A Complete Proof of the Robbins Conjecture

Matthew Wampler-Doty

September 19, 2015

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


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 \( \land \))
fixes sup :: 'a ⇒ 'a ⇒ 'a (infixl \( \lor \))
fixes bot :: 'a (⊥)
fixes top :: 'a (⊤)
assumes sup-assoc: \( x \lor (y \lor z) = (x \lor y) \lor z \)
assumes sup-comm: \( x \lor y = y \lor x \)

context common-algebra begin
definition less-eq :: 'a ⇒ 'a ⇒ bool (infix \( \leq \)) where
\( x \leq y = (x \lor y = y) \)
definition less :: 'a ⇒ 'a ⇒ bool (infix \( < \)) where
\( x < y = (x \leq y \land \neg y \leq x) \)
definition minus :: 'a ⇒ 'a ⇒ 'a (infixl - \( \land \)) where
\( \text{minus } x \; y = (x \land \neg y) \)
definition secret-object1 :: 'a (ι) where
\( ι = (\text{SOME } x. \; \text{True}) \)

end

class ext-common-algebra = common-algebra +
assumes inf-eq: \( x \lor y = (∁ \; x \land y) \)
assumes top-eq: \( ⊤ = ι \lor ⊥ \)
assumes bot-eq: \( ⊥ = (∁ ι \land ι) \)

2.2 Boolean Algebra

class boolean-algebra-II =
common-algebra +
assumes inf-comm: \( x \lor y = y \lor x \)
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: \( x \uplus -x = \top \)
assumes inf-compl: \( x \cap -x = \bot \)

2.3 Huntington’s Algebra

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

2.4 Robbins’ Algebra

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

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 (op \(\cap\)) (op \(\leq\)) (op \(\geq\)) (op \(\lor\)) \(\bot\) \(\top\)
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
lemma boolean-is-boolean-II:
class boolean-algebra-II uminus inf sup bot top
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

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-compl-bot
compl-sup sup-commute sup-compl-top
sup-compl-top sup-assoc)+
done
end

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

context huntington-algebra-II begin

lemma boolean-II-is-huntington:
class huntington-algebra-II uminus (op ⊓) (op ⊔)
proof –
interpret boolean:
  boolean-algebra minus uminus (op ∩) (op ⊔) ⊥ ⊤
  by (fact boolean-II-is-boolean)
show ?thesis by (simp add: boolean.boolean-is-huntington)
qed
end
from huntington have
x \cup -x = \neg(-x \cup -(\neg x)) \cup (\neg(-x) \cup \neg(-x)) \cup (\neg(-x) \cup \neg(-x))

    by simp
also from sup-comm have
\ldots = \neg(-(\neg x) \cup -(\neg x)) \cup (\neg(-x) \cup -(\neg x)) \cup (\neg(-x) \cup -(\neg x))

    by simp
also from sup-assoc have
\ldots = \neg(-x) \cup -(\neg x) \cup \neg(-x) \cup -(\neg x)

    by simp
also from sup-comm have
\ldots = -(\neg x) \cup -(\neg x) \cup -(\neg x) \cup -(\neg x)

    by simp
also from sup-assoc have
\ldots = -(\neg x) \cup -(\neg x) \cup -(\neg x) \cup -(\neg x)

    by simp
also from huntington have
\ldots = -x \cup -(\neg x)

    by simp
finally show thesis by simp
qed

lemma dbl-neg: \neg(-x) = x
apply (metis huntington huntington-id sup-comm)
done

lemma towards-sup-compl: x \cup -x = y \cup -y
proof
    from huntington have
x \cup -x = \neg(-x \cup -(\neg y)) \cup -(-x \cup -y) \cup -(\neg(-x) \cup -(\neg y)) \cup -(\neg(-x) \cup -(\neg y))

    by simp
also from sup-comm have
\ldots = -(\neg y) \cup -x \cup -(\neg y) \cup -(\neg x) \cup -(\neg y) \cup -(\neg x)

    by simp
also from sup-assoc have
\ldots = -(\neg y) \cup -x \cup -(\neg y) \cup -(\neg x) \cup -(\neg y) \cup -(\neg x)

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

\textbf{lemma} \textit{sup-compl}: \[ x \sqcup -x = \top \]
\textbf{by} (\textit{simp add: top-eq towards-sup-compl})

\textbf{lemma} \textit{towards-inf-compl}: \[ x \sqcap -x = y \sqcap -y \]
\textbf{by} (\textit{metis dbl-neg inf-eq sup-comm sup-compl})

\textbf{lemma} \textit{inf-compl}: \[ x \sqcap -x = \bot \]
\textbf{by} (\textit{metis dbl-neg sup-comm bot-eq towards-inf-compl inf-eq})

\textbf{lemma} \textit{towards-idem}: \[ \bot = \bot \sqcup \bot \]
\textbf{by} (\textit{metis dbl-neg sup-comm bot-eq towards-inf-compl inf-eq})

\textbf{lemma} \textit{sup-ident}: \[ x \sqcup \bot = x \]
\textbf{by} (\textit{metis dbl-neg huntington inf-compl inf-eq sup-assoc sup-comm sup-compl})

\textbf{lemma} \textit{inf-ident}: \[ x \sqcap \top = x \]
\textbf{by} (\textit{metis dbl-neg inf-compl inf-eq sup-ident sup-comm sup-compl})

\textbf{lemma} \textit{sup-idem}: \[ x \sqcup x = x \]
\textbf{by} (\textit{metis dbl-neg huntington inf-compl inf-eq sup-ident sup-comm sup-compl})

\textbf{lemma} \textit{inf-idem}: \[ x \sqcap x = x \]
\textbf{by} (\textit{metis dbl-neg inf-eq sup-idem})

\textbf{lemma} \textit{sup-nil}: \[ x \sqcup \top = \top \]
\textbf{by} (\textit{metis sup-idem sup-assoc sup-comm sup-compl})

\textbf{lemma} \textit{inf-nil}: \[ x \sqcap \bot = \bot \]
\textbf{by} (\textit{metis dbl-neg inf-compl inf-eq sup-nil sup-comm sup-compl})

\textbf{lemma} \textit{sup-absorb}: \[ x \sqcup x \sqcap y = x \]
\textbf{by} (\textit{metis huntington inf-eq sup-idem sup-assoc sup-comm})

\textbf{lemma} \textit{inf-absorb}: \[ x \sqcap (x \sqcup y) = x \]
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: \(- (x \cup y) = -x \cap -y\)
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
\( \ldots = ((x \cap y \cap z) \cup (x \cap y \cap z) \cup (x \cap y \cap -z)) \cup \)
\( (x \cap ((y \cup z) \cap (-y \cap z))) \) by simp
also from inf-comm have
\[
\ldots = ((x \land y \land z) \lor (x \land y \land z) \lor (x \land y \land -z)) \lor \\
(x \land ((-y \land z) \land (y \lor z))) \text{ by simp}
\]
also from sup-comm have
\[
\ldots = ((x \land y \land z) \lor (x \land y \land z) \lor (x \land y \land -z)) \lor \\
(x \land ((-y \land z) \land (z \lor y))) \text{ by simp}
\]
also from inf-assoc have
\[
\ldots = ((x \land y \land z) \lor (x \land (y \land z)) \lor (x \land y \land -z)) \lor \\
(x \land (-y \land (z \land (z \lor y)))) \text{ by simp}
\]
also from inf-absorb have
\[
\ldots = ((x \land y \land z) \lor (x \land (y \land z)) \lor (x \land y \land -z)) \lor (x \land (-y \land z)) \\
\text{ by simp}
\]
also from inf-comm have
\[
\ldots = ((x \land y \land z) \lor (x \land (y \land z)) \lor (x \land y \land -z)) \lor (x \land (z \land -y)) \\
\text{ by simp}
\]
also from sup-assoc have
\[
\ldots = ((x \land y \land z) \lor ((x \land (z \land y)) \lor (x \land y \land -z))) \lor (x \land (z \land -y)) \\
\text{ by simp}
\]
also from sup-comm have
\[
\ldots = ((x \land y \land z) \lor ((x \land (z \land y)) \lor (x \land y \land -z))) \lor (x \land (z \land -y)) \\
\text{ by simp}
\]
also from sup-assoc have
\[
\ldots = ((x \land y \land z) \lor (x \land y \land -z)) \lor (x \land (z \land y))) \lor (x \land (z \land -y)) \\
\text{ by simp}
\]
also from inf-assoc have
\[
\ldots = ((x \land y \land z) \lor (x \land y \land -z)) \lor ((x \land z \land y) \lor (x \land z \land -y)) \text{ by simp}
\]
also from partition have \[
\ldots = (x \land y) \lor (x \land z) \text{ by simp}
\]
finally show \(\text{thesis by simp}\)
qed

lemma sup-inf-distrib1:
\[
x \lor (y \land z) = (x \lor y) \land (x \lor z)
\]
proof
from dbl-neg have
\[
x \lor (y \land z) = -(-(-x) \lor (y \land z)) \text{ by simp}
\]
also from inf-eq have
\[
\ldots = -(-x \land (-y \lor -z)) \text{ by simp}
\]
also from inf-sup-distrib1 have
\[
\ldots = -((-x \land -y) \lor (-x \land -z)) \text{ by simp}
\]
also from demorgans2 have
\[
\ldots = -(-x \land -y) \lor -(-x \land -z) \text{ by simp}
\]
also from demorgans1 have
\[
\ldots = (-(-x) \lor (-y)) \land (-(-x) \lor -(-z)) \text{ by simp}
\]
also from dbl-neg have
\[
\ldots = (x \lor y) \land (x \lor z) \text{ by simp}
\]
finally show \(\text{thesis by simp}\)
qed

lemma huntington-is-boolean-II:
\[
\text{class boolean-algebra-II aminus (op \land) (op \lor) \land \top}
\]

8
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 (op ⊓) (op ⊑) (op ⊏) (op ⊔) ⊥⊤
proof –
  interpret boolean-II:
  boolean-algebra-II uminus op ⊓ op ⊑ ⊥⊤
  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 uminus 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 uminus inf sup bot top
proof –
  interpret boolean:
  boolean-algebra minus uminus op ⊓ op ⊑ ⊥⊤
  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 uminus inf sup bot top
proof –
  interpret boolean:
  boolean-algebra minus uminus op ⊓ op ⊑ ⊥⊤
  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 copyp :: nat ⇒ 'a ⇒ 'a (infix ⊗ 80)
where
copyp-0: 0 ⊗ x = x
| copyp-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
finally show \(\text{thesis by simp}\)
\[\text{qed}\]

**Lemma five**: \(5 \times x = x \sqcup x \sqcup x \sqcup x \sqcup x\)

**Proof** –
- have \(5 = \text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(0)))))\) by arith
- hence \(5 \times x = \text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(0)))))))) \times x\) by metis
- also have \(\ldots = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\) by simp
- finally show \(\text{thesis by simp}\)
\[\text{qed}\]

**Lemma six**: \(6 \times x = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\)

**Proof** –
- have \(6 = \text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(0))))))))))\) by arith
- hence \(6 \times x = \text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(\text{Suc}(0)))))))))))) \times x\) by metis
- also have \(\ldots = x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x \sqcup x\) by simp
- finally show \(\text{thesis by simp}\)
\[\text{qed}\]

**Lemma copyp-distrib**: \(k \otimes (x \sqcup y) = (k \otimes x) \sqcup (k \otimes y)\)

**Proof** (induct \(k\))
- case 0 show \(\text{case by simp}\)
- case Suc thus \(\text{case by (simp, metis sup-assoc sup-comm)}\)
\[\text{qed}\]

**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 \(\text{case by simp}\)
- case (Suc \(l\)) note \(\text{ind-hyp = this}\)
  - 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: \text{ind-hyp})
  - also from \(\text{ind-hyp}\) have \(\ldots = (k \otimes x) \sqcup (l \otimes x) \sqcup x\) by simp
  - also note \(\text{sup-assoc}\)
  - finally show \(\text{case by simp}\)
\[\text{qed}\]

**Lemma copy-arith**:
- assumes \(k \neq 0\) and \(l \neq 0\)
- shows \((k + l) \times x = (k \times x) \sqcup (l \times x)\)

**Using** \(\text{assms}\)

**Proof** –
- from \(\text{assms}\) have \(\exists k'. \text{Suc}(k') = k\)
and $\exists l'. \text{Suc}(l') = l$ by arith+
from this obtain $k' l'$ where $A: \text{Suc}(k') = k$
and $B: \text{Suc}(l') = l$ by fast+
from this have $A1: k \times x = k' \otimes x$
and $B1: l \times x = l' \otimes x$ by fastforce+
from $A \land B$ have $k + l = \text{Suc}(k' + l' + 1)$ by arith
hence $(k + l) \times x = (k' + l' + 1) \otimes x$ by simp
also from copy-p-arith have
\ldots $= k' \otimes x \sqcup l' \otimes x$ by fast
also from $A1 \land B1$ have
\ldots $= k \times x \sqcup l \times x$ by fastforce
finally show \qth by simp
qed
end

The theorem asserting all Robbins algebras are Boolean comes in 6 moves-
ments.
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.

context robbins-algebra begin

definition secret-object2 :: 'a (α) where
  $\alpha = -(\iota \sqcup \iota \sqcup \iota)$
definition secret-object3 :: 'a (β) where
  $\beta = \iota \sqcup \iota$
definition secret-object4 :: 'a (δ) where
  $\delta = \beta \sqcup -(\alpha \sqcup \beta) \sqcup -(\alpha \sqcup \beta)$
definition secret-object5 :: 'a (γ) where
  $\gamma = \delta \sqcup -(\delta \sqcup \delta)$
definition winker-object :: 'a (ϱ) where
  $\varrho = \gamma \sqcup \gamma \sqcup \gamma$
definition fake-bot :: 'a (⊥⊥) where
  $\bot \bot = -(\varrho \sqcup \neg \varrho)$

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

lemma mann1: \(-(-x \sqcup y) = -(\neg(-(-x \sqcup y) \sqcup x \sqcup y) \sqcup y)\)
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 \(?w = -(\neg (-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y)\)
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: \(-(y \sqcup z) =\)
\(-(-(-(-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(x \sqcup y) \sqcup -(y \sqcup z) \sqcup z) \sqcup -(y \sqcup z)\)
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 \neg (x \sqcup y) \sqcup (y \sqcup z) \sqcup z) \sqcup \neg (y \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
and y=u)]
mann4[where x=x and y=y and z=z]
sup-comm
by metis

lemma mann6:
\(-3 \times x \sqcup x = -(\neg (3 \times x \sqcup x) \sqcup 3 \times x) \sqcup -(\neg (3 \times x \sqcup x) \sqcup 5 \times x)\)
proof
have 3+2=(5::nat) and 3\#(0::nat) and 2\#(0::nat) by arith+
with copy-arith have \(\ominus: 3 \times x \sqcup 2 \times x = 5 \times x\) by metis
let \(?p = -(\neg (3 \times x \sqcup x)\)
{ fix q
from sup-comm have
\(-(q \sqcup 5 \times x) = -(3 \times x \sqcup q)\) by metis
13
also from \(\triangledown\) \textit{mann0}\[where\] \(x=3\times x\) and \(y=q \cup 2\times x\) \textit{sup-assoc} \textit{sup-comm}

\[
\ldots = -(-((3\times x \cup (q \cup 2\times x)) \cup 3\times x \cup (q \cup 2\times x)) \cup (q \cup 2\times x))
\]

by \textsf{metis}

also from \textit{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 \textsf{metis}

also from \textit{sup-comm} have

\[
\ldots = -(-((q \cup 3\times x) \cup 2\times x) \cup -3\times x \cup (q \cup 2\times x)) \cup (q \cup 2\times x))\]

by \textsf{metis}

also from \textit{sup-assoc} have

\[
\ldots = -(-q \cup (3\times x \cup 2\times x)) \cup -3\times x \cup (q \cup 2\times x)) \cup (q \cup 2\times x))\]

by \textsf{metis}

also from \(\triangledown\) have

\[
\ldots = -(-q \cup 5\times x) \cup -3\times x \cup (q \cup 2\times x)) \cup (q \cup 2\times x))\]

by \textsf{metis}

also from \textit{sup-assoc} have

\[
\ldots = -(-q \cup 5\times x) \cup (-3\times x \cup q) \cup 2\times x) \cup (q \cup 2\times x))\]

by \textsf{metis}

also from \textit{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 \textsf{metis}

also from \textit{sup-assoc} have

\[
\ldots = -(-q \cup 5\times x) \cup q \cup -3\times x \cup 2\times x) \cup 2\times x \cup q)\]

by \textsf{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 \textsf{simp}

} hence \(\blacklozenge\):

\[
-(?p \cup 5\times x) = -(-(-?p \cup 5\times x) \cup ?p \cup -3\times x \cup 2\times x) \cup 2\times x \cup ?p)
\]

by \textsf{simp}

from \textit{mann5}\[where\] \(x=3\times x\) and \(y=x\) and \(z=2\times x\) and \(u=?p\) \textit{sup-assoc} \textit{three}\[where\] \(x=x\) \textit{five}\[where\] \(x=x\) have

\[
?p =
\]

\[-(-(-(?p \cup 5\times x) \cup ?p \cup -(x \cup 2\times x) \cup 2\times x) \cup 2\times x \cup ?p) \cup
\]

\[-(-x \cup 2\times x) \cup ?p))\]

by \textsf{metis}

also from \textit{sup-comm} have

\[
\ldots =
\]

\[-(-(-(?p \cup 5\times x) \cup ?p \cup -(2\times x \cup x) \cup 2\times x) \cup 2\times x \cup ?p) \cup
\]

\[-(-2\times x \cup x) \cup ?p))\]

by \textsf{metis}

also from \textit{two}\[where\] \(x=x\) \textit{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 \textsf{metis}

also from \(\blacklozenge\) have \(\ldots = -(?p \cup 5\times x) \cup -(3\times x \cup ?p))\]

by \textsf{simp}

also from \textit{sup-comm} have \(\ldots = -(-?p \cup 5\times x) \cup -(?p \cup -3\times x))\]

by \textsf{simp}

also from \textit{sup-comm} have \(\ldots = -(-?p \cup -3\times x) \cup -(?p \cup 5\times x))\]

by \textsf{simp}

finally show \(\textit{thesis}\).

\textsf{qed}

\textbf{lemma} \textit{mann7}:

\[
-3\times x = -(-(3\times x \cup x) \cup 5\times x)
\]

\textbf{proof} –
let ?p = −(−3 × x ∪ x)
let ?q = ?p ∪ −3 × x
let ?r = −(?p ∪ 5 × x)

from robbins2[where x=?q and y=?r]
mann6[where x=x]

have ?r = − (?p ∪ − (?q ∪ ?r)) by simp
also from sup-comm have ... = − (?r ∪ ?q) ∪ ?p by simp
also from sup-comm have ... = − (?r ∪ ?q) ∪ ?p by simp
finally have ◊: ?r = − (?r ∪ ?q) ∪ ?p).

from mann3[where x=3 × x and y=x and z=− 3 × x]
sup-comm have
− 3 × x = −(− (?p ∪ 3 × x ∪ x ∪ x) ∪ ?p ∪ − 3 × x) ∪ ?p by metis
also from sup-assoc have
... = − (?r ∪ ?q) ∪ ?p by metis
also from three[where x=x] five[where x=x] have
... = − (?r ∪ ?q) ∪ ?p by metis
finally have 3 × x = − (?r ∪ ?q) ∪ ?p by metis

with ◊ show ?thesis by simp

qed

lemma mann8:
− (−3 × x ∪ x) ∪ 2 × x = − (− (−3 × x ∪ x) ∪ −3 × x ∪ 2 × x) ∪ −3 × x
(is ?lhs = ?rhs)

proof –

let ?p = − (−3 × x ∪ x)
let ?q = ?p ∪ 2 × x
let ?r = 3 × x

have 3+2=(5::nat) and 3≠(0::nat) and 2≠(0::nat) by arith+
with copy-arith have ⊔: 3 × x ∪ 2 × x = 5 × x by metis
from robbins2[where x=?r and y=?q] and sup-assoc
have ?q = − (−3 × x ∪ ?q) ∪ −(3 × x ∪ ?p ∪ 2 × x) by metis
also from sup-comm have
... = − (?q ∪ 3 × x) ∪ −(?p ∪ 3 × x ∪ 2 × x) by metis
also from ⊔ sup-assoc have
... = − (?q ∪ 3 × x) ∪ − (?p ∪ 5 × x)) by metis
also from mann7[where x=x] have
... = − (?q ∪ 3 × x) ∪ −3 × x by metis
also from sup-assoc have
... = − (?p ∪ (2 × x ∪ 3 × x)) ∪ −3 × x by metis
also from sup-comm have
... = − (?p ∪ (−3 × x ∪ 2 × x)) ∪ −3 × x by metis
also from sup-assoc have
... = ?rhs by metis

finally show ?thesis by simp

qed

lemma mann9: x = − (−(−3 × x ∪ x) ∪ −3 × x)

proof –
let \( p = -3x \) \\
let \( q = p \uplus 4x \) \\
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 \( \triangledown: 4x \uplus x = 5x \) by metis \\
with sup-assoc robbins2[where \( y=x \) and \( x=?q \)] \\
have \( x = -(-(-?q \uplus x) \uplus -(?p \uplus 5x)) \) by metis \\
with mann7 have \( x = -(-(?q \uplus x) \uplus -3x) \) 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 \( \lozenge: 3x \uplus x = 4x \) by metis \\
with mann1[where \( x=3x \) and \( y=x \)] sup-assoc have \\
\( -(?q \uplus x) = ?p \) by metis \\
ultimately show \( \text{thesis} \) by simp \\
qed 

lemma mann10: \( y = -(-(-3x \uplus x) \uplus -3x \uplus y) \uplus -(x \uplus y)) \) \\
using robbins2[where \( x=3x \uplus y \) ] \\
mann9[where \( x=x \)] sup-comm \\
by metis 

theorem mann: \( 2 \times x = -(-3x \uplus x) \uplus 2x \) \\
using mann10[where \( x=x \) and \( y=2x \) ] \\
mann8[where \( x=x \)] \\
two[where \( x=x \)] three[where \( x=x \)] sup-comm \\
by metis 

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

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

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

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

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

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

lemma eq-intro:
\[-(x \sqcup -(y \sqcup z)) = -(y \sqcup -(x \sqcup z)) \implies x = y\]
by (metis robbins sup-assoc sup-comm)

lemma copyp0:

assumes \(- (x \sqcup - y) = z\)
sows \(- (x \sqcup -(y \sqcup k \ominus (x \sqcup z))) = z\)

using assms
proof (induct k)
case 0 show ?case
by (simp, metis assms 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 \(- (\neg (x \sqcup - y) \sqcup - y) = x\)
sows \(- (y \sqcup k \ominus (x \sqcup -(x \sqcup - y))) = - y\)

using assms
proof
let ?z = \(- (x \sqcup - y)\)
let ?ky = y \sqcup k \ominus (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 \(?z = -(\neg y \sqcup -(\neg ?ky \sqcup ?z))\) by (metis sup-comm)
finally show \(\neg \thesis\) by (metis eq-intro)

qed

corollary copyp2:

assumes \(- (x \sqcup y) = - y\)
sows \(- (y \sqcup k \ominus (x \sqcup -(x \sqcup - y))) = - y\)
by (metis assms robbins sup-comm copyp1)

lemma two-threep:

assumes \(- (2 \times x \sqcup y) = - y\)
and \(- (3 \times x \sqcup y) = - y\)
sows \(2 \times x \sqcup y = 3 \times x \sqcup y\)

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
  copyp2[where x=x ⊔ x and y=y and k=0]
have −(y ⊔ x ⊔ x) = y by fastforce
  with sup-comm sup-assoc
have −(x ⊔ y ⊔ x ⊔ (x ⊔ −y))) = −y by metis
ultimately have
  −(x ⊔ x ⊔ y ⊔ (x ⊔ −y))) = −(x ⊔ x ⊔ y ⊔ x ⊔ −(x ⊔ −y)) by simp
with less-eq-intro have x ⊔ x ⊔ y = x ⊔ x ⊔ y ⊔ x by metis
with sup-comm sup-assoc two three show ?thesis by metis
qed

lemma two-three:
  assumes −(x ⊔ y) = −y ∨ −(−(x ⊔ −y) ⊔ −y) = x
  shows y ⊔ 2 × (x ⊔ −(x ⊔ −y)) = y ⊔ 3 × (x ⊔ −(x ⊔ −y))
  (is y ⊔ ?z2 = y ⊔ ?z3)
using assms
proof
  assume −(x ⊔ y) = −y
  with copyp2[where k=Suc(0)]
    copyp2[where k=Suc(Suc(0))]
    two three
  have −(y ⊔ ?z2) = −y and −(y ⊔ ?z3) = −y by simp+
  with two-three sup-comm show ?thesis by metis
next
  assume −(−(x ⊔ −y) ⊔ −y) = x
  with copyp1[where k=Suc(0)]
    copyp1[where k=Suc(Suc(0))]
    two three
  have −(y ⊔ ?z2) = −y and −(y ⊔ ?z3) = −y by simp+
  with two-three sup-comm show ?thesis by metis
qed

lemma sup-idem: ϱ ⊔ ϱ = ϱ
proof –
  from winkerr two
    copyp2[where x=α and y=β and k=Suc(0)] have
    −β = −(β ⊔ 2 × (α ⊔ −(α ⊔ −β))) by simp
also from copy-distrib sup-assoc have
    ... = −(β ⊔ 2 × α ⊔ 2 × (−(α ⊔ −β))) by simp
also from sup-assoc secret-object4-def two
    multi-winker[where k=2] have
    ... = −δ by metis
finally have −β = −δ by simp
  with secret-object4-def sup-assoc three have
  δ ⊔ −(α ⊔ −δ) = β ⊔ 3 × (−(α ⊔ −β)) by simp
also from copy-distrib[where k=3]
  multi-winker[where k=3]
  sup-assoc have

18
\[ \ldots = \beta \sqcup 2 \times (\alpha \sqcup -(\alpha \sqcup -\beta)) \] by \textit{metis}

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

\[ \ldots = \beta \sqcup 2 \times (\alpha \sqcup -(\alpha \sqcup -\beta)) \] by \textit{fastforce}

also from \textit{copy-distrib[where \(k=2\]}

\[
\text{multi-winker[where } k=2]
\text{ sup-assoc two secret-object\textunderscore 4-def have}
\]

\[ \ldots = \delta \] by \textit{metis}

finally have \(\bigtriangledown: \delta \sqcup -(\alpha \sqcup -\delta) = \delta\) by \textit{simp}

from \textit{secret-object\textunderscore 4-def winkerr sup-assoc have}

\[ \alpha \sqcup \delta = \delta \] by \textit{metis}

hence \(\delta \sqcup \alpha = \delta\) by (\textit{metis sup-comm})

hence \(-(-\delta \sqcup -\delta) \sqcup -\delta) = -(-\delta \sqcup (\alpha \sqcup -\delta)) \sqcup -\delta)\) by (\textit{metis sup-assoc})

also from \(\bigvee\) have

\[ \ldots = -(-\delta \sqcup (\alpha \sqcup -\delta)) \sqcup -(-\delta \sqcup -(\alpha \sqcup -\delta))\] by \textit{metis}

also from \textit{robbins have}

\[ \ldots = \delta \] by \textit{metis}

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

with \textit{two-three[where \(x=\delta\) and \(y=\delta\]}

\[
\text{secret-object\textunderscore 5-def sup-comm have}
\]

\[ 3 \times \gamma \sqcup \delta = 2 \times \gamma \sqcup \delta \] by \textit{fastforce}

with \textit{secret-object\textunderscore 5-def sup-assoc sup-comm have}

\[ 3 \times \gamma \sqcup \gamma = 2 \times \gamma \sqcup \gamma \] by \textit{fastforce}

with \textit{two three four five six have}

\[ 6 \times \gamma = 3 \times \gamma \] by \textit{simp}

moreover have \(3 + 3 = (6::nat)\) and \(3 \neq (0::nat)\) by \textit{arith+}

moreover note \textit{copy-arith[where \(k=3\) and \(l=3\) and \(x=\gamma\]}

\[
\text{winker-object-def three}
\]

ultimately show \(?thesis\) by \textit{simp}

\[
\text{qed}
\]

\[
\text{lemma sup-ident: } x \sqcup \bot \bot = x
\]

\[
\text{proof} -
\]

\[
\text{have I: } \rho = -(-\rho \sqcup \bot \bot)
\]

by \(\textit{metis fake-bot-def inf-eq robbins sup-comm sup-idem}\)

\[
\{ \text{fix } x \text{ have } x = -(-(x \sqcup -\rho \sqcup \bot \bot) \sqcup -(x \sqcup \rho)) \}
\]

by \(\textit{metis I robbins sup-assoc}\}

\[
\text{note II is this}
\]

\[
\text{have III: } -\rho = -(-\rho \sqcup -\rho \sqcup -\rho) \sqcup \rho
\]

by \(\textit{metis robbins[where } x=-\rho\text{ and } y=\rho \sqcup -\rho\]

\[
\text{I sup-comm fake-bot-def}\)
\]

hence \(\rho = -(-\rho \sqcup -\rho \sqcup -\rho) \sqcup -\rho\)

by \(\textit{metis robbins[where } x=\rho\text{ and } y=\rho \sqcup -\rho \sqcup -\rho\]

\[
\text{sup-comm[where } x=\rho\text{ and } y=-(-\rho \sqcup -\rho \sqcup -\rho)]
\]

\[
\text{sup-assoc sup-idem}\)
\]

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

lemma dbl-neg: \(-(-x) = x\)
proof –
{ fix \( x \) have \( \bot \bot = -(\bot \bot \sqcup -(-x)) \)
  by (metis robbins sup-comm sup-ident)
} note I = this

{ fix \( x \) have \( -x = -(\bot \bot \sqcup -(\bot \bot)) \)
  by (metis I robbins sup-comm sup-ident)
} note II = this

{ fix \( x \) have \( -(\bot \bot) = -(\bot \bot \sqcup -(\bot \bot)) \)
  by (metis I II robbins sup-assoc sup-comm sup-ident)
} note III = this

show \( \text{thesis} \) by (metis III robbins)
qed

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

theorem robbins-is-boolean-II:
  class.boolean-algebra-II uminus (op \( \sqcap \)) (op \( \sqcup \)) \( \bot \top \)
proof –
interpret huntington:
huntington-algebra $\text{uminus} \quad \cap \quad \cup \quad \bot \quad T$

by (fact robbins-is-huntington)

show $\text{thesis}$ by (simp add: huntington.huntington-is-boolean-II)

qed

theorem robbins-is-boolean:

class boolean-algebra $\text{minus} \quad \text{uminus} \quad (\cap) \quad (\cup) \quad (\bot) \quad (\top)$

proof

interpret huntington:

huntington-algebra $\text{uminus} \quad \cap \quad \cup \quad \bot \quad T$

by (fact robbins-is-huntington)

show $\text{thesis}$ 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 ($\rho$)

and less-eq (infix $\subseteq$ 50)

and less (infix $\subset$ 50)

and inf (infix $\cap$ 70)

and sup (infix $\cup$ 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