1 How do we check entailments?

1. Create set of "axioms" based on the program and the entailment to check
   (a) Datatype Declarations: Model everything as an enumeration type (Variables, Classes, Fields, Paths)
      - Extract field and class names from the program (constructor declarations).
      - Extract variable names from the entailment to be checked.
      - Enumerate all paths that do not exceed the depth limit based on the extracted information.
   (b) Function Declarations
      - Declare boolean predicates for the constraints: path-equivalence, instance-of, instantiated-by
      - For substitution we previously had

        $$\texttt{Path} \times \texttt{Variable} \times \texttt{Path} \to \texttt{Path}$$

        as the substitution function signature. This is no longer possible. The enumeration of paths with a depth limit would result in a partial function, as we would no longer be able to compute result paths for substitutions that would exceed the depth limit. Since the SMT solver only supports total functions, we form a relation

        $$\texttt{Path} \times \texttt{Variable} \times \texttt{Path} \times \texttt{Path} \to \mathcal{B}$$

        that only is true for each $\texttt{Path} \times \texttt{Variable} \times \texttt{Path}$ triple where the partial function would be defined at.
   (c) Calculus rules
      - Static Rules: C-Refl, C-Class, C-Subst
      - Dynamic Rules: C-Prog
        The creation of C-Prog rules is handled context-sensitive to the program. We transforme each constraint entailment declaration

        $$\forall x.\ c_1, c_2, ..., c_n \Rightarrow x :: cls$$

        from the program into a C-Prog rule template of the form

        $$p : \texttt{Path} \Rightarrow c_{1\,x \mapsto p} \wedge c_{2\,x \mapsto p} \wedge ... \wedge c_{n\,x \mapsto p} \to p :: cls$$

        and instantiate the template with all available paths. Instantiations where one of the paths post substitution exceed the depth limit are discarded.
   (d) Assert entailment to be checked: $c_1, ..., c_n \vdash c \Rightarrow \neg(c_1 \wedge ... \wedge c_n \to c)$
2. Obtain Solution: Does the entailment contradict the rules?
   - If unsat: there is a contradiction between the negated entailment and the calculus rules, thus the entailment holds.
   - If sat: there is no contradiction between the negated entailment and the calculus rules, thus the entailment does not hold.

2 Simplified example encoding

To encode: $y \equiv x \vdash x \equiv y$

$$\texttt{Variable} := \{\texttt{x}, \texttt{y}\} \tag{1}$$

$$p_{v \mapsto q} = s := \tag{2}$$

$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{x} \wedge s = \texttt{x}) \vee \tag{3}$$

$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{y} \wedge s = \texttt{y}) \vee \tag{4}$$

$$(p = \texttt{x} \wedge v = \texttt{y} \wedge q = \texttt{x} \wedge s = \texttt{x}) \vee \tag{5}$$

$$(p = \texttt{x} \wedge v = \texttt{y} \wedge q = \texttt{y} \wedge s = \texttt{x}) \vee \tag{6}$$

$$(p = \texttt{y} \wedge v = \texttt{x} \wedge q = \texttt{x} \wedge s = \texttt{y}) \vee \tag{7}$$

$$(p = \texttt{y} \wedge v = \texttt{x} \wedge q = \texttt{y} \wedge s = \texttt{y}) \vee \tag{8}$$

$$(p = \texttt{y} \wedge v = \texttt{y} \wedge q = \texttt{x} \wedge s = \texttt{x}) \vee \tag{9}$$

$$(p = \texttt{y} \wedge v = \texttt{y} \wedge q = \texttt{y} \wedge s = \texttt{y}) \tag{10}$$

$$\forall p.\ p \equiv p \tag{11}$$

$$\forall p, q, v, r, s, a, b, c, d. \tag{12}$$

$$s \equiv r \wedge p_{v \mapsto r} = a \wedge q_{v \mapsto r} = b \wedge \tag{13}$$

$$a \equiv b \wedge p_{v \mapsto s} = c \wedge q_{v \mapsto s} = d \tag{14}$$

$$\rightarrow c \equiv d \tag{15}$$

$$\neg(\texttt{y} \equiv \texttt{x} \rightarrow \texttt{x} \equiv \texttt{y}) \tag{16}$$

3 Complete example encoding

To encode: $x.\mathbf{cls} \equiv \texttt{Succ}, x.p.\mathbf{cls} \equiv \texttt{Zero}, x \equiv y \vdash y :: \texttt{Nat}$ with *depth–limit* $= 1$

$$\texttt{Variable} := \{\texttt{x}, \texttt{y}\} \tag{17}$$
$$\texttt{Class} := \{\texttt{Zero}, \texttt{Succ}, \texttt{Nat}\} \tag{18}$$
$$\texttt{Path} := \{\texttt{x}, \texttt{x.p}, \texttt{y}, \texttt{y.p}\} \tag{19}$$
$$p_{v \mapsto q} = s := \tag{20}$$
$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{x} \wedge s = \texttt{x}) \vee \tag{21}$$
$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{x.p} \wedge s = \texttt{x.p}) \vee \tag{22}$$
$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{y} \wedge s = \texttt{y}) \vee \tag{23}$$
$$(p = \texttt{x} \wedge v = \texttt{x} \wedge q = \texttt{y.p} \wedge s = \texttt{y.p}) \vee \tag{24}$$
$$(p = \texttt{x.p} \wedge v = \texttt{x} \wedge q = \texttt{x} \wedge s = \texttt{x.p}) \vee \tag{25}$$
$$(p = \texttt{x.p} \wedge v = \texttt{x} \wedge q = \texttt{y} \wedge s = \texttt{y.p}) \vee \tag{26}$$
$$\ldots \tag{27}$$
$$\forall p.\ p \equiv p \tag{28}$$
$$\forall p, c.\ p.\mathbf{cls} \equiv c \rightarrow p :: c \tag{29}$$
$$\forall p, q, v, r, s, a, b, c, d. \tag{30}$$
$$s \equiv r \wedge p_{v \mapsto r} = a \wedge q_{v \mapsto r} = b \wedge \tag{31}$$
$$a \equiv b \wedge p_{v \mapsto s} = c \wedge q_{v \mapsto s} = d \tag{32}$$
$$\rightarrow c \equiv d \tag{33}$$
$$\forall p, c, v, r, s, a, b. \tag{34}$$
$$s \equiv r \wedge p_{v \mapsto r} = a \wedge \tag{35}$$
$$a :: c \wedge p_{v \mapsto s} = b \tag{36}$$
$$\rightarrow b :: c \tag{37}$$
$$\forall p, c, v, r, s, a, b. \tag{38}$$
$$s \equiv r \wedge p_{v \mapsto r} = a \wedge \tag{39}$$
$$a.\mathbf{cls} \equiv c \wedge p_{v \mapsto s} = b \tag{40}$$
$$\rightarrow b.\mathbf{cls} \equiv c \tag{41}$$
$$\texttt{x} :: \texttt{Zero} \rightarrow \texttt{x} :: \texttt{Nat} \tag{42}$$
$$\texttt{x.p} :: \texttt{Zero} \rightarrow \texttt{x.p} :: \texttt{Nat} \tag{43}$$
$$\texttt{x} :: \texttt{Succ} \wedge \texttt{x.p} :: \texttt{Nat} \rightarrow \texttt{x} :: \texttt{Nat} \tag{44}$$
$$\neg(\texttt{x}.\mathbf{cls} \equiv \texttt{Succ} \wedge \texttt{x.p}.\mathbf{cls} \equiv \texttt{Zero} \wedge \texttt{x} \equiv \texttt{y} \rightarrow \texttt{y} :: \texttt{Nat}) \tag{45}$$

4 Is using a depth limit reasonable?

$$\frac{}{\epsilon \vdash p \equiv p}\ \text{(C-Refl)} \qquad \frac{}{a \vdash a}\ \text{(C-Ident)} \qquad \frac{\overline{a} \vdash p.\mathbf{cls} \equiv C}{\overline{a} \vdash p :: C}\ \text{(C-Class)}$$

$$\frac{\overline{a} \vdash a_{\{\!\{x \mapsto p\}\!\}} \qquad \overline{a} \vdash p' \equiv p}{\overline{a} \vdash a_{\{\!\{x \mapsto p'\}\!\}}}\ \text{(C-Subst)} \qquad \frac{(\forall x.\ \overline{a} \Rightarrow a) \in P \qquad \overline{b} \vdash \overline{a}_{\{\!\{x \mapsto p\}\!\}}}{\overline{b} \vdash a_{\{\!\{x \mapsto p\}\!\}}}\ \text{(C-Prog)}$$

4.1 Where can we use substitution?

$$\frac{\overline{a} \vdash a_{\{\!\{x \mapsto p\}\!\}} \qquad \overline{a} \vdash p' \equiv p}{\overline{a} \vdash a_{\{\!\{x \mapsto p'\}\!\}}}\ \text{(C-Subst)}$$

We can substitute an equivalent path in the conclusion of an entailment. This substitution might exceed the depth limit, but we might only alter the conclusion and not the context of the entailment. Also the prooftree-lowest conclusion is already the result of a substitution.

4.2 How can we close branches?

$$\frac{}{\epsilon \vdash p \equiv p}\ \text{(C-Refl)} \qquad \frac{}{a \vdash a}\ \text{(C-Ident)}$$

C-Refl Reflexive paths, independent of context.

C-Ident Constraints that are already in the context.

4.3 Observation

$$\frac{\dfrac{\dfrac{\ldots}{x :: \texttt{Zero}, x \equiv y \vdash x :: \texttt{Zero}}}{x :: \texttt{Zero}, x \equiv y \vdash y :: \texttt{Zero}_{y \mapsto x}} \qquad \dfrac{\ldots}{x :: \texttt{Zero}, x \equiv y \vdash x \equiv y}}{x :: \texttt{Zero}, x \equiv y \vdash y :: \texttt{Zero}}$$

To be able to close a branch in practice, we wan't to come back to a form that is already in the context. We wan't to substitute with path equivalences that the context already provides. The variable we want to substitute is the base of the path in the conclusion constraint.

4.4 Whats with program entailment instantiations?

$$\frac{(\forall x.\ \overline{a} \Rightarrow a) \in P \qquad \overline{b} \vdash \overline{a}_{\{\!\{x \mapsto p\}\!\}}}{\overline{b} \vdash a_{\{\!\{x \mapsto p\}\!\}}}\ \text{(C-Prog)}$$

Usages of rule C-Prog replace the conclusion of an entailment with new proof goals. It does so by instantiating a declaration from the program.

4.5 Hypothesis

The depth of paths used in proofs in the sequent calculus only grows within a certain bound.