Syntactic Theory of Finitary Sets

Denis Müller

Saarland University

10. Dezember 2014

Introduction

Hereditarily finitary sets consist of finitely many hereditarily finitary sets. Well-founded sets do not have cycles and can hence be represented by binary trees.
Semantically, this is how we can map finitary sets in our representation to ordinary sets:

- $[\![\emptyset]\!] = \emptyset$
- $[\![s.t]\!] = \{[\![s]\!]\} \cup [\![t]\!]$

Functions on Trees

List of all elements of a tree:
L $\emptyset$ = []
L (s . t) = s :: L t

Append: @
$\emptyset$ @ t = t
$(s.s')$ @ t = s . s' @ t

Table of Contents

Properties of Equivalence Relations

Important properties that should be satisfied:

- **Del** : s.s.t ≡ s.t
- **Swap** : s.t.u ≡ t.s.u

Table of Contents

We inductively define a first equivalence relation $\equiv$ satisfying the previous conditions

$$\frac{}{\emptyset \equiv \emptyset}$$

$$\frac{}{s.s.t \equiv s.t}$$

$$\frac{}{s.t.u \equiv t.s.u}$$

$$\frac{s \equiv t \qquad s \neq \emptyset \neq t}{t \equiv s}$$

$$\frac{s \equiv t \qquad t \equiv u}{s \equiv u}$$

$$\frac{s \equiv s' \qquad t \equiv t'}{s.t \equiv s'.t'}$$

$\equiv$ is (basically) the least congruence satisfying Del and Swap.

Properties of $\equiv$

- Symmetry, Transitivity, as well as the Deletion, Swap and Composition Rule are trivially fulfilled by $\equiv$.
- Reflexivity is also admissible: $s \equiv s$ is easily established by induction on s.

Table of Contents

Motivation

Wanted: normal form and normalizer η s.t.

- normal (η s)
- normal s $\rightarrow$ normal t $\rightarrow$ s $\equiv$ t $\rightarrow$ s $=$ t

The goal is to proof decidability of $\equiv$
$\equiv$ will be used to define set relations on trees.

Table of Contents

Lexicographic comparison

$$\text{Order} ::= \text{LE} \mid \text{EQ} \mid \text{GT}$$

$$
\begin{aligned}
&\text{cmp} : \text{Tree} \to \text{Tree} \to \text{Order}\\
&\text{cmp}\ \emptyset\ \emptyset = \text{EQ}\\
&\text{cmp}\ \emptyset\ \text{t} = \text{LT}\\
&\text{cmp}\ \text{s}\ \emptyset = \text{GT}\\
&\text{cmp}\ (\text{s1.s2})\ (\text{t1.t2}) = \text{match}\ (\text{cmp s1 t1})\ \text{with}\\
&\qquad \mid \text{EQ} \to \text{cmp}\ (\text{s2 t2})\\
&\qquad \mid \text{x} \to \text{x end}
\end{aligned}
$$

Table of Contents

Normal

$$\frac{}{\text{normalL } []}$$

$$\frac{\text{normal (L s)}}{\text{normalL [s]}}$$

$$\frac{\text{normalL(t :: l)} \qquad \text{normalL (L s)} \qquad \text{cmp s t} = \text{LT}}{\text{normalL (s :: t :: l)}}$$

normal is defined in terms of normalL as follows:
normal s := normalL (L s).

Table of Contents

Insertion Sort

```
insert s ∅ = s.∅
insert s (t1.t2) = match cmp s t1 with
    |EQ → t1.t2
    |LT → s.t1.t2
    |GT→ t1 . insert s t2 end

sort ∅ = ∅
sort (s.t) = insert (sort s) (sort t)
```

Properties of sort

We can prove the following properties by induction:

- s.t $\equiv$ insert s t
- s $\equiv$ sort s
- sort (s.s.t) = sort (s.t)
- sort(s.t.u) = sort(t.s.u)
- s $\equiv$ t $\rightarrow$ sort s = sort t
- normal (sort s)
- normal s $\rightarrow$ sort s = s ($\rightarrow$ Idempotency of sort)

Table of Contents

Decision Procedure

- two sorted sets are equivalent $\iff$ they are syntactically equal
- sort s $\equiv$ s
- $\equiv$ is transitive
- s $=$ t is decidable, since finitary sets are a simple inductive type

Table of Contents

$\in$, $\subseteq$

Element relationship

- $\mathsf{s} \in \mathsf{t} := \mathsf{t} \equiv \mathsf{s.t}$
- $\mathsf{s} \in \mathsf{t} \iff \exists\, \mathsf{t'}.\ \mathsf{t'} \in \mathsf{L}\ \mathsf{t} \land \mathsf{s} \equiv \mathsf{t'}.$

Subset relationship

- $\mathsf{s} \subseteq \mathsf{t} := \mathsf{t} \equiv \mathsf{s}\ @\ \mathsf{t}.$
- $\mathsf{s} \subseteq \mathsf{t} \iff \forall\, \mathsf{s'}.\ \mathsf{s'} \in \mathsf{s} \to \mathsf{s'} \in \mathsf{t}.$
- $s \subseteq t \to t \subseteq s \to s \equiv t$

Table of Contents

Coinductive Bisimulation

$s \sim t :=$
$\forall$ s. s' $\in$ L(s) $\rightarrow \exists$ t'. t $\in$ L(t) $\wedge$ s' $\sim$ t' $\wedge$
$\forall$ t. t' $\in$ L(t) $\rightarrow \exists$ s'. s $\in$ L(s) $\wedge$ t' $\sim$ s'.

Reflexivity, Symmetry, Swap and Del can be proven easily.

Transitivity of $\sim$

Coinduction lemma needed:
Lemma TreeCoInd (R : Tree → Tree → Prop) :
(∀ s t, R s t →
(∀ s. s' ∈ L(s) → ∃ t'. t ∈ L(t) ∧ R s' t' ∧
∀ t. t' ∈ L(t) → ∃ s'. s ∈ L(s) ∧ R t' s'.)) →
∀ s t, R s t → s ∼ t.

Transitivity follows by using the lemma with R := (∼ ∘ ∼)
⇒ s ≡ t → s ∼ t

$s \sim t \to s \equiv t$

If suffices to show:
$s \sim t \to s \subseteq t \land t \subseteq s$
Stronger induction lemma:

Lemma TreeInduction (P : Tree → Prop) :
(∀ s , (∀ t, t ∈ (L s) → P t) → P s) → ∀s, P s.

Table of Contents

References

Sangiorgi, Davide. Introduction to bisimulation and coinduction. Cambridge University Press, 2011.