# 1 Path Depth Limit 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* $= 1$

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

$$\texttt{Class} := \{\texttt{Zero}, \texttt{Succ}, \texttt{Nat}\} \tag{2}$$

$$\texttt{Path} := \{\texttt{x}, \texttt{x.p}, \texttt{y}, \texttt{y.p}\} \tag{3}$$

$$p_{v \mapsto q} = s := \tag{4}$$

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

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

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

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

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

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

$$\ldots \tag{11}$$

$$\forall p.\ p \equiv p \quad \text{(C-Refl)} \tag{12}$$

$$\forall p, c.\ p.\mathbf{cls} \equiv c \rightarrow p :: c \quad \text{(C-Class)} \tag{13}$$

$$\forall p, q, v, r, s, a, b, c, d. \quad \text{(C-Subst)} \tag{14}$$

$$s \equiv r \wedge p_{v \mapsto r} = a \wedge q_{v \mapsto r} = b \wedge \tag{15}$$

$$a \equiv b \wedge p_{v \mapsto s} = c \wedge q_{v \mapsto s} = d \tag{16}$$

$$\rightarrow c \equiv d \tag{17}$$

$$\forall p, c, v, r, s, a, b. \quad \text{(C-Subst)} \tag{18}$$

$$s \equiv r \wedge p_{v \mapsto r} = a \wedge \tag{19}$$

$$a :: c \wedge p_{v \mapsto s} = b \tag{20}$$

$$\rightarrow b :: c \tag{21}$$

$$\forall p, c, v, r, s, a, b. \quad \text{(C-Subst)} \tag{22}$$

$$s \equiv r \wedge p_{v \mapsto r} = a \wedge \tag{23}$$

$$a.\mathbf{cls} \equiv c \wedge p_{v \mapsto s} = b \tag{24}$$

$$\rightarrow b.\mathbf{cls} \equiv c \tag{25}$$

$$\texttt{x} :: \texttt{Zero} \rightarrow \texttt{x} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{26}$$

$$\texttt{x.p} :: \texttt{Zero} \rightarrow \texttt{x.p} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{27}$$

$$\texttt{x} :: \texttt{Succ} \wedge \texttt{x.p} :: \texttt{Nat} \rightarrow \texttt{x} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{28}$$

$$\texttt{y} :: \texttt{Zero} \rightarrow \texttt{y} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{29}$$

$$\texttt{y.p} :: \texttt{Zero} \rightarrow \texttt{y.p} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{30}$$

$$\texttt{y} :: \texttt{Succ} \wedge \texttt{y.p} :: \texttt{Nat} \rightarrow \texttt{y} :: \texttt{Nat} \quad \text{(C-Prog)} \tag{31}$$

$$\neg(\texttt{x}.\mathbf{cls} \equiv \texttt{Succ} \wedge \texttt{x.p}.\mathbf{cls} \equiv \texttt{Zero} \wedge \texttt{x} \equiv \texttt{y} \rightarrow \texttt{y} :: \texttt{Nat}) \tag{32}$$

## 2 Ground Encoding

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

| | | |
|---|---|---|
| $\texttt{Variable} := \{\texttt{x}, \texttt{y}\}$ | | (1) |
| $\texttt{Class} := \{\texttt{Zero}, \texttt{Succ}, \texttt{Nat}\}$ | | (2) |
| $\texttt{Path} := \{\texttt{x}, \texttt{x.p}, \texttt{y}, \texttt{y.p}\}$ | | (3) |
| $\texttt{x} \equiv \texttt{x} \wedge \texttt{x.p} \equiv \texttt{x.p} \wedge \texttt{y} \equiv \texttt{y} \wedge \texttt{y.p} \equiv \texttt{y.p}$ | (C-Refl) | (4) |
| $\texttt{x}.\mathbf{cls} \equiv \texttt{Zero} \rightarrow \texttt{x} :: \texttt{Zero}$ | (C-Class) | (5) |
| $\texttt{x.p}.\mathbf{cls} \equiv \texttt{Zero} \rightarrow \texttt{x.p} :: \texttt{Zero}$ | (C-Class) | (6) |
| ... | (C-Class) | (7) |
| $\texttt{x} \equiv \texttt{y} \wedge \texttt{y} \equiv \texttt{y} \rightarrow \texttt{y} \equiv \texttt{x}$ | (C-Subst) | (8) |
| $\texttt{x} \equiv \texttt{y} \wedge \texttt{y} :: \mathit{Nat} \rightarrow \texttt{x} :: \texttt{Nat}$ | (C-Subst) | (9) |
| $\texttt{x} \equiv \texttt{y} \wedge \texttt{y}.\mathbf{cls} \equiv \mathit{Nat} \rightarrow \texttt{x}.\mathbf{cls} \equiv \texttt{Nat}$ | (C-Subst) | (10) |
| ... | (C-Subst) | (11) |
| $\texttt{x} :: \texttt{Zero} \rightarrow \texttt{x} :: \texttt{Nat}$ | (C-Prog) | (12) |
| $\texttt{x.p} :: \texttt{Zero} \rightarrow \texttt{x.p} :: \texttt{Nat}$ | (C-Prog) | (13) |
| $\texttt{x} :: \texttt{Succ} \wedge \texttt{x.p} :: \texttt{Nat} \rightarrow \texttt{x} :: \texttt{Nat}$ | (C-Prog) | (14) |
| $\texttt{y} :: \texttt{Zero} \rightarrow \texttt{y} :: \texttt{Nat}$ | (C-Prog) | (15) |
| $\texttt{y.p} :: \texttt{Zero} \rightarrow \texttt{y.p} :: \texttt{Nat}$ | (C-Prog) | (16) |
| $\texttt{y} :: \texttt{Succ} \wedge \texttt{y.p} :: \texttt{Nat} \rightarrow \texttt{y} :: \texttt{Nat}$ | (C-Prog) | (17) |
| ... | | (18) |
| $\neg(\texttt{x}.\mathbf{cls} \equiv \texttt{Succ} \wedge \texttt{x.p}.\mathbf{cls} \equiv \texttt{Zero} \wedge \texttt{x} \equiv \texttt{y} \rightarrow y :: \texttt{Nat})$ | | (19) |

### 2.1 Substitution in the Ground Encoding

We want to finitely enumerate the quantified rule:

$$\begin{aligned}&\forall p, c, v, r, s, a, b. && \text{(C-Subst)}\\ &\quad s \equiv r \wedge p_{v \mapsto r} = a \wedge\\ &\quad a :: c \wedge p_{v \mapsto s} = b\\ &\qquad \rightarrow b :: c\end{aligned}$$

The naïve approach would be to take the cross product of all quantified variables. This would leave us with a lot of meaningless implications, e.g. if we instantiate the rule with $p = \mathtt{x}, v = \mathtt{y}, r = \mathtt{x}, a = \mathtt{y}, s = \mathtt{x}, b = \mathtt{x}, c = \mathtt{Nat}$

$$\begin{aligned}&\mathtt{x} \equiv \mathtt{x} \wedge \mathtt{x}_{\mathtt{y} \mapsto \mathtt{x}} = \mathtt{y} \ \wedge \mathtt{y} :: \mathtt{Nat} \wedge \mathtt{x}_{\mathtt{y} \mapsto \mathtt{x}} = \mathtt{x}\\ &\quad \rightarrow \mathtt{x} :: \mathtt{Nat}\end{aligned}$$

Since we know the substitution to be false, we do not have to include this instantiation into the encoding. Since we only need to include rule instantiations where the substitution predicate holds and we can calculate the substitution prior since all quantified variables are known, we can get rid of the substitution predicate in the encoding altogether.

E.g. the inatantiation with $p = \mathtt{x}, v = \mathtt{x}, r = \mathtt{y}, a = \mathtt{y}, s = \mathtt{x}, b = \mathtt{x}, c = \mathtt{Nat}$

$$\begin{aligned}&\mathtt{x} \equiv \mathtt{y} \wedge \mathtt{x}_{\mathtt{x} \mapsto \mathtt{y}} = \mathtt{y} \ \wedge \mathtt{y} :: \mathtt{Nat} \wedge \mathtt{x}_{\mathtt{x} \mapsto \mathtt{x}} = \mathtt{x}\\ &\quad \rightarrow \mathtt{x} :: \mathtt{Nat}\end{aligned}$$

turns into

$$\mathtt{x} \equiv \mathtt{y} \wedge \mathtt{y} :: \mathit{Nat} \rightarrow \mathtt{x} :: \mathtt{Nat}$$

# 3 Algorithmic Symmetry

We rely on the equivalency between the declarative- and the algorithmic system to set our depth limit for path enumeration as well as on the decidability of the declarative system to even have such a limit in place.

The entailment $a \equiv b \vdash b \equiv a$ is a counterexample to Lemma 2 (more precisely 1) and Theorem 1 as it relies on Lemma 5.5.16.

**Lemma 1** (5.5.15)**.** *If* $\mathsf{wf} P$ *and* $\overline{a} \vdash a$ *then* $\overline{a} \vdash_A a$*.*

**Lemma 2** (5.5.16)**.** *If* $\mathsf{wf} P$ *then* $\overline{a} \vdash a$ *iff* $\overline{a} \vdash_A a$*.*

**Theorem 1** (5.5.1)**.** *If* $\mathsf{wf} P$ *then derivation of* $\overline{a} \vdash a$ *is decidable.*

Counterexample for Lemma 1. Choose any well-formed program.

$$\dfrac{\dfrac{\dfrac{}{\cdot \vdash b \equiv b}\,\text{C-Refl}}{a \equiv b \vdash b \equiv a_{\{\!\{a \mapsto b\}\!\}}}\,\text{C-Weak} \qquad \dfrac{}{a \equiv b \vdash a \equiv b}\,\text{C-Ident}}{a \equiv b \vdash b \equiv a_{\{\!\{a \mapsto a\}\!\}}}\,\text{C-Subst}$$

$$\dfrac{\dfrac{}{a \equiv b \vdash_A a \equiv a}\ \text{CA-Refl} \qquad a \sqsubset a \equiv b \qquad \dfrac{\dots}{a \equiv b \vdash_A b \equiv a}}{a \equiv b \vdash_A b \equiv a}\ \text{CA-Subst3}$$

$$\dfrac{\dfrac{\dots}{a \equiv b \vdash_A b \equiv a} \qquad b \sqsubset a \equiv b \qquad \dfrac{}{a \equiv b \vdash_A b \equiv b}\ \text{CA-Refl}}{a \equiv b \vdash_A b \equiv a}\ \text{CA-Subst3}$$

## 3.1 Symmetry Fix

We can update rule CA-Subst3 to allow the entailment used as a counterexample to Lemma 1 to have a derivation.

There are two feasible ways to update the rule:

1. $$\dfrac{\overline{a} \vdash_A p \equiv p'' \qquad p \sqsubset \overline{a} \qquad \overline{a} \vdash_A p \equiv p'}{\overline{a} \vdash_A p' \equiv p''}\ \text{CA-Subst3Fix1}$$

2. $$\dfrac{\overline{a} \vdash_A p'' \equiv p \qquad p \sqsubset \overline{a} \qquad \overline{a} \vdash_A p' \equiv p}{\overline{a} \vdash_A p' \equiv p''}\ \text{CA-Subst3Fix2}$$

### 3.1.1 Fix 1 derivation

$$\dfrac{\dfrac{}{a \equiv b \vdash_A a \equiv a}\ \text{CA-Refl} \qquad a \sqsubset a \equiv b \qquad \dfrac{}{a \equiv b \vdash_A a \equiv b}\ \text{CA-Ident}}{a \equiv b \vdash_A b \equiv a}\ \text{CA-Subst3Fix1}$$

### 3.1.2 Fix 2 derivation

$$\dfrac{\dfrac{}{a \equiv b \vdash_A a \equiv b}\ \text{CA-Ident} \qquad b \sqsubset a \equiv b \qquad \dfrac{}{a \equiv b \vdash_A b \equiv b}\ \text{CA-Refl}}{a \equiv b \vdash_A b \equiv a}\ \text{CA-Subst3Fix2}$$

## 3.2 What to do with this?

- Update rule CA-Subst3 with one of the solutions from subsection 3.1.
- Redo the proofs?