1 How do we check entailments?

1. Create set of "axioms" based on the program and the entailment to check
   (a) Datatype Declarations
      - Enumeration types: Variables, Classes, Fields
      - ADT: Path
   (b) Function Declarations
      - Declared: path-equivalence, instance-of, instantiated-by
      - Defined recursively: substitute
   (c) Calculus rules as all quantified formulas
      - Static Rules: C-Refl, C-Class, C-Subst
      - Dynamic Rules: C-Prog
   (d) Assert entailment to be checked: $c_1, ..., c_n \vdash c \Rightarrow \neg(c_1 \wedge ... \wedge c_n \rightarrow c)$
2. Get Solution
   - If unsat: valid/correct entailment.
   - If sat: invalid entailment.

1.1 General First-Order Encoding Template

$$
\begin{align}
&\texttt{Variable (...)} \tag{1}\\
&\texttt{Field (...)} \tag{2}\\
&\texttt{Class (...)} \tag{3}\\
&\texttt{Path}\ ((\texttt{Variable})\ (\texttt{Path Field})) \tag{4}\\
&\equiv : \texttt{Path} \times \texttt{Path} \rightarrow \mathcal{B} \tag{5}\\
&:: : \texttt{Path} \times \texttt{Class} \rightarrow \mathcal{B} \tag{6}\\
&\textbf{.cls} \equiv : \texttt{Path} \times \texttt{Class} \rightarrow \mathcal{B} \tag{7}\\
&\_{\{\!\!\{\mapsto\}\!\!\}} : \texttt{Path} \times \texttt{Variable} \times \texttt{Path} \rightarrow \texttt{Path} \tag{8}\\
&\forall p{:}\texttt{Path}.\ p \equiv p \tag{9}\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}, c{:}\texttt{Class}. \tag{10}\\
&\quad (a \rightarrow p\textbf{.cls} \equiv c) \rightarrow (a \rightarrow p :: c) \tag{11}\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}, q{:}\texttt{Path}, r{:}\texttt{Path}, s{:}\texttt{Path}, x{:}\texttt{Variable}. \tag{12}\\
&\quad (a \rightarrow p_{\{\!\!\{x \mapsto r\}\!\!\}} \equiv q_{\{\!\!\{x \mapsto r\}\!\!\}} \wedge (a \rightarrow s \equiv r)) \rightarrow \tag{13}\\
&\qquad (a \rightarrow p_{\{\!\!\{x \mapsto s\}\!\!\}} \equiv q_{\{\!\!\{x \mapsto s\}\!\!\}}) \tag{14}\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}, c{:}\texttt{Class}, r{:}\texttt{Path}, s{:}\texttt{Path}, x{:}\texttt{Variable}. \tag{15}\\
&\quad (a \rightarrow p_{\{\!\!\{x \mapsto r\}\!\!\}} :: c \wedge (a \rightarrow s \equiv r)) \rightarrow \tag{16}\\
&\qquad (a \rightarrow p_{\{\!\!\{x \mapsto s\}\!\!\}} :: c) \tag{17}\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}, c{:}\texttt{Class}, r{:}\texttt{Path}, s{:}\texttt{Path}, x{:}\texttt{Variable}. \tag{18}\\
&\quad (a \rightarrow p_{\{\!\!\{x \mapsto r\}\!\!\}}\textbf{.cls} \equiv c \wedge (a \rightarrow s \equiv r)) \rightarrow \tag{19}\\
&\qquad (a \rightarrow p_{\{\!\!\{x \mapsto s\}\!\!\}}\textbf{.cls} \equiv c) \tag{20}\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}.\ (a \rightarrow \bigwedge \_) \rightarrow (a \rightarrow p :: \_) \tag{21}
\end{align}
$$

1.2 Natural Numbers Program

$$\texttt{Zero}(x.\ \epsilon) \tag{22}$$
$$\texttt{Succ}(x.\ x.p :: \texttt{Nat}) \tag{23}$$
$$\forall x.\ x :: \texttt{Zero} \Rightarrow x :: \texttt{Nat} \tag{24}$$
$$\forall x.\ x :: \texttt{Succ}, x.p :: \texttt{Nat} \Rightarrow x :: \texttt{Nat} \tag{25}$$
$$\texttt{prev}(x.\ x :: \texttt{Nat}) : [y.\ y :: \texttt{Nat}] \tag{26}$$
$$\texttt{prev}(x.\ x :: \texttt{Zero}) : [y.\ y :: \texttt{Nat}] := \textbf{new}\ \texttt{Zero}() \tag{27}$$
$$\texttt{prev}(x.\ x :: \texttt{Succ}, x.p :: \texttt{Nat}) : [y.\ y :: \texttt{Nat}] := x.p \tag{28}$$

1.3 C-Prog Rules for Natural Numbers Program

$$\forall a : \mathcal{B}, p : \texttt{Path}. \tag{29}$$
$$(a \rightarrow p :: \texttt{Zero}) \rightarrow \tag{30}$$
$$(a \rightarrow p :: \texttt{Nat}) \tag{31}$$
$$\forall a : \mathcal{B}, p : \texttt{Path}. \tag{32}$$
$$(a \rightarrow p :: \texttt{Succ} \wedge x.p_{\{\!\{x \mapsto p\}\!\}} :: \texttt{Nat}) \rightarrow \tag{33}$$
$$(a \rightarrow p :: \texttt{Nat}) \tag{34}$$

2 Example Entailments

2.1 Working valid entailment

$p.\mathbf{cls} \equiv \texttt{Zero} \vdash p :: \texttt{Nat}$

- Checks unsatisfiable
- Unsat Core: C-Class, C-Prog-Zero

2.2 Working invalid entailment

TODO: search typechecking tests for (non-trivial) SAT entailment
$\vdash x \equiv y$

- Checks satisfiable
- Model: $p \equiv q \doteq p = q$

2.3 Non-working invalid entailment

$x \equiv y \vdash x \equiv z$

- Checks unsatisfiable, but shouldn't
- Unsat Core: C-Subst-PathEq
- TODO: possible reason? (check instantiation of subst rule)

2.4 Non-working invalid entailment

fieldAccessTimeout.smt

- Neither checks satisfiable nor unsatisfiable.
- Solver has “infinite” runtime.
- TODO: check manually for a model?

3 Undefinedness

3.1 adding $\neg x \equiv x.p$ results in ???

3.2 asserting $x :: \texttt{Zero} \wedge x :: \texttt{Nat}$ is SAT

But with further investigation, again doesn’t seem to be a problem, as we are searching conflicts