Contents

1. Commutative Residu Algebras (CRAs)

- Branden Fitelson (University of California, Berkeley)
- Vincent van Oostrom (Universiteit Utrecht)
- Albert Visser (Universiteit Utrecht)

2. Origins of CRAs

Student Exercise:

- Write an (inductive) function `bubbel` in Coq which performs a single 'bubbel' step, and an (inductive) function `bubbel_sort` based upon it, which sorts a list via the bubblesort algorithm.
- Prove correctness of `bubbel` (what does it mean?).
- Bonus: prove correctness of `bubbel_sort` .

2. Origins of CRAs

Student Exercise:

- Write an (inductive) function `bubbel` in Coq which performs a single 'bubbel' step, and an (inductive) function `bubbel_sort` based upon it, which sorts a list via the bubblesort algorithm.
- Prove correctness of `bubbel` (what does it mean?).
- Bonus: prove correctness of `bubbel_sort` .

```
Theorem sort : (l:list){k:list| (is_sorted k)
& (list_eq l k)}.
```

2. Origins of CRAs

Student Exercise:

- Write an (inductive) function `bubbel` in Coq which performs a single 'bubbel' step, and an (inductive) function `bubbel_sort` based upon it, which sorts a list via the bubblesort algorithm.
- Prove correctness of `bubbel` (what does it mean?).
- Bonus: prove correctness of `bubbel_sort` .

```
Theorem sort : (l:list){k:list| (is_sorted k)
& (list_eq l k)}.
```

Proof is constructive, so a Haskell sort program can be extracted.

3. List equality

- Same up to permutation
- Same representative?

3. List equality

- Same up to permutation
- Same representative?

Same underlying multiset

3. List equality

- Same up to permutation
- Same representative?

Same underlying multiset

```
Fixpoint list_fmset [l:list] : fmset :=
Cases l of
  nil_list => fmset_empty
| (letter_list a k) =>
    (fmset_head a (list_fmset k))
end.
Definition list_eq :=
  [l,k:list](fmset_eq (list_fmset l)
  (list_fmset k)).
```

4. Formalizing Multisets?

- Coq: ‘multiplicity’ map to integers
  note: not finite

- Terese: residual system over ‘letters’
  note: not commutative

our solution:

4. Formalizing Multisets?

- Coq: ‘multiplicity’ map to integers
  note: not finite
- Terese: residual system over ‘letters’
  note: not commutative

our solution:
commutative residual systems over letters
(= commutative residual algebras)

4. Formalizing Multisets?

- Coq: ‘multiplicity’ map to integers
  note: not finite
- Terese: residual system over ‘letters’
  note: not commutative

our solution:
commutative residual systems over letters
(= commutative residual algebras)

bonus (for me):
formalization of part of Terese book

5. (Part of) Multiset Interface

- equality is an equivalence relation
- empty multiset, singleton, multiset sum
- sum is associative, commutative
- equality is congruence for sum
- empty multiset no elements
- element singleton is equal to it
- element sum is element of part

6. Definition of CRA

Algebra $(A, -, 0)$ satisfying the following axioms.

- cra1 $x - 0 = x$
- cra4 $(x - y) - (z - y) = (x - z) - (y - z)$ (cube)
- cra5 $(x - y) - x = 0$
- cra6 $x - (x - y) = y - (y - x)$ (commutativity)

7. Some CRAs

- natural numbers with cut-off subtraction
- bits with cut-off subtraction
- sets with set-difference
- multisets with multiset-difference
- positive natural numbers with c-o division
- rationals with cut-off subtraction

Non-example: integers with subtraction

8. Some facts on CRAs

- cra2 $x - x = 0$
- cra3 $0 - x = 0$
- $(x - y) - z = (x - z) - y$.
- $(x - y) - (x - z) = (z - y) - (z - x)$

8.1. Proof of $(x-y)-z=(x-z)-y$

$$
\begin{aligned}
(x-y)-z &= ((x-y)-z)-((z-y)-z) \\
&= ((x-y)-(z-y))-(z-(z-y)) \\
&= ((x-z)-(y-z))-(y-(y-z)) \\
&= ((x-z)-y)-((y-z)-y) \\
&= (x-z)-y
\end{aligned}
$$

9. The Natural Order

$x \leq y :\iff y - x = 0$

- $\leq$ is a partial ordering
- Minus is not determined by ordering (for infinite CRAs)
- Minus monotonic: if $x \leq x'$, then $x - y \leq x' - y$
- Minus is antitonic: if $y \leq y'$, then $x - y' \leq x - y$.

10. Infimum

$x \wedge y := (x - (x - y))$

- $\wedge$ is the infimum w.r.t. $\leq$.
- $\wedge$ is idempotent.
- $\wedge$ is commutative.
- $x \wedge y \leq x$ and $x \wedge y \leq y$.
- $x \leq y$ iff $x = x \wedge y$.
- $\wedge$ is associative.

10. Infimum

$x \wedge y := (x - (x - y))$

- $\wedge$ is the infimum w.r.t. $\leq$.
- $\wedge$ is idempotent.
- $\wedge$ is commutative.
- $x \wedge y \leq x$ and $x \wedge y \leq y$.
- $x \leq y$ iff $x = x \wedge y$.
- $\wedge$ is associative.

Otter

10.1. Minus and infimum

- $x - y = x - (x \wedge y)$.
- $(x - y) - (y - x) = x - y$.
- $(x - y) \wedge (y - x) = 0$.
- $(x \wedge y) - z = (x - z) \wedge (y - z)$.

11. Supremum and Sum

$a + b = c$ if $c - a = b$ and $a - c = 0$

$$a \vee b \equiv a + (b - a).$$

12. Related Work

12.1. Max-plus algebras

(()N,0,max,$+$)

not finitely axiomatiable

12.2. Iseki's BCK-algebras

- cbck1 $((x - y) - (x - z)) - (z - y) = 0$
- cbck2 $x - x = 0$
- cbck3 $0 - x = 0$
- cbck4 $(x - (x - y)) - y = 0)$
- cbck5 $x - y = y - x = 0$ implies $x = y$

- cbck6 $x - (x - y) = y - (y - x)$ (commutativity)

12.3. Unique Decomposition

Bas in two weeks