CEUR-WS.org/Vol-4071/paper21.pdf

The KLM Representation Theorem for System C, Formally

Jonathan Walther[1], Kai Sauerwald[1] and Jesse Heyninck[2,3]

[1]University of Hagen, Germany
[2]Open Universiteit, the Netherlands
[3]University of Cape Town and CAIR, Cape Town, South Africa

Abstract
We present a formalization of the proof of the correspondence between cumulative non-monotonic reasoning and System C in a proof assistant. Reasoning based on System C is the cornerstone of non-monotonic reasoning and was given a semantics via cumulative models by Kraus, Lehmann and Magidor. Our proof is inspired by the original proof and written in the proof system Rocq and focuses on propositional logic. Due to the features of Rocq, the proof implicitly yields a verified implementation of System C reasoning.

1. Introduction

Representation theorems often establish a basic connection between the *"how"* (constructive description) and the *"what"* (axiomatic description). However, proving representations theorems is a delicate matter and challenging to achieve, as one must carefully ensure that both directions of the theorem are fully matched. Likewise, verification of representation theorems requires much expertise and time. Proof assistants could be of aid in verifying and developing representation theorems. In this paper, we present a formalization and verification of the representation theorem on cumulative non-monotonic reasoning by Kraus, Lehmann, and Magidor [1] (KLM) in the proof assistant Rocq.

Cumulative reasoning is well accepted as the core of non-monotonic reasoning approaches [1], see, e.g., Gabbay et al. [2] or Brewka et al. [3]. Cumulative reasoning is based on its axiomatic presentation as System C going back to Gabbay [4]. Alternatively, cumulative reasoning can be presented semantically through cumulative models. A cumulative model provides a strict partial order $\prec$ on sets of interpretations of some underlying logic (which is here propositional logic). Then, one says a formula β is entailed from α within a cumulative model if all $\prec$-minimal models of α are models of β, i.e.,

$$\alpha \mathrel{|\!\sim} \beta \text{ if } \min(\mathrm{Mod}(\alpha), \prec) \subseteq \mathrm{Mod}(\beta).$$

The coincidence between the elegant semantic representation via cumulative models and the aximatic System C was first proved by KLM [1]. The proof is highly nontrivial and employs conditions similar to those in Lewis's limit assumption.

We formalized the proof by KLM [1] for the representation theorem on cumulative reasoning in *Rocq*, formerly known as *coq*. The proof is semi-automatic, as it relies on hints given to the proof system on how to proceed with the proof. Correctness of the proof is then guaranteed by the features of the proof language itself. Here, we outline our proof in Rocq and present examples on how certain parts of the soundness and correctness are implemented in Rocq. Notably, we could not directly implement the original proof and had to carry out the proof slightly differently. The full proof in Rocq is available online[1]. Our proof also includes implementation of reasoning via System C derivations and reasoning via cumulative models. Furthermore, we discuss the extension of the proof to other non-monotonic reasoning approaches. In summary, the main contributions are:

- a formalization in Rocq of the KLM-representation theorem for cumulative reasoning;
- two implementations of cumulative reasoning: via cumulative models and via System C proofs.

In the next section, we start by providing the background for this paper.

23nd International Workshop on Nonmonotonic Reasoning, November 11-13, 2025, Melbourne, Australia
0000-0002-1551-7016 (K. Sauerwald); 0000-0002-3825-4052 (J. Heyninck)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]https://github.com/jonawa-q9677453/KLMRocq

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

2. Background

Let $\Sigma = \{a, b, c, \ldots\}$ be a finite set of propositional atoms, and let $\mathcal{L}$ be a propositional language over Σ (including implication and bi-implication). The set of propositional interpretations over Σ is denoted by Ω and $\models$ is the usual model relation. The set of models of α is denoted by $\mathrm{Mod}(\alpha)$. Classical propositional entailment is denoted by $\models$ and is defined by $\alpha \models \beta$ if $\mathrm{Mod}(\alpha) \subseteq \mathrm{Mod}(\beta)$.

Any relation $\mid\!\sim\ \subseteq \mathcal{L} \times \mathcal{L}$ will be denoted here as a consequence relation. A central notion for consequence relations is monotonicity, stating that if $\alpha \mid\!\sim \gamma$, then also $\alpha \wedge \beta \mid\!\sim \gamma$. The propositional entailment relation $\models$ from above satisfies monotonicity. Consequence relations that violate monotonicity are denoted as non-monotonic. In the seminal work by Gabbay [4], the following properties for non-monotonic reasoning systems are given, which are collectively known as *System C*:

(Ref) $\alpha \mid\!\sim \alpha$ (Reflexivity)

(CM) $\dfrac{\alpha \mid\!\sim \beta \quad \alpha \mid\!\sim \gamma}{\alpha \wedge \beta \mid\!\sim \gamma}$ (Cautious Monotonicity)

(RW) $\dfrac{\models \alpha \to \beta \quad \gamma \mid\!\sim \alpha}{\gamma \mid\!\sim \beta}$ (Right Weakening)

(LLE) $\dfrac{\models \alpha \leftrightarrow \beta \quad \alpha \mid\!\sim \gamma}{\beta \mid\!\sim \gamma}$ (Left Logical Equivalence)

(Cut) $\dfrac{\alpha \wedge \beta \mid\!\sim \gamma \quad \alpha \mid\!\sim \beta}{\alpha \mid\!\sim \gamma}$

(Ref) is a basic property of consequence relations. (LLE) ensures that consequences respect classical equivalence, (RW) provides classical conclusions, (Cut) provides plausible conclusions, and (CM) is a restricted form of monotonicity. The System C postulates are widely accepted as the minimal requirements for good rational non-monotonic reasoning. A consequence relation $\mid\!\sim$ is called *cumulative* if System C is satisfied by $\mid\!\sim$. When we axiomatically claim that a consequence $\alpha \mid\!\sim \beta$ should hold, we call $\alpha \mid\!\sim \beta$ a conditional assertion. For a set of conditional assertions $\mathbf{K}$, we write $\mathbf{K} \models \alpha \mid\!\sim \beta$ if there is a proof for $\alpha \mid\!\sim \beta$ via the System C when assuming the assertions in $\mathbf{K}$. For every cumulative consequence relation $\mid\!\sim$, there is a set $\mathbf{K}$ that gives rise to $\mid\!\sim$.

In the following, we present a general semantic approach to System C that goes back to Kraus, Lehmann and Magidor [1]. Intuitively, in the semantics we will define below, each System C consequence relation can be represented by a state space of sets of interpretations that is endowed with an order $\prec$. Formally, a *state-order model* is a triple $\mathbb{O} = \langle \mathcal{S}, \ell, \prec \rangle$ such that

- $\mathcal{S}$ is set, whose elements are called *states*,
- $\ell : \mathcal{S} \to \mathcal{P}(\Omega)$ assigns to each state a set of interpretations and is called a *labelling function*, and
- $\prec \subseteq \mathcal{S} \times \mathcal{S}$ is a relation on the states.

Reasoning over $\mathbb{O}$ is then implemented by considering only minimal elements with respect to $\prec$. For a strict partial order $\prec \subseteq \mathcal{S} \times \mathcal{S}$ on a set $\mathcal{S}$ and a subset $S \subseteq \mathcal{S}$, an element $s \in S$ is called *minimal in S with respect to* $\prec$ if for each $t \in S$ holds $t \not\prec s$. Then, $\min(S, \prec)$ is the set of all $s \in S$ that are minimal in S with respect to $\prec$. In the context of a state-order model, we let $\mathcal{S}(\alpha) = \{s \in \mathcal{S} \mid \ell(s) \subseteq \mathrm{Mod}(\alpha)\}$.

Definition 1. *For a state-order model $\mathbb{O} = \langle \mathcal{S}, \ell, \prec \rangle$, we let $\alpha \mid\!\sim_{\mathbb{O}} \beta$ if $\min(\mathcal{S}(\alpha), \prec) \subseteq \mathcal{S}(\beta)$.*

To fully capture cumulative reasoning, one has to guarantee that minimal elements exist. A set $S \subseteq \mathcal{S}$ is called *smooth with respect to* $\prec$ if for all $t \in S$, either $t \in \min(S, \prec)$ or there exist $s \in \min(S, \prec)$ such that $s \prec t$. We say a relation $\prec$ *is smooth* if $\mathcal{S}(\alpha)$ is smooth for all formulas $\alpha \in \mathcal{L}$.

Definition 2. *A* cumulative model *is a state-order model $\mathbb{W} = \langle \mathcal{S}, \ell, \prec \rangle$ where $\prec$ is smooth.*

Note that in the context of this paper, where we restrict our investigations to propositional logic, it is sufficient to consider only cumulative models where $\mathcal{S}$ is finite [5]. In such finite models, smoothness is automatically satisfied. Nonetheless, we will encounter models with infinite state sets within proofs.

Kraus, Lehmann and Magidor establish a fundamental correspondence between cumulative consequence relations and consequence relations based on cumulative models [1].

Theorem 1 ([1]). *A consequence relation $\mid\!\sim$ is cumulative if and only if there exists a cumulative model $\mathbb{W}$ such that $\mid\!\sim\ =\ \mid\!\sim_{\mathbb{W}}$.*

In the following, we give an outline of the proof of Theorem 1 by Kraus, Lehmann and Magidor.

[Soundness.] The soundness direction proves that cumulative models satisfy all System C rules. The proof treats each System C postulate individually, with particular attention to Cut and Cautious Monotonicity. For Cut, if all minimal elements of $\mathcal{S}(\alpha)$ satisfy β and all minimal elements of $\mathcal{S}(\alpha \wedge \beta)$ satisfy γ, then any minimal element of $\mathcal{S}(\alpha)$ satisfies β and therefore $\alpha \wedge \beta$. Since it is minimal in $\mathcal{S}(\alpha)$ and $\mathcal{S}(\alpha \wedge \beta) \subseteq \mathcal{S}(\alpha)$, it is also minimal in $\mathcal{S}(\alpha \wedge \beta)$. For Cautious Monotonicity, assume $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \beta$ and $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \gamma$, then prove $\alpha \wedge \beta \mathrel{|\!\sim_{\mathbb{W}}} \gamma$. Take any state s minimal in $\mathcal{S}(\alpha \wedge \beta)$; since $s \in \mathcal{S}(\alpha)$, by the smoothness condition, either s is minimal in $\mathcal{S}(\alpha)$ or there exists s' minimal in $\mathcal{S}(\alpha)$ such that $s' \prec s$. If s' exists, then $s' \models \beta$ (by $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \beta$), so $s' \in \mathcal{S}(\alpha) \cap \mathcal{S}(\beta)$, making $s' \in \mathcal{S}(\alpha \wedge \beta)$, contradicting the minimality of s. Therefore, s is minimal in $\mathcal{S}(\alpha)$, and since $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \gamma$, we conclude $s \models \gamma$.

[Completeness.] The completeness direction constructs a canonical model based on the cumulative consequence relation $\mathrel{|\!\sim}$ that induces the same consequence relation. A model $m \in \mathrm{Mod}(\alpha)$ of α is *normal for* α if for all $\beta \in L$ with $\alpha \mathrel{|\!\sim} \beta$ holds $m \models \beta$. For two formulas α and β, we write $\alpha \sim \beta$ if $\alpha \mathrel{|\!\sim} \beta$ and $\beta \mathrel{|\!\sim} \alpha$. The canonical model $\mathbb{W} = (\mathcal{S}, \ell, \prec)$ for $\mathrel{|\!\sim}$ is defined by:

$$\mathcal{S} = \mathcal{L}/\sim \qquad \text{(states are $\sim$ equivalence classes)}$$

$$\ell([\alpha]_\sim) = \{m \in \Omega \mid m \text{ is normal for } \alpha\} \qquad \text{(labels are all normal world)}$$

$$[\alpha]_\sim \prec [\beta]_\sim \text{ if } [\alpha]_\sim \neq [\beta]_\sim \text{ and there is } \gamma \in [\alpha]_\sim \text{ with } \beta \mathrel{|\!\sim} \gamma$$

The proof is then by showing two central insights:

(A) A normal world m for α satisfies β if and only if $\alpha \mathrel{|\!\sim} \beta$.

(B) For every formula α, the state $[\alpha]_\sim$ is the unique minimal element in $\min(\mathcal{S}(\alpha), \prec)$.

Because of (B) and $\ell([\alpha]_\sim) = \{m \in \Omega \mid m \text{ is normal for } \alpha\}$, it holds $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \beta$ if and only if all normal worlds for α satisfy β. Because of (A), the latter is equivalent to stating that $\alpha \mathrel{|\!\sim} \beta$ holds. In summary, we obtain that $\alpha \mathrel{|\!\sim} \beta$ if and only if $\alpha \mathrel{|\!\sim_{\mathbb{W}}} \beta$.

3. Rocq as Proof Assistant

Rocq is an interactive theorem prover based on the Calculus of Inductive Constructions (CIC) [6]. Unlike automatic theorem provers that either succeed or fail without human intervention, Rocq combines human intuition with machine precision through interactive proving. Users guide the proof strategy by providing high-level proof ideas, while Rocq verifies the correctness of each individual step and ensures logical consistency.

The fundamental principle of interactive proving consists of a dialogue between a human and a machine. Proofs are constructed step-by-step using *tactics*, which are commands that transform proof goals. The proof state consists of a context (available hypotheses and definitions) and goals (statements to be proven). Common tactics essential for our formalization include:

- `intros`: Introduces hypotheses and variables from implications and quantifications
- `apply`: Applies existing theorems or hypotheses to transform goals
- `induction`: Performs structural induction over inductively defined types
- `destruct`: Performs case analysis on data structures or logical connectives
- `split`: Splits conjunctive goals into separate subgoals
- `unfold`: Expands definitions to reveal underlying structure

This interactive approach provides significant advantages over both manual proofs and fully automatic systems. Manual proofs may contain errors or use implicit assumptions, while automatic provers often fail on complex problems and provide little insight when they do fail. Interactive proving ensures absolute correctness while maintaining human control over proof strategies.

4. Formalizing Consequence Relations

Here we present our implementations of cumulative reasoning and reasoning based on cumulative models in Rocq. Instead of implementing propositional logic, we use an existing library by Dakai Guo and Wensheng Yu [7]. Notably, the type `Formula` is provided, which stands for propositional formula, and `Ensemble Formula`, which stands for sets of formulas. This allows us to focus primarily on the actual formalization of the KLM theorem.

4.1. Inductive Definition of Cumulative Consequences

Cumulative consequences are based on finite inferences via System C from some set of conditional assertions $\mathbf{K}$. We define conditional assertions as formula pairs through `ConditionalAssertion` and use `InKB` to check membership in knowledge bases. Cumulative consequences $\mathbf{K} \models \alpha \mid\!\sim \beta$ are implemented as an application of the System C rules as derivation steps in the following type `CumulCons`:

```
1   Inductive CumulCons :
2   KnowledgeBase -> Ensemble Formula -> Formula -> Formula -> Prop :=
3   | Ref : forall K Γ p,
4     CumulCons K Γ p p
5   | LLE : forall K Γ p q r,
6     In Formula Γ (p ↔ q) ->
7     CumulCons K Γ p r ->
8     CumulCons K Γ q r
9   | RW : forall K Γ p q r,
10    In Formula Γ (p → q) ->
11    CumulCons K Γ r p ->
12    CumulCons K Γ r q
13  | Cut : forall K Γ p q r,
14    CumulCons K Γ (p ∧ q) r ->
15    CumulCons K Γ p q ->
16    CumulCons K Γ p r
17  | CM : forall K Γ p q r,
18    CumulCons K Γ p q ->
19    CumulCons K Γ p r ->
20    CumulCons K Γ (p ∧ q) r.
21  | Base : forall K Γ p q,
22    InKB K p q ->
23    CumulCons K Γ p q.
```

The type `CumulCons` takes four parameters. First, `KnowledgeBase` $\mathbf{K}$ that contains the conditional assertions, an `Ensemble Formula` Γ, which represents a set of formulas, and formulas `p` and `q` representing premise and conclusion. The return type `Prop` indicates that this is a logical property. The set of formulas Γ is here for technical reasons, and it is meant to explicitly represent the valid formulas of propositional logic. Lines 3-4 implement `Ref` through universal quantification, using parameter `p` twice for reflexivity. Lines 5-8 define `LLE` with line 6 requiring the equivalence of formulas and line 7 using recursive `CumulCons` calls. The `RW` constructor (lines 9-12) follows a similar structure, but with implication instead of equivalence. Both `Cut` (lines 13-16) and `CM` (lines 17-20) use two recursive premises, where Cut removes conjunctions from premises while Cautious Monotonicity introduces them. Lines 21-23 implement a query to the set of conditional assertions $\mathbf{K}$. Our implementation also employs Rocq's ability to introduce intuitive notation for better readability:

```
Notation "K ⊕ Γ ⊢ p |~ q" := (CumulCons K Γ p q) (at level 80).
```

4.2. Modeling Cumulative Models

For formalizing cumulative models (Definition 1), we use a record type, which provides advantages over separate definitions or inductive types through bundled related components with automatic projection:

```
1   Record CumulModel : Type := {
2       States : Type;
3       Labeling : States -> State;
4       PreferenceRel : States -> States -> Prop;
5   }.
6   Definition State := Formula -> bool.
```

For the states `States` (line 2), we define them as abstract `Type`, which allows us to instantiate them with arbitrary objects (as the original definition). The function `Labeling` (line 3) represents a direct mapping of the labeling function from the KLM definition, where `State` (defined in line 7) is a function that assigns truth values to formulas. This represents a functional implementation, giving each state directly an interpretation function. The implemented labeling function is also adapted to propositional logic and simplified, since each world corresponds exactly to one truth value assignment. The preference relation `PreferenceRel` (line 4) is implemented as a binary relation of type `Prop`. The record type initially contains no restrictions in the definition – properties like the smoothness condition are formulated later as separate axioms, maintaining separation between structures and properties for modular proof construction.

For formalizing minimal elements, we additionally introduce the definition of `MinimalElements`:

```
1   Definition MinimalElements (model : CumulModel)
2   (formula : Formula) : Ensemble (States model) :=
3   fun state =>
4   entails (Labeling model state) formula /\
5   ~ exists state',
6   entails (Labeling model state') formula /\
7   PreferenceRel model state' state.
```

The function `MinimalElements` (lines 1-2) implements $\min(\mathcal{S}(\alpha), \prec)$ as a function with parameters `model` and `formula`, returning an ensemble of states (a set of states). The implementation first checks that the formula holds in the state (line 4), and second, that no preferred state exists in which the formula also holds (lines 5-7). This corresponds directly to the mathematical definition given in Section 2.

4.2.1. Definition of Model-based Consequence Relation

We implementing consequences $|\!\sim_{\mathbb{W}}$ from a cumulative model $\mathbb{W}$ as function `SemanticEntails`. For that, we employ the implementation `MinimalElements` for checking whether a conclusion `conclusion` holds in all minimal states from premise `premise`:

```
1   Definition SemanticEntails (model : CumulModel)
2   (premise conclusion : Formula) : Prop :=
3   forall state,
4   In (States model) (MinimalElements model premise) state ->
5   entails (Labeling model state) conclusion.
6
7   Notation "model : premise |~w conclusion" :=
8   (SemanticEntails model premise conclusion) (at level 80).
```

With `forall state` (line 3), we ensure that the inference relation must hold in all minimal states, and with the implication (line 4), we ensure that only minimal states are considered. In Lines 7-8, we define a short notation for semantic consequence relations in cumulative models.

4.3. The Smoothness Condition Formalized in Rocq

Since we consider only finite logical languages, it is sufficient to consider finite cumulative models [5]. In such models, smoothness is always satisfied. Because of that, we implement smoothness as an axiom:

```
1       Axiom smoothness : forall model formula state,
2       entails (Labeling model state) formula ->
3       exists minimal_state,
4       entails (Labeling model minimal_state) formula /\
5       (PreferenceRel model minimal_state state \/
6       minimal_state = state) /\
7       In (States model) (MinimalElements model formula) minimal_state.
```

As a premise, we set with `entails` (line 2) that the formula holds in the current state. Then we declare the existence of a minimal state via the existential quantifier (line 3). We construct the properties of the minimal state: ensuring the formula also holds in the minimal state (line 4), the disjunction (lines 5-6) stating the minimal state is preferred or identical to the original state, and finally (line 6) that this minimal state is an element of `MinimalElements` for the given formula.

5. Formalization of the Representation Theorem

In this section, we present our formalization of Theorem 1 in Rocq. The implementation of the final theorem is given in the following:

```
1    Theorem klm_theorem :
2    forall (𝐊 : KnowledgeBase) (Γ : Ensemble Formula) (p q : Formula),
3    𝐊 ⊕ Γ ⊢ p |∼ q <-> 𝐊 ⊕ Γ ⊨ p |∼w q.
4    Proof.
5    intros 𝐊 Γ p q.
6    split.
7    - apply soundness_klm.
8    - apply completeness_klm.
9    Qed.
```

Line 4-9 state the theorem, and the remaining lines represent the proof, which is separated into subproofs for soundness and completeness. In the remainder of this section, we present the implementation of these proofs.

5.1. Soundness Proof

The soundness proof shows that all consequence relations that a cumulative model induces satisfy System C. This proof is formalized as a named `soundness_klm`, which calls subproofs for each System C rule. We implement a separate lemma for each rule of System C that proves that semantic consequences satisfy the corresponding postulate. Due to space limitations, we consider here only the implementation of some System C properties.

5.1.1. Reflexivity Rule

The lemma `soundness_reflexivity` shows that each consequence relation induced from a cumulative model satisfies reflexivity (Ref). Technically, (Ref) holds if a formula α holds in all minimal states of α, i.e., $\min(\mathcal{S}(\alpha), \prec) \subseteq \mathcal{S}(\alpha)$. In our Rocq formalization, this corresponds to the statement `model : formula |∼w formula`.

```
1    Lemma soundness_reflexivity :
2    forall (model : CumulModel) (formula : Formula),
3   model : formula |∼w formula.     // SemanticEntails model premise conclusion
4   Proof.
5   unfold SemanticEntails, MinimalElements.
6   intros model formula state [H_satisfies _].
7   exact H_satisfies.
8   Qed.
```

Lines 1-3 claim that reflexivity is satisfied by $\mid\!\sim_{w}$, while lines 4-8 develop the corresponding proof. The proof starts by making the components of the goals available to the prover by unfolding definitions. In Line 5, we unfold the definitions of "`model : formula |∼ formula`" (`SemanticEntails`) from Line 3 and then unfold `MinimalElements` (which appears implicitly by unfolding `SemanticEntails`). In Line 6, we fill the variables in unfolded definitions with elements from our assumption provided in Line 2. We obtain a proof goal `entails (Labeling model state) formula`, which corresponds to stating that the minimal states of `formula` in the cumulative model satisfy `formula`. With `exact H_satisfies` tells the prover to check the truth of the last goal by employing unification.

5.1.2. Right Weakening Rule

In lemma `soundness_RW` is for proving satisfaction of Right Weakening (RW). The proof uses the implication property `H_imp` to directly derive that the conclusion also holds in minimal states. Our proof goal is to show that when `model : r |∼w p` and `p -> q` hold, then `model : r |∼w q` also holds, demonstrating that an implication is preserved in the minimal state for `r`.

```
1   Lemma soundness_RW :
2   forall (model : CumulModel) (p q r : Formula),
3   (forall state, entails (Labeling model state) p ->
4   entails (Labeling model state) q) ->
5   model : r |∼w p ->
6   model : r |∼w q.
7   Proof.
8   unfold SemanticEntails, MinimalElements.
9   intros model p q r H_imp H_entails state H_minimal.
10  apply H_imp.
11  apply H_entails; assumption.
12  Qed.
```

We first unfold the definition of `SemanticEntails` and `MinimalElements` (line 8). After introducing all necessary variables and hypotheses (line 9), hypothesis `H_imp` states that `p` implies `q`, `H_entails` is the hypothesis that `p` typically follows from `r`, and `H_minimal` states that `state` is minimal for `r`.

Our proof goal is `entails (Labeling model state) q`. We first apply hypothesis `H_imp` (line 10) to change our proof goal to `entails (Labeling model state) p`. We then show that `p` holds in `state` by applying hypothesis `H_entails` to `H_minimal` (line 11). Since we already have `H_minimal` that corresponds exactly to this condition, we can directly solve the proof goal with `assumption` and complete the proof.

5.1.3. Cut Rule

The proof for Cut is formalized in lemma `soundness_Cut`. The challenge here lies in showing that `p` $\mid\!\sim_{w}$ `r` holds when both `p` $\mid\!\sim_{w}$ `q` and `p` $\wedge$ `q` $\mid\!\sim_{w}$ `r` hold. We want to show that in minimal states where `p` holds, `q` also holds, to then prove that `r` must also hold in these states.

```
1       Lemma soundness_Cut :
2       forall (model : CumulModel) (p q r : Formula),
3       model : (p ∧ q) |∼w r ->
4       model : p |∼w q ->
```

```
5   model : p |~w r.
6   Proof.
7   unfold SemanticEntails, MinimalElements.
8   intros model p q r H_conj_entails H_p_entails_q
9   state [H_p H_minimal].
10  assert (H_q : entails (Labeling model state) q).
11  apply H_p_entails_q; split; [exact H_p | exact H_minimal].
12  apply H_conj_entails.
13  split.
14  - rewrite entails_conjunction.
15  split; assumption.
16  - intro H_exists.
17  destruct H_exists as [state’ [H_conj’ H_pref]].
18  rewrite entails_conjunction in H_conj’.
19  destruct H_conj’ as [H_p’ _].
20  exfalso.
21  apply H_minimal.
22  exists state’.
23  split; assumption.
24  Qed.
```

We unfold the definitions of `SemanticEntails` and `MinimalElements` (line 7) and introduce variables and hypotheses (lines 8-9). We first assert that q holds in `state` and prove this by applying hypothesis `H_p_entails_q` (lines 10-11), showing that in minimal states where p holds, q also holds.

We then apply the conjunction assumption `H_conj_entails` (line 12), requiring proof that `state` is minimal for `(p ∧ q)`. We use `split` (line 13) to split into two subgoals. For the first subgoal, we use lemma `entails_conjunction` with `rewrite` (line 14) to resolve the conjunction. For the second subgoal, we prove minimality through contradiction (lines 16-22). From the fact that `(p ∧ q)` holds in `state’`, it follows that p also holds in `state’`, contradicting the assumption that `state` is minimal for p. This demonstrates a transitivity property of the cumulative consequence relation.

5.1.4. Structural Induction for Soundness

The main soundness theorem combines all individual lemmas through structural induction over the inductive type `CumulCons`:

```
1   Theorem soundness_klm :
2   forall (K : KnowledgeBase) (Γ : Ensemble Formula) (p q : Formula),
3   K ⊕ Γ ⊢ p |~ q -> K ⊕ Γ ⊨ p |~w q.
4   Proof.
5   intros K Γ p q H_cons.
6   unfold CumulativeModelEntails.
7   intros model H_satisfies.

9   induction H_cons.

11  - apply soundness_reflexivity.

13  - apply soundness_LLE with p.
14  + intros state.
15  unfold SatisfiesKnowledgeBases in H_satisfies.
16  destruct H_satisfies as [_ H_classical].
17  unfold SatisfiesClassicalKB in H_classical.
```

```
18      assert (H_equiv : entails (Labeling model state) (p ↔ q)).
19      { apply H_classical. exact H. }
20      apply entails_equivalence in H_equiv.
21      assumption.
22      + apply IHH_cons. exact H_satisfies.

24      - apply soundness_RW with p.
25      + intros state H_p.
26      unfold SatisfiesKnowledgeBases in H_satisfies.
27      destruct H_satisfies as [_ H_classical].
28      unfold SatisfiesClassicalKB in H_classical.
29      assert (H_impl : entails (Labeling model state) (p → q)).
30      { apply H_classical. exact H. }
31      simpl in H_impl.
32      apply H_impl. exact H_p.
33      + apply IHH_cons. exact H_satisfies.

35      - apply soundness_Cut with q.
36      + apply IHH_cons1. exact H_satisfies.
37      + apply IHH_cons2. exact H_satisfies.

39      - apply soundness_CM.
40      + apply IHH_cons1. exact H_satisfies.
41      + apply IHH_cons2. exact H_satisfies.

43      - unfold SatisfiesKnowledgeBases in H_satisfies.
44      destruct H_satisfies as [H_conditional _].
45      unfold SatisfiesConditionalKB in H_conditional.
46      apply H_conditional.
47      exact H.
48      Qed.
```

After introducing variables and unfolding `CumulativeModelEntails` (lines 5-7), we use structural induction over `H_cons` (line 9). This creates five separate proof goals corresponding to each System C rule.

Reflexivity (line 11) directly applies the proven lemma `soundness_reflexivity`.

LLE (lines 13-22) requires transforming the syntactic equivalence from the universe of reference Γ into semantic equivalence in the model. We extract the equivalence from Γ using `SatisfiesClassicalKB` (lines 15-16), use the model's satisfaction to obtain semantic equivalence (line 18), and apply `entails_equivalence` to convert to the form needed by `soundness_LLE` (line 20).

RW (lines 24-33) similarly transforms syntactic implication to semantic implication. We extract the implication from Γ (lines 26-27), apply the model's respect for classical truths in Γ (line 29), and use `simpl` to simplify the entailment definition (line 31).

Cut and CM (lines 35-41) are simpler since they work purely with cumulative consequence relations without requiring knowledge base transformations. Both apply the corresponding lemmas with their respective induction hypotheses (`IHH_cons1` and `IHH_cons2`).

Base (lines 43-47) handles the crucial new case where conditional assertions are directly used from the knowledge base $\mathbf{K}$. We unfold `SatisfiesKnowledgeBases` to access the conditional component (line 43), extract `SatisfiesConditionalKB` (line 45), and directly apply it to the conditional assertion from $\mathbf{K}$ (line 46).

This modular approach demonstrates that each System C rule is individually sound, and their combination through structural induction establishes the complete soundness of the system. The theorem proves that syntactic derivability in System C implies semantic validity in all cumulative models respecting the knowledge base.

5.2. Completeness Proof

For formalizing completeness of the KLM theorem, we show that every cumulative consequence relation defined by System C rules is representable by a cumulative model. We prove completeness through contradiction by constructing a canonical model that serves as a counterexample.

The main challenge lies in constructing a canonical model from syntactic information. The proof is close to the construction given by Kraus, Lehmann and Magidor [1]. Central to their proof is to use states as equivalence classes (see Section 2). However, due to technical reasons, our Rocq proof uses maximal consistent sets as states, which corresponds logically to equivalence classes.

5.2.1. Canonical Model Construction

We construct the canonical model using maximal consistent sets as states:

```
1   Definition CanonicalStates := Ensemble Formula.

3   Definition CanonicalPreferenceRel
4   (w1 w2 : CanonicalStates) : Prop :=
5   exists p, w1 ⊢ p /\ ~ (w2 ⊢ p).

7   Definition CanonicalModel : CumulModel :=
8   {|
9       States := CanonicalStates;
10      Labeling := fun w p => valuemaxf w p;
11      PreferenceRel := CanonicalPreferenceRel
12  |}.
```

Line 1 defines states as ensembles of formulas (maximal consistent sets). Lines 3-4 implement the preference relation from Definition 3.21 of Kraus, Lehmann and Magidor: w_1 is preferred over w_2 if there exists a formula p that is derivable in w_1 but not in w_2. Lines 6-11 construct the canonical model using library function `valuemaxf` for the labeling function.

5.2.2. Axioms for Completeness

We introduce several axioms to handle the complexity of the canonical model construction. The key axiom ensures existence of appropriate maximal consistent sets:

```
1   Axiom exists_maximal_consistent : forall 𝐊 Γ p q,
2   ~ (CumulCons 𝐊 Γ p q) ->
3   exists w,
4   maximal_consistent_set w /\ Γ ⊆ w /\
5   p ∈ w /\ ~ q ∈ w.
```

Lines 1-5 formalize the existence of maximal consistent sets: if q is not derivable from p under Γ, then there exists a maximal consistent set containing Γ and p but not q. This axiom establishes the bridge between syntactic non-derivability and semantic representation, enabling us to construct a counterexample in the main proof.

We also need the axiom `canonical_entails` to establish semantic interpretation:

```
1   Axiom canonical_entails :
2   forall (w : CanonicalStates) (p : Formula),
3   maximal_consistent_set w ->
4   entails (Labeling CanonicalModel w) p <-> p ∈ w.
```

Lines 1-4 establish that in maximal consistent set w, formula p holds semantically if and only if p is an element of w. This connects the semantic interpretation in the canonical model with set membership in maximal consistent sets.

5.2.3. Main Completeness Proof

The main completeness theorem demonstrates that semantic validity implies syntactic derivability through contradiction:

```
1   Theorem completeness_klm :
2   forall (K : KnowledgeBase) (Γ : Ensemble Formula) (p q : Formula),
3   K ⊕ Γ ⊨ p |~w q. -> K ⊕ Γ ⊢ p |~ q
4   Proof.
5   intros K Γ p q H_sem.
6   destruct (classic (CumulCons K Γ p q)) as [H_syn | H_not_syn].
7   - exact H_syn.
8   - assert (H_sem_check : K Γ |= p |~w q).
9   { exact H_sem. }

11  destruct (exists_maximal_consistent K Γ p q H_not_syn)
12  as [w [H_max [H_sub [H_p_in_w H_not_q_in_w]]]].

14  assert (H_satisfies :
15  SatisfiesKnowledgeBase CanonicalModel K Γ).
16  apply canonical_satisfies_kbs with (w := w); auto.

18  assert (H_minimal : In CanonicalStates
19  (MinimalElements CanonicalModel p) w).
20  apply minimal_elements_canonical; auto.

22  assert (H_entails_q :
23  entails (Labeling CanonicalModel w) q).
24  apply H_sem; auto.

26  assert (H_q_in_w : q ∈ w).
27  apply canonical_entails; auto.

29  contradiction.
30  Qed.
```

The proof uses classical logic to case split on whether `CumulCons` $\mathbf{K}\ \Gamma\ p\ q$ holds (line 6). If it holds (line 7), we are done. Otherwise (lines 8-24), we construct a maximal consistent set w using the axiom `exists_maximal_consistent` (lines 11-12), which guarantees that w contains Γ and p but not q.

We then show the canonical model satisfies the knowledge base (lines 14-16) and prove w is minimal for p (lines 18-20). Next, we apply the semantic validity assumption to derive that q must hold in w (lines 22-23), since w is a minimal state where p holds and the semantic relation requires q to hold in all such states. Using the canonical interpretation, we conclude $q \in w$ (lines 26-27). This directly contradicts the construction of w where $q \notin w$ (from line 11), completing the proof by contradiction. The contradiction shows that our assumption $\sim$`CumulCons` $\Gamma\ p\ q$ was false, therefore the syntactic derivability $\Gamma : p \mid\!\sim q$ must hold.

This completeness proof demonstrates that the canonical model construction provides exactly the counterexample needed to show that every semantically valid inference is syntactically derivable, completing the second direction of the KLM representation theorem.

6. Discussion

The formalization of the KLM representation theorem in Rocq demonstrates both the feasibility and challenges of mechanizing complex non-monotonic logical systems. Our approach successfully captures all essential components while revealing important insights about the formal verification of infinite mathematical structures.

The encoding of System C through inductive types proved highly effective, with each rule naturally represented as a constructor in `CumulCons`. This design enables structural induction for soundness proofs and provides clear correspondence to the mathematical definition by Kraus, Lehmann and Magidor [1]. The modular structure with separate lemmas for each rule facilitates understanding and future extensions. Cumulative models required more complex design decisions. Our choice of record types, bundling states, labelling functions, and preference relations, provides clean abstraction while maintaining accessibility. The definition of minimal elements through ensemble-forming functions integrates well with Rocq's type system, though it necessitated careful handling of the smoothness condition. The strategic use of axiomatization emerged as a crucial methodological choice. We distinguished between theoretically justified axioms and those abstracting complex constructions. The smoothness condition is axiomatized based on its automatic satisfaction in propositional logic with finite descending chains. The canonical model axioms abstract the infinite construction that would otherwise require exponentially complex proofs. This approach, inspired by established techniques in modal logic [8], allows focus on the theorem's core logical content while maintaining formal rigor.

Interactive proving with Rocq revealed both the power and limitations of current proof assistants. Tactics like `induction`, `apply`, and `destruct` enable elegant proof construction, but type errors and unclear documentation create significant learning barriers. The external library [7] proved essential for propositional logic foundations, though its complexity required careful integration.

The restriction to propositional logic serves multiple purposes: it simplifies the smoothness condition, reduces the complexity of maximal consistent set construction, and focuses attention on the representation theorem's essential structure. This limitation is strategic rather than fundamental, as the modular design supports future extension to predicate logic. Our completeness proof through canonical model construction follows the approach of Kraus, Lehmann and Magidor while adapting to Rocq's constructive foundation. The use of maximal consistent sets as states, rather than equivalence classes, reflects practical considerations in formal verification while maintaining mathematical equivalence. The soundness direction proved more straightforward, with each System C rule verified independently through structural induction. The formalization reveals important connections between different areas of logic and computer science. The use of inductive types for inference systems, record types for mathematical structures, and axiomatic abstraction for complex constructions provides a methodology applicable to other logical systems.

7. Conclusion

The representation theorem for cumulative reasoning by Kraus, Lehmann and Magidor [1] forms the theoretical foundation of this work, describing the equivalence between syntactic System C rules and semantic cumulative models. This equivalence is central to non-monotonic reasoning, enabling knowledge representation with exceptions like "birds fly, but penguins don't" - essential for AI applications such as expert systems.

We have achieved our main goal of creating a complete formalization of the KLM theorem in Rocq. The formalization encompasses all components: System C with its five rules was completely formalized on the syntactic level, while cumulative models with states, labeling functions, and preference relations were defined semantically using maximal consistent sets as states. The soundness proof in theorem `soundness_klm` demonstrates that every System C derivable consequence is semantically valid, secured through structural induction over `CumulCons` with separate lemmas for each rule. The completeness proof in theorem `completeness_klm` shows that every semantically valid consequence is syntactically derivable, using contradiction with a canonical model construction. Both proofs were verified by Rocq, confirming their correctness.

Here, we restrict ourselves to propositional logic, which keeps the complexity manageable while preserving essential aspects of the representation theorem. The modular proof structure facilitates future extensions, and the axiomatization of complex constructions allows for a focus on core concepts without losing theoretical rigor. The formalization provides several key insights reflecting the challenges of mechanizing complex logical systems. The infinite size of the canonical model poses the central challenge, leading to our axioms that circumvent this complexity. We distinguished between theoretically justified axioms and those abstracting complex constructions, enabling focus on the KLM theorem's core concepts while maintaining formal rigor. The formalization successfully covers the complete KLM theorem with machine-verified correctness for the propositional case. All five System C rules are precisely encoded, cumulative models are formally represented, and the bidirectional equivalence between syntax and semantics is proven. The modular structure enables extensions, while axiomatization keeps the approach accessible.

Researchers in non-monotonic reasoning have explored automated reasoning approaches before. Related work includes KLMLean and KLMLean 2.0 by Pozzato [9] and Giordano et al. [10], which implement tableau-based theorem proving for KLM logics in SICStus Prolog using lean tableau methodology. Very general implementations of the proof theory of conditional logics can be found in the work by, e.g., Girlando [11]. Another line of work involves implementing KLM-style reasoning via translation into a system based on a different logic [12]. In our understanding, these works are automated reasoning systems that permit KLM-style reasoning rather than formal verification of representation theorems in proof assistants. We are not aware of any formalization attempts of representation theorems for non-monotonic reasoning approaches in proof assistants.

Note that our formalization creates a verified library for KLM-style reasoning that can be extended to other systems and applied to practical AI systems through code extraction. The strategic balance between axiomatic abstraction and constructive proof provides a methodology for similar formalizations, showing how to handle infinite constructions while maintaining focus on core theoretical concepts.

In future work, we aim to formalize more theorems from knowledge representation and reasoning. The closest theorem to the work here is the representation theorem by KLM for System P, which is an extension of System P by one additional rule [1]:

$$\text{(OR)}\quad \frac{a \mid\!\sim c \quad b \mid\!\sim c}{a \vee b \mid\!\sim c}$$

This rule states that if c typically follows from both a and b, then c also follows from $a \vee b$. Semantically, System P requires preferential models with transitive preference relations, unlike System C, which only involves irreflexivity. The modular approach from this paper allows for the implementation of this additional rule easily. For that we would add an additional constructor in `CumulCons` (see Section 4.1:

```
| OR : forall 𝐊 Γ a b c : Formula,
  CumulCons 𝐊 Γ a c ->
  CumulCons 𝐊 Γ b c ->
  CumulCons 𝐊 Γ (a ∨ b) c
```

Many other definitions also require only minor adjustments. A challenge is to implement a semantic representation of System P. This is by preferential models (which differ from cumulative models). Part of this challenge is that known canonical constructions for preferential models may produce infinite objects, even for finite languages [5]. We refrain from stating further details here and leave further investigation open for future work.

Acknowledgments

We thank the reviewers for their comments, which helped us improve the paper.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Kraus, D. Lehmann, M. Magidor, Nonmonotonic reasoning, preferential models, and cumulative logics, Artificial Intelligence 44 (1990) 167–207. doi:10.1016/0004-3702(90)90101-5.

[2] D. M. Gabbay, C. J. Hogger, J. A. Robinson, Handbook of Logic in Artificial Intelligence and Logic Programming, OXFORD UNIV PR, 1993.

[3] G. Brewka, J. Dix, K. Konolige, Nonmonotonic Reasoning: An Overview, volume 73 of *CSLI Lecture Notes*, CSLI Publications, Stanford, CA, 1997.

[4] D. M. Gabbay, Theoretical foundations for non-monotonic reasoning in expert systems, in: K. R. Apt (Ed.), Proceedings of the conference on Logics and Models of Concurrent Systems - Conference proceedings, volume 13 of *NATO ASI Series*, Springer, 1984, pp. 439–457. doi:10.1007/978-3-642-82453-1_15.

[5] D. Lehmann, M. Magidor, What does a conditional knowledge base entail?, Artificial intelligence 55 (1992) 1–60.

[6] C. Paulin-Mohring, Introduction to the calculus of inductive constructions, in: B. W. Paleo, D. Delahaye (Eds.), All about Proofs, Proofs for All, volume 55 of *Studies in Logic (Mathematical Logic and Foundations)*, College Publications, 2015.

[7] D. Guo, W. Yu, A comprehensive formalization of propositional logic in coq: Deduction systems, meta-theorems, and automation tactics, Mathematics 11 (2023). doi:10.3390/math11112504.

[8] E. N. Zalta, Basic concepts in modal logic, Center for the Study of Language and Information (1995).

[9] G. L. Pozzato, Proof methods for conditionaland preferential logics, 2007.

[10] L. Giordano, V. Gliozzi, G. L. Pozzato, KLMLean 2.0: A theorem prover for KLM logics of nonmonotonic reasoning, in: N. Olivetti (Ed.), Proceedings of the 16th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2007), volume 4548 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 238–244. doi:10.1007/978-3-540-73099-6_19.

[11] M. Girlando, L. Straßburger, MOIN: A nested sequent theorem prover for intuitionistic modal logics (system description), in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Proceedings of the 10th International Joint Conference on Automated Reasoning (IJCAR 2020), volume 12167 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 398–407. doi:10.1007/978-3-030-51054-1_25.

[12] X. Parent, C. Benzmüller, Conditional normative reasoning as a fragment of HOL, J. Appl. Non Class. Logics 34 (2024) 561–592. doi:10.1080/11663081.2024.2386917.