

# 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:  $c_1, \dots, c_n \vdash c \Rightarrow \neg(c_1 \wedge \dots \wedge c_n \rightarrow c)$
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

$$\text{Variable } (\dots) \quad (1)$$

$$\text{Field } (\dots) \quad (2)$$

$$\text{Class } (\dots) \quad (3)$$

$$\text{Path } ((\text{Variable}) (\text{Path Field})) \quad (4)$$

$$\equiv : \text{Path} \times \text{Path} \rightarrow \mathcal{B} \quad (5)$$

$$:: : \text{Path} \times \text{Class} \rightarrow \mathcal{B} \quad (6)$$

$$\text{.cls} \equiv : \text{Path} \times \text{Class} \rightarrow \mathcal{B} \quad (7)$$

$$\_ \{\!\!\{ \mapsto \}\!\!\} : \text{Path} \times \text{Variable} \times \text{Path} \rightarrow \text{Path} \quad (8)$$

$$\forall p : \text{Path}. p \equiv p \quad (9)$$

$$\forall a : \mathcal{B}, p : \text{Path}, c : \text{Class}. \quad (10)$$

$$(a \rightarrow p.\text{cls} \equiv c) \rightarrow (a \rightarrow p :: c) \quad (11)$$

$$\forall a : \mathcal{B}, p : \text{Path}, q : \text{Path}, r : \text{Path}, s : \text{Path}, x : \text{Variable}. \quad (12)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto r \}\!\!\} \equiv q \{\!\!\{ x \mapsto r \}\!\!\} \wedge (a \rightarrow s \equiv r)) \rightarrow \quad (13)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto s \}\!\!\} \equiv q \{\!\!\{ x \mapsto s \}\!\!\}) \quad (14)$$

$$\forall a : \mathcal{B}, p : \text{Path}, c : \text{Class}, r : \text{Path}, s : \text{Path}, x : \text{Variable}. \quad (15)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto r \}\!\!\} :: c \wedge (a \rightarrow s \equiv r)) \rightarrow \quad (16)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto s \}\!\!\} :: c) \quad (17)$$

$$\forall a : \mathcal{B}, p : \text{Path}, c : \text{Class}, r : \text{Path}, s : \text{Path}, x : \text{Variable}. \quad (18)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto r \}\!\!\}.\text{cls} \equiv c \wedge (a \rightarrow s \equiv r)) \rightarrow \quad (19)$$

$$(a \rightarrow p \{\!\!\{ x \mapsto s \}\!\!\}.\text{cls} \equiv c) \quad (20)$$

$$\forall a : \mathcal{B}, p : \text{Path}. (a \rightarrow \bigwedge \_) \rightarrow (a \rightarrow p :: \_) \quad (21)$$

## 1.2 Natural Numbers Program

$$\text{Zero}(x. \epsilon) \quad (22)$$

$$\text{Succ}(x. x.p :: \text{Nat}) \quad (23)$$

$$\forall x. x :: \text{Zero} \Rightarrow x :: \text{Nat} \quad (24)$$

$$\forall x. x :: \text{Succ}, x.p :: \text{Nat} \Rightarrow x :: \text{Nat} \quad (25)$$

$$\text{prev}(x. x :: \text{Nat}) : [y. y :: \text{Nat}] \quad (26)$$

$$\text{prev}(x. x :: \text{Zero}) : [y. y :: \text{Nat}] := \text{new Zero}() \quad (27)$$

$$\text{prev}(x. x :: \text{Succ}, x.p :: \text{Nat}) : [y. y :: \text{Nat}] := x.p \quad (28)$$

## 1.3 C-Prog Rules for Natural Numbers Program

$$\forall a:\mathcal{B}, p:\text{Path}. \quad (29)$$

$$(a \rightarrow p :: \text{Zero}) \rightarrow \quad (30)$$

$$(a \rightarrow p :: \text{Nat}) \quad (31)$$

$$\forall a:\mathcal{B}, p:\text{Path}. \quad (32)$$

$$(a \rightarrow p :: \text{Succ} \wedge x.p\llbracket x \mapsto p \rrbracket :: \text{Nat}) \rightarrow \quad (33)$$

$$(a \rightarrow p :: \text{Nat}) \quad (34)$$

## 2 Example Entailments

### 2.1 Working valid entailment

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

- 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 $x :: \text{Zero} \wedge x :: \text{Nat}$ is SAT

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