Theorem proving for Lewis Logics of Counterfactual Reasoning*

Marianna Girlando[1], Björn Lellmann[2], Nicola Olivetti[3], Stefano Pesce[4], and Gian Luca Pozzato[4] (✉)

[1] Inria Saclay, LIX - École Polytechnique, France - marianna.girlando@inria.fr
[2] Technische Universität Wien, Austria - lellmann@logic.at
[3] Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296, 13397, Marseille, France - nicola.olivetti@univ-amu.fr
[4] Dipartimento di Informatica, Universitá di Torino, Italy - {gianluca.pozzato@,stefano.pesce356@edu.}unito.it

Abstract. We present tuCLEVER, a theorem prover for the strongest conditional logics of counterfactual reasoning introduced by Lewis in the seventies. tuCLEVER implements some hypersequent calculi recently introduced for the system $\mathbb{VTU}$ and its main extensions. tuCLEVER is inspired by the methodology of leanT^AP and it is implemented in Prolog. Preliminary experimental results show that the performances of tuCLEVER are promising.

1 Introduction

Conditional logics are extensions of classical logic by a *conditional* operator $\Box\!\!\rightarrow$. They have a long history going back, e.g., to the works of Stalnaker, Lewis, Nute, Chellas, Burgess, Pollock in the 60's-70's [26, 18, 19, 5, 4]. Conditional logics have since found an interest in several fields of knowledge representation, from reasoning about prototypical properties and nonmonotonic reasoning [16] to modeling belief change. A successful attempt to relate conditional logic and belief update (as opposite to belief revision) was carried out by Grahne [13], who established a precise mapping between belief update operators and Lewis' logic $\mathbb{VCU}$, an extension of the basic system $\mathbb{VTU}$ mentioned above. The relation is expressed by the so-called *Ramsey's Rule*:

$A \circ B \rightarrow C$ holds if and only if $A \rightarrow (B \Box\!\!\rightarrow C)$ holds

where the operator $\circ$ is any *update* operator satisfying Katsuno and Mendelzon's postulates [15], that are considered the "core" properties for any concrete, plausible belief-update operator. The relation means that C is entailed by "A updated by B" if and only if the conditional $B \Box\!\!\rightarrow C$ is entailed by A. In this sense it can be said that the conditional $B \Box\!\!\rightarrow C$ expresses an hypothetical update of a piece of information A. They have even been also adopted to reason about access control policies [9].

One of the most important contribution to conditional logic is due to Lewis. In his seminal work [18], he proposed a formalization of conditional logics to capture hypothetical conditionals. His aim was to represent conditional sentences that cannot be captured by material implication and, in particular, *counterfactuals*, e.g. conditionals of the form "if A were the case, then B would be the case", where A is false. In [18] Lewis introduced a family of conditional logics semantically characterized by *sphere models*, in which each world x is equipped with a set of nested sets of worlds $\mathsf{SP}(x)$. Each set in $\mathsf{SP}(x)$ is called a *sphere*: the intuition is that according to x, worlds in inner spheres are more plausible than worlds belonging only to outer spheres.

Lewis takes as primitive the *comparative plausibility* operator $\preccurlyeq$, with a formula $A \preccurlyeq B$ meaning "A is at least as plausible as B". The conditional $A \Box\!\!\rightarrow B$ is "A is impossible or $A \wedge \neg B$ is less plausible than $A \wedge B$" (where the latter case can be simplified to "$A \wedge \neg B$ is less plausible than A"). Vice versa, $\preccurlyeq$ can be defined in terms of $\Box\!\!\rightarrow$.

Here we consider the logics of Lewis' family satisfying two natural properties for hypothetical reasoning and belief change modelling:

- *Uniformity*: all worlds have the same set of accessible worlds, where the worlds accessible from a world x are those belonging to any sphere $\alpha \in \mathsf{SP}(x)$;
- *Total reflexivity*: every world x belongs to some sphere $\alpha \in \mathsf{SP}(x)$.

The basic logic is $\mathbb{VTU}$. We also consider some of its extensions, including the above mentioned $\mathbb{VCU}$. It is worth mentioning that equivalent logics are those of Comparative Concept Similarity studied in the context of ontologies [25]. These logics contain a connective $\leftleftarrows$, which allows to express, e.g,

$$PicassoPainting \sqsubseteq BraquePainting \leftleftarrows GiottoPainting$$

asserting that "Picasso's paintings are more similar to Braque's paintings than to Giotto's ones". The semantics is provided in terms of Distance Space Models, defined as a set of worlds equipped with a distance function. It turns out that the basic logic of Comparative Concept Similarity coincides with Lewis' logic $\mathbb{VWU}$, an extension of the basic system $\mathbb{VTU}$ with a property known as *weak centering*, and the one defined by "minspace" Distance Models coincides with $\mathbb{VCU}$, so that Distance Space Models provide an alternative simple and natural semantics for conditional logics with uniformity [25, 1]. All these logics contain modal logic $\mathsf{S5}$ as a fragment: $\Box A$ can be defined as $\bot \preccurlyeq \neg A$ (or $\neg A \Box\!\!\rightarrow \bot$).

In previous works [24, 10] we proposed some internal sequent calculi for Lewis' logics *without* Uniformity. Internal calculi are proof methods where each configuration of a derivation corresponds to a formula of the corresponding logic, in contrast to external calculi which make use of extra-logical elements (such as labels, terms and relations on them). We implemented these calculi with the theorem prover VINTE [12]. However, the mere sequent structure is not powerful enough to capture conditional logic with Uniformity[5]. In [11] we proposed the first proof systems for $\mathbb{VTU}$ and its extensions in the form of *hypersequent* calculi. Hypersequents are finite sets of sequents; and in these calculi sequents are "extended" by a structural connective $\langle . \rangle$, representing disjunctions of $\Diamond$-formulae.

In this work we present a Prolog implementation of the hypersequent calculi for $\mathbb{VTU}$ and its extensions [11]. The program, called tuCLEVER (Total reflexivity and Uniformity Conditional LEwis logics theorem proVER) is, to the best of our knowledge, the only existing prover for conditional logics with Uniformity[6]. The conception of tuCLEVER is inspired by the methodology of leanT^AP [3]. The idea is that each axiom or rule of the sequent calculi is implemented by a single Prolog clause. No ad-hoc data structure is used. The resulting code is therefore simple and compact: the implementation of tuCLEVER for the basic system $\mathbb{VTU}$ consists of only 3 predicates, 21 clauses and 118 lines of code.

The prover provides a decision procedure for the respective logics: it implements the invertible version of the calculi in [11], where the principal formula or structure is kept in the premises of each rule (similarly to the so-called *kleened* calculi). In this way, termination is obtained by simply avoiding *redundant* applications of the rules.

Even if a set of benchmark formulae does not exist, the experimental results obtained so far show that the performances of tuCLEVER are promising. Being the unique theorem prover for conditional logics with Uniformity, tuCLEVER is not directly comparable with any other prover for conditional logics. Nonetheless, we show that on sets of formulae provable in other (weaker) conditional logics and on randomly generated formulas, the performances of tuCLEVER are surprisingly better than the ones of other provers for conditional logics, notably VINTE [11] which covers weaker logics of the Lewis family. Whether this fact depends on the strength of the logic implemented by tuCLEVER, on the features of the calculi, or on the implementation is an open question.

The program tuCLEVER, as well as all the Prolog source files, are available for free usage and download at http://193.51.60.97:8000/tuclever/.

The article is organized as follows. Section 2 introduces the axioms and the models of the logics under scope. In Section 3 we recall the hypersequent calculi from [11]. Section 4 presents the design of tuCLEVER, and Section 5 treats its performances.

2 Lewis' Conditional Logics

We consider the *conditional logics* of [18]. The set of *conditional formulae* is given by

$$A ::= p \mid \bot \mid \top \mid \neg A \mid A \rightarrow A \mid A \wedge A \mid A \vee A \mid A \preccurlyeq A$$

[5] Conditional logics without Uniformity are PSPACE complete, whereas conditional logics with Uniformity (but without Absoluteness) are EXPTIME complete [8].

[6] The only possible exception is the theorem prover CSLLean [2] which implements a calculus for the logic of Comparative Concept Similarity over minspaces, which is equivalent to logic $\mathbb{VCU}$.

where $p \in \mathcal{V}$ is a propositional variable. Intuitively, a formula $A \preccurlyeq B$ is interpreted as "A is at least as plausible as B". Lewis' *counterfactual implication* $\Box\!\!\rightarrow$ is defined by $A \Box\!\!\rightarrow B \;\equiv\; (\bot \preccurlyeq A) \vee \neg((A \wedge \neg B) \preccurlyeq A)$, whereas the *outer modality* $\Box$ is defined by $\Box A \equiv (\bot \preccurlyeq \neg A)$. The logics we consider are defined as follows:

Definition 1. *A model is a triple* $\langle W, \mathsf{SP}, [\![.]\!]\rangle$*, consisting of a non-empty set* W *of elements, called* worlds*, a mapping* $\mathsf{SP} : W \rightarrow 2^{2^W}$*, and a propositional valuation* $[\![.]\!] : \mathcal{V} \rightarrow 2^W$*. Elements of* $\mathsf{SP}(x)$ *are called* spheres*. We assume the following conditions:*

- *For every* $\alpha \in \mathsf{SP}(w)$ *we have* $\alpha \neq \emptyset$ *(non-emptiness)*
- *For every* $\alpha, \beta \in \mathsf{SP}(w)$ *we have* $\alpha \subseteq \beta$ *or* $\beta \subseteq \alpha$ *(sphere nesting)*
- *For all* $w \in W$ *we have* $\mathsf{SP}(w) \neq \emptyset$ *(normality)*
- *For all* $w \in W$ *we have* $w \in \bigcup \mathsf{SP}(w)$ *(total reflexivity)*
- *For all* $w, v \in W$ *we have* $\bigcup \mathsf{SP}(w) = \bigcup \mathsf{SP}(v)$ *(uniformity)*

The valuation $[\![.]\!]$ *is extended to all formulae as follows:*

$$
\begin{aligned}
&[\![\bot]\!] = \emptyset\\
&[\![\top]\!] = W\\
&[\![\neg A]\!] = W - [\![A]\!]\\
&[\![A \wedge B]\!] = [\![A]\!] \cap [\![B]\!]\\
&[\![A \vee B]\!] = [\![A]\!] \cup [\![B]\!]\\
&[\![A \rightarrow B]\!] = (W - [\![A]\!]) \cup [\![B]\!]\\
&[\![A \preccurlyeq B]\!] = \{w \in W \mid \forall \alpha \in \mathsf{SP}(w).\ \textit{if}\ [\![B]\!] \cap \alpha \neq \emptyset,\ \textit{then}\ [\![A]\!] \cap \alpha \neq \emptyset\}
\end{aligned}
$$

Validity and satisfiability of formulae in a class of models are defined as usual. The logic $\mathbb{VTU}$ *is the set of formulae valid in all models.*

We can add to the syntax the conditional operator $A \Box\!\!\rightarrow B$, since it will be used in formulas handled by the prover. $A \Box\!\!\rightarrow B$ can be defined in terms of Lewis' plausibility $\preccurlyeq$ as recalled in the Introduction, and its truth condition is as follows:

$$
\begin{aligned}
[\![A \Box\!\!\rightarrow B]\!] = \{w \in W \mid \text{either } \textstyle\bigcup \mathsf{SP}(w) \cap [\![A]\!] = \emptyset \text{ or } \exists \alpha \in \mathsf{SP}(w) \text{ such that}\\
\alpha \cap [\![A]\!] \neq \emptyset \text{ and } \alpha \cap [\![A]\!] \subseteq [\![B]\!]\}.
\end{aligned}
$$

Extensions of $\mathbb{VTU}$ are defined by adding conditions on the class of models:

- For all $\alpha \in \mathsf{SP}(w)$ we have $w \in \alpha$ *(weak centering)*
- For all $w \in W$ we have $\{w\} \in \mathsf{SP}(w)$ *(centering)*
- For all $w, v \in W$ we have $\mathsf{SP}(w) = \mathsf{SP}(v)$ *(absoluteness)*

Extensions of $\mathbb{VTU}$ are denoted by concatenating letters for these properties: $\mathbb{W}$ for weak centering, $\mathbb{C}$ for centering, and $\mathbb{A}$ for absoluteness. We consider[7]:

$\mathbb{VTU}$	$\mathbb{VTA}$: $\mathbb{VTU}$ + *absoluteness*
$\mathbb{VWU}$: $\mathbb{VTU}$ + *weak centering*	$\mathbb{VWA}$: $\mathbb{VTA}$ + *weak centering*
$\mathbb{VCU}$: $\mathbb{VTU}$ + *centering*	$\mathbb{VCA}$: $\mathbb{VTA}$ + *centering*

[7] Observe that $\mathbb{VTA}$ + *weak centering* collapses to $\mathsf{S5}$, and that $\mathbb{VTA}$ + *centering* collapses to classical logic.

(CPR) $\dfrac{\vdash B \to A}{\vdash A \preccurlyeq B}$	(CPA) $(A \preccurlyeq A \vee B) \vee (B \preccurlyeq A \vee B)$
(TR) $(A \preccurlyeq B) \wedge (B \preccurlyeq C) \to (A \preccurlyeq C)$	(CO) $(A \preccurlyeq B) \vee (B \preccurlyeq A)$
(N) $\neg(\bot \preccurlyeq \top)$	(T) $(\bot \preccurlyeq \neg A) \to A$
(U1) $\neg(\bot \preccurlyeq A) \to (\bot \preccurlyeq (\bot \preccurlyeq A))$	(U2) $(\bot \preccurlyeq \neg A) \to (\bot \preccurlyeq \neg(\bot \preccurlyeq \neg A))$
(W) $A \to (A \preccurlyeq \top)$	(C) $(A \preccurlyeq \top) \to A$
(A1) $(A \preccurlyeq B) \to \big(\bot \preccurlyeq \neg(A \preccurlyeq B)\big)$	(A2) $\neg(A \preccurlyeq B) \to \big(\bot \preccurlyeq (A \preccurlyeq B)\big)$

$$\mathcal{A}_{\mathbb{VTU}} := \{(\mathsf{CPR}), (\mathsf{CPA}), (\mathsf{TR}), (\mathsf{CO}), (\mathsf{N}), (\mathsf{T}), (\mathsf{U1}), (\mathsf{U2})\}$$

$$\mathcal{A}_{\mathbb{VWU}} := \mathcal{A}_{\mathbb{VTU}} \cup \{(\mathsf{W})\} \qquad \mathcal{A}_{\mathbb{VCU}} := \mathcal{A}_{\mathbb{VTU}} \cup \{(\mathsf{W}), (\mathsf{C})\} \qquad \mathcal{A}_{\mathbb{VTA}} := \mathcal{A}_{\mathbb{VTU}} \cup \{(\mathsf{A1}), (\mathsf{A2})\}$$

$$\mathcal{A}_{\mathbb{VWA}} := \mathcal{A}_{\mathbb{VTU}} \cup \{(\mathsf{W}), (\mathsf{A1}), (\mathsf{A2})\} \quad \mathcal{A}_{\mathbb{VCA}} := \mathcal{A}_{\mathbb{VTU}} \cup \{(\mathsf{W}), (\mathsf{C}), (\mathsf{A1}), (\mathsf{A2})\}$$

Table 1. Lewis' logics and axioms.

These logics can be characterized by axioms in a Hilbert-style system [18, Chp. 6]. The modal axioms in the language with only the comparative plausibility operator are given in Table 1 ($\vee$ and $\wedge$ bind stronger than $\preccurlyeq$). Propositional axioms and rules are standard.

3 Hypersequent Calculi for Lewis' Logics

We recall hypersequent calculi for $\mathbb{VTU}$ and extensions from [11]. These calculi are based on hypersequents, namely non-empty, finite multisets of *extended* sequents. The extended sequents contain in the succedent a structural connective $\langle . \rangle$ interpreting possible formulae.

Formally, we define:

- a *conditional block*, which is a tuple $[\Sigma \lhd C]$ containing a finite multiset Σ of formulae and a single formula C;
- a *transfer block*, which is a finite multiset of formulae, written $\langle \Theta \rangle$;
- an *extended sequent*, which is a tuple $\Gamma \Rightarrow \Delta$ consisting of a finite multiset Γ of formulae and a finite multiset Δ containing formulae, conditional blocks, and transfer blocks;
- an *extended hypersequent*, which is a finite multiset containing extended sequents, written $\Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_n \Rightarrow \Delta_n$.

The rules of the calculi introduced in [11] are shown in Fig. 1. Given $\Diamond A \equiv \neg(\bot \preccurlyeq A)$, the formula interpretation of an extended sequent and of an extended hypersequent are given by:

$$\iota_e(\Gamma \Rightarrow \Delta, [\Sigma_1 \lhd C_1], \ldots, [\Sigma_n \lhd C_n], \langle \Theta_1 \rangle, \ldots, \langle \Theta_m \rangle) := \bigwedge \Gamma \to \bigvee \Delta \vee \bigvee_{i=1}^{n} \bigvee_{B \in \Sigma_i} (B \preccurlyeq C_i) \vee \bigvee_{j=1}^{m} \Diamond(\bigvee \Theta_j)$$

$$\iota_e(\Gamma_1 \Rightarrow \Delta_1 \mid \ldots \mid \Gamma_n \Rightarrow \Delta_n) \quad := \quad \Box\, \iota_e(\Gamma_1 \Rightarrow \Delta_1) \vee \cdots \vee \Box\, \iota_e(\Gamma_n \Rightarrow \Delta_n).$$

Theorem 2 (Soundness and Completeness). *For A formula, $A \in \mathcal{L}$ if and only if $\mathsf{SH}^i_{\mathcal{L}} \vdash \Rightarrow A$.*

The calculi of Fig. 1 can be used to define a decision procedure for the corresponding logics.

$$\frac{}{\mathcal{G} \mid \Omega, \bot \Rightarrow \Theta}\ \bot_L \qquad \frac{}{\mathcal{G} \mid \Omega \Rightarrow \Theta, \top}\ \top_R \qquad \frac{}{\mathcal{G} \mid \Omega, p \Rightarrow \Theta, p}\ \mathsf{init} \qquad \frac{\mathcal{G} \mid \Omega, \neg A \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, \neg A \Rightarrow \Theta}\ \neg^i_L$$

$$\frac{\mathcal{G} \mid \Omega, A \Rightarrow \Theta, \neg A}{\mathcal{G} \mid \Omega \Rightarrow \Theta, \neg A}\ \neg^i_R \qquad \frac{\mathcal{G} \mid \Omega, A \wedge B, A, B \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, A \wedge B \Rightarrow \Theta}\ \wedge^i_L \qquad \frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \vee B, A, B}{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \vee B}\ \vee^i_R$$

$$\frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \wedge B, A \quad \mathcal{G} \mid \Omega \Rightarrow \Theta, A \wedge B, B}{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \wedge B}\ \wedge^i_R \qquad \frac{\mathcal{G} \mid \Omega, A \vee B, A \Rightarrow \Theta \quad \mathcal{G} \mid \Omega, A \vee B, B \Rightarrow \Theta}{\mathcal{G} \mid \Omega, A \vee B \Rightarrow \Theta}\ \vee^i_L$$

$$\frac{\mathcal{G} \mid \Omega, A \to B, B \Rightarrow \Theta \quad \mathcal{G} \mid \Omega, A \to B \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, A \to B \Rightarrow \Theta}\ \to^i_L \qquad \frac{\mathcal{G} \mid \Omega, A \Rightarrow \Theta, A \to B, B}{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \to B}\ \to^i_R$$

$$\frac{\mathcal{G} \mid \Sigma \Rightarrow \Pi, A \preccurlyeq B, [A \lhd B]}{\mathcal{G} \mid \Sigma \Rightarrow \Pi, A \preccurlyeq B}\ \preccurlyeq^i_R \qquad \frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma \lhd A] \mid A \Rightarrow \Sigma}{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma \lhd A]}\ \mathsf{jump}^i$$

$$\frac{\mathcal{G} \mid \Omega, C \preccurlyeq D \Rightarrow \Theta, [D, \Sigma \lhd A] \quad \mathcal{G} \mid \Omega, C \preccurlyeq D \Rightarrow \Theta, [\Sigma \lhd A], [\Sigma \lhd C]}{\mathcal{G} \mid \Omega, C \preccurlyeq D \Rightarrow \Theta, [\Sigma \lhd A]}\ \preccurlyeq^i_L$$

$$\frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1, \Sigma_2 \lhd A], [\Sigma_2 \lhd B] \quad \mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1 \lhd A], [\Sigma_1, \Sigma_2 \lhd B]}{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1 \lhd A], [\Sigma_2 \lhd B]}\ \mathsf{com}^i$$

$$\frac{\mathcal{G} \mid \Sigma, A \preccurlyeq B \Rightarrow \Pi, \langle \Theta \rangle \mid A \Rightarrow \Theta \quad \mathcal{G} \mid \Sigma, A \preccurlyeq B \Rightarrow \Pi, \langle \Theta, B \rangle}{\mathcal{G} \mid \Sigma, A \preccurlyeq B \Rightarrow \Pi, \langle \Theta \rangle}\ \mathsf{T}^i$$

$$\frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \bot \rangle}{\mathcal{G} \mid \Gamma \Rightarrow \Delta}\ \mathsf{in}^i_{\mathsf{trf}} \qquad \frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle \mid \Sigma \Rightarrow \Theta, \Pi}{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle \mid \Sigma \Rightarrow \Pi}\ \mathsf{jump}^i_U \qquad \frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle, \Theta}{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle}\ \mathsf{jump}^i_T$$

$$\frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, [\Sigma \lhd A], \Sigma}{\mathcal{G} \mid \Gamma \Rightarrow \Delta, [\Sigma \lhd A]}\ \mathsf{W}^i \qquad \frac{\mathcal{G} \mid \Gamma, C \preccurlyeq D, C \Rightarrow \Delta \quad \mathcal{G} \mid \Gamma, C \preccurlyeq D \Rightarrow D, \Delta}{\mathcal{G} \mid \Gamma, C \preccurlyeq D \Rightarrow \Delta}\ \mathsf{C}^i$$

$$\frac{\mathcal{G} \mid \Gamma, A \preccurlyeq B \Rightarrow \Delta \mid \Sigma, A \preccurlyeq B \Rightarrow \Pi}{\mathcal{G} \mid \Gamma, A \preccurlyeq B \Rightarrow \Delta \mid \Sigma \Rightarrow \Pi}\ \mathsf{abs}^i_L \qquad \frac{\mathcal{G} \mid \Gamma \Rightarrow A \preccurlyeq B, \Delta \mid \Sigma \Rightarrow A \preccurlyeq B, \Pi}{\mathcal{G} \mid \Gamma \Rightarrow A \preccurlyeq B, \Delta \mid \Sigma \Rightarrow \Pi}\ \mathsf{abs}^i_R$$

$$\mathsf{PC} = \{\bot_L, \top_R, \mathsf{init}, \neg^i_L, \neg^i_R, \wedge^i_L, \wedge^i_R, \vee^i_L, \vee^i_R, \to^i_L, \to^i_R\}$$

$$\mathsf{SH}^i_{\mathbb{VTU}} = \mathsf{PC} \cup \{\preccurlyeq^i_R, \preccurlyeq^i_L, \mathsf{com}^i, \mathsf{jump}^i, \mathsf{T}^i, \mathsf{in}^i_{\mathsf{trf}}, \mathsf{jump}^i_U, \mathsf{jump}^i_T\}$$

$$\mathsf{SH}^i_{\mathbb{VWU}} = \mathsf{SH}^i_{\mathbb{VTU}} \cup \{\mathsf{W}^i\} \qquad \mathsf{SH}^i_{\mathbb{VCU}} = \mathsf{SH}^i_{\mathbb{VWU}} \cup \{\mathsf{C}^i\} \qquad \mathsf{SH}^i_{\mathbb{VTA}} = \mathsf{SH}^i_{\mathbb{VTU}} \cup \{\mathsf{abs}^i_L, \mathsf{abs}^i_R\}$$

$$\mathsf{SH}^i_{\mathbb{VWA}} = \mathsf{SH}^i_{\mathbb{VWU}} \cup \{\mathsf{abs}^i_L, \mathsf{abs}^i_R\} \qquad \mathsf{SH}^i_{\mathbb{VCA}} = \mathsf{SH}^i_{\mathbb{VCU}} \cup \{\mathsf{abs}^i_L, \mathsf{abs}^i_R\}$$

Fig. 1. The hypersequent calculi for $\mathbb{VTU}$ and its extensions.

4 Design of tuCLEVER

In this section we present a Prolog implementation of the hypersequent calculi recalled in Section 3. The program, called tuCLEVER, is inspired by the “lean” methodology of leanT^AP, even if it does not follow its style in a rigorous manner.

The program comprises a set of clauses, each of them implementing a sequent rule or axiom of the calculi. tuCLEVER implements a *cumulative*, or *kleened*, version of the calculi $\mathsf{SH}^i_{\mathcal{L}}$, in which each rule keeps its principal formula in the premises. In this way, termination is ensured in an immediate way by checking redundancy of the rules applications. The proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism.

tuCLEVER represents an hypersequent as a Prolog list of extended sequents. In turn, an extended sequent is represented as a pair of Prolog lists `[Gamma,Delta]`, where `Gamma` and `Delta` represent the left-hand and the right-hand side of the extended sequent, respectively. An extended sequent contains conditional blocks and transfer blocks. A conditional block $[\Sigma \lhd C]$ is a pair `[Sigma,C]`, i.e. a Prolog list with two elements, where `Sigma` is a list of formulas. A transfer block $\langle\Theta\rangle$ is implemented by a term `transfer Theta`, where again `Theta` is a Prolog list. Symbols $\top$ and $\bot$ are represented by constants `true` and `false`, respectively, whereas connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\preccurlyeq$, and $\Box\!\!\rightarrow$ are represented by `-`, `and`, `or`, `->`, `<`, and `=>`. Propositional variables are represented by Prolog atoms. As an example, the sequent $A, \neg B \vee C \Rightarrow A \wedge C, D, A \rightarrow B, \langle\bot\rangle, [A \preccurlyeq C, B \lhd A \vee C]$ is represented by the list: $[[\mathtt{a}, -\mathtt{b}\ \mathtt{or}\ \mathtt{c}], [\mathtt{a}\ \mathtt{and}\ \mathtt{c}, \mathtt{d}, \mathtt{a} - > \mathtt{b}, \mathrm{transfer}[\mathtt{false}], [[\mathtt{a} < \mathtt{c}, \mathtt{b}], \mathtt{a}\ \mathrm{or}\ \mathtt{c}]]]$.

The hypersequent calculi are implemented for each logic by the predicate

```
prove(Hypersequent,ProofTree).
```

This predicate succeeds if and only if the hypersequent represented by the list `Hypersequent` is derivable. When it succeeds, the output term `ProofTree` matches with a representation of the derivation found by the prover. For instance, in order to prove the formula $(A \preccurlyeq A \vee B) \vee (B \preccurlyeq A \vee B)$ in $\mathbb{VTU}$, one queries tuCLEVER with the goal: `prove([[],[(a < a or b) or (b < a or b)]],ProofTree).` Each clause of `prove` implements an axiom or rule of the calculi in Figure 1. To search a derivation, tuCLEVER proceeds as follows. First of all, if the hypersequent is an instance of either $\bot_L$ or $\top_R$ or init, the goal will succeed immediately by using one of the following clauses for the axioms:

```
prove(Hypersequent,tree(...)) :-
     member([Gamma,Delta],Hypersequent),member(false,Gamma),!.
prove(Hypersequent,tree(...)) :-
     member([Gamma,Delta],Hypersequent),member(true,Delta),!.
prove(Hypersequent,tree(...)) :- member([Gamma,Delta],Hypersequent),
     member(X,Gamma),member(X,Delta),atom(X),!.
```

If the hypersequent is not an instance of the ending rules, then the first applicable rule will be chosen, e.g. if a sequent $\Gamma \Rightarrow \Delta$ contains a formula `A < B` in the right-hand side Δ, then the clause implementing the $\preccurlyeq^i_R$ rule will be chosen, and tuCLEVER will be recursively invoked on the unique premise of such a rule introducing a conditional block $[A \lhd B]$. tuCLEVER proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible. As an example, the clause implementing $\preccurlyeq^i_R$ is as follows:

```
1. prove(Hypersequent,tree(condR,Hypersequent,[Gamma,Delta],no,
                                                  SubTree1,no)) :-
2.     select([Gamma,Delta],Hypersequent,Remainder),
3.     member(A < B, Delta),
4.     \+findBlock(Delta,[[A],B]),!,
5.     prove([[Gamma,[[[A],B]|Delta]]|Remainder],SubTree1).
```

In line 4, the auxiliary predicate `findBlock` is invoked in order to implement the decision procedure described at the beginning of this section: if a conditional block $[A \lhd B]$, represented by the Prolog pair `[[A],B]`, already belongs to Δ, then the negation as failure returns a failure, and the rule is no longer applied. Since the rule is invertible, Prolog cut `!` is used in line 4 to eventually block backtracking.

As an another example, the following clause implements the rule jump_U^i:

```
1. prove(Hypersequent,tree(jumpU,Hypersequent,[Gamma,Delta],
                                  [Gamma2,Delta2],SubTree1,no)) :-
2.     select([Gamma,Delta],Hypersequent,Remainder),
3.     member(transfer Theta, Delta),
4.     select([Gamma2,Delta2],Remainder,Remainder2),
5.     \+subset(Theta,Delta2),
6.     append(Delta2,Theta,NewDelta2),
7.     !,
8.     prove([[Gamma,Delta],[Gamma2,NewDelta2]|Remainder2],SubTree1).
```

In line 3, the predicate `member` checks whether there is a block $\langle \Theta \rangle$, represented by the Prolog term `transfer Theta`, in this case the main predicate is recursively invoked on the only premise of the rule, by adding formulas in Θ in the right hand side of the sequent represented by `Gamma2` and `Delta2`.

Implementations of the calculi for extensions of $\mathbb{VTU}$ proceed in a similar way. To show some examples, here are the clauses implementing the rules W^i and T^i, belonging to the implementations of the systems involving axioms $\mathbb{W}$ and $\mathbb{C}$.

```
1.  prove(Hypersequent,tree(w,Hypersequent,[Gamma,Delta],no,
                                                  SubTree1,no)) :-
2.     select([Gamma,Delta],Hypersequent,Remainder),
3.     member([Sigma,_],Delta),
4.     \+subset(Sigma,Delta),
6.     append(Delta,Sigma,NewDelta),!,
7.     prove([[Gamma,NewDelta]|Remainder],SubTree1).
```

In line 4 the predicate `\+subset(Sigma,Delta)` checks whether Σ, represented by the Prolog list `Sigma`, belongs to Δ, represented by the Prolog list `Delta`, in order to avoid multiple applications of the rule over the same set Σ.

The Prolog source code implementing the rule T^i is as follows:

```
1.  prove(Hypersequent,tree(t,Hypersequent,[Gamma,Delta],no,
                                            SubTree1,SubTree2)) :-
```

```
2.     select([Gamma,Delta],Hypersequent,Remainder),
3.     member(A < B,Gamma),
4.     select(transfer Theta,Delta,Delta2),
5.     \+member(B,Theta),
6.     \+findSequent(Hypersequent,[[A],Theta]),
7.     !,
8.     prove([[Gamma,Delta],[[A],Theta]|Remainder],SubTree1),
9.     prove([[Gamma,[transfer [B|Theta]|Delta2]]|Remainder],
                                                        SubTree2).
```

In line 8 an extended sequent $A \Rightarrow \Theta$ is added as a new component of the hypersequent, whereas in line 9 the formula B is added to $\langle\Theta\rangle$ in the right-hand side of the sequent under consideration. Lines 5 and 6 are used in order to implement the decision procedure, by avoiding useless applications of the rule in case either B already belongs to Θ or an extended sequent $A \Rightarrow \langle\Theta\rangle$ already exists in the hypersequent.

Let us conclude by showing the Prolog clauses implementing the rules abs^i_L and abs^i_R characterizing the systems allowing the axiom $\mathbb{A}$.

```
prove(Hypersequent,tree(absL,Hypersequent,[Gamma,Delta],
                             [Gamma2,Delta2],SubTree1,no)) :-
   select([Gamma,Delta],Hypersequent,Remainder),
   member(A < B,Gamma),
   select([Gamma2,Delta2],Remanider,Remainder2),
   \+member(A < B,Gamma2),
   !,
   prove([[Gamma,Delta],[[A < B|Gamma2],Delta2]!Remainder2],
                                                      SubTree1).
```

```
prove(Hypersequent,tree(absR,Hypersequent,[Gamma,Delta],
                               [Gamma,Delta],SubTree1,no)) :-
   select([Gamma,Delta],Hypersequent,Remainder),
   member(A < B,Delta),
   select([Gamma2,Delta2],Remainder,Remainder2),
   \+member(A < B,Delta2),
   !,
   prove([[Gamma,Delta],[Gamma2,[A < B|Delta2]]|Remainder2],
                                                     SubTree1).
```

The system tuCLEVER has also a graphical user interface implemented in the form of a responsive Web Application. As already mentioned in the Introduction, the program tuCLEVER, as well as all the Prolog source files, are available for free usage and download at http://193.51.60.97:8000/tuclever/.

5 Performance of tuCLEVER

The performance of tuCLEVER are promising. We have tested it by running SWI Prolog 7.6.4 on an Acer Aspire E5-575G, 2.7 GHz Intel Core i7 7500U, 16GB

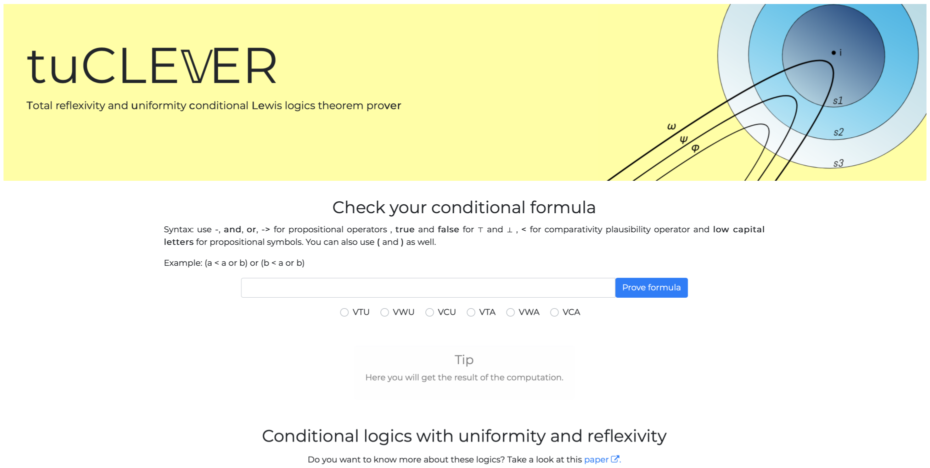

Fig. 2. Home page of tuCLEVER. When the users want to check whether a formula F is valid, then (i) they select the conditional logic to use, (ii) they type F in the form and (iii) click the button in order to execute the calculi presented in Section 3.

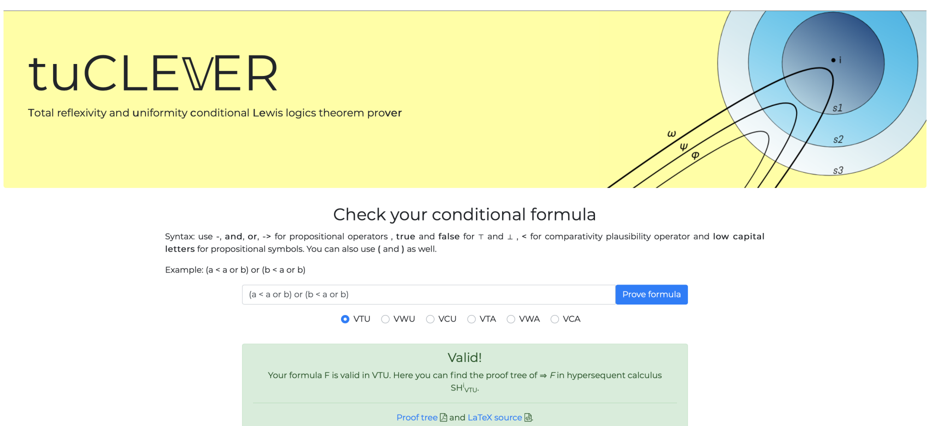

Fig. 3. When the formula is valid, tuCLEVER computes both a pdf containing the derivation found by the prover and its LaTeX source file.

RAM, Ubuntu 19.04 amd64 machine. In absence of theorem provers specifically tailored for Lewis' logics, we have compared the performances of tuCLEVER with those of VINTE [12] on formulas provable in both systems. We have performed two kinds of experiments. On the one hand, we have tested the two provers over a set of valid formulas, on the other hand we have tested tuCLEVER with randomly generated formulas, therefore including not provable ones.

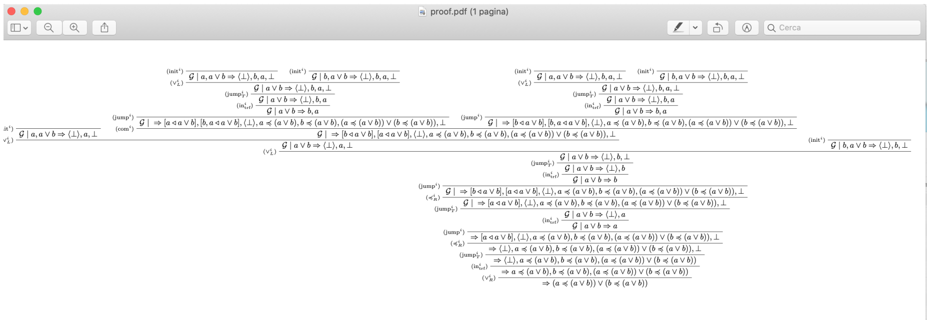

Fig. 4. When the submitted formula is valid, then the user can have a look at the derivation built by tuCLEVER, stored in a pdf file. As an alternative, the user can download the LaTeX source file of the derivation.

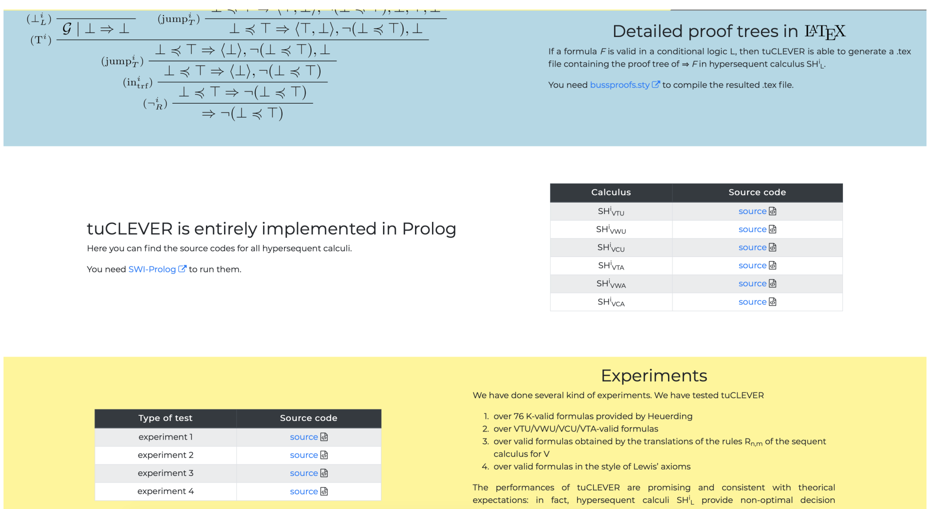

Fig. 5. All Prolog source files, including those for testing the performance of tuCLEVER, are available on the web page.

5.1 Tests over valid formulas

First of all, we have tested both tuCLEVER and VINTE over 76 valid formulas in the basic Lewis' system $\mathbb{V}$ without Uniformity [18], obtained by translating valid formulas of the basic modal logic K [14] provided by Heuerding in conditional formulas: $\Box A$ is replaced by $\top \Box\!\!\rightarrow A$[8], whereas $\Diamond A$ is replaced by $\neg(\top \Box\!\!\rightarrow \neg A)$. We have observed the results in Figure 6 concerning the number of timeouts, witnessing a significant increase of performances with respect to those of VINTE.

[8] It is worth noticing that this translation introduces an exponential blowup.

Theorem prover	1 s	60 s	180 s
VINTE	49	34	31
tuCLEVER	8	3	3

Fig. 6. Percentage of timeouts for tuCLEVER and VINTE over valid formulas.

This result could be explained by the fact that, even if tuCLEVER manipulates "heavier" hypersequents, all rules implemented by tuCLEVER are invertible, avoiding backtracking points that are present in VINTE.

We have then compared the performance of both the provers tuCLEVER and VINTE with valid formulas obtained as instances of three different schemas, by fixing a time limit of 60 seconds, and by letting a parameter n vary, starting from $n = 1$. The first schema is as follows:

$$(A_1 \preccurlyeq A_2) \vee (A_2 \preccurlyeq A_3) \vee \cdots \vee (A_n \preccurlyeq A_1),$$

We have observed that tuCLEVER is able to answer also with $n = 25$, whereas VINTE is able to answer only until $n = 9$. Similarly, we have compared the performance of the provers on:

$$(A_1 \preccurlyeq A_2) \wedge (A_2 \preccurlyeq A_3) \wedge \cdots \wedge (A_{n-1} \preccurlyeq A_n) \rightarrow (A_1 \preccurlyeq A_n)$$

obtaining that tuCLEVER is able to answer also with $n = 15$, whereas VINTE is able to answer only until $n = 5$. The prover VINTE has, however, better performances than those of tuCLEVER over formulas following the following schema:

$$(A_1 \preccurlyeq (A_1 \vee A_2 \vee \cdots \vee A_n)) \vee (A_2 \preccurlyeq (A_1 \vee A_2 \vee \cdots \vee A_n)) \vee \ldots$$
$$\ldots \vee (A_n \preccurlyeq (A_1 \vee A_2 \vee \cdots \vee A_n))$$

where tuCLEVER is able to answer with $n = 4$, whereas VINTE is able to answer also for $n = 15$.

5.2 Tests over randomly generated formulas

We have tested tuCLEVER over randomly generated formulas, fixing two different time limits, namely 1 second and 10 seconds, and varying the depth of a formula (i.e. the maximum level of nesting of connectives) as well as the number of different propositional variables. We have considered the system $\mathbb{VTU}$ as well as all the extensions, obtaining the percentages of timeouts in Figures 7 and 8.
In all cases, the quite low percentages of timeouts suggest that the performance of tuCLEVER are encouraging.

6 Conclusions and Future Issues

We have introduced tuCLEVER, a theorem prover implementing hypersequent calculi for Lewis' conditional logics with Total Reflexivity and Uniformity introduced in [11]. As far as we know, this is the first theorem prover for these stronger logics of the Lewis' family.

Depth / var	1 s	10 s
5/3	0%	0%
6/3	2%	0%
7/3	4%	2%
8/3	7%	5%
5/5	0%	0%
6/5	2%	1%
7/5	6%	4%
8/5	10%	7%

Depth / var	1 s	10 s
5/3	0%	0%
6/3	1%	0%
7/3	3%	2%
8/3	7%	4%
5/5	0%	0%
6/5	2%	1%
7/5	6%	4%
8/5	10%	6%

Fig. 7. Percentage of timeouts in $\mathrm{SH}^i_{\mathbb{VTU}}$ (left) and $\mathrm{SH}^i_{\mathbb{VWU}}$ (right).

Depth / var	1 s	10 s
5/3	0%	0%
6/3	2%	1%
7/3	5%	3%
8/3	8%	5%
5/5	0%	0%
6/5	4%	2%
7/5	7%	5%
8/5	11%	9%

Depth / var	1 s	10 s
5/3	6%	3%
6/3	12%	9%
7/3	21%	17%
8/3	25%	22%
5/5	8%	7%
6/5	20%	16%
7/5	27%	20%
8/5	31%	28%

Fig. 8. Percentage of timeouts in $\mathrm{SH}^i_{\mathbb{VCU}}$ (left) and $\mathrm{SH}^i_{\mathbb{VTA}}$ (right).

We have compared the performance of tuCLEVER with those of VINTE, a theorem prover for the weaker Lewis' logics, and we have observed that the performance of tuCLEVER are promising. We aim at extending our performance evaluation by considering other significant schemas of valid formulas: as an example, we plan to consider valid formulas obtained by the translations of the rules $R_{n,m}$ of the sequent calculus for $\mathbb{V}$ according to the translation from rules to axioms described in [17]. Furthermore, we aim at comparing the performance of tuCLEVER also with those of other provers for conditional logic, like CondLean [21], GoalD*UCK* [20], and NESCOND [22, 23]. As already mentioned, the theorem prover CSLLean [2] implements a labelled calculus for the logic of Comparative Concept Similarity over minspaces, which is equivalent to logic $\mathbb{VCU}$: we aim at comparing tuCLEVER with CSLLean, by repeating tests both over randomly generated formulas and over valid $\mathbb{VCU}$ formulas.

Finally, we are currently working on extending tuCLEVER in order to handle countermodel generation for unprovable formulas: intuitively, given a failed proof, tuCLEVER checks another Prolog predicate essentially implementing the same clauses of `prove`, with the objective of finding an open, saturated branch, following the line of the theorem provers for Lewis' logics of counterfactual reasoning [6, 7]. Clauses introducing a branch in the computation, i.e. those implementing rules with two premises, are split in two clauses, each one considering a single branch. The last clause of this additional Prolog predicate will check whether the hypersequent is *not* an instance of the initial sequents: in this way, this predicate will succeed if and only if (i) no rule of the calculi is further applicable

(ii) the hypersequent does not contain a valid extended sequent, therefore a model falsifying it can be extracted from the sequent itself.

References

1. Alenda, R., Olivetti, N.: Preferential semantics for the logic of comparative similarity over triangular and metric models. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012, LNAI, vol. 7519, pp. 1–13. Springer-Verlag Berlin Heidelberg (2012)
2. Alenda, R., Olivetti, N., Pozzato, G.L.: Csl-lean: A theorem-prover for the logic of comparative concept similarity. Electr. Notes Theor. Comput. Sci. 262, 3–16 (2010), https://doi.org/10.1016/j.entcs.2010.04.002
3. Beckert, B., Posegga, J.: leantap: Lean tableau-based deduction. Journal of Automated Reasoning 15(3), 339–358 (1995)
4. Burgess, J.P.: Quick completeness proofs for some logics of conditionals. Notre Dame Journal of Formal Logic 22, 76–84 (1981)
5. Chellas, B.F.: Basic conditional logics. Journal of Philosophical Logic 4, 133–153 (1975)
6. Dalmonte, T., Negri, S., Olivetti, N., Pozzato, G.L.: PRONOM: proof-search and countermodel generation for non-normal modal logics. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019 - Advances in Artificial Intelligence - XVIIIth International Conference of the Italian Association for Artificial Intelligence, Rende, Italy, November 19-22, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11946, pp. 165–179. Springer (2019), https://doi.org/10.1007/978-3-030-35166-3_12
7. Dalmonte, T., Olivetti, N., Pozzato, G.L.: HYPNO: theorem proving with hypersequent calculi for non-normal modal logics (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12167, pp. 378–387. Springer (2020), https://doi.org/10.1007/978-3-030-51054-1_23
8. Friedman, N., Halpern, J.Y.: On the complexity of conditional logics. In: Principles of Knowledge Representation and Reasoning. pp. 202–213. Elsevier (1994)
9. Genovese, V., Giordano, L., Gliozzi, V., Pozzato, G.L.: Logics in access control: a conditional approach. Journal of Logic and Computation 24(4), 705–762 (2014)
10. Girlando, M., Lellmann, B., Olivetti, N., Pozzato, G.L.: Standard sequent calculi for Lewis' logics of counterfactuals. In: Logics in Artificial Intelligence. JELIA 2016., vol. 10021, pp. 272–287. Springer (2016)
11. Girlando, M., Lellmann, B., Olivetti, N., Pozzato, G.L.: Hypersequent calculi for lewis' conditional logics with uniformity and reflexivity. In: Schmidt, R.A., Nalon, C. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods - 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10501, pp. 131–148. Springer (2017), https://doi.org/10.1007/978-3-319-66902-1_8
12. Girlando, M., Lellmann, B., Olivetti, N., Pozzato, G.L., Vitalis, Q.: VINTE: an implementation of internal calculi for lewis' logics of counterfactual reasoning. In: Schmidt, R.A., Nalon, C. (eds.) Automated Reasoning with Analytic Tableaux and Related Methods - 26th International Conference, TABLEAUX 2017, Brasília, Brazil, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10501, pp. 149–159. Springer (2017), https://doi.org/10.1007/978-3-319-66902-1_9

13. Grahne, G.: Updates and counterfactuals. Journal of Logic and Computation 8(1), 87–117 (1998)
14. Hughes, G., Cresswell, J.: An Introduction to Modal Logic. Methuen (1968)
15. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Allen, J.F., Fikes, R., Sandewall, E. (eds.) Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991. pp. 387–394. Morgan Kaufmann (1991)
16. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
17. Lellmann, B., Pattinson, D.: Correspondence between modal hilbert axioms and sequent rules with an application to S5. In: Automated Reasoning with Analytic Tableaux and Related Methods - 22th International Conference, TABLEAUX 2013, Nancy, France, September 16-19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8123, pp. 219–233. Springer (2013)
18. Lewis, D.: Counterfactuals. Blackwell (1973)
19. Nute, D.: Topics in Conditional Logic. Reidel, Dordrecht (1980)
20. Olivetti, N., Pozzato, G.L.: Theorem Proving for Conditional Logics: CondLean and GoalDuck. J. of Applied Non-Classical Logics 18(4), 427–473 (2008)
21. Olivetti, N., Pozzato, G.L.: Condlean 3.0: Improving condlean for stronger conditional logics. In: Beckert, B. (ed.) Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2005, Koblenz, Germany, September 14-17, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3702, pp. 328–332. Springer (2005), https://doi.org/10.1007/11554554_27
22. Olivetti, N., Pozzato, G.L.: NESCOND: an implementation of nested sequent calculi for conditional logics. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8562, pp. 511–518. Springer (2014), https://doi.org/10.1007/978-3-319-08587-6_39
23. Olivetti, N., Pozzato, G.L.: Nested sequent calculi and theorem proving for normal conditional logics: The theorem prover NESCOND. Intelligenza Artificiale 9(2), 109–125 (2015)
24. Olivetti, N., Pozzato, G.L.: A standard internal calculus for lewis' counterfactual logics. In: de Nivelle, H. (ed.) Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015. Proceedings, Lecture Notes in Computer Science, vol. 9323, pp. 270–286. Springer (2015), https://doi.org/10.1007/978-3-319-24312-2_19
25. Sheremet, M., Tishkovsky, D., Wolter, F., Zakharyaschev, M.: A logic for concepts and similarity. J. Log. Comput. 17(3), 415–452 (2007)
26. Stalnaker, R.: A theory of conditionals. In: Rescher, N. (ed.) Studies in Logical Theory, pp. 98–112. Blackwell (1968)

Acknowledgements

This work is partially supported by the Projects TICAMORE ANR-16-CE91-0002-01, WWTF project MA16-28, and INDAM-GNCS "METALLIC #2" INDAM-GNCS Project 2020.