



HAL
open science

Theorem proving for Lewis Logics of Counterfactual Reasoning

Marianna Girlando, Bjoern Lellmann, Nicola Olivetti, Stefano Pesce, Gian Luca Pozzato

► **To cite this version:**

Marianna Girlando, Bjoern Lellmann, Nicola Olivetti, Stefano Pesce, Gian Luca Pozzato. Theorem proving for Lewis Logics of Counterfactual Reasoning. CILC 2020 - 35th Edition of the Italian Conference on Computational Logic, Oct 2020, Rende / Virtual, Italy. hal-03080670

HAL Id: hal-03080670

<https://hal.science/hal-03080670>

Submitted on 17 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Theorem proving for Lewis Logics of Counterfactual Reasoning^{*}

Marianna Girlando¹, Björn Lellmann², Nicola Olivetti³, Stefano Pesce⁴, and Gian Luca Pozzato⁴ (✉)

¹ Équipe Partout, Inria Saclay, LIX - École Polytechnique, France - marianna.girlando@inria.fr

² Technische Universität Wien, Austria - llemmann@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, Università di Torino, Italy - gianluca.pozzato@unito.it, 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 $\mathbb{V}TU$ and its main extensions. **tuCLEVER** is inspired by the methodology of **lean** $T^A P$ 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 $\mathbb{V}CU$, an extension of the basic system $\mathbb{V}TU$ 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

^{*} Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

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 $\text{SP}(x)$. Each set in $\text{SP}(x)$ 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 \preceq , with a formula $A \preceq B$ 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, \preceq 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 $\alpha \in \text{SP}(x)$;
- *Total reflexivity*: every world x belongs to some sphere $\alpha \in \text{SP}(x)$.

The basic logic is VTU . We also consider some of its extensions, including the above mentioned 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 \Leftarrow , which allows to express, e.g,

$$\text{PicassoPainting} \sqsubseteq \text{BraquePainting} \Leftarrow \text{GiottoPainting}$$

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 VWU , an extension of the basic system VTU with a property known as *weak centering*, and the one defined by “minspace” Distance Models coincides with 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 S5 as a fragment: $\Box A$ can be defined as $\perp \preceq \neg A$ (or $\neg A \Box \rightarrow \perp$).

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 Uniformity⁵. In [11] we proposed the first proof systems for $\mathbb{V}\text{TU}$ 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 $\mathbb{V}\text{TU}$ 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 Uniformity⁶. The conception of **tuCLEVER** is inspired by the methodology of *lean^{TAP}* [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 $\mathbb{V}\text{TU}$ 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

$$A ::= p \mid \perp \mid \top \mid \neg A \mid A \rightarrow A \mid A \wedge A \mid A \vee A \mid A \approx A$$

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 $\mathbb{V}\text{CU}$.

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

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

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

The valuation $\llbracket \cdot \rrbracket$ is extended to all formulae as follows:

$$\begin{aligned}
\llbracket \perp \rrbracket &= \emptyset \\
\llbracket \top \rrbracket &= W \\
\llbracket \neg A \rrbracket &= W - \llbracket A \rrbracket \\
\llbracket A \wedge B \rrbracket &= \llbracket A \rrbracket \cap \llbracket B \rrbracket \\
\llbracket A \vee B \rrbracket &= \llbracket A \rrbracket \cup \llbracket B \rrbracket \\
\llbracket A \rightarrow B \rrbracket &= (W - \llbracket A \rrbracket) \cup \llbracket B \rrbracket \\
\llbracket A \preceq B \rrbracket &= \{w \in W \mid \forall \alpha \in \text{SP}(w). \text{ if } \llbracket B \rrbracket \cap \alpha \neq \emptyset, \text{ then } \llbracket A \rrbracket \cap \alpha \neq \emptyset\}
\end{aligned}$$

Validity and satisfiability of formulae in a class of models are defined as usual. The logic $\mathbb{V}\text{TU}$ 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 \preceq as recalled in the Introduction, and its truth condition is as follows:

$$\llbracket A \Box \rightarrow B \rrbracket = \{w \in W \mid \text{either } \bigcup \text{SP}(w) \cap \llbracket A \rrbracket = \emptyset \text{ or } \exists \alpha \in \text{SP}(w) \text{ such that } \alpha \cap \llbracket A \rrbracket \neq \emptyset \text{ and } \alpha \cap \llbracket A \rrbracket \subseteq \llbracket B \rrbracket\}.$$

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

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

Extensions of $\mathbb{V}\text{TU}$ are denoted by concatenating letters for these properties: \mathbb{W} for weak centering, \mathbb{C} for centering, and \mathbb{A} for absoluteness. We consider⁷:

$$\begin{array}{ll}
\text{VTU} & \text{VTA: VTU} + \text{absoluteness} \\
\text{VWU: VTU} + \text{weak centering} & \text{VWA: VTA} + \text{weak centering} \\
\text{VCU: VTU} + \text{centering} & \text{VCA: VTA} + \text{centering}
\end{array}$$

⁷ Observe that $\text{VTA} + \text{weak centering}$ collapses to S5 , and that $\text{VTA} + \text{centering}$ collapses to classical logic.

(CPR) $\frac{\vdash B \rightarrow A}{\vdash A \preceq B}$ (TR) $(A \preceq B) \wedge (B \preceq C) \rightarrow (A \preceq C)$ (N) $\neg(\perp \preceq \top)$ (U1) $\neg(\perp \preceq A) \rightarrow (\perp \preceq (\perp \preceq A))$ (W) $A \rightarrow (A \preceq \top)$ (A1) $(A \preceq B) \rightarrow (\perp \preceq \neg(A \preceq B))$	(CPA) $(A \preceq A \vee B) \vee (B \preceq A \vee B)$ (CO) $(A \preceq B) \vee (B \preceq A)$ (T) $(\perp \preceq \neg A) \rightarrow A$ (U2) $(\perp \preceq \neg A) \rightarrow (\perp \preceq \neg(\perp \preceq \neg A))$ (C) $(A \preceq \top) \rightarrow A$ (A2) $\neg(A \preceq B) \rightarrow (\perp \preceq (A \preceq B))$
$\mathcal{A}_{\forall\text{TTU}} := \{(\text{CPR}), (\text{CPA}), (\text{TR}), (\text{CO}), (\text{N}), (\text{T}), (\text{U1}), (\text{U2})\}$	
$\mathcal{A}_{\forall\text{WU}} := \mathcal{A}_{\forall\text{TTU}} \cup \{(\text{W})\}$ $\mathcal{A}_{\forall\text{CU}} := \mathcal{A}_{\forall\text{TTU}} \cup \{(\text{W}), (\text{C})\}$ $\mathcal{A}_{\forall\text{TA}} := \mathcal{A}_{\forall\text{TTU}} \cup \{(\text{A1}), (\text{A2})\}$	
$\mathcal{A}_{\forall\text{WA}} := \mathcal{A}_{\forall\text{TTU}} \cup \{(\text{W}), (\text{A1}), (\text{A2})\}$ $\mathcal{A}_{\forall\text{CA}} := \mathcal{A}_{\forall\text{TTU}} \cup \{(\text{W}), (\text{C}), (\text{A1}), (\text{A2})\}$	

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 \preceq). Propositional axioms and rules are standard.

3 Hypersequent Calculi for Lewis' Logics

We recall hypersequent calculi for $\forall\text{TTU}$ 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 $\langle \cdot \rangle$ interpreting possible formulae.

Formally, we define:

- a *conditional block*, which is a tuple $[\Sigma \triangleleft C]$ 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 $\Gamma_1 \Rightarrow \Delta_1 \mid \dots \mid \Gamma_n \Rightarrow \Delta_n$.

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

$$\begin{aligned} \iota_e(\Gamma \Rightarrow \Delta, [\Sigma_1 \triangleleft C_1], \dots, [\Sigma_n \triangleleft C_n], \langle \Theta_1 \rangle, \dots, \langle \Theta_m \rangle) &:= \bigwedge \Gamma \rightarrow \bigvee \Delta \vee \bigvee_{i=1}^n \bigvee_{B \in \Sigma_i} (B \preceq C_i) \vee \bigvee_{j=1}^m \diamond(\bigvee \Theta_j) \\ \iota_e(\Gamma_1 \Rightarrow \Delta_1 \mid \dots \mid \Gamma_n \Rightarrow \Delta_n) &:= \square \iota_e(\Gamma_1 \Rightarrow \Delta_1) \vee \dots \vee \square \iota_e(\Gamma_n \Rightarrow \Delta_n). \end{aligned}$$

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

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

$$\begin{array}{c}
\frac{}{\mathcal{G} \mid \Omega, \perp \Rightarrow \Theta} \perp_L \quad \frac{}{\mathcal{G} \mid \Omega \Rightarrow \Theta, \top} \top_R \quad \frac{}{\mathcal{G} \mid \Omega, p \Rightarrow \Theta, p} \text{init} \quad \frac{\mathcal{G} \mid \Omega, \neg A \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, \neg A \Rightarrow \Theta} \neg_L^i \\
\frac{\mathcal{G} \mid \Omega, A \Rightarrow \Theta, \neg A}{\mathcal{G} \mid \Omega \Rightarrow \Theta, \neg A} \neg_R^i \quad \frac{\mathcal{G} \mid \Omega, A \wedge B, A, B \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, A \wedge B \Rightarrow \Theta} \wedge_L^i \quad \frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \vee B, A, B}{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \vee B} \vee_R^i \\
\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_R^i \quad \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_L^i \\
\frac{\mathcal{G} \mid \Omega, A \rightarrow B, B \Rightarrow \Theta \quad \mathcal{G} \mid \Omega, A \rightarrow B \Rightarrow \Theta, A}{\mathcal{G} \mid \Omega, A \rightarrow B \Rightarrow \Theta} \rightarrow_L^i \quad \frac{\mathcal{G} \mid \Omega, A \Rightarrow \Theta, A \rightarrow B, B}{\mathcal{G} \mid \Omega \Rightarrow \Theta, A \rightarrow B} \rightarrow_R^i \\
\frac{\mathcal{G} \mid \Sigma \Rightarrow \Pi, A \preceq B, [A \triangleleft B]}{\mathcal{G} \mid \Sigma \Rightarrow \Pi, A \preceq B} \preceq_R^i \quad \frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma \triangleleft A] \mid A \Rightarrow \Sigma}{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma \triangleleft A]} \text{jump}^i \\
\frac{\mathcal{G} \mid \Omega, C \preceq D \Rightarrow \Theta, [D, \Sigma \triangleleft A] \quad \mathcal{G} \mid \Omega, C \preceq D \Rightarrow \Theta, [\Sigma \triangleleft A], [\Sigma \triangleleft C]}{\mathcal{G} \mid \Omega, C \preceq D \Rightarrow \Theta, [\Sigma \triangleleft A]} \preceq_L^i \\
\frac{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1, \Sigma_2 \triangleleft A], [\Sigma_2 \triangleleft B] \quad \mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1 \triangleleft A], [\Sigma_1, \Sigma_2 \triangleleft B]}{\mathcal{G} \mid \Omega \Rightarrow \Theta, [\Sigma_1 \triangleleft A], [\Sigma_2 \triangleleft B]} \text{com}^i \\
\frac{\mathcal{G} \mid \Sigma, A \preceq B \Rightarrow \Pi, \langle \Theta \rangle \mid A \Rightarrow \Theta \quad \mathcal{G} \mid \Sigma, A \preceq B \Rightarrow \Pi, \langle \Theta, B \rangle}{\mathcal{G} \mid \Sigma, A \preceq B \Rightarrow \Pi, \langle \Theta \rangle} \top^i \\
\frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \perp \rangle}{\mathcal{G} \mid \Gamma \Rightarrow \Delta} \text{in}_{\text{trf}}^i \quad \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} \text{jump}_U^i \quad \frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle, \Theta}{\mathcal{G} \mid \Gamma \Rightarrow \Delta, \langle \Theta \rangle} \text{jump}_T^i \\
\frac{\mathcal{G} \mid \Gamma \Rightarrow \Delta, [\Sigma \triangleleft A], \Sigma}{\mathcal{G} \mid \Gamma \Rightarrow \Delta, [\Sigma \triangleleft A]} W^i \quad \frac{\mathcal{G} \mid \Gamma, C \preceq D, C \Rightarrow \Delta \quad \mathcal{G} \mid \Gamma, C \preceq D \Rightarrow D, \Delta}{\mathcal{G} \mid \Gamma, C \preceq D \Rightarrow \Delta} C^i \\
\frac{\mathcal{G} \mid \Gamma, A \preceq B \Rightarrow \Delta \mid \Sigma, A \preceq B \Rightarrow \Pi}{\mathcal{G} \mid \Gamma, A \preceq B \Rightarrow \Delta \mid \Sigma \Rightarrow \Pi} \text{abs}_L^i \quad \frac{\mathcal{G} \mid \Gamma \Rightarrow A \preceq B, \Delta \mid \Sigma \Rightarrow A \preceq B, \Pi}{\mathcal{G} \mid \Gamma \Rightarrow A \preceq B, \Delta \mid \Sigma \Rightarrow \Pi} \text{abs}_R^i \\
\hline
\text{PC} = \{ \perp_L, \top_R, \text{init}, \neg_L^i, \neg_R^i, \wedge_L^i, \wedge_R^i, \vee_L^i, \vee_R^i, \rightarrow_L^i, \rightarrow_R^i \} \\
\text{SH}_{\text{VTU}}^i = \text{PC} \cup \{ \preceq_R^i, \preceq_L^i, \text{com}^i, \text{jump}^i, \top^i, \text{in}_{\text{trf}}^i, \text{jump}_U^i, \text{jump}_T^i \} \\
\text{SH}_{\text{VWU}}^i = \text{SH}_{\text{VTU}}^i \cup \{ W^i \} \quad \text{SH}_{\text{VCU}}^i = \text{SH}_{\text{VWU}}^i \cup \{ C^i \} \quad \text{SH}_{\text{VTA}}^i = \text{SH}_{\text{VTU}}^i \cup \{ \text{abs}_L^i, \text{abs}_R^i \} \\
\text{SH}_{\text{VWA}}^i = \text{SH}_{\text{VWU}}^i \cup \{ \text{abs}_L^i, \text{abs}_R^i \} \quad \text{SH}_{\text{VCA}}^i = \text{SH}_{\text{VCU}}^i \cup \{ \text{abs}_L^i, \text{abs}_R^i \} \\
\hline
\end{array}$$

Fig. 1. The hypersequent calculi for 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 $\text{lean}T^AP$, 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 $\text{SH}_{\mathcal{L}}^i$, 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 $[\text{Gamma}, \text{Delta}]$, 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 $[\Sigma \triangleleft C]$ is a pair $[\text{Sigma}, \text{C}]$, 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 \perp are represented by constants **true** and **false**, respectively, whereas connectives \neg , \wedge , \vee , \rightarrow , \preceq , and $\square\rightarrow$ are represented by **-**, **and**, **or**, **->**, **<**, and **=>**. Propositional variables are represented by Prolog atoms. As an example, the sequent $A, \neg B \vee C \Rightarrow A \wedge C, D, A \rightarrow B, \langle \perp \rangle, [A \preceq C, B \triangleleft A \vee C]$ is represented by the list: $[[\text{a}, \text{-b or c}], [\text{a and c, d, a -> b, transfer[false], [[\text{a} < \text{c}, \text{b}], \text{a or c}]]]$.

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 $(A \preceq A \vee B) \vee (B \preceq A \vee B)$ in 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 \perp_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 \preceq_R^i rule will be chosen, and **tuCLEVER** will be recursively invoked on the unique premise of such a rule introducing a conditional block $[A \triangleleft B]$. **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 \preceq_R^i is as follows:

```

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_L^i and abs_R^i characterizing the systems allowing the axiom \mathbb{A} .

```

1.   prove(Hypersequent,tree(absL,Hypersequent,[Gamma,Delta],
[Gamma2,Delta2],SubTree1,no)) :-
2.   select([Gamma,Delta],Hypersequent,Remainder),
3.   member(A < B,Gamma),
4.   select([Gamma2,Delta2],Remainder,Remainder2),
5.   \+member(A < B,Gamma2),
6.   !,
7.   prove([[Gamma,Delta],[[A < B|Gamma2],Delta2]|Remainder2],
SubTree1).

```

```

1.   prove(Hypersequent,tree(absR,Hypersequent,[Gamma,Delta],
[Gamma,Delta],SubTree1,no)) :-
2.   select([Gamma,Delta],Hypersequent,Remainder),
3.   member(A < B,Delta),
4.   select([Gamma2,Delta2],Remainder,Remainder2),
5.   \+member(A < B,Delta2),
6.   !,
7.   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

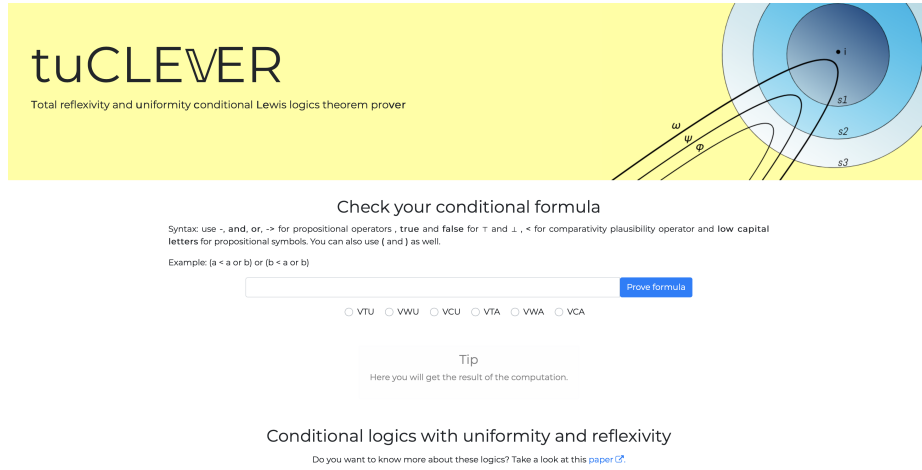


Fig. 2. Home page of tuCLEVER. When the users want to check whether a formula F is valid, then (i) they select the conditional logic to use, (ii) they type F in the form and (iii) click the button in order to execute the calculi presented in Section 3.

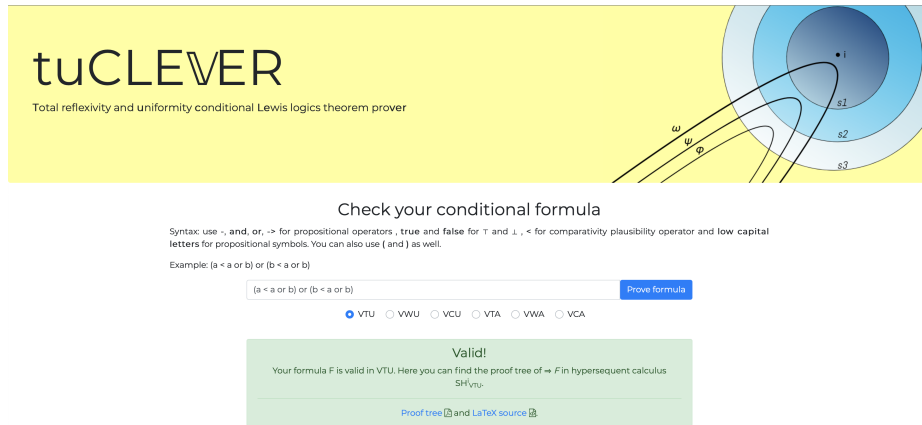


Fig. 3. When the formula is valid, tuCLEVER computes both a pdf containing the derivation found by the prover and its $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ source file.

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.

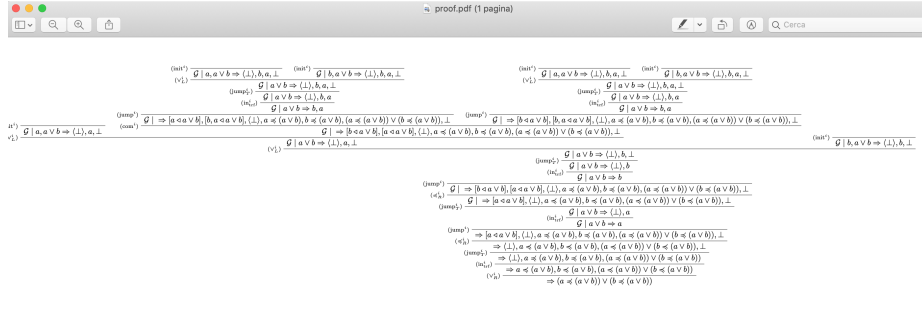


Fig. 4. When the submitted formula is valid, then the user can have a look at the derivation built by tuCLEVER, stored in a pdf file. As an alternative, the user can download the L^AT_EX source file of the derivation.

$$\begin{array}{c}
 \frac{(\perp \perp)}{\mathcal{G} \mid \perp \Rightarrow \perp} \quad \frac{(\text{jump}_T^*)}{\perp \not\approx T \Rightarrow \langle T, \perp \rangle, \neg(\perp \not\approx T), \perp} \\
 \frac{(\text{imp}_T^*)}{\perp \not\approx T \Rightarrow \langle \perp \rangle, \neg(\perp \not\approx T), \perp} \\
 \frac{(\text{init}_{\perp})}{\perp \not\approx T \Rightarrow \langle \perp \rangle, \neg(\perp \not\approx T)} \\
 \frac{(\neg \perp)}{\Rightarrow \neg(\perp \not\approx T)}
 \end{array}$$

Detailed proof trees in L^AT_EX

If a formula F is valid in a conditional logic L , then tuCLEVER is able to generate a .tex file containing the proof tree of $\Rightarrow F$ in hypersequent calculus SH_L.

You need [busprofsty](#) to compile the resulted .tex file.

Calculus	Source code
SH _{VNI}	source
SH _{VNIU}	source
SH _{VNIU}	source
SH _{VNIU}	source
SH _{VNIU}	source
SH _{VNIU}	source
SH _{VNIU}	source

tuCLEVER is entirely implemented in Prolog

Here you can find the source codes for all hypersequent calculi.

You need [SWI-Prolog](#) to run them.

Experiments

We have done several kind of experiments. We have tested tuCLEVER

- over 76 K-valid formulas provided by Heuerding
- over VTU/VWU/VCU/VTA-valid formulas
- over valid formulas obtained by the translations of the rules R_{com} of the sequent calculus for V
- over valid formulas in the style of Lewis' axioms

The performances of tuCLEVER are promising and consistent with theoretical expectations. In fact, hypersequent calculi SH_L provide non-optimal decision procedures for VTU and its extensions. In this respect, it is known that deciding

Type of test	Source code
experiment 1	source
experiment 2	source
experiment 3	source
experiment 4	source

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.

<i>Theorem prover</i>	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:

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

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:

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

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:

$$(A_1 \preceq (A_1 \vee A_2 \vee \cdots \vee A_n)) \vee (A_2 \preceq (A_1 \vee A_2 \vee \cdots \vee A_n)) \vee \dots \\ \dots \vee (A_n \preceq (A_1 \vee A_2 \vee \cdots \vee A_n))$$

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 ∇TU 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.

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

Fig. 7. Percentage of timeouts in SH_{VTU}^i (left) and SH_{VWU}^i (right).

<i>Depth / var</i>	1 s	10 s	<i>Depth / var</i>	1 s	10 s
5/3	0%	0%	5/3	6%	3%
6/3	2%	1%	6/3	12%	9%
7/3	5%	3%	7/3	21%	17%
8/3	8%	5%	8/3	25%	22%
5/5	0%	0%	5/5	8%	7%
6/5	4%	2%	6/5	20%	16%
7/5	7%	5%	7/5	27%	20%
8/5	11%	9%	8/5	31%	28%

Fig. 8. Percentage of timeouts in SH_{VCU}^i (left) and SH_{VTA}^i (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], **GOALDUCK** [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 $\mathbb{V}\text{CU}$: we aim at comparing **tuCLEVER** with **CSLLean**, by repeating tests both over randomly generated formulas and over valid $\mathbb{V}\text{CU}$ 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, Brasilia, 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, Brasilia, 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., Zakharyashev, 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.