# 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
      - Template Rules: C-Prog
   (d) Assert entailment to be checked: $c_1, ..., c_n \vdash c \Rightarrow \neg(c_1 \land ... \land c_n \to c)$
2. Obtain Solution: Does the entailment contradict the rules?
   - If unsat: valid/correct entailment.
   - If sat: invalid entailment.

## 1.1 General First-Order Encoding Template

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

### 1.2 Natural Numbers Program

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

### 1.3 C-Prog Rules for Natural Numbers Program

$$\begin{aligned}
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}. && (29)\\
&\quad (a \to p :: \texttt{Zero}) \to && (30)\\
&\qquad (a \to p :: \texttt{Nat}) && (31)\\
&\forall a{:}\mathcal{B}, p{:}\texttt{Path}. && (32)\\
&\quad (a \to p :: \texttt{Succ} \land x.p_{\{\!\{x \mapsto p\}\!\}} :: \texttt{Nat}) \to && (33)\\
&\qquad (a \to p :: \texttt{Nat}) && (34)
\end{aligned}$$

## 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

$\vdash x \equiv y$

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

$a \equiv b \vdash a \equiv c$
Checks satisfiable

Model:

$$
\begin{array}{l}
helper(q\,{:}\,\texttt{Path}) := \\
\quad q = a \;?\; a \;\;: \\
\qquad q = c \;?\; c \;\;: \\
\qquad\quad q = x \;?\; x \;\;: \\
\qquad\qquad q = b \;?\; b \;\;: \\
\qquad\qquad\quad q = b.p \;?\; b.p \;\;: \\
\qquad\qquad\qquad q = a.b \;?\; a.b \;\;:\;\; c.p \\
p\,{:}\,\texttt{Path} \equiv q\,{:}\,\texttt{Path} := \\
\quad \textbf{let } a_1 := helper(p) = a.p \wedge helper(q) = b.p \\
\qquad\quad a_2 := helper(p) = a.p \wedge helper(q) = a.p \\
\qquad\quad a_3 := helper(p) = c.p \wedge helper(q) = c.p \\
\qquad\quad a_4 := helper(p) = b.p \wedge helper(q) = b.p \\
\qquad\quad a_5 := helper(p) = b.p \wedge helper(q) = a.p \\
\quad \textbf{in } \displaystyle\bigvee_{i \in \{j \mid 1 \leq j \leq 5\}} a_i \\
\quad \vee\, helper(p) = a \wedge helper(q) = a \\
\quad \vee\, helper(p) = b \wedge helper(q) = b \\
\quad \vee\, helper(p) = b \wedge helper(q) = a \\
\quad \vee\, helper(p) = c \wedge helper(q) = c \\
\quad \vee\, helper(p) = x \wedge helper(q) = x \\
\quad \vee\, helper(p) = a \wedge helper(q) = b
\end{array}
$$

### 2.3 Non-working invalid entailment

$x :: \texttt{Succ}, x.p :: \texttt{Zero} \vdash x :: \texttt{Zero}$

- Location: /paper/dep-classes/smt/semantic_entailment/fieldAccessTimeout.smt
- Solver has “infinite” runtime.
- Wanted behavior: Checks satisfiable.
- Current Solution: Impose a timeout on the solver.
  - Assume an entailment to be invalid if the solver times out (to retain soundness).
  - Procedure isn’t complete anymore.
  - Balance the timeout between the amount of checkable valid entailments and tolerable waiting time. (Set it such that the majority of valid entailments can still be checked.)

# 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

# 4 Faithful encoding

# 5 Change entailment translation

Instead $\bigwedge c_i \to \mathit{conclusion}$, use multiple asserts. for $c_i$ in *context*. assert $c_i$ assert $\neg \mathit{conclusion}$

# 6 Explicitly define constraint predicates

## 6.1 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)