A Complete Proof of the Robbins Conjecture

Matthew Wampler-Doty

December 14, 2021

Abstract

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

Contents

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* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\sqcap$ *70*)
  fixes *sup* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $\sqcup$ *65*)
  fixes *bot* :: $'a$ ($\bot$)
  fixes *top* :: $'a$ ($\top$)
  assumes *sup-assoc*: $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$
  assumes *sup-comm*: $x \sqcup y = y \sqcup x$

context *common-algebra* **begin**

definition *less-eq* :: $'a \Rightarrow 'a \Rightarrow$ *bool* (**infix** $\sqsubseteq$ *50*) **where**
  $x \sqsubseteq y = (x \sqcup y = y)$
definition *less* :: $'a \Rightarrow 'a \Rightarrow$ *bool* (**infix** $\sqsubset$ *50*) **where**
  $x \sqsubset y = (x \sqsubseteq y \wedge \neg\, y \sqsubseteq x)$
definition *minus* :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixl** $-$ *65*) **where**
  minus $x\ y = (x \sqcap -\ y)$

definition *secret-object1* :: $'a$ (ι) **where**
  $\iota = (SOME\ x.\ True)$

end

class *ext-common-algebra* = *common-algebra* +
  assumes *inf-eq*: $x \sqcap y = -(-\ x \sqcup -\ y)$
  assumes *top-eq*: $\top = \iota \sqcup -\ \iota$
  assumes *bot-eq*: $\bot = -(\iota \sqcup -\ \iota)$

2.2 Boolean Algebra

class *boolean-algebra-II* =
  common-algebra +
  assumes *inf-comm*: $x \sqcap y = y \sqcap x$

assumes *inf-assoc*: $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$
assumes *sup-absorb*: $x \sqcup (x \sqcap y) = x$
assumes *inf-absorb*: $x \sqcap (x \sqcup y) = x$
assumes *sup-inf-distrib1*: $x \sqcup y \sqcap z = (x \sqcup y) \sqcap (x \sqcup z)$
assumes *sup-compl*: $x \sqcup - x = \top$
assumes *inf-compl*: $x \sqcap - x = \bot$

2.3 Huntington's Algebra

class *huntington-algebra* = *ext-common-algebra* +
assumes *huntington*: $- (-x \sqcup -y) \sqcup - (-x \sqcup \ y) = x$

2.4 Robbins' Algebra

class *robbins-algebra* = *ext-common-algebra* +
assumes *robbins*: $- (- (x \sqcup y) \sqcup - (x \sqcup -y)) = x$

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 ($\sqcap$) ($\sqsubseteq$) ($\sqsubset$) ($\sqcup$) $\bot$ $\top$
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 *boolean-algebra-II* **begin**

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

end

context *huntington-algebra* **begin**

lemma *huntington-id*: $x \sqcup -x = -x \sqcup -(-x)$
proof −

from *huntington* **have**
$x \sqcup -x = -(-x \sqcup -(-(-x))) \sqcup -(-x \sqcup -(-x)) \sqcup$
$(-(-(-x) \sqcup -(-(-x))) \sqcup -(-(-x) \sqcup -(-x)))$
by *simp*
also from *sup-comm* **have**
$\ldots = -(-(-x) \sqcup -(-x)) \sqcup -(-(-x) \sqcup -(-(-x))) \sqcup$
$(-(-(-x) \sqcup -x) \sqcup -(-(-(-x)) \sqcup -x))$
by *simp*
also from *sup-assoc* **have**
$\ldots = -(-(-x) \sqcup -(-x)) \sqcup$
$(-(-(-x) \sqcup -(-(-x))) \sqcup -(-(-x) \sqcup -x)) \sqcup$
$-(-(-(-x)) \sqcup -x)$
by *simp*
also from *sup-comm* **have**
$\ldots = -(-(-x) \sqcup -(-x)) \sqcup$
$(-(-(-x) \sqcup -x) \sqcup -(-(-x) \sqcup -(-(-x)))) \sqcup$
$-(-(-(-x)) \sqcup -x)$
by *simp*
also from *sup-assoc* **have**
$\ldots = -(-(-x) \sqcup -(-x)) \sqcup -(-(-x) \sqcup -x) \sqcup$
$(-(-(-x) \sqcup -(-(-x))) \sqcup -(-(-(-x)) \sqcup -x))$
by *simp*
also from *sup-comm* **have**
$\ldots = -(-(-x) \sqcup -(-x)) \sqcup -(-(-x) \sqcup -x) \sqcup$
$(-(-(-(-x)) \sqcup -(-x)) \sqcup -(-(-(-x)) \sqcup -x))$
by *simp*
also from *huntington* **have**
$\ldots = -x \sqcup -(-x)$
by *simp*
finally show *?thesis* **by** *simp*
qed

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

lemma *towards-sup-compl*: $x \sqcup -x = y \sqcup -y$
proof −
from *huntington* **have**
$x \sqcup -x = -(-x \sqcup -(-y)) \sqcup -(-x \sqcup -y) \sqcup (-(-(-x) \sqcup -(-y)) \sqcup -(-(-x)$
$\sqcup -y))$
by *simp*
also from *sup-comm* **have**
$\ldots = -(-(-y) \sqcup -x) \sqcup -(-y \sqcup -x) \sqcup (-(-y \sqcup -(-x)) \sqcup -(-(-y) \sqcup -(-x)))$
by *simp*
also from *sup-assoc* **have**
$\ldots = -(-(-y) \sqcup -x) \sqcup (-(-y \sqcup -x) \sqcup -(-y \sqcup -(-x))) \sqcup -(-(-y) \sqcup -(-x))$
by *simp*
also from *sup-comm* **have**

$\ldots = -(-y \sqcup -(-x)) \sqcup -(-y \sqcup -x) \sqcup -(-(-y) \sqcup -x) \sqcup -(-(-y) \sqcup -(-x))$
by *simp*
also from *sup-assoc* **have**
$\ldots = -(-y \sqcup -(-x)) \sqcup -(-y \sqcup -x) \sqcup (-(-(-y) \sqcup -x) \sqcup -(-(-y) \sqcup -(-x)))$
by *simp*
also from *sup-comm* **have**
$\ldots = -(-y \sqcup -(-x)) \sqcup -(-y \sqcup -x) \sqcup (-(-(-y) \sqcup -(-x)) \sqcup -(-(-y) \sqcup -x))$
by *simp*
also from *huntington* **have**
$y \sqcup -y = \ldots$ **by** *simp*
finally show *?thesis* **by** *simp*
qed

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

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

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

lemma *towards-idem*: $\bot = \bot \sqcup \bot$
by (*metis dbl-neg huntington inf-compl inf-eq sup-assoc sup-comm sup-compl*)

lemma *sup-ident*: $x \sqcup \bot = x$
by (*metis dbl-neg huntington inf-compl inf-eq sup-assoc*
sup-comm sup-compl towards-idem)

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

lemma *sup-idem*: $x \sqcup x = x$
by (*metis dbl-neg huntington inf-compl inf-eq sup-ident sup-comm sup-compl*)

lemma *inf-idem*: $x \sqcap x = x$
by (*metis dbl-neg inf-eq sup-idem*)

lemma *sup-nil*: $x \sqcup \top = \top$
by (*metis sup-idem sup-assoc sup-comm sup-compl*)

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

lemma *sup-absorb*: $x \sqcup x \sqcap y = x$
by (*metis huntington inf-eq sup-idem sup-assoc sup-comm*)

lemma *inf-absorb*: $x \sqcap (x \sqcup y) = x$
by (*metis dbl-neg inf-eq sup-absorb*)

lemma *partition*: $x \sqcap y \sqcup x \sqcap -y = x$
by (*metis dbl-neg huntington inf-eq sup-comm*)

lemma *demorgans1*: $-(x \sqcap y) = -x \sqcup -y$
by (*metis dbl-neg inf-eq*)

lemma *demorgans2*: $-(x \sqcup y) = -x \sqcap -y$
by (*metis dbl-neg inf-eq*)

lemma *inf-comm*: $x \sqcap y = y \sqcap x$
by (*metis inf-eq sup-comm*)

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

lemma *inf-sup-distrib1*: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$
proof −
from *partition* **have**
$x \sqcap (y \sqcup z) = x \sqcap (y \sqcup z) \sqcap y \sqcup x \sqcap (y \sqcup z) \sqcap -y$ **..**
also from *inf-assoc* **have**
$\ldots = x \sqcap ((y \sqcup z) \sqcap y) \sqcup x \sqcap (y \sqcup z) \sqcap -y$ **by** *simp*
also from *inf-comm* **have**
$\ldots = x \sqcap (y \sqcap (y \sqcup z)) \sqcup x \sqcap (y \sqcup z) \sqcap -y$ **by** *simp*
also from *inf-absorb* **have**
$\ldots = (x \sqcap y) \sqcup (x \sqcap (y \sqcup z) \sqcap -y)$ **by** *simp*
also from *partition* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap (y \sqcup z) \sqcap -y \sqcap z) \sqcup (x \sqcap (y \sqcup z) \sqcap -y \sqcap -z))$ **by** *simp*
also from *inf-assoc* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z))) \sqcup (x \sqcap ((y \sqcup z) \sqcap (-y \sqcap -z))))$ **by** *simp*
also from *demorgans2* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z))) \sqcup (x \sqcap ((y \sqcup z) \sqcap -(y \sqcup z))))$ **by** *simp*
also from *inf-compl* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z))) \sqcup (x \sqcap \bot))$ **by** *simp*
also from *inf-nil* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z))) \sqcup \bot)$ **by** *simp*
also from *sup-idem* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$((x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z))) \sqcup \bot)$ **by** *simp*
also from *sup-ident* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$(x \sqcap ((y \sqcup z) \sqcap (-y \sqcap z)))$ **by** *simp*
also from *inf-comm* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$

$(x \sqcap ((-y \sqcap z) \sqcap (y \sqcup z)))$ **by** *simp*
also from *sup-comm* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$(x \sqcap ((-y \sqcap z) \sqcap (z \sqcup y)))$ **by** *simp*
also from *inf-assoc* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap (y \sqcap z)) \sqcup (x \sqcap y \sqcap -z)) \sqcup$
$(x \sqcap (-y \sqcap (z \sqcap (z \sqcup y))))$ **by** *simp*
also from *inf-absorb* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap (y \sqcap z)) \sqcup (x \sqcap y \sqcap -z)) \sqcup (x \sqcap (-y \sqcap z))$
by *simp*
also from *inf-comm* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap (z \sqcap y)) \sqcup (x \sqcap y \sqcap -z)) \sqcup (x \sqcap (z \sqcap -y))$
by *simp*
also from *sup-assoc* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup ((x \sqcap (z \sqcap y)) \sqcup (x \sqcap y \sqcap -z))) \sqcup (x \sqcap (z \sqcap -y))$
by *simp*
also from *sup-comm* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup ((x \sqcap y \sqcap -z) \sqcup (x \sqcap (z \sqcap y)))) \sqcup (x \sqcap (z \sqcap -y))$
by *simp*
also from *sup-assoc* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup ((x \sqcap (z \sqcap y)) \sqcup (x \sqcap (z \sqcap -y)))$
by *simp*
also from *inf-assoc* **have**
$\ldots = ((x \sqcap y \sqcap z) \sqcup (x \sqcap y \sqcap -z)) \sqcup ((x \sqcap z \sqcap y) \sqcup (x \sqcap z \sqcap -y))$ **by** *simp*
also from *partition* **have** $\ldots = (x \sqcap y) \sqcup (x \sqcap z)$ **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *sup-inf-distrib1*:
$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$
proof −
from *dbl-neg* **have**
$x \sqcup (y \sqcap z) = -(-(-(-x) \sqcup (y \sqcap z)))$ **by** *simp*
also from *inf-eq* **have**
$\ldots = -(-x \sqcap (-y \sqcup -z))$ **by** *simp*
also from *inf-sup-distrib1* **have**
$\ldots = -((-x \sqcap -y) \sqcup (-x \sqcap -z))$ **by** *simp*
also from *demorgans2* **have**
$\ldots = -(-x \sqcap -y) \sqcap -(-x \sqcap -z)$ **by** *simp*
also from *demorgans1* **have**
$\ldots = (-(-x) \sqcup -(-y)) \sqcap (-(-x) \sqcup -(-z))$ **by** *simp*
also from *dbl-neg* **have**
$\ldots = (x \sqcup y) \sqcap (x \sqcup z)$ **by** *simp*
finally show *?thesis* **by** *simp*
qed

lemma *huntington-is-boolean-II*:
class.boolean-algebra-II uminus $(\sqcap)$ $(\sqcup)$ $\bot$ $\top$
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 (⊓) (⊑) (⊏) (⊔) ⊥ ⊤
proof −
interpret *boolean-II*:
boolean-algebra-II uminus (⊓) (⊔) ⊥ ⊤
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 (⊓) (⊑) (⊏) (⊔) ⊥ ⊤
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 (⊓) (⊑) (⊏) (⊔) ⊥ ⊤
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 *?thesis* **by** *simp*

qed

lemma *five*: $5 \times x = x \sqcup x \sqcup x \sqcup x \sqcup x$
proof −
  have $5 = Suc(Suc(Suc(Suc(Suc(0)))))$ **by** *arith*
  hence $5 \times x = Suc(Suc(Suc(Suc(Suc(0))))) \times x$ **by** *metis*
  also have $\ldots = x \sqcup x \sqcup x \sqcup x \sqcup x$ **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 $6 = Suc(Suc(Suc(Suc(Suc(Suc(0))))))$ **by** *arith*
  hence $6 \times x = Suc(Suc(Suc(Suc(Suc(Suc(0)))))) \times x$ **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 *copy-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** $k + 0 + 1 = Suc(k)$ **by** *arith*
    thus *?case* **by** *simp*
  case (*Suc l*) **note** *ind-hyp* = *this*
    have $k + Suc(l) + 1 = Suc(k + l + 1)$ **by** *arith*+
    hence $(k + Suc(l) + 1) \otimes x = (k + l + 1) \otimes x \sqcup x$ **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** $\exists\ k'.\ Suc(k') = k$
            and $\exists\ l'.\ Suc(l') = l$ **by** *arith*+

from *this* **obtain** k' l' **where** *A*: $Suc(k') = k$
and *B*: $Suc(l') = l$ **by** *fast+*
from *this* **have** *A1*: $k \times x = k' \otimes x$
and *B1*: $l \times x = l' \otimes x$ **by** *fastforce+*
from *A B* **have** $k + l = Suc(k' + l' + 1)$ **by** *arith*
hence $(k + l) \times x = (k' + l' + 1) \otimes x$ **by** *simp*
also from *copyp-arith* **have**
$\ldots = k' \otimes x \sqcup l' \otimes x$ **by** *fast*
also from *A1 B1* **have**
$\ldots = k \times x \sqcup l \times x$ **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.

context *robbins-algebra* **begin**

definition *secret-object2* :: $'a$ (α) **where**
$\alpha = -(-(\iota \sqcup \iota \sqcup \iota) \sqcup \iota)$
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* :: $'a$ $(\bot\bot)$ **where**
$\bot\bot = -(\varrho \sqcup -\varrho)$

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

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

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 $-(y \sqcup z) =$
$-(z \sqcup -(-(-(-x \sqcup y) \sqcup x \sqcup y \sqcup y) \sqcup -(-x \sqcup y) \sqcup z \sqcup -(y \sqcup z)))$
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 $3{+}2{=}(5{::}nat)$ **and** $3{\neq}(0{::}nat)$ **and** $2{\neq}(0{::}nat)$ **by** *arith+*
with *copy-arith* **have** ♡: $3{\times}x \sqcup 2{\times}x = 5{\times}x$ **by** *metis*
let *?p* $= -(-\ 3{\times}x \sqcup x)$
{ fix q
from *sup-comm* **have**
$-(q \sqcup 5{\times}x) = -(5{\times}x \sqcup q)$ **by** *metis*
also from ♡ *mann0*[**where** $x{=}3{\times}x$ **and** $y{=}q \sqcup 2{\times}x$] *sup-assoc sup-comm*

have
  $\ldots = -(-(-(3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup - 3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup (q \sqcup 2{\times}x))$
    by *metis*
  also from *sup-assoc* **have**
  $\ldots = -(-(-((3{\times}x \sqcup q) \sqcup 2{\times}x) \sqcup - 3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup (q \sqcup 2{\times}x))$ **by** *metis*
  also from *sup-comm* **have**
  $\ldots = -(-(-((q \sqcup 3{\times}x) \sqcup 2{\times}x) \sqcup - 3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup (q \sqcup 2{\times}x))$ **by** *metis*
  also from *sup-assoc* **have**
  $\ldots = -(-(-(q \sqcup (3{\times}x \sqcup 2{\times}x)) \sqcup - 3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup (q \sqcup 2{\times}x))$ **by** *metis*
  also from ♡ **have**
  $\ldots = -(-(-(q \sqcup 5{\times}x) \sqcup - 3{\times}x \sqcup (q \sqcup 2{\times}x)) \sqcup (q \sqcup 2{\times}x))$ **by** *metis*
  also from *sup-assoc* **have**
  $\ldots = -(-(-(q \sqcup 5{\times}x) \sqcup (- 3{\times}x \sqcup q) \sqcup 2{\times}x) \sqcup (q \sqcup 2{\times}x))$ **by** *metis*
  also from *sup-comm* **have**
  $\ldots = -(-(-(q \sqcup 5{\times}x) \sqcup (q \sqcup - 3{\times}x) \sqcup 2{\times}x) \sqcup (2{\times}x \sqcup q))$ **by** *metis*
  also from *sup-assoc* **have**
  $\ldots = -(-(-(q \sqcup 5{\times}x) \sqcup q \sqcup - 3{\times}x \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup q)$ **by** *metis*
  finally have
  $-(q \sqcup 5{\times}x) = -(-(-(q \sqcup 5{\times}x) \sqcup q \sqcup - 3{\times}x \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup q)$ **by** *simp*
 } hence ♠:
  $-(?p \sqcup 5{\times}x) = -(-(-(?p \sqcup 5{\times}x) \sqcup ?p \sqcup - 3{\times}x \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup ?p)$
  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**
 $?p =$
  $-(-(-(-(?p \sqcup 5{\times}x) \sqcup ?p \sqcup -(x \sqcup 2{\times}x) \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup ?p) \sqcup$
    $-(-(x \sqcup 2{\times}x) \sqcup ?p))$ **by** *metis*
 also from *sup-comm* **have**
 $\ldots =$
  $-(-(-(-(?p \sqcup 5{\times}x) \sqcup ?p \sqcup -(2{\times}x \sqcup x) \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup ?p) \sqcup$
    $-(-(2{\times}x \sqcup x) \sqcup ?p))$ **by** *metis*
 also from *two*[**where** $x{=}x$] *three*[**where** $x{=}x$] **have**
 $\ldots =$
  $-(-(-(-(?p \sqcup 5{\times}x) \sqcup ?p \sqcup - 3{\times}x \sqcup 2{\times}x) \sqcup 2{\times}x \sqcup ?p) \sqcup$
    $-(- 3{\times}x \sqcup ?p))$ **by** *metis*
 also from ♠ **have** $\ldots = -(-(?p \sqcup 5{\times}x) \sqcup -(- 3{\times}x \sqcup ?p))$ **by** *simp*
 also from *sup-comm* **have** $\ldots = -(-(?p \sqcup 5{\times}x) \sqcup -(?p \sqcup - 3{\times}x))$ **by** *simp*
 also from *sup-comm* **have** $\ldots = -(-(?p \sqcup - 3{\times}x) \sqcup -(?p \sqcup 5{\times}x))$ **by** *simp*
 finally show *?thesis* **.**
qed

lemma *mann7*:
$- 3{\times}x = -(-(- 3{\times}x \sqcup x) \sqcup 5{\times}x)$
proof −
 let $?p = -(- 3{\times}x \sqcup x)$

```
  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 ... = − (− (?q ⊔ ?r) ⊔ ?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
       ... = −(−(−(?p ⊔ (3×x ⊔ x ⊔ x)) ⊔ ?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 = −(− 3×x ⊔ x)
```

let *?q* = *?p* ⊔ *4*×*x*
have *4*+*1*=(*5*::*nat*) **and** *1*≠(*0*::*nat*) **and** *4*≠(*0*::*nat*) **by** *arith*+
with *copy-arith one* **have** ♡: *4*×*x* ⊔ *x* = *5*×*x* **by** *metis*
with *sup-assoc robbins2*[**where** *y*=*x* **and** *x*=*?q*]
have *x* = −(−(−*?q* ⊔ *x*) ⊔ −(*?p* ⊔ *5*×*x*)) **by** *metis*
with *mann7* **have** *x* = −(−(−*?q* ⊔ *x*) ⊔ − *3*×*x*) **by** *metis*
moreover
have *3*+*1*=(*4*::*nat*) **and** *1*≠(*0*::*nat*) **and** *3*≠(*0*::*nat*) **by** *arith*+
with *copy-arith one* **have** ♠: *3*×*x* ⊔ *x* = *4*×*x* **by** *metis*
with *mann1*[**where** *x*=*3*×*x* **and** *y*=*x*] *sup-assoc* **have**
−(−*?q* ⊔ *x*) = *?p* **by** *metis*
ultimately show *?thesis* **by** *simp*
qed

lemma *mann10*: *y* = −(−(−(− *3*×*x* ⊔ *x*) ⊔ − *3*×*x* ⊔ *y*) ⊔ −(*x* ⊔ *y*))
using *robbins2*[**where** *x*=−(− *3*×*x* ⊔ *x*) ⊔ − *3*×*x* **and** *y*=*y*]
mann9[**where** *x*=*x*]
sup-comm
by *metis*

theorem *mann*: *2*×*x* = −(− *3*×*x* ⊔ *x*) ⊔ *2*×*x*
using *mann10*[**where** *x*=*x* **and** *y*=*2*×*x*]
mann8[**where** *x*=*x*]
two[**where** *x*=*x*] *three*[**where** *x*=*x*] *sup-comm*
by *metis*

corollary *winkerr*: *α* ⊔ *β* = *β*
using *mann secret-object2-def secret-object3-def two three*
by *metis*

corollary *winker*: *β* ⊔ *α* = *β*
by (*metis winkerr sup-comm*)

corollary *multi-winkerp*: *β* ⊔ *k* ⊗ *α* = *β*
by (*induct k*, (*simp add*: *winker sup-comm sup-assoc*)+)

corollary *multi-winker*: *β* ⊔ *k* × *α* = *β*
by (*induct k*, (*simp add*: *multi-winkerp winker sup-comm sup-assoc*)+)

lemma *less-eq-introp*:
−(*x* ⊔ −(*y* ⊔ *z*)) = −(*x* ⊔ *y* ⊔ −*z*) ⟹ *y* ⊑ *x*
by (*metis robbins sup-assoc less-eq-def*
sup-comm[**where** *x*=*x* **and** *y*=*y*])

corollary *less-eq-intro*:
−(*x* ⊔ −(*y* ⊔ *z*)) = −(*x* ⊔ *y* ⊔ −*z*) ⟹ *x* ⊔ *y* = *x*
by (*metis less-eq-introp less-eq-def sup-comm*)

lemma *eq-intro*:
$-(x \sqcup -(y \sqcup z)) = -(y \sqcup -(x \sqcup z)) \Longrightarrow x = y$
by (*metis robbins sup-assoc 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 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 $-(-(x \sqcup -y) \sqcup -y) = x$
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 assms robbins sup-comm copyp1*)

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
       copyp2[where x=x ⊔ x and y=y and k=0]
  have −(y ⊔ x ⊔ x ⊔ −(x ⊔ x ⊔ −y)) = −y by fastforce
  with sup-comm sup-assoc
  have −(x ⊔ x ⊔ y ⊔ −(x ⊔ (x ⊔ −y))) = −y by metis
  ultimately have
   −(x ⊔ x ⊔ y ⊔ −(x ⊔ (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-threep 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-threep 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
    ... = β ⊔ 3 × (α ⊔ −(α ⊔ −β)) by metis
```

```
  also from winker sup-comm two-three[where x=α and y=β] have
  ... = β ⊔ 2 × (α ⊔ −(α ⊔ −β)) by fastforce
  also from copy-distrib[where k=2]
            multi-winker[where k=2]
            sup-assoc two secret-object4-def have
  ... = δ by metis
  finally have ♡: δ ⊔ −(α ⊔ −δ) = δ by simp
  from secret-object4-def winkerr sup-assoc have
       α ⊔ δ = δ by metis
  hence δ ⊔ α = δ by (metis sup-comm)
  hence −(−(δ ⊔ −δ) ⊔ −δ) = −(−(δ ⊔ (α ⊔ −δ)) ⊔ −δ) by (metis sup-assoc)
  also from ♡ have
       ... = −(−(δ ⊔ (α ⊔ −δ)) ⊔ −(δ ⊔ −(α ⊔ −δ))) by metis
  also from robbins have
       ... = δ by metis
  finally have −(−(δ ⊔ −δ) ⊔ −δ) = δ by simp
  with two-three[where x=δ and y=δ]
       secret-object5-def sup-comm
  have 3 × γ ⊔ δ = 2 × γ ⊔ δ by fastforce
  with secret-object5-def sup-assoc sup-comm have
       3 × γ ⊔ γ = 2 × γ ⊔ γ by fastforce
  with two three four five six have
       6 × γ = 3 × γ by simp
  moreover have 3 + 3 = (6::nat) and 3 ≠ (0::nat) by arith+
  moreover note copy-arith[where k=3 and l=3 and x=γ]
               winker-object-def three
  ultimately show ?thesis by simp
qed



lemma sup-ident: x ⊔ ⊥⊥ = x
proof −
  have I: ϱ = −(−ϱ ⊔ ⊥⊥)
    by (metis fake-bot-def inf-eq robbins sup-comm sup-idem)

  { fix x have x = −(−(x ⊔ −ϱ ⊔ ⊥⊥) ⊔ −(x ⊔ ϱ))
    by (metis I robbins sup-assoc) }
  note II = this

  have III: −ϱ = −(−(ϱ ⊔ −ϱ ⊔ −ϱ) ⊔ ϱ)
    by (metis robbins[where x=−ϱ and y=ϱ ⊔ −ϱ]
              I sup-comm fake-bot-def)
  hence ϱ = −(−(ϱ ⊔ −ϱ ⊔ −ϱ) ⊔ −ϱ)
    by (metis robbins[where x=ϱ and y=ϱ ⊔ −ϱ ⊔ −ϱ]
              sup-comm[where x=ϱ and y=−(ϱ ⊔ −ϱ ⊔ −ϱ)]
              sup-assoc sup-idem)
  hence −(ϱ ⊔ −ϱ ⊔ −ϱ) = ⊥⊥
    by (metis robbins[where x=−(ϱ ⊔ −ϱ ⊔ −ϱ) and y=ϱ]
```

```
        III sup-comm fake-bot-def)
  hence −ϱ = −(ϱ ⊔ ⊥⊥)
    by (metis III sup-comm)
  hence ϱ = −(−(ϱ ⊔ ⊥⊥) ⊔ −(ϱ ⊔ ⊥⊥ ⊔ −ϱ))
    by (metis II sup-idem sup-comm sup-assoc)
  moreover have ϱ ⊔ ⊥⊥ = −(−(ϱ ⊔ ⊥⊥) ⊔ −(ϱ ⊔ ⊥⊥ ⊔ −ϱ))
    by (metis robbins[where x=ϱ ⊔ ⊥⊥ and y=ϱ]
        sup-comm[where y=ϱ]
        sup-assoc sup-idem)
  ultimately have ϱ = ϱ ⊔ ⊥⊥ by auto
  hence x ⊔ ⊥⊥ = −(−(x ⊔ ϱ) ⊔ −(x ⊔ ⊥⊥ ⊔ −ϱ))
    by (metis robbins[where x=x ⊔ ⊥⊥ and y=ϱ]
        sup-comm[where x=⊥⊥ and y=ϱ]
        sup-assoc)
  thus ?thesis by (metis sup-assoc sup-comm II)
qed



lemma dbl-neg: − (−x) = x
proof −
  { fix x have ⊥⊥ = −(−x ⊔ −(−x))
      by (metis robbins sup-comm sup-ident)
  } note I = this

  { fix x have −x = −(−(−x ⊔ −(−(−x))))
    by (metis I robbins sup-comm sup-ident)
  } note II = this

  { fix x have −(−(−x)) = −(−(−x ⊔ −(−(−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 (⊓) (⊔) ⊥ ⊤
apply unfold-locales
apply (metis dbl-neg robbins sup-comm)
done

theorem robbins-is-boolean-II:
  class.boolean-algebra-II uminus (⊓) (⊔) ⊥ ⊤
proof −
  interpret huntington:
    huntington-algebra uminus (⊓) (⊔) ⊥ ⊤
```

```
    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 (⊓) (⊑) (⊏) (⊔) ⊥ ⊤
proof −
  interpret huntington:
    huntington-algebra uminus (⊓) (⊔) ⊥ ⊤
      by (fact robbins-is-huntington)
  show ?thesis by (simp add: huntington.huntington-is-boolean)
qed

end

no-notation secret-object1 (ι)
   and secret-object2 (α)
   and secret-object3 (β)
   and secret-object4 (δ)
   and secret-object5 (γ)
   and winker-object (ϱ)
   and less-eq  (infix ⊑ 50)
   and less (infix ⊏ 50)
   and inf (infixl ⊓ 70)
   and sup (infixl ⊔ 65)
   and top (⊤)
   and bot (⊥)
   and copyp (infix ⊗ 80)
   and copy (infix × 85)

notation
  Product-Type.Times  (infixr × 80)

end
```