : \tau\ d}) \to \tau' \\
& | & (\overline{x : \tau}) \to \tau' \\
& | & \tau\ \mathsf{set} \\
& | & \mathsf{header\_union}\ T\ \{\overline{f : \tau}\} \\
& | & T\ \mathsf{stack}\langle n \rangle \\
& | & \mathsf{struct}\ T\ \{\overline{f : \tau}\} \\
& | & \mathsf{error}.f : \mathsf{error} \\
& | & \mathsf{enum}\ T\ \{\overline{f}\} \\
& | & \mathsf{enum}\ T\ \{\overline{f : e}\} \\
& | & \mathsf{extern}\ T\langle\overline{T}\rangle\ \{\overline{m}\} \\
& | & \mathsf{newtype}\ T\ \tau \\
& | & \mathsf{control}\ T\langle\overline{T}\rangle(\overline{x : \tau\ p}) \\
& | & \mathsf{parser}\ T\langle\overline{T}\rangle(\overline{x : \tau\ p}) \\
& | & \mathsf{package}\ T\langle\overline{T}\rangle(\overline{x : \tau})
\end{array}
$$

Base types.

$$
\begin{array}{rcl}
\sigma & ::= & \mathsf{void} \\
& | & \mathsf{error} \\
& | & \mathsf{bool} \\
& | & \mathsf{int} \\
& | & \mathsf{int}\langle n \rangle \\
& | & \mathsf{bit}\langle n \rangle \\
& | & \mathsf{varbit}\langle n \rangle
\end{array}
$$

$$
\begin{array}{rcl}
\hat{\tau} & ::= & .T \\
& | & T \\
& | & \alpha \\
& | & T\langle\overline{\hat{\tau}}\rangle
\end{array}
$$

Type contexts $\Delta = T_1 : \tau_1, \ldots, T_n : \tau_n$ where all T_i are distinct. Declaration contexts $\Xi = D_1, D_2, \ldots, D_n$. Contexts $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$.

Typing judgements for expressions.

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

Rules for expression typing: booleans. TODO ternary operator.

$$\frac{}{\Xi;\Delta;\Gamma \vdash \mathsf{true} : \mathsf{bool}} \quad \frac{}{\Xi;\Delta;\Gamma \vdash \mathsf{false} : \mathsf{bool}}$$

$$\frac{\ominus \in \{!\} \qquad \Xi;\Delta;\Gamma \vdash e : \mathsf{bool}}{\Xi;\Delta;\Gamma \vdash \ominus e : \mathsf{bool}}$$

$$\frac{\odot \in \{\&\&, ||\} \qquad \Xi;\Delta;\Gamma \vdash e_1 : \mathsf{bool} \qquad \Xi;\Delta;\Gamma \vdash e_2 : \mathsf{bool}}{\Xi;\Delta;\Gamma \vdash e_1 \odot e_2 : \mathsf{bool}}$$

Rules for expression typing: integers. TODO: operations on integers.

$$\frac{}{\Xi;\Delta;\Gamma \vdash n : \mathsf{int}} \quad \frac{m > 1}{\Xi;\Delta;\Gamma \vdash msn : \mathsf{int}\langle m\rangle} \quad \frac{}{\Xi;\Delta;\Gamma \vdash mun : \mathsf{bit}\langle m\rangle}$$

Rules for expression typing: variables.

$$\frac{\mathsf{lookup}\,\Gamma\, x = \tau}{\Xi;\Delta;\Gamma \vdash x : \tau}$$

Rules for expresion typing: array operations. TODO
Rules for expresion typing: sets. TODO
Rules for expresion typing: casts. TODO
Rules for expresion typing: errors. TODO
Rules for expresion typing: set operations. TODO
Rules for expresion typing: applications.

$$\frac{\Xi;\Delta;\Gamma \vdash e : (\overline{\tau\; p}) \to \tau' \qquad e_i : \tau_i \qquad \mathsf{dir}(e_i) \sqsubseteq p_i}{\Xi;\Delta;\Gamma \vdash e(\bar{e}) : \tau}$$

Rules for expression typing: top level fields .f
Rules for expresion typing: type members $\tau.f$.
Typing judgements for statements.

$$\Xi;\Delta;\Gamma \vdash s : \tau \dashv \Gamma'$$

Typing judgements for declarations.

$$\Xi;\Delta;\Gamma \vdash D : \tau \dashv \Gamma';\Delta'$$

Declarations.

$$\begin{array}{rcl} D & ::= & \tau(\overline{e})\, x \\ & | & \mathsf{package}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p}) \\ & | & \mathsf{control}\ T\langle\overline{T}\rangle(\overline{x:\tau\ p}) \\ & | & \mathsf{control}\ T(\overline{x:\tau\ p})(\overline{x:\tau})\ \{\overline{D}\ \mathsf{apply}\ B\} \end{array}$$

Types.

$$\begin{array}{rcl} \tau & ::= & T \\ & | & \mathsf{control}\ \langle\overline{T}\rangle(\overline{x:\tau\ p}) \\ & | & \mathsf{parser}\ \langle\overline{T}\rangle(\overline{x:\tau\ p}) \\ & | & \mathsf{package}\ \langle\overline{T}\rangle(\overline{x:\tau}) \end{array}$$

Expressions.

$$\begin{array}{rcl} e & ::= & x \\ & | & \tau(\overline{e}) \end{array}$$

How should we typecheck the following program?

$$\begin{array}{l} \mathsf{control}\ A(); \\ \mathsf{package}\ P(a:A); \\ \mathsf{control}\ C()\{\}; \\ \mathsf{control}\ T()\ \{\mathsf{apply}\ \{\}\} \\ A(C())\ x; \end{array}$$

Our judgment for checking declarations has the shape $\Delta;\Gamma \vdash D \dashv \Delta';\Gamma'$. We'll set it up so that $\Delta' = \Delta, D$ regardless of D, and it should typecheck bodies of controls. The last line (the instantiation) is the tricky part.

Rules for typechecking instantiations.

INST-CONTROL

$$\frac{\mathsf{lookup}_\Delta(\tau) = \mathsf{control}\ T(\overline{x:\tau\ p})(x_1:\tau_1,\ldots,x_n:\tau_n)\ \{\overline{D}\ \mathsf{apply}\ B\} \qquad \Delta;\Gamma \vdash e_i : \tau_i \text{ for each } i}{\Delta;\Gamma \vdash \tau(e_1,\ldots,e_n) : \mathsf{control}\ (\overline{x:\tau p})}$$

INST-PACKAGE

$$\frac{\mathsf{lookup}_\Delta(\tau) = \mathsf{package}\ T(x_1:\tau_1,\ldots,x_n:\tau_n) \qquad \Delta;\Gamma \vdash e_i : \tau_i \text{ for each } i}{\Delta;\Gamma \vdash \tau(e_1,\ldots,e_n) : \mathsf{package}}$$

This isn't enough to typecheck the example. The type of the constructor argument to P is A, not control $()$. This can be dealt with by adding a rule for "resolving" type names, but I think another option would be to inline types. Maybe the way we do it in the formalization should be different from how we do it in the implementation.

The rules do not handle generics.

RESOLVE-CONTROL-TYPE

$$\frac{\Delta;\Gamma \vdash e : \mathsf{control}\ \langle\overline{T}\rangle(\overline{x:\tau d}) \qquad \mathsf{lookup}_\Delta(T_0) = \mathsf{control}\ T_0\langle\overline{T}\rangle(\overline{x:\tau d})}{\Delta;\Gamma \vdash e : T_0}$$