

# Regular Algebras

Simon Foster and Georg Struth

March 17, 2025

## Abstract

Regular algebras axiomatise the equational theory of regular expressions as induced by regular language identity. We use Isabelle/HOL for a detailed systematic study of regular algebras given by Boffa, Conway, Kozen and Salomaa. We investigate the relationships between these classes, formalise a soundness proof for the smallest class (Salomaa's) and obtain completeness of the largest one (Boffa's) relative to a deep result by Krob. In addition we provide a large collection of regular identities in the general setting of Boffa's axiom.

## Contents

<b>1</b>	<b>Introductory Remarks</b>	<b>2</b>
<b>2</b>	<b>Diodoids, Powers and Finite Sums</b>	<b>2</b>
<b>3</b>	<b>Regular Algebras</b>	<b>5</b>
3.1	Conway's Classical Axioms . . . . .	5
3.2	Boffa's Axioms . . . . .	9
3.3	Boffa Monoid Identities . . . . .	20
3.4	Conway's Conjectures . . . . .	22
3.5	Kozen's Kleene Algebras . . . . .	27
3.6	Salomaa's Axioms . . . . .	30
<b>4</b>	<b>Models of Regular Algebra</b>	<b>32</b>
4.1	Language Model of Salomaa Algebra . . . . .	32
4.2	Regular Language Model of Salomaa Algebra . . . . .	37
<b>5</b>	<b>Pratt's Counterexamples</b>	<b>41</b>
<b>6</b>	<b>Variants of Regular Algebra</b>	<b>47</b>

## 1 Introductory Remarks

These Isabelle theories complement the article on *On the Fine-Structure of Regular Algebra* [5]. For an introduction to the topic, conceptual explanations and references we refer to this article. Our regular algebra hierarchy is orthogonal to the Kleene algebra hierarchy in the Archive of Formal Proofs [1]; we have not aimed at an integration for pragmatic reasons.

## 2 Diods, Powers and Finite Sums

```
theory Diod-Power-Sum
  imports Kleene-Algebra.Diod Kleene-Algebra.Finite-Suprema
```

```
begin
```

We add a few facts about powers and finite sums—in fact, finite suprema—to an existing theory field for dioids.

```
context doid-one-zero
```

```
begin
```

```
lemma add-iso-r:  $y \leq z \Rightarrow x + y \leq x + z$ 
  using local.join.sup-mono by blast
```

```
notation power (op^ [101,50] 100)
```

```
lemma power-subdist:  $x^n \leq (x + y)^n$ 
  apply (induct n)
  apply simp
  using local.mult-isol-var local.power-Suc2 by auto
```

```
lemma power-inductl-var:  $x \cdot y \leq y \Rightarrow x^n \cdot y \leq y$ 
  apply (induct n)
  apply simp
  by (metis (no-types, lifting) local.dual-order.trans local.mult-isol local.power-Suc2
mult-assoc)
```

```
lemma power-inductr-var:  $y \cdot x \leq y \Rightarrow y \cdot x^n \leq y$ 
  by (induct n, metis eq-refl mult-oner power.simps(1), metis mult.assoc mult-isor
order-refl order-trans power.simps(2) power-commutes)
```

```
definition powsum :: 'a :: nat ⇒ nat ⇒ nat :: (op^ [101,50,50] 100) where
  powsum x m n = sum ((op^ x) {m..n} + m)
```

```
lemmas powsum-simps = powsum-def atLeastAtMostSuc-conv numerals
```

```
lemma powsum1 [simp]:  $x_n^0 = x^n$ 
  by (simp add:powsum-simps)
```

```

lemma powsum2:  $x_n^{Suc m} = x_n^m + x^{n+Suc m}$ 
proof-
  have  $x_n^{Suc m} = \text{sum } ((\wedge x) \{n..(Suc m)+n\})$ 
    using powsum-def by blast
  also have ... =  $\text{sum } ((\wedge x) \{n..m+n\}) + x^{n+Suc m}$ 
    by (simp add: ab-semigroup-add-class.add.commute atLeastAtMostSuc-conv loc.al.join.sup-commute)
  finally show ?thesis
    by (simp add: powsum-def)
qed

lemma powsum-00 [simp]:  $x_0^0 = 1$ 
  by (simp add: powsum-def)

lemma powsum-01 [simp]:  $x_0^1 = 1 + x$ 
  by (simp add: powsum2)

lemma powsum-10 [simp]:  $x_1^0 = x$ 
  by (simp add: powsum-simps)

lemma powsum-split:  $x_m^{i+Suc n} = x_m^i + x_{m+Suc i}^n$ 
  by (induct n, simp-all add:powsum-simps ac-simps)

lemma powsum-split-var1:  $x_0^{n+1} = 1 + x_1^n$ 
proof -
  have  $x_0^{n+1} = x_0^n + Suc n$ 
    by simp
  also have ... =  $x_0^n + x_0 + Suc_0^n$ 
    by (subst powsum-split, rule refl)
  also have ... =  $1 + x_0 + Suc_0^n$ 
    by simp
  finally show ?thesis
    by simp
qed

lemma powsum-split-var2 [simp]:  $x^m + x_0^m = x_0^m$ 
proof (induct m)
  case 0 show ?case
    by (metis add-idem' power-0 powsum-00)
  case (Suc n) show ?case
    by (simp add: add-commute powsum2)
qed

lemma powsum-split-var3:  $x_0^{m+Suc n} = x_0^m + x_{0+Suc m}^n$ 
  by (subst powsum-split, simp)

lemma powsum-split-var4 [simp]:  $x_0^{m+n} + x_m^n = x_0^{m+n}$ 
proof (induct n)

```

```

case 0 show ?case
  by (metis add-0-iff add-comm powsum1 powsum-split-var2)
next
  case (Suc n) note hyp = this
  show ?case
  proof -
    have  $x_0^m + \text{Suc } n + x_m^{\text{Suc } n} = x_0^m + n + x^{\text{Suc } (m+n)} + (x_m^n + x^m + \text{Suc } n)$ 
      by (auto simp add: powsum2)
    also have ... =  $(x_0^m + n + x_m^n) + x^{\text{Suc } (m+n)} + x^m + \text{Suc } n$ 
      by (metis add.assoc add.left-commute)
    also have ... =  $x^{\text{Suc } (m+n)} + x_0^{m+n}$ 
      by (metis add-Suc-right add.assoc add.commute add-idem' hyp)
    also have ... =  $x_0^m + \text{Suc } n$ 
      by (simp add: add-commute powsum2)
    finally show ?thesis .
  qed
qed

lemma powsum-split-var6:  $x_0^{(\text{Suc } k)+\text{Suc } n} = x_0^{\text{Suc } k} + x_0^{+\text{Suc } (\text{Suc } k)^n}$ 
  by (metis powsum-split-var3)

lemma powsum-ext:  $x \leq x_0^{\text{Suc } n}$ 
proof (induct n)
  case 0 show ?case
    by (metis One-nat-def local.join.sup-ge2 powsum-01)
next
  case (Suc n) thus ?case
    by (auto simp add:less-eq-def powsum-simps, metis (lifting, no-types) add.left-commute)
qed

lemma powsum-one:  $1 \leq x_0^{\text{Suc } n}$ 
  by (induct n, metis One-nat-def local.join.sup.cobounded1 powsum-01, metis (full-types) Suc-eq-plus1 local.join.sup.cobounded1 powsum-split-var1)

lemma powsum-shift1:  $x \cdot x_m^n = x_{m+1}^n$ 
  apply (induct n)
  apply (simp-all add: powsum-simps)
  apply (metis local.add-left-comm local.distrib-left powsum-def)
  done

lemma powsum-shift:  $x^k \cdot x_m^n = x_{k+m}^n$ 
  by (induct k, simp-all, metis Suc-eq-plus1 mult.assoc powsum-shift1)

lemma powsum-prod-suc:  $x_0^m \cdot x_0^{\text{Suc } n} = x_0^{\text{Suc } (m+n)}$ 
proof (induct m)
  case 0 show ?case
    by simp
  case (Suc m)
  note hyp = this

```

```

show ?case
proof -
have  $x_0^{Suc m} \cdot x_0^{Suc n} = x_0^m \cdot x_0^{Suc n} + x^{Suc m} \cdot x_0^{Suc n}$ 
  by (simp add: powsum2)
also have ... =  $x_0^{Suc(m+n)} + x^{Suc m} \cdot x_0^{Suc n}$ 
  by (simp add:hyp)
also have ... =  $x_0^{Suc(m+n)} + x^{Suc m}^{Suc n}$ 
  by (subst powsum-shift, simp)
also have ... =  $x_0^{Suc(m+n)} + (x^{Suc m}^n + x^{Suc m} + Suc n)$ 
  by (simp add:powsum2)
also have ... =  $x_0^{Suc(m+n)} + x^{Suc m}^n + x^{Suc(Suc(m+n))}$ 
  by (metis add-Suc-right add-Suc-shift add.assoc add.left-commute)
also have ... =  $x_0^{Suc(m+n)} + x^{Suc(Suc(m+n))}$ 
  by (simp only: add-Suc-right[THEN sym] add-Suc-shift[THEN sym] powsum-split-var4)
also have ... =  $x_0^{Suc(Suc m + n)}$ 
  by (simp add: powsum2)
finally show ?thesis .
qed
qed

lemma powsum-prod:  $x_0^m \cdot x_0^n = x_0^{m+n}$ 
  by (cases n, simp, simp add: powsum-prod-suc)

end

end

```

### 3 Regular Algebras

```

theory Regular-Algebras
imports Diod-Power-Sum Kleene-Algebra Finite-Suprema Kleene-Algebra Kleene-Algebra
begin

```

#### 3.1 Conway's Classical Axioms

Conway's classical axiomatisation of Regular Algebra from [4].

```

class star-diodoid = dioid-one-zero + star-op + plus-ord

class conway-diodoid = star-diodoid +
assumes C11:  $(x + y)^* = (x^* \cdot y)^* \cdot x^*$ 
and C12:  $(x \cdot y)^* = 1 + x \cdot (y \cdot x)^* \cdot y$ 

class strong-conway-diodoid = conway-diodoid +
assumes C13:  $(x^*)^* = x^*$ 

class C-algebra = strong-conway-diodoid +

```

**assumes** *C14*:  $x^* = (x^{n+1})^* \cdot x_0^n$

We tried to dualise using sublocales, but this causes an infinite loop on dual.dual.dual....

**lemma** (*in conway-diodoid*) *C11-var*:  $(x + y)^* = x^* \cdot (y \cdot x^*)^*$

**proof** –

**have**  $x^* \cdot (y \cdot x^*)^* = x^* + x^* \cdot y \cdot (x^* \cdot y)^* \cdot x^*$

**by** (*metis C12 distrib-left mult.assoc mult-oner*)

**also have** ... =  $(1 + x^* \cdot y \cdot (x^* \cdot y)^*) \cdot x^*$

**by** (*metis distrib-right mult.assoc mult-oner*)

**finally show** *?thesis*

**by** (*metis C11 C12 mult-oner mult-oner*)

**qed**

**lemma** (*in conway-diodoid*) *dual-conway-diodoid*:

*class.conway-diodoid (+) (⊖) 1 0 (≤) (<) star*

**proof**

**fix**  $x y z :: 'a$

**show**  $(x ⊖ y) ⊕ z = x ⊖ (y ⊕ z)$

**by** (*metis mult.assoc opp-mult-def*)

**show**  $(x + y) ⊖ z = x ⊖ z + y ⊖ z$

**by** (*metis distrib-left opp-mult-def*)

**show**  $x + x = x$

**by** (*metis add-idem*)

**show**  $1 ⊖ x = x$

**by** (*metis mult-oner opp-mult-def*)

**show**  $x ⊖ 1 = x$

**by** (*metis mult-onel opp-mult-def*)

**show**  $0 + x = x$

**by** (*metis add-zero*)

**show**  $0 ⊖ x = 0$

**by** (*metis annil opp-mult-def*)

**show**  $x ⊖ (y + z) = x ⊖ y + x ⊖ z$

**by** (*metis opp-mult-def distrib-right'*)

**show**  $x ⊖ 0 = 0$

**by** (*metis annil opp-mult-def*)

**show**  $(x + y)^* = (x^* ⊖ y)^* ⊕ x^*$

**by** (*metis C11-var opp-mult-def*)

**show**  $(x ⊖ y)^* = 1 + x ⊖ (y ⊕ x)^* ⊕ y$

**by** (*metis C12 mult.assoc opp-mult-def*)

**qed**

**lemma** (*in strong-conway-diodoid*) *dual-strong-conway-diodoid*: *class.strong-conway-diodoid*

*((+) ) ((⊖) ) 1 0 (≤) (<) star*

**proof**

**fix**  $x y z :: 'a$

**show**  $(x ⊖ y) ⊕ z = x ⊖ (y ⊕ z)$

**by** (*metis mult.assoc opp-mult-def*)

**show**  $(x + y) ⊖ z = x ⊖ z + y ⊖ z$

```

by (metis distrib-left opp-mult-def)
show  $x + x = x$ 
  by (metis add-idem)
show  $1 \odot x = x$ 
  by (metis mult-oner opp-mult-def)
show  $x \odot 1 = x$ 
  by (metis mult-onel opp-mult-def)
show  $0 + x = x$ 
  by (metis add-zero)
show  $0 \odot x = 0$ 
  by (metis annir opp-mult-def)
show  $x \odot (y + z) = x \odot y + x \odot z$ 
  by (metis opp-mult-def distrib-right')
show  $x \odot 0 = 0$ 
  by (metis annil opp-mult-def)
show  $(x + y)^* = (x^* \odot y)^* \odot x^*$ 
  by (metis C11-var opp-mult-def)
show  $(x \odot y)^* = 1 + x \odot (y \odot x)^* \odot y$ 
  by (metis C12 mult.assoc opp-mult-def)
show  $(x^*)^* = x^*$ 
  by (metis C13)
qed

```

Nitpick finds counterexamples to the following claims.

```

lemma (in conway-diodoid)  $1^* = 1$ 
  nitpick [expect=genuine] — 3-element counterexample
  oops

```

```

lemma (in conway-diodoid)  $(x^*)^* = x^*$ 
  nitpick [expect=genuine] — 3-element counterexample
  oops

```

```

context C-algebra
begin

```

```

lemma C-unfoldl [simp]:  $1 + x \cdot x^* = x^*$ 
  by (metis C12 mult-onel mult-oner)

```

```

lemma C-slide:  $(x \cdot y)^* \cdot x = x \cdot (y \cdot x)^*$ 

```

```

proof-
  have  $(x \cdot y)^* \cdot x = x + x \cdot (y \cdot x)^* \cdot y \cdot x$ 
    by (metis C12 mult-onel distrib-right')
  also have ... =  $x \cdot (1 + (y \cdot x)^* \cdot y \cdot x)$ 
    by (metis distrib-left mult.assoc mult-oner)
  finally show ?thesis
    by (metis C12 mult.assoc mult-onel mult-oner)
qed

```

```

lemma powsum-ub:  $i \leq n \implies x^i \leq x_0^n$ 

```

```

proof (induct n)
  case 0 show ?case
    by (metis (opaque-lifting, mono-tags) 0.prems order.eq-iff le-0-eq power-0 pow-
sum-00)
  next
    case (Suc n) show ?case
    proof -
      { assume aa1: Suc n ≠ i
        have ff1:  $x_0^{Suc n} \leq x_0^{Suc n} \wedge Suc n \neq i$ 
          using aa1 by fastforce
        have ff2:  $\exists x_1. x_0^n + x_1 \leq x_0^{Suc n} \wedge Suc n \neq i$ 
          using ff1 powsum2 by auto
        have  $x^i \leq x_0^{Suc n}$ 
          by (metis Suc.hyps Suc.prems ff2 le-Suc-eq local.dual-order.trans local.join.le-supE)
      }
      thus  $x^i \leq x_0^{Suc n}$ 
        using local.less-eq-def local.powsum-split-var2 by blast
    qed
    qed

lemma C14-aux:  $m \leq n \implies x^m \cdot (x^n)^* = (x^n)^* \cdot x^m$ 
proof -
  assume assm:  $m \leq n$ 
  hence  $x^m \cdot (x^n)^* = x^m \cdot (x^{n-m} \cdot x^m)^*$ 
    by (metis (full-types) le-add-diff-inverse2 power-add)
  also have ... =  $(x^{n-m} \cdot x^m)^* \cdot x^m$ 
    by (metis (opaque-lifting, mono-tags) C-slide ab-semigroup-add-class.add.commute
power-add)
  finally show ?thesis
    by (metis (full-types) assm le-add-diff-inverse ab-semigroup-add-class.add.commute
power-add)
  qed

end

context dioid-one-zero
begin

lemma opp-power-def:
  power.power 1 (⊖)  $x n = x^n$ 
proof (induction n)
  case 0 thus ?case
    by (metis power.power.power-0)
  next
    case (Suc n) thus ?case
      by (metis power.power.power-Suc power-Suc2 times.opp-mult-def)
  qed

lemma opp-powsum-def:

```

```

diod-one-zero.powsum (+) (⊖) 1 0 x m n = xmn
proof –
  have sum (power.power 1 (⊖) x) {m..n + m} = sum ((⊸) x) {m..n + m}
    by (induction n, simp-all add:opp-power-def)
  thus ?thesis
    by (simp add: dioid-one-zero.powsum-def[of ---- (≤) (<)] dual-dioid-one-zero
          powsum-def)
  qed

end

lemma C14-dual:
  fixes x::'a::C-algebra
  shows x* = x0n · (xn+1)*
proof –
  have x* = (xn+1)* · x0n
    by (rule C14)
  also have ... = (xn+1)* · (Σ i=0..n. x⊖i)
    by (subst powsum-def, auto)
  also have ... = (Σ i=0..n. (xn+1)* · x⊖i)
    by (metis le0 sum-interval-distl)
  also have ... = (Σ i=0..n. x⊖i · (xn+1)*)
    by (auto intro: sum-interval-cong simp only:C14-aux)
  also have ... = x0n · (xn+1)*
    by (simp only: sum-interval-distr[THEN sym] powsum-def Nat.add-0-right)
  finally show ?thesis .
qed

```

```

lemma C-algebra: class.C-algebra (+) (⊖) (1::'a::C-algebra) 0 (≤) (<) star
proof
  fix x y :: 'a and n :: nat
  show (x + y)* = (x* ⊖ y)* ⊖ x*
    by (metis C11-var opp-mult-def)
  show (x ⊖ y)* = 1 + x ⊖ (y ⊖ x)* ⊖ y
    by (metis C12 mult.assoc opp-mult-def)
  show (x*)* = x*
    by (metis C13)
  show x* = power.power 1 (⊖) x (n + 1)* ⊖ dioid-one-zero.powsum (+) (⊖) 1
  0 x 0 n
    by (metis C14-dual opp-mult-def opp-power-def opp-powsum-def)
  qed (simp-all add: opp-mult-def mult.assoc distrib-left)

```

### 3.2 Boffa's Axioms

Boffa's two axiomatisations of Regular Algebra from [2, 3].

```

class B1-algebra = conway-dioid +
  assumes R: x · x = x ==> x* = 1 + x

class B2-algebra = star-dioid +

```

**assumes**  $B21: 1 + x \leq x^*$   
**and**  $B22 [simp]: x^* \cdot x^* = x^*$   
**and**  $B23: \llbracket 1 + x \leq y; y \cdot y = y \rrbracket \implies x^* \leq y$

**lemma (in  $B1$ -algebra)**  $B1$ -algebra:  
*class.B1-algebra (+) (⊖) 1 0 (≤) (<) star*

**proof**

```

fix x y z :: 'a
show x ⊕ y ⊕ z = x ⊕ (y ⊕ z)
  by (metis mult.assoc opp-mult-def)
show (x + y) ⊕ z = x ⊕ z + y ⊕ z
  by (metis distrib-left opp-mult-def)
show x + x = x
  by (metis add-idem)
show 1 ⊕ x = x
  by (metis mult-oner opp-mult-def)
show x ⊕ 1 = x
  by (metis mult-onel opp-mult-def)
show 0 + x = x
  by (metis add-zerol)
show 0 ⊕ x = 0
  by (metis annir opp-mult-def)
show x ⊕ (y + z) = x ⊕ y + x ⊕ z
  by (metis distrib-right opp-mult-def)
show x ⊕ 0 = 0
  by (metis annil opp-mult-def)
show (x + y)^* = (x^* ⊕ y)^* ⊕ x^*
  by (metis C11-var opp-mult-def)
show (x ⊕ y)^* = 1 + x ⊕ (y ⊕ x)^* ⊕ y
  by (metis C12 mult.assoc opp-mult-def)
show x ⊕ x = x  $\implies$  x^* = 1 + x
  by (metis R opp-mult-def)

```

**qed**

**lemma (in  $B2$ -algebra)**  $B2$ -algebra:  
*class.B2-algebra (+) (⊖) 1 0 (≤) (<) star*

**proof**

```

fix x y z :: 'a
show x ⊕ y ⊕ z = x ⊕ (y ⊕ z)
  by (metis mult.assoc opp-mult-def)
show (x + y) ⊕ z = x ⊕ z + y ⊕ z
  by (metis distrib-left times.opp-mult-def)
show x + x = x
  by (metis add-idem)
show 1 ⊕ x = x
  by (metis mult-oner opp-mult-def)
show x ⊕ 1 = x
  by (metis mult-onel opp-mult-def)
show 0 + x = x

```

```

by (metis add-zero)
show 0 ⊕ x = 0
  by (metis annir opp-mult-def)
show x ⊕ (y + z) = x ⊕ y + x ⊕ z
  by (metis distrib-right opp-mult-def)
show x ⊕ 0 = 0
  by (metis annil opp-mult-def)
show 1 + x ≤ x*
  by (metis B21)
show x* ⊕ x* = x*
  by (metis B22 opp-mult-def)
show [| 1 + x ≤ y; y ⊕ y = y |] ⇒ x* ≤ y
  by (metis B23 opp-mult-def)
qed

instance B1-algebra ⊆ B2-algebra
proof
  fix x y :: 'a
  show 1 + x ≤ x*
    by (metis C12 add-iso-r distrib-right join.sup.cobounded1 mult-onel)
  show two: x* · x* = x*
    by (metis (no-types, lifting) C11-var C12 R add-idem' mult-onel mult-oner)
  show [| 1 + x ≤ y; y · y = y |] ⇒ x* ≤ y
    by (metis (no-types, lifting) C11-var R two distrib-left join.sup.bounded-iff
less-eq-def mult.assoc mult.right-neutral)
  qed

context B2-algebra
begin

lemma star-ref: 1 ≤ x*
  using local.B21 by auto

lemma star-plus-one [simp]: 1 + x* = x*
  by (metis less-eq-def star-ref)

lemma star-trans: x* · x* ≤ x*
  by (metis B22 order-refl)

lemma star-trans-eq [simp]: x* · x* = x*
  by (metis B22)

lemma star-invol [simp]: (x*)* = x*
  by (metis B21 B22 B23 order.antisym star-plus-one)

lemma star-1l: x · x* ≤ x*
  by (metis local.B21 local.join.sup.boundedE local.mult-isor local.star-trans-eq)

lemma star-one [simp]: 1* = 1

```

**by** (metis B23 add-idem order.antisym mult-oner order-refl star-ref)

**lemma** star-subdist:  $x^* \leq (x + y)^*$   
**by** (meson local.B21 local.B23 local.join.sup.bounded-iff local.star-trans-eq)

**lemma** star-iso:  $x \leq y \implies x^* \leq y^*$   
**by** (metis less-eq-def star-subdist)

**lemma** star2:  $(1 + x)^* = x^*$   
**by** (metis B21 add.commute less-eq-def star-invol star-subdist)

**lemma** star-unfoldl:  $1 + x \cdot x^* \leq x^*$   
**by** (metis local.join.sup.bounded-iff star-1l star-ref)

**lemma** star-unfoldr:  $1 + x^* \cdot x \leq x^*$   
**by** (metis (full-types) B21 local.join.sup.bounded-iff mult-isol star-trans-eq)

**lemma** star-ext:  $x \leq x^*$   
**by** (metis B21 local.join.sup.bounded-iff)

**lemma** star-1r:  $x^* \cdot x \leq x^*$   
**by** (metis mult-isol star-ext star-trans-eq)

**lemma** star-unfoldl-eq [simp]:  $1 + x \cdot x^* = x^*$   
**proof** –  
**have**  $(1 + x \cdot x^*) \cdot (1 + x \cdot x^*) = 1 \cdot (1 + x \cdot x^*) + x \cdot x^* \cdot (1 + x \cdot x^*)$   
**by** (metis distrib-right)  
**also have** ... =  $1 + x \cdot x^* + (x \cdot x^* \cdot x \cdot x^*)$   
**by** (metis add-assoc' add-idem' distrib-left mult.assoc mult-onel mult-oner)  
**also have** ... =  $1 + x \cdot x^*$   
**by** (metis add.assoc add.commute distrib-left less-eq-def mult.assoc star-1l star-trans-eq)  
**finally show** ?thesis  
**by** (metis B23 local.join.sup.mono local.join.sup.cobounded1 distrib-left order.eq-iff mult-1-right star-plus-one star-unfoldl)  
**qed**

**lemma** star-unfoldr-eq [simp]:  $1 + x^* \cdot x = x^*$   
**proof** –  
**have**  $(1 + x^* \cdot x) \cdot (1 + x^* \cdot x) = 1 \cdot (1 + x^* \cdot x) + x^* \cdot x \cdot (1 + x^* \cdot x)$   
**by** (metis distrib-right)  
**also have** ... =  $1 + x^* \cdot x + (x^* \cdot x \cdot x^* \cdot x)$   
**by** (metis add.assoc add-idem' distrib-left mult-1-left mult-1-right mult.assoc)  
**also have** ... =  $1 + x^* \cdot x$   
**by** (metis add-assoc' distrib-left mult.assoc mult-oner distrib-right' star-trans-eq star-unfoldl-eq)  
**finally show** ?thesis  
**by** (metis B21 B23 add.commute local.join.sup.mono local.join.sup.cobounded1 order.eq-iff eq-refl mult-1-left distrib-right' star-unfoldl-eq star-unfoldr)

**qed**

**lemma** star-prod-unfold-le:  $(x \cdot y)^* \leq 1 + x \cdot (y \cdot x)^* \cdot y$

**proof** –

have  $(1 + x \cdot (y \cdot x)^* \cdot y) \cdot (1 + x \cdot (y \cdot x)^* \cdot y) =$   
 $1 \cdot (1 + x \cdot (y \cdot x)^* \cdot y) + (x \cdot (y \cdot x)^* \cdot y) \cdot (1 + x \cdot (y \cdot x)^* \cdot y)$

by (metis distrib-right')

also have ... =  $1 + x \cdot (y \cdot x)^* \cdot y + x \cdot (y \cdot x)^* \cdot y \cdot x \cdot (y \cdot x)^* \cdot y$

by (metis add.assoc local.join.sup.cobounded1 distrib-left less-eq-def mult-1-right mult.assoc mult-onel)

finally have  $(1 + x \cdot (y \cdot x)^* \cdot y) \cdot (1 + x \cdot (y \cdot x)^* \cdot y) = 1 + x \cdot (y \cdot x)^* \cdot y$   
by (metis add.assoc distrib-left distrib-right mult.assoc mult-oner star-trans-eq star-unfoldr-eq)

moreover have  $(x \cdot y) \leq 1 + x \cdot (y \cdot x)^* \cdot y$

by (metis local.join.sup.cobounded2 mult-1-left mult.assoc mult-double-iso order-trans star-ref)

ultimately show ?thesis

by (simp add: local.B23)

**qed**

**lemma** star-prod-unfold [simp]:  $1 + x \cdot (y \cdot x)^* \cdot y = (x \cdot y)^*$

**proof** –

have  $1 + x \cdot (y \cdot x)^* \cdot y \leq 1 + x \cdot (1 + y \cdot (x \cdot y)^* \cdot x) \cdot y$

by (metis local.join.sup.mono mult-double-iso order-refl star-prod-unfold-le)

also have ... =  $1 + x \cdot y + x \cdot y \cdot (x \cdot y)^* \cdot x \cdot y$

by (metis add.assoc distrib-left mult-1-left mult.assoc distrib-right')

finally have  $1 + x \cdot (y \cdot x)^* \cdot y \leq (x \cdot y)^*$

by (metis add.assoc distrib-left mult-1-right mult.assoc star-unfoldl-eq star-unfoldr-eq)

thus ?thesis

by (metis order.antisym star-prod-unfold-le)

**qed**

**lemma** star-slide1:  $(x \cdot y)^* \cdot x \leq x \cdot (y \cdot x)^*$

**proof** –

have  $(x \cdot y)^* \cdot x = (1 + x \cdot (y \cdot x)^* \cdot y) \cdot x$

by (metis star-prod-unfold)

also have ... =  $(x + x \cdot (y \cdot x)^* \cdot y \cdot x)$

by (metis mult-onel distrib-right')

also have ... =  $x \cdot (1 + (y \cdot x)^* \cdot y \cdot x)$

by (metis distrib-left mult.assoc mult-oner)

finally show ?thesis

by (metis eq-refl mult.assoc star-unfoldr-eq)

**qed**

**lemma** star-slide-var1:  $x^* \cdot x \leq x \cdot x^*$

by (metis mult-onel mult-oner star-slide1)

**lemma** star-slide:  $(x \cdot y)^* \cdot x = x \cdot (y \cdot x)^*$

**proof** (rule order.antisym)

```

show  $(x \cdot y)^* \cdot x \leq x \cdot (y \cdot x)^*$ 
  by (metis star-slide1)
have  $x \cdot (y \cdot x)^* = x \cdot (1 + y \cdot (x \cdot y)^* \cdot x)$ 
  by (metis star-prod-unfold)
also have ... =  $x + x \cdot y \cdot (x \cdot y)^* \cdot x$ 
  by (metis distrib-left mult.assoc mult-oner)
also have ... =  $(1 + x \cdot y \cdot (x \cdot y)^*) \cdot x$ 
  by (metis mult-onel distrib-right')
finally show  $x \cdot (y \cdot x)^* \leq (x \cdot y)^* \cdot x$ 
  by (metis mult-isor star-unfoldl)
qed

lemma star rtc1:  $1 + x + x^* \cdot x^* \leq x^*$ 
  using local.B21 local.join.sup-least local.star-trans by blast

lemma star rtc1-eq:  $1 + x + x^* \cdot x^* = x^*$ 
  by (metis B21 B22 less-eq-def)

lemma star-subdist-var-1:  $x \leq (x + y)^*$ 
  using local.join.le-supE local.star-ext by blast

lemma star-subdist-var-2:  $x \cdot y \leq (x + y)^*$ 
  by (metis (full-types) local.join.le-supE mult-isol-var star-ext star-trans-eq)

lemma star-subdist-var-3:  $x^* \cdot y^* \leq (x + y)^*$ 
  by (metis add.commute mult-isol-var star-subdist star-trans-eq)

lemma R-lemma:
  assumes  $x \cdot x = x$ 
  shows  $x^* = 1 + x$ 
  proof (rule order.antisym)
    show  $1 + x \leq x^*$ 
      by (metis B21)
    have  $(1 + x) \cdot (1 + x) = 1 + x$ 
      by (metis add.commute add-idem' add.left-commute assms distrib-left mult-onel
            mult-onel distrib-right')
    thus  $x^* \leq 1 + x$ 
      by (metis B23 order-refl)
  qed

lemma star-denest-var-0:  $(x + y)^* = (x^* \cdot y)^* \cdot x^*$ 
  proof (rule order.antisym)
    have one-below:  $1 \leq (x^* \cdot y)^* \cdot x^*$ 
      by (metis mult-isol-var star-one star-ref star-trans-eq)
    have x-below:  $x \leq (x^* \cdot y)^* \cdot x^*$ 
      by (metis mult-isol mult-onel order-trans star-ext star-ref star-slide)
    have y-below:  $y \leq (x^* \cdot y)^* \cdot x^*$ 
      by (metis mult-isol-var mult-onel order-trans star-ext star-slide star-unfoldl-eq
            subdistl)

```

```

from one-below x-below y-below have  $1 + x + y \leq (x^* \cdot y)^* \cdot x^*$ 
  by simp
  moreover have  $(x^* \cdot y)^* \cdot x^* \cdot (x^* \cdot y)^* \cdot x^* = (x^* \cdot y)^* \cdot x^*$ 
    by (metis star-trans-eq star-slide mult.assoc)
  ultimately show  $(x + y)^* \leq (x^* \cdot y)^* \cdot x^*$ 
    by (metis B23 add-assoc' mult.assoc)
  show  $(x^* \cdot y)^* \cdot x^* \leq (x + y)^*$ 
    by (metis (full-types) add.commute mult-isol-var star-invol star-iso star-subdist-var-1
      star-trans-eq)
qed

lemma star-denest:  $(x + y)^* = (x^* \cdot y^*)^*$ 
  by (metis R-lemma add.commute star-denest-var-0 star-plus-one star-prod-unfold
    star-slide star-trans-eq)

lemma star-sum-var:  $(x + y)^* = (x^* + y^*)^*$ 
  by (metis star-denest star-invol)

lemma star-denest-var:  $(x + y)^* = x^* \cdot (y \cdot x^*)^*$ 
  by (metis star-denest-var-0 star-slide)

lemma star-denest-var-2:  $(x^* \cdot y^*)^* = x^* \cdot (y \cdot x^*)^*$ 
  by (metis star-denest star-denest-var)

lemma star-denest-var-3:  $(x^* \cdot y^*)^* = x^* \cdot (y^* \cdot x^*)^*$ 
  by (metis B22 add-comm mult.assoc star-denest star-denest-var-2)

lemma star-denest-var-4:  $(x^* \cdot y^*)^* = (y^* \cdot x^*)^*$ 
  by (metis add-comm star-denest)

lemma star-denest-var-5:  $x^* \cdot (y \cdot x^*)^* = y^* \cdot (x \cdot y^*)^*$ 
  by (metis add.commute star-denest-var)

lemma star-denest-var-6:  $(x + y)^* = x^* \cdot y^* \cdot (x + y)^*$ 
  by (metis mult.assoc star-denest star-denest-var-3)

lemma star-denest-var-7:  $(x + y)^* = (x + y)^* \cdot x^* \cdot y^*$ 
  by (metis star-denest star-denest-var star-denest-var-3 star-denest-var-5 star-slide)

lemma star-denest-var-8:  $(x^* \cdot y^*)^* = x^* \cdot y^* \cdot (x^* \cdot y^*)^*$ 
  by (metis star-denest star-denest-var-6)

lemma star-denest-var-9:  $(x^* \cdot y^*)^* = (x^* \cdot y^*)^* \cdot x^* \cdot y^*$ 
  by (metis star-denest star-denest-var-7)

lemma star-slide-var:  $x^* \cdot x = x \cdot x^*$ 
  by (metis mult-1-left mult-oner star-slide)

lemma star-sum-unfold:  $(x + y)^* = x^* + x^* \cdot y \cdot (x + y)^*$ 

```

by (metis distrib-left mult-1-right mult.assoc star-denest-var star-unfoldl-eq)

**lemma** troeger:  $x^* \cdot (y \cdot ((x + y)^* \cdot z) + z) = (x + y)^* \cdot z$

**proof** –

have  $x^* \cdot (y \cdot ((x + y)^* \cdot z) + z) = (x^* + x^* \cdot y \cdot (x + y)^*) \cdot z$

by (metis add-comm distrib-left mult.assoc distrib-right')

thus ?thesis

by (metis star-sum-unfold)

qed

**lemma** meyer-1:  $x^* = (1 + x) \cdot (x \cdot x)^*$

**proof** (rule order.antisym)

have  $(1 + x) \cdot (x \cdot x)^* \cdot (1 + x) \cdot (x \cdot x)^* = ((x \cdot x)^* + x \cdot (x \cdot x)^*) \cdot ((x \cdot x)^* + x \cdot (x \cdot x)^*)$

by (metis mult.assoc mult-onel distrib-right')

also have ... =  $((x \cdot x)^* + x \cdot (x \cdot x)^*) \cdot (x \cdot x)^* + ((x \cdot x)^* + x \cdot (x \cdot x)^*) \cdot x \cdot (x \cdot x)^*$

by (metis distrib-left mult.assoc)

also have ... =  $(x \cdot x)^* \cdot (x \cdot x)^* + x \cdot (x \cdot x)^* \cdot (x \cdot x)^* + (x \cdot x)^* \cdot x \cdot (x \cdot x)^* + x \cdot (x \cdot x)^* \cdot x \cdot (x \cdot x)^*$

by (metis combine-common-factor distrib-right)

also have ... =  $(x \cdot x)^* + x \cdot (x \cdot x)^* + x \cdot x \cdot (x \cdot x)^*$

by (metis add.assoc add-idem' mult.assoc star-slide star-trans-eq)

also have ... =  $1 + x \cdot x \cdot (x \cdot x)^* + x \cdot (x \cdot x)^* + x \cdot x \cdot (x \cdot x)^*$

by (metis star-unfoldl-eq)

also have ... =  $(x \cdot x)^* + x \cdot (x \cdot x)^*$

by (metis add-comm add-idem' add.left-commute star-unfoldl-eq)

finally have  $(1 + x) \cdot (x \cdot x)^* \cdot (1 + x) \cdot (x \cdot x)^* = (1 + x) \cdot (x \cdot x)^*$

by (metis mult-1-left distrib-right')

moreover have  $1 + x \leq (1 + x) \cdot (x \cdot x)^*$

by (metis mult-1-right star-unfoldl-eq subdistl)

ultimately show  $x^* \leq (1 + x) \cdot (x \cdot x)^*$

by (metis B23 mult.assoc)

next

have  $(1 + x) \cdot (x \cdot x)^* = (x \cdot x)^* + x \cdot (x \cdot x)^*$

by (metis mult-1-left distrib-right')

thus  $(1 + x) \cdot (x \cdot x)^* \leq x^*$

by (metis local.add-zeror local.join.sup-least local.mult-isol-var local.mult-oner local.star-ext local.star-invol local.star-iso local.star-subdist-var-2 local.star-trans-eq local.subdistl-eq)

qed

**lemma** star-zero [simp]:  $0^* = 1$

by (metis add-zeror star2 star-one)

**lemma** star-subsum [simp]:  $x^* + x^* \cdot x = x^*$

by (metis add.assoc add-idem star-slide-var star-unfoldl-eq)

**lemma** prod-star-closure:  $x \leq z^* \implies y \leq z^* \implies x \cdot y \leq z^*$

```

by (metis mult-isol-var star-trans-eq)

end

sublocale B2-algebra ⊆ B1-algebra
  by unfold-locales (metis star-denest-var-0, metis star-prod-unfold, metis R-lemma)

context B2-algebra
begin

lemma power-le-star:  $x^n \leq x^*$ 
  by (induct n, simp-all add: star-ref prod-star-closure star-ext)

lemma star-power-slide:
  assumes  $k \leq n$ 
  shows  $x^k \cdot (x^n)^* = (x^n)^* \cdot x^k$ 
proof -
  from assms have  $x^k \cdot (x^n)^* = (x^k \cdot x^{n-k})^* \cdot x^k$ 
    by (metis (full-types) le-add-diff-inverse2 power-add star-slide)
  with assms show ?thesis
    by (metis (full-types) le-add-diff-inverse2 ab-semigroup-add-class.add.commute
power-add)
qed

lemma powsum-le-star:  $x_m^n \leq x^*$ 
  by (induct n, simp-all add: powsum2, metis power-le-star, metis power-Suc
power-le-star)

lemma star-sum-power-slide:
  assumes  $m \leq n$ 
  shows  $x_0^m \cdot (x^n)^* = (x^n)^* \cdot x_0^m$ 
using assms
proof (induct m)
  case 0 thus ?case
    by (metis mult-onel mult-oner powsum-00)
next
  case (Suc m) note hyp = this
  have  $x_0^{Suc m} \cdot (x^n)^* = (x_0^m + x^{Suc m}) \cdot (x^n)^*$ 
    by (simp add:powsum2)
  also have ... =  $x_0^m \cdot (x^n)^* + x^{Suc m} \cdot (x^n)^*$ 
    by (metis distrib-right')
  also have ... =  $(x^n)^* \cdot x_0^m + x^{Suc m} \cdot (x^n)^*$ 
    by (metis Suc.hyps Suc.premS Suc-leD)
  also have ... =  $(x^n)^* \cdot x_0^m + (x^n)^* \cdot x^{Suc m}$ 
    by (metis Suc.premS star-power-slide)
  also have ... =  $(x^n)^* \cdot (x_0^m + x^{Suc m})$ 
    by (metis distrib-left)
  finally show ?case
    by (simp add:powsum2)

```

qed

**lemma** aarden-aux:

assumes  $y \leq y \cdot x + z$

shows  $y \leq y \cdot x^{(\text{Suc } n)} + z \cdot x^*$

**proof** (induct n)

case 0

have  $y \cdot x + z \leq y \cdot x^{(\text{Suc } 0)} + z \cdot x^*$

by (metis (mono-tags) One-nat-def add.commute add-iso mult-1-right power-one-right star-plus-one subdistl)

thus ?case

by (metis assms order-trans)

next

case (Suc n)

have  $y \cdot x + z \leq (y \cdot x^{(\text{Suc } n)} + z \cdot x^*) \cdot x + z$

by (metis Suc add-iso mult-isor)

also have ... =  $y \cdot x^{(\text{Suc } n)} \cdot x + z \cdot (x^* \cdot x + 1)$

by (subst distrib-right, metis add-assoc' distrib-left mult.assoc mult-oner)

finally show ?case

by (metis add.commute assms mult.assoc order-trans power-Suc2 star-unfoldr-eq)

qed

**lemma** conway-powerstar1:  $(x^{n+1})^* \cdot x_0^n \cdot (x^{n+1})^* \cdot x_0^n = (x^{n+1})^* \cdot x_0^n$

**proof** (cases n)

case 0 thus ?thesis

by simp

next

case (Suc m) thus ?thesis

**proof** –

assume assm:  $n = \text{Suc } m$

have  $(x^{m+2})^* \cdot x_0^{m+1} \cdot (x^{m+2})^* \cdot x_0^{m+1} = (x^{m+2})^* \cdot (x^{m+2})^* \cdot x_0^{m+1} \cdot x_0^{m+1}$

by (subgoal-tac  $m + 1 \leq m + 2$ , metis mult.assoc star-sum-power-slide, simp)

also have ... =  $(x^{m+2})^* \cdot x_0^{m+1} \cdot x_0^{m+1}$

by (metis star-trans-eq)

also have ... =  $(x^{m+2})^* \cdot x_0^{(\text{Suc } m) + (\text{Suc } m)}$

by (simp add: mult.assoc powsum-prod)

also have ... =  $(x^{m+2})^* \cdot (x_0^{\text{Suc } m} + x_m + 2^m)$

by (metis monoid-add-class.add.left-neutral powsum-split-var3 add-2-eq-Suc')

also have ... =  $(x^{m+2})^* \cdot x_0^{\text{Suc } m} + (x^{m+2})^* \cdot x_{(m+2)+0^m}$

by (simp add: local.distrib-left)

also have ... =  $(x^{m+2})^* \cdot x_0^{\text{Suc } m} + (x^{m+2})^* \cdot x^{m+2} \cdot x_0^m$

by (subst powsum-shift[THEN sym], metis mult.assoc)

also have ... =  $(x^{m+2})^* \cdot (x_0^m + x^{m+1}) + (x^{m+2})^* \cdot x^{m+2} \cdot x_0^m$

by (simp add:powsum2)

also have ... =  $(x^{m+2})^* \cdot x_0^m + (x^{m+2})^* \cdot x^{m+2} \cdot x_0^m + (x^{m+2})^* \cdot x^{m+1}$

by (metis add.assoc add.commute add.left-commute distrib-left mult.assoc)

also have ... =  $(x^{m+2})^* \cdot x_0^m + (x^{m+2})^* \cdot x_0^m + (x^{m+2})^* \cdot x^{m+1}$

```

by (metis add-idem' distrib-right' star-subsum)
also have ... =  $(x^{m+2})^* \cdot (x_0^m + x^{m+1})$ 
  by (metis add-idem' distrib-left)
also have ... =  $(x^{m+2})^* \cdot x_0^{m+1}$ 
  by (simp add:powsum2)
finally show ?thesis
  by (simp add: assm)
qed
qed

lemma conway-powerstar2:  $1 + x \leq (x^{n+1})^* \cdot x_0^n$ 
proof (cases n)
  case 0 show ?thesis
    using 0 local.B21 by auto
next
  case (Suc m) show ?thesis
  proof -
    have one:  $x \leq (x^{m+2})^* \cdot x_0^{m+1}$ 
      by (metis Suc-eq-plus1 powsum-ext mult-isor mult-onel order-trans star-ref)
    have two:  $1 \leq (x^{m+2})^* \cdot x_0^{m+1}$ 
      by (metis Suc-eq-plus1 local.join.le-supE mult-isor mult-onel powsum-split-var1
star-ref)
    from one two show ?thesis
      by (metis Suc Suc-eq-plus1 add-2-eq-Suc' local.join.sup-least)
  qed
qed

theorem powerstar:  $x^* = (x^{n+1})^* \cdot x_0^n$ 
proof (rule order.antisym)
  show  $x^* \leq (x^{n+1})^* \cdot x_0^n$ 
    by (metis conway-powerstar1 conway-powerstar2 mult.assoc B23)
  have  $(x^{n+1})^* \cdot x_0^n \leq (x^*)^* \cdot x_0^n$ 
    by (metis mult-isor power-le-star star-iso)
  also have ... =  $x^* \cdot x_0^n$ 
    by (metis star-invol)
  also have ...  $\leq x^* \cdot x^*$ 
    by (simp add: local.prod-star-closure powsum-le-star)
  finally show  $(x^{n+1})^* \cdot x_0^n \leq x^*$ 
    by (metis star-trans-eq)
qed

end

sublocale B2-algebra ⊆ strong-conway-diodoid
  by unfold-locales (metis star-invol)

sublocale B2-algebra ⊆ C-algebra
  by unfold-locales (metis powerstar)

```

The following fact could neither be verified nor falsified in Isabelle. It does

not hold for other reasons.

```
lemma (in C-algebra)  $x \cdot x = x \longrightarrow x^* = 1 + x$ 
oops
```

### 3.3 Boffa Monoid Identities

```
typedef ('a , 'b) boffa-mon = {f :: 'a::{finite,monoid-mult}  $\Rightarrow$  'b::B1-algebra.
True}
by auto
```

**notation**

 $Rep\text{-}boffa\text{-}mon (\langle \cdot \cdot \rangle)$ 

```
lemma finite (range (Rep-boffa-mon M))
by (metis finite-code finite-imageI)
```

```
abbreviation boffa-pair :: ('a, 'b) boffa-mon  $\Rightarrow$  'a::{finite,monoid-mult}  $\Rightarrow$  'a  $\Rightarrow$ 
'b::B1-algebra where
boffa-pair x i j  $\equiv$   $\sum \{ x_k \mid k. i \cdot k = j \}$ 
```

**notation**

 $boffa\text{-}pair (\langle \cdot \cdot \cdot \rangle)$ 

**abbreviation** conway-assms **where**

 $conway\text{-}assms x \equiv (\forall i j. (x_i \cdot x_j \leq x_{i \cdot j}) \wedge (x_{i,i})^* = x_{i,i})$ 

```
lemma pair-one:  $x_{1,1} = x_1$ 
by (simp)
```

```
definition conway-assm1 where conway-assm1 x = ( $\forall i j. x_i \cdot x_j \leq x_{i \cdot j}$ )
definition conway-assm2 where conway-assm2 x = ( $\forall i. x_{i,i}^* = x_{i,i}$ )
```

```
lemma pair-star:
assumes conway-assm2 x
shows  $x_1^* = x_1$ 
proof -
  have  $x_1^* = x_{1,1}^*$ 
  by simp
  also from assms have ...  $= x_{1,1}$ 
  by (metis (mono-tags) conway-assm2-def)
  finally show ?thesis
  by simp
qed
```

```
lemma conway-monoid-one:
assumes conway-assm2 x
shows  $x_1 = 1 + x_1$ 
proof -
  from assms have  $x_1 = x_1^*$ 
```

```

by (metis pair-star)
thus ?thesis
  by (metis star-plus-one)
qed

lemma conway-monoid-split:
  assumes conway-assm2 x
  shows  $\sum \{x_i \mid i . i \in \text{UNIV}\} = 1 + \sum \{x_i \mid i . i \in \text{UNIV}\}$ 
proof -
  have  $\sum \{x_i \mid i . i \in \text{UNIV}\} = \sum \{x_i \mid i . i \in (\text{insert } 1 (\text{UNIV} - \{1\}))\}$ 
    by (metis UNIV-I insert-Diff-single insert-absorb)
  also have ... =  $\sum (\text{Rep-boffa-mon } x ` (\text{insert } 1 (\text{UNIV} - \{1\})))$ 
    by (metis fset-to-im)
  also have ... =  $x_1 + \sum (\text{Rep-boffa-mon } x ` (\text{UNIV} - \{1\}))$ 
    by (subst sum-fun-insert, auto)
  also have ... =  $x_1 + \sum \{x_i \mid i . i \in (\text{UNIV} - \{1\})\}$ 
    by (metis fset-to-im)
  also from assms have unfld:... =  $1 + x_1 + \sum \{x_i \mid i . i \in (\text{UNIV} - \{1\})\}$ 
    by (metis (lifting, no-types) conway-monoid-one)
  finally show ?thesis
    by (metis (lifting, no-types) ac-simps unfld)
qed

lemma boffa-mon-aux1:  $\{x_{i,j} \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\} = \{x_i \mid i . i \in \text{UNIV}\}$ 
  by (auto, metis monoid-mult-class.mult.left-neutral)

lemma sum-intro' [intro]:
  [| finite (A :: 'a::join-semilattice-zero set); finite B;  $\forall a \in A . \exists b \in B . a \leq b$  |]  $\implies$ 
   $\sum A \leq \sum B$ 
  by (metis sum-intro)

lemma boffa-aux2:
  conway-assm1 x  $\implies$ 
   $\sum \{x_i x_j \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\} \leq \sum \{x_{i,j} \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\}$ 
  unfolding conway-assm1-def
  using [[simproc add: finite-Collect]]
  by force

lemma boffa-aux3:
  assumes conway-assm1 x
  shows  $(\sum \{x_i \mid i . i \in \text{UNIV}\}) + (\sum \{x_i \cdot x_j \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\}) = (\sum \{x_i \mid i . i \in \text{UNIV}\})$ 
proof -
  from assms
  have  $(\sum \{x_i \mid i . i \in \text{UNIV}\}) + (\sum \{x_i \cdot x_j \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\}) \leq (\sum \{x_i \mid i . i \in \text{UNIV}\}) + (\sum \{x_{i,j} \mid i j . i \in \text{UNIV} \wedge j \in \text{UNIV}\})$ 
    apply (subst add-iso-r)
    apply (subst boffa-aux2)
    by simp-all

```

```

also have ... = ( $\sum \{x_i \mid i. i \in UNIV\}$ )
  by (metis (mono-tags) add-idem boffa-mon-aux1)
ultimately show ?thesis
  by (simp add: dual-order.antisym)
qed

lemma conway-monoid-identity:
assumes conway-assm1 x conway-assm2 x
shows ( $\sum \{x_i \mid i. i \in UNIV\}$ ) $^*$  = ( $\sum \{x_i \mid i. i \in UNIV\}$ )
proof -
  have one:( $\sum \{x_i \mid i. i \in UNIV\}$ )  $\cdot$  ( $\sum \{x_i \mid i. i \in UNIV\}$ ) = (1 + ( $\sum \{x_i \mid i. i \in UNIV\}$ ))
  · (1 + ( $\sum \{x_i \mid i. i \in UNIV\}$ ))
    by (metis (mono-tags) assms(2) conway-monoid-split)
  also have ... = 1 + ( $\sum \{x_i \mid i. i \in UNIV\}$ ) + (( $\sum \{x_i \mid i. i \in UNIV\}$ )  $\cdot$  ( $\sum \{x_i \mid i. i \in UNIV\}$ ))
    by (metis (lifting, no-types) calculation less-eq-def mult-isol mult-isol-equiv-subdistl
      mult-oner)
  also have ... = 1 + ( $\sum \{x_i \mid i. i \in UNIV\}$ ) + ( $\sum \{x_i \cdot x_j \mid i. j. i \in UNIV \wedge j \in UNIV\}$ )
    by (simp only: dioiod-sum-prod finite-UNIV)
  finally have  $\sum \{x_i \mid i. i \in UNIV\} \cdot \sum \{x_i \mid i. i \in UNIV\} = \sum \{x_i \mid i. i \in UNIV\}$ 
    apply (simp only:)
proof -
  assume a1:  $\sum \{x_i \mid i. i \in UNIV\} \cdot \sum \{x_i \mid i. i \in UNIV\} = 1 + \sum \{x_i \mid i. i \in UNIV\} + \sum \{x_i \cdot x_j \mid i. j. i \in UNIV \wedge j \in UNIV\}$ 
  hence  $\sum \{x_R \mid R. R \in UNIV\} \cdot \sum \{x_R \mid R. R \in UNIV\} = \sum \{x_R \mid R. R \in UNIV\}$ 
    using assms(1) assms(2) boffa-aux3 conway-monoid-split by fastforce
    thus 1 +  $\sum \{x_i \mid i. i \in UNIV\} + \sum \{x_i \cdot x_j \mid i. j. i \in UNIV \wedge j \in UNIV\} = \sum \{x_i \mid i. i \in UNIV\}$ 
      using a1 by simp
qed
thus ?thesis
  by (metis (mono-tags) one B1-algebra-class.R star-trans-eq)
qed

```

### 3.4 Conway's Conjectures

```

class C0-algebra = strong-conway-dioiod +
assumes C0:  $x \cdot y = y \cdot z \implies x^* \cdot y = y \cdot z^*$ 

class C1l-algebra = strong-conway-dioiod +
assumes C1l:  $x \cdot y \leq y \cdot z \implies x^* \cdot y \leq y \cdot z^*$ 

class C1r-algebra = strong-conway-dioiod +
assumes C1r:  $y \cdot x \leq z \cdot y \implies y \cdot x^* \leq z^* \cdot y$ 

class C2l-algebra = conway-dioiod +

```

```

assumes C2l:  $x = y \cdot x \implies x = y^* \cdot x$ 

class C2r-algebra = conway-dioïd +
assumes C2r:  $x = x \cdot y \implies x = x \cdot y^*$ 

class C3l-algebra = conway-dioïd +
assumes C3l:  $x \cdot y \leq y \implies x^* \cdot y \leq y$ 

class C3r-algebra = conway-dioïd +
assumes C3r:  $y \cdot x \leq y \implies y \cdot x^* \leq y$ 

sublocale C1r-algebra  $\subseteq$  dual: C1l-algebra
  (+) ( $\odot$ ) 1 0 ( $\leq$ ) (<) star
proof
  fix x y z :: 'a
  show  $x \odot y \odot z = x \odot (y \odot z)$ 
    by (metis mult.assoc opp-mult-def)
  show  $(x + y) \odot z = x \odot z + y \odot z$ 
    by (metis distrib-left opp-mult-def)
  show  $x + x = x$ 
    by (metis add-idem)
  show  $1 \odot x = x$ 
    by (metis mult-onel opp-mult-def)
  show  $x \odot 1 = x$ 
    by (metis mult-onel opp-mult-def)
  show  $0 + x = x$ 
    by (metis add-zerol)
  show  $0 \odot x = 0$ 
    by (metis annir opp-mult-def)
  show  $x \odot (y + z) = x \odot y + x \odot z$ 
    by (metis distrib-right opp-mult-def)
  show  $x \odot 0 = 0$ 
    by (metis annil opp-mult-def)
  show  $(x + y)^* = (x^* \odot y)^* \odot x^*$ 
    by (metis C11-var opp-mult-def)
  show  $(x \odot y)^* = 1 + x \odot (y \odot x)^* \odot y$ 
    by (metis C12 mult.assoc opp-mult-def)
  show  $(x^*)^* = x^*$ 
    by (metis C13)
  show  $x \odot y \leq y \odot z \implies x^* \odot y \leq y \odot z^*$ 
    by (metis C1r opp-mult-def)
qed

sublocale C2r-algebra  $\subseteq$  dual: C2l-algebra
  (+) ( $\odot$ ) 1 0 ( $\leq$ ) (<) star
proof
  fix x y z :: 'a
  show  $x \odot y \odot z = x \odot (y \odot z)$ 
    by (metis mult.assoc opp-mult-def)

```

```

show  $(x + y) \odot z = x \odot z + y \odot z$ 
  by (metis distrib-left opp-mult-def)
show  $x + x = x$ 
  by (metis add-idem)
show  $1 \odot x = x$ 
  by (metis mult-oner opp-mult-def)
show  $x \odot 1 = x$ 
  by (metis mult-onel opp-mult-def)
show  $0 + x = x$ 
  by (metis add-zero)
show  $0 \odot x = 0$ 
  by (metis annir opp-mult-def)
show  $x \odot (y + z) = x \odot y + x \odot z$ 
  by (metis distrib-right opp-mult-def)
show  $x \odot 0 = 0$ 
  by (metis annil opp-mult-def)
show  $(x + y)^* = (x^* \odot y)^* \odot x^*$ 
  by (metis C11-var opp-mult-def)
show  $(x \odot y)^* = 1 + x \odot (y \odot x)^* \odot y$ 
  by (metis C12 mult.assoc opp-mult-def)
show  $x = y \odot x \implies x = y^* \odot x$ 
  by (metis C2r opp-mult-def)
qed

sublocale C3r-algebra ⊆ dual: C3l-algebra
  (+) (⊖) 0 (≤) (<) star
proof
  fix x y z :: 'a
  show  $x \odot y \odot z = x \odot (y \odot z)$ 
    by (metis mult.assoc opp-mult-def)
  show  $(x + y) \odot z = x \odot z + y \odot z$ 
    by (metis distrib-left opp-mult-def)
  show  $x + x = x$ 
    by (metis add-idem)
  show  $1 \odot x = x$ 
    by (metis mult-oner opp-mult-def)
  show  $x \odot 1 = x$ 
    by (metis mult-onel opp-mult-def)
  show  $0 + x = x$ 
    by (metis add-zero)
  show  $0 \odot x = 0$ 
    by (metis annir opp-mult-def)
  show  $x \odot (y + z) = x \odot y + x \odot z$ 
    by (metis distrib-right opp-mult-def)
  show  $x \odot 0 = 0$ 
    by (metis annil opp-mult-def)
  show  $(x + y)^* = (x^* \odot y)^* \odot x^*$ 
    by (metis C11-var opp-mult-def)
  show  $(x \odot y)^* = 1 + x \odot (y \odot x)^* \odot y$ 
    by (metis C12 mult.assoc opp-mult-def)

```

```

by (metis C12 mult.assoc opp-mult-def)
show  $x \odot y \leq y \implies x^* \odot y \leq y$ 
  by (metis C3r opp-mult-def)
qed

```

```

lemma (in C3l-algebra) k2-var:  $z + x \cdot y \leq y \implies x^* \cdot z \leq y$ 
  by (metis local.C3l local.join.le-supE local.join.sup.absorb2 local.subdistl)

```

**instance**  $C2l\text{-algebra} \subseteq B1\text{-algebra}$

```

  by (intro-classes, metis C2l monoid-mult-class.mult.left-neutral mult-oner con-
way-diodid-class.C12)

```

**instance**  $C2r\text{-algebra} \subseteq B1\text{-algebra}$

```

  by (intro-classes, metis C2r conway-diodid-class.C12)

```

The following claims are refuted by Nitpick

```

lemma (in conway-diodid)
  assumes  $x \cdot y = y \cdot z \implies x^* \cdot y = y \cdot z^*$ 
  shows  $1^* = 1$ 

```

**oops**

```

lemma (in conway-diodid)
  assumes  $x \cdot y \leq y \cdot z \implies x^* \cdot y \leq y \cdot z^*$ 
  shows  $1^* = 1$ 

```

**oops**

The following fact could not be refuted by Nitpick or Quickcheck; but an infinite counterexample exists.

```

lemma (in B1-algebra)  $x = x \cdot y \rightarrow x = x \cdot y^*$ 
  oops

```

**instance**  $C3l\text{-algebra} \subseteq C2l\text{-algebra}$

```

  by (intro-classes, metis C3l conway-diodid-class.C12 dual-order.antisym join.sup.cobounded1
mult-isol-var mult-onel order-refl)

```

**sublocale**  $C2l\text{-algebra} \subseteq C3l\text{-algebra}$

**proof**

**fix**  $x y$

**show**  $x \cdot y \leq y \implies x^* \cdot y \leq y$

**proof** –

**assume**  $x \cdot y \leq y$

**hence**  $(x + 1) \cdot y = y$

**by** (metis less-eq-def mult-onel distrib-right')

**hence**  $(x + 1)^* \cdot y = y$

**by** (metis C2l)

**hence**  $x^* \cdot y = y$

**by** (metis C11 C2l add-comm mult-1-left mult-1-right)

```

thus  $x^* \cdot y \leq y$ 
  by (metis eq-refl)
qed
qed

sublocale C1l-algebra ⊆ C3l-algebra
  by unfold-locales (metis mult-oner C1l C12 C13 add-zeror annir)

sublocale C3l-algebra ⊆ C1l-algebra
proof
  fix x y z
  show  $(x^*)^* = x^*$ 
    by (metis local.C11-var local.C12 local.C3l order.eq-iff local.eq-refl local.join.sup.absorb2
local.join.sup-ge1 local.mult-onel local.mult-oner)
  show  $x \cdot y \leq y \cdot z \implies x^* \cdot y \leq y \cdot z^*$ 
  proof -
    assume assm:  $x \cdot y \leq y \cdot z$ 
    have r1:  $y \leq y \cdot z^*$ 
      by (metis C12 mult-isol mult-oner order-prop)
    from assm have  $x \cdot y \cdot z^* \leq y \cdot z \cdot z^*$ 
      by (metis mult-isor)
    also have ... ≤  $y \cdot z^*$ 
      by (metis local.C12 local.join.sup-commute local.mult-onel local.mult-oner
local.subdistl mult-assoc)
    finally have  $y + x \cdot y \cdot z^* \leq y \cdot z^*$ 
      by (simp add: r1)
    thus  $x^* \cdot y \leq y \cdot z^*$ 
      by (metis k2-var mult.assoc)
  qed
qed

sublocale C1l-algebra ⊆ C2l-algebra
  by (unfold-locales, metis C12 C3l add.commute local.join.sup.cobounded1 dis-
trib-right less-eq-def mult-1-left order-refl)

sublocale C3r-algebra ⊆ C2r-algebra
  by (unfold-locales, metis C12 C3r add.commute local.join.sup.cobounded1 dis-
trib-left less-eq-def mult-1-right order-refl)

sublocale C2r-algebra ⊆ C3r-algebra
  by unfold-locales (metis dual.C3l opp-mult-def)

sublocale C1r-algebra ⊆ C3r-algebra
  by unfold-locales (metis dual.C3l opp-mult-def)

sublocale C3r-algebra ⊆ C1r-algebra
  by (unfold-locales, metis dual.C13, metis dual.C1l opp-mult-def)

class C1-algebra = C1l-algebra + C1r-algebra

```

```

class C2-algebra = C2l-algebra + C2r-algebra

class C3-algebra = C3l-algebra + C3r-algebra

sublocale C0-algebra ⊆ C2-algebra
  by unfold-locales (metis C12 C13 add-zeror annil mult-oner mult-onel C0) +
sublocale C2-algebra ⊆ C0-algebra
  by unfold-locales (metis C1l C1r order.eq-iff)

sublocale C2-algebra ⊆ C1-algebra ..
sublocale C1-algebra ⊆ C2-algebra ..
sublocale C2-algebra ⊆ C3-algebra ..
sublocale C3-algebra ⊆ C2-algebra ..

```

### 3.5 Kozen's Kleene Algebras

Kozen's Kleene Algebras [7, 6].

```

class Kl-base = star-diodoid +
  assumes Kl:  $1 + x \cdot x^* \leq x^*$ 

class Kr-base = star-diodoid +
  assumes Kr:  $1 + x^* \cdot x \leq x^*$ 

class K1l-algebra = Kl-base +
  assumes star-inductl:  $x \cdot y \leq y \implies x^* \cdot y \leq y$ 

class K1r-algebra = Kr-base +
  assumes star-inductr:  $y \cdot x \leq y \implies y \cdot x^* \leq y$ 

class K2l-algebra = Kl-base +
  assumes star-inductl-var:  $z + x \cdot y \leq y \implies x^* \cdot z \leq y$ 

class K2r-algebra = Kr-base +
  assumes star-inductr-var:  $z + y \cdot x \leq y \implies z \cdot x^* \leq y$ 

class K1-algebra = K1l-algebra + K1r-algebra

class K2-algebra = K2l-algebra + K2r-algebra

sublocale K1r-algebra ⊆ dual: K1l-algebra
  (+) (⊖) 1 0 (≤) (<) star
proof
  fix x y z :: 'a
  show x ⊕ y ⊕ z = x ⊕ (y ⊕ z)

```

```

    by (metis mult.assoc opp-mult-def)
show (x + y) ⊕ z = x ⊕ z + y ⊕ z
    by (metis distrib-left opp-mult-def)
show x + x = x
    by (metis add-idem)
show 1 ⊕ x = x
    by (metis mult-oner opp-mult-def)
show x ⊕ 1 = x
    by (metis mult-onel opp-mult-def)
show 0 + x = x
    by (metis add-zerol)
show 0 ⊕ x = 0
    by (metis annir opp-mult-def)
show x ⊕ (y + z) = x ⊕ y + x ⊕ z
    by (metis distrib-right opp-mult-def)
show x ⊕ 0 = 0
    by (metis annil opp-mult-def)
show 1 + x ⊕ x* ≤ x*
    by (metis Kr opp-mult-def)
show x ⊕ y ≤ y ==> x* ⊕ y ≤ y
    by (metis star-inductr opp-mult-def)
qed

sublocale K1l-algebra ⊆ B2-algebra
proof
fix x y :: 'a
show 1 + x ≤ x*
    by (metis add-iso-r local.join.sup.cobounded1 mult-isol mult-oner order-trans
Kl)
show x* · x* = x*
    using local.Kl order.eq-iff local.phl-cons1 local.star-inductl by fastforce
show [| 1 + x ≤ y; y · y = y |] ==> x* ≤ y
    by (metis local.distrib-right' local.join.le-sup-iff local.join.sup.order-iff local.mult-isol
local.mult-oner local.star-inductl)
qed

sublocale K1r-algebra ⊆ B2-algebra
by unfold-locales (metis dual.B21 dual.B22 dual.B23 opp-mult-def)+

sublocale K1l-algebra ⊆ C2l-algebra
by (unfold-locales, metis less-eq-def mult-1-left mult.assoc star-inductl star-invol
star-one star-plus-one star-trans-eq troeger)

sublocale C2l-algebra ⊆ K1l-algebra
by (unfold-locales, metis C12 order.eq-iff mult-1-left mult-1-right, metis C3l)

sublocale K1r-algebra ⊆ C2r-algebra
by unfold-locales (metis dual.C2l opp-mult-def)

```

```

sublocale C2r-algebra  $\subseteq$  K1r-algebra
  by (unfold-locales, metis dual.star-unfoldl opp-mult-def, metis C3r)

sublocale K1-algebra  $\subseteq$  C0-algebra
  by unfold-locales (metis C1l C1r order.eq-iff)

sublocale C0-algebra  $\subseteq$  K1l-algebra ..

sublocale K2r-algebra  $\subseteq$  dual: K2l-algebra
  (+) ( $\odot$ ) 1 0 ( $\leq$ ) ( $<$ ) star
proof
  fix x y z :: 'a
  show x  $\odot$  y  $\odot$  z = x  $\odot$  (y  $\odot$  z)
    by (metis mult.assoc opp-mult-def)
  show (x + y)  $\odot$  z = x  $\odot$  z + y  $\odot$  z
    by (metis distrib-left opp-mult-def)
  show x + x = x
    by (metis add-idem)
  show 1  $\odot$  x = x
    by (metis mult-oner opp-mult-def)
  show x  $\odot$  1 = x
    by (metis mult-onel opp-mult-def)
  show 0 + x = x
    by (metis add-zero)
  show 0  $\odot$  x = 0
    by (metis annir opp-mult-def)
  show x  $\odot$  (y + z) = x  $\odot$  y + x  $\odot$  z
    by (metis distrib-right opp-mult-def)
  show x  $\odot$  0 = 0
    by (metis annil opp-mult-def)
  show 1 + x  $\odot$  x $^*$   $\leq$  x $^*$ 
    by (metis opp-mult-def Kr)
  show z + x  $\odot$  y  $\leq$  y  $\implies$  x $^*$   $\odot$  z  $\leq$  y
    by (metis opp-mult-def star-inductr-var)
qed

sublocale K1l-algebra  $\subseteq$  K2l-algebra
  by unfold-locales (metis local.join.le-supE star-inductl order-prop subdistl)

sublocale K2l-algebra  $\subseteq$  K1l-algebra
  by (unfold-locales, simp add: local.star-inductl-var)

sublocale K1r-algebra  $\subseteq$  K2r-algebra
  by unfold-locales (metis dual.star-inductl-var opp-mult-def)

sublocale K2r-algebra  $\subseteq$  K1r-algebra
  by unfold-locales (metis dual.star-inductl opp-mult-def)

sublocale kleene-algebra  $\subseteq$  K1-algebra

```

**by** (*unfold-locales, metis star-unfoldl, metis star-inductl-var, metis star-unfoldr, metis star-inductr-var*)

**sublocale**  $K1\text{-algebra} \subseteq K2\text{-algebra} ..$

**sublocale**  $K2\text{-algebra} \subseteq koz\text{: kleene-algebra}$

**by** (*unfold-locales, metis Kl, metis star-inductl-var, metis star-inductr-var*)

### 3.6 Salomaa's Axioms

Salomaa's axiomatisations of Regular Algebra [9].

```
class salomaa-base = star-diod +  
  fixes ewp :: 'a ⇒ bool  
  assumes S11:  $(1 + x)^* = x^*$   
  and EWP:  $\text{ewp } x \longleftrightarrow (\exists y. x = 1 + y \wedge \neg \text{ewp } y)$ 
```

```
class Sr-algebra = salomaa-base +  
  assumes S12r:  $1 + x^* \cdot x = x^*$   
  and Ar:  $\llbracket \neg \text{ewp } y; x = x \cdot y + z \rrbracket \implies x = z \cdot y^*$ 
```

The following claim is ruled out by Nitpick. The unfold law cannot be weakened as in Kleene algebra.

```
lemma (in salomaa-base)  
  assumes S12r':  $1 + x^* \cdot x \leq x^*$   
  and Ar':  $\llbracket \neg \text{ewp } y; x = x \cdot y + z \rrbracket \implies x = z \cdot y^*$   
  shows  $x^* \leq 1 + x^* \cdot x$ 
```

oops

```
class Sl-algebra = salomaa-base +  
  assumes S12l:  $1 + x \cdot x^* = x^*$   
  and Al:  $\llbracket \neg \text{ewp } y; x = y \cdot x + z \rrbracket \implies x = y^* \cdot z$ 
```

class S-algebra = Sl-algebra + Sr-algebra

**sublocale**  $Sl\text{-algebra} \subseteq dual\text{: } Sr\text{-algebra}$   
 $(+) (\odot) 1 0 (\leq) (<) star \text{ ewp}$

**proof**

```
  fix x y z :: 'a  
  show  $(x \odot y) \odot z = x \odot (y \odot z)$   
    by (metis mult.assoc opp-mult-def)  
  show  $(x + y) \odot z = x \odot z + y \odot z$   
    by (metis distrib-left opp-mult-def)  
  show  $x + x = x$   
    by (metis add-idem)  
  show  $1 \odot x = x$   
    by (metis mult-oner opp-mult-def)  
  show  $x \odot 1 = x$ 
```

```

    by (metis mult-onel opp-mult-def)
show 0 + x = x
    by (metis add-zero)
show 0 ⊕ x = 0
    by (metis annir times.opp-mult-def)
show x ⊕ (y + z) = x ⊕ y + x ⊕ z
    by (metis opp-mult-def distrib-right')
show x ⊕ 0 = 0
    by (metis annil opp-mult-def)
show (1 + x)★ = x★
    by (metis S11)
show ewp x = (∃ y. x = 1 + y ∧ ¬ ewp y)
    by (metis EWP)
show 1 + x★ ⊕ x = x★
    by (metis S12l opp-mult-def)
show [¬ ewp y; x = x ⊕ y + z] ⇒ x = z ⊕ y★
    by (metis opp-mult-def Al)
qed

context Sr-algebra
begin

lemma kozen-induct-r:
assumes y · x + z ≤ y
shows z · x★ ≤ y
proof (cases ewp x)
case False thus ?thesis
    by (metis add-commute assms local.Ar local.join.le-supE local.join.sup.orderE
local.mult-isor)
next
case True thus ?thesis
proof -
assume ewp x
then obtain x' where assm1: x = 1 + x' and assm2: ¬ ewp x'
    by (metis EWP)
have y = (z + y) · x★
    by (metis S11 local.join.le-supE assm1 assm2 assms order.eq-iff less-eq-def Ar
subdistl)
thus z · x★ ≤ y
    by (metis local.join.sup.cobounded1 local.mult-isor)
qed
qed

end

context Sl-algebra
begin

lemma kozen-induct-l:

```

```

assumes  $x \cdot y + z \leq y$ 
shows  $x^* \cdot z \leq y$ 
by (metis dual.kozen-induct-r times.opp-mult-def assms)

end

sublocale Sr-algebra ⊆ K2r-algebra
by unfold-locales (metis S12r order-refl, metis add-comm kozen-induct-r)

sublocale Sr-algebra ⊆ K1r-algebra ..

sublocale Sl-algebra ⊆ K2l-algebra
by unfold-locales (metis S12l order-refl, metis add-comm kozen-induct-l)

sublocale Sl-algebra ⊆ K1l-algebra ..

sublocale S-algebra ⊆ K1-algebra ..

sublocale S-algebra ⊆ K2-algebra ..

The following claim could be refuted.

lemma (in K2-algebra) ( $\neg 1 \leq x$ ) —>  $x = x \cdot y + z \rightarrow x = z \cdot y^*$ 
oops

class salomaa-conj-r = salomaa-base +
assumes salomaa-small-unfold:  $1 + x^* \cdot x = x^*$ 
assumes salomaa-small-r:  $\llbracket \neg ewp y ; x = x \cdot y + 1 \rrbracket \implies x = y^*$ 

sublocale Sr-algebra ⊆ salomaa-conj-r
by (unfold-locales, metis S12r, metis mult-onel Ar)

lemma (in salomaa-conj-r) ( $\neg ewp y$ ) ∧ ( $x = x \cdot y + z$ ) —>  $x = z \cdot y^*$ 
oops

end

```

## 4 Models of Regular Algebra

```

theory Regular-Algebra-Models
imports Regular-Algebras Kleene-Algebra.Kleene-Algebra-Models
begin

```

### 4.1 Language Model of Salomaa Algebra

```

abbreviation w-length :: 'a list ⇒ nat ( | - | )
where  $|x| \equiv \text{length } x$ 

```

```

definition l-ewp :: 'a lan ⇒ bool where

```

*l-ewp*  $X \longleftrightarrow \{\emptyset\} \subseteq X$

**interpretation lan-kozen:** *K2-algebra* (+) (·) 1 :: 'a lan 0 ( $\subseteq$ ) ( $\subset$ ) star ..

**interpretation lan-boffa:** *B1-algebra* (+) (·) 1 :: 'a lan 0 ( $\subseteq$ ) ( $\subset$ ) star ..

**lemma** *length-lang-pow-lb*:

**assumes**  $\forall x \in X. |x| \geq k \ x \in X^{\hat{n}}$

**shows**  $|x| \geq k * n$

**using assms proof** (induct n arbitrary: x)

**case** 0 **thus** ?case **by** auto

**next**

**case** (Suc n)

**note** hyp = this

**thus** ?case

**proof** –

**have**  $x \in X^{Suc n} \longleftrightarrow (\exists y z. x = y @ z \wedge y \in X \wedge z \in X^n)$

**by** (simp add:c-prod-def times-list-def)

**also from** hyp **have** ...  $\longrightarrow (\exists y z. x = y @ z \wedge |y| \geq k \wedge |z| \geq k * n)$

**by** auto

**also have** ...  $\longleftrightarrow (\exists y z. x = y @ z \wedge |y| \geq k \wedge |z| \geq k * n \wedge (|x| = |y| + |z|))$

**by** force

**also have** ...  $\longleftrightarrow (\exists y z. x = y @ z \wedge |y| \geq k \wedge |z| \geq k * n \wedge (|x| \geq (n + 1) * k))$

**by** (auto, metis add-mono mult.commute, force)

**finally show** ?thesis

**by** (metis Suc-eq-plus1 hyp(3) mult.commute)

**qed**

**qed**

**lemma** *l-prod-elim*:  $w \in X \cdot Y \longleftrightarrow (\exists u v. w = u @ v \wedge u \in X \wedge v \in Y)$

**by** (simp add: c-prod-def times-list-def)

**lemma** *power-minor-var*:

**assumes**  $\forall w \in X. k \leq |w|$

**shows**  $\forall w \in X^{Suc n}. n * k \leq |w|$

**using** assms

**apply** (auto simp add: l-prod-elim)

**using** length-lang-pow-lb trans-le-add2

**by** (simp add: length-lang-pow-lb trans-le-add2 mult.commute)

**lemma** *power-lb*:  $(\forall w \in X. k \leq |w|) \longrightarrow (\forall w. w \in X^{Suc n} \longrightarrow n * k \leq |w|)$

**by** (metis power-minor-var)

**lemma** *prod-lb*:

$\llbracket (\forall w \in X. m \leq \text{length } w); (\forall w \in Y. n \leq \text{length } w) \rrbracket \implies (\forall w \in (X \cdot Y). (m + n) \leq \text{length } w)$

**by** (metis l-prod-elim add-le-mono length-append)

```

lemma suicide-aux-l:
   $\llbracket (\forall w \in Y. 0 \leq \text{length } w); (\forall w \in X^{\text{Suc } n}. n \leq \text{length } w) \rrbracket \implies (\forall w \in X^{\text{Suc } n} . Y. n \leq \text{length } w)$ 
  apply (auto simp: l-prod-elim)
  apply (drule-tac x=ua @ va in bspec)
  apply (auto simp add: l-prod-elim)
done

lemma suicide-aux-r:
   $\llbracket (\forall w \in Y. 0 \leq \text{length } w); (\forall w \in X^{\text{Suc } n}. n \leq \text{length } w) \rrbracket \implies (\forall w \in Y . X^{\text{Suc } n}. n \leq \text{length } w)$ 
  by (auto, metis (full-types) le0 plus-nat.add-0 prod-lb)

lemma word-suicide-l:
  assumes  $\neg \text{l-ewp } X Y \neq \{\}$ 
  shows  $(\forall w \in Y. \exists n. w \notin X^{\text{Suc } n} . Y)$ 
proof -
  have  $\forall v \in Y. 0 \leq \text{length } v$ 
  by (metis le0)
  from assms have  $\forall v \in X. 1 \leq \text{length } v$ 
  by (simp add: l-ewp-def, metis le-0-eq length-0-conv not-less-eq-eq)
  hence  $\forall w \in Y. \exists n. w \notin X^{\text{Suc } n} . Y$ 
  by (metis nat-mult-1-right power-lb suicide-aux-l le0 Suc-n-not-le-n)
  thus ?thesis by metis
qed

lemma word-suicide-r:
  assumes  $\neg \text{l-ewp } X Y \neq \{\}$ 
  shows  $(\forall w \in Y. \exists n. w \notin Y . X^{\text{Suc } n})$ 
proof -
  have  $\forall v \in Y. 0 \leq \text{length } v$ 
  by (metis le0)
  from assms have  $\forall v \in X. 1 \leq \text{length } v$ 
  by (simp add: l-ewp-def, metis le-0-eq length-0-conv not-less-eq-eq)
  hence  $\forall w \in Y. \exists n. w \notin Y . X^{\text{Suc } n}$ 
  by (metis nat-mult-1-right power-lb suicide-aux-r le0 Suc-n-not-le-n)
  thus ?thesis by metis
qed

lemma word-suicide-lang-l:  $\llbracket \neg \text{l-ewp } X; Y \neq \{\} \rrbracket \implies \exists n. \neg (Y \leq X^{\text{Suc } n} . Y)$ 
  by (metis Set.set-eqI empty-iff in-mono word-suicide-l)

lemma word-suicide-lang-r:  $\llbracket \neg \text{l-ewp } X; Y \neq \{\} \rrbracket \implies \exists n. \neg (Y \leq Y . X^{\text{Suc } n})$ 
  by (metis Set.set-eqI empty-iff in-mono word-suicide-r)

```

These duality results cannot be relocated easily

```

context K1-algebra
begin

```

```

lemma power-dual-transfer [simp]:
  power.power (1::'a) (⊖) x n = xn
  by (induct n, simp-all, metis opp-mult-def power-commutes)

lemma aarden-aux-l:
  y ≤ x · y + z ⟹ y ≤ xSuc n · y + x* · z
  using dual.aarden-aux[of y x z n]
  by (auto simp add:opp-mult-def)

end

lemma arden-l:
  assumes ¬ l-ewp y x = y · x + z
  shows x = y* · z
  proof (rule antisym)
    show one: y* · z ≤ x
    by (metis assms(2) join-semilattice-class.add-comm left-near-kleene-algebra-class.star-inductl-eq)
    show x ≤ y* · z
    proof (cases x = 0)
      show x = 0 ⟹ x ≤ y* · z
      by simp
      assume assms': x ≠ 0
      have ⋀ n. x ≤ ySuc n · x + y* · z
      by (metis assms(2) kleene-algebra-class.aarden-aux-l subsetI)
      moreover then have ⋀ w n. w ∈ x ⟹ w ∈ ySuc n · x ∨ w ∈ y* · z
      by (force simp: plus-set-def)
      ultimately show x ≤ y* · z
      by (metis (full-types) all-not-in-conv assms(1) subsetI word-suicide-l)
    qed
  qed

lemma arden-r:
  assumes ¬ l-ewp y x = x · y + z
  shows x = z · y*
  proof (rule antisym)
    show one: z · y* ≤ x
    by (metis assms(2) join.sup-commute kleene-algebra-class.star-inductr-var order-refl)
    show x ≤ z · y*
    proof (cases x = 0)
      show x = 0 ⟹ x ≤ z · y*
      by simp
      assume assms': x ≠ 0
      have ⋀ n. x ≤ x · ySuc n + z · y*
      by (metis assms(2) kleene-algebra-class.aarden-aux subsetI)
      moreover then have ⋀ w n. w ∈ x ⟹ w ∈ x · ySuc n ∨ w ∈ z · y*
      by (force simp: plus-set-def)
      ultimately show x ≤ z · y*
      by (metis (full-types) all-not-in-conv assms(1) subsetI word-suicide-r)

```

qed  
qed

The following two facts provide counterexamples to Arden's rule if the empty word property is not considered.

**lemma arden-l-counter:**  $\exists (x::'a lan) (y::'a lan) (z::'a lan). x = y \cdot x + z \wedge x \neq y^* \cdot z$

**proof –**

- have one:  $(0::'a lan) + 1 \cdot 1 = 1$ 
  - by (metis ab-near-semiring-one-class.mult-onel\_kleene-algebra-class.dual.add-zerol)
- have two:  $(1::'a lan) \neq 1^* \cdot 0$ 
  - proof –
  - have  $\exists x_1. (0::'a list set) \neq x_1$ 
    - by auto
  - hence  $(1::'a list set) \neq 0$ 
    - by (metis kleene-algebra-class.dual.annir\_kleene-algebra-class.dual.mult.right-neutral)
  - thus  $(1::'a list set) \neq 1^* \cdot 0$ 
    - by simp

qed

show ?thesis using one and two

by (metis kleene-algebra-class.dual.add-zerol\_kleene-algebra-class.dual.add-zeror)

qed

**lemma arden-r-counter:**  $\exists (x::'a lan) (y::'a lan) (z::'a lan). x = x \cdot y + z \wedge x \neq z \cdot y^*$

**proof –**

- have one:  $(0::'a lan) + 1 \cdot 1 = 1$ 
  - by (metis ab-near-semiring-one-class.mult-onel\_kleene-algebra-class.dual.add-zerol)
- have two:  $(1::'a lan) \neq 0 \cdot 1^*$ 
  - proof –
  - have  $\exists x_1. (0::'a list set) \neq x_1$ 
    - by auto
  - hence  $(1::'a list set) \neq 0$ 
    - by (metis kleene-algebra-class.dual.annir\_kleene-algebra-class.dual.mult.right-neutral)
  - thus  $(1::'a list set) \neq 0 \cdot 1^*$ 
    - by simp

qed

show ?thesis using one and two

by (metis kleene-algebra-class.dual.add-zerol\_kleene-algebra-class.dual.add-zeror)

qed

**interpretation lan-salomaa-l:** Sl-algebra (+) (·) 1 :: 'a lan 0 ( $\subseteq$ ) ( $\subset$ ) star l-ewp

**proof**

fix x y z :: 'a lan

show  $(1 + x)^* = x^*$

by (metis kleene-algebra-class.dual.star2)

show l-ewp x =  $(\exists y. x = 1 + y \wedge \neg l\text{-ewp } y)$

by (simp add:l-ewp-def one-set-def one-list-def plus-set-def, metis insertCI mk-disjoint-insert)

```

show 1 + x · x* = x*
  by (metis kleene-algebra-class.star-unfoldl-eq)
show ¬ l-ewp y ⇒ x = y · x + z ⇒ x = y* · z
  by (metis arden-l)
qed

interpretation lan-salomaa-r: Sr-algebra (+) (·) 1 :: 'a lan 0 (⊆) (⊂) star l-ewp
proof
  fix x y z :: 'a lan
  show 1 + x* · x = x*
    by (metis kleene-algebra-class.star-unfoldr-eq)
  show ¬ l-ewp y ⇒ x = x · y + z ⇒ x = z · y*
    by (metis arden-r)
qed

```

## 4.2 Regular Language Model of Salomaa Algebra

**notation**

*Atom* ( $\langle \langle - \rangle \rangle$ ) **and**  
*Plus* (**infixl**  $\langle +_r \rangle$  65) **and**  
*Times* (**infixl**  $\langle \cdot_r \rangle$  70) **and**  
*Star* ( $\langle \cdot^*_r \rangle$  [101] 100) **and**  
*Zero* ( $\langle 0_r \rangle$ ) **and**  
*One* ( $\langle 1_r \rangle$ )

```

fun rexp-ewp :: 'a rexprp ⇒ bool where
  rexpr-ewp 0r = False |
  rexpr-ewp 1r = True |
  rexpr-ewp ⟨a⟩ = False |
  rexpr-ewp (s +r t) = (rexpr-ewp s ∨ rexpr-ewp t) |
  rexpr-ewp (s ·r t) = (rexpr-ewp s ∧ rexpr-ewp t) |
  rexpr-ewp (s*r) = True

```

**abbreviation** ro(s) ≡ (if (rexpr-ewp s) then 1<sub>r</sub> else 0<sub>r</sub>)

**lift-definition** r-ewp :: 'a reg-lan ⇒ bool **is** l-ewp .

**lift-definition** r-lang :: 'a rexprp ⇒ 'a reg-lan **is** lang  
**by** (simp)

**abbreviation** r-sim :: 'a rexprp ⇒ 'a rexprp ⇒ bool (**infix**  $\langle \sim \rangle$  50) **where**  
 $p \sim q \equiv r\text{-lang } p = r\text{-lang } q$

```

declare Rep-reg-lan [simp]
declare Rep-reg-lan-inverse [simp]
declare Abs-reg-lan-inverse [simp]

```

**lemma** rexpr-ewp-l-ewp: l-ewp (lang x) = rexpr-ewp x  
**proof** (induct x)

```

case (Star x) thus ?case
  by (simp, metis lan-salomaa-l.EWP left-near-kleene-algebra-class.star-plus-one)
qed (simp-all add:l-ewp-def zero-set-def one-set-def one-list-def plus-set-def c-prod-def
times-list-def)

theorem regexp-ewp:
defines P-def: P(t)  $\equiv \exists t'. t \sim ro(t) +_r t' \wedge ro(t') = 0_r$ 
shows P t
proof (induct t)
show P(0r)
  by (simp add:P-def r-lang-def, rule-tac x=0r in exI, simp)
next
fix a
show P(⟨a⟩)
  by (simp add:P-def r-lang-def, rule-tac x=⟨a⟩ in exI, simp)
next
show P(1r)
  by (simp add:P-def r-lang-def, rule-tac x=0r in exI, simp)
next
fix t1 t2
assume P(t1) P(t2)
then obtain t1' t2'
  where t1  $\sim ro(t1) +_r t1'  $ro(t1') = 0_r$ 
        t2  $\sim ro(t2) +_r t2'  $ro(t2') = 0_r$ 
  by (metis assms rexp.distinct(1))
thus P(t1 +r t2)
  apply (subst P-def, transfer)
  apply (rule-tac x=t1' +r t2' in exI)
  apply clarsimp
  by (metis (no-types, lifting) add.left-commute join.sup-assoc join.sup-left-idem
rexp.distinct(2))
next
fix t1 t2
assume P(t1) P(t2)
then obtain t1' t2'
  where t1: t1  $\sim ro(t1) +_r t1'  $ro(t1') = 0_r$  and
        t2: t2  $\sim ro(t2) +_r t2'  $ro(t2') = 0_r$ 
  by (metis assms rexp.distinct(1))
thus P(t1 ·r t2)
proof -
  let ?t' = ro(t1) ·r t2' +r t1' ·r ro(t2) +r t1' ·r t2'
  from t1 t2 have r1: ro(?t') = 0r
  by (auto)
  from t1 t2 have t1 ·r t2  $\sim (ro(t1) +_r t1') \cdot_r (ro(t2) +_r t2')$  (is ?l  $\sim$  ?r)
  by (transfer, simp)
  also have ?r  $\sim ro(t1) \cdot_r ro(t2) +_r ro(t1) \cdot_r t2' +_r t1' \cdot_r ro(t2) +_r t1' \cdot_r t2'$ 
(is ?l  $\sim$  ?r)
  apply (transfer, unfold lang.simps)
  apply (simp only: distrib-right' semiring-class.distrib-left)$$$$ 
```

```

apply (metis (opaque-lifting, no-types) join-semilattice-class.add-comm semiring-class.combine-common-factor)
done
also have ?r ~ ro(t1 ·r t2) +r ro(t1) ·r t2' +r t1' ·r ro(t2) +r t1' ·r t2' (is
?l ~ ?r)
  by (transfer, simp)
also have ?r ~ ro(t1 ·r t2) +r ?t'
  apply (transfer, unfold lang.simps)
  apply (metis (mono-tags) join-semilattice-class.add-assoc')
done
finally show ?thesis using r1
  apply (unfold P-def)
  apply (rule-tac x=?t' in exI, simp)
done
qed
next
fix s
assume assm:P s
then obtain s' where r1: s ~ ro(s) +r s' ro(s') = 0r
  by (metis assms rexp.distinct(1))
thus P (s*r)
proof -
  let ?t' = s' ·r (s')*r
  have r2: ro(?t') = 0r
    by (metis r1(2) rexp.distinct(1) rexp-ewp.simps(5))
  from assm r1 have (ro(s) +r s')*r ~ (s')*r (is ?l ~ ?r)
    by (transfer, auto)
  also have ?r ~ 1r +r s' ·r (s')*r (is ?l ~ ?r)
    by (transfer, auto)
  also have ?r ~ ro(s*r) +r ?t'
    by (metis (full-types) rexp-ewp.simps(6))
  finally show ?thesis
    by (metis assms lang.simps(6) r1(1) r2 r-lang.abs-eq r-lang.rep-eq)
qed
qed

instantiation reg-lan :: (type) Sr-algebra
begin

lift-definition ewp-reg-lan :: 'a reg-lan ⇒ bool is l-ewp .

instance proof
fix x :: 'a reg-lan
show (1 + x)* = x*
  by (metis kleene-algebra-class.dual.star2)
next
fix x :: 'a reg-lan
show 1 + x* · x = x*
  by (metis kleene-algebra-class.star-unfoldr-eq)

```

```

next
  fix  $x :: 'a reg-lan$ 
  show  $\text{ewp } x = (\exists y. x = 1 + y \wedge \neg \text{ewp } y)$ 
  proof -
    obtain  $t$  where  $r\text{-lang } t = x$ 
    by (transfer, auto)
    moreover obtain  $t'$  where  $t \sim ro(t) +_r t' ro(t') = 0_r$ 
    by (metis regexp-ewp)
    ultimately show ?thesis
    apply (transfer, auto)
    apply (metis (full-types) lang.simps(2) rexp.distinct(1) rexp-ewp-l-ewp)
    apply (metis lan-salomaa-l.EWP)
    done
  qed
next
  fix  $x y z :: 'a reg-lan$ 
  show  $\llbracket \neg \text{ewp } y; x = x \cdot y + z \rrbracket \implies x = z \cdot y^*$ 
  by (transfer, metis lan-salomaa-r.Ar)
qed
end

instantiation reg-lan :: (type) Sl-algebra
begin

  instance proof
    fix  $x :: 'a reg-lan$ 
    show  $1 + x \cdot x^* = x^*$ 
    by (metis left-pre-kleene-algebra-class.star-unfoldl-eq)
  next
    fix  $x y z :: 'a reg-lan$ 
    show  $\llbracket \neg \text{ewp } y; x = y \cdot x + z \rrbracket \implies x = y^* \cdot z$ 
    by (transfer, metis lan-salomaa-l.Al)
  qed
end

instance reg-lan :: (type) S-algebra ..

theorem arden-regexp-l:
  assumes  $ro(y) = 0_r$   $x \sim y \cdot_r x +_r z$ 
  shows  $x \sim y^* \cdot_r z$ 
  using assms
  by (transfer, metis arden-l lang.simps(4) lang.simps(5) lang.simps(6) rexp.distinct(1) rexp-ewp-l-ewp)

theorem arden-regexp-r:
  assumes  $ro(y) = 0_r$   $x \sim x \cdot_r y +_r z$ 
  shows  $x \sim z \cdot_r y^*$ 
  using assms
  by transfer (metis lan-salomaa-r.Ar lang.simps(4) lang.simps(5) lang.simps(6))

```

```
rexp.distinct(1) rexp-ewp-l-ewp)
```

```
end
```

## 5 Pratt's Counterexamples

```
theory Pratts-Counterexamples
  imports Regular-Algebras
begin
```

We create two regular algebra models due to Pratt [8] which are used to distinguish K1 algebras from K1l and K1r algebras.

```
datatype pratt1 =
  P1Bot (<⊥1>) |
  P1Nat nat (<[-]1>) |
  P1Infty (<∞1>) |
  P1Top (<⊤1>)
```

```
datatype pratt2 =
  P2Bot (<⊥2>) |
  P2Nat nat (<[-]2>) |
  P2Infty (<∞2>) |
  P2Top (<⊤2>)
```

```
fun pratt1-max where
  pratt1-max [x]1 [y]1 = [max x y]1 |
  pratt1-max x ⊥1 = x |
  pratt1-max ⊥1 y = y |
  pratt1-max ∞1 [y]1 = ∞1 |
  pratt1-max [y]1 ∞1 = ∞1 |
  pratt1-max ∞1 ∞1 = ∞1 |
  pratt1-max ⊤1 x = ⊤1 |
  pratt1-max x ⊤1 = ⊤1
```

```
fun pratt1-plus :: pratt1 ⇒ pratt1 ⇒ pratt1 (infixl <+_1> 65) where
  [x]1 +1 [y]1 = [x + y]1 |
  ⊥1 +1 x = ⊥1 |
  x +1 ⊥1 = ⊥1 |
  ∞1 +1 [0]1 = ∞1 |
  [0]1 +1 ∞1 = ∞1 |
  ∞1 +1 ∞1 = ⊤1 |
  ⊤1 +1 x = ⊤1 |
  x +1 ⊤1 = ⊤1 |
  [x]1 +1 ∞1 = ∞1 |
  ∞1 +1 x = ⊤1
```

```
lemma plusl-top-infty: ⊤1 +1 ∞1 = ⊤1
  by (simp)
```

```

lemma plusl-infty-top:  $\infty_1 +_1 \top_1 = \top_1$ 
by (simp)

lemma plusl-bot-infty:  $\perp_1 +_1 \infty_1 = \perp_1$ 
by (simp)

lemma plusl-infty-bot:  $\infty_1 +_1 \perp_1 = \perp_1$ 
by (simp)

lemma plusl-zero-infty:  $[0]_1 +_1 \infty_1 = \infty_1$ 
by (simp)

lemma plusl-infty-zero:  $\infty_1 +_1 [0]_1 = \infty_1$ 
by (simp)

lemma plusl-infty-num [simp]:  $x > 0 \implies \infty_1 +_1 [x]_1 = \top_1$ 
by (case-tac x, simp-all)

lemma plusl-num-infty [simp]:  $x > 0 \implies [x]_1 +_1 \infty_1 = \infty_1$ 
by (case-tac x, simp-all)

fun pratt2-max where
  pratt2-max  $[x]_2 [y]_2 = [\max x y]_2$  |
  pratt2-max  $x \perp_2 = x$  |
  pratt2-max  $\perp_2 y = y$  |
  pratt2-max  $\infty_2 [y]_2 = \infty_2$  |
  pratt2-max  $[y]_2 \infty_2 = \infty_2$  |
  pratt2-max  $\infty_2 \infty_2 = \infty_2$  |
  pratt2-max  $\top_2 x = \top_2$  |
  pratt2-max  $x \top_2 = \top_2$ 

fun pratt2-plus :: pratt2  $\Rightarrow$  pratt2  $\Rightarrow$  pratt2 (infixl  $\langle +_2 \rangle$  65) where
   $[x]_2 +_2 [y]_2 = [x + y]_2$  |
   $\perp_2 +_2 x = \perp_2$  |
   $x +_2 \perp_2 = \perp_2$  |
   $\infty_2 +_2 \infty_2 = \infty_2$  |
   $[0]_2 +_2 \infty_2 = \infty_2$  |
   $\infty_2 +_2 \infty_2 = \top_2$  |
   $\top_2 +_2 x = \top_2$  |
   $x +_2 \top_2 = \top_2$  |
   $x +_2 \infty_2 = \top_2$  |
   $\infty_2 +_2 [x]_2 = \infty_2$ 

instantiation pratt1 :: selective-semiring
begin

definition zero-pratt1-def:
   $0 \equiv \perp_1$ 

```

```

definition one-pratt1-def:
  1 ≡ [0]₁

definition plus-pratt1-def:
  x + y ≡ pratt1-max x y

definition times-pratt1-def:
  x * y ≡ x +₁ y

definition less-eq-pratt1-def:
  (x::pratt1) ≤ y ≡ x + y = y

definition less-pratt1-def:
  (x::pratt1) < y ≡ x ≤ y ∧ x ≠ y

instance
proof
  fix x y z :: pratt1
  show x + y + z = x + (y + z)
    by (case-tac x, case-tac[!] y, case-tac[!] z, simp-all add: plus-pratt1-def)
  show x + y = y + x
    by (cases x, case-tac[!] y, simp-all add: plus-pratt1-def)
  show x * y * z = x * (y * z)
    apply (cases x, case-tac[!] y, case-tac[!] z, simp-all add: plus-pratt1-def
times-pratt1-def)
    apply (rename-tac[!] n, case-tac[!] n, auto)
    apply (metis nat.exhaust plusl-zero-infty pratt1-plus.simps(14))
    apply (metis not0-implies-Suc plusl-infty-zero plusl-zero-infty pratt1-plus.simps(14))
    apply (metis neq0-conv plusl-num-infty plusl-zero-infty pratt1-plus.simps(15))
    apply (metis not0-implies-Suc plusl-infty-zero pratt1-plus.simps(15) pratt1-plus.simps(9))
    apply (metis not0-implies-Suc plusl-infty-zero pratt1-plus.simps(15) pratt1-plus.simps(9))
  done
  show (x + y) * z = x * z + y * z
    apply (cases x, case-tac[!] y, case-tac[!] z, simp-all add: plus-pratt1-def
times-pratt1-def)
    apply (rename-tac[!] n, case-tac[!] n, auto)
    apply (metis neq0-conv plusl-num-infty plusl-zero-infty pratt1-max.simps(8))+
  done
  show 1 * x = x
    by (cases x, simp-all add: one-pratt1-def times-pratt1-def)
  show x * 1 = x
    by (cases x, simp-all add: one-pratt1-def times-pratt1-def)
  show 0 + x = x
    by (cases x, simp-all add: plus-pratt1-def zero-pratt1-def)
  show 0 * x = 0
    by (cases x, simp-all add: times-pratt1-def zero-pratt1-def)
  show x * 0 = 0
    by (cases x, simp-all add: times-pratt1-def zero-pratt1-def)
  show x ≤ y ↔ x + y = y

```

```

by (metis less-eq-pratt1-def)
show x < y  $\longleftrightarrow$  x ≤ y ∧ x ≠ y
  by (metis less-pratt1-def)
show x + y = x ∨ x + y = y
  by (cases x, case-tac[!] y, simp-all add: plus-pratt1-def max-def)
show x * (y + z) = x * y + x * z
  apply (cases x, case-tac[!] y, case-tac[!] z, simp-all add: plus-pratt1-def
times-pratt1-def)
    apply (rename-tac[!] n, case-tac[!] n, auto)
  apply (metis nat.exhaust plusl-zero-infty pratt1-max.simps(7) pratt1-plus.simps(14))
    apply (metis neq0-conv plusl-num-infty plusl-zero-infty pratt1-max.simps(7))
  apply (metis nat.exhaust plusl-zero-infty pratt1-max.simps(6) pratt1-plus.simps(14))
    apply (metis neq0-conv plusl-num-infty plusl-zero-infty pratt1-max.simps(6))
    apply (metis neq0-conv plusl-infty-num plusl-infty-zero pratt1-max.simps(10)
pratt1-max.simps(8))
    apply (metis not0-implies-Suc plusl-infty-zero pratt1-max.simps(11) pratt1-max.simps(13)
pratt1-plus.simps(15)))
  done
qed
end

instance pratt1 :: dioid-one-zero ..

instantiation pratt2 :: selective-semiring
begin

definition zero-pratt2-def:
  0 ≡ ⊥2

definition one-pratt2-def:
  1 ≡ [0]2

definition times-pratt2-def:
  x * y ≡ x +2 y

definition plus-pratt2 :: pratt2 ⇒ pratt2 ⇒ pratt2 where
  plus-pratt2 x y ≡ pratt2-max x y

definition less-eq-pratt2-def:
  (x::pratt2) ≤ y ≡ x + y = y

definition less-pratt2-def:
  (x::pratt2) < y ≡ x ≤ y ∧ x ≠ y

instance
proof
fix x y z :: pratt2
show x + y + z = x + (y + z)
  by (case-tac x, case-tac[!] y, case-tac[!] z, simp-all add: plus-pratt2-def)

```

```

show  $x + y = y + x$ 
  by (cases  $x$ , case-tac[!]  $y$ , simp-all add: plus-pratt2-def)
show  $x * y * z = x * (y * z)$ 
  apply (cases  $x$ , case-tac[!]  $y$ , case-tac[!]  $z$ , auto simp add: times-pratt2-def)
  apply (rename-tac[!]  $n$ , case-tac[!]  $n$ , auto)
    apply (metis not0-implies-Suc pratt2-plus.simps(14) pratt2-plus.simps(6)
pratt2-plus.simps(7) pratt2-plus.simps(9))
    apply (metis not0-implies-Suc pratt2-plus.simps(14) pratt2-plus.simps(15)
pratt2-plus.simps(7) pratt2-plus.simps(9))
    apply (metis not0-implies-Suc pratt2-plus.simps(15) pratt2-plus.simps(6))
    apply (metis not0-implies-Suc pratt2-plus.simps(15) pratt2-plus.simps(6))
done
show  $(x + y) * z = x * z + y * z$ 
  apply (cases  $x$ , case-tac[!]  $y$ , case-tac[!]  $z$ , simp-all add: plus-pratt2-def
times-pratt2-def)
  apply (rename-tac[!]  $n$ , case-tac[!]  $n$ , auto)
    apply (metis not0-implies-Suc pratt2-max.simps(10) pratt2-max.simps(8)
pratt2-plus.simps(14) pratt2-plus.simps(7))
    apply (metis max-0L max-0R max.left-idem nat.exhaust pratt2-max.simps(11)

pratt2-max.simps(13) pratt2-plus.simps(14) pratt2-plus.simps(7))
done
show  $1 * x = x$ 
  by (cases  $x$ , simp-all add: one-pratt2-def times-pratt2-def)
show  $x * 1 = x$ 
  by (cases  $x$ , simp-all add: one-pratt2-def times-pratt2-def)
show  $0 + x = x$ 
  by (cases  $x$ , simp-all add: plus-pratt2-def zero-pratt2-def)
show  $0 * x = 0$ 
  by (cases  $x$ , simp-all add: times-pratt2-def zero-pratt2-def)
show  $x * 0 = 0$ 
  by (cases  $x$ , simp-all add: times-pratt2-def zero-pratt2-def)
show  $x \leq y \longleftrightarrow x + y = y$ 
  by (metis less-eq-pratt2-def)
show  $x < y \longleftrightarrow x \leq y \wedge x \neq y$ 
  by (metis less-pratt2-def)
show  $x + y = x \vee x + y = y$ 
  by (cases  $x$ , case-tac[!]  $y$ , simp-all add: plus-pratt2-def max-def)
show  $x * (y + z) = x * y + x * z$ 
  apply (cases  $x$ , case-tac[!]  $y$ , case-tac[!]  $z$ , simp-all add: plus-pratt2-def
times-pratt2-def)
  apply (rename-tac[!]  $n$ , case-tac[!]  $n$ , auto)
    apply (metis not0-implies-Suc pratt2-max.simps(12) pratt2-max.simps(7)
pratt2-plus.simps(14) pratt2-plus.simps(7))
    apply (metis nat.exhaust pratt2-max.simps(12) pratt2-max.simps(7) pratt2-plus.simps(14)
pratt2-plus.simps(7))
    apply (metis not0-implies-Suc pratt2-max.simps(6) pratt2-max.simps(9)
pratt2-plus.simps(14) pratt2-plus.simps(7))
    apply (metis nat.exhaust pratt2-max.simps(6) pratt2-max.simps(9) pratt2-plus.simps(14)
pratt2-plus.simps(7))

```

```

pratt2-plus.simps(7)
  apply (metis not0-implies-Suc pratt2-max.simps(8) pratt2-plus.simps(15)
pratt2-plus.simps(6))
  apply (metis not0-implies-Suc pratt2-max.simps(8) pratt2-plus.simps(15)
pratt2-plus.simps(6))
    done
  qed
end

lemma top-greatest:  $x \leq \top_1$ 
  by (case-tac x, auto simp add: less-eq-pratt1-def plus-pratt1-def)

instantiation pratt1 :: star-op
begin

definition star-pratt1 where
   $x^* \equiv \text{if } (x = \perp_1 \vee x = [0]_1) \text{ then } [0]_1 \text{ else } \top_1$ 
  instance ..
end

instantiation pratt2 :: star-op
begin

definition star-pratt2 where
   $x^* \equiv \text{if } (x = \perp_2 \vee x = [0]_2) \text{ then } [0]_2 \text{ else } \top_2$ 
  instance ..
end

instance pratt1 :: K1r-algebra
proof
  fix x :: pratt1
  show  $1 + x^* \cdot x \leq x^*$ 
    by (case-tac x, auto simp add: star-pratt1-def one-pratt1-def times-pratt1-def
less-eq-pratt1-def plus-pratt1-def)
  next
    fix x y :: pratt1
    show  $y \cdot x \leq y \implies y \cdot x^* \leq y$ 
      apply (case-tac x, case-tac[!] y)
      apply (auto simp add: star-pratt1-def one-pratt1-def times-pratt1-def less-eq-pratt1-def
plus-pratt1-def)
      apply (rename-tac n, case-tac n)
      apply (simp-all)
      by (metis not0-implies-Suc plusl-zero-infty pratt1.distinct(7) pratt1-max.simps(6)
pratt1-plus.simps(14))
  qed

instance pratt2 :: K1l-algebra
proof
  fix x :: pratt2
  show  $1 + x \cdot x^* \leq x^*$ 

```

```

by (case-tac x, auto simp add: star-pratt2-def one-pratt2-def times-pratt2-def
less-eq-pratt2-def plus-pratt2-def)
next
  fix x y :: pratt2
  show x · y ≤ y  $\implies$  x* · y ≤ y
    apply (case-tac x, case-tac[!] y)
    apply (auto simp add:star-pratt2-def one-pratt2-def times-pratt2-def less-eq-pratt2-def
plus-pratt2-def)
    apply (rename-tac n, case-tac n)
    apply (simp-all)
    apply (rename-tac n, case-tac n)
    apply (auto simp add:star-pratt2-def one-pratt2-def times-pratt2-def less-eq-pratt2-def
plus-pratt2-def)
    done
  qed

lemma one-star-top: [1]1* = ⊤1
  by (simp add:star-pratt1-def one-pratt1-def)

lemma pratt1-kozen-1l-counterexample:
   $\exists x y :: \text{pratt1}. \neg (x \cdot y \leq y \wedge x^* \cdot y \leq y)$ 
proof –
  have  $\neg ([1]_1 \cdot \infty_1 \leq \infty_1 \wedge [1]_1^* \cdot \infty_1 \leq \infty_1)$ 
  by (auto simp add: star-pratt1-def times-pratt1-def less-eq-pratt1-def plus-pratt1-def)
  thus ?thesis
    by auto
  qed

lemma pratt2-kozen-1r-counterexample:
   $\exists x y :: \text{pratt2}. \neg (y \cdot x \leq y \wedge y \cdot x^* \leq y)$ 
proof –
  have  $\neg (\infty_2 \cdot [1]_2 \leq \infty_2 \wedge \infty_2 \cdot [1]_2^* \leq \infty_2)$ 
  by (auto simp add: star-pratt2-def times-pratt2-def less-eq-pratt2-def plus-pratt2-def)
  thus ?thesis
    by auto
  qed

end

```

## 6 Variants of Regular Algebra

```

theory Regular-Algebra-Variants
  imports Regular-Algebras Pratts-Counterexamples
  begin

```

Replacing Kozen's induction axioms by Boffa's leads to incompleteness.

```

lemma (in star-doid)
  assumes  $\bigwedge x. 1 + x \cdot x^* = x^*$ 
  and  $\bigwedge x. 1 + x^* \cdot x = x^*$ 

```

**and**  $\bigwedge x. x \cdot x = x \implies x^* = 1 + x$   
**shows**  $\bigwedge x y. (x + y)^* = x^* \cdot (y \cdot x^*)^*$

**oops**

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x. x \cdot x = x \implies x^* = 1 + x$   
**shows**  $\bigwedge x y. x \leq y \longrightarrow x^* \leq y^*$

**oops**

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x y. 1 + x \leq y \wedge y \cdot y \leq y \longrightarrow x^* \leq y$   
**shows**  $\bigwedge x. 1 + x \leq x^*$   
**by** (*metis add-iso-r assms(1) mult-oner subdistl*)

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x y. 1 + x \leq y \wedge y \cdot y \leq y \longrightarrow x^* \leq y$   
**shows**  $\bigwedge x. x^* = (1 + x)^*$   
**oops** — need to reconsider this

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x y. 1 + x + y \cdot y \leq y \implies x^* \leq y$   
**shows**  $\bigwedge x. x^* \cdot x^* \leq x^*$   
**oops**

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x y z. x \cdot y = y \cdot z \implies x^* \cdot y = y \cdot z^*$   
**shows**  $1^* = 1$

**oops**

**lemma (in star-diod)**  
**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$   
**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$   
**and**  $\bigwedge x y z. x \cdot y \leq y \cdot z \implies x^* \cdot y \leq y \cdot z^*$   
**and**  $\bigwedge x y z. y \cdot x \leq z \cdot y \implies y \cdot x^* \leq z^* \cdot y$   
**shows**  $1^* = 1$

**oops**

**lemma (in star-diooid)**

**assumes**  $\bigwedge x. 1 + x \cdot x^* = x^*$

**and**  $\bigwedge x. 1 + x^* \cdot x = x^*$

**and**  $\bigwedge x y. x = y \cdot x \implies x = y^* \cdot x$

**and**  $\bigwedge x y. x = x \cdot y \implies x = x \cdot y^*$

**shows**  $\bigwedge x y. y \cdot x \leq y \implies y \cdot x^* \leq y$

**oops**

**class**  $C3l\text{-var} = \text{conway-diooid} +$

**assumes**  $C3l\text{-var}: z + x \cdot y \leq y \implies x^* \cdot z \leq y$

**class**  $C3r\text{-var} = \text{conway-diooid} +$

**assumes**  $C3r\text{-var}: z + y \cdot x \leq y \implies z \cdot x^* \leq y$

**class**  $C3\text{-var} = C3l\text{-var} + C3r\text{-var}$

**sublocale**  $C3l\text{-var} \subseteq C3l\text{-algebra}$

**apply** unfold-locales

**by** (simp add: local. $C3l\text{-var}$ )

**sublocale**  $C3l\text{-algebra} \subseteq C3l\text{-var}$

**by** (unfold-locales, metis star-inductl-var)

**sublocale**  $C3\text{-var} \subseteq C3\text{-algebra}$

**apply** unfold-locales

**by** (simp add: local. $C3r\text{-var}$ )

**sublocale**  $C3\text{-algebra} \subseteq C3\text{-var}$

**by** (unfold-locales, metis star-inductr-var)

**class**  $Brtc\text{-algebra} = \text{star-diooid} +$

**assumes**  $rtc1: 1 + x^* \cdot x^* + x \leq x^*$

**and**  $rtc2: 1 + x + y \cdot y \leq y \implies x^* \leq y$

**sublocale**  $B2\text{-algebra} \subseteq Brtc\text{-algebra}$

**proof**

**fix**  $x y$

**show**  $1 + x^* \cdot x^* + x \leq x^*$

**by** (simp add: local.star-ext)

**show**  $1 + x + y \cdot y \leq y \implies x^* \leq y$

**by** (metis B23 local.join.le-sup-iff order.eq-iff mult-1-right subdistl)

**qed**

**sublocale**  $Brtc\text{-algebra} \subseteq B2\text{-algebra}$

**proof**

**fix**  $x y$

**show**  $1 + x \leq x^*$

```

by (metis rtc1 join.le-sup-iff)
show  $x^* \cdot x^* = x^*$ 
proof (rule order.antisym)
show  $x^* \cdot x^* \leq x^*$ 
by (metis rtc1 join.le-sup-iff)
show  $x^* \leq x^* \cdot x^*$ 
by (metis rtc1 add.commute join.le-sup-iff less-eq-def mult-isor mult-onel)
qed
show  $\llbracket 1 + x \leq y; y \cdot y = y \rrbracket \implies x^* \leq y$ 
by (metis rtc2 eq-refl less-eq-def)
qed

class wB1-algebra = conway-dioid +
assumes wR:  $x \cdot x \leq x \implies x^* = 1 + x$ 

sublocale wB1-algebra ⊆ B1-algebra
by (unfold-locales, metis order-refl wR)

lemma (in B1-algebra) one-plus-star:  $x^* = (1 + x)^*$ 
by (metis C11-var R add-idem' mult-onel mult-oner)

sublocale B1-algebra ⊆ wB1-algebra
proof unfold-locales
fix x :: 'a
assume  $x \cdot x \leq x$ 
hence  $x \cdot (1 + x) = x$ 
by (simp add: local.distrib-left local.join.sup-absorb1)
hence  $(1 + x) \cdot (1 + x) = 1 + x$ 
using add.left-commute distrib-right' by simp
thus  $x^* = 1 + x$ 
by (metis R add-assoc' add-idem' one-plus-star)
qed

end

```

## References

- [1] A. Armstrong, G. Struth, and T. Weber. Kleene algebra. *Archive of Formal Proofs*, 2013. [http://isa-afp.org/entries/Kleene\\_Algebra.shtml](http://isa-afp.org/entries/Kleene_Algebra.shtml), Formal proof development.
- [2] M. Boffa. Une remarque sur les systèmes complets d’identités rationnelles. *Informatique théorique et applications*, 24(4):419–423, 1990.
- [3] M. Boffa. Une condition impliquant toutes les identités rationnelles. *Informatique théorique et applications*, 29(6):515–518, 1995.

- [4] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.
- [5] S. Foster and G. Struth. On the fine-structure of regular algebra. *J. Automated Reasoning*, 54(2):165–197, 2015.
- [6] D. Kozen. On Kleene algebras and closed semirings. In B. Rovan, editor, *MFCS'90*, volume 452 of *LNCS*, pages 26–47. Springer, 1990.
- [7] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.
- [8] V. Pratt. Action logic and pure induction. Technical Report STAN-CS-90-1343, Department of Computer Science, Stanford University, 1990.
- [9] A. Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.