Topological semantics for paraconsistent and paracomplete logics

David Fuenmayor

March 17, 2025

Abstract

We investigate mathematical structures that provide natural semantics for families of (quantified) non-classical logics featuring special unary connectives, known as recovery operators, that allow us to 'recover' the properties of classical logic in a controlled manner. These structures are known as topological Boolean algebras, which are Boolean algebras extended with additional operations subject to specific conditions of a topological nature. In this study we focus on the paradigmatic case of negation. We demonstrate how these algebras are well-suited to provide a semantics for some families of paraconsistent Logics of Formal Inconsistency and paracomplete Logics of Formal Undeterminedness. These logics feature recovery operators used to earmark propositions that behave 'classically' when interacting with non-classical negations. We refer to the companion paper [1] for more information.

Contents

1 Shallow semantical embedding of (a logic of) Boolean algebras **2**
- 1.1 Encoding Boolean operations . . . 3
- 1.2 Atomicity and primitive equality . . . 5
- 1.3 Miscellaneous notions . . . 5
- 1.4 Operations on set-valued functions . . . 5
- 1.5 Operators and their transformations . . . 6
- 1.6 Encoding infinitary Boolean operations . . . 9

2 Topological Conditions **10**
- 2.1 Positive Conditions . . . 11
- 2.2 Infinitary Positive Conditions . . . 13

3 Alexandrov topologies and (generalized) specialization preorders **15**
- 3.1 Negative Conditions . . . 17
- 3.2 Infinitary Negative Conditions . . . 20
- 3.3 Relativized Conditions . . . 21
- 3.4 Infinitary Relativized Conditions . . . 25

4 Logics based on Topological Boolean Algebras **27**
- 4.1 Logical Consequence and Validity . . . 27
- 4.2 Converting between topological operators . . . 28
- 4.3 Properties of negation(-like) operators . . . 29
  - 4.3.1 Principles of excluded middle, contradiction and explosion . . . 29
  - 4.3.2 Contraposition rules . . . 30
  - 4.3.3 Modus tollens rules . . . 31

4.3.4 Double negation introduction and elimination . . . 31
4.3.5 (Anti)Normality and its dual . . . 32
4.3.6 De Morgan laws . . . 32
4.3.7 Strong contraposition (axiom-like) . . . 32
4.3.8 Local modus tollens axioms . . . 33
4.3.9 Disjunctive syllogism . . . 34
4.4 Quantifiers (restricted and unrestricted) . . . 34
4.5 Examples on Quantifiers . . . 36
4.6 Barcan formula and its converse . . . 37
4.7 Logics of Formal Inconsistency (LFIs) . . . 37
4.8 Logics of Formal Undeterminedness (LFUs) . . . 40

theory *boolean-algebra*
imports *Main*
begin

Technical configuration

declare[[*smt-timeout=30*]]
declare[[*show-types*]]

sledgehammer-params[*isar-proof=false*]
nitpick-params[*assms=true*, *user-axioms=true*, *show-all*, *expect=genuine*, *format=3*, *atoms=a b c d*]

We hide some Isabelle/HOL notation from the libraries (which we don't use) to avoid overloading

hide-const(**open**) *List.list.Nil* **unbundle** *no list-syntax*
hide-const(**open**) *Relation.converse* **unbundle** *no converse-syntax*
hide-const(**open**) *Fun.comp* **no-notation** *Fun.comp* (**infixl** ‹○› *55*)
hide-const(**open**) *Groups.plus-class.plus* **no-notation** *Groups.plus-class.plus* (**infixl** ‹+› *65*)
hide-const(**open**) *Groups.times-class.times* **no-notation** *Groups.times-class.times* (**infixl** ‹∗› *70*)
hide-const(**open**) *Groups.minus-class.minus* **no-notation** *Groups.minus-class.minus* (**infixl** ‹−› *65*)
hide-const(**open**) *Groups.uminus-class.uminus* **unbundle** *no uminus-syntax*

1 Shallow semantical embedding of (a logic of) Boolean algebras

We encode Boolean algebras via their (Stone) representation as algebras of sets ('fields of sets'). This means that each element of (the carrier of) the algebra will be a set of 'points'. Inspired by the 'propositions as sets of worlds' paradigm from modal logic, we may think of points as being 'worlds', and thus of the elements of our Boolean algebras as 'propositions'. Of course, this is just one among many possible interpretations, and nothing stops us from thinking of points as any other kind of object (e.g., they can be sets, functions, sets of functions, etc.).

We utilize a particular naming convention: The type parameter 'w is employed for the domain/universe of 'points'. We conveniently introduce the (parametric) type-alias $('w)\sigma$ as shorthand for $'w \Rightarrow bool$. Hence, the elements of our algebra are objects of type $('w)\sigma$, and thus correspond to (characteristic functions of) sets of 'points'. Set-valued (resp. set-domain) functions are thus functions that have sets as their codomain (resp. domain), they are basically anything with a (parametric) type $'a \Rightarrow ('w)\sigma$ (resp. $('w)\sigma \Rightarrow 'a$).

Type for (characteristic functions of) sets (of 'points')

type-synonym $'w\ \sigma = \langle 'w \Rightarrow bool \rangle$

In the sequel, we will (try to) enforce the following naming convention:

(i) Upper-case latin letters (A, B, D, M, P, S, X, etc.) denote arbitrary sets (type $('w)\sigma$). We will employ lower-case letters (p, q, x, w, etc.) to denote variables playing the role of 'points'. In some contexts, the letters S and D will be employed to denote sets/domains of sets (of 'points').

(ii) Greek letters denote arbitrary set-valued functions (type $'a \Rightarrow ('w)\sigma$). We employ the letters φ, ψ *and* η to denote arbitrary unary operations (with type $('w)\sigma \Rightarrow ('w)\sigma$).

(iii) Upper-case calligraphic letters ($\mathcal{B}$, $\mathcal{I}$, $\mathcal{C}$, $\mathcal{F}$, *etc.*) are reserved for unary operations that are intended to act as 'topological operators' in the given context.

1.1 Encoding Boolean operations

Standard inclusion-based order structure on sets.

definition *subset*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow bool$ (**infixr** $\langle \leq \rangle$ *45*)
  where $A \leq B \equiv \forall p.\ A\ p \longrightarrow B\ p$
definition *setequ*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow bool$ (**infixr** $\langle = \rangle$ *45*)
  where $A = B \equiv \forall p.\ A\ p \longleftrightarrow B\ p$

named-theorems *order*
declare *setequ-def*[*order*] *subset-def*[*order*]

lemma *subset-char1*: $(A \leq B) = (\forall C.\ B \leq C \longrightarrow A \leq C)$ ⟨*proof*⟩
lemma *subset-char2*: $(A \leq B) = (\forall C.\ C \leq A \longrightarrow C \leq B)$ ⟨*proof*⟩

These (trivial) lemmas are intended to help automated tools.

lemma *setequ-char*: $(A = B) = (A \leq B \wedge B \leq A)$ ⟨*proof*⟩
lemma *setequ-ext*: $(A = B) = (A = B)$ ⟨*proof*⟩

We now encode connectives for (distributive and complemented) bounded lattices, mostly by reusing their counterpart meta-logical HOL connectives.

definition *meet*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \wedge \rangle$ *54*)
  where $A \wedge B \equiv \lambda p.\ (A\ p) \wedge (B\ p)$ — intersection
definition *join*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \vee \rangle$ *53*)
  where $A \vee B \equiv \lambda p.\ (A\ p) \vee (B\ p)$ — union
definition *top*$::'w\ \sigma$ ($\langle \top \rangle$)
  where $\top \equiv \lambda w.\ True$ — universe
definition *bottom*$::'w\ \sigma$ ($\langle \bot \rangle$)
  where $\bot \equiv \lambda w.\ False$ — empty-set

And introduce further operations to obtain a Boolean algebra (of sets).

definition *compl*$::'w\ \sigma \Rightarrow 'w\ \sigma$ ($\langle - \rangle$ [*57*]*58*)
  where $-A\ \equiv \lambda p.\ \neg(A\ p)$ — (set-)complement
definition *impl*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \rightarrow \rangle$ *51*)
  where $A \rightarrow B \equiv \lambda p.\ (A\ p) \longrightarrow (B\ p)$ — (set-)implication
definition *diff*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \leftharpoondown \rangle$ *51*)
  where $A \leftharpoondown B \equiv \lambda p.\ (A\ p) \wedge \neg(B\ p)$ — (set-)difference
definition *dimpl*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \leftrightarrow \rangle$ *51*)
  where $A \leftrightarrow B \equiv \lambda p.\ (A\ p) = (B\ p)$ — double implication
definition *sdiff*$::'w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma$ (**infixr** $\langle \triangle \rangle$ *51*)
  where $A \triangle B \equiv \lambda p.\ (A\ p) \neq (B\ p)$ — symmetric difference (aka. xor)

named-theorems *conn*
declare *meet-def*[*conn*] *join-def*[*conn*] *top-def*[*conn*] *bottom-def*[*conn*]
impl-def[*conn*] *dimpl-def*[*conn*] *diff-def*[*conn*] *sdiff-def*[*conn*] *compl-def*[*conn*]

Verify characterization for some connectives.

lemma *compl-char*: $-A = A \rightarrow \bot$ ⟨*proof*⟩
lemma *impl-char*: $A \rightarrow B = -A \vee B$ ⟨*proof*⟩
lemma *dimpl-char*: $A \leftrightarrow B = (A \rightarrow B) \wedge (B \rightarrow A)$ ⟨*proof*⟩
lemma *diff-char1*: $A \leftharpoonup B = A \wedge -B$ ⟨*proof*⟩
lemma *diff-char2*: $A \leftharpoonup B = -(A \rightarrow B)$ ⟨*proof*⟩
lemma *sdiff-char1*: $A \triangle B = (A \leftharpoonup B) \vee (B \leftharpoonup A)$ ⟨*proof*⟩
lemma *sdiff-char2*: $A \triangle B = -(A \leftrightarrow B)$ ⟨*proof*⟩

We can verify that (quite trivially) this algebra satisfies some properties of lattices.

lemma *L1*: $A = A \vee A$ ⟨*proof*⟩
lemma *L2*: $A = A \wedge A$ ⟨*proof*⟩
lemma *L3*: $A \leq A \vee B$ ⟨*proof*⟩
lemma *L4*: $A \wedge B \leq A$ ⟨*proof*⟩
lemma *L5*: $(A \wedge B) \vee B = B$ ⟨*proof*⟩
lemma *L6*: $A \wedge (A \vee B) = A$ ⟨*proof*⟩
lemma *L7*: $A \leq C \wedge B \leq C \longrightarrow A \vee B \leq C$ ⟨*proof*⟩
lemma *L8*: $C \leq A \wedge C \leq B \longrightarrow C \leq A \wedge B$ ⟨*proof*⟩
lemma *L9*: $A \leq B \longleftrightarrow (A \vee B) = B$ ⟨*proof*⟩
lemma *L10*: $B \leq A \longleftrightarrow (A \wedge B) = B$ ⟨*proof*⟩
lemma *L11*: $A \leq B \wedge C \leq D \longrightarrow A \vee C \leq B \vee D$ ⟨*proof*⟩
lemma *L12*: $A \leq B \wedge C \leq D \longrightarrow A \wedge C \leq B \wedge D$ ⟨*proof*⟩
lemma *L13*: $A \wedge \top = A$ ⟨*proof*⟩
lemma *L14*: $A \vee \bot = A$ ⟨*proof*⟩
lemma *L15*: $A \leq B \longleftrightarrow (\forall\, C.\ C \wedge A \leq C \wedge B)$ ⟨*proof*⟩
lemma *L16*: $A \leq B \longleftrightarrow (\forall\, C.\ C \vee A \leq C \vee B)$ ⟨*proof*⟩

These properties below hold in particular also for Boolean algebras.

lemma *BA-impl*: $A \leq B \longleftrightarrow A \rightarrow B = \top$ ⟨*proof*⟩
lemma *BA-distr1*: $(A \wedge (B \vee C)) = ((A \wedge B) \vee (A \wedge C))$ ⟨*proof*⟩
lemma *BA-distr2*: $(A \vee (B \wedge C)) = ((A \vee B) \wedge (A \vee C))$ ⟨*proof*⟩
lemma *BA-cp*: $A \leq B \longleftrightarrow -B \leq -A$ ⟨*proof*⟩
lemma *BA-deMorgan1*: $-(A \vee B) = (-A \wedge -B)$ ⟨*proof*⟩
lemma *BA-deMorgan2*: $-(A \wedge B) = (-A \vee -B)$ ⟨*proof*⟩
lemma *BA-dn*: $--A = A$ ⟨*proof*⟩
lemma *BA-cmpl-equ*: $(-A = B) = (A = -B)$ ⟨*proof*⟩

We conveniently introduce these properties of sets of sets (of points).

definition *meet-closed*::$('w\ \sigma)\sigma \Rightarrow$ *bool*
where *meet-closed* $S \equiv\ \forall\, X\ Y.\ (S\ X \wedge S\ Y) \longrightarrow S(X \wedge\ Y)$
definition *join-closed*::$('w\ \sigma)\sigma \Rightarrow$ *bool*
where *join-closed* $S \equiv\ \forall\, X\ Y.\ (S\ X \wedge S\ Y) \longrightarrow S(X \vee\ Y)$

definition *upwards-closed*::$('w\ \sigma)\sigma \Rightarrow$ *bool*
where *upwards-closed* $S \equiv \forall\, X\ Y.\ S\ X \wedge X \leq Y \longrightarrow S\ Y$
definition *downwards-closed*::$('w\ \sigma)\sigma \Rightarrow$ *bool*
where *downwards-closed* $S \equiv \forall\, X\ Y.\ S\ X \wedge Y \leq X \longrightarrow S\ Y$

1.2 Atomicity and primitive equality

We can verify indeed that the algebra is atomic (in three different ways) by relying on the presence of primitive equality in HOL.

lemma *atomic1*: $\forall\, w.\ \exists\, Q.\ Q\ w \wedge (\forall\, P.\ P\ w \longrightarrow Q \leq P)$ ⟨*proof*⟩

definition *atom* $A \equiv \neg(A = \bot) \wedge (\forall\, P.\ A \leq P \vee A \leq -P)$

lemma *atomic2*: $\forall\, w.\ \exists\, Q.\ Q\ w \wedge \mathit{atom}\ Q$ ⟨*proof*⟩
lemma *atomic3*: $\forall\, P.\ \neg(P = \bot) \longrightarrow (\exists\, Q.\ \mathit{atom}\ Q \wedge Q \leq P)$ ⟨*proof*⟩

Now with interactive proof:

lemma $\forall\, P.\ \neg(P = \bot) \longrightarrow (\exists\, Q.\ \mathit{atom}\ Q \wedge Q \leq P)$
⟨*proof*⟩

1.3 Miscellaneous notions

We add some miscellaneous notions that will be useful later.

abbreviation *isEmpty* $S \equiv \forall\, x.\ \neg S\ x$
abbreviation *nonEmpty* $S \equiv \exists\, x.\ S\ x$

Function composition.

definition *fun-comp* :: $('b \Rightarrow 'c) \Rightarrow (\ 'a \Rightarrow 'b) \Rightarrow 'a \Rightarrow 'c$ (**infixl** ‹○› *75*)
  where $\varphi \circ \psi \equiv \lambda x.\ \varphi\ (\psi\ x)$

Inverse projection maps a unary function to a 'projected' binary function wrt. its 1st argument.

abbreviation *inv-proj*::‹$('a \Rightarrow 'c) \Rightarrow ('a \Rightarrow 'b \Rightarrow 'c)$› (‹(-)⫤›)
  where $D⫤ \equiv \lambda A\ B.\ D\ A$

Image of a mapping φ, with range as an special case.

definition *image*::$('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow \mathit{bool}) \Rightarrow ('b \Rightarrow \mathit{bool})$ (‹⟦- -⟧›)
  where $⟦\varphi\ S⟧ \equiv \lambda y.\ \exists\, x.\ (S\ x) \wedge (\varphi\ x) = y$
definition *range*::$('a \Rightarrow 'b) \Rightarrow ('b \Rightarrow \mathit{bool})$ (‹⟦-'-⟧›)
  where $⟦\varphi\ \text{-}⟧ \equiv \lambda Y.\ \exists\, x.\ (\varphi\ x) = Y$
lemma *range-char1*: $⟦\varphi\ \text{-}⟧ = ⟦\varphi\ (\lambda x.\ \mathit{True})⟧$ ⟨*proof*⟩
lemma *range-char2*: $⟦\varphi\ \text{-}⟧ = (\lambda X.\ \exists\, S.\ ⟦\varphi\ S⟧\ X)$ ⟨*proof*⟩

lemma *image-comp*: $⟦(f \circ g)\ S⟧ = ⟦f\ ⟦g\ S⟧⟧$ ⟨*proof*⟩

end
theory *boolean-algebra-operators*
  imports *boolean-algebra*
begin

1.4 Operations on set-valued functions

Functions with sets in their codomain will be called here 'set-valued functions'. We conveniently define some (2nd-order) Boolean operations on them.

The 'meet' and 'join' of two set-valued functions.

definition *svfun-meet*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma)$ (**infixr** ‹∧:› *62*)
  where $\varphi \wedge^{:} \psi \equiv \lambda x.\ (\varphi\ x) \wedge (\psi\ x)$
definition *svfun-join*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma)$ (**infixr** ‹∨:› *61*)

where $\varphi \vee^{:} \psi \equiv \lambda x.\ (\varphi\ x) \vee (\psi\ x)$

Analogously, we can define an 'implication' and a 'complement'.

definition *svfun-impl*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma)$ (**infixr** ‹$\rightarrow^{:}$› *61*)
where $\psi \rightarrow^{:} \varphi \equiv \lambda x.\ (\psi\ x) \rightarrow (\varphi\ x)$
definition *svfun-compl*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma)$ (‹(-^{-})›)
where $\varphi^{-} \equiv \lambda x.\ -(\varphi\ x)$

There are two natural 0-ary connectives (aka. constants).

definition *svfun-top*::$'a \Rightarrow 'w\ \sigma$ (‹$\top^{:}$›)
where $\top^{:} \equiv \lambda x.\ \top$
definition *svfun-bot*::$'a \Rightarrow 'w\ \sigma$ (‹$\bot^{:}$›)
where $\bot^{:} \equiv \lambda x.\ \bot$

named-theorems *conn2*
declare *svfun-meet-def*[*conn2*] *svfun-join-def*[*conn2*] *svfun-impl-def*[*conn2*]
svfun-compl-def[*conn2*] *svfun-top-def*[*conn2*] *svfun-bot-def*[*conn2*]

And, of course, set-valued functions are naturally ordered in the expected way:

definition *svfun-sub*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma) \Rightarrow$ *bool* (**infixr** ‹$\leq^{:}$› *55*)
where $\psi \leq^{:} \varphi \equiv \forall x.\ (\psi\ x) \leq (\varphi\ x)$
definition *svfun-equ*::$('a \Rightarrow 'w\ \sigma) \Rightarrow ('a \Rightarrow 'w\ \sigma) \Rightarrow$ *bool* (**infixr** ‹$=^{:}$› *55*)
where $\psi =^{:} \varphi \equiv \forall x.\ (\psi\ x) = (\varphi\ x)$

named-theorems *order2*
declare *svfun-sub-def*[*order2*] *svfun-equ-def*[*order2*]

These (trivial) lemmas are intended to help automated tools.

lemma *svfun-sub-char*: $(\psi \leq^{:} \varphi) = (\psi \rightarrow^{:} \varphi =^{:} \top^{:})$ ⟨*proof*⟩
lemma *svfun-equ-char*: $(\psi =^{:} \varphi) = (\psi \leq^{:} \varphi \wedge \varphi \leq^{:} \psi)$ ⟨*proof*⟩
lemma *svfun-equ-ext*: $(\psi =^{:} \varphi) = (\psi = \varphi)$ ⟨*proof*⟩

Clearly, set-valued functions form a Boolean algebra. We can prove some interesting relationships:

lemma *svfun-compl-char*: $\varphi^{-} = (\varphi \rightarrow^{:} \bot^{:})$ ⟨*proof*⟩
lemma *svfun-impl-char1*: $(\psi \rightarrow^{:} \varphi) = (\psi^{-} \vee^{:} \varphi)$ ⟨*proof*⟩
lemma *svfun-impl-char2*: $(\psi \rightarrow^{:} \varphi) = (\psi \wedge^{:} (\varphi^{-}))^{-}$ ⟨*proof*⟩
lemma *svfun-deMorgan1*: $(\psi \wedge^{:} \varphi)^{-} = (\psi^{-}) \vee^{:} (\varphi^{-})$ ⟨*proof*⟩
lemma *svfun-deMorgan2*: $(\psi \vee^{:} \varphi)^{-} = (\psi^{-}) \wedge^{:} (\varphi^{-})$ ⟨*proof*⟩

1.5 Operators and their transformations

Dual to set-valued functions we can have set-domain functions. For them we can define the 'dual-complement'.

definition *sdfun-dcompl*::$('w\ \sigma \Rightarrow 'a) \Rightarrow ('w\ \sigma \Rightarrow 'a)$ (‹(-^{d-})›)
where $\varphi^{d-} \equiv \lambda X.\ \varphi(-X)$
lemma *sdfun-dcompl-char*: $\varphi^{d-} = (\lambda X.\ \exists\, Y.\ (\varphi\ Y) \wedge (X = -Y))$ ⟨*proof*⟩

Operators are a particularly important kind of functions. They are both set-valued and set-domain. Thus our algebra of operators inherits the connectives defined above plus the ones below.

We conveniently define the 'dual' of an operator.

definition *op-dual*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)\ (\langle(\text{-}^d)\rangle)$
where $\varphi^d \equiv \lambda X.\ -(\varphi(-X))$

The following two 0-ary connectives (i.e. operator 'constants') exist already (but somehow implicitly). We just make them explicit by introducing some convenient notation.

definition *id-op*::$'w\ \sigma \Rightarrow 'w\ \sigma\ (\langle\mathbf{e}\rangle)$
where $\mathbf{e} \equiv \lambda X.\ X$
definition *compl-op*::$'w\ \sigma \Rightarrow 'w\ \sigma\ (\langle\mathbf{n}\rangle)$
where $\mathbf{n} \equiv \lambda X.\ -X$

declare *sdfun-dcompl-def*[*conn2*] *op-dual-def*[*conn2*] *id-op-def*[*conn2*] *compl-op-def*[*conn2*]

We now prove some lemmas (some of them might help provers in their hard work).

lemma *dual-compl-char1*: $\varphi^{d-} = (\varphi^d)^-$ ⟨*proof*⟩
lemma *dual-compl-char2*: $\varphi^{d-} = (\varphi^-)^d$ ⟨*proof*⟩
lemma *sfun-compl-invol*: $\varphi^{--} = \varphi$ ⟨*proof*⟩
lemma *dual-invol*: $\varphi^{dd} = \varphi$ ⟨*proof*⟩
lemma *dualcompl-invol*: $(\varphi^{d-})^{d-} = \varphi$ ⟨*proof*⟩

lemma *op-prop1*: $\mathbf{e}^d = \mathbf{e}$ ⟨*proof*⟩
lemma *op-prop2*: $\mathbf{n}^d = \mathbf{n}$ ⟨*proof*⟩
lemma *op-prop3*: $\mathbf{e}^- = \mathbf{n}$ ⟨*proof*⟩
lemma *op-prop4*: $(\varphi \vee^: \psi)^d = (\varphi^d) \wedge^: (\psi^d)$ ⟨*proof*⟩
lemma *op-prop5*: $(\varphi \vee^: \psi)^- = (\varphi^-) \wedge^: (\psi^-)$ ⟨*proof*⟩
lemma *op-prop6*: $(\varphi \wedge^: \psi)^d = (\varphi^d) \vee^: (\psi^d)$ ⟨*proof*⟩
lemma *op-prop7*: $(\varphi \wedge^: \psi)^- = (\varphi^-) \vee^: (\psi^-)$ ⟨*proof*⟩
lemma *op-prop8*: $\top^: = \mathbf{n} \vee^: \mathbf{e}$ ⟨*proof*⟩
lemma *op-prop9*: $\bot^: = \mathbf{n} \wedge^: \mathbf{e}$ ⟨*proof*⟩

The notion of a fixed-point is fundamental. We speak of sets being fixed-points of operators. We define a function that given an operator returns the set of all its fixed-points.

definition *fixpoints*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma)\sigma\ (\langle fp\rangle)$
where $fp\ \varphi \equiv \lambda X.\ (\varphi\ X) = X$

We can in fact 'operationalize' the function above, thus obtaining a 'fixed-point operation'.

definition *op-fixpoint*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)\ (\langle(\text{-}^{fp})\rangle)$
where $\varphi^{fp} \equiv \lambda X.\ (\varphi\ X) \leftrightarrow X$

declare *fixpoints-def*[*conn2*] *op-fixpoint-def*[*conn2*]

Interestingly, the fixed-point operation (or transformation) is definable in terms of the others.

lemma *op-fixpoint-char*: $\varphi^{fp} = (\varphi \wedge^: \mathbf{e}) \vee^: (\varphi^- \wedge^: \mathbf{n})$ ⟨*proof*⟩

Given an operator φ the fixed-points of it's dual is the set of complements of $\varphi's$ fixed-points.

lemma *fp-dual*: $fp\ \varphi^d = (fp\ \varphi)^{d-}$ ⟨*proof*⟩

The fixed-points of $\varphi's$ complement is the set of complements of the fixed-points of $\varphi's$ dual-complement.

lemma *fp-compl*: $fp\ \varphi^- = (fp\ (\varphi^{d-}))^{d-}$ ⟨*proof*⟩

The fixed-points of $\varphi's$ dual-complement is the set of complements of the fixed-points of $\varphi's$ complement.

lemma *fp-dcompl*: $fp\ (\varphi^{d-}) = (fp\ \varphi^-)^{d-}$ ⟨*proof*⟩

The fixed-points function and the fixed-point operation are essentially related.

lemma *fp-rel*: $fp\ \varphi\ A \longleftrightarrow (\varphi^{fp}\ A) = \top$ ⟨*proof*⟩
lemma *fp-d-rel*: $fp\ \varphi^{d}\ A \longleftrightarrow \varphi^{fp}(-A) = \top$ ⟨*proof*⟩
lemma *fp-c-rel*: $fp\ \varphi^{-}\ A \longleftrightarrow \varphi^{fp}\ A = \bot$ ⟨*proof*⟩
lemma *fp-dc-rel*: $fp\ (\varphi^{d-})\ A \longleftrightarrow \varphi^{fp}(-A) = \bot$ ⟨*proof*⟩

The fixed-point operation is involutive.

lemma *ofp-invol*: $(\varphi^{fp})^{fp} = \varphi$ ⟨*proof*⟩

And commutes the dual with the dual-complement operations.

lemma *ofp-comm-dc1*: $(\varphi^{d})^{fp} = (\varphi^{fp})^{d-}$ ⟨*proof*⟩
lemma *ofp-comm-dc2*:$(\varphi^{d-})^{fp} = (\varphi^{fp})^{d}$ ⟨*proof*⟩

The fixed-point operation commutes with the complement.

lemma *ofp-comm-compl*: $(\varphi^{-})^{fp} = (\varphi^{fp})^{-}$ ⟨*proof*⟩

The above motivates the following alternative definition for a 'complemented-fixed-point' operation.

lemma *ofp-fixpoint-compl-def*: $\varphi^{fp-} = (\lambda X.\ (\varphi\ X)\ \triangle\ X)$ ⟨*proof*⟩

Analogously, the 'complemented fixed-point' operation is also definable in terms of the others.

lemma *op-fixpoint-compl-char*: $\varphi^{fp-} = (\varphi\ \vee^{:}\ \mathbf{e})\ \wedge^{:}\ (\varphi^{-}\ \vee^{:}\ \mathbf{n})$ ⟨*proof*⟩

In fact, function composition can be seen as an additional binary connective for operators. We show below some interesting relationships that hold.

lemma *op-prop10*: $\varphi = (\mathbf{e} \circ \varphi)$ ⟨*proof*⟩
lemma *op-prop11*: $\varphi = (\varphi \circ \mathbf{e})$ ⟨*proof*⟩
lemma *op-prop12*: $\mathbf{e} = (\mathbf{n} \circ \mathbf{n})$ ⟨*proof*⟩
lemma *op-prop13*: $\varphi^{-} = (\mathbf{n} \circ \varphi)$ ⟨*proof*⟩
lemma *op-prop14*: $\varphi^{d-} = (\varphi \circ \mathbf{n})$ ⟨*proof*⟩
lemma *op-prop15*: $\varphi^{d} = (\mathbf{n} \circ \varphi \circ \mathbf{n})$ ⟨*proof*⟩

There are also some useful properties regarding the images of operators.

lemma *im-prop1*: $[\![\varphi\ D]\!]^{d-} = [\![\varphi^{d}\ D^{d-}]\!]$ ⟨*proof*⟩
lemma *im-prop2*: $[\![\varphi^{-}\ D]\!]^{d-} = [\![\varphi\ D]\!]$ ⟨*proof*⟩
lemma *im-prop3*: $[\![\varphi^{d}\ D]\!]^{d-} = [\![\varphi\ D^{d-}]\!]$ ⟨*proof*⟩
lemma *im-prop4*: $[\![\varphi^{d-}\ D]\!]^{d-} = [\![\varphi^{d}\ D]\!]$ ⟨*proof*⟩
lemma *im-prop5*: $[\![\varphi^{-}\ D^{d-}]\!] = [\![\varphi^{d-}\ D]\!]^{d-}$ ⟨*proof*⟩
lemma *im-prop6*: $[\![\varphi^{d-}\ D^{d-}]\!] = [\![\varphi\ D]\!]$ ⟨*proof*⟩

Observe that all results obtained by assuming fixed-point predicates extend to their associated operators.

lemma $\varphi^{fp}(A) \wedge \Gamma(A) \leq \Delta(A) \longrightarrow (fp\ \varphi)(A) \longrightarrow \Gamma(A) \leq \Delta(A)$
⟨*proof*⟩
lemma $\varphi^{fp}(A) \wedge \varphi^{fp}(B) \wedge (\Gamma\ A\ B) \leq (\Delta\ A\ B) \longrightarrow (fp\ \varphi)(A) \wedge (fp\ \varphi)(B) \longrightarrow (\Gamma\ A\ B) \leq (\Delta\ A\ B)$
⟨*proof*⟩

end
theory *boolean-algebra-infinitary*
imports *boolean-algebra-operators*
begin

1.6 Encoding infinitary Boolean operations

Our aim is to encode complete Boolean algebras (of sets) which we can later be used to interpret quantified formulas (somewhat in the spirit of Boolean-valued models for set theory).

We start by defining infinite meet (infimum) and infinite join (supremum) operations.

definition *infimum*:: $('w\ \sigma)\sigma \Rightarrow 'w\ \sigma$ ($\langle\bigwedge\text{-}\rangle$)
  where $\bigwedge S \equiv \lambda w.\ \forall X.\ S\ X \longrightarrow X\ w$
definition *supremum*::$('w\ \sigma)\sigma \Rightarrow 'w\ \sigma$ ($\langle\bigvee\text{-}\rangle$)
  where $\bigvee S \equiv \lambda w.\ \exists X.\ S\ X\ \wedge\ X\ w$

declare *infimum-def*[*conn*] *supremum-def*[*conn*]

Infimum and supremum satisfy an infinite variant of the De Morgan laws.

lemma *iDM-a*: $-(\bigwedge S) = \bigvee(S^{d-})$ ⟨*proof*⟩
lemma *iDM-b*: $-(\bigvee S) = \bigwedge(S^{d-})$ ⟨*proof*⟩

We show that our encoded Boolean algebras are lattice-complete. The functions below return the set of upper-/lower-bounds of a set of sets S (wrt. domain D).

definition *upper-bounds*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle ub\rangle$)
  where $ub\ S \equiv \lambda U.\ \forall X.\ S\ X \longrightarrow X \leq U$
definition *upper-bounds-restr*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle ub^{-}\rangle$)
  where $ub^D\ S \equiv \lambda U.\ D\ U \wedge (\forall X.\ S\ X \longrightarrow X \leq U)$
definition *lower-bounds*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle lb\rangle$)
  where $lb\ S \equiv \lambda L.\ \forall X.\ S\ X \longrightarrow L \leq X$
definition *lower-bounds-restr*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle lb^{-}\rangle$)
  where $lb^D\ S \equiv \lambda L.\ D\ L \wedge (\forall X.\ S\ X \longrightarrow L \leq X)$

lemma *ub-char*: $ub\ S = (let\ D{=}\top\ in\ ub^D\ S)$ ⟨*proof*⟩
lemma *lb-char*: $lb\ S = (let\ D{=}\top\ in\ lb^D\ S)$ ⟨*proof*⟩

Similarly, the functions below return the set of least/greatest upper-/lower-bounds for S (wrt. D).

definition *lub*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle lub\rangle$)
  where $lub\ S \equiv \lambda U.\ ub\ S\ U \wedge (\forall X.\ ub\ S\ X \longrightarrow U \leq X)$
definition *lub-restr*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle lub^{-}\rangle$)
  where $lub^D\ S \equiv \lambda U.\ ub^D\ S\ U \wedge (\forall X.\ ub^D\ S\ X \longrightarrow U \leq X)$
definition *glb*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle glb\rangle$)
  where $glb\ S \equiv \lambda L.\ lb\ S\ L \wedge (\forall X.\ lb\ S\ X \longrightarrow X \leq L)$
definition *glb-restr*::$('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma \Rightarrow ('w\ \sigma)\sigma$ ($\langle glb^{-}\rangle$)
  where $glb^D\ S \equiv \lambda L.\ lb^D\ S\ L \wedge (\forall X.\ lb^D\ S\ X \longrightarrow X \leq L)$

Both pairs of definitions above are suitably related. (Note that the term $\top$ below denotes the top element in the algebra of sets of sets (i.e. the powerset).)

lemma *lub-char*: $lub\ S = (let\ D{=}\top\ in\ lub^D\ S)$ ⟨*proof*⟩
lemma *glb-char*: $glb\ S = (let\ D{=}\top\ in\ glb^D\ S)$ ⟨*proof*⟩

Clearly, the notions of infimum/supremum correspond to least/greatest upper-/lower-bound.

lemma *sup-lub*: $lub\ S \bigvee S$ ⟨*proof*⟩
lemma *sup-exist-unique*: $\forall S.\ \exists!X.\ lub\ S\ X$ ⟨*proof*⟩
lemma *inf-glb*: $glb\ S \bigwedge S$ ⟨*proof*⟩
lemma *inf-exist-unique*: $\forall S.\ \exists!X.\ glb\ S\ X$ ⟨*proof*⟩

The property of being closed under arbitrary (resp. nonempty) supremum/infimum.

definition *infimum-closed* :: $('w\ \sigma)\sigma \Rightarrow bool$
  where *infimum-closed* $S \equiv \forall D.\ D \leq S \longrightarrow S(\bigwedge D)$
definition *supremum-closed* :: $('w\ \sigma)\sigma \Rightarrow bool$
  where *supremum-closed* $S \equiv \forall D.\ D \leq S \longrightarrow S(\bigvee D)$
definition *infimum-closed'* :: $('w\ \sigma)\sigma \Rightarrow bool$
  where*infimum-closed'* $S \equiv \forall D.\ nonEmpty\ D \wedge D \leq S \longrightarrow S(\bigwedge D)$
definition *supremum-closed'* :: $('w\ \sigma)\sigma \Rightarrow bool$
  where *supremum-closed'* $S \equiv \forall D.\ nonEmpty\ D \wedge D \leq S \longrightarrow S(\bigvee D)$

lemma *inf-empty*: $isEmpty\ S \Longrightarrow \bigwedge S = \top$ ⟨*proof*⟩
lemma *sup-empty*: $isEmpty\ S \Longrightarrow \bigvee S = \bot$ ⟨*proof*⟩

Note that arbitrary infimum- (resp. supremum-) closed sets include the top (resp. bottom) element.

lemma *infimum-closed* $S \Longrightarrow S\ \top$ ⟨*proof*⟩
lemma *supremum-closed* $S \Longrightarrow S\ \bot$ ⟨*proof*⟩

However, the above does not hold for non-empty infimum- (resp. supremum-) closed sets.

lemma *infimum-closed'* $S \Longrightarrow S\ \top$ **nitpick** ⟨*proof*⟩
lemma *supremum-closed'* $S \Longrightarrow S\ \bot$ **nitpick** ⟨*proof*⟩

We have in fact the following characterizations for the notions above.

lemma *inf-closed-char*: *infimum-closed* S = (*infimum-closed'* $S \wedge S\ \top$)
  ⟨*proof*⟩
lemma *sup-closed-char*: *supremum-closed* S = (*supremum-closed'* $S \wedge S\ \bot$)
  ⟨*proof*⟩

lemma *inf-sup-closed-dc*: *infimum-closed* S = *supremum-closed* S^{d-} ⟨*proof*⟩
lemma *inf-sup-closed-dc'*: *infimum-closed'* S = *supremum-closed'* S^{d-} ⟨*proof*⟩

We check some further properties.

lemma *fp-inf-sup-closed-dual*: *infimum-closed* (*fp* φ) = *supremum-closed* (*fp* φ^d)
  ⟨*proof*⟩
lemma *fp-inf-sup-closed-dual'*: *infimum-closed'* (*fp* φ) = *supremum-closed'* (*fp* φ^d)
  ⟨*proof*⟩

We verify that being infimum-closed' (resp. supremum-closed') entails being meet-closed (resp. join-closed).

lemma *inf-meet-closed*: $\forall S.$ *infimum-closed'* $S \longrightarrow$ *meet-closed* S ⟨*proof*⟩
lemma *sup-join-closed*: $\forall P.$ *supremum-closed'* $P \longrightarrow$ *join-closed* P ⟨*proof*⟩

end
theory *conditions-positive*
  imports *boolean-algebra-operators*
begin

2 Topological Conditions

We define and interrelate some useful axiomatic conditions on unary operations (operators) having a 'w-parametric type $('w)\sigma \Rightarrow ('w)\sigma$. Boolean algebras extended with such operators give us different sorts of topological Boolean algebras.

2.1 Positive Conditions

Monotonicity (MONO).

definition *MONO*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*MONO*›)
  where *MONO* $\varphi \equiv \forall A\ B.\ A \leq B \longrightarrow \varphi\ A \leq \varphi\ B$

named-theorems *cond*
declare *MONO-def*[*cond*]

MONO is self-dual.

lemma *MONO-dual*: *MONO* φ = *MONO* φ^d ⟨*proof*⟩

Expansive/extensive (EXPN) and its dual contractive (CNTR).

definition *EXPN*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*EXPN*›)
  where *EXPN* $\varphi\ \equiv \forall A.\ A \leq \varphi\ A$
definition *CNTR*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*CNTR*›)
  where *CNTR* $\varphi \equiv \forall A.\ \varphi\ A \leq A$

declare *EXPN-def*[*cond*] *CNTR-def*[*cond*]

EXPN and CNTR are dual to each other.

lemma *EXPN-CNTR-dual1*: *EXPN* φ = *CNTR* φ^d ⟨*proof*⟩
lemma *EXPN-CNTR-dual2*: *CNTR* φ = *EXPN* φ^d ⟨*proof*⟩

Normality (NORM) and its dual (DNRM).

definition *NORM*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*NORM*›)
  where *NORM* $\varphi\ \equiv (\varphi\ \bot) = \bot$
definition *DNRM*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*DNRM*›)
  where *DNRM* $\varphi \equiv (\varphi\ \top) = \top$

declare *NORM-def*[*cond*] *DNRM-def*[*cond*]

NORM and DNRM are dual to each other.

lemma *NOR-dual1*: *NORM* φ = *DNRM* φ^d ⟨*proof*⟩
lemma *NOR-dual2*: *DNRM* φ = *NORM* φ^d ⟨*proof*⟩

EXPN (CNTR) entails DNRM (NORM).

lemma *EXPN-impl-DNRM*: *EXPN* $\varphi \longrightarrow$ *DNRM* φ ⟨*proof*⟩
lemma *CNTR-impl-NORM*: *CNTR* $\varphi \longrightarrow$ *NORM* φ ⟨*proof*⟩

Idempotence (IDEM).

definition *IDEM*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*IDEM*›)
  where *IDEM* $\varphi\ \equiv \forall A.\ \varphi(\varphi\ A) = (\varphi\ A)$
definition *IDEM-a*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*IDEM*^{a}›)
  where *IDEM*^{a} $\varphi \equiv \forall A.\ \varphi(\varphi\ A) \leq (\varphi\ A)$
definition *IDEM-b*::('*w* $\sigma \Rightarrow$ '*w* σ) $\Rightarrow$ *bool* (‹*IDEM*^{b}›)
  where *IDEM*^{b} $\varphi \equiv \forall A.\ \ (\varphi\ A) \leq \varphi(\varphi\ A)$

declare *IDEM-def*[*cond*] *IDEM-a-def*[*cond*] *IDEM-b-def*[*cond*]

IDEM-a and IDEM-b are dual to each other.

lemma *IDEM-dual1*: *IDEM*^{a} φ = *IDEM*^{b} φ^d ⟨*proof*⟩
lemma *IDEM-dual2*: *IDEM*^{b} φ = *IDEM*^{a} φ^d ⟨*proof*⟩

lemma *IDEM-char*: $IDEM\ \varphi = (IDEM^a\ \varphi \wedge IDEM^b\ \varphi)$ ⟨*proof*⟩
lemma *IDEM-dual*: $IDEM\ \varphi = IDEM\ \varphi^d$ ⟨*proof*⟩

EXPN (CNTR) entail IDEM-b (IDEM-a).

lemma *EXPN-impl-IDEM-b*: $EXPN\ \varphi \longrightarrow IDEM^b\ \varphi$ ⟨*proof*⟩
lemma *CNTR-impl-IDEM-a*: $CNTR\ \varphi \longrightarrow IDEM^a\ \varphi$ ⟨*proof*⟩

Moreover, IDEM has some other interesting characterizations. For example, via function composition:

lemma *IDEM-fun-comp-char*: $IDEM\ \varphi = (\varphi = \varphi \circ \varphi)$ ⟨*proof*⟩

Or having the property of collapsing the range and the set of fixed-points of an operator:

lemma *IDEM-range-fp-char*: $IDEM\ \varphi = (\llbracket\varphi\ \text{-}\rrbracket = fp\ \varphi)$ ⟨*proof*⟩

Distribution over joins or additivity (ADDI).

definition *ADDI*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*ADDI*›)
  where $ADDI\ \varphi \equiv \forall A\ B.\ \varphi(A \vee B) = (\varphi\ A) \vee (\varphi\ B)$
definition *ADDI-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$ADDI^a$›)
  where $ADDI^a\ \varphi \equiv \forall A\ B.\ \varphi(A \vee B) \leq (\varphi\ A) \vee (\varphi\ B)$
definition *ADDI-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$ADDI^b$›)
  where $ADDI^b\ \varphi \equiv \forall A\ B.\ (\varphi\ A) \vee (\varphi\ B) \leq \varphi(A \vee B)$

Distribution over meets or multiplicativity (MULT).

definition *MULT*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*MULT*›)
  where $MULT\ \varphi \equiv \forall A\ B.\ \varphi(A \wedge B) = (\varphi\ A) \wedge (\varphi\ B)$
definition *MULT-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$MULT^a$›)
  where $MULT^a\ \varphi \equiv \forall A\ B.\ \varphi(A \wedge B) \leq (\varphi\ A) \wedge (\varphi\ B)$
definition *MULT-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$MULT^b$›)
  where $MULT^b\ \varphi \equiv \forall A\ B.\ (\varphi\ A) \wedge (\varphi\ B) \leq \varphi(A \wedge B)$

declare *ADDI-def*[*cond*] *ADDI-a-def*[*cond*] *ADDI-b-def*[*cond*]
MULT-def[*cond*] *MULT-a-def*[*cond*] *MULT-b-def*[*cond*]

lemma *ADDI-char*: $ADDI\ \varphi = (ADDI^a\ \varphi \wedge ADDI^b\ \varphi)$ ⟨*proof*⟩
lemma *MULT-char*: $MULT\ \varphi = (MULT^a\ \varphi \wedge MULT^b\ \varphi)$ ⟨*proof*⟩

MONO, MULT-a and ADDI-b are equivalent.

lemma *MONO-MULTa*: $MULT^a\ \varphi = MONO\ \varphi$ ⟨*proof*⟩
lemma *MONO-ADDIb*: $ADDI^b\ \varphi = MONO\ \varphi$ ⟨*proof*⟩

Below we prove several duality relationships between ADDI(a/b) and MULT(a/b).

Duality between MULT-a and ADDI-b (an easy corollary from the self-duality of MONO).

lemma *MULTa-ADDIb-dual1*: $MULT^a\ \varphi = ADDI^b\ \varphi^d$ ⟨*proof*⟩
lemma *MULTa-ADDIb-dual2*: $ADDI^b\ \varphi = MULT^a\ \varphi^d$ ⟨*proof*⟩

Duality between ADDI-a and MULT-b.

lemma *ADDIa-MULTb-dual1*: $ADDI^a\ \varphi = MULT^b\ \varphi^d$ ⟨*proof*⟩
lemma *ADDIa-MULTb-dual2*: $MULT^b\ \varphi = ADDI^a\ \varphi^d$ ⟨*proof*⟩

Duality between ADDI and MULT.

lemma *ADDI-MULT-dual1*: $ADDI\ \varphi = MULT\ \varphi^d$ ⟨*proof*⟩

lemma *ADDI-MULT-dual2*: $MULT\ \varphi = ADDI\ \varphi^d$ ⟨*proof*⟩

We verify properties regarding closure over meets/joins for fixed-points.

MULT implies meet-closedness of the set of fixed-points (the converse requires additional assumptions).

lemma *MULT-meetclosed*: $MULT\ \varphi \Longrightarrow$ *meet-closed* (*fp* φ) ⟨*proof*⟩
lemma *meet-closed* (*fp* φ) $\Longrightarrow MULT\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *meetclosed-MULT*: $MONO\ \varphi \Longrightarrow CNTR\ \varphi \Longrightarrow IDEM^b\ \varphi \Longrightarrow$ *meet-closed* (*fp* φ) $\Longrightarrow MULT\ \varphi$ ⟨*proof*⟩

ADDI implies join-closedness of the set of fixed-points (the converse requires additional assumptions).

lemma *ADDI-joinclosed*: $ADDI\ \varphi \Longrightarrow$ *join-closed* (*fp* φ) ⟨*proof*⟩
lemma *join-closed* (*fp* φ) $\Longrightarrow ADDI\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *joinclosed-ADDI*: $MONO\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow IDEM^a\ \varphi \Longrightarrow$ *join-closed* (*fp* φ) $\Longrightarrow ADDI\ \varphi$ ⟨*proof*⟩

Assuming MONO, we have that EXPN (CNTR) implies meet-closed (join-closed) for the set of fixed-points.

lemma *EXPN-meetclosed*: $MONO\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow$ *meet-closed* (*fp* φ) ⟨*proof*⟩
lemma *CNTR-joinclosed*: $MONO\ \varphi \Longrightarrow CNTR\ \varphi \Longrightarrow$ *join-closed* (*fp* φ) ⟨*proof*⟩

Further assuming IDEM the above results can be stated to the whole range of an operator.

lemma $MONO\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow IDEM\ \varphi \Longrightarrow$ *meet-closed* ($[\![\varphi$ -$]\!]$) ⟨*proof*⟩
lemma $MONO\ \varphi \Longrightarrow CNTR\ \varphi \Longrightarrow IDEM\ \varphi \Longrightarrow$ *join-closed* ($[\![\varphi$ -$]\!]$) ⟨*proof*⟩

end
theory *conditions-positive-infinitary*
imports *conditions-positive boolean-algebra-infinitary*
begin

2.2 Infinitary Positive Conditions

We define and interrelate infinitary variants for some previously introduced axiomatic conditions on operators.

Distribution over infinite joins (suprema) or infinite additivity (iADDI).

definition $iADDI::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*iADDI*›)
where $iADDI\ \varphi \equiv \forall S.\ \varphi(\bigvee S) = \bigvee[\![\varphi\ S]\!]$
definition $iADDI\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$iADDI^a$›)
where $iADDI^a\ \varphi \equiv \forall S.\ \varphi(\bigvee S) \leq \bigvee[\![\varphi\ S]\!]$
definition $iADDI\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$iADDI^b$›)
where $iADDI^b\ \varphi \equiv \forall S.\ \bigvee[\![\varphi\ S]\!] \leq \varphi(\bigvee S)$

Distribution over infinite meets (infima) or infinite multiplicativity (iMULT).

definition $iMULT::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*iMULT*›)
where $iMULT\ \varphi \equiv \forall S.\ \varphi(\bigwedge S) = \bigwedge[\![\varphi\ S]\!]$
definition $iMULT\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$iMULT^a$›)
where $iMULT^a\ \varphi \equiv \forall S.\ \varphi(\bigwedge S) \leq \bigwedge[\![\varphi\ S]\!]$
definition $iMULT\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$iMULT^b$›)
where $iMULT^b\ \varphi \equiv \forall S.\ \bigwedge[\![\varphi\ S]\!] \leq \varphi(\bigwedge S)$

declare *iADDI-def*[*cond*] *iADDI-a-def*[*cond*] *iADDI-b-def*[*cond*]
iMULT-def[*cond*] *iMULT-a-def*[*cond*] *iMULT-b-def*[*cond*]

lemma *iADDI-char*: $iADDI\ \varphi = (iADDI^a\ \varphi \wedge iADDI^b\ \varphi)$ ⟨*proof*⟩
lemma *iMULT-char*: $iMULT\ \varphi = (iMULT^a\ \varphi \wedge iMULT^b\ \varphi)$ ⟨*proof*⟩

ADDI-b and iADDI-b are in fact equivalent.

lemma *iADDIb-equ*: $iADDI^b\ \varphi = ADDI^b\ \varphi$ ⟨*proof*⟩

MULT-a and iMULT-a are also equivalent.

lemma *iMULTa-equ*: $iMULT^a\ \varphi = MULT^a\ \varphi$ ⟨*proof*⟩

Thus we have that MONO, ADDI-b/iADDI-b and MULT-a/iMULT-a are all equivalent.

lemma *MONO-iADDIb*: $iADDI^b\ \varphi = MONO\ \varphi$ ⟨*proof*⟩
lemma *MONO-iMULTa*: $iMULT^a\ \varphi = MONO\ \varphi$ ⟨*proof*⟩

Below we prove several duality relationships between iADDI(a/b) and iMULT(a/b).

Duality between iMULT-a and iADDI-b (an easy corollary from the previous equivalence).

lemma *iMULTa-iADDIb-dual1*: $iMULT^a\ \varphi = iADDI^b\ \varphi^d$ ⟨*proof*⟩
lemma *iMULTa-iADDIb-dual2*: $iADDI^b\ \varphi = iMULT^a\ \varphi^d$ ⟨*proof*⟩

Duality between iADDI-a and iMULT-b.

lemma *iADDIa-iMULTb-dual1*: $iADDI^a\ \varphi = iMULT^b\ \varphi^d$ ⟨*proof*⟩
lemma *iADDIa-iMULTb-dual2*: $iMULT^b\ \varphi = iADDI^a\ \varphi^d$ ⟨*proof*⟩

Duality between iADDI and iMULT.

lemma *iADDI-iMULT-dual1*: $iADDI\ \varphi = iMULT\ \varphi^d$ ⟨*proof*⟩
lemma *iADDI-iMULT-dual2*: $iMULT\ \varphi = iADDI\ \varphi^d$ ⟨*proof*⟩

In fact, infinite additivity (multiplicativity) entails (dual) normality:

lemma *iADDI-NORM*: $iADDI\ \varphi \longrightarrow NORM\ \varphi$ ⟨*proof*⟩
lemma *iMULT-DNRM*: $iMULT\ \varphi \longrightarrow DNRM\ \varphi$ ⟨*proof*⟩

Suitable conditions on an operation can make the set of its fixed-points closed under infinite meets/joins.

lemma *fp-sup-closed-cond1*: $iADDI\ \varphi \longrightarrow$ *supremum-closed* (*fp* φ) ⟨*proof*⟩
lemma *fp-sup-closed-cond2*: $iADDI^a\ \varphi \wedge EXPN\ \varphi \longrightarrow$ *supremum-closed* (*fp* φ) ⟨*proof*⟩
lemma *fp-sup-closed-cond3*: $MONO\ \varphi \wedge CNTR\ \varphi \longrightarrow$ *supremum-closed* (*fp* φ) ⟨*proof*⟩

lemma *fp-inf-closed-cond1*: $iMULT\ \varphi \longrightarrow$ *infimum-closed* (*fp* φ) ⟨*proof*⟩
lemma *fp-inf-closed-cond2*: $iMULT^b\ \varphi \wedge CNTR\ \varphi \longrightarrow$ *infimum-closed* (*fp* φ) ⟨*proof*⟩
lemma *fp-inf-closed-cond3*: $MONO\ \varphi \wedge EXPN\ \varphi \longrightarrow$ *infimum-closed* (*fp* φ) ⟨*proof*⟩

OTOH, the converse conjectures have (finite) countermodels. They need additional assumptions.

lemma *infimum-closed* (*fp* φ) $\longrightarrow iMULT\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *supremum-closed* (*fp* φ) $\longrightarrow iADDI\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *fp-inf-closed-iMULT*: $MONO\ \varphi \Longrightarrow CNTR\ \varphi \Longrightarrow IDEM^b\ \varphi \Longrightarrow$ *infimum-closed* (*fp* φ) $\longrightarrow$ $iMULT\ \varphi$
⟨*proof*⟩
lemma *fp-sup-closed-iADDI*: $MONO\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow IDEM^a\ \varphi \Longrightarrow$ *supremum-closed* (*fp* φ) $\longrightarrow$ $iADDI\ \varphi$

⟨*proof*⟩

3 Alexandrov topologies and (generalized) specialization preorders

A topology is called 'Alexandrov' (after the Russian mathematician Pavel Alexandrov) if the intersection (resp. union) of any (finite or infinite) family of open (resp. closed) sets is open (resp. closed); in algebraic terms, this means that the set of fixed points of the interior (closure) operation is closed under infinite meets (joins). Another common algebraic formulation requires the closure (interior) operation to satisfy the infinitary variants of additivity (multiplicativity), i.e. iADDI (iMULT) as introduced before.

In the literature, the well-known Kuratowski conditions for the closure (resp. interior) operation are assumed, namely: ADDI, EXPN, NORM, IDEM (resp. MULT, CNTR, DNRM, IDEM). This makes both formulations equivalent. However, this is not the case in general if those conditions become negotiable.

Alexandrov topologies have interesting properties relating them to the semantics of modal logic. Assuming Kuratowski conditions, Alexandrov topological operations defined on subsets of S are in one-to-one correspondence with preorders on S; in topological terms, Alexandrov topologies are uniquely determined by their specialization preorders. Since we do not presuppose any Kuratowski conditions to begin with, the preorders in question are in general not even transitive. Here we just call them 'reachability relations'. We will still call (generalized) closure/interior-like operations as such (for lack of a better name). We explore minimal conditions under which some relevant results for the semantics of modal logic obtain.

Closure/interior(-like) operators can be derived from an arbitrary relation (as in modal logic).

definition *Cl-rel*::$('w \Rightarrow 'w \Rightarrow bool) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ ($\langle\mathcal{C}[\text{-}]\rangle$)
  where $\mathcal{C}[R] \equiv \lambda A.\ \lambda w.\ \exists\, v.\ R\ w\ v \wedge A\ v$
definition *Int-rel*::$('w \Rightarrow 'w \Rightarrow bool) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ ($\langle\mathcal{I}[\text{-}]\rangle$)
  where $\mathcal{I}[R] \equiv \lambda A.\ \lambda w.\ \forall\, v.\ R\ w\ v \longrightarrow A\ v$

Duality between interior and closure follows directly:

lemma *dual-CI*: $\mathcal{C}[R] = \mathcal{I}[R]^d$ $\langle proof\rangle$

We explore minimal conditions of the reachability relation under which some operation's conditions obtain.

Define some relevant properties of relations:

abbreviation *serial* $R \equiv \forall x.\ \exists y.\ R\ x\ y$
abbreviation *reflexive* $R \equiv \forall x.\ R\ x\ x$
abbreviation *transitive* $R \equiv \forall x\ y\ z.\ R\ x\ y \wedge R\ y\ z \longrightarrow R\ x\ z$
abbreviation *antisymmetric* $R \equiv \forall x\ y.\ R\ x\ y \wedge R\ y\ x \longrightarrow x = y$
abbreviation *symmetric* $R \equiv \forall x\ y.\ R\ x\ y \longrightarrow R\ y\ x$

lemma *rC1*: *iADDI* $\mathcal{C}[R]$ $\langle proof\rangle$
lemma *rC2*: *reflexive* R = *EXPN* $\mathcal{C}[R]$ $\langle proof\rangle$
lemma *rC3*: *NORM* $\mathcal{C}[R]$ $\langle proof\rangle$
lemma *rC4*: *transitive* R = *IDEM*^{a} $\mathcal{C}[R]$ $\langle proof\rangle$

A reachability (specialization) relation (preorder) can be derived from a given operation (intended as a closure-like operation).

definition $\mathcal{R}$::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w \Rightarrow 'w \Rightarrow bool)$ ($\langle\mathcal{R}[\text{-}]\rangle$)
  where $\mathcal{R}[\varphi] \equiv \lambda w\ v.\ \varphi\ (\lambda x.\ x = v)\ w$

Preorder properties of the reachability relation follow from the corresponding operation's conditions.

lemma *rel-refl*: $EXPN\ \varphi \longrightarrow reflexive\ \mathcal{R}[\varphi]$ ⟨*proof*⟩
lemma *rel-trans*: $MONO\ \varphi \wedge IDEM^a\ \varphi \longrightarrow transitive\ \mathcal{R}[\varphi]$ ⟨*proof*⟩
lemma $IDEM^a\ \varphi \longrightarrow transitive\ \mathcal{R}[\varphi]$ **nitpick** ⟨*proof*⟩

lemma $reflexive\ \mathcal{R}[\varphi] \longrightarrow EXPN\ \varphi$ **nitpick** ⟨*proof*⟩
lemma $transitive\ \mathcal{R}[\varphi] \longrightarrow IDEM^a\ \varphi$ **nitpick** ⟨*proof*⟩
lemma $transitive\ \mathcal{R}[\varphi] \longrightarrow MONO\ \varphi$ **nitpick** ⟨*proof*⟩

However, we can obtain finite countermodels for antisymmetry and symmetry given all relevant conditions. We will revisit this issue later and examine their relation with the topological separation axioms T0 and T1 resp.

lemma $iADDI\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow IDEM^a\ \varphi \Longrightarrow antisymmetric\ \mathcal{R}[\varphi]$ **nitpick** ⟨*proof*⟩
lemma $iADDI\ \varphi \Longrightarrow EXPN\ \varphi \Longrightarrow IDEM^a\ \varphi \Longrightarrow symmetric\ \mathcal{R}[\varphi]$ **nitpick** ⟨*proof*⟩

As mentioned previously, Alexandrov closure (and by duality interior) operations correspond to specialization orderings (reachability relations). It is worth mentioning that in Alexandrov topologies every point has a minimal/smallest neighborhood, namely the set of points related to it by the specialization preorder (reachability relation). We examine below minimal conditions under which these relations obtain.

lemma *Cl-rel′-a*: $MONO\ \varphi \longrightarrow (\forall A.\ \mathcal{C}[\mathcal{R}[\varphi]]\ A \leq \varphi\ A)$ ⟨*proof*⟩
lemma *Cl-rel′-b*: $iADDI^a\ \varphi \longrightarrow (\forall A.\ \varphi\ A \leq \mathcal{C}[\mathcal{R}[\varphi]]\ A)$ ⟨*proof*⟩
lemma *Cl-rel′*: $iADDI\ \varphi \longrightarrow \varphi =^{\cdot} \mathcal{C}[\mathcal{R}[\varphi]]$ ⟨*proof*⟩
lemma *Cl-rel*: $iADDI\ \varphi \longleftrightarrow \varphi = \mathcal{C}[\mathcal{R}[\varphi]]$ ⟨*proof*⟩

It is instructive to expand the definitions in the above result:

lemma $iADDI\ \varphi \longleftrightarrow (\forall A.\ \forall w.\ (\varphi\ A)\ w \longleftrightarrow (\exists v.\ A\ v \wedge \varphi\ (\lambda x.\ x{=}v)\ w))$ ⟨*proof*⟩

Closure (interior) operations derived from relations are thus closed under infinite joins (meets).

lemma $supremum\text{-}closed\ (fp\ \mathcal{C}[R])$ ⟨*proof*⟩
lemma $infimum\text{-}closed\ \ (fp\ \mathcal{I}[R])$ ⟨*proof*⟩

We can now revisit the relationship between (anti)symmetry and the separation axioms T1 and T0.

T0: any two distinct points in the space can be separated by a closed (or open) set (i.e. containing one point and not the other).

abbreviation $T0\ \mathcal{C} \equiv (\forall p\ q.\ p \neq q \longrightarrow (\exists G.\ (fp\ \mathcal{C})\ G \wedge \neg(G\ p \longleftrightarrow G\ q)))$

T1: any two distinct points can be separated by (two not necessarily disjoint) closed (or open) sets.

abbreviation $T1\ \mathcal{C} \equiv (\forall p\ q.\ p \neq q \longrightarrow (\exists G\ H.\ (fp\ \mathcal{C})\ G \wedge (fp\ \mathcal{C})\ H \wedge G\ p \wedge \neg G\ q \wedge H\ q \wedge \neg H\ p))$

We can (sanity) check that T1 entails T0 but not viceversa.

lemma $T1\ \mathcal{C} \Longrightarrow T0\ \mathcal{C}$ ⟨*proof*⟩
lemma $T0\ \mathcal{C} \Longrightarrow T1\ \mathcal{C}$ **nitpick** ⟨*proof*⟩

Under appropriate conditions, T0-separation corresponds to antisymmetry of the specialization relation (here an ordering).

lemma $T0\ \mathcal{C} \longleftrightarrow antisymmetric\ \mathcal{R}[\mathcal{C}]$ **nitpick** ⟨*proof*⟩

lemma *T0-antisymm-a*: $MONO\ \mathcal{C} \Longrightarrow T0\ \mathcal{C} \longrightarrow antisymmetric\ \mathcal{R}[\mathcal{C}]$ ⟨*proof*⟩
lemma *T0-antisymm-b*: $EXPN\ \mathcal{C} \Longrightarrow IDEM^a\ \mathcal{C} \Longrightarrow antisymmetric\ \mathcal{R}[\mathcal{C}] \longrightarrow T0\ \mathcal{C}$ ⟨*proof*⟩
lemma *T0-antisymm*: $MONO\ \mathcal{C} \Longrightarrow EXPN\ \mathcal{C} \Longrightarrow IDEM^a\ \mathcal{C} \Longrightarrow T0\ \mathcal{C} = antisymmetric\ \mathcal{R}[\mathcal{C}]$ ⟨*proof*⟩

Also, under the appropriate conditions, T1-separation corresponds to symmetry of the specialization relation.

lemma *T1-symm-a*: $MONO\ \mathcal{C} \Longrightarrow T1\ \mathcal{C} \longrightarrow symmetric\ \mathcal{R}[\mathcal{C}]$ ⟨*proof*⟩
lemma *T1-symm-b*: $MONO\ \mathcal{C} \Longrightarrow EXPN\ \mathcal{C} \Longrightarrow T0\ \mathcal{C} \Longrightarrow symmetric\ \mathcal{R}[\mathcal{C}] \longrightarrow T1\ \mathcal{C}$ ⟨*proof*⟩
lemma *T1-symm*: $MONO\ \mathcal{C} \Longrightarrow EXPN\ \mathcal{C} \Longrightarrow T0\ \mathcal{C} \Longrightarrow symmetric\ \mathcal{R}[\mathcal{C}] = T1\ \mathcal{C}$ ⟨*proof*⟩

end
theory *conditions-negative*
  imports *conditions-positive*
begin

3.1 Negative Conditions

We continue defining and interrelating axiomatic conditions on unary operations (operators). We now move to conditions commonly satisfied by negation-like logical operations.

Anti-tonicity (ANTI).

definition $ANTI::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*ANTI*›)
  where $ANTI\ \varphi \equiv \forall\, A\ B.\ A \leq B \longrightarrow \varphi\ B \leq \varphi\ A$

declare *ANTI-def*[*cond*]

ANTI is self-dual.

lemma *ANTI-dual*: $ANTI\ \varphi = ANTI\ \varphi^d$ ⟨*proof*⟩

ANTI is the 'complement' of MONO.

lemma *ANTI-MONO*: $MONO\ \varphi = ANTI\ \varphi^-$ ⟨*proof*⟩

Anti-expansive/extensive (nEXPN) and its dual anti-contractive (nCNTR).

definition $nEXPN::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nEXPN*›)
  where $nEXPN\ \varphi \equiv \forall\, A.\ \varphi\ A \leq -A$
definition $nCNTR::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nCNTR*›)
  where $nCNTR\ \varphi \equiv \forall\, A.\ -A \leq \varphi\ A$

declare *nEXPN-def*[*cond*] *nCNTR-def*[*cond*]

nEXPN and nCNTR are dual to each other.

lemma *nEXPN-nCNTR-dual1*: $nEXPN\ \varphi = nCNTR\ \varphi^d$ ⟨*proof*⟩
lemma *nEXPN-nCNTR-dual2*: $nCNTR\ \varphi = nEXPN\ \varphi^d$ ⟨*proof*⟩

nEXPN and nCNTR are the 'complements' of EXPN and CNTR respectively.

lemma *nEXPN-CNTR-compl*: $EXPN\ \varphi = nEXPN\ \varphi^-$ ⟨*proof*⟩
lemma *nCNTR-EXPN-compl*: $CNTR\ \varphi = nCNTR\ \varphi^-$ ⟨*proof*⟩

Anti-Normality (nNORM) and its dual (nDNRM).

definition $nNORM::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nNORM*›)
  where $nNORM\ \varphi \equiv (\varphi\ \bot) = \top$
definition $nDNRM::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nDNRM*›)
  where $nDNRM\ \varphi \equiv (\varphi\ \top) = \bot$

declare *nNORM-def*[*cond*] *nDNRM-def*[*cond*]

nNORM and nDNRM are dual to each other.

lemma *nNOR-dual1*: $nNORM\ \varphi = nDNRM\ \varphi^d\ \langle proof \rangle$
lemma *nNOR-dual2*: $nDNRM\ \varphi = nNORM\ \varphi^d\ \langle proof \rangle$

nNORM and nDNRM are the 'complements' of NORM and DNRM respectively.

lemma *nNORM-NORM-compl*: $NORM\ \varphi = nNORM\ \varphi^-\ \langle proof \rangle$
lemma *nDNRM-DNRM-compl*: $DNRM\ \varphi = nDNRM\ \varphi^-\ \langle proof \rangle$

nEXPN (nCNTR) entail nDNRM (nNORM).

lemma *nEXPN-impl-nDNRM*: $nEXPN\ \varphi \longrightarrow nDNRM\ \varphi\ \langle proof \rangle$
lemma *nCNTR-impl-nNORM*: $nCNTR\ \varphi \longrightarrow nNORM\ \varphi\ \langle proof \rangle$

Anti-Idempotence (nIDEM).

definition *nIDEM*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nIDEM \rangle$)
  where $nIDEM\ \varphi\ \equiv \forall A.\ \varphi(-(\varphi\ A)) = (\varphi\ A)$
definition *nIDEM-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nIDEM^a \rangle$)
  where $nIDEM\text{-}a\ \varphi \equiv \forall A.\ (\varphi\ A) \leq \varphi(-(\varphi\ A))$
definition *nIDEM-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nIDEM^b \rangle$)
  where $nIDEM\text{-}b\ \varphi \equiv \forall A.\ \varphi(-(\varphi\ A)) \leq (\varphi\ A)$

declare *nIDEM-def*[*cond*] *nIDEM-a-def*[*cond*] *nIDEM-b-def*[*cond*]

nIDEM-a and nIDEM-b are dual to each other.

lemma *nIDEM-dual1*: $nIDEM^a\ \varphi = nIDEM^b\ \varphi^d\ \langle proof \rangle$
lemma *nIDEM-dual2*: $nIDEM^b\ \varphi = nIDEM^a\ \varphi^d\ \langle proof \rangle$

lemma *nIDEM-char*: $nIDEM\ \varphi = (nIDEM^a\ \varphi \wedge nIDEM^b\ \varphi)\ \langle proof \rangle$
lemma *nIDEM-dual*: $nIDEM\ \varphi = nIDEM\ \varphi^d\ \langle proof \rangle$

nIDEM(a/b) and IDEM(a/b) are the 'complements' each other.

lemma *nIDEM-a-compl*: $IDEM^a\ \varphi = nIDEM^a\ \varphi^-\ \langle proof \rangle$
lemma *nIDEM-b-compl*: $IDEM^b\ \varphi = nIDEM^b\ \varphi^-\ \langle proof \rangle$
lemma *nIDEM-compl*: $nIDEM\ \varphi = IDEM\ \varphi^-\ \langle proof \rangle$

nEXPN (nCNTR) entail nIDEM-a (nIDEM-b).

lemma *nEXPN-impl-nIDEM-a*: $nEXPN\ \varphi \longrightarrow nIDEM^b\ \varphi\ \langle proof \rangle$
lemma *nCNTR-impl-nIDEM-b*: $nCNTR\ \varphi \longrightarrow nIDEM^a\ \varphi\ \langle proof \rangle$

Anti-distribution over joins or anti-additivity (nADDI) and its dual.

definition *nADDI*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nADDI \rangle$)
  where $nADDI\ \varphi\ \equiv \forall A\ B.\ \varphi(A \vee B) = (\varphi\ A) \wedge (\varphi\ B)$
definition *nADDI-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nADDI^a \rangle$)
  where $nADDI^a\ \varphi \equiv \forall A\ B.\ (\varphi\ A) \wedge (\varphi\ B) \leq \varphi(A \vee B)$
definition *nADDI-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nADDI^b \rangle$)
  where $nADDI^b\ \varphi \equiv \forall A\ B.\ \varphi(A \vee B) \leq (\varphi\ A) \wedge (\varphi\ B)$

Anti-distribution over meets or anti-multiplicativity (nMULT).

definition *nMULT*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nMULT \rangle$)
  where $nMULT\ \varphi\ \equiv \forall A\ B.\ \varphi(A \wedge B) = (\varphi\ A) \vee (\varphi\ B)$
definition *nMULT-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ ($\langle nMULT^a \rangle$)

where $nMULT^a\ \varphi \equiv \forall\ A\ B.\ (\varphi\ A) \vee (\varphi\ B) \leq \varphi(A \wedge B)$
definition $nMULT\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle nMULT^b \rangle)$
where $nMULT^b\ \varphi \equiv \forall\ A\ B.\ \varphi(A \wedge B) \leq (\varphi\ A) \vee (\varphi\ B)$

declare *nADDI-def*[*cond*] *nADDI-a-def*[*cond*] *nADDI-b-def*[*cond*]
nMULT-def[*cond*] *nMULT-a-def*[*cond*] *nMULT-b-def*[*cond*]

lemma *nADDI-char*: $nADDI\ \varphi = (nADDI^a\ \varphi \wedge nADDI^b\ \varphi)$ ⟨*proof*⟩
lemma *nMULT-char*: $nMULT\ \varphi = (nMULT^a\ \varphi \wedge nMULT^b\ \varphi)$ ⟨*proof*⟩

ANTI, nMULT-a and nADDI-b are equivalent.

lemma *ANTI-nMULTa*: $nMULT^a\ \varphi = ANTI\ \varphi$ ⟨*proof*⟩
lemma *ANTI-nADDIb*: $nADDI^b\ \varphi = ANTI\ \varphi$ ⟨*proof*⟩

Below we prove several duality relationships between nADDI(a/b) and nMULT(a/b).

Duality between nMULT-a and nADDI-b (an easy corollary from the self-duality of ANTI).

lemma *nMULTa-nADDIb-dual1*: $nMULT^a\ \varphi = nADDI^b\ \varphi^d$ ⟨*proof*⟩
lemma *nMULTa-nADDIb-dual2*: $nADDI^b\ \varphi = nMULT^a\ \varphi^d$ ⟨*proof*⟩

Duality between nADDI-a and nMULT-b.

lemma *nADDIa-nMULTb-dual1*: $nADDI^a\ \varphi = nMULT^b\ \varphi^d$ ⟨*proof*⟩
lemma *nADDIa-nMULTb-dual2*: $nMULT^b\ \varphi = nADDI^a\ \varphi^d$ ⟨*proof*⟩

Duality between ADDI and MULT.

lemma *nADDI-nMULT-dual1*: $nADDI\ \varphi = nMULT\ \varphi^d$ ⟨*proof*⟩
lemma *nADDI-nMULT-dual2*: $nMULT\ \varphi = nADDI\ \varphi^d$ ⟨*proof*⟩

nADDI and nMULT are the 'complements' of ADDI and MULT respectively.

lemma *nADDIa-compl*: $ADDI^a\ \varphi = nADDI^a\ \varphi^-$ ⟨*proof*⟩
lemma *nADDIb-compl*: $ADDI^b\ \varphi = nADDI^b\ \varphi^-$ ⟨*proof*⟩
lemma *nADDI-compl*: $ADDI\ \varphi = nADDI\ \varphi^-$ ⟨*proof*⟩
lemma *nMULTa-compl*: $MULT^a\ \varphi = nMULT^a\ \varphi^-$ ⟨*proof*⟩
lemma *nMULTb-compl*: $MULT^b\ \varphi = nMULT^b\ \varphi^-$ ⟨*proof*⟩
lemma *nMULT-compl*: $MULT\ \varphi = nMULT\ \varphi^-$ ⟨*proof*⟩

We verify properties regarding closure over meets/joins for fixed-points.

nMULT for an operator implies join-closedness of the set of fixed-points of its dual-complement.

lemma *nMULT-joinclosed*: $nMULT\ \varphi \Longrightarrow join\text{-}closed\ (fp\ (\varphi^{d-}))$ ⟨*proof*⟩
lemma $join\text{-}closed\ (fp\ (\varphi^{d-})) \Longrightarrow nMULT\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *joinclosed-nMULT*: $ANTI\ \varphi \Longrightarrow nCNTR\ \varphi \Longrightarrow nIDEM^b\ \varphi \Longrightarrow join\text{-}closed\ (fp\ (\varphi^{d-})) \Longrightarrow nMULT\ \varphi$ ⟨*proof*⟩

nADDI for an operator implies meet-closedness of the set of fixed-points of its dual-complement.

lemma *nADDI-meetclosed*: $nADDI\ \varphi \Longrightarrow meet\text{-}closed\ (fp\ (\varphi^{d-}))$ ⟨*proof*⟩
lemma $meet\text{-}closed\ (fp\ (\varphi^{d-})) \Longrightarrow nADDI\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *meetclosed-nADDI*: $ANTI\ \varphi \Longrightarrow nEXPN\ \varphi \Longrightarrow nIDEM^a\ \varphi \Longrightarrow meet\text{-}closed\ (fp\ (\varphi^{d-})) \Longrightarrow nADDI\ \varphi$ ⟨*proof*⟩

Assuming ANTI, we have that nEXPN (nCNTR) implies meet-closed (join-closed) for the set of fixed-points.

lemma *nEXPN-meetclosed*: $ANTI\ \varphi \Longrightarrow nEXPN\ \varphi \Longrightarrow meet\text{-}closed\ (fp\ \varphi)$ ⟨*proof*⟩
lemma *nCNTR-joinclosed*: $ANTI\ \varphi \Longrightarrow nCNTR\ \varphi \Longrightarrow join\text{-}closed\ (fp\ \varphi)$ ⟨*proof*⟩

end
theory *conditions-negative-infinitary*
imports *conditions-negative conditions-positive-infinitary*
begin

3.2 Infinitary Negative Conditions

We define and interrelate infinitary variants for some previously introduced axiomatic conditions on operators.

Anti-distribution over infinite joins (suprema) or infinite anti-additivity (inADDI).

definition $inADDI::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDI \rangle)$
where $inADDI\ \varphi \equiv \forall S.\ \varphi(\bigvee S) = \bigwedge[\![\varphi\ S]\!]$
definition $inADDI\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDI^a \rangle)$
where $inADDI^a\ \varphi \equiv \forall S.\ \bigwedge[\![\varphi\ S]\!] \leq \varphi(\bigvee S)$
definition $inADDI\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDI^b \rangle)$
where $inADDI^b\ \varphi \equiv \forall S.\ \varphi(\bigvee S) \leq \bigwedge[\![\varphi\ S]\!]$

Anti-distribution over infinite meets (infima) or infinite anti-multiplicativity (inMULT).

definition $inMULT::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULT \rangle)$
where $inMULT\ \varphi \equiv \forall S.\ \varphi(\bigwedge S) = \bigvee[\![\varphi\ S]\!]$
definition $inMULT\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULT^a \rangle)$
where $inMULT^a\ \varphi \equiv \forall S.\ \bigvee[\![\varphi\ S]\!] \leq \varphi(\bigwedge S)$
definition $inMULT\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULT^b \rangle)$
where $inMULT^b\ \varphi \equiv \forall S.\ \varphi(\bigwedge S) \leq \bigvee[\![\varphi\ S]\!]$

declare *inADDI-def*[*cond*] *inADDI-a-def*[*cond*] *inADDI-b-def*[*cond*]
inMULT-def[*cond*] *inMULT-a-def*[*cond*] *inMULT-b-def*[*cond*]

lemma *inADDI-char*: $inADDI\ \varphi = (inADDI^a\ \varphi \wedge inADDI^b\ \varphi)$ ⟨*proof*⟩
lemma *inMULT-char*: $inMULT\ \varphi = (inMULT^a\ \varphi \wedge inMULT^b\ \varphi)$ ⟨*proof*⟩

nADDI-b and inADDI-b are in fact equivalent.

lemma *inADDIb-equ*: $inADDI^b\ \varphi = nADDI^b\ \varphi$ ⟨*proof*⟩

nMULT-a and inMULT-a are also equivalent.

lemma *inMULTa-equ*: $inMULT^a\ \varphi = nMULT^a\ \varphi$ ⟨*proof*⟩

Thus we have that ANTI, nADDI-b/inADDI-b and nMULT-a/inMULT-a are all equivalent.

lemma *ANTI-inADDIb*: $inADDI^b\ \varphi = ANTI\ \varphi$ ⟨*proof*⟩
lemma *ANTI-inMULTa*: $inMULT^a\ \varphi = ANTI\ \varphi$ ⟨*proof*⟩

Below we prove several duality relationships between inADDI(a/b) and inMULT(a/b).

Duality between inMULT-a and inADDI-b (an easy corollary from the previous equivalence).

lemma *inMULTa-inADDIb-dual1*: $inMULT^a\ \varphi = inADDI^b\ \varphi^d$ ⟨*proof*⟩
lemma *inMULTa-inADDIb-dual2*: $inADDI^b\ \varphi = inMULT^a\ \varphi^d$ ⟨*proof*⟩

Duality between inADDI-a and inMULT-b.

lemma *inADDIa-inMULTb-dual1*: $inADDI^a\ \varphi = inMULT^b\ \varphi^d$ ⟨*proof*⟩
lemma *inADDIa-inMULTb-dual2*: $inMULT^b\ \varphi = inADDI^a\ \varphi^d$ ⟨*proof*⟩

Duality between inADDI and inMULT.

lemma *inADDI-inMULT-dual1*: $inADDI\ \varphi = inMULT\ \varphi^d$ ⟨*proof*⟩
lemma *inADDI-inMULT-dual2*: $inMULT\ \varphi = inADDI\ \varphi^d$ ⟨*proof*⟩

inADDI and inMULT are the 'complements' of iADDI and iMULT respectively.

lemma *inADDIa-compl*: $iADDI^a\ \varphi = inADDI^a\ \varphi^-$ ⟨*proof*⟩
lemma *inADDIb-compl*: $iADDI^b\ \varphi = inADDI^b\ \varphi^-$ ⟨*proof*⟩
lemma *inADDI-compl*: $iADDI\ \varphi = inADDI\ \varphi^-$ ⟨*proof*⟩
lemma *inMULTa-compl*: $iMULT^a\ \varphi = inMULT^a\ \varphi^-$ ⟨*proof*⟩
lemma *inMULTb-compl*: $iMULT^b\ \varphi = inMULT^b\ \varphi^-$ ⟨*proof*⟩
lemma *inMULT-compl*: $iMULT\ \varphi = inMULT\ \varphi^-$ ⟨*proof*⟩

In fact, infinite anti-additivity (anti-multiplicativity) entails (dual) anti-normality:

lemma *inADDI-nNORM*: $inADDI\ \varphi \longrightarrow nNORM\ \varphi$ ⟨*proof*⟩
lemma *inMULT-nDNRM*: $inMULT\ \varphi \longrightarrow nDNRM\ \varphi$ ⟨*proof*⟩

end
theory *conditions-relativized*
imports *conditions-negative*
begin

3.3 Relativized Conditions

We continue defining and interrelating axiomatic conditions on unary operations (operators). This time we consider their 'relativized' variants.

Relativized order and equality relations.

definition *subset-in*::‹'p σ ⇒ 'p σ ⇒ 'p σ ⇒ bool› (‹-≤_-›)
where ‹$A \leq_U B \equiv \forall x.\ U\ x \longrightarrow (A\ x \longrightarrow B\ x)$›
definition *subset-out*::‹'p σ ⇒ 'p σ ⇒ 'p σ ⇒ bool› (‹-≤⁻-›)
where ‹$A \leq^U B \equiv \forall x.\ \neg U\ x \longrightarrow (A\ x \longrightarrow B\ x)$›
definition *setequ-in*::‹'p σ ⇒ 'p σ ⇒ 'p σ ⇒ bool› (‹-=_-›)
where ‹$A =_U B \equiv \forall x.\ U\ x \longrightarrow (A\ x \longleftrightarrow B\ x)$›
definition *setequ-out*::‹'p σ ⇒ 'p σ ⇒ 'p σ ⇒ bool› (‹-=⁻-›)
where ‹$A =^U B \equiv \forall x.\ \neg U\ x \longrightarrow (A\ x \longleftrightarrow B\ x)$›

declare *subset-in-def*[*order*] *subset-out-def*[*order*] *setequ-in-def*[*order*] *setequ-out-def*[*order*]

lemma *subset-in-out*: $(let\ U{=}C\ in\ (A \leq_U B)) = (let\ U{=}{-}C\ in\ (A \leq^U B))$ ⟨*proof*⟩
lemma *setequ-in-out*: $(let\ U{=}C\ in\ (A =_U B)) = (let\ U{=}{-}C\ in\ (A =^U B))$ ⟨*proof*⟩

lemma *subset-in-char*: $(A \leq_U B) = (U \wedge A \leq U \wedge B)$ ⟨*proof*⟩
lemma *subset-out-char*: $(A \leq^U B) = (U \vee A \leq U \vee B)$ ⟨*proof*⟩
lemma *setequ-in-char*: $(A =_U B) = (U \wedge A = U \wedge B)$ ⟨*proof*⟩
lemma *setequ-out-char*: $(A =^U B) = (U \vee A = U \vee B)$ ⟨*proof*⟩

Relativization cannot be meaningfully applied to conditions (n)NORM or (n)DNRM.

lemma $NORM\ \varphi\ = (let\ U = \top\ in\ ((\varphi\ \bot) =_U \bot))$ ⟨*proof*⟩
lemma $(let\ U = \bot\ in\ ((\varphi\ \bot) =_U \bot))$ ⟨*proof*⟩

Relativization ('in' resp. 'out') leaves (n)EXPN/(n)CNTR unchanged or trivializes them.

lemma $EXPN\ \varphi\ = (\forall A.\ A \leq_A \varphi\ A)$ ⟨*proof*⟩
lemma $CNTR\ \varphi\ = (\forall A.\ (\varphi\ A) \leq^A A)$ ⟨*proof*⟩
lemma $\forall A.\ A \leq^A \varphi\ A$ ⟨*proof*⟩
lemma $\forall A.\ (\varphi\ A) \leq_A A$ ⟨*proof*⟩

Relativized ADDI variants.

definition $ADDIr::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle ADDIr \rangle)$
  where $ADDIr\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \vee B)\ in \quad (\varphi(A \vee B) =^U (\varphi\ A) \vee (\varphi\ B))$
definition $ADDIr\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle ADDIr^a \rangle)$
  where $ADDIr^a\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \vee B)\ in \quad (\varphi(A \vee B) \leq^U (\varphi\ A) \vee (\varphi\ B))$
definition $ADDIr\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle ADDIr^b \rangle)$
  where $ADDIr^b\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \vee B)\ in\ ((\varphi\ A) \vee (\varphi\ B) \leq^U \varphi(A \vee B))$

declare $ADDIr\text{-}def[cond]\ ADDIr\text{-}a\text{-}def[cond]\ ADDIr\text{-}b\text{-}def[cond]$

lemma $ADDIr\text{-}char$: $ADDIr\ \varphi = (ADDIr^a\ \varphi \wedge ADDIr^b\ \varphi)\ \langle proof \rangle$

lemma $ADDIr\text{-}a\text{-}impl$: $ADDI^a\ \varphi \longrightarrow ADDIr^a\ \varphi\ \langle proof \rangle$
lemma $ADDIr\text{-}a\text{-}equ$: $EXPN\ \varphi \Longrightarrow ADDIr^a\ \varphi = ADDI^a\ \varphi\ \langle proof \rangle$
lemma $ADDIr\text{-}a\text{-}equ'$:$nEXPN\ \varphi \Longrightarrow ADDIr^a\ \varphi = ADDI^a\ \varphi\ \langle proof \rangle$

lemma $ADDIr\text{-}b\text{-}impl$: $ADDI^b\ \varphi \longrightarrow ADDIr^b\ \varphi\ \langle proof \rangle$
lemma $nEXPN\ \varphi \Longrightarrow ADDIr^b\ \varphi \longrightarrow ADDI^b\ \varphi$ **nitpick** $\langle proof \rangle$
lemma $ADDIr\text{-}b\text{-}equ$: $EXPN\ \varphi \Longrightarrow ADDIr^b\ \varphi = ADDI^b\ \varphi\ \langle proof \rangle$

Relativized MULT variants.

definition $MULTr::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle MULTr \rangle)$
  where $MULTr\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \wedge B)\ in \quad (\varphi(A \wedge B) =_U (\varphi\ A) \wedge (\varphi\ B))$
definition $MULTr\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle MULTr^a \rangle)$
  where $MULTr^a\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \wedge B)\ in \quad (\varphi(A \wedge B) \leq_U (\varphi\ A) \wedge (\varphi\ B))$
definition $MULTr\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle MULTr^b \rangle)$
  where $MULTr^b\ \varphi \equiv \forall\, A\ B.\ let\ U = (A \wedge B)\ in\ ((\varphi\ A) \wedge (\varphi\ B) \leq_U \varphi(A \wedge B))$

declare $MULTr\text{-}def[cond]\ MULTr\text{-}a\text{-}def[cond]\ MULTr\text{-}b\text{-}def[cond]$

lemma $MULTr\text{-}char$: $MULTr\ \varphi = (MULTr^a\ \varphi \wedge MULTr^b\ \varphi)\ \langle proof \rangle$

lemma $MULTr\text{-}a\text{-}impl$: $MULT^a\ \varphi \longrightarrow MULTr^a\ \varphi\ \langle proof \rangle$
lemma $nCNTR\ \varphi \Longrightarrow MULTr^a\ \varphi \longrightarrow MULT^a\ \varphi$ **nitpick** $\langle proof \rangle$
lemma $MULTr\text{-}a\text{-}equ$: $CNTR\ \varphi \Longrightarrow MULTr^a\ \varphi = MULT^a\ \varphi\ \langle proof \rangle$

lemma $MULTr\text{-}b\text{-}impl$: $MULT^b\ \varphi \longrightarrow MULTr^b\ \varphi\ \langle proof \rangle$
lemma $MULTr^b\ \varphi \longrightarrow MULT^b\ \varphi$ **nitpick** $\langle proof \rangle$
lemma $MULTr\text{-}b\text{-}equ$: $CNTR\ \varphi \Longrightarrow MULTr^b\ \varphi = MULT^b\ \varphi\ \langle proof \rangle$
lemma $MULTr\text{-}b\text{-}equ'$:$nCNTR\ \varphi \Longrightarrow MULTr^b\ \varphi = MULT^b\ \varphi\ \langle proof \rangle$

Weak variants of monotonicity.

definition $MONOw1::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle MONOw^1 \rangle)$
  where $MONOw^1\ \varphi \equiv \forall\, A\ B.\ A \leq B \longrightarrow (\varphi\ A) \leq B \vee (\varphi\ B)$
definition $MONOw2::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle MONOw^2 \rangle)$
  where $MONOw^2\ \varphi \equiv \forall\, A\ B.\ A \leq B \longrightarrow A \wedge (\varphi\ A) \leq (\varphi\ B)$

declare $MONOw1\text{-}def[cond]\ MONOw2\text{-}def[cond]$

lemma $MONOw1\text{-}ADDIr\text{-}b$: $MONOw^1\ \varphi = ADDIr^b\ \varphi\ \langle proof \rangle$
lemma $MONOw2\text{-}MULTr\text{-}a$: $MONOw^2\ \varphi = MULTr^a\ \varphi\ \langle proof \rangle$

lemma $MONOw1\text{-}impl$: $MONO\ \varphi \longrightarrow MONOw^1\ \varphi\ \langle proof \rangle$
lemma $MONOw^1\ \varphi \longrightarrow MONO\ \varphi$ **nitpick** $\langle proof \rangle$
lemma $MONOw2\text{-}impl$: $MONO\ \varphi \longrightarrow MONOw^2\ \varphi\ \langle proof \rangle$

lemma $MONOw^2\ \varphi \longrightarrow MONO\ \varphi$ **nitpick** ⟨*proof*⟩

We have in fact that (n)CNTR (resp. (n)EXPN) implies MONOw-1/ADDIr-b (resp. MONOw-2/MULTr-a).

lemma *CNTR-MONOw1-impl*: $CNTR\ \varphi \longrightarrow MONOw^1\ \varphi$ ⟨*proof*⟩
lemma *nCNTR-MONOw1-impl*: $nCNTR\ \varphi \longrightarrow MONOw^1\ \varphi$ ⟨*proof*⟩
lemma *EXPN-MONOw2-impl*: $EXPN\ \varphi \longrightarrow MONOw^2\ \varphi$ ⟨*proof*⟩
lemma *nEXPN-MONOw2-impl*: $nEXPN\ \varphi \longrightarrow MONOw^2\ \varphi$ ⟨*proof*⟩

Relativized nADDI variants.

definition $nADDIr::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nADDIr*›)
  where $nADDIr\ \varphi \equiv \forall A\ B.\ let\ U = (A \vee B)\ in \quad (\varphi(A \vee B) =^U (\varphi\ A) \wedge (\varphi\ B))$
definition $nADDIr\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$nADDIr^a$›)
  where $nADDIr^a\ \varphi \equiv \forall A\ B.\ let\ U = (A \vee B)\ in\ ((\varphi\ A) \wedge (\varphi\ B) \leq^U \varphi(A \vee B))$
definition $nADDIr\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$nADDIr^b$›)
  where $nADDIr^b\ \varphi \equiv \forall A\ B.\ let\ U = (A \vee B)\ in \quad (\varphi(A \vee B) \leq^U (\varphi\ A) \wedge (\varphi\ B))$

declare *nADDIr-def*[*cond*] *nADDIr-a-def*[*cond*] *nADDIr-b-def*[*cond*]

lemma *nADDIr-char*: $nADDIr\ \varphi = (nADDIr^a\ \varphi \wedge nADDIr^b\ \varphi)$ ⟨*proof*⟩

lemma *nADDIr-a-impl*: $nADDI^a\ \varphi \longrightarrow nADDIr^a\ \varphi$ ⟨*proof*⟩
lemma $nADDIr^a\ \varphi \longrightarrow nADDI^a\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *nADDIr-a-equ*: $EXPN\ \varphi \Longrightarrow nADDIr^a\ \varphi = nADDI^a\ \varphi$ ⟨*proof*⟩
lemma *nADDIr-a-equ'*:$nEXPN\ \varphi \Longrightarrow nADDIr^a\ \varphi = nADDI^a\ \varphi$ ⟨*proof*⟩

lemma *nADDIr-b-impl*: $nADDI^b\ \varphi \longrightarrow nADDIr^b\ \varphi$ ⟨*proof*⟩
lemma $EXPN\ \varphi \Longrightarrow nADDIr^b\ \varphi \longrightarrow nADDI^b\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *nADDIr-b-equ*: $nEXPN\ \varphi \Longrightarrow nADDIr^b\ \varphi = nADDI^b\ \varphi$ ⟨*proof*⟩

Relativized nMULT variants.

definition $nMULTr::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹*nMULTr*›)
  where $nMULTr\ \varphi \equiv \forall A\ B.\ let\ U = (A \wedge B)\ in \quad (\varphi(A \wedge B) =_U (\varphi\ A) \vee (\varphi\ B))$
definition $nMULTr\text{-}a::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$nMULTr^a$›)
  where $nMULTr^a\ \varphi \equiv \forall A\ B.\ let\ U = (A \wedge B)\ in\ ((\varphi\ A) \vee (\varphi\ B) \leq_U \varphi(A \wedge B))$
definition $nMULTr\text{-}b::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$nMULTr^b$›)
  where $nMULTr^b\ \varphi \equiv \forall A\ B.\ let\ U = (A \wedge B)\ in \quad (\varphi(A \wedge B) \leq_U (\varphi\ A) \vee (\varphi\ B))$

declare *nMULTr-def*[*cond*] *nMULTr-a-def*[*cond*] *nMULTr-b-def*[*cond*]

lemma *nMULTr-char*: $nMULTr\ \varphi = (nMULTr^a\ \varphi \wedge nMULTr^b\ \varphi)$ ⟨*proof*⟩

lemma *nMULTr-a-impl*: $nMULT^a\ \varphi \longrightarrow nMULTr^a\ \varphi$ ⟨*proof*⟩
lemma $CNTR\ \varphi \Longrightarrow nMULTr^a\ \varphi \longrightarrow nMULT^a\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *nMULTr-a-equ*: $nCNTR\ \varphi \Longrightarrow nMULTr^a\ \varphi = nMULT^a\ \varphi$ ⟨*proof*⟩

lemma *nMULTr-b-impl*: $nMULT^b\ \varphi \longrightarrow nMULTr^b\ \varphi$ ⟨*proof*⟩
lemma $nMULTr^b\ \varphi \longrightarrow nMULT^b\ \varphi$ **nitpick** ⟨*proof*⟩
lemma *nMULTr-b-equ*: $CNTR\ \varphi \Longrightarrow nMULTr^b\ \varphi = nMULT^b\ \varphi$ ⟨*proof*⟩
lemma *nMULTr-b-equ'*:$nCNTR\ \varphi \Longrightarrow nMULTr^b\ \varphi = nMULT^b\ \varphi$ ⟨*proof*⟩

Weak variants of antitonicity.

definition $ANTIw1::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$ANTIw^1$›)
  where $ANTIw^1\ \varphi \equiv \forall A\ B.\ A \leq B \longrightarrow (\varphi\ B) \leq B \vee (\varphi\ A)$
definition $ANTIw2::('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$ (‹$ANTIw^2$›)

where $ANTIw^2\ \varphi \equiv \forall\ A\ B.\ A \leq B \longrightarrow A \wedge (\varphi\ B) \leq (\varphi\ A)$

declare *ANTIw1-def*[*cond*] *ANTIw2-def*[*cond*]

lemma *ANTIw1-nADDIr-b*: $ANTIw^1\ \varphi = nADDIr^b\ \varphi\ \langle proof \rangle$
lemma *ANTIw2-nMULTr-a*: $ANTIw^2\ \varphi = nMULTr^a\ \varphi\ \langle proof \rangle$

lemma $ANTI\ \varphi \longrightarrow ANTIw^1\ \varphi\ \langle proof \rangle$
lemma $ANTIw^1\ \varphi \longrightarrow ANTI\ \varphi$ **nitpick** $\langle proof \rangle$
lemma $ANTI\ \varphi \longrightarrow ANTIw^2\ \varphi\ \langle proof \rangle$
lemma $ANTIw^2\ \varphi \longrightarrow ANTI\ \varphi$ **nitpick** $\langle proof \rangle$

We have in fact that (n)CNTR (resp. (n)EXPN) implies ANTIw-1/nADDIr-b (resp. ANTIw-2/nMULTr-a).

lemma *CNTR-ANTIw1-impl*: $CNTR\ \varphi \longrightarrow ANTIw^1\ \varphi\ \langle proof \rangle$
lemma *nCNTR-ANTIw1-impl*: $nCNTR\ \varphi \longrightarrow ANTIw^1\ \varphi\ \langle proof \rangle$
lemma *EXPN-ANTIw2-impl*: $EXPN\ \varphi \longrightarrow ANTIw^2\ \varphi\ \langle proof \rangle$
lemma *nEXPN-ANTIw2-impl*: $nEXPN\ \varphi \longrightarrow ANTIw^2\ \varphi\ \langle proof \rangle$

Dual interrelations.

lemma *ADDIr-dual1*: $ADDIr^a\ \varphi = MULTr^b\ \varphi^d\ \langle proof \rangle$
lemma *ADDIr-dual2*: $ADDIr^b\ \varphi = MULTr^a\ \varphi^d\ \langle proof \rangle$
lemma *ADDIr-dual*: $ADDIr\ \varphi = MULTr\ \varphi^d\ \langle proof \rangle$

lemma *nADDIr-dual1*: $nADDIr^a\ \varphi = nMULTr^b\ \varphi^d\ \langle proof \rangle$
lemma *nADDIr-dual2*: $nADDIr^b\ \varphi = nMULTr^a\ \varphi^d\ \langle proof \rangle$
lemma *nADDIr-dual*: $nADDIr\ \varphi = nMULTr\ \varphi^d\ \langle proof \rangle$

Complement interrelations.

lemma *ADDIr-a-cmpl*: $ADDIr^a\ \varphi = nADDIr^a\ \varphi^-\ \langle proof \rangle$
lemma *ADDIr-b-cmpl*: $ADDIr^b\ \varphi = nADDIr^b\ \varphi^-\ \langle proof \rangle$
lemma *ADDIr-cmpl*: $ADDIr\ \varphi = nADDIr\ \varphi^-\ \langle proof \rangle$
lemma *MULTr-a-cmpl*: $MULTr^a\ \varphi = nMULTr^a\ \varphi^-\ \langle proof \rangle$
lemma *MULTr-b-cmpl*: $MULTr^b\ \varphi = nMULTr^b\ \varphi^-\ \langle proof \rangle$
lemma *MULTr-cmpl*: $MULTr\ \varphi = nMULTr\ \varphi^-\ \langle proof \rangle$

Fixed-point interrelations.

lemma *EXPN-fp*: $EXPN\ \varphi = EXPN\ \varphi^{fp}\ \langle proof \rangle$
lemma *EXPN-fpc*: $EXPN\ \varphi = nEXPN\ \varphi^{fp-}\ \langle proof \rangle$
lemma *CNTR-fp*: $CNTR\ \varphi = nCNTR\ \varphi^{fp}\ \langle proof \rangle$
lemma *CNTR-fpc*: $CNTR\ \varphi = CNTR\ \varphi^{fp-}\ \langle proof \rangle$

lemma *nNORM-fp*: $NORM\ \varphi = nNORM\ \varphi^{fp}\ \langle proof \rangle$
lemma *NORM-fpc*: $NORM\ \varphi = NORM\ \varphi^{fp-}\ \langle proof \rangle$
lemma *DNRM-fp*: $DNRM\ \varphi = DNRM\ \varphi^{fp}\ \langle proof \rangle$
lemma *DNRM-fpc*: $DNRM\ \varphi = nDNRM\ \varphi^{fp-}\ \langle proof \rangle$

lemma *ADDIr-a-fpc*: $ADDIr^a\ \varphi = ADDIr^a\ \varphi^{fp-}\ \langle proof \rangle$
lemma *ADDIr-a-fp*: $ADDIr^a\ \varphi = nADDIr^a\ \varphi^{fp}\ \langle proof \rangle$
lemma *ADDIr-b-fpc*: $ADDIr^b\ \varphi = ADDIr^b\ \varphi^{fp-}\ \langle proof \rangle$
lemma *ADDIr-b-fp*: $ADDIr^b\ \varphi = nADDIr^b\ \varphi^{fp}\ \langle proof \rangle$

lemma *MULTr-a-fp*: $MULTr^a\ \varphi = MULTr^a\ \varphi^{fp}\ \langle proof \rangle$
lemma *MULTr-a-fpc*: $MULTr^a\ \varphi = nMULTr^a\ \varphi^{fp-}\ \langle proof \rangle$
lemma *MULTr-b-fp*: $MULTr^b\ \varphi = MULTr^b\ \varphi^{fp}\ \langle proof \rangle$

lemma *MULTr-b-fpc*: $MULTr^b\ \varphi = nMULTr^b\ \varphi^{fp-}\ \langle proof\rangle$

Relativized IDEM variants.

definition *IDEMr-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle IDEMr^a\rangle)$
  where $IDEMr^a\ \varphi \equiv \forall\, A.\ \varphi(A \vee \varphi\ A) \leq^A (\varphi\ A)$
definition *IDEMr-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle IDEMr^b\rangle)$
  where $IDEMr^b\ \varphi \equiv \forall\, A.\ (\varphi\ A) \leq_A \varphi(A \wedge \varphi\ A)$
declare *IDEMr-a-def*[*cond*] *IDEMr-b-def*[*cond*]

Relativized nIDEM variants.

definition *nIDEMr-a*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle nIDEMr^a\rangle)$
  where $nIDEMr^a\ \varphi \equiv \forall\, A.\ (\varphi\ A) \leq^A \varphi(A \vee -(\varphi\ A))$
definition *nIDEMr-b*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle nIDEMr^b\rangle)$
  where $nIDEMr^b\ \varphi \equiv \forall\, A.\ \varphi(A \wedge -(\varphi\ A)) \leq_A (\varphi\ A)$

declare *nIDEMr-a-def*[*cond*] *nIDEMr-b-def*[*cond*]

Complement interrelations.

lemma *IDEMr-a-cmpl*: $IDEMr^a\ \varphi = nIDEMr^a\ \varphi^-\ \langle proof\rangle$
lemma *IDEMr-b-cmpl*: $IDEMr^b\ \varphi = nIDEMr^b\ \varphi^-\ \langle proof\rangle$

Dual interrelation.

lemma *IDEMr-dual*: $IDEMr^a\ \varphi = IDEMr^b\ \varphi^d\ \langle proof\rangle$
lemma *nIDEMr-dual*: $nIDEMr^a\ \varphi = nIDEMr^b\ \varphi^d\ \langle proof\rangle$

Fixed-point interrelations.

lemma *nIDEMr-a-fpc*: $nIDEMr^a\ \varphi = nIDEMr^a\ \varphi^{fp-}\ \langle proof\rangle$
lemma *IDEMr-a-fp*: $IDEMr^a\ \varphi = nIDEMr^a\ \varphi^{fp}\ \langle proof\rangle$
lemma *IDEMr-a-fpc*: $IDEMr^a\ \varphi = IDEMr^a\ \varphi^{fp-}\ \langle proof\rangle$
lemma *IDEMr-b-fp*: $IDEMr^b\ \varphi = IDEMr^b\ \varphi^{fp}\ \langle proof\rangle$
lemma *IDEMr-b-fpc*: $IDEMr^b\ \varphi = nIDEMr^b\ \varphi^{fp-}\ \langle proof\rangle$

The original border condition B1' (by Zarycki) is equivalent to the conjuntion of nMULTr and CNTR.

abbreviation $B1'\ \varphi \equiv \forall\, A\ B.\ \varphi(A \wedge B) = (A \wedge \varphi\ B) \vee (\varphi\ A \wedge B)$

lemma $B1'\ \varphi = (nMULTr\ \varphi \wedge CNTR\ \varphi)\ \langle proof\rangle$

Modulo conditions nMULTr and CNTR the border condition B4 is equivalent to nIDEMr-b.

abbreviation $B4\ \varphi \equiv \forall\, A.\ \varphi(-\varphi(-A)) \leq A$

lemma $nMULTr\ \varphi \Longrightarrow CNTR\ \varphi \Longrightarrow B4\ \varphi = nIDEMr^b\ \varphi\ \langle proof\rangle$

end
theory *conditions-relativized-infinitary*
  imports *conditions-relativized conditions-negative-infinitary*
begin

3.4 Infinitary Relativized Conditions

We define and interrelate infinitary variants for some previously introduced axiomatic conditions on operators.

definition *iADDIr*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iADDIr\rangle)$

where $iADDIr\ \varphi\ \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\varphi(\bigvee S) =^U \bigvee[\![\varphi\ S]\!])$
definition $iADDIr\text{-}a{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iADDIr^a\rangle)$
where $iADDIr^a\ \varphi \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\varphi(\bigvee S) \leq^U \bigvee[\![\varphi\ S]\!])$
definition $iADDIr\text{-}b{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iADDIr^b\rangle)$
where $iADDIr^b\ \varphi \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\bigvee[\![\varphi\ S]\!] \leq^U \varphi(\bigvee S))$

definition $inADDIr{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDIr\rangle)$
where $inADDIr\ \varphi\ \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\varphi(\bigvee S) =^U \bigwedge[\![\varphi\ S]\!])$
definition $inADDIr\text{-}a{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDIr^a\rangle)$
where $inADDIr^a\ \varphi \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\bigwedge[\![\varphi\ S]\!] \leq^U \varphi(\bigvee S))$
definition $inADDIr\text{-}b{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inADDIr^b\rangle)$
where $inADDIr^b\ \varphi \equiv \forall\, S.\ let\ U{=}\bigvee S\ in\ (\varphi(\bigvee S) \leq^U \bigwedge[\![\varphi\ S]\!])$

declare *iADDIr-def*[*cond*] *iADDIr-a-def*[*cond*] *iADDIr-b-def*[*cond*]
inADDIr-def[*cond*] *inADDIr-a-def*[*cond*] *inADDIr-b-def*[*cond*]

definition $iMULTr{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iMULTr\rangle)$
where $iMULTr\ \varphi\ \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\varphi(\bigwedge S) =_U \bigwedge[\![\varphi\ S]\!])$
definition $iMULTr\text{-}a{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iMULTr^a\rangle)$
where $iMULTr^a\ \varphi \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\varphi(\bigwedge S) \leq_U \bigwedge[\![\varphi\ S]\!])$
definition $iMULTr\text{-}b{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle iMULTr^b\rangle)$
where $iMULTr^b\ \varphi \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\bigwedge[\![\varphi\ S]\!] \leq_U \varphi(\bigwedge S))$

definition $inMULTr{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULTr\rangle)$
where $inMULTr\ \varphi\ \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\varphi(\bigwedge S) =_U \bigvee[\![\varphi\ S]\!])$
definition $inMULTr\text{-}a{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULTr^a\rangle)$
where $inMULTr^a\ \varphi \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\bigvee[\![\varphi\ S]\!] \leq_U \varphi(\bigwedge S))$
definition $inMULTr\text{-}b{::}('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool\ (\langle inMULTr^b\rangle)$
where $inMULTr^b\ \varphi \equiv \forall\, S.\ let\ U{=}\bigwedge S\ in\ (\varphi(\bigwedge S) \leq_U \bigvee[\![\varphi\ S]\!])$

declare *iMULTr-def*[*cond*] *iMULTr-a-def*[*cond*] *iMULTr-b-def*[*cond*]
inMULTr-def[*cond*] *inMULTr-a-def*[*cond*] *inMULTr-b-def*[*cond*]

lemma *iADDIr-char*: $iADDIr\ \varphi = (iADDIr^a\ \varphi \wedge iADDIr^b\ \varphi)$ ⟨*proof*⟩
lemma *iMULTr-char*: $iMULTr\ \varphi = (iMULTr^a\ \varphi \wedge iMULTr^b\ \varphi)$ ⟨*proof*⟩

lemma *inADDIr-char*: $inADDIr\ \varphi = (inADDIr^a\ \varphi \wedge inADDIr^b\ \varphi)$ ⟨*proof*⟩
lemma *inMULTr-char*: $inMULTr\ \varphi = (inMULTr^a\ \varphi \wedge inMULTr^b\ \varphi)$ ⟨*proof*⟩

Dual interrelations.

lemma *iADDIr-dual1*: $iADDIr^a\ \varphi = iMULTr^b\ \varphi^d$ ⟨*proof*⟩
lemma *iADDIr-dual2*: $iADDIr^b\ \varphi = iMULTr^a\ \varphi^d$ ⟨*proof*⟩
lemma *iADDIr-dual*: $iADDIr\ \varphi = iMULTr\ \varphi^d$ ⟨*proof*⟩

lemma *inADDIr-dual1*: $inADDIr^a\ \varphi = inMULTr^b\ \varphi^d$ ⟨*proof*⟩
lemma *inADDIr-dual2*: $inADDIr^b\ \varphi = inMULTr^a\ \varphi^d$ ⟨*proof*⟩
lemma *inADDIr-dual*: $inADDIr\ \varphi = inMULTr\ \varphi^d$ ⟨*proof*⟩

Complement interrelations.

lemma *iADDIr-a-cmpl*: $iADDIr^a\ \varphi = inADDIr^a\ \varphi^-$ ⟨*proof*⟩
lemma *iADDIr-b-cmpl*: $iADDIr^b\ \varphi = inADDIr^b\ \varphi^-$ ⟨*proof*⟩
lemma *iADDIr-cmpl*: $iADDIr\ \varphi = inADDIr\ \varphi^-$ ⟨*proof*⟩

lemma *iMULTr-a-cmpl*: $iMULTr^a\ \varphi = inMULTr^a\ \varphi^-$ ⟨*proof*⟩
lemma *iMULTr-b-cmpl*: $iMULTr^b\ \varphi = inMULTr^b\ \varphi^-$ ⟨*proof*⟩

lemma *iMULTr-cmpl*: $MULTr\ \varphi = nMULTr\ \varphi^{-}$ ⟨*proof*⟩

Fixed-point interrelations.

lemma *iADDIr-a-fpc*: $iADDIr^{a}\ \varphi = iADDIr^{a}\ \varphi^{fp-}$ ⟨*proof*⟩
lemma *iADDIr-a-fp*: $iADDIr^{a}\ \varphi = inADDIr^{a}\ \varphi^{fp}$ ⟨*proof*⟩
lemma *iADDIr-b-fpc*: $iADDIr^{b}\ \varphi = iADDIr^{b}\ \varphi^{fp-}$ ⟨*proof*⟩
lemma *iADDIr-b-fp*: $iADDIr^{b}\ \varphi = inADDIr^{b}\ \varphi^{fp}$ ⟨*proof*⟩

lemma *iMULTr-a-fp*: $iMULTr^{a}\ \varphi = iMULTr^{a}\ \varphi^{fp}$ ⟨*proof*⟩
lemma *iMULTr-a-fpc*: $iMULTr^{a}\ \varphi = inMULTr^{a}\ \varphi^{fp-}$ ⟨*proof*⟩
lemma *iMULTr-b-fp*: $iMULTr^{b}\ \varphi = iMULTr^{b}\ \varphi^{fp}$ ⟨*proof*⟩
lemma *iMULTr-b-fpc*: $iMULTr^{b}\ \varphi = inMULTr^{b}\ \varphi^{fp-}$ ⟨*proof*⟩

end
theory *logics-consequence*
imports *boolean-algebra*
begin

4 Logics based on Topological Boolean Algebras

4.1 Logical Consequence and Validity

Logical validity can be defined as truth in all points (i.e. as denoting the top element).

abbreviation (*input*) *gtrue*::'*w* $\sigma \Rightarrow$ *bool* (‹[⊢ -]›)
where $[\vdash A] \equiv \forall w.\ A\ w$
lemma *gtrue-def*: $[\vdash A] \equiv A = \top$ ⟨*proof*⟩

When defining a logic over an existing algebra we have two choices: a global (truth preserving) and a local (degree preserving) notion of logical consequence. For LFIs/LFUs we prefer the latter.

abbreviation (*input*) *conseq-global1*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[- $\vdash_g$ -]›)
where $[A \vdash_g B] \equiv [\vdash A] \longrightarrow [\vdash B]$
abbreviation (*input*) *conseq-global21*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[-,- $\vdash_g$ -]›)
where $[A_1,\ A_2 \vdash_g B] \equiv [\vdash A_1] \wedge [\vdash A_2] \longrightarrow [\vdash B]$
abbreviation (*input*) *conseq-global31*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[-,-,- $\vdash_g$ -]›)
where $[A_1,\ A_2,\ A_3 \vdash_g B] \equiv [\vdash A_1] \wedge [\vdash A_2] \wedge [\vdash A_3] \longrightarrow [\vdash B]$

abbreviation (*input*) *conseq-local1*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[- ⊢ -]›)
where $[A \vdash B] \equiv A \leq B$
abbreviation (*input*) *conseq-local21*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[-,- ⊢ -]›)
where $[A_1,\ A_2 \vdash B] \equiv A_1 \wedge A_2 \leq B$
abbreviation (*input*) *conseq-local12*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[- ⊢ -,-]›)
where $[A \vdash B_1,\ B_2] \equiv A \leq B_1 \vee B_2$
abbreviation (*input*) *conseq-local31*::'*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ '*w* $\sigma \Rightarrow$ *bool* (‹[-,-,- ⊢ -]›)
where $[A_1,\ A_2,\ A_3 \vdash B] \equiv A_1 \wedge A_2 \wedge A_3 \leq B$

end
theory *logics-operators*
imports *conditions-positive*
begin

4.2 Converting between topological operators

We verify minimal conditions under which operators resulting from conversion functions coincide.

Conversions between interior, closure and exterior are straightforward and hold without restrictions: Interior and closure are each other duals. Exterior is the complement of closure. We focus here on conversions involving the border and frontier operators.

Interior operator as derived from border.

definition *Int-br*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{I}_B \rangle)$
where $\mathcal{I}_B\ \mathcal{B} \equiv \lambda A.\ A \leftharpoonup (\mathcal{B}\ A)$

Interior operator as derived from frontier.

definition *Int-fr*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{I}_F \rangle)$
where $\mathcal{I}_F\ \mathcal{F} \equiv \lambda A.\ A \leftharpoonup (\mathcal{F}\ A)$

Closure operator as derived from border.

definition *Cl-br*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{C}_B \rangle)$
where $\mathcal{C}_B\ \mathcal{B} \equiv \lambda A.\ A \vee \mathcal{B}(-A)$

Closure operator as derived from frontier.

definition *Cl-fr*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{C}_F \rangle)$
where $\mathcal{C}_F\ \mathcal{F} \equiv \lambda A.\ A \vee (\mathcal{F}\ A)$

Frontier operator as derived from interior.

definition *Fr-int*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{F}_I \rangle)$
where $\mathcal{F}_I\ \mathcal{I} \equiv \lambda A.\ -(\mathcal{I}\ A \vee \mathcal{I}(-A))$

Frontier operator as derived from closure.

definition *Fr-cl*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{F}_C \rangle)$
where $\mathcal{F}_C\ \mathcal{C} \equiv \lambda A.\ (\mathcal{C}\ A) \wedge \mathcal{C}(-A)$

Frontier operator as derived from border.

definition *Fr-br*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{F}_B \rangle)$
where $\mathcal{F}_B\ \mathcal{B} \equiv \lambda A.\ \mathcal{B}\ A \vee \mathcal{B}(-A)$

Border operator as derived from interior.

definition *Br-int*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{B}_I \rangle)$
where $\mathcal{B}_I\ \mathcal{I} \equiv \lambda A.\ A \leftharpoonup (\mathcal{I}\ A)$

Border operator as derived from closure.

definition *Br-cl*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{B}_C \rangle)$
where $\mathcal{B}_C\ \mathcal{C} \equiv \lambda A.\ A \wedge \mathcal{C}(-A)$

Border operator as derived from frontier.

definition *Br-fr*::$('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow ('w\ \sigma \Rightarrow 'w\ \sigma)$ $(\langle \mathcal{B}_F \rangle)$
where $\mathcal{B}_F\ \mathcal{F} \equiv \lambda A.\ A \wedge (\mathcal{F}\ A)$

Inter-definitions involving border or frontier do not hold without restrictions.

lemma $\mathcal{B} = \mathcal{B}_C\ (\mathcal{C}_B\ \mathcal{B})$ **nitpick** $\langle proof \rangle$
lemma $\mathcal{B} = \mathcal{B}_I\ (\mathcal{I}_B\ \mathcal{B})$ **nitpick** $\langle proof \rangle$

lemma $\mathcal{B} = \mathcal{B}_F\ (\mathcal{F}_B\ \mathcal{B})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{F} = \mathcal{F}_C\ (\mathcal{C}_F\ \mathcal{F})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{F} = \mathcal{F}_I\ (\mathcal{I}_F\ \mathcal{F})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{F} = \mathcal{F}_B\ (\mathcal{B}_F\ \mathcal{F})$ **nitpick** ⟨*proof*⟩

lemma $\mathcal{C} = \mathcal{C}_B\ (\mathcal{B}_C\ \mathcal{C})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{C} = \mathcal{C}_F\ (\mathcal{F}_C\ \mathcal{C})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{I} = \mathcal{I}_B\ (\mathcal{B}_C\ \mathcal{I})$ **nitpick** ⟨*proof*⟩
lemma $\mathcal{I} = \mathcal{I}_F\ (\mathcal{F}_C\ \mathcal{I})$ **nitpick** ⟨*proof*⟩

Inter-definitions involving border or frontier always assume the second Kuratowski condition (or its respective counterpart: C2, I2, B2 or F2).

abbreviation $C2\ \varphi \equiv EXPN\ \varphi$
abbreviation $I2\ \varphi \equiv CNTR\ \varphi$
abbreviation $B2\ \varphi \equiv CNTR\ \varphi$
abbreviation $F2\ \varphi \equiv \forall\, A.\ \varphi(-A) = \varphi\ A$

lemma $B2\ \mathcal{B} \Longrightarrow \mathcal{B} = \mathcal{B}_C\ (\mathcal{C}_B\ \mathcal{B})$ ⟨*proof*⟩
lemma $B2\ \mathcal{B} \Longrightarrow \mathcal{B} = \mathcal{B}_I\ (\mathcal{I}_B\ \mathcal{B})$ ⟨*proof*⟩
lemma $B2\ \mathcal{B} \Longrightarrow \mathcal{B} = \mathcal{B}_F\ (\mathcal{F}_B\ \mathcal{B})$ ⟨*proof*⟩
lemma $F2\ \mathcal{F} \Longrightarrow \mathcal{F} = \mathcal{F}_C\ (\mathcal{C}_F\ \mathcal{F})$ ⟨*proof*⟩
lemma $F2\ \mathcal{F} \Longrightarrow \mathcal{F} = \mathcal{F}_I\ (\mathcal{I}_F\ \mathcal{F})$ ⟨*proof*⟩
lemma $F2\ \mathcal{F} \Longrightarrow \mathcal{F} = \mathcal{F}_B\ (\mathcal{B}_F\ \mathcal{F})$ ⟨*proof*⟩

lemma $C2\ \mathcal{C} \Longrightarrow \mathcal{C} = \mathcal{C}_B\ (\mathcal{B}_C\ \mathcal{C})$ ⟨*proof*⟩
lemma $C2\ \mathcal{C} \Longrightarrow \mathcal{C} = \mathcal{C}_F\ (\mathcal{F}_C\ \mathcal{C})$ ⟨*proof*⟩
lemma $I2\ \mathcal{I} \Longrightarrow \mathcal{I} = \mathcal{I}_B\ (\mathcal{B}_I\ \mathcal{I})$ ⟨*proof*⟩
lemma $I2\ \mathcal{I} \Longrightarrow \mathcal{I} = \mathcal{I}_F\ (\mathcal{F}_I\ \mathcal{I})$ ⟨*proof*⟩

end
theory *logics-negation*
imports *logics-consequence conditions-relativized*
begin

4.3 Properties of negation(-like) operators

To avoid visual clutter we introduce convenient notation for type for properties of operators.

type-synonym $'w\ \Omega = ('w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow bool$

named-theorems *neg*

4.3.1 Principles of excluded middle, contradiction and explosion

TND: tertium non datur, aka. law of excluded middle (resp. strong, weak, minimal).

abbreviation $pTND\ \ (\langle TND^{\text{-}}\ \text{-}\rangle)$ **where** $TND^a\ \ \eta \equiv \quad [\vdash\ a \vee \eta\ a]$
abbreviation $pTNDw\ (\langle TNDw^{\text{-}}\ \text{-}\rangle)$ **where** $TNDw^a\ \eta \equiv \forall\, b.\ [\eta\ b \vdash a,\ \eta\ a]$
abbreviation $pTNDm\ (\langle TNDm^{\text{-}}\ \text{-}\rangle)$ **where** $TNDm^a\ \eta \equiv \quad [\eta\ \bot \vdash a,\ \eta\ a]$
definition $TND :: 'w\ \Omega$ **where** $TND\ \ \eta \equiv \forall\, \varphi.\ TND^\varphi\ \ \eta$
definition $TNDw :: 'w\ \Omega$ **where** $TNDw\ \eta \equiv \forall\, \varphi.\ TNDw^\varphi\ \eta$
definition $TNDm :: 'w\ \Omega$ **where** $TNDm\ \eta \equiv \forall\, \varphi.\ TNDm^\varphi\ \eta$
declare *TND-def*[*neg*] *TNDw-def*[*neg*] *TNDm-def*[*neg*]

Explore some (non)entailment relations.

lemma $TND\ \ \eta \Longrightarrow TNDw\ \eta$ ⟨*proof*⟩

lemma $TNDw\ \eta \Longrightarrow TND\ \ \eta$ **nitpick** ⟨*proof*⟩
lemma $TNDw\ \eta \Longrightarrow TNDm\ \eta$ ⟨*proof*⟩
lemma $TNDm\ \eta \Longrightarrow TNDw\ \eta$ **nitpick** ⟨*proof*⟩

ECQ: ex contradictione (sequitur) quodlibet (variants: strong, weak, minimal).

abbreviation $pECQ$ ($\langle ECQ^{-}\ \text{-}\rangle$) **where** $ECQ^a\ \ \eta \equiv\ \ \ [a, \eta\ a \vdash \bot]$
abbreviation $pECQw$ ($\langle ECQw^{-}\ \text{-}\rangle$) **where** $ECQw^a\ \eta \equiv \forall\, b.\ [a, \eta\ a \vdash \eta\ b]$
abbreviation $pECQm$ ($\langle ECQm^{-}\ \text{-}\rangle$) **where** $ECQm^a\ \eta \equiv\ \ \ [a, \eta\ a \vdash \eta\ \top]$
definition $ECQ ::'w\ \Omega$ **where** $\ ECQ\ \eta \equiv \forall\, a.\ ECQ^a\ \ \eta$
definition $ECQw::'w\ \Omega$ **where** $ECQw\ \eta \equiv \forall\, a.\ ECQw^a\ \eta$
definition $ECQm::'w\ \Omega$ **where** $ECQm\ \eta \equiv \forall\, a.\ ECQm^a\ \eta$
declare *ECQ-def*[*neg*] *ECQw-def*[*neg*] *ECQm-def*[*neg*]

Explore some (non)entailment relations.

lemma $ECQ\ \ \eta \Longrightarrow ECQw\ \eta$ ⟨*proof*⟩
lemma $ECQw\ \eta \Longrightarrow ECQ\ \ \eta$ **nitpick** ⟨*proof*⟩
lemma $ECQw\ \eta \Longrightarrow ECQm\ \eta$ ⟨*proof*⟩
lemma $ECQm\ \eta \Longrightarrow ECQw\ \eta$ **nitpick** ⟨*proof*⟩

LNC: law of non-contradiction.

abbreviation $pLNC$ ($\langle LNC^{-}\ \text{-}\rangle$) **where** $LNC^a\ \eta \equiv [\vdash \eta(a \wedge \eta\ a)]$
definition $LNC::'w\ \Omega$ **where** $LNC\ \eta \equiv \forall\, a.\ LNC^a\ \eta$
declare *LNC-def*[*neg*]

ECQ and LNC are in general independent.

lemma $ECQ\ \eta \Longrightarrow LNC\ \eta$ **nitpick** ⟨*proof*⟩
lemma $LNC\ \eta \Longrightarrow ECQm\ \eta$ **nitpick** ⟨*proof*⟩

4.3.2 Contraposition rules

CoP: contraposition (weak 'rule-like' variants). Variant 0 is antitonicity (ANTI). Variants 1-3 are stronger.

abbreviation $pCoP1$ ($\langle CoP1^{--}\ \text{-}\rangle$) **where** $CoP1^{ab}\ \eta \equiv [a \vdash \eta\ b] \longrightarrow [b \vdash \eta\ a]$
abbreviation $pCoP2$ ($\langle CoP2^{--}\ \text{-}\rangle$) **where** $CoP2^{ab}\ \eta \equiv [\eta\ a \vdash b] \longrightarrow [\eta\ b \vdash a]$
abbreviation $pCoP3$ ($\langle CoP3^{--}\ \text{-}\rangle$) **where** $CoP3^{ab}\ \eta \equiv [\eta\ a \vdash \eta\ b] \longrightarrow [b \vdash a]$

abbreviation $CoP0 ::'w\ \Omega$ **where** $CoP0\ \ \eta \equiv ANTI\ \eta$
definition $\ \ CoP1 ::'w\ \Omega$ **where** $CoP1\ \ \eta \equiv \forall\, a\ b.\ CoP1^{ab}\ \eta$
definition $\ \ CoP2 ::'w\ \Omega$ **where** $CoP2\ \ \eta \equiv \forall\, a\ b.\ CoP2^{ab}\ \eta$
definition $\ \ CoP3 ::'w\ \Omega$ **where** $CoP3\ \ \eta \equiv \forall\, a\ b.\ CoP3^{ab}\ \eta$

declare *CoP1-def*[*neg*] *CoP2-def*[*neg*] *CoP3-def*[*neg*]

Explore some (non)entailment relations.

lemma $CoP1\ \eta \Longrightarrow CoP0\ \eta$ ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow CoP1\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP2\ \eta \Longrightarrow CoP0\ \eta$ ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow CoP2\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP3\ \eta \Longrightarrow CoP0\ \eta$ ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow CoP3\ \eta$ **nitpick** ⟨*proof*⟩

All three strong variants are pairwise independent. However, CoP3 follows from CoP1 plus CoP2.

lemma *CoP123*: $CoP1\ \eta \Longrightarrow CoP2\ \eta \Longrightarrow CoP3\ \eta$ ⟨*proof*⟩

Taking all CoP together still leaves room for a boldly paraconsistent resp. paracomplete logic.

lemma $CoP1\ \eta \Longrightarrow CoP2\ \eta \Longrightarrow ECQm\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP1\ \eta \Longrightarrow CoP2\ \eta \Longrightarrow TNDm\ \eta$ **nitpick** ⟨*proof*⟩

4.3.3 Modus tollens rules

MT: modus (tollendo) tollens (weak 'rule-like' variants).

abbreviation *pMT0* (‹$MT0^{--}$ -›) **where** $MT0^{ab}\ \eta \equiv [a \vdash b] \wedge [\vdash \eta\ b] \longrightarrow [\vdash \eta\ a]$
abbreviation *pMT1* (‹$MT1^{--}$ -›) **where** $MT1^{ab}\ \eta \equiv [a \vdash \eta\ b] \wedge [\vdash b] \longrightarrow [\vdash \eta\ a]$
abbreviation *pMT2* (‹$MT2^{--}$ -›) **where** $MT2^{ab}\ \eta \equiv [\eta\ a \vdash b] \wedge [\vdash \eta\ b] \longrightarrow [\vdash a]$
abbreviation *pMT3* (‹$MT3^{--}$ -›) **where** $MT3^{ab}\ \eta \equiv [\eta\ a \vdash \eta\ b] \wedge [\vdash b] \longrightarrow [\vdash a]$
definition $MT0::'w\ \Omega$ **where** $MT0\ \eta \equiv \forall\, a\ b.\ MT0^{ab}\ \eta$
definition $MT1::'w\ \Omega$ **where** $MT1\ \eta \equiv \forall\, a\ b.\ MT1^{ab}\ \eta$
definition $MT2::'w\ \Omega$ **where** $MT2\ \eta \equiv \forall\, a\ b.\ MT2^{ab}\ \eta$
definition $MT3::'w\ \Omega$ **where** $MT3\ \eta \equiv \forall\, a\ b.\ MT3^{ab}\ \eta$

declare *MT0-def*[*neg*] *MT1-def*[*neg*] *MT2-def*[*neg*] *MT3-def*[*neg*]

Again, all MT variants are pairwise independent. We explore some (non)entailment relations.

lemma $CoP0\ \eta \Longrightarrow MT0\ \eta$ ⟨*proof*⟩
lemma $CoP1\ \eta \Longrightarrow MT1\ \eta$ ⟨*proof*⟩
lemma $CoP2\ \eta \Longrightarrow MT2\ \eta$ ⟨*proof*⟩
lemma $CoP3\ \eta \Longrightarrow MT3\ \eta$ ⟨*proof*⟩
lemma $MT0\ \eta \Longrightarrow MT1\ \eta \Longrightarrow MT2\ \eta \Longrightarrow MT3\ \eta \Longrightarrow CoP0\ \eta$ **nitpick** ⟨*proof*⟩
lemma $MT0\ \eta \Longrightarrow MT1\ \eta \Longrightarrow MT2\ \eta \Longrightarrow MT3\ \eta \Longrightarrow ECQm\ \eta$ **nitpick** ⟨*proof*⟩
lemma $MT0\ \eta \Longrightarrow MT1\ \eta \Longrightarrow MT2\ \eta \Longrightarrow MT3\ \eta \Longrightarrow TNDm\ \eta$ **nitpick** ⟨*proof*⟩
lemma *MT123*: $MT1\ \eta \Longrightarrow MT2\ \eta \Longrightarrow MT3\ \eta$ ⟨*proof*⟩

4.3.4 Double negation introduction and elimination

DNI/DNE: double negation introduction/elimination (strong 'axiom-like' variants).

abbreviation *pDNI* (‹DNI^{-} -›) **where** $DNI^{a}\ \eta \equiv [a \vdash \eta(\eta\ a)]$
abbreviation *pDNE* (‹DNE^{-} -›) **where** $DNE^{a}\ \eta \equiv [\eta(\eta\ a) \vdash a]$
definition $DNI::'w\ \Omega$ **where** $DNI\ \eta \equiv \forall\, a.\ DNI^{a}\ \eta$
definition $DNE::'w\ \Omega$ **where** $DNE\ \eta \equiv \forall\, a.\ DNE^{a}\ \eta$
declare *DNI-def*[*neg*] *DNE-def*[*neg*]

CoP1 (resp. CoP2) can alternatively be defined as CoPw plus DNI (resp. DNE).

lemma $DNI\ \eta \Longrightarrow CoP1\ \eta$ **nitpick** ⟨*proof*⟩
lemma *CoP1-def2*: $CoP1\ \eta = (CoP0\ \eta \wedge DNI\ \eta)$ ⟨*proof*⟩
lemma $DNE\ \eta \Longrightarrow CoP2\ \eta$ **nitpick** ⟨*proof*⟩
lemma *CoP2-def2*: $CoP2\ \eta = (CoP0\ \eta \wedge DNE\ \eta)$ ⟨*proof*⟩

Explore some non-entailment relations:

lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow CoP0\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow TNDm\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow ECQm\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow MT0\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow MT1\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow MT2\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow MT3\ \eta$ **nitpick** ⟨*proof*⟩

DNI/DNE: double negation introduction/elimination (weak 'rule-like' variants).

abbreviation *prDNI* (‹*rDNI*$^{-}$ -›) **where** $rDNI^a\ \eta \equiv [\vdash a] \longrightarrow [\vdash \eta(\eta\ a)]$
abbreviation *prDNE* (‹*rDNE*$^{-}$ -›) **where** $rDNE^a\ \eta \equiv [\vdash \eta(\eta\ a)] \longrightarrow [\vdash a]$
definition *rDNI*::$'w\ \Omega$ **where** $rDNI\ \eta \equiv \forall\, a.\ rDNI^a\ \eta$
definition *rDNE*::$'w\ \Omega$ **where** $rDNE\ \eta \equiv \forall\, a.\ rDNE^a\ \eta$
declare *rDNI-def*[*neg*] *rDNE-def*[*neg*]

The 'rule-like' variants for DNI/DNE are strictly weaker than the 'axiom-like' ones.

lemma $DNI\ \eta \Longrightarrow rDNI\ \eta$ ⟨*proof*⟩
lemma $rDNI\ \eta \Longrightarrow DNI\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNE\ \eta \Longrightarrow rDNE\ \eta$ ⟨*proof*⟩
lemma $rDNE\ \eta \Longrightarrow DNE\ \eta$ **nitpick** ⟨*proof*⟩

The 'rule-like' variants for DNI/DNE follow already from modus tollens.

lemma *MT1-rDNI*: $MT1\ \eta \Longrightarrow rDNI\ \eta$ ⟨*proof*⟩
lemma *MT2-rDNE*: $MT2\ \eta \Longrightarrow rDNE\ \eta$ ⟨*proof*⟩

4.3.5 (Anti)Normality and its dual

nNORM (resp. nDNRM) is entailed by CoP1 (resp. CoP2).

lemma *CoP1-NORM*: $CoP1\ \eta \Longrightarrow nNORM\ \eta$ ⟨*proof*⟩
lemma *CoP2-DNRM*: $CoP2\ \eta \Longrightarrow nDNRM\ \eta$ ⟨*proof*⟩
lemma $DNI\ \eta \Longrightarrow nNORM\ \eta$ **nitpick** ⟨*proof*⟩
lemma $DNE\ \eta \Longrightarrow nDNRM\ \eta$ **nitpick** ⟨*proof*⟩

nNORM and nDNRM together entail the rule variant of DNI (rDNI).

lemma *nDNRM-rDNI*: $nNORM\ \eta \Longrightarrow nDNRM\ \eta \Longrightarrow rDNI\ \eta$ ⟨*proof*⟩
lemma $nNORM\ \eta \Longrightarrow nDNRM\ \eta \Longrightarrow rDNE\ \eta$ **nitpick** ⟨*proof*⟩

4.3.6 De Morgan laws

De Morgan laws correspond to anti-additivity and anti-multiplicativity).

DM3 (resp. DM4) are entailed by CoP0/ANTI together with DNE (resp. DNI).

lemma *CoP0-DNE-nMULTb*: $CoP0\ \eta \Longrightarrow DNE\ \eta \Longrightarrow nMULT^b\ \eta$ ⟨*proof*⟩
lemma *CoP0-DNI-nADDIa*: $CoP0\ \eta \Longrightarrow DNI\ \eta \Longrightarrow nADDI^a\ \eta$ ⟨*proof*⟩

From this follows that DM3 (resp. DM4) is entailed by CoP2 (resp. CoP1).

lemma *CoP2-nMULTb*: $CoP2\ \eta \Longrightarrow nMULT^b\ \eta$ ⟨*proof*⟩
lemma *CoP1-nADDIa*: $CoP1\ \eta \Longrightarrow nADDI^a\ \eta$ ⟨*proof*⟩

Explore some non-entailment relations:

lemma $CoP0\ \eta \Longrightarrow nADDI^a\ \eta \Longrightarrow nMULT^b\ \eta \Longrightarrow nNORM\ \eta \Longrightarrow nDNRM\ \eta \Longrightarrow DNI\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow nADDI^a\ \eta \Longrightarrow nMULT^b\ \eta \Longrightarrow nNORM\ \eta \Longrightarrow nDNRM\ \eta \Longrightarrow DNE\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow nADDI^a\ \eta \Longrightarrow nMULT^b\ \eta \Longrightarrow DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow ECQm\ \eta$ **nitpick** ⟨*proof*⟩
lemma $CoP0\ \eta \Longrightarrow nADDI^a\ \eta \Longrightarrow nMULT^b\ \eta \Longrightarrow DNI\ \eta \Longrightarrow DNE\ \eta \Longrightarrow TNDm\ \eta$ **nitpick** ⟨*proof*⟩

4.3.7 Strong contraposition (axiom-like)

Observe that the definitions below take implication as an additional parameter: ι.

lCoP: (local) contraposition (strong 'axiom-like' variants, using local consequence).

abbreviation *plCoP0* (‹*lCoP0*$^{--}$ - -›) **where** *lCoP0*^{ab} $\iota\ \eta \equiv [\iota\ a\ b \vdash \iota\ (\eta\ b)\ (\eta\ a)]$
abbreviation *plCoP1* (‹*lCoP1*$^{--}$ - -›) **where** *lCoP1*^{ab} $\iota\ \eta \equiv [\iota\ a\ (\eta\ b) \vdash \iota\ b\ (\eta\ a)]$
abbreviation *plCoP2* (‹*lCoP2*$^{--}$ - -›) **where** *lCoP2*^{ab} $\iota\ \eta \equiv [\iota\ (\eta\ a)\ b \vdash \iota\ (\eta\ b)\ a]$
abbreviation *plCoP3* (‹*lCoP3*$^{--}$ - -›) **where** *lCoP3*^{ab} $\iota\ \eta \equiv [\iota\ (\eta\ a)\ (\eta\ b) \vdash \iota\ b\ a]$
definition *lCoP0*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lCoP0* $\iota\ \eta \equiv \forall\ a\ b.\ lCoP0^{ab}\ \iota\ \eta$
definition *lCoP1*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lCoP1* $\iota\ \eta \equiv \forall\ a\ b.\ lCoP1^{ab}\ \iota\ \eta$
definition *lCoP2*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lCoP2* $\iota\ \eta \equiv \forall\ a\ b.\ lCoP2^{ab}\ \iota\ \eta$
definition *lCoP3*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lCoP3* $\iota\ \eta \equiv \forall\ a\ b.\ lCoP3^{ab}\ \iota\ \eta$

declare *lCoP0-def*[*neg*] *lCoP1-def*[*neg*] *lCoP2-def*[*neg*] *lCoP3-def*[*neg*]

All these contraposition variants are in general independent from each other. However if we employ classical implication we can verify some relationships.

lemma *lCoP1-def2*: *lCoP1*$(\rightarrow)\ \eta$ = (*lCoP0*$(\rightarrow)\ \eta \wedge$ *DNI* η) ⟨*proof*⟩
lemma *lCoP2-def2*: *lCoP2*$(\rightarrow)\ \eta$ = (*lCoP0*$(\rightarrow)\ \eta \wedge$ *DNE* η) ⟨*proof*⟩
lemma *lCoP1*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP0*$(\rightarrow)\ \eta$ ⟨*proof*⟩
lemma *lCoP0*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP1*$(\rightarrow)\ \eta$ **nitpick** ⟨*proof*⟩
lemma *lCoP2*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP0*$(\rightarrow)\ \eta$ ⟨*proof*⟩
lemma *lCoP0*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP2*$(\rightarrow)\ \eta$ **nitpick** ⟨*proof*⟩
lemma *lCoP3*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP0*$(\rightarrow)\ \eta$ ⟨*proof*⟩
lemma *lCoP0*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP3*$(\rightarrow)\ \eta$ **nitpick** ⟨*proof*⟩
lemma *lCoP123*: *lCoP1*$(\rightarrow)\ \eta \wedge$ *lCoP2*$(\rightarrow)\ \eta \Longrightarrow$ *lCoP3*$(\rightarrow)\ \eta$ ⟨*proof*⟩

Strong/axiom-like variants imply weak/rule-like ones as expected.

lemma *lCoP0*$(\rightarrow)\ \eta \Longrightarrow$ *CoP0* η ⟨*proof*⟩
lemma *lCoP1*$(\rightarrow)\ \eta \Longrightarrow$ *CoP1* η ⟨*proof*⟩
lemma *lCoP2*$(\rightarrow)\ \eta \Longrightarrow$ *CoP2* η ⟨*proof*⟩
lemma *lCoP3*$(\rightarrow)\ \eta \Longrightarrow$ *CoP3* η ⟨*proof*⟩

Explore some (non)entailment relations.

lemma *lCoP1-TND*: *lCoP1*$(\rightarrow)\ \eta \Longrightarrow$ *TND* η ⟨*proof*⟩
lemma *TND* $\eta \Longrightarrow$ *lCoP1*$(\rightarrow)\ \eta$ **nitpick** ⟨*proof*⟩
lemma *lCoP2-ECQ*: *lCoP2*$(\rightarrow)\ \eta \Longrightarrow$ *ECQ* η ⟨*proof*⟩
lemma *ECQ* $\eta \Longrightarrow$ *lCoP2*$(\rightarrow)\ \eta$ **nitpick** ⟨*proof*⟩

4.3.8 Local modus tollens axioms

lMT: (local) Modus tollens (strong, 'axiom-like' variants, using local consequence).

abbreviation *plMT0* (‹*lMT0*$^{--}$ - -›) **where** *lMT0*^{ab} $\iota\ \eta \equiv [\iota\ a\ b, \eta\ b \vdash \eta\ a]$
abbreviation *plMT1* (‹*lMT1*$^{--}$ - -›) **where** *lMT1*^{ab} $\iota\ \eta \equiv [\iota\ a\ (\eta\ b),\ b \vdash \eta\ a]$
abbreviation *plMT2* (‹*lMT2*$^{--}$ - -›) **where** *lMT2*^{ab} $\iota\ \eta \equiv [\iota\ (\eta\ a)\ b,\ \eta\ b \vdash a]$
abbreviation *plMT3* (‹*lMT3*$^{--}$ - -›) **where** *lMT3*^{ab} $\iota\ \eta \equiv [\iota\ (\eta\ a)\ (\eta\ b),\ b \vdash a]$
definition *lMT0*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lMT0* $\iota\ \eta \equiv \forall\ a\ b.\ lMT0^{ab}\ \iota\ \eta$
definition *lMT1*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lMT1* $\iota\ \eta \equiv \forall\ a\ b.\ lMT1^{ab}\ \iota\ \eta$
definition *lMT2*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lMT2* $\iota\ \eta \equiv \forall\ a\ b.\ lMT2^{ab}\ \iota\ \eta$
definition *lMT3*::$('w\ \sigma \Rightarrow 'w\ \sigma \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \Omega$ **where** *lMT3* $\iota\ \eta \equiv \forall\ a\ b.\ lMT3^{ab}\ \iota\ \eta$

declare *lMT0-def*[*neg*] *lMT1-def*[*neg*] *lMT2-def*[*neg*] *lMT3-def*[*neg*]

All these MT variants are in general independent from each other and also from (strong) CoP instances. However if we take classical implication we can verify that local MT and CoP are indeed equivalent.

lemma *lMT0*($\rightarrow$) η = *lCoP0*($\rightarrow$) η ⟨*proof*⟩
lemma *lMT1*($\rightarrow$) η = *lCoP1*($\rightarrow$) η ⟨*proof*⟩
lemma *lMT2*($\rightarrow$) η = *lCoP2*($\rightarrow$) η ⟨*proof*⟩
lemma *lMT3*($\rightarrow$) η = *lCoP3*($\rightarrow$) η ⟨*proof*⟩

4.3.9 Disjunctive syllogism

DS: disjunctive syllogism.

abbreviation *pDS1* (‹*DS1*$^{--}$ - -›) **where** *DS1*^{ab} ι η ≡ [$a \vee b \vdash \iota\ (\eta\ a)\ b$]
abbreviation *pDS2* (‹*DS2*$^{--}$ - -›) **where** *DS2*^{ab} ι η ≡ [$\iota\ (\eta\ a)\ b \vdash a \vee b$]
abbreviation *pDS3* (‹*DS3*$^{--}$ - -›) **where** *DS3*^{ab} ι η ≡ [$\eta\ a \vee b \vdash \iota\ a\ b$]
abbreviation *pDS4* (‹*DS4*$^{--}$ - -›) **where** *DS4*^{ab} ι η ≡ [$\iota\ a\ b \vdash \eta\ a \vee b$]
definition *DS1*::('*w* σ⇒'*w* σ⇒'*w* σ) ⇒ '*w* Ω **where** *DS1* ι η ≡ ∀ *a b*. *DS1*^{ab} ι η
definition *DS2*::('*w* σ⇒'*w* σ⇒'*w* σ) ⇒ '*w* Ω **where** *DS2* ι η ≡ ∀ *a b*. *DS2*^{ab} ι η
definition *DS3*::('*w* σ⇒'*w* σ⇒'*w* σ) ⇒ '*w* Ω **where** *DS3* ι η ≡ ∀ *a b*. *DS3*^{ab} ι η
definition *DS4*::('*w* σ⇒'*w* σ⇒'*w* σ) ⇒ '*w* Ω **where** *DS4* ι η ≡ ∀ *a b*. *DS4*^{ab} ι η

declare *DS1-def*[*neg*] *DS2-def*[*neg*] *DS3-def*[*neg*] *DS4-def*[*neg*]

All DS variants are in general independent from each other. However if we take classical implication we can verify that the pairs DS1-DS3 and DS2-DS4 are indeed equivalent.

lemma *DS1*($\rightarrow$) η = *DS3*($\rightarrow$) η ⟨*proof*⟩
lemma *DS2*($\rightarrow$) η = *DS4*($\rightarrow$) η ⟨*proof*⟩

Explore some (non)entailment relations.

lemma *DS1-nDNor*: *DS1*($\rightarrow$) η ⟹ *nDNRM* η ⟨*proof*⟩
lemma *DS2-nNor*: *DS2*($\rightarrow$) η ⟹ *nNORM* η ⟨*proof*⟩
lemma *lCoP2-DS1*: *lCoP2*($\rightarrow$) η ⟹ *DS1*($\rightarrow$) η ⟨*proof*⟩
lemma *lCoP1-DS2*: *lCoP1*($\rightarrow$) η ⟹ *DS2*($\rightarrow$) η ⟨*proof*⟩
lemma *CoP2* η ⟹ *DS1*($\rightarrow$) η **nitpick** ⟨*proof*⟩
lemma *CoP1* η ⟹ *DS2*($\rightarrow$) η **nitpick** ⟨*proof*⟩

end
theory *logics-quantifiers*
  imports *boolean-algebra-infinitary*
begin

4.4 Quantifiers (restricted and unrestricted)

Introduce pedagogically convenient notation.

notation *HOL.All* (‹Π›) **notation** *HOL.Ex* (‹Σ›)

Let us recall that in HOL we have:

lemma (∀ *x*. *P*) = Π(λ*x*. *P*) ⟨*proof*⟩
lemma (∃ *x*. *P*) = Σ(λ*x*. *P*) ⟨*proof*⟩
lemma Σ = (λ*P*. ¬Π(λ*x*. ¬*P x*)) ⟨*proof*⟩

We can introduce their respective 'w-type-lifted variants as follows:

definition *mforall*::('*i*⇒'*w* σ)⇒'*w* σ (‹**Π**-›)
  where **Π**φ ≡ λ*w*. ∀ *X*. φ *X w*
definition *mexists*::('*i*⇒'*w* σ)⇒'*w* σ (‹**Σ**-›)
  where **Σ**φ ≡ λ*w*. ∃ *X*. φ *X w*

To improve readability, we introduce for them standard binder notation.

notation *mforall* (**binder** ‹∀› [48]49) **notation** *mexists* (**binder** ‹∃› [48]49)

And thus we obtain the 'w-type-lifted variant of the standard (variable-binding) quantifiers.

lemma $(\forall X.\ \varphi) = \mathbf{\Pi}(\lambda X.\ \varphi)$ ⟨*proof*⟩
lemma $(\exists X.\ \varphi) = \mathbf{\Sigma}(\lambda X.\ \varphi)$ ⟨*proof*⟩

Quantifiers are dual to each other in the expected way.

lemma $\mathbf{\Pi}\varphi = -(\mathbf{\Sigma}\varphi^{-})$ ⟨*proof*⟩
lemma $(\forall X.\ \varphi\ X) = -(\exists X.\ -(\varphi\ X))$ ⟨*proof*⟩

Relationship between quantifiers and the infinitary supremum and infimum operations.

lemma *mforall-char*: $\mathbf{\Pi}\varphi = \bigwedge[\![\varphi\ \text{-}]\!]$ ⟨*proof*⟩
lemma *mexists-char*: $\mathbf{\Sigma}\varphi = \bigvee[\![\varphi\ \text{-}]\!]$ ⟨*proof*⟩

lemma *mforallb-char*: $(\forall X.\ \varphi) = \bigwedge[\![(\lambda X.\ \varphi)\ \text{-}]\!]$ ⟨*proof*⟩
lemma *mexistsb-char*: $(\exists X.\ \varphi) = \bigvee[\![(\lambda X.\ \varphi)\ \text{-}]\!]$ ⟨*proof*⟩

We now consider quantification restricted over constant and varying domains.

Constant domains: first generalization of quantifiers above (e.g. free logic).

definition *mforall-const*::$'i\ \sigma \Rightarrow ('i \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \sigma$ (‹$\mathbf{\Pi}$[-]-›)
  where $\mathbf{\Pi}[D]\varphi \equiv \lambda w.\ \forall X.\ (D\ X) \longrightarrow (\varphi\ X)\ w$
definition *mexists-const*::$'i\ \sigma \Rightarrow ('i \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \sigma$ (‹$\mathbf{\Sigma}$[-]-›)
  where $\mathbf{\Sigma}[D]\varphi \equiv \lambda w.\ \exists X.\ (D\ X)\ \wedge\ (\varphi\ X)\ w$

Constant-domain quantification generalises its unrestricted counterpart.

lemma $\mathbf{\Pi}\varphi = \mathbf{\Pi}[\top]\varphi$ ⟨*proof*⟩
lemma $\mathbf{\Sigma}\varphi = \mathbf{\Sigma}[\top]\varphi$ ⟨*proof*⟩

Constant-domain quantification can also be characterised using infimum and supremum.

lemma *mforall-const-char*: $\mathbf{\Pi}[D]\varphi = \bigwedge[\![\varphi\ D]\!]$ ⟨*proof*⟩
lemma *mexists-const-char*: $\mathbf{\Sigma}[D]\varphi = \bigvee[\![\varphi\ D]\!]$ ⟨*proof*⟩

Constant-domain quantifiers also allow us to nicely characterize the interaction between function composition and (restricted) quantification:

lemma *mforall-comp*: $\mathbf{\Pi}(\varphi \circ \psi) = \mathbf{\Pi}[\![\psi\ \text{-}]\!]\ \varphi$ ⟨*proof*⟩
lemma *mexists-comp*: $\mathbf{\Sigma}(\varphi \circ \psi) = \mathbf{\Sigma}[\![\psi\ \text{-}]\!]\ \varphi$ ⟨*proof*⟩

Varying domains: we can also restrict quantifiers by taking a 'functional domain' as additional parameter. The latter is a set-valued mapping each element 'i to a set of points (e.g. where it 'exists').

definition *mforall-var*::$('i \Rightarrow 'w\ \sigma) \Rightarrow ('i \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \sigma$ (‹$\mathbf{\Pi}${-}-›)
  where $\mathbf{\Pi}\{\psi\}\varphi \equiv \lambda w.\ \forall X.\ (\psi\ X)\ w \longrightarrow (\varphi\ X)\ w$
definition *mexists-var*::$('i \Rightarrow 'w\ \sigma) \Rightarrow ('i \Rightarrow 'w\ \sigma) \Rightarrow 'w\ \sigma$ (‹$\mathbf{\Sigma}${-}-›)
  where $\mathbf{\Sigma}\{\psi\}\varphi \equiv \lambda w.\ \exists X.\ (\psi\ X)\ w\ \wedge\ (\varphi\ X)\ w$

Varying-domain quantification generalizes its constant-domain counterpart.

lemma $\mathbf{\Pi}[D]\varphi = \mathbf{\Pi}\{D\rceil\}\varphi$ ⟨*proof*⟩
lemma $\mathbf{\Sigma}[D]\varphi = \mathbf{\Sigma}\{D\rceil\}\varphi$ ⟨*proof*⟩

Restricted quantifiers are dual to each other in the expected way.

lemma $\mathbf{\Pi}[D]\varphi = -(\mathbf{\Sigma}[D]\varphi^{-})$ ⟨*proof*⟩
lemma $\mathbf{\Pi}\{\psi\}\varphi = -(\mathbf{\Sigma}\{\psi\}\varphi^{-})$ ⟨*proof*⟩

We can use 2nd-order connectives on set-valued functions to encode restricted quantifiers as unrestricted.

lemma $\mathbf{\Pi}\{\psi\}\varphi = \mathbf{\Pi}(\psi \rightarrow^{:} \varphi)$ ⟨*proof*⟩
lemma $\mathbf{\Sigma}\{\psi\}\varphi = \mathbf{\Sigma}(\psi \wedge^{:} \varphi)$ ⟨*proof*⟩

Observe that using these operators has the advantage of allowing for binder notation.

lemma $\mathbf{\Pi}\{\psi\}\varphi = (\forall X.\ (\psi \rightarrow^{:} \varphi)\ X)$ ⟨*proof*⟩
lemma $\mathbf{\Sigma}\{\psi\}\varphi = (\exists X.\ (\psi \wedge^{:} \varphi)\ X)$ ⟨*proof*⟩

To sumarize: different sorts of restricted quantification can be emulated by employing 2nd-order operations to adequately relativize predicates.

lemma $\mathbf{\Pi}[D]\varphi = (\forall X.\ (D\restriction \rightarrow^{:} \varphi)\ X)$ ⟨*proof*⟩
lemma $\mathbf{\Pi}\{\top^{:}\}\varphi = (\forall X.\ (\top^{:} \rightarrow^{:} \varphi)\ X)$ ⟨*proof*⟩
lemma $\mathbf{\Pi}\varphi = \mathbf{\Pi}\{\top^{:}\}\varphi$ ⟨*proof*⟩
lemma $(\forall X.\ \varphi\ X) = \mathbf{\Pi}\{\top^{:}\}\varphi$ ⟨*proof*⟩

named-theorems *quant*
declare *mforall-def*[*quant*] *mexists-def*[*quant*]
mforall-const-def[*quant*] *mexists-const-def*[*quant*]
mforall-var-def[*quant*] *mexists-var-def*[*quant*]

end
theory *logics-quantifiers-example*
imports *logics-quantifiers conditions-positive-infinitary*
begin

4.5 Examples on Quantifiers

First-order quantification example.

lemma $(\forall x.\ A\ x \longrightarrow (\exists y.\ B\ x\ y)) \longleftrightarrow (\forall x.\ \exists y.\ A\ x \longrightarrow B\ x\ y)$ ⟨*proof*⟩
lemma $(\forall x.\ A\ x \rightarrow (\exists y.\ B\ x\ y)) = (\forall x.\ \exists y.\ A\ x \rightarrow B\ x\ y)$ ⟨*proof*⟩

Propositional quantification example.

lemma $\forall A.\ (\exists B.\ (A \longleftrightarrow \neg B))$ ⟨*proof*⟩
lemma $(\forall A.\ (\exists B.\ A \leftrightarrow -B)) = \top$ ⟨*proof*⟩

Drinker's principle.

lemma $\exists x.\ Drunk\ x \rightarrow (\forall y.\ Drunk\ y) = \top$
⟨*proof*⟩

Example in non-classical logics.

typedecl w
type-synonym $\sigma = (w\ \sigma)$

consts $\mathcal{C}::\sigma \Rightarrow \sigma$
abbreviation $\mathcal{I} \equiv \mathcal{C}^d$
abbreviation $CLOSURE\ \varphi \equiv ADDI\ \varphi \wedge EXPN\ \varphi \wedge NORM\ \varphi \wedge IDEM\ \varphi$
abbreviation $INTERIOR\ \varphi \equiv MULT\ \varphi \wedge CNTR\ \varphi \wedge DNRM\ \varphi \wedge IDEM\ \varphi$

definition *mforallInt*::$(\sigma \Rightarrow \sigma) \Rightarrow \sigma$ ($\langle \mathbf{\Pi}^I \text{-} \rangle$)
where $\mathbf{\Pi}^I\varphi \equiv \mathbf{\Pi}[fp\ \mathcal{I}]\varphi$
definition *mexistsInt*::$(\sigma \Rightarrow \sigma) \Rightarrow \sigma$ ($\langle \mathbf{\Sigma}^I \text{-} \rangle$)
where $\mathbf{\Sigma}^I\varphi \equiv \mathbf{\Sigma}[fp\ \mathcal{I}]\varphi$

notation *mforallInt* (**binder** ‹$\forall^I$› [48]49)
notation *mexistsInt* (**binder** ‹$\exists^I$› [48]49)

abbreviation *intneg* (‹$\neg^I$-›) **where** $\neg^I A \equiv \mathcal{I}^{d-}\ A$
abbreviation *parneg* (‹$\neg^C$-›) **where** $\neg^C A \equiv \mathcal{C}^{d-}\ A$

lemma $(\forall X.\ (\exists B.\ (X\ \leftrightarrow -B))) = \top$ ⟨*proof*⟩
lemma $(\forall^I X.\ (\exists^I B.\ (X\ \leftrightarrow \neg^I B))) = \top$ **nitpick** ⟨*proof*⟩

4.6 Barcan formula and its converse

The converse Barcan formula follows readily from monotonicity.

lemma *CBarcan1*: $MONO\ \varphi \Longrightarrow \forall \pi.\ \ \varphi(\forall x.\ \pi\ x)\ \leq (\forall x.\ \varphi(\pi\ x))$ ⟨*proof*⟩
lemma *CBarcan2*: $MONO\ \varphi \Longrightarrow \forall \pi.\ (\exists x.\ \varphi(\pi\ x)) \leq \varphi(\exists x.\ \pi\ x)$ ⟨*proof*⟩

However, the Barcan formula requires a stronger assumption (of an infinitary character).

lemma *Barcan1*: $iMULT^b\ \varphi \Longrightarrow \forall \pi.\ (\forall x.\ \varphi(\pi\ x)) \leq \varphi(\forall x.\ \pi\ x)$ ⟨*proof*⟩
lemma *Barcan2*: $iADDI^a\ \varphi \Longrightarrow \forall \pi.\ \varphi(\exists x.\ \pi\ x) \leq (\exists x.\ \varphi(\pi\ x))$ ⟨*proof*⟩

Converse Barcan Formula and composition.

lemma $MONO\ \varphi \Longrightarrow \forall \pi.\ \ \varphi(\mathbf{\Pi}\ \pi) \leq \mathbf{\Pi}(\varphi \circ \pi)$ ⟨*proof*⟩
lemma $MONO\ \varphi \Longrightarrow \forall \pi.\ \ \varphi(\mathbf{\Pi}[D]\ \pi) \leq \mathbf{\Pi}[D](\varphi \circ \pi)$ ⟨*proof*⟩
lemma $CNTR\ \varphi \Longrightarrow iMULT\ \varphi \Longrightarrow IDEM\ \varphi \Longrightarrow\ \forall \pi.\ \ \varphi(\mathbf{\Pi}\{\psi\}\ \pi) \leq \mathbf{\Pi}\{\psi\}(\varphi \circ \pi)$ **nitpick** ⟨*proof*⟩

Barcan Formula and composition.

lemma $iMULT^b\ \varphi \Longrightarrow \forall \pi.\ \mathbf{\Pi}(\varphi \circ \pi) \leq \varphi(\mathbf{\Pi}\ \pi)$ ⟨*proof*⟩
lemma $iMULT^b\ \varphi \Longrightarrow \forall \pi.\ \mathbf{\Pi}[D](\varphi \circ \pi) \leq \varphi(\mathbf{\Pi}[D]\ \pi)$ ⟨*proof*⟩
lemma $iADDI\ \varphi \Longrightarrow iMULT\ \varphi \Longrightarrow \forall \pi.\ \mathbf{\Pi}\{\psi\}(\varphi \circ \pi) \leq \varphi(\mathbf{\Pi}\{\psi\}\ \pi)$ **nitpick** ⟨*proof*⟩

end
theory *logics-LFI*
imports *logics-consequence conditions-relativized-infinitary*
begin

4.7 Logics of Formal Inconsistency (LFIs)

The LFIs are a family of paraconsistent logics featuring a 'consistency' operator $\circ$ that can be used to recover some classical properties of negation (in particular ECQ). We show a shallow semantical embedding of a family of self-extensional LFIs using the border operator as primitive.

Let us assume a concrete type w (for 'worlds' or 'points').

typedecl *w*

Let us assume the following primitive unary operation intended as a border operator.

consts $\mathcal{B}$::$w\ \sigma \Rightarrow w\ \sigma$

From the topological cube of opposition we have that:

abbreviation $\mathcal{C} \equiv (\mathcal{B}^{fp})^{d-}$
abbreviation $\mathcal{I} \equiv \mathcal{B}^{fp-}$
lemma $\mathcal{C}^{d-} = \mathcal{B}^{fp}$ ⟨*proof*⟩

Let us recall that:

lemma *expn-cntr*: $EXPN\ \mathcal{C} = CNTR\ \mathcal{B}$ ⟨*proof*⟩

For LFIs we use the negation previously defined as $\mathcal{C}^{d-} = \mathcal{B}^{fp}$.

abbreviation *neg* (‹¬›[70]71) **where** $neg \equiv \mathcal{B}^{fp}$

In terms of the border operator the negation looks as follows (under appropriate assumptions):

lemma *neg-char*: $CNTR\ \mathcal{B} \Longrightarrow \neg A = (-A \vee \mathcal{B}\ A)$ ⟨*proof*⟩

This negation is of course boldly paraconsistent (for both local and global consequence).

lemma $[a, \neg a \vdash b]$ **nitpick** ⟨*proof*⟩
lemma $[a, \neg a \vdash \neg b]$ **nitpick** ⟨*proof*⟩
lemma $[a, \neg a \vdash_g b]$ **nitpick** ⟨*proof*⟩
lemma $[a, \neg a \vdash_g \neg b]$ **nitpick** ⟨*proof*⟩

We define two pairs of in/consistency operators and show how they relate to each other. Using LFIs terminology, the minimal logic so encoded corresponds to RmbC + 'ciw' axiom.

abbreviation *op-inc-a*::$w\ \sigma \Rightarrow w\ \sigma$ (‹$\cdot^A$›) [57]58)
  where $\cdot^A A \equiv A \wedge \neg A$
abbreviation *op-con-a*::$w\ \sigma \Rightarrow w\ \sigma$ (‹$\circ^A$›) [57]58)
  where $\circ^A A \equiv -\cdot^A A$
abbreviation *op-inc-b*::$w\ \sigma \Rightarrow w\ \sigma$ (‹$\cdot^B$›) [57]58)
  where $\cdot^B A \equiv \mathcal{B}\ A$
abbreviation *op-con-b*::$w\ \sigma \Rightarrow w\ \sigma$ (‹$\circ^B$›) [57]58)
  where $\circ^B A \equiv \mathcal{B}^-\ A$

Observe that assumming CNTR for border we are allowed to exchange A and B variants.

lemma *pincAB*: $CNTR\ \mathcal{B} \Longrightarrow \cdot^A A = \cdot^B A$ ⟨*proof*⟩
lemma *pconAB*: $CNTR\ \mathcal{B} \Longrightarrow \circ^A A = \circ^B A$ ⟨*proof*⟩

Variants A and B give us slightly different properties (there are countermodels for those not shown).

lemma *Prop1*: $\circ^B A = \mathcal{I}^{fp}\ A$ ⟨*proof*⟩
lemma *Prop2*: $\circ^A A = (A \rightarrow \mathcal{I}\ A)$ ⟨*proof*⟩
lemma *Prop3*: $fp\ \mathcal{C}\ A \longleftrightarrow [\vdash \circ^B -A]$ ⟨*proof*⟩
lemma *Prop4a*: $fp\ \mathcal{I}\ A \longleftrightarrow [\vdash \circ^B A]$ ⟨*proof*⟩
lemma *Prop4b*: $fp\ \mathcal{I}\ A \longrightarrow [\vdash \circ^A A]$ ⟨*proof*⟩

The 'principle of gentle explosion' works for both variants (both locally and globally).

lemma $[\circ^A a, a, \neg a \vdash b]$ ⟨*proof*⟩
lemma $[\circ^A a, a, \neg a \vdash_g b]$ ⟨*proof*⟩
lemma $[\circ^B a, a, \neg a \vdash b]$ ⟨*proof*⟩
lemma $[\circ^B a, a, \neg a \vdash_g b]$ ⟨*proof*⟩

abbreviation $BORDER\ \varphi \equiv nMULTr\ \varphi \wedge CNTR\ \varphi \wedge nDNRM\ \varphi \wedge nIDEMr^b\ \varphi$

We show how (local) contraposition variants (among others) can be recovered using the consistency operators.

lemma $[\circ^A b, a \rightarrow b \vdash \neg b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop0-A*: $CNTR\ \mathcal{B} \longrightarrow [\circ^A b,\ \ a \rightarrow b \vdash \neg b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[\circ^B b, a \rightarrow b \vdash \neg b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop0-B*: $CNTR\ \mathcal{B} \longrightarrow [\circ^B b,\ \ a \rightarrow b \vdash \neg b \rightarrow \neg a]$ ⟨*proof*⟩

lemma $[\circ^A b,\ a \rightarrow \neg b \vdash b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop1-A*: $CNTR\ \mathcal{B} \longrightarrow [\circ^A b,\ a \rightarrow \neg b \vdash\ \ b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[\circ^B b,\ a \rightarrow \neg b \vdash b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop1-B*: $CNTR\ \mathcal{B} \longrightarrow [\circ^B b,\ a \rightarrow \neg b \vdash\ \ b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[\circ^A b,\ \neg a \rightarrow b \vdash \neg b \rightarrow a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop2-A*: $CNTR\ \mathcal{B} \longrightarrow [\circ^A b,\ \neg a \rightarrow b \vdash \neg b \rightarrow a]$ ⟨*proof*⟩
lemma $[\circ^B b,\ \neg a \rightarrow b \vdash \neg b \rightarrow a]$ **nitpick** ⟨*proof*⟩
lemma *cons-lcop2-B*: $CNTR\ \mathcal{B} \longrightarrow [\circ^B b,\ \neg a \rightarrow b \vdash \neg b \rightarrow a]$ ⟨*proof*⟩

The following axioms are commonly employed in the literature on LFIs to obtain stronger logics. We explore under which conditions they can be assumed while keeping the logic boldly paraconsistent.

abbreviation *cf* **where** $cf \equiv \forall\, P.\ [\neg\neg P \vdash P]$
abbreviation *ce* **where** $ce \equiv \forall\, P.\ [P \vdash \neg\neg P]$
abbreviation *ciw-a* **where** $ciw\text{-}a \equiv \forall\, P.\ [\vdash \circ^A P \vee \cdot^A P]$
abbreviation *ciw-b* **where** $ciw\text{-}b \equiv \forall\, P.\ [\vdash \circ^B P \vee \cdot^B P]$
abbreviation *ci-a* **where** $ci\text{-}a \equiv \forall\, P.\ [\neg(\circ^A P) \vdash \cdot^A P]$
abbreviation *ci-b* **where** $ci\text{-}b \equiv \forall\, P.\ [\neg(\circ^B P) \vdash \cdot^B P]$
abbreviation *cl-a* **where** $cl\text{-}a \equiv \forall\, P.\ \ [\neg(\cdot^A P) \vdash \circ^A P]$
abbreviation *cl-b* **where** $cl\text{-}b \equiv \forall\, P.\ \ [\neg(\cdot^B P) \vdash \circ^B P]$
abbreviation *ca-conj-a* **where** $ca\text{-}conj\text{-}a \equiv \forall\, P\ Q.\ [\circ^A P, \circ^A Q \vdash \circ^A (P \wedge Q)]$
abbreviation *ca-conj-b* **where** $ca\text{-}conj\text{-}b \equiv \forall\, P\ Q.\ [\circ^B P, \circ^B Q \vdash \circ^B (P \wedge Q)]$
abbreviation *ca-disj-a* **where** $ca\text{-}disj\text{-}a \equiv \forall\, P\ Q.\ [\circ^A P, \circ^A Q \vdash \circ^A (P \vee Q)]$
abbreviation *ca-disj-b* **where** $ca\text{-}disj\text{-}b \equiv \forall\, P\ Q.\ [\circ^B P, \circ^B Q \vdash \circ^B (P \vee Q)]$
abbreviation *ca-impl-a* **where** $ca\text{-}impl\text{-}a \equiv \forall\, P\ Q.\ [\circ^A P, \circ^A Q \vdash \circ^A (P \rightarrow Q)]$
abbreviation *ca-impl-b* **where** $ca\text{-}impl\text{-}b \equiv \forall\, P\ Q.\ [\circ^B P, \circ^B Q \vdash \circ^B (P \rightarrow Q)]$
abbreviation *ca-a* **where** $ca\text{-}a \equiv ca\text{-}conj\text{-}a \wedge ca\text{-}disj\text{-}a \wedge ca\text{-}impl\text{-}a$
abbreviation *ca-b* **where** $ca\text{-}b \equiv ca\text{-}conj\text{-}b \wedge ca\text{-}disj\text{-}b \wedge ca\text{-}impl\text{-}b$

cf

lemma $BORDER\ \mathcal{B} \Longrightarrow cf$ **nitpick** ⟨*proof*⟩

ce

lemma $BORDER\ \mathcal{B} \Longrightarrow ce$ **nitpick** ⟨*proof*⟩

ciw

lemma *prop-ciw-a*: *ciw-a* ⟨*proof*⟩
lemma *prop-ciw-b*: *ciw-b* ⟨*proof*⟩

ci

lemma $BORDER\ \mathcal{B} \Longrightarrow ci\text{-}a$ **nitpick** ⟨*proof*⟩
lemma $BORDER\ \mathcal{B} \Longrightarrow ci\text{-}b$ **nitpick** ⟨*proof*⟩

cl

lemma $BORDER\ \mathcal{B} \Longrightarrow cl\text{-}a$ **nitpick** ⟨*proof*⟩
lemma $BORDER\ \mathcal{B} \Longrightarrow cl\text{-}b$ **nitpick** ⟨*proof*⟩

ca-conj

lemma *prop-ca-conj-b*: $nMULT^b\ \mathcal{B} = ca\text{-}conj\text{-}b$ ⟨*proof*⟩
lemma *prop-ca-conj-a*: $nMULTr^b\ \mathcal{B} = ca\text{-}conj\text{-}a$ ⟨*proof*⟩

ca-disj

lemma *prop-ca-disj-b*: $ADDI^a\ \mathcal{B} = ca\text{-}disj\text{-}b$ ⟨*proof*⟩

lemma *prop-ca-disj-a*: $nMULTr^a\ \mathcal{B} = ca\text{-}disj\text{-}a\ \langle proof \rangle$

ca-impl

lemma $BORDER\ \mathcal{B} \Longrightarrow ca\text{-}impl\text{-}a$ **nitpick** $\langle proof \rangle$
lemma $BORDER\ \mathcal{B} \Longrightarrow ca\text{-}impl\text{-}b$ **nitpick** $\langle proof \rangle$

end
theory *logics-LFU*
imports *logics-consequence conditions-relativized-infinitary*
begin

4.8 Logics of Formal Undeterminedness (LFUs)

The LFUs are a family of paracomplete logics featuring a 'determinedness' operator ¤ that can be used to recover some classical properties of negation (in particular TND). LFUs behave in a sense dually to LFIs. Both can be semantically embedded as extensions of Boolean algebras. We show a shallow semantical embedding of a family of self-extensional LFUs using the closure operator as primitive.

typedecl w

consts $\mathcal{C}::w\ \sigma \Rightarrow w\ \sigma$

abbreviation $\mathcal{I} \equiv \mathcal{C}^d$
abbreviation $\mathcal{B} \equiv (\mathcal{C}^{fp})^d$

lemma $EXPN\ \mathcal{C} = CNTR\ \mathcal{B}\ \langle proof \rangle$
lemma $EXPN\ \mathcal{C} = CNTR\ \mathcal{I}\ \langle proof \rangle$

For LFUs we use the negation previously defined as $\mathcal{I}^{d-} = \mathcal{C}^-$.

abbreviation $neg\ (\langle\neg\text{-}\rangle[70]\,71)$ **where** $neg \equiv \mathcal{C}^-$

In terms of the border operator the negation looks as follows:

lemma *neg-char*: $EXPN\ \mathcal{C} \Longrightarrow \neg A = (-A \wedge \mathcal{B}^d\ A)\ \langle proof \rangle$

abbreviation $CLOSURE\ \varphi \equiv ADDI\ \varphi \wedge EXPN\ \varphi \wedge NORM\ \varphi \wedge IDEM\ \varphi$

This negation is of course paracomplete.

lemma $CLOSURE\ \mathcal{C} \Longrightarrow [\vdash a \vee \neg a]$ **nitpick** $\langle proof \rangle$

We define two pairs of un/determinedness operators and show how they relate to each other. This logic corresponds to the paracomplete dual of the LFI 'RmbC-ciw'.

abbreviation *op-det-a*$::w\ \sigma \Rightarrow w\ \sigma\ (\langle ¤^A\text{-}\rangle\ [57]58)$
where $¤^A A \equiv A \vee \neg A$
abbreviation *op-und-a*$::w\ \sigma \Rightarrow w\ \sigma\ (\langle \star^A\text{-}\rangle\ [57]58)$
where $\star^A A \equiv -¤^A A$
abbreviation *op-det-b*$::w\ \sigma \Rightarrow w\ \sigma\ (\langle ¤^B\text{-}\rangle\ [57]58)$
where $¤^B A \equiv \mathcal{B}^d\ A$
abbreviation *op-und-b*$::w\ \sigma \Rightarrow w\ \sigma\ (\langle \star^B\text{-}\rangle\ [57]58)$
where $\star^B A \equiv \mathcal{B}^{d-}\ A$

Observe that assumming EXPN for closure we are allowed to exchange A and B variants.

lemma *pundAB*: $EXPN\ \mathcal{C} \Longrightarrow \star^A A = \star^B A$ ⟨*proof*⟩
lemma *pdetAB*: $EXPN\ \mathcal{C} \Longrightarrow ¤^A A = ¤^B A$ ⟨*proof*⟩

Variants A and B give us slightly different properties (there are countermodels for those not shown).

lemma *Prop1*: $¤^B A = \mathcal{C}^{fp}\ A$ ⟨*proof*⟩
lemma *Prop2*: $¤^A A = (\mathcal{C}\ A \rightarrow A)$ ⟨*proof*⟩
lemma *Prop3*: $fp\ \mathcal{I}\ A \longleftrightarrow [\vdash\ ¤^B{-}A]$ ⟨*proof*⟩
lemma *Prop4a*: $fp\ \mathcal{C}\ A \longleftrightarrow [\vdash\ ¤^B A]$ ⟨*proof*⟩
lemma *Prop4b*: $fp\ \mathcal{C}\ A \longrightarrow [\vdash\ ¤^A A]$ ⟨*proof*⟩

Recovering TND works for both variants.

lemma $[¤^A a \vdash a, \neg a]$ ⟨*proof*⟩
lemma $[\vdash \star^A a \vee a \vee \neg a]$ ⟨*proof*⟩
lemma $[¤^B a \vdash a, \neg a]$ ⟨*proof*⟩
lemma $[\vdash \star^B a \vee a \vee \neg a]$ ⟨*proof*⟩

We show how (local) contraposition variants (among others) can be recovered using the determinedness operators.

lemma $[¤^A a, a \rightarrow b \vdash \neg b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop0-A*: $EXPN\ \mathcal{C} \Longrightarrow [¤^A a,\ a \rightarrow b \vdash \neg b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[¤^B a, a \rightarrow b \vdash \neg b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop0-B*: $EXPN\ \mathcal{C} \Longrightarrow [¤^B a,\ a \rightarrow b \vdash \neg b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[¤^A a, a \rightarrow \neg b \vdash b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop1-A*: $EXPN\ \mathcal{C} \Longrightarrow [¤^A a, a \rightarrow \neg b \vdash\ b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[¤^B a, a \rightarrow \neg b \vdash b \rightarrow \neg a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop1-B*: $EXPN\ \mathcal{C} \Longrightarrow [¤^B a, a \rightarrow \neg b \vdash\ b \rightarrow \neg a]$ ⟨*proof*⟩
lemma $[¤^A a, \neg a \rightarrow b \vdash \neg b \rightarrow a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop2-A*: $EXPN\ \mathcal{C} \Longrightarrow [¤^A a, \neg a \rightarrow b \vdash \neg b \rightarrow a]$ ⟨*proof*⟩
lemma $[¤^B a, \neg a \rightarrow b \vdash \neg b \rightarrow a]$ **nitpick** ⟨*proof*⟩
lemma *det-lcop2-B*: $EXPN\ \mathcal{C} \Longrightarrow [¤^B a, \neg a \rightarrow b \vdash \neg b \rightarrow a]$ ⟨*proof*⟩

end

References

[1] D. Fuenmayor. Semantical investigations on non-classical logics with recovery operators: Negation. *The Logic Journal of the IGPL. Special Issue on Non-classical Modal and Predicate Logics*, 202X. To appear (preprint: https://arxiv.org/abs/2104.04284).