Belief Revision Theory

Valentin Fouillard Safouan TAHA Frederic Boulanger
Nicolas Sabouret

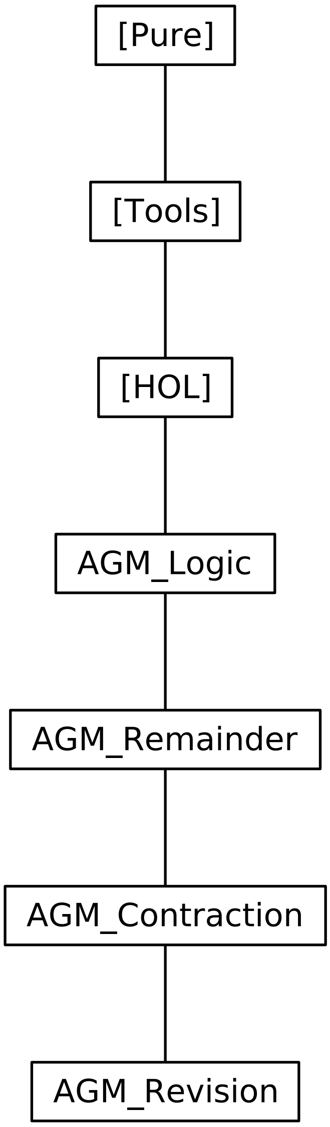
December 14, 2021

Abstract

The 1985 paper by Carlos Alchourrón, Peter Gärdenfors, and David Makinson (AGM), “On the Logic of Theory Change: Partial Meet Contraction and Revision Functions” launches a large and rapidly growing literature that employs formal models and logics to handle changing beliefs of a rational agent and to take into account new piece of information observed by this agent. In 2011, a review book titled ”AGM 25 Years: Twenty-Five Years of Research in Belief Change” was edited to summarize the first twenty five years of works based on AGM.

This HOL-based AFP entry is a faithful formalization of the AGM operators (e.g. contraction, revision, remainder ...) axiomatized in the original paper. It also contains the proofs of all the theorems stated in the paper that show how these operators combine. Both proofs of Harper and Levi identities are established.

Contents


[Pure]
[Tools]
[HOL]
AGM_Logic
AGM_Remainder
AGM_Contraction
AGM_Revision

1 Introduction

The 1985 paper by Carlos Alchourrón, Peter Gärdenfors, and David Makinson (AGM), "On the Logic of Theory Change: Partial Meet Contraction and Revision Functions" [1] launches a large and rapidly growing literature that employs formal models and logics to handle changing beliefs of a rational agent and to take into account new piece of information observed by this agent. In 2011, a review book titled "AGM 25 Years: Twenty-Five Years of Research in Belief Change" was edited to summarize the first twenty five years of works based on AGM [2].

According to Google Scholar, the original AGM paper was cited 4000 times! This AFP entry is HOL-based and it is a faithful formalization of the logic operators (e.g. contraction, revision, remainder ...) axiomatized in the AGM paper. It also contains the proofs of all the theorems stated in the paper that show how these operators combine. Both proofs of Harper and Levi identities are established.

A belief state can be considered as a consistent set of beliefs (logical propositions) closed under logical reasoning. Belief changes represent the operations that apply on a belief state to remove some of it and/or to add new beliefs (propositions). In the latter case, it is possible that other beliefs are affected by these changes (to preserve consistency for example). AGM define several postulates to guarantee that such operations preserve consistency meaning that the agent keeps rational. Three kinds of operators are defined :

- The contraction $\div$: where a proposition is removed from a belief set
- The expansion $\oplus$: where a proposition is added to a belief set
- The revision $*$: where a proposition is added to a belief set such that the belief set remains consistent

In this AFP entry, there are three theory files:

1. The AGM Logic file contains a classification of logics used in the AGM framework.
2. The AGM Remainder defines a important operator used in the AGM framework.
3. The AGM Contraction file contains the postulates of the AGM contraction and its relation with the meet contraction.
4. The AGM Revision file contains the postulates of the AGM revision and its relation with the meet revision.

2 Logics

The AGM framework depends on the underlying logic used to express beliefs. AGM requires at least a Tarskian propositional calculus. If this logic is also supra-classical and/or compact, new properties are established and the main theorems of AGM are strengthened. To model AGM it is therefore important to start by formalizing this underlying logic and its various extensions. We opted for a deep embedding in HOL which required the redefinition of all the logical operators and an axiomatization of their rules. This is certainly not efficient in terms of proof but it gives a total control of our formalization and an assurance that the logic used has no hidden properties depending on the Isabelle/HOL implementation. We use the Isabelle *locales* feature and we take advantage of the inheritance/extension mechanisms between locales.

2.1 Tarskian Logic

The first locale formalizes a Tarskian logic based on the famous Tarski's consequence operator: *Cn A* which gives the set of all propositions (**closure**) that can be inferred from the set of propositions A, Exactly as it is classically axiomatized in the literature, three assumptions of the locale define the consequence operator.

locale *Tarskian-logic* =
fixes *Cn*::‹$'a\ set \Rightarrow\ 'a\ set$›
assumes *monotonicity-L*: ‹$A \subseteq B \Longrightarrow Cn(A) \subseteq Cn(B)$›
  and *inclusion-L*: ‹$A \subseteq Cn(A)$›
  and *transitivity-L*: ‹$Cn(Cn(A)) \subseteq Cn(A)$›

— Short notation for "φ can be inferred from the propositions in A".
fixes *infer*::‹$'a\ set \Rightarrow\ 'a \Rightarrow bool$› (**infix** ‹$\vdash$› *50*)
defines ‹$A \vdash \varphi \equiv \varphi \in Cn(A)$›

— φ is valid (a tautology) if it can be inferred from nothing.
fixes *valid*::‹$'a \Rightarrow bool$› (‹$\Vdash$›)
defines ‹$\Vdash \varphi \equiv \{\} \vdash \varphi$›

— $A \oplus \varphi$ is all that can be infered from A and φ.
fixes *expansion*::‹$'a\ set \Rightarrow\ 'a \Rightarrow\ 'a\ set$› (**infix** ‹$\oplus$› *57*)
defines ‹$A \oplus \varphi \equiv Cn(A \cup \{\varphi\})$›

begin

lemma *idempotency-L*: ‹$Cn(Cn(A)) = Cn(A)$›
 by (*simp add*: *inclusion-L transitivity-L subset-antisym*)

lemma *assumption-L*: ‹$\varphi \in A \Longrightarrow A \vdash \varphi$›
 using *inclusion-L infer-def* **by** *blast*

lemma *validD-L*: ‹$\Vdash \varphi \Longrightarrow \varphi \in Cn(A)$›
  using *monotonicity-L valid-def infer-def* **by** *fastforce*

lemma *valid-expansion*: ‹$K = Cn(A) \Longrightarrow \Vdash \varphi \Longrightarrow K \oplus \varphi = K$›
  by (*simp add*: *idempotency-L insert-absorb validD-L valid-def expansion-def*)

lemma *transitivity2-L*:
  assumes ‹$\forall \varphi \in B.\ A \vdash \varphi$›
    and ‹$B \vdash \psi$›
   shows ‹$A \vdash \psi$›
proof –
  from *assms*(*1*) **have** ‹$B \subseteq Cn(A)$› **by** (*simp add*: *infer-def subsetI*)
  hence ‹$Cn(B) \subseteq Cn(A)$› **using** *idempotency-L monotonicity-L* **by** *blast*
  moreover from *assms*(*2*) **have** ‹$\psi \in Cn(B)$› **by** (*simp add*: *infer-def*)
  ultimately show *?thesis* **using** *infer-def* **by** *blast*
qed

lemma *Cn-same*: ‹$(Cn(A) = Cn(B)) \longleftrightarrow (\forall C.\ A \subseteq Cn(C) \longleftrightarrow B \subseteq Cn(C))$›
proof
  { **assume** *h*:‹$Cn(A) = Cn(B)$›
   from *h* **have** ‹$\forall \varphi \in B.\ A \vdash \varphi$›
    by (*simp add*: *Tarskian-logic.assumption-L Tarskian-logic-axioms infer-def*)
   moreover from *h*[*symmetric*] **have** ‹$\forall \varphi \in A.\ B \vdash \varphi$›
    by (*simp add*: *Tarskian-logic.assumption-L Tarskian-logic-axioms infer-def*)
   ultimately have ‹$\forall C.\ A \subseteq Cn(C) \longleftrightarrow B \subseteq Cn(C)$›
    using *h idempotency-L inclusion-L monotonicity-L* **by** *blast*
  } **thus** ‹$Cn(A) = Cn(B) \Longrightarrow \forall C.\ (A \subseteq Cn(C)) = (B \subseteq Cn(C))$› **.**
next
  { **assume** *h*:‹$\forall C.\ (A \subseteq Cn(C)) = (B \subseteq Cn(C))$›
    from *h* **have** ‹$(A \subseteq Cn(A)) = (B \subseteq Cn(A))$› **and** ‹$(A \subseteq Cn(B)) = (B \subseteq$
$Cn(B))$› **by** *simp+*
   hence ‹$B \subseteq Cn(A)$› **and** ‹$A \subseteq Cn(B)$› **by** (*simp add*: *inclusion-L*)+
   hence ‹$Cn(A) = Cn(B)$›
    using *idempotency-L monotonicity-L* **by** *blast*
  } **thus** ‹$(\forall C.\ (A \subseteq Cn(C)) = (B \subseteq Cn(C))) \Longrightarrow Cn(A) = Cn(B)$› **.**
qed

— The closure of the union of two consequence closures.
lemma *Cn-union*: ‹$Cn(Cn(A) \cup Cn(B)) = Cn(A \cup B)$›
proof
  have ‹$Cn(Cn(A) \cup Cn(B)) \subseteq Cn(Cn\ (A \cup B))$› **by** (*simp add*: *monotonicity-L*)
  thus ‹$Cn(Cn(A) \cup Cn(B)) \subseteq Cn(A \cup B)$› **by** (*simp add*: *idempotency-L*)
next
  have ‹$(A \cup B) \subseteq (Cn(A) \cup Cn(B))$› **using** *inclusion-L* **by** *blast*
  thus ‹$Cn(A \cup B) \subseteq Cn(Cn(A) \cup Cn(B))$› **by** (*simp add*: *monotonicity-L*)
qed

— The closure of an infinite union of consequence closures.

lemma *Cn-Union*: ‹$Cn(\bigcup\{Cn(B)|B.\ P\ B\}) = Cn(\bigcup\{B.\ P\ B\})$› (**is** ‹$?A = ?B$›)
proof
  have ‹$?A \subseteq Cn\ ?B$›
    apply(*rule monotonicity-L*, *rule Union-least*, *auto*)
    by (*metis Sup-upper in-mono mem-Collect-eq monotonicity-L*)
  then show ‹$?A \subseteq ?B$›
    by (*simp add*: *idempotency-L*)
next
  show ‹$?B \subseteq ?A$›
    by (*metis* (*mono-tags*, *lifting*) *Union-subsetI inclusion-L mem-Collect-eq monotonicity-L*)
qed

— The intersection of two closures is closed.
lemma *Cn-inter*: ‹$K = Cn(A) \cap Cn(B) \Longrightarrow K = Cn(K)$›
proof –
  { fix K **assume** h:‹$K = Cn(A) \cap Cn(B)$›
    from h **have** ‹$K \subseteq Cn(A)$› **and** ‹$K \subseteq Cn(B)$› **by** *simp+*
    hence ‹$Cn(K) \subseteq Cn(A)$› **and** ‹$Cn(K) \subseteq Cn(B)$› **using** *idempotency-L monotonicity-L* **by** *blast+*
    hence ‹$Cn(K) \subseteq Cn(A) \cap Cn(B)$› **by** *simp*
    with h **have** ‹$K = Cn(K)$› **by** (*simp add*: *inclusion-L subset-antisym*)
  } thus ‹$K = Cn(A) \cap Cn(B) \Longrightarrow K = Cn(K)$› **.**
qed

— An infinite intersection of closures is closed.
lemma *Cn-Inter*: ‹$K = \bigcap\{Cn(B)|B.\ P\ B\} \Longrightarrow K = Cn(K)$›
proof –
  { fix K **assume** h:‹$K = \bigcap\{Cn(B)|B.\ P\ B\}$›
    from h **have** ‹$\forall B.\ P\ B \longrightarrow K \subseteq Cn(B)$› **by** *blast*
    hence ‹$\forall B.\ P\ B \longrightarrow Cn(K) \subseteq Cn(B)$› **using** *idempotency-L monotonicity-L* **by** *blast*
    hence ‹$Cn(K) \subseteq \bigcap\{Cn(B)|B.\ P\ B\}$› **by** *blast*
    with h **have** ‹$K = Cn(K)$› **by** (*simp add*: *inclusion-L subset-antisym*)
  } thus ‹$K = \bigcap\{Cn(B)|B.\ P\ B\} \Longrightarrow K = Cn(K)$› **.**
qed

end

2.2 Supraclassical Logic

A Tarskian logic has only one abstract operator catching the notion of consequence. A basic case of such a logic is a **Supraclassical** logic that is a logic with all classical propositional operators (e.g. conjunction ($\wedge$), implication($\longrightarrow$), negation ($\neg$) ...) together with their classical semantics.

We define a new locale. In order to distinguish the propositional operators of our supraclassical logic from those of Isabelle/HOL, we use dots (e.g. $.\wedge.$ stands for conjunction). We axiomatize the introduction and elimination rules of each operator as it is commonly established in the classical literature. As explained before, we give priority to a complete control of our logic instead of an efficient shallow embedding in Isabelle/HOL.

locale *Supraclassical-logic* = *Tarskian-logic* +

fixes *true-PL*:: ‹$'a$› (‹$\top$›)
 and *false-PL*:: ‹$'a$› (‹$\bot$›)
 and *imp-PL*:: ‹$'a \Rightarrow 'a \Rightarrow 'a$› (**infix** ‹$.\longrightarrow.$› *55*)
 and *not-PL*:: ‹$'a \Rightarrow 'a$› (‹$.\neg$›)
 and *conj-PL*:: ‹$'a \Rightarrow 'a \Rightarrow 'a$› (**infix** ‹$.\wedge.$› *55*)
 and *disj-PL*:: ‹$'a \Rightarrow 'a \Rightarrow 'a$› (**infix** ‹$.\vee.$› *55*)
 and *equiv-PL*:: ‹$'a \Rightarrow 'a \Rightarrow 'a$› (**infix** ‹$.\longleftrightarrow.$› *55*)

assumes *true-PL*: ‹$A \vdash \top$›

 and *false-PL*: ‹$\{\bot\} \vdash p$›

 and *impI-PL*: ‹$A \cup \{p\} \vdash q \Longrightarrow A \vdash (p \;.\!\longrightarrow\!.\; q)$›
 and *mp-PL*: ‹$A \vdash p \;.\!\longrightarrow\!.\; q \Longrightarrow A \vdash p \Longrightarrow A \vdash q$›

 and *notI-PL*: ‹$A \vdash p \;.\!\longrightarrow\!.\; \bot \Longrightarrow A \vdash .\neg\ p$›
 and *notE-PL*: ‹$A \vdash .\neg\ p \Longrightarrow A \vdash (p \;.\!\longrightarrow\!.\; \bot)$›

 and *conjI-PL*: ‹$A \vdash p \Longrightarrow A \vdash q \Longrightarrow A \vdash (p \;.\wedge.\; q)$›
 and *conjE1-PL*: ‹$A \vdash p \;.\wedge.\; q \Longrightarrow A \vdash p$›
 and *conjE2-PL*: ‹$A \vdash p \;.\wedge.\; q \Longrightarrow A \vdash q$›

 and *disjI1-PL*: ‹$A \vdash p \Longrightarrow A \vdash (p \;.\vee.\; q)$›
 and *disjI2-PL*: ‹$A \vdash q \Longrightarrow A \vdash (p \;.\vee.\; q)$›
 and *disjE-PL*: ‹$A \vdash p \;.\vee.\; q \Longrightarrow A \vdash p \;.\!\longrightarrow\!.\; r \Longrightarrow A \vdash q.\!\longrightarrow\!.\; r \Longrightarrow A \vdash r$›

 and *equivI-PL*: ‹$A \vdash p \;.\!\longrightarrow\!.\; q \Longrightarrow A \vdash q \;.\!\longrightarrow\!.\; p \Longrightarrow A \vdash (p \;.\!\longleftrightarrow\!.\; q)$›
 and *equivE1-PL*: ‹$A \vdash p \;.\!\longleftrightarrow\!.\; q \Longrightarrow A \vdash p \;.\!\longrightarrow\!.\; q$›
 and *equivE2-PL*: ‹$A \vdash p \;.\!\longleftrightarrow\!.\; q \Longrightarrow A \vdash q \;.\!\longrightarrow\!.\; p$›

— non intuitionistic rules
 and *absurd-PL*: ‹$A \vdash .\neg\ (.\neg\ p) \Longrightarrow A \vdash p$›
 and *ex-mid-PL*: ‹$A \vdash p \;.\vee.\; (.\neg\ p)$›

begin

In the following, we will first retrieve the classical logic operators semantics coming from previous introduction and elimination rules

lemma *non-consistency*:
 assumes ‹$A \vdash .\neg\ p$›
 and ‹$A \vdash p$›
 shows ‹$A \vdash q$›
 by (*metis assms*(*1*) *assms*(*2*) *false-PL mp-PL notE-PL singleton-iff transitivity2-L*)

— this direct result brings directly many remarkable properties of implication (i.e. transitivity)
lemma *imp-PL*: ‹A ⊢ p .⟶. q ⟷ A ∪ {p} ⊢ q›
  apply (*intro iffI impI-PL*)
   apply(*rule mp-PL*[**where** *p=p*], *meson UnI1 assumption-L transitivity2-L*)
  using *assumption-L* **by** *auto*

lemma *not-PL*: ‹A ⊢ .¬ p ⟷ A ∪ {p} ⊢ ⊥›
  using *notE-PL notI-PL imp-PL* **by** *blast*

— Classical logic result
lemma *notnot-PL*: ‹A ⊢ .¬ (.¬ p) ⟷ A ⊢ p›
  apply(*rule iffI*, *simp add*:*absurd-PL*)
  by (*meson mp-PL notE-PL Un-upper1 Un-upper2 assumption-L infer-def monotonicity-L not-PL singletonI subsetD*)

lemma *conj-PL*: ‹A ⊢ p .∧. q ⟷ (A ⊢ p ∧ A ⊢ q)›
  using *conjE1-PL conjE2-PL conjI-PL* **by** *blast*

lemma *disj-PL*: ‹A ⊢ p .∨. q ⟷ A ∪ {.¬ p} ⊢ q›
proof
  assume *a*:‹A ⊢ p .∨. q›
  have *b*:‹A ⊢ p .⟶. (.¬ p .⟶. q)›
    by (*intro impI-PL*) (*meson Un-iff assumption-L insertI1 non-consistency*)
  have *c*:‹A ⊢ q .⟶. (.¬ p .⟶. q)›
    by (*simp add*: *assumption-L impI-PL*)
  from *a b c* **have** ‹A ⊢ .¬ p .⟶. q›
    by (*erule-tac disjE-PL*) *simp-all*
  then show ‹A ∪ {.¬ p} ⊢ q›
    using *imp-PL* **by** *blast*
next
  assume *a*:‹A ∪ {.¬ p} ⊢ q›
  hence *b*:‹A ⊢ .¬ p .⟶. q› **by** (*simp add*: *impI-PL*)
  then show ‹A ⊢ p .∨. q›
    apply(*rule-tac disjE-PL*[*OF ex-mid-PL*, *of A p* ‹p .∨. q›])
    by (*auto simp add*: *assumption-L disjI2-PL disjI1-PL impI-PL imp-PL*)
qed

lemma *equiv-PL*:‹A ⊢ p .⟷. q ⟷ (A ∪ {p} ⊢ q ∧ A ∪ {q} ⊢ p)›
  using *imp-PL equivE1-PL equivE2-PL equivI-PL* **by** *blast*

corollary *valid-imp-PL*: ‹⊩ (p .⟶. q) = ({p} ⊢ q)›
    and *valid-not-PL*: ‹⊩ (.¬ p) = ({p} ⊢ ⊥)›
    and *valid-conj-PL*: ‹⊩ (p .∧. q) = (⊩ p ∧ ⊩ q)›
    and *valid-disj-PL*: ‹⊩ (p .∨. q) = ({.¬ p} ⊢ q)›
    and *valid-equiv-PL*:‹⊩ (p .⟷. q) = ({p} ⊢ q ∧ {q} ⊢ p)›
  using *imp-PL not-PL conj-PL disj-PL equiv-PL valid-def* **by** *auto*

Second, we will combine each logical operator with the consequence operator

Cn: it is a trick to profit from set theory to get many essential lemmas without complex inferences

declare *infer-def*[*simp*]

lemma *nonemptyCn*: ‹$Cn(A) \neq \{\}$›
  using *true-PL* **by** *auto*

lemma *Cn-true*: ‹$Cn(\{\top\}) = Cn(\{\})$›
  using *Cn-same true-PL* **by** *auto*

lemma *Cn-false*: ‹$Cn(\{\bot\}) = UNIV$›
  using *false-PL* **by** *auto*

lemma *Cn-imp*: ‹$A \vdash (p \mathbin{.{\longrightarrow}.} q) \longleftrightarrow Cn(\{q\}) \subseteq Cn(A \cup \{p\})$›
  and *Cn-imp-bis*: ‹$A \vdash (p \mathbin{.{\longrightarrow}.} q) \longleftrightarrow Cn(A \cup \{q\}) \subseteq Cn(A \cup \{p\})$›
  using *Cn-same imp-PL idempotency-L inclusion-L infer-def subset-insertI* **by** *force+*

lemma *Cn-not*: ‹$A \vdash .\neg\ p \longleftrightarrow Cn(A \cup \{p\}) = UNIV$›
  using *Cn-false Cn-imp notE-PL not-PL* **by** *fastforce*

lemma *Cn-conj*: ‹$A \vdash (p \mathbin{.{\wedge}.} q) \longleftrightarrow Cn(\{p\}) \cup Cn(\{q\}) \subseteq Cn(A)$›
  apply(*intro iffI conjI-PL*, *frule conjE1-PL*, *frule conjE2-PL*)
  using *Cn-same Un-insert-right bot.extremum idempotency-L inclusion-L* **by** *auto*

lemma *Cn-conj-bis*: ‹$Cn(\{p \mathbin{.{\wedge}.} q\}) = Cn(\{p, q\})$›
  by (*unfold Cn-same*)
    (*meson Supraclassical-logic.conj-PL Supraclassical-logic-axioms insert-subset*)

lemma *Cn-disj*: ‹$A \vdash (p \mathbin{.{\vee}.} q) \longleftrightarrow Cn(\{q\}) \subseteq Cn(A \cup \{.\neg\ p\})$›
  and *Cn-disj-bis*: ‹$A \vdash (p \mathbin{.{\vee}.} q) \longleftrightarrow Cn(A \cup \{q\}) \subseteq Cn(A \cup \{.\neg\ p\})$›
  using *disj-PL Cn-same imp-PL idempotency-L inclusion-L infer-def subset-insertI* **by** *force+*

lemma *Cn-equiv*: ‹$A \vdash (p \mathbin{.{\longleftrightarrow}.} q) \longleftrightarrow Cn(A \cup \{p\}) = Cn(A \cup \{q\})$›
  by (*metis Cn-imp-bis equivE1-PL equivE2-PL equivI-PL set-eq-subset*)

corollary *valid-nonemptyCn*: ‹$Cn(\{\}) \neq \{\}$›
    and *valid-Cn-imp*: ‹$\Vdash (p \mathbin{.{\longrightarrow}.} q) \longleftrightarrow Cn(\{q\}) \subseteq Cn(\{p\})$›
    and *valid-Cn-not*: ‹$\Vdash (.\neg\ p) \longleftrightarrow Cn(\{p\}) = UNIV$›
    and *valid-Cn-conj*: ‹$\Vdash (p \mathbin{.{\wedge}.} q) \longleftrightarrow Cn(\{p\}) \cup Cn(\{q\}) \subseteq Cn(\{\})$›
    and *valid-Cn-disj*: ‹$\Vdash (p \mathbin{.{\vee}.} q) \longleftrightarrow Cn(\{q\}) \subseteq Cn(\{.\neg\ p\})$›
    and *valid-Cn-equiv*: ‹$\Vdash (p \mathbin{.{\longleftrightarrow}.} q) \longleftrightarrow Cn(\{p\}) = Cn(\{q\})$›
  using *nonemptyCn Cn-imp Cn-not Cn-conj Cn-disj Cn-equiv valid-def* **by** *auto*

— Finally, we group additional lemmas that were essential in further proofs
lemma *consistency*: ‹$Cn(\{p\}) \cap Cn(\{.\neg\ p\}) = Cn(\{\})$›
proof
  { **fix** *q*

```
    assume ‹{p} ⊢ q› and ‹{.¬ p} ⊢ q›
    hence {} ⊢ p .⟶. q and {} ⊢ (.¬ p) .⟶. q
      using impI-PL by auto
    hence ‹{} ⊢ q›
      using ex-mid-PL by (rule-tac disjE-PL[where p=p and q=‹.¬ p›]) blast
  }
  then show ‹Cn({p}) ∩ Cn({.¬ p}) ⊆ Cn({})› by (simp add: subset-iff)
next
  show ‹Cn({}) ⊆ Cn({p}) ∩ Cn({.¬ p})› by (simp add: monotonicity-L)
qed

lemma Cn-notnot: ‹Cn({.¬ (.¬ φ)}) = Cn({φ})›
  by (metis (no-types, opaque-lifting) notnot-PL valid-Cn-equiv valid-equiv-PL)

lemma conj-com: ‹A ⊢ p .∧. q ⟷ A ⊢ q .∧. p›
  using conj-PL by auto

lemma conj-com-Cn: ‹Cn({p .∧. q}) = Cn({q .∧. p})›
  by (simp add: Cn-conj-bis insert-commute)

lemma disj-com: ‹A ⊢ p .∨. q ⟷ A ⊢ q .∨. p›
proof –
  { fix p q
    have ‹A ⊢ p .∨. q ⟹ A ⊢ q .∨. p›
      apply(erule disjE-PL)
      using assumption-L disjI2-PL disjI1-PL impI-PL by auto
  }
  then show ?thesis by auto
qed

lemma disj-com-Cn: ‹Cn({p .∨. q}) = Cn({q .∨. p})›
  unfolding Cn-same using disj-com by simp

lemma imp-contrapos: ‹A ⊢ p .⟶. q ⟷ A ⊢ .¬ q .⟶. .¬ p›
  by (metis Cn-not Un-insert-left Un-insert-right imp-PL notnot-PL)

lemma equiv-negation: ‹A ⊢ p .⟷. q ⟷ A ⊢ .¬ p .⟷. .¬ q›
  using equivE1-PL equivE2-PL equivI-PL imp-contrapos by blast

lemma imp-trans: ‹A ⊢ p .⟶.q ⟹ A ⊢ q .⟶.r ⟹ A ⊢ p .⟶.r›
  using Cn-imp-bis by auto

lemma imp-recovery0: ‹A ⊢ p .∨. (p .⟶. q)›
  apply(subst disj-PL, subst imp-contrapos)
  using assumption-L impI-PL by auto

lemma imp-recovery1: ‹A ∪ {p .⟶. q} ⊢ p ⟹ A ⊢ p›
  using disjE-PL[OF imp-recovery0, of A p p q] assumption-L imp-PL by auto
```

lemma *imp-recovery2*: ‹$A \vdash p$.⟶. $q \Longrightarrow A \vdash (q$.⟶. $p)$.⟶. $p \Longrightarrow A \vdash q$›
using *imp-PL imp-recovery1 imp-trans* **by** *blast*

lemma *impI2*: ‹$A \vdash q \Longrightarrow A \vdash p$.⟶. q›
by (*meson assumption-L impI-PL in-mono sup-ge1 transitivity2-L*)

lemma *conj-equiv*: ‹$A \vdash p \Longrightarrow A \vdash ((p$.∧. $q)$.⟷. $q)$›
by (*metis Un-insert-right assumption-L conjE2-PL conjI-PL equiv-PL impI2 imp-PL insertI1 sup-bot.right-neutral*)

lemma *conj-imp*: ‹$A \vdash (p$.∧. $q)$.⟶. $r \longleftrightarrow A \vdash p$.⟶. $(q$.⟶. $r)$›
proof
assume $A \vdash (p$.∧. $q)$.⟶. r
then have $Cn\ (A \cup \{r\}) \subseteq Cn\ (A \cup \{p, q\})$
by (*metis* (*no-types*) *Cn-conj-bis Cn-imp-bis Cn-union Un-insert-right sup-bot.right-neutral*)
then show ‹$A \vdash p$.⟶. $(q$.⟶. $r)$›
by (*metis Un-insert-right impI-PL inclusion-L infer-def insert-commute insert-subset subset-eq sup-bot.right-neutral*)
next
assume $A \vdash p$.⟶. $(q$.⟶. $r)$
then have $A \cup \{p\} \cup \{q\} \vdash r$
using *imp-PL* **by** *auto*
then show $A \vdash (p$.∧. $q)$.⟶. r
by (*metis* (*full-types*) *Cn-conj-bis Cn-union impI-PL infer-def insert-is-Un sup-assoc*)
qed

lemma *conj-not-impE-PL*: ‹$A \vdash (p$.∧. $q)$.⟶. $r \Longrightarrow A \vdash (p$.∧. .¬ $q)$.⟶. $r \Longrightarrow A \vdash p$.⟶. r›
by (*meson conj-imp disjE-PL ex-mid-PL imp-PL*)

lemma *disj-notE-PL*: ‹$A \vdash q \Longrightarrow A \vdash p$.∨. .¬ $q \Longrightarrow A \vdash p$›
using *Cn-imp Cn-imp-bis Cn-not disjE-PL notnot-PL* **by** *blast*

lemma *disj-not-impE-PL*: ‹$A \vdash (p$.∨. $q)$.⟶. $r \Longrightarrow A \vdash (p$.∨. .¬ $q)$.⟶. $r \Longrightarrow A \vdash r$›
by (*metis Un-insert-right disjE-PL disj-PL disj-com ex-mid-PL insert-commute sup-bot.right-neutral*)

lemma *imp-conj*: ‹$A \vdash p$.⟶. $q \Longrightarrow A \vdash r$.⟶. $s \Longrightarrow A \vdash (p$.∧. $r)$.⟶. $(q$.∧. $s)$›
apply(*intro impI-PL conjI-PL*, *unfold imp-PL*[*symmetric*])
by (*meson assumption-L conjE1-PL conjE2-PL imp-trans infer-def insertI1 validD-L valid-imp-PL*)+

lemma *conj-overlap*: ‹$A \vdash (p$.∧. $q) \longleftrightarrow A \vdash (p$.∧. $((.\neg\ p)$.∨. $q))$›
by (*meson conj-PL disjI2-PL disj-com disj-notE-PL*)

lemma *morgan*: ‹$A \vdash$.¬ $(p$.∧. $q) \longleftrightarrow A \vdash (.\neg\ p)$.∨. $(.\neg\ q)$›

by (*meson conj-imp disj-PL disj-com imp-PL imp-contrapos notE-PL notI-PL*)

lemma *conj-superexpansion1*: ‹$A \vdash .\neg\ (p\ .\wedge.\ q)\ .\wedge.\ .\neg\ p \longleftrightarrow A \vdash .\neg\ p$›
using *conj-PL disjI1-PL morgan* **by** *auto*

lemma *conj-superexpansion2*: ‹$A \vdash (p\ .\vee.\ q)\ .\wedge.\ p \longleftrightarrow A \vdash p$›
using *conj-PL disjI1-PL* **by** *auto*

end

2.3 Compact Logic

If the logic is compact, which means that any result is based on a finite set of hypothesis

locale *Compact-logic* = *Tarskian-logic* +
assumes *compactness-L*: ‹$A \vdash \varphi \Longrightarrow (\exists A'.\ A' \subseteq A \wedge$ *finite* $A' \wedge A' \vdash \varphi)$›

begin

Very important lemma preparing the application of the Zorn's lemma. It states that in a compact logic, we can not deduce φ if we accumulate an infinity of hypothesis groups which locally do not deduce phi

lemma *chain-closure*: ‹$\neg \Vdash \varphi \Longrightarrow$ *subset.chain* $\{B.\ \neg\ B \vdash \varphi\}\ C \Longrightarrow \neg \bigcup C \vdash \varphi$›
proof
assume *a*:‹*subset.chain* $\{B.\ \neg\ B \vdash \varphi\}\ C$› **and** *b*:‹$\neg \Vdash \varphi$› **and** ‹$\bigcup\ C \vdash \varphi$›
then obtain A' **where** *c*:‹$A' \subseteq \bigcup\ C$› **and** *d*:‹*finite* A'› **and** *e*:‹$A' \vdash \varphi$› **using** *compactness-L* **by** *blast*
define *f* **where** *f*:‹$f \equiv \lambda a.$ *SOME* $B.\ B \in C \wedge a \in B$›
have *g*:‹*finite* $(f\ `\ A')$› **using** *f d* **by** *simp*
have *h*:‹$(f\ `\ A') \subseteq C$›
unfolding *f* **by** *auto* (*metis* (*mono-tags*, *lifting*) *Union-iff c someI-ex subset-eq*)
have *i*:‹*subset.chain* $\{B.\ \neg\ B \vdash \varphi\}\ (f\ `\ A')$› **using** *a h*
using *a h* **by** (*meson subsetD subset-chain-def subset-trans*)
have ‹$A' \neq \{\} \Longrightarrow \bigcup\ (f\ `\ A') \in \{B.\ \neg\ B \vdash \varphi\}$› **using** *g i*
by (*meson Union-in-chain image-is-empty subset-chain-def subset-eq*)
hence *j*:‹$A' \neq \{\} \Longrightarrow \neg \bigcup (f\ `\ A') \vdash \varphi$› **by** *simp*
have ‹$A' \subseteq \bigcup (f\ `\ A')$›
unfolding *f* **by** *auto* (*metis* (*no-types*, *lifting*) *Union-iff c someI-ex subset-iff*)
with *j e b* **show** *False*
by (*metis infer-def monotonicity-L subsetD valid-def*)
qed

end

end

3 Remainders

In AGM, one important feature is to eliminate some proposition from a set of propositions by ensuring that the set of retained clauses is maximal and that nothing among these clauses allows to retrieve the eliminated proposition

3.1 Remainders in a Tarskian logic

In a general context of a Tarskian logic, we consider a descriptive definition (by comprehension)

context *Tarskian-logic*

begin
definition *remainder*::‹$'a\ set \Rightarrow 'a \Rightarrow 'a\ set\ set$› (**infix** ‹$.\bot.$› *55*)
  where *rem*: ‹$A\ .\bot.\ \varphi \equiv \{B.\ B \subseteq A \wedge \neg\ B \vdash \varphi \wedge (\forall B' \subseteq A.\ B \subset B' \longrightarrow B' \vdash \varphi)\}$›

lemma *rem-inclusion*: ‹$B \in A\ .\bot.\ \varphi \Longrightarrow B \subseteq A$›
  by (*auto simp add*:*rem split*:*if-splits*)

lemma *rem-closure*: $K = Cn(A) \Longrightarrow B \in K\ .\bot.\ \varphi \Longrightarrow B = Cn(B)$
  apply(*cases* ‹$K\ .\bot.\ \varphi = \{\}$›, *simp*)
  by (*simp add*:*rem infer-def*) (*metis idempotency-L inclusion-L monotonicity-L psubsetI*)

lemma *remainder-extensionality*: ‹$Cn(\{\varphi\}) = Cn(\{\psi\}) \Longrightarrow A\ .\bot.\ \varphi = A\ .\bot.\ \psi$›
  unfolding *rem infer-def* **apply** *safe*
  by (*simp-all add*: *Cn-same*) *blast+*

lemma *nonconsequence-remainder*: ‹$A\ .\bot.\ \varphi = \{A\} \longleftrightarrow \neg\ A \vdash \varphi$›
  unfolding *rem* **by** *auto*

— As we will see further, the other direction requires compactness!
lemma *taut2emptyrem*: ‹$\Vdash \varphi \Longrightarrow A\ .\bot.\ \varphi = \{\}$›
  unfolding *rem* **by** (*simp add*: *infer-def validD-L*)

end

3.2 Remainders in a supraclassical logic

In case of a supraclassical logic, remainders get impressive properties

context *Supraclassical-logic*

begin

— As an effect of being maximal, a remainder keeps the eliminated proposition in its propositions hypothesis

lemma *remainder-recovery*: ‹$K = Cn(A) \Longrightarrow K \vdash \psi \Longrightarrow B \in K .\bot. \varphi \Longrightarrow B \vdash \varphi .\longrightarrow. \psi$›
proof –
  { fix ψ **and** B
    assume a:‹$K = Cn(A)$› **and** c:‹$\psi \in K$› **and** d:‹$B \in K .\bot. \varphi$› **and** e:‹$\varphi .\longrightarrow. \psi \notin Cn(B)$›
    with a **have** f:‹$\varphi .\longrightarrow. \psi \in K$› **using** *impI2 infer-def* **by** *blast*
    with d e **have** ‹$\varphi \in Cn(B \cup \{\varphi .\longrightarrow. \psi\})$›
      apply (*simp add*:*rem*, *elim conjE*)
      by (*metis dual-order.order-iff-strict inclusion-L insert-subset*)
    with d **have** *False* **using** *rem imp-recovery1*
      by (*metis* (*no-types*, *lifting*) *CollectD infer-def*)
  }
  thus ‹$K = Cn(A) \Longrightarrow K \vdash \psi \Longrightarrow B \in K .\bot. \varphi \Longrightarrow B \vdash \varphi .\longrightarrow. \psi$›
    using *idempotency-L* **by** *auto*
qed

— When you remove some proposition φ several other propositions can be lost. An important lemma states that the resulting remainder is also a remainder of any lost proposition
lemma *remainder-recovery-bis*: ‹$K = Cn(A) \Longrightarrow K \vdash \psi \Longrightarrow \neg\ B \vdash \psi \Longrightarrow B \in K .\bot. \varphi \Longrightarrow B \in K .\bot. \psi$›
proof–
  assume a:‹$K = Cn(A)$› **and** b:‹$\neg\ B \vdash \psi$› **and** c:‹$B \in K .\bot. \varphi$› **and** d:‹$K \vdash \psi$›
  hence d:‹$B \vdash \varphi .\longrightarrow. \psi$› **using** *remainder-recovery* **by** *simp*
  with c **show** ‹$B \in K .\bot. \psi$›
    by (*simp add*:*rem*) (*meson b dual-order.trans infer-def insert-subset monotonicity-L mp-PL order-refl psubset-imp-subset*)
qed

corollary *remainder-recovery-imp*: ‹$K = Cn(A) \Longrightarrow K \vdash \psi \Longrightarrow \Vdash (\psi .\longrightarrow. \varphi) \Longrightarrow B \in K .\bot. \varphi \Longrightarrow B \in K .\bot. \psi$›
  apply(*rule remainder-recovery-bis*, *simp-all*)
  by (*simp add*:*rem*) (*meson infer-def mp-PL validD-L*)

— If we integrate back the eliminated proposition into the remainder, we retrieve the original set!
lemma *remainder-expansion*: ‹$K = Cn(A) \Longrightarrow K \vdash \psi \Longrightarrow \neg\ B \vdash \psi \Longrightarrow B \in K .\bot. \varphi \Longrightarrow B \oplus \psi = K$›
proof
  assume a:‹$K = Cn(A)$› **and** b:‹$K \vdash \psi$› **and** c:‹$\neg\ B \vdash \psi$› **and** d:‹$B \in K .\bot. \varphi$›
  then show ‹$B \oplus \psi \subseteq K$›
    by (*metis Un-insert-right expansion-def idempotency-L infer-def insert-subset monotonicity-L rem-inclusion sup-bot.right-neutral*)
next
  assume a:‹$K = Cn(A)$› **and** b:‹$K \vdash \psi$› **and** c:‹$\neg\ B \vdash \psi$› **and** d:‹$B \in K .\bot. \varphi$›
  { fix χ
    assume ‹$\chi \in K$›
   hence e:‹$B \vdash \varphi .\longrightarrow. \chi$› **using** *remainder-recovery*[*OF a - d*, *of* χ] *assumption-L*

by *blast*
  have ‹$\psi \in K$› **using** *a b idempotency-L infer-def* **by** *blast*
  hence *f*:‹$B \cup \{\psi\} \vdash \varphi$› **using** *b c d* **apply**(*simp add*:*rem*)
    by (*meson inclusion-L insert-iff insert-subsetI less-le-not-le subset-iff*)
  from *e f* **have** ‹$B \cup \{\psi\} \vdash \chi$› **using** *imp-PL imp-trans* **by** *blast*
 }
 then show ‹$K \subseteq B \oplus \psi$›
  by (*simp add*: *expansion-def subsetI*)
qed

To eliminate a conjunction, we only need to remove one side

lemma *remainder-conj*: ‹$K = Cn(A) \Longrightarrow K \vdash \varphi$.∧. $\psi \Longrightarrow K$.⊥. (φ .∧. ψ) = (K .⊥. φ) ∪ (K .⊥. ψ)›
 apply(*intro subset-antisym Un-least subsetI*, *simp add*:*rem*)
  apply (*meson conj-PL infer-def*)
  using *remainder-recovery-imp*[*of K A* ‹φ .∧. ψ› φ]
  apply (*meson assumption-L conjE1-PL singletonI subsetI valid-imp-PL*)
 using *remainder-recovery-imp*[*of K A* ‹φ .∧. ψ› ψ]
 by (*meson assumption-L conjE2-PL singletonI subsetI valid-imp-PL*)

end

3.3 Remainders in a compact logic

In case of a supraclassical logic, remainders get impressive properties

context *Compact-logic*
begin

The following lemma is the Lindembaum's lemma requiring the Zorn's lemma (already available in standard Isabelle/HOL). For more details, please refer to the book "Theory of logical calculi" [5]. This very important lemma states that we can get a maximal set (remainder B') starting from any set B if this latter does not infer the proposition φ we want to eliminate

lemma *upper-remainder*: ‹$B \subseteq A \Longrightarrow \neg\ B \vdash \varphi \Longrightarrow \exists B'.\ B \subseteq B' \land\ B' \in A$.⊥. φ›
proof –
 assume *a*:‹$B \subseteq A$› **and** *b*:‹$\neg\ B \vdash \varphi$›
 have *c*:‹$\neg \Vdash \varphi$›
  using *b infer-def validD-L* **by** *blast*
 define $\mathcal{B}$ **where** $\mathcal{B} \equiv \{B'.\ B \subseteq B' \land B' \subseteq A \land \neg\ B' \vdash \varphi\}$
 have *d*:‹*subset.chain* $\mathcal{B}$ $C \Longrightarrow$ *subset.chain* $\{B.\ \neg\ B \vdash \varphi\}$ C› **for** C
  unfolding $\mathcal{B}$*-def*
  by (*simp add*: *le-fun-def less-eq-set-def subset-chain-def*)
 have *e*:‹$C \neq \{\} \Longrightarrow$ *subset.chain* $\mathcal{B}$ $C \Longrightarrow B \subseteq \bigcup\ C$› **for** C
  by (*metis* (*no-types*, *lifting*) $\mathcal{B}$*-def subset-chain-def less-eq-Sup mem-Collect-eq subset-iff*)
 { fix C
  assume *f*:‹$C \neq \{\}$› **and** *g*:‹*subset.chain* $\mathcal{B}$ C›

have ‹$\bigcup$ $C \in \mathcal{B}$›
using $\mathcal{B}$*-def* *e*[*OF f g*] *chain-closure*[*OF c d*[*OF g*]]
by *simp* (*metis* (*no-types*, *lifting*) *CollectD Sup-least Sup-subset-mono g subset.chain-def subset-trans*)
} **note** *f=this*
have ‹*subset.chain* $\mathcal{B}$ $C \Longrightarrow \exists\, U{\in}\mathcal{B}.\ \forall\, X{\in}C.\ X \subseteq U$› **for** C
apply (*cases* ‹$C \neq \{\}$›)
apply (*meson Union-upper f*)
using $\mathcal{B}$*-def a b* **by** *blast*
with *subset-Zorn*[*OF this*, *simplified*] **obtain** B' **where** *f*:‹$B'{\in}\ \mathcal{B} \wedge (\forall\, X{\in}\mathcal{B}.\ B' \subseteq X \longrightarrow X = B')$› **by** *auto*
then show *?thesis*
by (*simp add*:*rem* $\mathcal{B}$*-def*, *rule-tac* $x{=}B'$ **in** *exI*) (*metis psubsetE subset-trans*)
qed

— An immediate corollary ruling tautologies
corollary *emptyrem2taut*: ‹$A\ .\!\perp\!.\ \varphi = \{\} \Longrightarrow\ \Vdash \varphi$›
by (*metis bot.extremum empty-iff upper-remainder valid-def*)

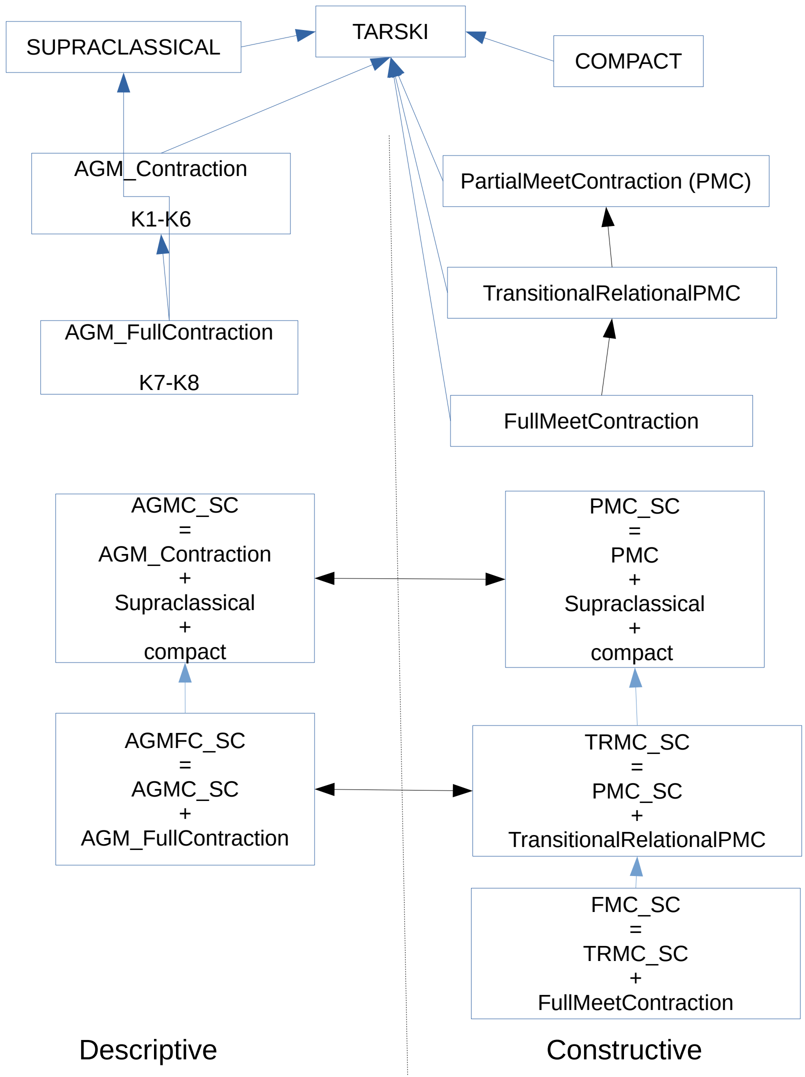
end

end

4 Contractions

The first operator of belief change of the AGM framework is contraction. This operator consist to remove a sentence φ from a belief set K in such a way that K no longer imply φ.

In the following we will first axiomatize such operators at different levels of logics (Tarskian, supraclassical and compact) and then we will give constructions satisfying these axioms. The following graph summarizes all equivalences we established:



We will use the extension feature of locales in Isabelle/HOL to incrementally define the contraction operator as shown by blue arrows in the previous figure. Then, using the interpretation feature of locales, we will prove the equivalence between descriptive and constructive approaches at each level depending on the adopted logics (black arrows).

4.1 AGM contraction postulates

The operator of contraction is denoted by the symbol $\div$ and respects the six following conditions :

- *contract-closure* : a belief set K contracted by φ should be logically closed
- *contract-inclusion* : a contracted set K should be a subset of the original one
- *contract-vacuity* : if φ is not included in a set K then the contraction of K by φ involves no change at all
- *contract-success* : if a set K is contracted by φ then K does not imply φ
- *contract-recovery*: all propositions removed in a set K by contraction of φ will be recovered by expansion of φ
- *contract-extensionality* : Extensionality guarantees that the logic of contraction is extensional in the sense of allowing logically quivalent sentences to be freely substituted for each other

locale *AGM-Contraction* = *Tarskian-logic* +
fixes *contraction*::‹*'a set* ⇒ *'a* ⇒ *'a set*› (**infix** ‹÷› *55*)
assumes *contract-closure*: ‹$K = Cn(A) \Longrightarrow K \div \varphi = Cn(K \div \varphi)$›
 and *contract-inclusion*: ‹$K = Cn(A) \Longrightarrow K \div \varphi \subseteq K$›
 and *contract-vacuity*: ‹$K = Cn(A) \Longrightarrow \varphi \notin K \Longrightarrow K \div \varphi = K$›
 and *contract-success*: ‹$K = Cn(A) \Longrightarrow \varphi \notin Cn(\{\}) \Longrightarrow \varphi \notin K \div \varphi$›
 and *contract-recovery*: ‹$K = Cn(A) \Longrightarrow K \subseteq ((K \div \varphi) \oplus \varphi)$›
 and *contract-extensionality*: ‹$K = Cn(A) \Longrightarrow Cn(\{\varphi\}) = Cn(\{\psi\}) \Longrightarrow K \div \varphi = K \div \psi$›

A full contraction is defined by two more postulates to rule the conjunction. We base on a supraclassical logic.

- *contract-conj-overlap* : An element in both $K \div \varphi$ and $K \div \psi$ is also an element of $K \div (\varphi \wedge \psi)$
- *contract-conj-inclusion* : If φ not in $K \div (\varphi \wedge \psi)$ then all elements removed by this contraction are also removed from $K \div \varphi$

locale *AGM-FullContraction* = *AGM-Contraction* + *Supraclassical-logic* +
 assumes *contract-conj-overlap*: ‹$K = Cn(A) \Longrightarrow (K \div \varphi) \cap (K \div \psi) \subseteq (K \div (\varphi \mathbin{.\wedge.} \psi))$›
 and *contract-conj-inclusion*: ‹$K = Cn(A) \Longrightarrow \varphi \notin (K \div (\varphi \mathbin{.\wedge.} \psi)) \Longrightarrow ((K \div (\varphi \mathbin{.\wedge.} \psi) \subseteq (K \div \varphi)))$›

begin
— two important lemmas/corollaries that can replace the two assumptions *contract-conj-overlap* and *contract-conj-inclusion*

contract-conj-overlap-variant does not need ψ to occur in the left side!

corollary *contract-conj-overlap-variant*: ‹$K = Cn(A) \Longrightarrow (K \div \varphi) \cap Cn(\{\varphi\}) \subseteq (K \div (\varphi .\wedge. \psi))$›
proof –
  assume *a*:‹$K = Cn(A)$›
  { assume *b*:‹$K \vdash \varphi$› **and** *c*:‹$K \vdash \psi$›
    hence *d*:‹$K \div (\varphi .\wedge. \psi) = K \div (\varphi .\wedge. ((.\neg \varphi) .\vee. \psi))$›
      apply(*rule-tac contract-extensionality*[*OF a*])
      using *conj-overlap*[*of - φ ψ*] **by** (*simp add*: *Cn-same*)
    have *e*:‹$K \cap Cn\ \{\varphi\} \subseteq K \div (.\neg\ \varphi .\vee. \psi)$›
    proof(*safe*)
      fix χ
      assume *f*:‹$\chi \in K$› **and** *g*:‹$\chi \in Cn\ \{\varphi\}$›
      have ‹$K \div (.\neg\ \varphi .\vee. \psi) \vdash (.\neg\ \varphi .\vee. \psi) .\longrightarrow. \chi$›
        by (*metis a contract-recovery expansion-def f impI-PL infer-def subset-eq*)
      hence ‹$K \div (.\neg\ \varphi .\vee. \psi) \vdash .\neg\ \varphi .\longrightarrow. \chi$›
        by (*meson disjI1-PL imp-trans inclusion-L infer-def insert-subset validD-L valid-imp-PL*)
      with *g* **show** ‹$\chi \in K \div (.\neg\ \varphi .\vee. \psi)$›
      by (*metis a contract-closure disjE-PL ex-mid-PL infer-def validD-L valid-imp-PL*)
    qed
    have *?thesis*
      unfolding *d* **using** *e contract-conj-overlap*[*OF a*, *of* φ ‹$(.\neg\ \varphi .\vee. \psi)$›] *a contract-inclusion* **by** *force*
  }
  then show *?thesis*
    apply (*cases* ‹$\neg\ K \vdash \varphi \vee \neg\ K \vdash \psi$›)
    by (*metis IntE a assumption-L conjE1-PL conjE2-PL contract-inclusion contract-vacuity subsetD subsetI*) *blast*
qed

contract-conj-inclusion-variant: Everything retained in $K \div (\varphi \wedge \psi)$ is retained in $K \div \psi$

corollary *contract-conj-inclusion-variant* : ‹$K = Cn(A) \Longrightarrow (K \div (\varphi .\wedge. \psi) \subseteq (K \div \varphi)) \vee (K \div (\varphi .\wedge. \psi) \subseteq (K \div \psi))$›
proof –
  assume *a*:‹$K = Cn(A)$›
  { assume *d*:‹$\varphi \in (K \div (\varphi .\wedge. \psi)) \wedge \psi \in (K \div (\varphi .\wedge. \psi))$›
    hence ‹$\varphi .\wedge. \psi \in (K \div (\varphi .\wedge. \psi))$›
    using *Supraclassical-logic.conjI-PL Supraclassical-logic-axioms a contract-closure* **by** *fastforce*
    with *d* **have** *?thesis*
      by (*metis* (*no-types*, *lifting*) *Supraclassical-logic.valid-conj-PL Supraclassical-logic-axioms*

Tarskian-logic.valid-expansion Tarskian-logic-axioms a contract-closure contract-inclusion
contract-recovery contract-success dual-order.trans expansion-def)
}
then show *?thesis*
by (*metis a conj-com-Cn contract-conj-inclusion contract-extensionality*)
qed

end

4.2 Partial meet contraction definition

A partial meet contraction of K by φ is the intersection of some sets that not imply φ. We define these sets as the "remainders" (K .⊥. φ. The function of selection γ select the best set of the remainders that do not imply φ. This function respect these postulates :

- *is-selection* : if there exist some set that do not imply φ then the function selection γ is a subset of these sets
- *tautology-selection* : if there is no set that do not imply φ then the result of the selection function is K
- *nonempty-selection* : An empty selection function do not exist
- *extensional-selection* : Two proposition with the same closure have the same selection function

locale *PartialMeetContraction* = *Tarskian-logic* +

fixes *selection*::‹$'a$ *set* ⇒ $'a$ ⇒ $'a$ *set set*› (‹γ›)
assumes *is-selection*: ‹$K = Cn(A) \Longrightarrow (K\ .\bot.\ \varphi) \neq \{\} \Longrightarrow \gamma\ K\ \varphi \subseteq (K\ .\bot.\ \varphi)$›
assumes *tautology-selection*: ‹$K = Cn(A) \Longrightarrow (K\ .\bot.\ \varphi) = \{\} \Longrightarrow \gamma\ K\ \varphi = \{K\}$›
assumes *nonempty-selection*: ‹$K = Cn(A) \Longrightarrow \gamma\ K\ \varphi \neq \{\}$›
assumes *extensional-selection*: ‹$K = Cn(A) \Longrightarrow Cn(\{\varphi\}) = Cn(\{\psi\}) \Longrightarrow \gamma\ K\ \varphi = \gamma\ K\ \psi$›

— extensionality seems very hard to implement for a constructive approach, one basic implementation will be to ignore A and φ and only base on A .⊥. φ that is already proved as extensional (lemma *remainder-extensionality*)

begin

A partial meet is the intersection of set of selected element.

definition (**in** *Tarskian-logic*) *meet-contraction*::‹$'a$ *set* ⇒ ($'a$ *set* ⇒ $'a$ ⇒ $'a$ *set set*) ⇒ $'a$ ⇒ $'a$ *set*› (‹- ÷₋ -› [*60,50,60*]*55*)

where *mc*: ‹$(A \div_\gamma \varphi) \equiv \bigcap(\gamma\ A\ \varphi)$›

Following this definition 4 postulates of AGM can be proved on a partial meet contraction:

- *contract-inclusion*
- *contract-vacuity*
- *contract-closure*
- *contract-extensionality*

pmc-inclusion :a partial meet contraction is a subset of the contracted set

lemma *pmc-inclusion*: ‹$K = Cn(A) \Longrightarrow K \div_\gamma \varphi \subseteq K$›
apply (*cases* ‹$(K\ .\bot.\ \varphi) = \{\}$›, *simp-all add*: *mc tautology-selection*)
by (*meson Inf-less-eq in-mono is-selection nonempty-selection rem-inclusion*)

pmc-vacuity : if φ is not included in a set K then the partial meet contraction of K by φ involves not change at all

lemma *pmc-vacuity*: ‹$K = Cn(A) \Longrightarrow \neg\ K \vdash \varphi \Longrightarrow K \div_\gamma \varphi = K$›
unfolding *mc nonconsequence-remainder*
by (*metis Inf-superset-mono Un-absorb1 cInf-singleton insert-not-empty is-selection mc nonconsequence-remainder pmc-inclusion sup-commute*)

pmc-closure : a partial meet contraction is logically closed

lemma *pmc-closure*: ‹$K = Cn(A) \Longrightarrow (K \div_\gamma \varphi) = Cn(K \div_\gamma \varphi)$›
proof (*rule subset-antisym*, *simp-all add*:*inclusion-L mc transitivity-L*, *goal-cases*)
case *1*
have ‹$\bigcap(\gamma\ (Cn\ A)\ \varphi) = \bigcap\{Cn(B)|B.\ B \in \gamma\ (Cn\ A)\ \varphi\}$›
by *auto* (*metis idempotency-L insert-absorb insert-iff insert-subset is-selection rem-closure tautology-selection*)+
from *Cn-Inter*[*OF this*] **show** *?case* **by** *blast*
qed

pmc-extensionality : Extensionality guarantees that the logic of contraction is extensional in the sense of allowing logically equivalent sentences to be freely substituted for each other

lemma *pmc-extensionality*: ‹$K = Cn(A) \Longrightarrow Cn(\{\varphi\}) = Cn(\{\psi\}) \Longrightarrow K \div_\gamma \varphi = K \div_\gamma \psi$›
by (*metis extensional-selection mc*)

pmc-tautology : if φ is a tautology then the partial meet contraction of K by φ is K

lemma *pmc-tautology*: ‹$K = Cn(A) \Longrightarrow \Vdash \varphi \Longrightarrow K \div_\gamma \varphi = K$›
by (*simp add*: *mc taut2emptyrem tautology-selection*)

completion is a an operator that can build an equivalent selection from an existing one

definition (**in** *Tarskian-logic*) *completion*::‹($'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ *set set*) $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ *set set*› (‹$*$›)
where ‹$* \ \gamma \ A \ \varphi \equiv$ *if* $(A \ .\bot. \ \varphi) = \{\}$ *then* $\{A\}$ *else* $\{B. \ B \in A \ .\bot. \ \varphi \wedge \bigcap \ (\gamma \ A \ \varphi) \subseteq B\}$›

lemma *selection-completion*: $K = Cn(A) \Longrightarrow \gamma \ K \ \varphi \subseteq * \ \gamma \ K \ \varphi$
using *completion-def is-selection tautology-selection* **by** *fastforce*

lemma (**in** *Tarskian-logic*) *completion-completion*: $K = Cn(A) \Longrightarrow * \ (* \ \gamma) \ K \ \varphi = * \ \gamma \ K \ \varphi$
by (*auto simp add*:*completion-def*)

lemma *pmc-completion*: ‹$K = Cn(A) \Longrightarrow K \div_{*\gamma} \varphi = K \div_{\gamma} \varphi$›
apply(*auto simp add*: *mc completion-def tautology-selection*)
by (*metis Inter-lower equals0D in-mono is-selection*)

end

A transitively relational meet contraction is a partial meet contraction using a binary relation between the elements of the selection function

A relation is :

- transitive (*trans-rel*)
- non empty (there is always an element preferred to the others (*nonempty-rel*))

A selection function γ_{TR} is transitively relational *rel-sel* with the following condition :

- If the the remainders $K \ .\bot. \ \varphi$ is empty then the selection function return K
- Else the selection function return a non empty transitive relation on the remainders

locale *TransitivelyRelationalMeetContraction* = *Tarskian-logic* +

fixes *relation*::‹$'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ $'a$ *set* $\Rightarrow$ *bool*› (‹- $\preceq$- -› [*60*,*50*,*60*]*55*)
assumes *trans-rel*: ‹$K = Cn(A) \Longrightarrow B \preceq_K C \Longrightarrow C \preceq_K D \Longrightarrow B \preceq_K D$›
assumes *nonempty-rel*: ‹$K = Cn(A) \Longrightarrow (K \ .\bot. \ \varphi) \neq \{\} \Longrightarrow \exists \, B \in (K \ .\bot. \ \varphi). (\forall \, C \in (K \ .\bot. \ \varphi). \ C \preceq_K B)$› — pas clair dans la litterrature

fixes *rel-sel*::‹$'a$ *set* $\Rightarrow$ $'a$ $\Rightarrow$ $'a$ *set set*› (‹γ_{TR}›)

defines *rel-sel*: ‹γ_{TR} K $\varphi \equiv$ *if* (K .⊥. φ) = {} *then* {K}
else {B. B∈(K .⊥. φ) ∧ (∀ C∈(K .⊥. φ). C $\preceq_K$ B)}›

begin

A transitively relational selection function respect the partial meet contraction postulates.

sublocale *PartialMeetContraction* **where** *selection* = γ_{TR}
apply(*unfold-locales*)
apply(*simp-all add*: *rel-sel*)
using *nonempty-rel* **apply** *blast*
using *remainder-extensionality* **by** *blast*

end

A full meet contraction is a limiting case of the partial meet contraction where if the remainders are not empty then the selection function return all the remainders (as defined by *full-sel*

locale *FullMeetContraction* = *Tarskian-logic* +

fixes *full-sel*::‹$'a$ *set* ⇒ $'a$ ⇒ $'a$ *set set*› (‹γ_{FC}›)
defines *full-sel*: ‹γ_{FC} K $\varphi \equiv$ *if* K .⊥. φ = {} *then* {K} *else* K .⊥. φ›

begin

A full selection and a relation ? is a transitively relational meet contraction postulates.

sublocale *TransitivelyRelationalMeetContraction* **where** *relation* = ‹λ K A B. *True*› **and** *rel-sel*=γ_{FC}
by (*unfold-locales*, *auto simp add*:*full-sel*, *rule eq-reflection*, *simp*)

end

4.3 Equivalence of partial meet contraction and AGM contraction

locale *PMC-SC* = *PartialMeetContraction* + *Supraclassical-logic* + *Compact-logic*

begin

In a context of a supraclassical and a compact logic the two remaining postulates of AGM contraction :

- *contract-recovery*
- *contract-success* can be proved on a partial meet contraction.

pmc-recovery : all proposition removed by a partial meet contraction of φ will be recovered by the expansion of φ

lemma *pmc-recovery*: ‹$K = Cn(A) \Longrightarrow K \subseteq ((K \div_\gamma \varphi) \oplus \varphi)$›
  apply(*cases* ‹$(K\ .\bot.\ \varphi) = \{\}$›, *simp-all* (*no-asm*) *add*:*mc expansion-def*)
  using *inclusion-L tautology-selection* **apply** *fastforce*
  proof −
    assume *a*:‹$K = Cn(A)$› **and** *b*:‹$K\ .\bot.\ \varphi \neq \{\}$›
    { fix ψ
      assume *d*:‹$K \vdash \psi$›
      have ‹$\varphi\ .\longrightarrow.\ \psi \in \bigcap(\gamma\ K\ \varphi)$›
        using *is-selection*[*OF a b*]
        by *auto* (*metis a d infer-def rem-closure remainder-recovery subsetD*)
    }
    with *a b* **show** ‹$K \subseteq Cn\ (insert\ \varphi\ (\bigcap\ (\gamma\ K\ \varphi)))$›
      by (*metis* (*no-types*, *lifting*) *Un-commute assumption-L imp-PL infer-def insert-is-Un subsetI*)
  qed

pmc-success : a partial meet contraction of K by φ not imply φ

lemma *pmc-success*: ‹$K = Cn(A) \Longrightarrow \varphi \notin Cn(\{\}) \Longrightarrow \varphi \notin K \div_\gamma \varphi$›
proof
  assume *a*:‹$K = Cn(A)$› **and** *b*:‹$\varphi \notin Cn(\{\})$› **and** *c*:‹$\varphi \in K \div_\gamma \varphi$›
  from *c* **show** *False* **unfolding** *mc*
  proof(*cases* ‹$K\ .\bot.\ \varphi = \{K\}$›)
    case *True*
    then show *?thesis*
      by (*meson assumption-L c nonconsequence-remainder pmc-inclusion*[*OF a*] *subsetD*)
  next
    case *False*
    hence ‹$\forall\ B{\in}K\ .\bot.\ \varphi.\ \varphi \notin B$› **using** *assumption-L rem* **by** *auto*
    moreover have ‹$K\ .\bot.\ \varphi \neq \{\}$› **using** *b emptyrem2taut validD-L* **by** *blast*
    ultimately show *?thesis*
    using *b c mc nonempty-selection*[*OF a*] *validD-L emptyrem2taut is-selection*[*OF a*]
      by (*metis Inter-iff bot.extremum-uniqueI subset-iff*)
  qed
qed

As a partial meet contraction has been proven to respect all postulates of AGM contraction the equivalence between the both are straightforward

sublocale *AGM-Contraction* **where** *contraction* = ‹$\lambda A\ \varphi.\ A \div_\gamma \varphi$›
  using *pmc-closure pmc-inclusion pmc-vacuity*
    pmc-success pmc-recovery pmc-extensionality
    expansion-def idempotency-L infer-def
  by (*unfold-locales*) *metis*+

end

locale *AGMC-SC* = *AGM-Contraction* + *Supraclassical-logic* + *Compact-logic*

begin

obs 2.5 page 514

definition *AGM-selection*::‹*'a set* ⇒ *'a* ⇒ *'a set set*› (‹γ_{AGM}›)
  where *AGM-sel*: ‹γ_{AGM} *A* φ ≡ *if A* .⊥. φ = {} *then* {*A*} *else* {*B*. *B* ∈ *A* .⊥. φ ∧ *A* ÷ φ ⊆ *B*}›

The selection function γ_{AGM} respect the partial meet contraction postulates

sublocale *PartialMeetContraction* **where** *selection* = γ_{AGM}
proof(*unfold-locales*, *unfold AGM-sel*, *simp-all*, *goal-cases*)
  case (*1 K A* φ) — *non-emptiness* of selection requires a compact logic
  then show *?case* **using** *upper-remainder*[*of* ‹*K* ÷ φ› *K* φ] *contract-success*[*OF 1*(*1*)]
    by (*metis contract-closure contract-inclusion infer-def taut2emptyrem valid-def*)
next
  case (*2 K A* φ ψ)
  then show *?case*
      by (*metis* (*mono-tags*, *lifting*) *contract-extensionality Collect-cong remainder-extensionality*)
qed

contraction-is-pmc : an AGM contraction is equivalent to a partial met contraction using the selection function γ_{AGM}

lemma *contraction-is-pmc*: ‹*K* = *Cn*(*A*) ⟹ *K* ÷ φ = *K* $\div_{\gamma_{AGM}}$ φ› — requires a supraclassical logic
proof
  assume *a*:‹*K* = *Cn*(*A*)›
  show ‹*K* ÷ φ ⊆ *K* $\div_{\gamma_{AGM}}$ φ›
    using *contract-inclusion*[*OF a*] **by** (*auto simp add*:*mc AGM-sel*)
next
  assume *a*:‹*K* = *Cn*(*A*)›
  show ‹*K* $\div_{\gamma_{AGM}}$ φ ⊆ *K* ÷ φ›
  proof (*cases* ‹⊩ φ›)
    case *True*
    hence ‹*K* .⊥. φ = {}›
      using *nonconsequence-remainder taut2emptyrem* **by** *auto*
    then show *?thesis*
      apply(*simp-all add*:*mc AGM-sel*)
     by (*metis a emptyrem2taut contract-closure contract-recovery valid-expansion*)
  next
    case *validFalse*:*False*
    then show *?thesis*
    proof (*cases* ‹*K* ⊢ φ›)
      case *True*
      hence *b*:‹*K* .⊥. φ ≠ {}›

using *emptyrem2taut validFalse* **by** *blast*
have *d*:‹$\psi \in K \Longrightarrow \varphi \;.\!\!\longrightarrow\!\!.\; \psi \in K \div \varphi$› **for** ψ
using *Supraclassical-logic.impI-PL Supraclassical-logic-axioms a contract-closure contract-recovery expansion-def* **by** *fastforce*
{ fix ψ
assume *e*:‹$\psi \in K$ › **and** *f*:‹$\psi \notin K \div \varphi$›
have ‹$(\psi \;.\!\!\longrightarrow\!\!.\; \varphi) \;.\!\!\longrightarrow\!\!.\; \varphi \notin K \div \varphi$›
using *imp-recovery2*[*of* ‹$K \div \varphi$› φ ψ] *a contract-closure d e f* **by** *auto*
hence *g*:‹$\neg\ (K \div \varphi) \cup \{\psi \;.\!\!\longrightarrow\!\!.\; \varphi\} \vdash \varphi$›
using *a contract-closure impI-PL* **by** *fastforce*
then obtain B **where** *h*:‹$(K \div \varphi) \cup \{\psi \;.\!\!\longrightarrow\!\!.\; \varphi\} \subseteq B$› **and** *i*:‹$B \in K \;.\!\perp\!.\; \varphi$›
using *upper-remainder*[*of* ‹$(K \div \varphi) \cup \{\psi \;.\!\!\longrightarrow\!\!.\; \varphi\}$› K φ] *a True contract-inclusion idempotency-L impI2* **by** *auto*
hence *j*:‹$\psi \notin Cn(B)$›
by (*metis* (*no-types*, *lifting*) *CollectD mp-PL Un-insert-right a infer-def insert-subset rem rem-closure*)
have ‹$\psi \notin K \div_{\gamma_{AGM}} \varphi$›
apply(*simp add*:*mc AGM-sel b*, *rule-tac x*=*B* **in** *exI*)
by (*meson Tarskian-logic.assumption-L Tarskian-logic-axioms h i j le-sup-iff*)
}
then show *?thesis*
using *a pmc-inclusion* **by** *fastforce*
next
case *False*
hence ‹$K \;.\!\perp\!.\; \varphi = \{K\}$›
using *nonconsequence-remainder taut2emptyrem* **by** *auto*
then show *?thesis*
using *False a contract-vacuity idempotency-L pmc-vacuity* **by** *auto*
qed
qed
qed

lemma *contraction-with-completion*: ‹$K = Cn(A) \Longrightarrow K \div \varphi = K \div_{*}\, \gamma_{AGM}\ \varphi$›
by (*simp add*: *contraction-is-pmc pmc-completion*)

end

locale *TRMC-SC* = *TransitivelyRelationalMeetContraction* + *PMC-SC* **where** *selection* = γ_{TR}

begin

A transitively relational selection function respect conjuctive overlap.

lemma *rel-sel-conj-overlap*: ‹$K = Cn(A) \Longrightarrow \gamma_{TR}\ K\ (\varphi \;.\!\wedge\!.\; \psi) \subseteq \gamma_{TR}\ K\ \varphi \cup \gamma_{TR}\ K\ \psi$›
proof(*intro subsetI*)

fix B
assume a:‹$K = Cn(A)$› **and** b:‹$B \in \gamma_{TR}\ K\ (\varphi\ .\wedge.\ \psi)$›
show ‹$B \in \gamma_{TR}\ K\ \varphi \cup \gamma_{TR}\ K\ \psi$› (**is** *?A*)
proof(*cases* ‹$\Vdash \varphi \vee \Vdash \psi \vee \neg\ K \vdash \varphi \vee \neg\ K \vdash \psi$›, *elim disjE*)
assume ‹$\Vdash \varphi$›
hence c:‹$Cn(\{\varphi\ .\wedge.\ \psi\}) = Cn(\{\psi\})$›
using *conj-equiv valid-Cn-equiv valid-def* **by** *blast*
from b **show** *?A*
by (*metis Un-iff a c extensional-selection*)
next
assume ‹$\Vdash \psi$›
hence c:‹$Cn(\{\varphi\ .\wedge.\ \psi\}) = Cn(\{\varphi\})$›
by (*simp add*: *Cn-conj-bis Cn-same validD-L*)
from b **show** *?A*
by (*metis Un-iff a c extensional-selection*)
next
assume ‹$\neg\ K \vdash \varphi$›
then show *?A*
by (*metis UnI1 a b conjE1-PL is-selection nonconsequence-remainder nonempty-selection tautology-selection subset-singletonD*)
next
assume ‹$\neg\ K \vdash \psi$›
then show *?A*
by (*metis UnI2 a b conjE2-PL is-selection nonconsequence-remainder nonempty-selection tautology-selection subset-singletonD*)
next
assume d:‹$\neg\ (\Vdash \varphi \vee \Vdash \psi \vee \neg\ K \vdash \varphi \vee \neg\ K \vdash \psi)$›
hence h:‹$K\ .\bot.\ \varphi\ \neq \{\}$› **and** i:‹$K\ .\bot.\ \psi \neq \{\}$› **and** j:‹$K\ .\bot.\ (\varphi\ .\wedge.\ \psi) \neq \{\}$› **and** k:$K \vdash \varphi\ .\wedge.\ \psi$
using d *emptyrem2taut valid-conj-PL* **apply** *auto*
by (*meson Supraclassical-logic.conjI-PL Supraclassical-logic-axioms d*)
show *?A*
using *remainder-conj*[*OF a k*] *b h i j rel-sel* **by** *auto*
qed
qed

A transitively relational meet contraction respect conjuctive overlap.

lemma *trmc-conj-overlap*: ‹$K = Cn(A) \Longrightarrow (K \div_{\gamma_{TR}}\ \varphi) \cap (K \div_{\gamma_{TR}}\ \psi) \subseteq (K \div_{\gamma_{TR}}\ (\varphi\ .\wedge.\ \psi))$›
unfolding *mc* **using** *rel-sel-conj-overlap* **by** *blast*

A transitively relational selection function respect conjuctive inclusion

lemma *rel-sel-conj-inclusion*: ‹$K = Cn(A) \Longrightarrow \gamma_{TR}\ K\ (\varphi\ .\wedge.\ \psi) \cap (K\ .\bot.\ \varphi) \neq \{\} \Longrightarrow \gamma_{TR}\ K\ \varphi \subseteq \gamma_{TR}\ K\ (\varphi\ .\wedge.\ \psi)$›
proof(*intro subsetI*)
fix B
assume a:‹$K = Cn(A)$› **and** b:‹$\gamma_{TR}\ K\ (\varphi\ .\wedge.\ \psi) \cap (K\ .\bot.\ \varphi) \neq \{\}$› **and** c:‹$B \in \gamma_{TR}\ K\ \varphi$›
show ‹$B \in \gamma_{TR}\ K\ (\varphi\ .\wedge.\ \psi)$› (**is** *?A*)

```
proof(cases ‹⊩ φ ∨ ⊩ ψ ∨ ¬ K ⊢ φ ∨ ¬ K ⊢ ψ›, auto)
  assume ‹⊩ φ›
  then show ?A
    using b taut2emptyrem by auto
next
  assume ‹⊩ ψ›
  hence ‹Cn({φ .∧. ψ}) = Cn({φ})›
    by (simp add: Cn-conj-bis Cn-same validD-L)
  then show ?A
    using a c extensional-selection by blast
next
  assume d:‹φ ∉ Cn K›
  with d show ?A
  by (metis Int-emptyI Tarskian-logic.nonconsequence-remainder Tarskian-logic-axioms
a b c idempotency-L
          inf-bot-right is-selection nonempty-selection singletonD subset-singletonD)
next
  assume d:‹ψ ∉ Cn K›
  hence e:‹(φ .∧. ψ) ∉ Cn K›
    by (meson Supraclassical-logic.conjE2-PL Supraclassical-logic-axioms)
  hence f:‹γ_TR K (φ .∧. ψ) = {K}›
     by (metis Tarskian-logic.nonconsequence-remainder Tarskian-logic-axioms a
insert-not-empty is-selection
              nonempty-selection subset-singletonD)
  with b have g:‹(K .⊥. φ) = {K}›
    unfolding nonconsequence-remainder[symmetric] using rem by auto
  with d f show ?A
    using a c is-selection by fastforce
next
  assume d:‹¬ ⊩ φ› and e:‹¬ ⊩ ψ› and f:‹φ ∈ Cn K› and g:‹ψ ∈ Cn K›
  hence h:‹K .⊥. φ  ≠ {}› and i:‹K .⊥. ψ ≠ {}› and j:‹K .⊥. (φ .∧. ψ) ≠ {}›
and k:K ⊢ φ .∧. ψ
    using e d emptyrem2taut valid-conj-PL apply auto
    by (meson Supraclassical-logic.conjI-PL Supraclassical-logic-axioms f g)
  have o:‹B ∈ K .⊥. φ ⟹ B ∈ K .⊥. (φ .∧. ψ)› for B
    using a k remainder-conj by auto
  from b obtain B' where l:‹B' ∈ K .⊥. (φ .∧. ψ)› and m:‹∀ C∈K .⊥. (φ .∧.
ψ). C ≼_K B'› and n:‹φ ∉ B'›
    apply (auto simp add:mc rel-sel j)
    using assumption-L rem by force
  have p:‹B' ∈ K .⊥. φ›
    apply(simp add: rem)
      by (metis (no-types, lifting) Supraclassical-logic.conjE1-PL Supraclassi-
cal-logic-axioms
                Tarskian-logic.rem Tarskian-logic-axioms a l mem-Collect-eq n
rem-closure)
  from c show ?A
    apply (simp add:rel-sel o j h)
    using m p trans-rel a by blast
```

qed
qed

A transitively relational meet contraction respect conjuctive inclusion

lemma *trmc-conj-inclusion*: ‹$K = Cn(A) \Longrightarrow \varphi \notin (K \div_{\gamma_{TR}} (\varphi .\wedge. \psi)) \Longrightarrow ((K \div_{\gamma_{TR}} (\varphi .\wedge. \psi) \subseteq (K \div_{\gamma_{TR}} \varphi)))$›
proof –
assume *a*:‹$K = Cn(A)$› **and** *b*:‹$\varphi \notin (K \div_{\gamma_{TR}} (\varphi .\wedge. \psi))$›
then obtain B **where** *c*:‹$B \in \gamma_{TR}\ K\ (\varphi .\wedge. \psi)$› **and** *d*:‹$\neg\ B \vdash \varphi$› **apply**(*simp add*:*mc*)
by (*metis b emptyrem2taut is-selection pmc-tautology rem-closure subset-iff validD-L valid-conj-PL*)
hence ‹$B \in (K .\bot. \varphi)$›
using *remainder-recovery-bis*[*OF a - d*, *of* ‹$\varphi .\wedge. \psi$›]
by (*metis* (*no-types*, *opaque-lifting*) *a conj-PL emptyrem2taut insert-not-empty is-selection*
nonconsequence-remainder subsetD taut2emptyrem)
with *c* **have** *e*:‹$\gamma_{TR}\ K\ (\varphi .\wedge. \psi) \cap (K .\bot. \varphi) \neq \{\}$› **by** *blast*
then show ‹$((K \div_{\gamma_{TR}} (\varphi .\wedge. \psi) \subseteq (K \div_{\gamma_{TR}} \varphi)))$›
unfolding *mc* **using** *rel-sel-conj-inclusion*[*OF a e*] **by** *blast*
qed

As a transitively relational meet contraction has been proven to respect all postulates of AGM full contraction the equivalence between the both are straightforward

sublocale *AGM-FullContraction* **where** *contraction* = ‹$\lambda A\ \varphi.\ A\ \div_{\gamma_{TR}} \varphi$›
using *trmc-conj-inclusion trmc-conj-overlap*
by (*unfold-locales*, *simp-all*)

end

locale *AGMFC-SC* = *AGM-FullContraction* + *AGMC-SC*

begin

An AGM relation is defined as ?

definition *AGM-relation*::‹$'a\ set \Rightarrow\ 'a\ set \Rightarrow\ 'a\ set \Rightarrow bool$›
where *AGM-rel*: ‹$AGM\text{-}relation\ C\ K\ B \equiv (C = K \wedge B = K) \vee (\ (\exists \varphi.\ K \vdash \varphi \wedge C \in K .\bot. \varphi)$
$\wedge (\exists \varphi.\ K \vdash \varphi \wedge B \in K .\bot. \varphi \wedge\ K \div \varphi \subseteq B)$
$\wedge (\forall \varphi.\ (K \vdash \varphi \wedge C \in K .\bot. \varphi \wedge B \in K .\bot. \varphi \wedge K \div \varphi \subseteq C) \longrightarrow K \div \varphi \subseteq B))$›

An AGM relational selection is defined as a function that return K if the remainders of $K\ .\bot.\ \varphi$ is empty and the best element of the remainders according to an AGM relation

definition *AGM-relational-selection*::‹'*a set* ⇒ '*a* ⇒ '*a set set*› (‹γ_{AGMTR}›)
  where *AGM-rel-sel*: ‹γ_{AGMTR} K φ ≡ *if* (K .⊥. φ) = {}
    then {K}
    else {B. B∈(K .⊥. φ) ∧ (∀ C∈(K .⊥. φ). *AGM-relation* C K B)}›

lemma *AGM-rel-sel-completion*: ‹K = *Cn*(A) ⟹ γ_{AGMTR} K φ = ∗ γ_{AGM} K φ›
  apply (*unfold AGM-rel-sel*, *simp add*:*completion-def split*: *if-splits*)
proof(*auto simp add*:*AGM-sel*)
  fix S B C
  assume *a*:‹S ∈ *Cn*(A) .⊥. φ› **and** *b*:‹B ∈ *Cn*(A) .⊥. φ› **and** *c*:‹⋂ {B ∈ *Cn*(A) .⊥. φ. *Cn*(A) ÷ φ ⊆ B} ⊆ B›
    and *d*:‹C ∈ *Cn*(A) .⊥. φ›
  hence *e*:‹φ ∉ *Cn*(A) ÷ φ›
    using *Tarskian-logic.taut2emptyrem Tarskian-logic-axioms contract-success* **by** *fastforce*
  show ‹*AGM-relation* C (*Cn*(A)) B›
  proof(*cases* ‹φ ∈ *Cn*(A)›)
    case *True*
    { fix ψ
      assume ‹*Cn* A ÷ ψ ⊆ C›
      hence ‹*Cn* A ÷ (φ .∧. ψ) ⊆ *Cn* A ÷ φ›
        using *contract-conj-inclusion-variant*[*of* ‹*Cn*(A)› A φ ψ]
          by (*metis* (*mono-tags*, *lifting*) *assumption-L contract-conj-inclusion d mem-Collect-eq rem subset-iff*)
    } note f = *this*
    { fix ψ φ'
      assume *g*:‹ψ ∈ *Cn* A ÷ φ'› **and** *h*:‹B ∈ *Cn* A .⊥. φ'› **and** *j*:‹*Cn* A ÷ φ' ⊆ C› **and** *i*:‹ψ ∉ B›
      hence ‹φ' .∨. ψ ∈ *Cn* A ÷ φ'›
      using *Supraclassical-logic.disjI2-PL Supraclassical-logic-axioms contract-closure* **by** *fastforce*
      hence *k*:‹φ' .∨. ψ ∈ *Cn* A ÷ φ›
        using *contract-conj-overlap-variant*[*of* ‹*Cn*(A)› A φ' φ] *f*[*OF j*]
          by (*metis IntI Supraclassical-logic.disjI1-PL Supraclassical-logic-axioms conj-com-Cn*
            contract-extensionality inclusion-L singletonI subsetD)
      hence *l*:‹*Cn* A ÷ φ ⊆ B› **using** *c* **by** *auto*
      from *k l* **have** *m*:‹φ' .∨. ψ ∈ B› **and** *n*:‹B =*Cn*(B)›
        using *b rem-closure* **by** *blast+*
      have ‹B ∪ {ψ} ⊢ φ'› **using** *g h i*
        by (*simp add*:*rem*) (*metis contract-inclusion insertI1 insert-subsetI psubsetI subsetD subset-insertI*)
      with *n m* **have** ‹B ⊢ φ'›
        by (*metis Cn-equiv assumption-L disjE-PL disj-com equiv-PL imp-PL*)
      with *h* **have** *False*
        using *assumption-L rem* **by** *auto*
    } note g = *this*

with *True* **show** *?thesis*
apply(*unfold AGM-rel*, *rule-tac disjI2*)
using *d b c* **by** (*auto simp add*:*AGM-rel idempotency-L del*:*subsetI*) *blast+*
next
case *False*
then show *?thesis*
by (*metis AGM-rel b d idempotency-L infer-def nonconsequence-remainder singletonD*)
qed
next
fix *S B* ψ
assume *a*:‹$S \in Cn(A)$.⊥. φ› **and** *b*:‹$B \in Cn(A)$.⊥. φ› **and** *c*:‹$\forall C{\in}Cn\ A$.⊥. φ. *AGM-relation* C ($Cn\ A$) B›
and *d*:‹$\forall C'$. $C' \in Cn\ A$.⊥. $\varphi \wedge Cn\ A \div \varphi \subseteq C' \longrightarrow \psi \in C'$›
then show ‹$\psi \in B$›
unfolding *AGM-rel*
by (*metis* (*no-types*, *lifting*) *AGM-sel empty-Collect-eq insert-Diff insert-not-empty*

nonconsequence-remainder nonempty-selection singletonD)
qed

A transitively relational selection and an AGM relation is a transitively relational meet contraction

sublocale *TransitivelyRelationalMeetContraction* **where** *relation* = *AGM-relation* **and** *rel-sel* = ‹γ_{AGMTR}›
proof(*unfold-locales*, *simp-all* (*no-asm*) *only*:*atomize-eq*, *goal-cases*)
case *a*:(*1 K A C B′ B*) — Very difficult proof requires litterature and high automation of isabelle!
from *a*(*2*,*3*) **show** *?case*
unfolding *AGM-rel* **apply**(*elim disjE conjE*, *simp-all*)
proof(*intro disjI2 allI impI*, *elim exE conjE*, *goal-cases*)
case (*1* ψ - - φ)
have *b*:‹$B \in K$.⊥. (φ .∧. ψ)› **and** *c*:‹$B' \in K$.⊥. (φ .∧. ψ)› **and** *d*:‹$C \in K$.⊥. (φ .∧. ψ)›
using *remainder-conj*[*OF a*(*1*)] *1 conjI-PL* **by** *auto*
hence *e*:‹$K \div (\varphi$.∧. $\psi) \subseteq B$›
using *contract-conj-inclusion-variant*[*OF a*(*1*), *of* φ ψ]
by (*meson 1*(*1*) *1*(*12*) *1*(*16*) *1*(*2*) *1*(*3*) *1*(*8*) *Supraclassical-logic.conj-PL Supraclassical-logic-axioms dual-order.trans*)
{ fix χ
assume *f*:‹$\chi \in K \div \psi$›
have ‹ψ .∨. $\chi \in (K \div \psi) \cap Cn\ \{\psi\}$›
by (*metis Int-iff Supraclassical-logic.disjI1-PL Supraclassical-logic.disjI2-PL Supraclassical-logic-axioms*
f a(*1*) *contract-closure in-mono inclusion-L singletonI*)
hence *g*:‹ψ .∨. $\chi \in B$›
using *contract-conj-overlap-variant*[*OF a*(*1*), *of* ψ]
by (*metis AGM-Contraction.contract-extensionality AGM-Contraction-axioms a*(*1*) *conj-com-Cn e in-mono*)

have ‹ψ .⟶. $\chi \in B$›
by (*metis a*(*1*) *1*(*10*) *1*(*15*) *1*(*16*) *assumption-L f in-mono infer-def rem-closure rem-inclusion remainder-recovery*)
with g **have** ‹$\chi \in B$›
by (*metis 1*(*15*) *a*(*1*) *disjE-PL infer-def order-refl rem-closure validD-L valid-Cn-imp*)
}
then show *?case* **by** *blast*
qed
next
case (*2 K A* φ)
hence ‹$*\ \gamma_{AGM}\ \ K\ \varphi \neq \{\}$›
using *nonempty-selection*[*OF 2*(*1*), *of* φ] *selection-completion*[*OF 2*(*1*), *of* φ] **by** *blast*
then show *?case*
using *AGM-rel-sel-completion*[*OF 2*(*1*), *of* φ] *AGM-rel-sel 2*(*1*,*2*) **by** *force*
next
case (*3 K* φ)
then show *?case* **using** *AGM-rel-sel-completion AGM-rel-sel* **by** *simp*
qed

— ça marche tout seul! ==> Je ne vois pas où sont utilisés ces lemmas
lemmas *fullcontraction-is-pmc* = *contraction-is-pmc*
lemmas *fullcontraction-is-trmc* = *contraction-with-completion*

end

locale *FMC-SC* = *FullMeetContraction* + *TRMC-SC*

begin

lemma *full-meet-weak1*: ‹$K = Cn(A) \Longrightarrow K \vdash \varphi \Longrightarrow (K \div_{\gamma_{FC}} \varphi) = K \cap Cn(\{.\neg\ \varphi\})$›
proof(*intro subset-antisym Int-greatest*)
assume *a*:‹$K = Cn(A)$› **and** *b*:‹$K \vdash \varphi$›
then show ‹ $(K \div_{\gamma_{FC}} \varphi) \subseteq K$›
by (*simp add*: *Inf-less-eq full-sel mc rem-inclusion*)
next
assume *a*:‹$K = Cn(A)$› **and** *b*:‹$K \vdash \varphi$›
show ‹$(K \div_{\gamma_{FC}} \varphi) \subseteq Cn(\{.\neg\ \varphi\})$›
proof
fix ψ
assume *c*:‹$\psi \in (K \div_{\gamma_{FC}} \varphi)$›
{ assume ‹$\neg\ \{.\neg\ \varphi\} \vdash \psi$›
hence ‹$\neg\ \{.\neg\ \psi\} \vdash \varphi$›
by (*metis Un-insert-right insert-is-Un not-PL notnot-PL*)
hence ‹$\neg\ \{\varphi\ .\vee.\ .\neg\ \psi\} \vdash \varphi$›
by (*metis assumption-L disjI2-PL singleton-iff transitivity2-L*)

then obtain B **where** d:‹$\{\varphi$.∨. .¬ $\psi\} \subseteq B$› **and** e:‹$B \in K$.⊥. φ›
by (*metis a b disjI1-PL empty-subsetI idempotency-L infer-def insert-subset upper-remainder*)
hence f:‹¬ $\psi \in B$›
by (*metis* (*no-types*, *lifting*) *CollectD assumption-L insert-subset disj-notE-PL rem*)
hence ‹¬ $\psi \in (K \div_{\gamma_{FC}} \varphi)$›
using *e mc full-sel* **by** *auto*
}
then show ‹$\psi \in Cn(\{$.¬ $\varphi\})$›
using *c infer-def* **by** *blast*
qed
next
assume a:‹$K = Cn(A)$› **and** b:‹$K \vdash \varphi$›
show ‹$K \cap Cn(\{$.¬ $\varphi\}) \subseteq (K \div_{\gamma_{FC}} \varphi)$›
proof(*safe*)
fix ψ
assume c:‹$\psi \in K$› **and** d: ‹$\psi \in Cn$ {.¬ φ}›
have e:‹$B \vdash$.¬ φ .⟶. ψ› **for** B
by (*simp add*: *d validD-L valid-imp-PL*)
{ fix B
assume f:‹$B \in K$.⊥. φ›
hence ‹$B \vdash \varphi$.⟶. ψ›
using *a assumption-L c remainder-recovery* **by** *auto*
then have f:‹$B \vdash \psi$› **using** *d e*
using *disjE-PL ex-mid-PL* **by** *blast*
}
then show ‹$\psi \in (K \div_{\gamma_{FC}} \varphi)$›
apply(*simp-all add*:*mc c full-sel*)
using *a rem-closure* **by** *blast*
qed
qed

lemma *full-meet-weak2*:‹$K = Cn(A) \Longrightarrow K \vdash \varphi \Longrightarrow Cn((K \div_{\gamma_{FC}} \varphi) \cup \{$.¬ $\varphi\}) = Cn(\{$.¬ $\varphi\})$›
unfolding *full-meet-weak1*
by (*metis Cn-union idempotency-L inf.cobounded2 sup.absorb-iff2 sup-commute*)

end

end

5 Revisions

The third operator of belief change introduce by the AGM framework is the revision. In revision a sentence φ is added to the belief set K in such a way that other sentences of K are removed if needed so that K is consistent

5.1 AGM revision postulates

The revision operator is denoted by the symbol $*$ and respect the following conditions :

- *revis-closure* : a belief set K revised by φ should be logically closed
- *revis-inclusion* : a belief set K revised by φ should be a subset of K expanded by φ
- *revis-vacuity* : if $\neg\varphi$ is not in K then the revision of K by φ is equivalent of the expansion of K by φ
- *revis-success* : a belief set K revised by φ should contain φ
- *revis-extensionality* : Extensionality guarantees that the logic of contraction is extensional in the sense of allowing logically equivalent sentences to be freely substituted for each other
- *revis-consistency* : a belief set K revised by φ is consistent if φ is consistent

locale *AGM-Revision* = *Supraclassical-logic* +

fixes *revision*:: ‹$'a\ set \Rightarrow 'a \Rightarrow 'a\ set$› (**infix** ‹$*$› *55*)

assumes *revis-closure*: ‹$K = Cn(A) \Longrightarrow K * \varphi = Cn(K * \varphi)$›
and *revis-inclusion*: ‹$K = Cn(A) \Longrightarrow K * \varphi \subseteq K \oplus \varphi$›
and *revis-vacuity*: ‹$K = Cn(A) \Longrightarrow .\neg\ \varphi \notin K \Longrightarrow K \oplus \varphi \subseteq K * \varphi$›
and *revis-success*: ‹$K = Cn(A) \Longrightarrow \varphi \in K * \varphi$›
and *revis-extensionality*: ‹$K = Cn(A) \Longrightarrow Cn(\{\varphi\}) = Cn(\{\psi\}) \Longrightarrow K * \varphi = K * \psi$›
and *revis-consistency*: ‹$K = Cn(A) \Longrightarrow .\neg\ \varphi \notin Cn(\{\}) \Longrightarrow \bot \notin K * \varphi$›

A full revision is defined by two more postulates :

- *revis-superexpansion* : An element of $K * (\varphi\ .\wedge.\ \psi)$ is also an element of K revised by φ and expanded by ψ
- *revis-subexpansion* : An element of $(K * \varphi) \oplus \psi$ is also an element of K revised by $\varphi\ .\wedge.\ \psi$ if $(K * \varphi)$ do not imply $\neg\ \psi$

locale *AGM-FullRevision* = *AGM-Revision* +
assumes *revis-superexpansion*: ‹$K = Cn(A) \Longrightarrow K * (\varphi\ .\wedge.\ \psi) \subseteq (K * \varphi) \oplus \psi$›
and *revis-subexpansion*: ‹$K = Cn(A) \Longrightarrow .\neg\ \psi \notin (K * \varphi) \Longrightarrow (K * \varphi) \oplus \psi \subseteq K * (\varphi\ .\wedge.\ \psi)$›

begin

— important lemmas/corollaries that can replace the previous assumptions
corollary *revis-superexpansion-ext* : ‹$K = Cn(A) \Longrightarrow (K * \varphi) \cap (K * \psi) \subseteq (K * (\varphi\ .\vee.\ \psi))$›
proof(*intro subsetI*, *elim IntE*)
  fix χ
  assume *a*:‹$K = Cn(A)$› **and** *b*:‹$\chi \in (K * \varphi)$› **and** *c*:‹$\chi \in (K * \psi)$›
  have ‹ $Cn(\{(\varphi'\ .\vee.\ \psi')\ .\wedge.\ \varphi'\}) = Cn(\{\varphi'\})$› **for** $\varphi'\ \psi'$
    using *conj-superexpansion2* **by** (*simp add*: *Cn-same*)
  hence *d*:‹$K * \varphi' \subseteq (K * (\varphi'\ .\vee.\ \psi')) \oplus \varphi'$› **for** $\varphi'\ \psi'$
    using *revis-superexpansion*[*OF a*, *of* ‹$\varphi'\ .\vee.\ \psi'$› φ'] *revis-extensionality a* **by** *metis*
  hence ‹$\varphi\ .\longrightarrow.\ \chi \in (K * (\varphi\ .\vee.\ \psi))$› **and** ‹$\psi\ .\longrightarrow.\ \chi \in (K * (\varphi\ .\vee.\ \psi))$›
    using *d*[*of* $\varphi\ \psi$] *d*[*of* $\psi\ \varphi$] *revis-extensionality*[*OF a disj-com-Cn*, *of* $\psi\ \varphi$]
    using *imp-PL a b c expansion-def revis-closure* **by** *fastforce+*
  then show *c*:‹$\chi \in (K * (\varphi\ .\vee.\ \psi))$›
    using *disjE-PL a revis-closure revis-success* **by** *fastforce*
qed

end

5.2 Relation of AGM revision and AGM contraction

The AGM contraction of K by φ can be defined as the AGM revision of K by $\neg\varphi$ intersect with K (to remove $\neg\varphi$ from K revised). This definition is known as Harper identity [3]

sublocale *AGM-Revision* $\subseteq$ *AGM-Contraction* **where** *contraction* = ‹$\lambda K\ \varphi.\ K \cap (K * .\neg\ \varphi)$›
proof(*unfold-locales*, *goal-cases*)
  case *closure*:(*1 K A* φ)
  then show *?case*
    by (*metis Cn-inter revis-closure*)
next
  case *inclusion*:(*2 K A* φ)
  then show *?case* **by** *blast*
next
  case *vacuity*:(*3 K A* φ)
  hence ‹$.\neg\ (.\neg\ \varphi) \notin K$›
    using *absurd-PL infer-def* **by** *blast*
  hence ‹$K \subseteq (K * .\neg\ \varphi)$›
    using *revis-vacuity*[**where** φ=‹$.\neg\ \varphi$›] *expansion-def inclusion-L vacuity*(*1*) **by** *fastforce*
  then show *?case*
    by *fast*
next
  case *success*:(*4 K A* φ)
  hence ‹$.\neg\ (.\neg\ \varphi) \notin Cn(\{\})$›
    using *infer-def notnot-PL* **by** *blast*
  hence *a*:‹$\bot \notin K * (.\neg\ \varphi)$›

by (*simp add*: *revis-consistency success*(*1*))
have ‹$.\neg\ \varphi \in\ K * (.\neg\ \varphi)$›
by (*simp add*: *revis-success success*(*1*))
with *a* **have** ‹$\varphi \notin\ K * (.\neg\ \varphi)$›
using *infer-def non-consistency revis-closure success*(*1*) **by** *blast*
then show *?case*
by *simp*
next
case *recovery*:(*5 K A* φ)
show *?case*
proof
fix ψ
assume *a*:‹$\psi \in K$›
hence *b*:‹$\varphi\ .\longrightarrow.\ \psi \in K$› **using** *impI2 recovery* **by** *auto*
have ‹$.\neg\ \psi\ .\longrightarrow.\ .\neg\ \varphi \in K * .\neg\ \varphi$›
using *impI2 recovery revis-closure revis-success* **by** *fastforce*
hence ‹$\varphi\ .\longrightarrow.\ \psi \in K * .\neg\ \varphi$›
using *imp-contrapos recovery revis-closure* **by** *fastforce*
with *b* **show** ‹$\psi \in Cn\ (K \cap (K * .\neg\ \varphi) \cup \{\varphi\})$›
by (*meson Int-iff Supraclassical-logic.imp-PL Supraclassical-logic-axioms inclusion-L subsetD*)
qed
next
case *extensionality*:(*6 K A* φ ψ)
hence ‹$Cn(\{.\neg\ \varphi\}) = Cn(\{.\neg\ \psi\})$›
using *equiv-negation*[*of* ‹{}› φ ψ] *valid-Cn-equiv valid-def* **by** *auto*
hence ‹$(K * .\neg\ \varphi) = (K * .\neg\ \psi)$›
using *extensionality*(*1*) *revis-extensionality* **by** *blast*
then show *?case* **by** *simp*
qed

locale *AGMC-S* = *AGM-Contraction* + *Supraclassical-logic*

The AGM revision of K by φ can be defined as the AGM contraction of K by $\neg\varphi$ followed by an expansion by φ. This definition is known as Levi identity [4].

sublocale *AGMC-S* ⊆ *AGM-Revision* **where** *revision* = ‹$\lambda K\ \varphi.\ (K \div .\neg\ \varphi) \oplus \varphi$›
proof(*unfold-locales*, *goal-cases*)
case *closure*:(*1 K A* φ)
then show *?case*
by (*simp add*: *expansion-def idempotency-L*)
next
case *inclusion*:(*2 K A* φ)
have $K \div .\neg\ \varphi \subseteq K \cup \{\varphi\}$
using *contract-inclusion inclusion* **by** *auto*
then show *?case*
by (*simp add*: *expansion-def monotonicity-L*)

```
next
  case vacuity:(3 K A φ)
  then show ?case
    by (simp add: contract-vacuity expansion-def)
next
  case success:(4 K A φ)
  then show ?case
    using assumption-L expansion-def by auto
next
  case extensionality:(5 K A φ ψ)
  hence ‹Cn({.¬ φ}) = Cn({.¬ ψ})›
    using  equiv-negation[of ‹{}› φ ψ] valid-Cn-equiv valid-def by auto
  hence ‹(K ÷ .¬ φ) = (K ÷ .¬ ψ)›
    using contract-extensionality extensionality(1) by blast
  then show ?case
    by (metis Cn-union expansion-def extensionality(2))
next
  case consistency:(6 K A φ)
  then show ?case
    by (metis contract-closure contract-success expansion-def infer-def not-PL)
qed
```

The relationship between AGM full revision and AGM full contraction is the same as the relation between AGM revison and AGM contraction

```
sublocale AGM-FullRevision ⊆ AGM-FullContraction where contraction = ‹λK
φ. K ∩ (K ∗ .¬ φ)›
proof(unfold-locales, goal-cases)
  case conj-overlap:(1 K A φ ψ)
  have a:‹Cn({.¬ (φ .∧. ψ)}) = Cn({(.¬ φ) .∨. (.¬ ψ)})›
    using Cn-same morgan by simp
  show ?case (is ?A)
    using revis-superexpansion-ext[OF conj-overlap(1), of ‹.¬ φ› ‹.¬ ψ›]
          revis-extensionality[OF conj-overlap(1) a] by auto
next
  case conj-inclusion:(2 K A φ ψ)
  have a:‹Cn({.¬ (φ .∧. ψ) .∧. .¬ φ}) = Cn({.¬ φ})›
    using conj-superexpansion1 by (simp add: Cn-same)
  from conj-inclusion show ?case
  proof(cases ‹φ ∈ K›)
    case True
    hence b:‹.¬ (.¬ φ) ∉ K ∗ .¬ (φ .∧. ψ)›
      using absurd-PL  conj-inclusion revis-closure by fastforce
    show ?thesis
        using revis-subexpansion[OF conj-inclusion(1) b] revis-extensionality[OF
conj-inclusion(1) a]
            expansion-def inclusion-L by fastforce
  next
    case False
    then show ?thesis
```

by (*simp add*: *conj-inclusion*(*1*) *contract-vacuity*)
qed
qed

locale *AGMFC-S* = *AGM-FullContraction* + *AGMC-S*

sublocale *AGMFC-S* $\subseteq$ *AGM-FullRevision* **where** *revision* = ‹$\lambda K\ \varphi.\ (K \div .\neg\ \varphi)$ $\oplus\ \varphi$›
proof(*unfold-locales*, *safe*, *goal-cases*)
case *super*:(*1 K A* $\varphi\ \psi\ \chi$)
hence *a*:‹$(\varphi .\wedge. \psi) .\longrightarrow. \chi \in Cn(Cn(A) \div .\neg\ (\varphi .\wedge. \psi))$›
using *Supraclassical-logic.imp-PL Supraclassical-logic-axioms expansion-def* **by** *fastforce*
have *b*:‹$(\varphi .\wedge. \psi) .\longrightarrow. \chi \in Cn(\{.\neg\ (\varphi .\wedge. \psi)\})$›
by (*meson Supraclassical-logic.imp-recovery0 Supraclassical-logic.valid-disj-PL Supraclassical-logic-axioms*)
have *c*:‹$(\varphi .\wedge. \psi) .\longrightarrow. \chi \in Cn(A) \div (.\neg\ (\varphi .\wedge. \psi) .\wedge. .\neg\ \varphi)$›
using *contract-conj-overlap-variant*[*of* ‹$Cn(A)$› *A* ‹$.\neg\ (\varphi .\wedge. \psi)$› ‹$.\neg\ \varphi$›] *a b*
using *AGM-Contraction.contract-closure AGM-FullContraction-axioms AGM-FullContraction-def* **by** *fastforce*
have *d*:‹$Cn(\{.\neg\ (\varphi .\wedge. \psi) .\wedge. .\neg\ \varphi\}) = Cn(\{.\neg\ \varphi\})$›
using *conj-superexpansion1* **by** (*simp add*: *Cn-same*)
hence *e*:‹$(\varphi .\wedge. \psi) .\longrightarrow. \chi \in Cn(A) \div .\neg\ \varphi$›
using *AGM-Contraction.contract-extensionality*[*OF - - d*] *c*
AGM-FullContraction-axioms AGM-FullContraction-def **by** *fastforce*
hence *f*:‹$\varphi .\longrightarrow. (\psi .\longrightarrow. \chi) \in Cn(A) \div .\neg\ \varphi$›
using *conj-imp AGM-Contraction.contract-closure AGM-FullContraction-axioms AGM-FullContraction-def conj-imp* **by** *fastforce*
then show *?case*
by (*metis assumption-L expansion-def imp-PL infer-def*)
next
case *sub*:(*2 K A* $\psi\ \varphi\ \chi$)
hence *a*:‹$\varphi .\longrightarrow. (\psi .\longrightarrow. \chi) \in Cn(A) \div .\neg\ \varphi$›
by (*metis AGMC-S.axioms*(*1*) *AGMC-S-axioms AGM-Contraction.contract-closure expansion-def impI-PL infer-def revis-closure*)
have *b*:‹$Cn(\{.\neg\ (\varphi .\wedge. \psi) .\wedge. .\neg\ \varphi\}) = Cn(\{.\neg\ \varphi\})$›
using *conj-superexpansion1* **by** (*simp add*: *Cn-same*)
have *c*:‹$.\neg\ (\varphi .\wedge. \psi) \notin Cn\ A \div (.\neg\ \varphi)$›
using *sub*(*1*) **by** (*metis assumption-L conj-imp expansion-def imp-PL infer-def not-PL*)
have *c*:‹$Cn(A) \div .\neg\ \varphi \subseteq Cn(A) \div (.\neg\ (\varphi .\wedge. \psi))$›
using *contract-conj-inclusion*[*of* ‹$Cn(A)$› *A* ‹$.\neg\ (\varphi .\wedge. \psi)$› ‹$.\neg\ \varphi$›]
by (*metis AGM-Contraction.contract-extensionality AGM-FullContraction.axioms*(*1*) *AGM-FullContraction-axioms b c*)
then show *?case*
by (*metis a assumption-L conj-imp expansion-def imp-PL in-mono infer-def*)
qed

end

References

[1] C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The journal of symbolic logic*, 50(2):510–530, 1985.

[2] E. Fermé and S. O. Hansson. Agm 25 years: Twenty-five years of research in belief change. *Journal of Philosophical Logic*, 40:295–331, 04 2011.

[3] W. L. Harper. Rational conceptual change. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1976:462–494, 1976.

[4] I. Levi. Subjunctives, dispositions and chances. *Synthese*, 34:303–335, 1977.

[5] R. Wójcicki. *Theory of logical calculi: basic theory of consequence operations*, volume 199. Springer Science & Business Media, 2013.