A Complete Proof of the Robbins Conjecture

Matthew Wampler-Doty

August 16, 2018

Abstract

Contents

1 Robbins Conjecture 1
2 Axiom Systems 1
  2.1 Common Algebras 2
  2.2 Boolean Algebra 2
  2.3 Huntington’s Algebra 3
  2.4 Robbins’ Algebra 3

3 Equivalence 3
  3.1 Boolean Algebra 3
  3.2 Huntington Algebra 4
  3.3 Robbins’ Algebra 9

1 Robbins Conjecture

theory Robbins-Conjecture
imports Main
begin

  The document gives a formalization of the proof of the Robbins conjecture, following A. Mann, A Complete Proof of the Robbins Conjecture, 2003, DOI 10.1.1.6.7838

2 Axiom Systems

The following presents several axiom systems that shall be under study.
The first axiom sets common systems that underly all of the systems we shall be looking at.

The second system is a reformulation of Boolean algebra. We shall follow pages 7–8 in S. Koppelberg, General Theory of Boolean Algebras, Volume 1 of Handbook of Boolean Algebras. North Holland, 1989. Note that our formulation deviates slightly from this, as we only provide one distribution axiom, as the dual is redundant.

The third system is Huntington’s algebra and the fourth system is Robbins’ algebra.

Apart from the common system, all of these systems are demonstrated to be equivalent to the library formulation of Boolean algebra, under appropriate interpretation.

2.1 Common Algebras

class common-algebra = uminus +
fixes inf : 'a => 'a => 'a (infixl \( \cap \))
fixes sup : 'a => 'a => 'a (infixl \( \cup \))
fixes bot : 'a (\( \bot \))
fixes top : 'a (\( \top \))
assumes sup-assoc: x \( \cup \) (y \( \cup \) z) = (x \( \cup \) y) \( \cup \) z
assumes sup-comm: x \( \cup \) y = y \( \cup \) x

definition less-eq :: 'a => 'a => bool (infix \( \leq \)) where
  x \( \leq \) y = (x \( \cup \) y = y)
definition less :: 'a => 'a => bool (infix \( \lt \)) where
  x \( \lt \) y = (x \( \leq \) y \( \land \) \neg y \( \in \) x)
definition minus :: 'a => 'a => 'a (infix - 65) where
  minus x y = (x \( \cap \) - y)
definition secret-object1 :: 'a (\( \iota \)) where
  \( \iota \) = (SOME x. True)
end

class ext-common-algebra = common-algebra +
  assumes inf-eq: x \( \cap \) y = -(x \( \cup \) - y)
  assumes top-eq: \( \top \) = \( \iota \) \( \cup \) - \( \iota \)
  assumes bot-eq: \( \bot \) = -(\( \iota \) \( \cup \) - \( \iota \))

2.2 Boolean Algebra

class boolean-algebra-II =
  common-algebra +
  assumes inf-comm: x \( \cap \) y = y \( \cap \) x

2
assumes inf-assoc: \( x \cap (y \cap z) = (x \cap y) \cap z \)
assumes sup-absorb: \( x \cup (x \cap y) = x \)
assumes inf-absorb: \( x \cap (x \cup y) = x \)
assumes sup-inf-distrib1: \( x \cup y \cap z = (x \cup y) \cap (x \cup z) \)
assumes sup-compl: \( x \cup -x = \top \)
assumes inf-compl: \( x \cap -x = \bot \)

2.3 Huntington’s Algebra

class huntington-algebra = ext-common-algebra +
assumes huntington: \(-(-x \cup -y) \cup (-x \cup y) = x\)

2.4 Robbins’ Algebra

class robbins-algebra = ext-common-algebra +
assumes robbins: \(-((-x \cup y) \cup (-x \cup -y)) = x\)

3 Equivalence

With our axiom systems defined, we turn to providing equivalence results between them.

We shall begin by illustrating equivalence for our formulation and the library formulation of Boolean algebra.

3.1 Boolean Algebra

The following provides the canonical definitions for order and relative complementation for Boolean algebras. These are necessary since the Boolean algebras presented in the Isabelle/HOL library have a lot of structure, while our formulation is considerably simpler.

Since our formulation of Boolean algebras is considerably simple, it is easy to show that the library instantiates our axioms.

context boolean-algebra-II begin

lemma boolean-II-is-boolean:
class boolean-algebra minus uminus (\( \cap \)) (\( \cup \)) (\( \subseteq \)) (\( \supseteq \)) (\( \bot \)) (\( \top \))
apply unfold-locales
apply (metis inf-absorb inf-assoc inf-comm inf-compl
less-def less-eq-def minus-def
sup-absorb sup-assoc sup-comm
sup-compl sup-inf-distrib1
sup-absorb inf-comm)+
done

end

context boolean-algebra begin


3.2 Huntington Algebra

We shall illustrate here that all Boolean algebra using our formulation are Huntington algebras, and illustrate that every Huntington algebra may be interpreted as a Boolean algebra.

Since the Isabelle/HOL library has good automation, it is convenient to first show that the library instances Huntington algebras to exploit previous results, and then use our previously derived correspondence.

context boolean-algebra begin
lemma boolean-is-huntington:
  class.huntington-algebra uminus inf sup bot top
apply unfold-locales
apply (metis double-compl inf-sup-distrib1 inf-top-right
     compl-inf inf-commute inf-sup-absorb inf-compl-top
     sup-compl-top sup-assoc)+
done
end

class boolean-algebra-II uminus inf sup bot top
lemma boolean-II-is-huntington:
apply unfold-locales
apply (metis sup-assoc sup-commute sup-inf-absorb sup-compl-top
      inf-assoc inf-commute inf-sup-absorb inf-compl-bot
      sup-inf-distrib1)+
done
end

class huntington-algebra
lemma huntington-id:
x ⊔¬x = ¬x ⊔¬(¬x)
proof –
interpret boolean:
  boolean-algebra minus uminus (∩) (∪) ⊥ ⊤
  by (fact boolean-II-is-boolean)
show ?thesis by (simp add: boolean.boolean-is-huntington)
qed
end

class huntington-algebra
lemma huntington-id:
x ⊔¬x = ¬x ⊔¬(¬x)
proof –
from \textit{huntington} have
\[
x \sqcup -x = -(-x \sqcup -(x)) \sqcup -(x \sqcup -x) \sqcup (-(-x) \sqcup -(x)) \sqcup -(x \sqcup -(-x))
\]
by \textit{simp}

also from \textit{sup-comm} have
\[
\ldots = -(-x) \sqcup -(x) \sqcup -(-x) \sqcup -(x) \sqcup -(x) \sqcup -(x)
\]
by \textit{simp}

also from \textit{sup-assoc} have
\[
\ldots = -(-x) \sqcup -(x) \sqcup -(-x) \sqcup -(x) \sqcup -(x) \sqcup -(x)
\]
by \textit{simp}

also from \textit{sup-comm} have
\[
\ldots = -(-x) \sqcup -(x) \sqcup -(-x) \sqcup -(x) \sqcup -(x) \sqcup -(x)
\]
by \textit{simp}

also from \textit{sup-assoc} have
\[
\ldots = -(-x) \sqcup -(x) \sqcup -(-x) \sqcup -(x) \sqcup -(x) \sqcup -(x)
\]
by \textit{simp}

also from \textit{huntington} have
\[
\ldots = -x \sqcup -(x)
\]
by \textit{simp}

finally show \textit{thesis} by \textit{simp}

qed

\textbf{lemma} \textit{dbl-neg}: \(-x = x\)

apply (metis \textit{huntington} \textit{huntington-id} \textit{sup-comm})

done

\textbf{lemma} \textit{towards-sup-compl}: \(x \sqcup -x = y \sqcup -y\)

proof

from \textit{huntington} have
\[
x \sqcup -x = -(x \sqcup -(y)) \sqcup -(x \sqcup -y) \sqcup -(x) \sqcup -(y)
\]
by \textit{simp}

also from \textit{sup-comm} have
\[
\ldots = -(y) \sqcup -x \sqcup -(y \sqcup -x) \sqcup -(y \sqcup -(x)) \sqcup -(y \sqcup -(x))
\]
by \textit{simp}

also from \textit{sup-assoc} have
\[
\ldots = -(y) \sqcup -x \sqcup -(y \sqcup -(x)) \sqcup -(y \sqcup -(x)) \sqcup -(y \sqcup -(x))
\]
by \textit{simp}

also from \textit{sup-comm} have
\[ \ldots = -(-y \cup -(-z)) \cup -(-y \cup -x) \cup -(-(y \cup -x) \cup -(-(-y) \cup -(-x))) \]

by simp
also from sup-assoc have
\[ \ldots = -(-y \cup -(-x)) \cup -(-y \cup -x) \cup -(-(y \cup -(-y) \cup -(-x))) \]
by simp
also from sup-comm have
\[ \ldots = -(-y \cup -(-x)) \cup -(-y \cup -x) \cup -(-(y \cup -(-y) \cup -(-x))) \]
by simp
also from huntington have
\[ y \cup -y = \ldots \text{ by simp} \]
finally show \(?thesis by simp\)
qed

lemma sup-compl: \(x \cup -x = \top\)
by (simp add: top-eq towards-sup-compl)

lemma towards-inf-compl: \(x \cap -x = y \cap -y\)
by (metis dbl-neg inf-eq sup-comm sup-compl)

lemma inf-compl: \(x \cap -x = \bot\)
by (metis dbl-neg sup-comm bot-eq towards-inf-compl inf-eq)

lemma towards-idem: \(\bot = \bot \cup \bot\)
by (metis dbl-neg sup-comm bot-eq towards-inf-compl inf-eq)

lemma sup-ident: \(x \cup \bot = x\)
by (metis dbl-neg huntington inf-compl inf-eq sup-assoc sup-comm sup-compl)

lemma inf-ident: \(x \cap \top = x\)
by (metis dbl-neg inf-compl inf-eq sup-ident sup-comm sup-compl)

lemma sup-ident: \(x \cup x = x\)
by (metis dbl-neg huntington inf-compl inf-eq sup-ident sup-comm sup-compl)

lemma inf-ident: \(x \cap x = x\)
by (metis dbl-neg inf-eq sup-ident)

lemma sup-nil: \(x \cup \top = \top\)
by (metis sup-ident sup-assoc sup-comm sup-compl)

lemma inf-nil: \(x \cap \bot = \bot\)
by (metis dbl-neg inf-compl inf-eq sup-nil sup-comm sup-compl)

lemma sup-absorb: \(x \cup x \cap y = x\)
by (metis huntington inf-eq sup-ident sup-assoc sup-comm)

lemma inf-absorb: \(x \cap (x \cup y) = x\)
by (metis dbl-neg inf-eq sup-absorb)

lemma partition: \( x \cap y \cup x \cap -y = x \)
by (metis dbl-neg huntington inf-eq sup-comm)

lemma demorgans1: \(- (x \cap y) = -x \cup -y \)
by (metis dbl-neg inf-eq)

lemma demorgans2: \(- (x \cap y) = -x \cap -y \)
by (metis dbl-neg inf-eq)

lemma inf-comm: \( x \cap y = y \cap x \)
by (metis inf-eq sup-comm)

lemma inf-assoc: \( x \cap (y \cap z) = x \cap y \cap z \)
by (metis dbl-neg inf-eq sup-assoc)

lemma inf-sup-distrib1: \( x \cap (y \cup z) = (x \cap y) \cup (x \cap z) \)
proof
from partition have
  \( x \cap (y \cup z) = x \cap (y \cup z) \cap y \cup x \cap (y \cup z) \cap -y \) ..
also from inf-assoc have
  \( \ldots = x \cap ((y \cup z) \cap y) \cup x \cap (y \cup z) \cap -y \) by simp
also from inf-comm have
  \( \ldots = x \cap (y \cap (y \cup z)) \cup x \cap (y \cup z) \cap -y \) by simp
also from inf-absorb have
  \( \ldots = (x \cap y) \cup (x \cap (y \cup z) \cap -y) \) by simp
also from partition have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap (y \cup z) \cap -y \cap -z) \cup (x \cap (y \cup z) \cap -y \cap z)) \) by simp
also from inf-assoc have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (x \cap ((y \cup z) \cap (-y \cap z)))) \) by simp
also from demorgans2 have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (x \cap ((y \cup z) \cap (-y \cup z)))) \) by simp
also from inf-compl have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (x \cap (\bot))) \) by simp
also from inf-nil have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (\bot)) \) by simp
also from sup-idem have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (\bot)) \) by simp
also from sup-ident have
  \( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap ((y \cup z) \cap (-y \cap z))) \cup (\bot)) \) by simp
also from inf-comm have
... = ((x \cap y \cap z) \cup (x \cap y \cap z) \cup (x \cap y \cap -z)) \cup 
(x \cap ((-y \cap z) \cap (y \cup z))) \text{ by simp}
also from sup-comm have
... = ((x \cap y \cap z) \cup (x \cap y \cap z) \cup (x \cap y \cap -z)) \cup 
(x \cap ((-y \cap z) \cap (z \cup y))) \text{ by simp}
also from inf-assoc have
... = ((x \cap y \cap z) \cup (x \cap (y \cap z)) \cup (x \cap y \cap -z)) \cup 
(x \cap (-y \cap z)) \text{ by simp}
also from inf-absorb have
... = ((x \cap y \cap z) \cup (x \cap (y \cap z)) \cup (x \cap y \cap -z)) \cup (x \cap (-y \cap z)) 
\text{ by simp}
also from inf-comm have
... = ((x \cap y \cap z) \cup (x \cap (z \cap y)) \cup (x \cap y \cap -z)) \cup (x \cap (z \cap -y)) 
\text{ by simp}
also from sup-assoc have
... = ((x \cap y \cap z) \cup ((x \cap (z \cap y)) \cup (x \cap y \cap -z))) \cup (x \cap (z \cap -y)) 
\text{ by simp}
also from sup-comm have
... = ((x \cap y \cap z) \cup ((x \cap y \cap -z) \cup (x \cap (z \cap y)))) \cup (x \cap (z \cap -y)) 
\text{ by simp}
also from sup-assoc have
... = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap (z \cap y)) \cup (x \cap (z \cap -y))) 
\text{ by simp}
also from inf-assoc have
... = ((x \cap y \cap z) \cup (x \cap y \cap -z)) \cup ((x \cap z \cap y) \cup (x \cap z \cap -y)) \text{ by simp}
also from partition have ... = (x \cap y) \cup (x \cap z) \text{ by simp}
finally show \text{ ?thesis by simp}
qed

lemma sup-inf-distrib1:
x \cup (y \cap z) = (x \cup y) \cap (x \cup z)
proof –
  from dbl-neg have
  x \cup (y \cap z) = -(-(-x) \cup (y \cap z)) \text{ by simp}
  also from inf-eq have
  ... = -(-x \cap (-y \cup -z)) \text{ by simp}
  also from inf-sup-distrib1 have
  ... = -((-x \cap -y) \cup (-x \cap -z)) \text{ by simp}
  also from demorgans2 have
  ... = -(-x \cap -y) \cap -(-x \cap -z) \text{ by simp}
  also from demorgans1 have
  ... = (-(-x) \cup -(-y)) \cap (-(-x) \cup -(-z)) \text{ by simp}
  also from dbl-neg have
  ... = (x \cup y) \cap (x \cup z) \text{ by simp}
  finally show ?thesis by simp
qed

lemma huntington-is-boolean-II:
class boolean-algebra-II aminus (\cap) (\cup) \bot \top
apply unfold-locales
apply (metis inf-comm inf-assoc sup-absorb
     inf-absorb sup-inf-distrib1
     sup-compl inf-compl)+
done

lemma huntington-is-boolean:
  class.boolean-algebra minus uminus (Π) (⊑) (⊏) (⊔) ⊥⊤
proof −
  interpret boolean-II:
    boolean-algebra-II minus (Π) (⊔) ⊥⊤
  by (fact huntington-is-boolean-II)
  show ?thesis by (simp add: boolean-II.boolean-II-is-boolean)
qed
end

3.3 Robbins’ Algebra

context boolean-algebra begin
lemma boolean-is-robbins:
  class.robbins-algebra minus inf sup bot top
apply unfold-locales
apply (metis sup-assoc sup-commute compl-inf double-compl sup-compl-top
       inf-compl-bot diff-eq sup-bot-right sup-inf-distrib1)+
done
end

context boolean-algebra-II begin
lemma boolean-II-is-robbins:
  class.robbins-algebra minus inf sup bot top
proof −
  interpret boolean:
    boolean-algebra minus uminus (Π) (⊑) (⊔) ⊥⊤
  by (fact boolean-II-is-boolean)
  show ?thesis by (simp add: boolean.boolean-is-robbins)
qed
end

context huntington-algebra begin
lemma huntington-is-robbins:
  class.robbins-algebra minus inf sup bot top
proof −
  interpret boolean:
    boolean-algebra minus uminus (Π) (⊑) (⊔) ⊥⊤
  by (fact huntington-is-boolean)
  show ?thesis by (simp add: boolean.boolean-is-robbins)
qed
end

Before diving into the proof that the Robbins algebra is Boolean, we
shall present some shorthand machinery

context common-algebra begin

primrec copy :: nat ⇒ 'a ⇒ 'a (infix ⊗ 80)
where
  copy-0: 0 ⊗ x = x
| copy-Suc: (Suc k) ⊗ x = (k ⊗ x) ⊔ x

no-notation
  Product-Type.Times (infixr × 80)

primrec copy :: nat ⇒ 'a ⇒ 'a (infix × 85)
where
  0 × x = x
| (Suc k) × x = k ⊗ x

lemma one: 1 × x = x
proof –
  have 1 = Suc(0) by arith
  hence 1 × x = Suc(0) × x by metis
  also have ... = x by simp
  finally show ?thesis by simp
qed

lemma two: 2 × x = x ⊔ x
proof –
  have 2 = Suc(Suc(0)) by arith
  hence 2 × x = Suc(Suc(0)) × x by metis
  also have ... = x ⊔ x by simp
  finally show ?thesis by simp
qed

lemma three: 3 × x = x ⊔ x ⊔ x
proof –
  have 3 = Suc(Suc(Suc(0))) by arith
  hence 3 × x = Suc(Suc(Suc(0))) × x by metis
  also have ... = x ⊔ x ⊔ x by simp
  finally show ?thesis by simp
qed

lemma four: 4 × x = x ⊔ x ⊔ x ⊔ x
proof –
  have 4 = Suc(Suc(Suc(Suc(0)))) by arith
  hence 4 × x = Suc(Suc(Suc(Suc(0)))) × x by metis
  also have ... = x ⊔ x ⊔ x ⊔ x by simp

10
lemma five: \(5 \times x = x \sqcup x \sqcup x \sqcup x \sqcup x\)
proof -
  have \(5 = \text{Suc}((\text{Suc}(\text{Suc}(\text{Suc}(0))))))\) by arith
  hence \(5 \times x = \text{Suc}((\text{Suc}(\text{Suc}(\text{Suc}(0)))))) \times x\) by metis
also have \(\ldots = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\) by simp
finally show \(?thesis\) by simp
qed

lemma six: \(6 \times x = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\)
proof -
  have \(6 = \text{Suc}((\text{Suc}(\text{Suc}(\text{Suc}(0))))))\) by arith
  hence \(6 \times x = \text{Suc}((\text{Suc}(\text{Suc}(\text{Suc}(0)))))) \times x\) by metis
also have \(\ldots = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\) by simp
finally show \(?thesis\) by simp
qed

lemma copyp-distrib: \(k \otimes (x \sqcup y) = (k \otimes x) \sqcup (k \otimes y)\)
proof (induct \(k\))
  case 0 show \(?case\) by simp
  case Suc thus \(?case\) by (simp, metis sup-assoc sup-comm)
qed

corollary copyp-distrib: \(k \times (x \sqcup y) = (k \times x) \sqcup (k \times y)\)
by (induct \(k\), (simp add: sup-assoc sup-comm copyp-distrib)+)

lemma copyp-arith: \((k + l + 1) \otimes x = (k \otimes x) \sqcup (l \otimes x)\)
proof (induct \(l\))
  case 0 have \(k + 0 + 1 = \text{Suc}(k)\) by arith
  thus \(?case\) by simp
  case (Suc \(l\)) note ind-hyp = this
  have \(k + \text{Suc}(l) + 1 = \text{Suc}(k + l + 1)\) by arith+
  hence \((k + \text{Suc}(l) + 1) \otimes x = (k + l + 1) \otimes x \sqcup x\) by (simp add: ind-hyp)
  also from ind-hyp have \(\ldots = (k \otimes x) \sqcup (l \otimes x) \sqcup x\) by simp
  also note sup-assoc
  finally show \(?case\) by simp
qed

lemma copy-arith:
  assumes \(k \neq 0\) and \(l \neq 0\)
  shows \((k + l) \times x = (k \times x) \sqcup (l \times x)\)
using assms
proof -
  from assms have \(\exists k'. \text{Suc}(k') = k\)
\[ \exists l'. \text{Suc}(l') = l \text{ by arith+} \]

from this obtain \(k' l'\) where
\[ A: \text{Suc}(k') = k \]
and \( B: \text{Suc}(l') = l \text{ by fast+} \)

from this have \( A1: k \times x = k' \odot x \)
and \( B1: l \times x = l' \odot x \text{ by fastforce+} \)
from \( A B \) have \( k + l = \text{Suc}(k' + l' + 1) \) by arith

hence \((k + l) \times x = (k' + l' + 1) \odot x \text{ by simp} \)
also from copyp-arith have
\[ \ldots = k' \odot x \sqcup l' \odot x \text{ by fast} \]
also from \( A1 B1 \) have
\[ \ldots = k \times x \sqcup l \times x \text{ by fastforce} \]
finally show \( ? \)thesis by simp
qed

end

The theorem asserting all Robbins algebras are Boolean comes in 6 movements.

First: The Winker identity is proved.
Second: Idempotence for a particular object is proved. Note that falsum is defined in terms of this object.
Third: An identity law for falsum is derived.
Fourth: Idempotence for supremum is derived.
Fifth: The double negation law is proven
Sixth: Robbin’s algebras are proven to be Huntington Algebras.

class robbins-algebra begin

definition secret-object2 :: 'a (α) where
\[ α = -(-(-(-x \sqcup y) \sqcup -x) \sqcup y) \]
definition secret-object3 :: 'a (β) where
\[ β = x \sqcup y \]
definition secret-object4 :: 'a (δ) where
\[ δ = β \sqcup (-(-α \sqcup -β) \sqcup -(-α \sqcup -β)) \]
definition secret-object5 :: 'a (γ) where
\[ γ = δ \sqcup -(-δ \sqcup -δ) \]
definition winker-object :: 'a (ϱ) where
\[ ϱ = γ \sqcup γ \sqcup γ \]
definition fake-bot :: 'a (⊥⊥) where
\[ ⊥⊥ = -(ϱ \sqcup -ϱ) \]

lemma robbins2: \( y = -(-(-x \sqcup y) \sqcup -x \sqcup y) \)
by (metis robbins sup-comm)
lemma mann0: \( -(x \sqcup y) = -(-(-x \sqcup y) \sqcup -x \sqcup y) \sqcup y \)

end
lemma mann1: \(-(-x \sqcup y) = -(-(-x \sqcup y) \sqcup x \sqcup y) \sqcup y\)
  by (metis robbins sup-comm sup-assoc)

lemma mann2: \(y = -((-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y)\)
  by (metis mann1 robbins sup-comm sup-assoc)

lemma mann3: \(z = -(-(-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y) \sqcup z) \sqcup -(y \sqcup z)\)
  proof –
  let \(?w = -(-(-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y)\)
  from robbins where \(x=z\) and \(y=?w\) sup-comm mann2
  have \(z = -((-y \sqcup z) \sqcup -(?w \sqcup z))\) by metis
  thus \(?thesis\) by (metis sup-comm)

qed

lemma mann4: \(-(y \sqcup z) =\)
\[-(-((-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y) \sqcup -(y \sqcup z) \sqcup z) \sqcup z\]
  proof –
  from robbins2 where \(x=-((-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y) \sqcup z\)
    and \(y=-((y \sqcup z))\) mann3 where \(x=x\) and \(y=y\) and \(z=z\)
  have \(-((y \sqcup z)) =\)
\[-(-((-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y) \sqcup z \sqcup -(y \sqcup z)))\]
  by metis

with sup-comm sup-assoc show \(?thesis\) by metis

qed

lemma mann5: \(u =\)
\[-(-(-(-x \sqcup y) \sqcup x \sqcup y \sqcup y)) \sqcup\]
\[-((-x \sqcup y) \sqcup -(y \sqcup z) \sqcup z) \sqcup z \sqcup u) \sqcup\]
\[-(-(y \sqcup z) \sqcup u))\]
  using robbins2 where \(x=-((-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup
\[-(-x \sqcup y) \sqcup -(y \sqcup z) \sqcup z) \sqcup z\)
    and \(y=u\)
  mann4 where \(x=x\) and \(y=y\) and \(z=z\)

  sup-comm

by metis

lemma mann6:
\[-(3 \times x \sqcup x) = -(-(-3 \times x \sqcup x) \sqcup -3 \times x) \sqcup -((-3 \times x \sqcup x) \sqcup 5 \times x))\)
  proof –
  have \(3+2=\(5::\text{nat}\)\) and \(3\neq(0::\text{nat})\) and \(2\neq(0::\text{nat})\) by arith+

  with copy-arith have \(\exists:3 \times x \sqcup 2 \times x = 5 \times x\) by metis
  let \(?p = -(-3 \times x \sqcup x)\)

  \{ fix \(q\)
  from sup-comm have\n  \(- (q \sqcup 5 \times x) = -(3 \times x \sqcup q)\) by metis

  \}
also from $\lor$ mann0[where $x=3\times x$ and $y=q \cup 2 \times x$] sup-assoc sup-comm have
\[ \ldots = -(-(-3 \times x \cup (q \cup 2 \times x)) \cup -3 \times x \cup (q \cup 2 \times x)) \cup (q \cup 2 \times x)) \]
by metis
also from sup-assoc have
\[ \ldots = -(-((-3 \times x \cup q) \cup 2 \times x) \cup -3 \times x \cup (q \cup 2 \times x)) \cup (q \cup 2 \times x)) \]
by metis
also from sup-comm have
\[ \ldots = -(-((3 \times x \cup q) \cup 2 \times x) \cup -3 \times x \cup (q \cup 2 \times x)) \cup (q \cup 2 \times x)) \]
by metis
also from \lor have
\[ \ldots = -(-(-q \cup 5 \times x) \cup -3 \times x \cup (q \cup 2 \times x)) \cup (q \cup 2 \times x)) \]
by metis
also from sup-assoc have
\[ \ldots = -(-(-(q \cup 5 \times x) \cup (q \cup -3 \times x) \cup 2 \times x) \cup (q \cup 2 \times x)) \]
by metis
also from sup-comm have
\[ \ldots = -(-(-q \cup 5 \times x) \cup q \cup -3 \times x \cup 2 \times x) \cup 2 \times x \cup q) \]
by metis
finally have
\[ -(q \cup 5 \times x) = -(-(q \cup 5 \times x) \cup q \cup -3 \times x \cup 2 \times x) \cup 2 \times x \cup q) \]
by simp
\}
hence $\blacklozenge$:
\[ -(?p \cup 5 \times x) = -(-(?q \cup 5 \times x) \cup ?p \cup -3 \times x \cup 2 \times x) \cup 2 \times x \cup ?p) \]
by simp

from mann5[where $x=3\times x$ and $y=x$ and $z=2 \times x$ and $u=?p$]
\[ sup-assoc three[where $x=x$] five[where $x=x$] have \]
\[ ?p = \]
\[ -(-(-(?p \cup 5 \times x) \cup ?p \cup -(x \cup 2 \times x) \cup 2 \times x \cup ?p)) \cup \]
\[ -(x \cup 2 \times x) \cup ?p)) \]
by metis
also from sup-comm have
\[ \ldots = \]
\[ -(-(-(?p \cup 5 \times x) \cup ?p \cup -(2 \times x \cup x) \cup 2 \times x \cup ?p)) \cup \]
\[ -(2 \times x \cup x) \cup ?p)) \]
by metis
also from two[where $x=x$] three[where $x=x$] have
\[ \ldots = \]
\[ -(-(-(?p \cup 5 \times x) \cup ?p \cup -3 \times x \cup 2 \times x) \cup 2 \times x \cup ?p) \cup \]
\[ -(3 \times x \cup ?p)) \]
by metis
also from $\blacklozenge$ have \ldots = \[-(?(?p \cup 5 \times x) \cup -(3 \times x \cup ?p)) \]
by simp
also from sup-comm have \ldots = \[-(?(?p \cup 5 \times x) \cup -(?(p \cup -3 \times x)) \]
by simp
also from sup-comm have \ldots = \[-(?(?p \cup -3 \times x) \cup -(?(p \cup 5 \times x)) \]
by simp
finally show $\lnot$thesis.
qed

lemma mann7:
\[ -3 \times x = -(-3 \times x \cup x) \cup 5 \times x) \]
proof –
let $?p = -(- 3 \times x \sqcup x)
let $?q = {?p} \sqcup - 3 \times x
let $?r = -(?p \sqcup 5 \times x)
from robbins2[where x=?q
  and y=?r]
mann6[where x=x]
  have $?r = -(?p \qsqcup - (?q \qsqcup ?r)) by simp
also from sup-comm have ... = -(- (?q \qsqcup ?r) \sqcup ?p) by simp
also from sup-comm have ... = -(- (?r \qsqcup ?q) \sqcup ?p) by simp
finally have ♠: $?r = -(- (?r \sqcup ?q) \sqcup ?p) .
from mann3[where x=3 \times x and y=x and z=- 3 \times x]
sup-comm have
- 3 \times x = -(-(- (?p \sqcup 3 \times x \sqcup x) \sqcup ?p \sqcup -3 \times x) \sqcup ?p) by metis
also from sup-assoc have
... = -(-(-(?p \sqcup (3 \times x \sqcup x) \sqcup ?q) \sqcup ?p) by metis
also from three[where x=x] five[where x=x] have
... = -(-(?r \sqcup ?q) \sqcup ?p) by metis
finally have - 3 \times x = -(-(?r \sqcup ?q) \sqcup ?p) by metis
with ♠ show ?thesis by simp
qed

lemma mann8:
-(- 3 \times x \sqcup x) \sqcup 2 \times x = -(-(- 3 \times x \sqcup x) \sqcup -3 \times x \sqcup 2 \times x) \sqcup -3 \times x
(is ?lhs = ?rhs)
proof –
  let $?p = -(- 3 \times x \sqcup x)
  let $?q = {?p} \sqcup 2 \times x
  let $?r = 3 \times x
  have 3+2=(5::nat) and 3\neq(0::nat) and 2\neq(0::nat) by arith+
  with copy-arith have \lor: 3 \times x \sqcup 2 \times x = 5 \times x by metis
  from robbins2[where x=?r and y=?q] and sup-assoc
  have $?q = -(-3 \times x \sqcup ?q) \sqcup -(3 \times x \sqcup ?p \sqcup 2 \times x)) by metis
  also from sup-comm have
  ... = -(-(?q \sqcup -3 \times x) \sqcup -(?p \sqcup 3 \times x \sqcup 2 \times x)) by metis
  also from \lor sup-assoc have
  ... = -(-(?q \sqcup -3 \times x) \sqcup -(?p \sqcup 5 \times x)) by metis
  also from mann7[where x=x] have
  ... = -(-(?q \sqcup -3 \times x) \sqcup -3 \times x) by metis
  also from sup-assoc have
  ... = -(-(?p \sqcup (2 \times x \sqcup -3 \times x)) \sqcup -3 \times x) by metis
  also from sup-comm have
  ... = -(-(?p \sqcup -(3 \times x \sqcup 2 \times x)) \sqcup -3 \times x) by metis
  also from sup-assoc have
  ... = ?rhs by metis
  finally show ?thesis by simp
qed

lemma mann9: x = -(-3 \times x \sqcup x) \sqcup -3 \times x
proof –
let \( ?p = -(3 \times x \sqcup x) \)
let \( ?q = ?p \sqcup 4 \times x \)

have \( 4 + 1 = (5 :: \text{nat}) \) and \( 1 \neq (0 :: \text{nat}) \) and \( 4 \neq (0 :: \text{nat}) \) by arith+

with copy-arith one have \( \bowtie: 4 \times x \sqcup x = 5 \times x \) by metis

with sup-assoc robbins2[where \( y = x \) and \( x = ?q \)] have \( x = -(-(-?q \sqcup x) \sqcup (\neg ?q \sqcup 5 \times x)) \) by metis

with mann7 have \( x = -(-(-?q \sqcup x) \sqcup 3 \times x) \) by metis

moreover have \( 3 + 1 = (4 :: \text{nat}) \) and \( 1 \neq (0 :: \text{nat}) \) and \( 3 \neq (0 :: \text{nat}) \) by arith+

with copy-arith one have \( \spadesuit: 3 \times x \sqcup x = 4 \times x \) by metis

with mann1[where \( x = 3 \times x \) and \( y = x \)] sup-assoc have 
\(-(-?q \sqcup x) = ?p \) by metis

ultimately show \( ?thesis \) by simp

qed

lemma mann10: \( y = -(-(-3 \times x \sqcup x) \sqcup -3 \times x \sqcup y) \sqcup -(x \sqcup y)) \)
using robbins2[where \( x = -(3 \times x \sqcup x) \sqcup -3 \times x \sqcup y \) and \( y = y \)]

mann9[where \( x = x \)]

sup-comm

by metis

theorem mann: \( 2 \times x = -(3 \times x \sqcup x) \sqcup 2 \times x \)
using mann10[where \( x = x \) and \( y = 2 \times x \)]

mann8[where \( x = x \)]

two[where \( x = x \)]

three[where \( x = x \)]
sup-comm

by metis

corollary winkerr: \( \alpha \sqcup \beta = \beta \)
using mann secret-object2-def secret-object3-def two three

by metis

corollary winker: \( \beta \sqcup \alpha = \beta \)
by (metis winkerr sup-comm)

corollary multi-winkerp: \( \beta \sqcup k \otimes \alpha = \beta \)
by (induct k, (simp add: winkerp sup-comm sup-assoc)+)

corollary multi-winker: \( \beta \sqcup k \times \alpha = \beta \)
by (induct k, (simp add: multi-winkerp winkerp sup-comm sup-assoc)+)

lemma less-eq-introp:
\( -(x \sqcup -(y \sqcup z)) = -(x \sqcup y \sqcup -z) \Rightarrow y \subseteq x \)
by (metis robbins sup-assoc less-eq-def
sup-comm[where \( x = x \) and \( y = y \)])

corollary less-eq-intro:
\( -(x \sqcup -(y \sqcup z)) = -(x \sqcup y \sqcup -z) \Rightarrow x \sqcup y = x \)
lemma eq-intro:
\(- (x \sqcup -(y \sqcup z)) = -(y \sqcup -(x \sqcup z)) \implies x = y\)
by (metis less-eq-introp less-eq-def sup-comm)

lemma copyp0:
assumes \(-(x \sqcup -y) = z\)
shows \(-(x \sqcup -(y \sqcup k \otimes (x \sqcup z))) = z\)
using assms
proof (induct k)
case 0 show ?case by (simp, metis robbins sup-assoc sup-comm)
case Suc note ind-hyp = this show ?case by (simp, metis ind-hyp robbins sup-assoc sup-comm)
qed

lemma copyp1:
assumes \-(x \sqcup -y) = x\)
shows \-(y \sqcup k \otimes (x \sqcup -(x \sqcup -y))) = -y\)
using assms proof
let \(?z = -(x \sqcup -y)\)
let \(?ky = y \sqcup k \otimes (x \sqcup z)\)
have \(-(x \sqcup -?ky) = ?z\) by (simp add: copyp0)
hence \(-(?ky \sqcup -(y \sqcup ?z)) = ?z\) by (metis assms sup-comm)
also have \-(?z \sqcup -?ky) = x\) by (metis assms copyp0 sup-comm)
hence \(?z = -(y \sqcup -(?ky \sqcup ?z))\) by (metis sup-comm)
finally show \?thesis by (metis eq-intro)
qed

corollary copyp2:
assumes \-(x \sqcup y) = -y\)
shows \-(y \sqcup k \otimes (x \sqcup -(x \sqcup -y))) = -y\)
by (metis less-eq-introp less-eq-def sup-comm)

lemma two-threep:
assumes \-(2 \times x \sqcup y) = -y\)
and \-(3 \times x \sqcup y) = -y\)
shows 2 \times x \sqcup y = 3 \times x \sqcup y
using assms
proof
from assms two three have
A: \-(x \sqcup x \sqcup y) = -y\) and
B: \-(x \sqcup x \sqcup x \sqcup y) = -y\) by simp+
with sup-assoc
copyp2[where x=x and y=x \sqcup x \sqcup y and k=0]
have \(-(x \sqcup x \sqcup y \sqcup x \sqcup -(x \sqcup -y)) = -y\) by simp
moreover
from sup-comm sup-assoc A B
  copypp2[where x=x U x and y=y and k=0]
have -(y U x U x U -(x U x U -y)) = -y by fastforce
with sup-comm sup-assoc
have -(x U x U y U -(x U (x U -y))) = -y by metis
ultimately have
-((x U x U y U -(x U (x U -y)))) = -(x U x U y U x U -(x U -y)) by simp
with less-eq-intro have x U x U y = x U x U y U x by metis
with sup-comm sup-assoc two three show ?thesis by metis
qed

two-three:
assumes -(x U y) = -y V -(x U -y U -y) = x
shows y U 2 * (x U -(x U -y)) = y U 3 * (x U -(x U -y))
  (is y U 2 = y U 3)
using assms
proof
  assume -(x U y) = -y
  with copypp2[where k=Suc(0)]
    copypp2[where k=Suc(Suc(0))]
    two three
    have -(y U 2 = -y and -(y U 3 = -y) = -y by simp+
    with two-three sup-comm show ?thesis by metis
next
  assume -(x U -y U -y) = x
  with copypp1[where k=Suc(0)]
    copypp1[where k=Suc(Suc(0))]
    two three
    have -(y U 2 = -y and -(y U 3 = -y) = -y by simp+
    with two-three sup-comm show ?thesis by metis
qed

sup-idem: g U g = g
proof
  from winkerr two
    copypp2[where x=alpha and y=beta and k=Suc(0)] have
    -beta = -(beta U 2 * (alpha U -(alpha U -beta))) by simp
    also from copy-distrib sup-assoc have
    ... = -(beta U 2 * alpha U 2 * ... by simp
    also from sup-assoc secret-object4-def two
    multi-winker[where k=2] have
    ... = -delta by metis
    finally have -beta = -delta by simp
    with secret-object4-def sup-assoc three have
    delta U -(alpha U -delta) = beta U 3 * ... by simp
    also from copy-distrib[where k=3]
    multi-winker[where k=3]
    sup-assoc have
\( \ldots = \beta \cup 2 \times (\alpha \cup - (\alpha \cup -\beta)) \) by \textit{metis}

also from \textit{winker sup-comm two-three[where } x=\alpha \textit{ and } y=\beta \textit{]} have
\( \ldots = \beta \cup 2 \times (\alpha \cup - (\alpha \cup -\beta)) \) by \textit{fastforce}

also from \textit{copy-distrib[where } k=2 \textit{]}
\begin{align*}
\text{multi-winker[where } k=2 \textit{] sup-assoc two secret-object4-def have } \\
\delta = \delta \text{ by \textit{metis}}
\end{align*}

finally have \( \bowtie : \delta \cup - (\alpha \cup -\delta) = \delta \text{ by \textit{simp}} \)

from \textit{secret-object4-def winker sup-assoc have}
\( \alpha \cup \delta = \delta \text{ by \textit{metis}} \)

hence \( \delta \cup \alpha = \delta \text{ by (metis sup-comm)} \)

hence \( -(-\delta \cup -\delta) \cup -\delta) = -(-\delta \cup (\alpha \cup -\delta)) \cup -\delta) \text{ by (metis sup-assoc)} \)

also from \( \bowtie \text{ have} \)
\( \ldots = -(\delta \cup (\alpha \cup -\delta)) \cup -(\delta \cup - (\alpha \cup -\delta)) \) by \textit{metis}

also from \textit{robbins have}
\( \ldots = \delta \text{ by \textit{metis}} \)

finally have \( -(-\delta \cup -\delta) \cup -\delta) = \delta \text{ by \textit{simp}} \)

with \textit{two-three[where } x=\delta \textit{ and } y=\delta \textit{]}
\textit{secret-object5-def sup-comm}

have \( 3 \times \gamma \cup \delta = 2 \times \gamma \cup \delta \text{ by \textit{fastforce}} \)

with \textit{secret-object5-def sup-assoc sup-comm have}
\( 3 \times \gamma \cup \gamma = 2 \times \gamma \cup \gamma \text{ by \textit{fastforce}} \)

with \textit{two three four five six have}
\( 6 \times \gamma = 3 \times \gamma \text{ by \textit{simp}} \)

moreover have \( 3 + 3 = (6::nat) \text{ and } 3 \neq (\theta::nat) \text{ by \textit{arith}} \)

moreover note \textit{copy-arith[where } k=3 \textit{ and } l=3 \textit{ and } x=\gamma \textit{]}
\textit{winker-object-def three}

ultimately show \textit{thesis by \textit{simp}}

\textbf{qed}

\textbf{lemma} \textit{sup-ident: } x \cup \bot \bot = x

\textbf{proof} –

\texttt{have I: } \varrho = -(-\varrho \cup \bot \bot) 
\texttt{by (metis fake-bot-def inf-eq robbins sup-comm sup-idem)}

\{ \texttt{fix } x \texttt{ have } x = -(x \cup -\varrho \cup \bot \bot) \cup -(x \cup \varrho) \}
\texttt{by (metis I robbins sup-assoc} \}

\textbf{note} \( II = \textit{this} \)

\texttt{have III: } -\varrho = -(\varrho \cup -\varrho \cup -\varrho) \cup \varrho 
\texttt{by (metis robbins[where } x=-\varrho \textit{ and } y=\varrho \cup -\varrho \textit{]}
\texttt{I sup-comm fake-bot-def)}

\texttt{hence } \varrho = -(-\varrho \cup -\varrho \cup -\varrho) \cup -\varrho 
\texttt{by (metis robbins[where } x=\varrho \texttt{ and } y=\varrho \cup -\varrho \cup -\varrho \]} 
\texttt{sup-comm[where } x=\varrho \texttt{ and } y=-(-\varrho \cup -\varrho \cup -\varrho)] 
\texttt{sup-assoc sup-idem)}

\texttt{hence } -(\varrho \cup -\varrho \cup -\varrho) = \bot \bot
by (metis robbins[where \( x=- (q \sqcup -q \sqcup -q) \) and \( y=q \])
   III sup-comm fake-bot-def)
hence \(-q = -(q \sqcup \bot \bot) \)
   by (metis III sup-comm)
hence \(q = -(- (q \sqcup \bot \bot) \sqcup - (q \sqcup \bot \bot \sqcup -q)) \)
   by (metis II sup-idem sup-comm sup-assoc)
moreover have \( q \sqcup \bot \bot = - (- (q \sqcup \bot \bot) \sqcup - (q \sqcup \bot \bot \sqcup -q)) \)
   by (metis robbins[where \( x=q \sqcup \bot \bot \) and \( y=q \])
   sup-comm[where \( y=q \])
   sup-assoc sup-idem)
ultimately have \( q = q \sqcup \bot \bot \) by auto
hence \( x \sqcup \bot \bot = - (- (x \sqcup q) \sqcup - (x \sqcup \bot \bot \sqcup -q)) \)
   by (metis I robbins sup-comm sup-ident)
thus \(?thesis \) by (metis sup-assoc sup-comm II)
qed

lemma dbl-neg: \(-(-x) = x \)
proof –
\{ fix \( x \) have \( \bot \bot = -(-x \sqcup -(-x)) \)
   by (metis robbins sup-comm sup-ident) \}
   note I = this
\{ fix \( x \) have \(-x = -(-x \sqcup -(-x))) \)
   by (metis I robbins sup-comm sup-ident) \}
   note II = this
\{ fix \( x \) have \(-(-x)) = -(-x \sqcup -(-x))) \)
   by (metis I II robbins sup-assoc sup-comm sup-ident) \}
   note III = this
show \(?thesis \) by (metis II III robbins)
qed

theorem robbins-is-huntington:
\class. huntington-algebra uminus \( \sqcap \) \( \sqcup \) \( \bot \) \( \top \)
apply unfold-locales
apply (metis dbl-neg robbins sup-comm)
done

theorem robbins-is-boolean-II:
\class. boolean-algebra-II uminus \( \sqcap \) \( \sqcup \) \( \bot \) \( \top \)
proof –
interpret huntington:
huntington-algebra uminus \((\land) (\cup) \bot \top\)
by (fact robbins-is-huntington)
show \(?thesis\) by (simp add: huntington.huntington-is-boolean-II)
qed

theorem robbins-is-boolean:
class boolean-algebra minus uminus \((\land) (\cup) (\equiv) (\sqsubset) (\sqcup) \bot \top\)
proof –
interpret huntington:
  huntington-algebra uminus \((\land) (\cup) \bot \top\)
  by (fact robbins-is-huntington)
show \(?thesis\) by (simp add: huntington.huntington-is-boolean)
qed

end

no-notation secret-object1 (\(\iota\))
  and secret-object2 (\(\alpha\))
  and secret-object3 (\(\beta\))
  and secret-object4 (\(\delta\))
  and secret-object5 (\(\gamma\))
  and winker-object (\(\eta\))
  and less-eq (infix \(\sqsubseteq\) 50)
  and less (infix \(\sqsubset\) 50)
  and inf (infix 1 \(\sqcap\) 70)
  and sup (infix 1 \(\sqcup\) 65)
  and top (\(\top\))
  and bot (\(\bot\))
  and copypp (infix \(\otimes\) 80)
  and copy (infix \(\times\) 85)
notation
  Product-Type.Times (infixr \(\times\) 80)
end