Proof Methods and Theorem Proving for Conditional Logics with Strong Centering

Valentina Gliozzi[1], Gian Luca Pozzato[1,*] and Alberto Valese[1]

[1]Dipartimento di Informatica, Università degli Studi di Torino, 10149, Turin, Italy

Abstract

In this work we continue our investigation on proof methods and theorem proving for Conditional Logics with the selection function semantics. Conditional Logics recently have received a renewed attention and have found several applications in knowledge representation and artificial intelligence. We present a labelled sequent calculus for systems including the axiom of strong centering CS, as well as a theorem prover implementing the sequent calculus in Prolog.

Keywords

Conditional logics, Sequent calculi, Proof methods, Prolog, Nonmonotonic reasoning, Theorem proving

1. Introduction

Conditional Logics have a long history, starting with the seminal works [1, 2, 3, 4, 5] in the seventies. In the last decades, Conditional Logics have found a renewed interest in several fields of artificial intelligence and knowledge representation, from hypothetical reasoning to belief revision, from diagnosis to nonmonotonic reasoning and planning [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16].

Conditional Logics are extensions of classical logic by a binary operator $\Rightarrow$, called *conditional operator*, used in order to express formulas of the form $A \Rightarrow B$. Similarly to modal logics, the semantics of Conditional Logics can be defined in terms of possible world structures. In this respect, Conditional Logics can be seen as a generalization of modal logics (or a type of multi-modal logic) where the conditional operator is a sort of modality indexed by a formula of the same language. However, as a difference with modal logics, the lack of a universally accepted semantics led to a partial underdevelopment of proof methods and theorem provers for these logics.

An attempt to fill this gap has been proposed in [17], where labelled sequent calculi for conditional logics with the *selection function semantics* ([2]) are introduced. Intuitively, the selection function f selects, for a world w and a formula A, the set of worlds $f(w, A)$ which

CILC'23: 38th Italian Conference on Computational Logic, June 21–23, 2023, Udine, Italy

*Corresponding author.

valentina.gliozzi@unito.it (V. Gliozzi); gianluca.pozzato@unito.it (G. L. Pozzato); alberto.valese@edu.unito.it (A. Valese)

https://www.di.unito.it/~gliozzi (V. Gliozzi); https://www.di.unito.it/~pozzato (G. L. Pozzato); https://www.di.unito.it/~valese (A. Valese)

0000-0003-1045-8018 (V. Gliozzi); 0000-0002-3952-4624 (G. L. Pozzato)

are "most-similar to w" given the information A. In *normal* conditional logics, f depends on the set of worlds satisfying A rather than on A itself, so that $f(w, A) = f(w, A')$ whenever A and A' are true in the same worlds. A conditional formula $A \Rightarrow B$ is true in w whenever B is true in every world selected by f for A and w. Since we adopt the selection function semantics, CK is the fundamental system, it has the same role as the system K (from which it derives its name) in modal logic. Formulas valid in CK are exactly those ones that are valid in every selection function model. Extensions are then obtained by imposing restrictions on the selection function. In [17], CK and some standard extensions with conditions ID (conditional identity), MP (conditional modus ponens), CEM (conditional third excluded middle), and CS (conditional strong centering), as well as most of the combinations of them are investigated. The proposed calculi, called SeqS, are modular, in some cases optimal and have been implemented by several versions of a "lean" style Prolog program called CondLean [18, 19, 20].

The original SeqS calculi, as well as their implementation CondLean, have two main drawbacks: (i) they neglect all the systems including both the axioms CEM and MP: the reason is that the proof of cut elimination, needed in order to prove the completeness of the calculi, does not work when such axioms are considered together; (ii) the rules dealing with the conditions CS and CEM are based on a label substitution mechanism that leads to a very inefficient implementation. A solution to (i) has been provided in [20, 21], where calculi for the whole cube of the extensions of CK generated by the above axioms are proposed, including those with both CEM and MP. The theorem prover implementing these calculi has better performance than CondLean, however, the problem (ii) is far from being solved: indeed, the rule for CS is still based on the label substitution mechanism, and the performances of the theorem prover are just increased in systems with both MP and CEM where the rule for CS can be omitted since the axiom (CS) is derivable (see Proposition 3). The relevance of strong centering is motivated by the concept of *similarity*: if A holds in a world w, then the A-world which is more similar to w is w itself. Moreover, strong centering is needed in order to perform inferences that are plausible in several contexts, such as if $A \Rightarrow B$ holds and A holds, then also B holds (a kind of conditional modus ponens), as well as the principle that a subjunctive conditional is false if its antecedent is true and its consequent is false, namely if $A \wedge \neg B$ holds, then so is $\neg(A \Rightarrow B)$.

In this work we try to provide a final solution to the problem of efficient theorem proving for Conditional Logics with selection function semantics. We can identify two main contributions of this work:

1. we further refine labelled sequent calculi for conditional logics in order to tackle question (ii) above. To this aim, we introduce SeqS_{23}, labelled sequent calculi for CK and the whole cube of the combinations of extensions with axioms CS, CEM, MP, and ID, where also the rule for CS does no longer exploit label substitutions. We show that one can derive a decision procedure from the cut-free calculi, providing a constructive proof of decidability of the logics considered. By estimating the size of the finite derivations of a given sequent, we also obtain a polynomial space complexity bound for these logics;
2. we introduce CondLean 4.0, a Prolog implementation of the proposed calculi SeqS_{23}: the program is inspired by the "lean" methodology, whose aim is to write short programs and exploit the power of Prolog's engine as much as possible: in this respect, every clause of a single predicate, called `prove`, implements an axiom or rule of the calculi and the

proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism. We have tested the performances of SeqS_{23} and compared them with those of the most recent version of CondLean [21], obtaining promising results that confirm that avoiding label substitution leads to a significant improvement of the performances of the prover.

2. Conditional Logics and the Selection Function Semantics

In this section we recall Conditional Logics. We first define the language and the syntax of Conditional Logics, then we present the semantics based on a selection function. Last, we provide an axiomatization of the systems of Conditional Logics we consider.

Definition 1 (Syntax of Conditional Logics). *A propositional conditional language $\mathcal{L}$ contains:*

- *a set of propositional variables ATM;*
- *the constants $\bot$ and $\top$;*
- *the set of connectives $\neg$ (unary), $\wedge, \vee, \rightarrow, \Rightarrow$ (binary).*

Formulas of $\mathcal{L}$ are inductively defined as follows:

- *atomic formulas $P \in ATM$, $\bot$ and $\top$ are formulas of $\mathcal{L}$;*
- *given A and B formulas of $\mathcal{L}$, complex formulas $\neg A, A \wedge B, A \vee B, A \rightarrow B$, and $A \Rightarrow B$ are formulas of $\mathcal{L}$.*

Let us now introduce the selection function semantics [2]. Intuitively, a conditional formula $A \Rightarrow B$ holds in a possible world w if B holds in all the worlds that are *most similar* to w given the information A. Such worlds are those selected by the *selection function* for w and A.

We define the *selection function semantics* as follows:

Definition 2 (Selection function semantics). *A model is a triple*

$$\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$$

where:

- *$\mathcal{W}$ is a non-empty set of* worlds*;*
- *f is the* selection function *$f : \mathcal{W} \times 2^{\mathcal{W}} \longrightarrow 2^{\mathcal{W}}$*
- *$[\,]$ is the* evaluation function*, which assigns to an atom $P \in ATM$ the set of worlds where P is true, and is extended to complex formulas as follows:*
  - $[\bot] = \emptyset$
  - $[\top] = \mathcal{W}$
  - $[\neg A] = \mathcal{W} \setminus [A]$
  - $[A \wedge B] = [A] \cap [B]$
  - $[A \vee B] = [A] \cup [B]$

- $[A \rightarrow B] = (\mathcal{W} \setminus [A]) \cup [B]$
- $[A \Rightarrow B] = \{w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}$

As usual, a formula F is valid in a model $\mathcal{M}$, written $\mathcal{M} \models F$, if it holds in all the worlds of $\mathcal{M}$, i.e. $\mathcal{M} \models F$ if and only if $[F] = \mathcal{W}$. A formula F is valid, written $\models F$, if it is valid in all models, i.e. $\models F$ if and only if $\mathcal{M} \models F$ for all $\mathcal{M}$.

As mentioned above, the selection function f selects, for a world w and a formula A, the set of worlds of $\mathcal{W}$ which are *closer/similar* to w given the information A. A conditional formula $A \Rightarrow B$ holds in a world w if the formula B holds in *all the worlds selected by f for w and A*. It is worth noticing that we have defined f taking $[A]$ rather than A (i.e. $f(w, [A])$ rather than $f(w, A)$) as an argument; this is equivalent to define f on formulas, i.e. $f(w, A)$ but imposing that if $[A] = [A']$ in the model, then $f(w, A) = f(w, A')$. This condition is called *normality*.

The semantics above characterizes the basic conditional logic CK. Formulas that are valid in CK are valid in all selection function models.

An axiomatization of this system is given by:

- any axiomatization of classical propositional calculus;
- (Modus Ponens) $\dfrac{A \quad A \rightarrow B}{B}$
- (RCEA) $\dfrac{A \rightarrow B \qquad B \rightarrow A}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$
- (RCK) $\dfrac{(A_1 \wedge \cdots \wedge A_n) \rightarrow B}{(C \Rightarrow A_1 \wedge \cdots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)}$

As for modal logics, in order to perform useful inferences, we can consider extensions of CK by assuming further properties on the selection function. Let us consider, for instance, the *conditional third excluded middle*, namely the formula

$$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$$

This formula is not valid: as an example, consider the model $\mathcal{M}_1 = \langle \{w, u, v\}, f_1, [\,]_1 \rangle$, where the selection function f_1 is such that $f_1(w, [A]) = \{u, v\}$ and the evaluation is $[B]_1 = \{u\}$. We have that $\mathcal{M}_1 \not\models (A \Rightarrow B) \vee (A \Rightarrow \neg B)$. Indeed, $w \notin [(A \Rightarrow B) \vee (A \Rightarrow \neg B)]$, since neither $w \in [A \Rightarrow B]$ nor $w \in [A \Rightarrow \neg B]$. $A \Rightarrow B$ does not hold in w, since there exists v, which is similar to w given A (selected by f_1 for w and A) where B does not hold (B holds only in u). Similarly, $A \Rightarrow \neg B$ does not hold in w, since there exists u, which is similar to w given A (selected by f_1 for w and A) where B holds. Such a formula is however valid if we restrict our concern to models where the selection function can select at most one world, that is to say imposing the condition:

$$\mid f(w, [A]) \mid \leq 1$$

Obviously, the above counter model $\mathcal{M}_1$ cannot be considered, in other words either u or v can be the only world selected by f for w and A, then either $A \Rightarrow B$ or $A \Rightarrow \neg B$ holds in w.

We consider the following standard extensions of the basic system of Conditional Logics CK:

Logic	Axiom	Model condition
ID	$A \Rightarrow A$	$f(w,[A]) \subseteq [A]$
CS	$(A \wedge B) \rightarrow (A \Rightarrow B)$	$w \in [A] \rightarrow f(w,[A]) \subseteq \{w\}$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w,[A]) \mid \leq 1$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	$w \in [A] \rightarrow w \in f(w,[A])$

The above axiomatizations are complete with respect to the respective semantics [2]. We can observe that:

Proposition 3. *In systems with both axioms (CEM) and (MP), axiom (CS) is derivable.*

Indeed, for (CEM) we have that $\mid f(w,[A]) \mid \leq 1$. For (MP), we have that, if $w \in [A]$, then $w \in f(w,[A])$. Therefore, it follows that if $w \in [A]$, then $f(w,[A]) = \{w\}$, satisfying the (CS) condition.

3. Sequent Calculi for Conditional Logics with Strong Centering

In this section we introduce SeqS_{23}, a family of labelled sequent calculi for the conditional systems under consideration. The calculi are modular and they are able to deal with the basic system CK as well as with the whole cube of extensions with axioms ID, CS, CEM and MP.

The calculi make use of labels to represent possible worlds. We consider a language $\mathcal{L}$ and a denumerable alphabet of labels $\mathcal{A}$, whose elements are denoted by *x, y, z, ...* There are two kinds of labelled formulas: *world formulas*, denoted by *x: A*, where $x \in \mathcal{A}$ and $A \in \mathcal{L}$, used to represent that *A* holds in a world *x*; *transition formulas*, denoted by $x \xrightarrow{A} y$, where $x, y \in \mathcal{A}$ and $A \in \mathcal{L}$. A transition formula $x \xrightarrow{A} y$ represents that $y \in f(x, [A])$.

A *sequent* is a pair $\langle \Gamma, \Delta \rangle$, usually denoted with $\Gamma \vdash \Delta$, where Γ and Δ are multisets of labelled formulas. The intuitive meaning of $\Gamma \vdash \Delta$ is: every model that satisfies all labelled formulas of Γ in the respective worlds (specified by the labels) satisfies at least one of the labelled formulas of Δ (in those worlds).

Formally: G

Definition 4 (Validity of a sequent). *Given a model $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$ for $\mathcal{L}$, and a label alphabet $\mathcal{A}$, we consider any* mapping $I : \mathcal{A} \rightarrow \mathcal{W}$. *Let F be a labelled formula, we define $\mathcal{M} \models_I F$ as $\mathcal{M} \models_I x$: A if and only if $I(x) \in [A]$ and $\mathcal{M} \models_I x \xrightarrow{A} y$ if and only if $I(y) \in f(I(x), [A])$.*

We say that $\Gamma \vdash \Delta$ is valid *in $\mathcal{M}$ if for every mapping $I : \mathcal{A} \rightarrow \mathcal{W}$, if $\mathcal{M} \models_I F$ for every $F \in \Gamma$, then $\mathcal{M} \models_I G$ for some $G \in \Delta$.*

We say that $\Gamma \vdash \Delta$ is valid in a system (CK or any extension of it) if it is valid in every $\mathcal{M}$ satisfying the specific conditions for that system.

We say that a sequent $\Gamma \vdash \Delta$ is *derivable* if it admits a derivation in SeqS_{23}, i.e. a proof tree, obtained by applying backwards the rules of the calculi, having $\Gamma \vdash \Delta$ as a root and whose leaves are all instances of closing rules, i.e. rules with valid sequents called (AX). As usual, the idea is as follows: in order to prove that a formula F is valid in a conditional logic, then one has

to check whether the sequent $\vdash x : F$ is derivable in SeqS$_{23}$, i.e. if we can obtain a proof tree by applying backwards the rules, starting from the root $\vdash x : F$.

The novelty with respect to the calculi introduced in [17] and [20] is the following invertible rule for CS, no longer exploiting a label substitution, whose principal formula is a conditional $x : A \Rightarrow B$ on the right-hand side of the sequent:

$$\frac{\Gamma \vdash \Delta, x : A \Rightarrow B, x : A \qquad \Gamma \vdash \Delta, x : A \Rightarrow B, x : B}{\Gamma \vdash \Delta, x : A \Rightarrow B} (\mathbf{CS})$$

replacing the following one, introduced in [17]:

$$\frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, x : A \qquad \Gamma[x/u, y/u], u \xrightarrow{A} u \vdash \Delta[x/u, y/u]}{\Gamma, x \xrightarrow{A} y \vdash \Delta} (\mathbf{CS})$$

where $\Gamma[x/u, y/u]$ (respectively, $\Delta[x/u, y/u]$) is obtained from Γ (respectively, from Δ) by replacing every occurrence of x and y with a new label u.

The basic idea is as follows: when a conditional $x : A \Rightarrow B$ belongs to the right-hand side of a sequent, this means that there exists a world w selected by f for the world represented by x and A such that B does not hold in w. By the strong centering condition, if the world represented by x is an A world (left premise), then w is necessarily the world itself, then B does not hold in it, therefore $x : B$ is added in the right-hand side of the sequent of the right premise.

As an example, consider the following valid sequent in systems with axiom (CS):

$$x : A \wedge B \vdash x : (A \Rightarrow B) \vee (A \Rightarrow C) \vee (A \Rightarrow D)$$

In the calculi SeqS, the derivation could be as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}(AX) \quad \dfrac{\dfrac{}{u : A \vdash u : A}(AX) \quad \dfrac{\dfrac{}{v : A \vdash v : A}(AX) \quad \dfrac{}{w : B, \ldots, w \xrightarrow{A} w \vdash w : C, w : B, w : D}(AX)}{v : A, v : B, v \xrightarrow{A} v, v \xrightarrow{A} v, v \xrightarrow{A} y \vdash v : C, y : B, v : D}(\mathbf{CS})}{u : A, u : B, u \xrightarrow{A} u, u \xrightarrow{A} z, u \xrightarrow{A} y \vdash z : C, y : B, u : D}(\mathbf{CS})}{x : A, x : B, x \xrightarrow{A} w, x \xrightarrow{A} z, x \xrightarrow{A} y \vdash z : C, y : B, w : D}(\Rightarrow \mathbf{R})}{x : A, x : B, x \xrightarrow{A} z, x \xrightarrow{A} y \vdash z : C, y : B, x : A \Rightarrow D}(\Rightarrow \mathbf{R})}{x : A, x : B, x \xrightarrow{A} y \vdash y : B, x : A \Rightarrow C, x : A \Rightarrow D}(\Rightarrow \mathbf{R})}{x : A, x : B \vdash x : A \Rightarrow B, x : A \Rightarrow C, x : A \Rightarrow D}(\vee \mathbf{R})}{x : A, x : B \vdash x : A \Rightarrow B, x : (A \Rightarrow C) \vee (A \Rightarrow D)}(\vee \mathbf{R})}{x : A, x : B \vdash x : (A \Rightarrow B) \vee (A \Rightarrow C) \vee (A \Rightarrow D)}(\wedge \mathbf{L})}{x : A \wedge B \vdash x : (A \Rightarrow B) \vee (A \Rightarrow C) \vee (A \Rightarrow D)}$$

whereas in the novel calculi SeqS$_{23}$ one can obtain the following, shorter derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:A,x:B\vdash x:A,\dots}(AX) \quad \dfrac{}{x:A,x:B\vdash x:B,\dots}(AX)}{x:A,x:B\vdash x:A\Rightarrow B,x:A\Rightarrow C,x:A\Rightarrow D}(\mathbf{CS})}{x:A,x:B\vdash x:A\Rightarrow B,x:(A\Rightarrow C)\vee(A\Rightarrow D)}(\vee\mathbf{R})}{x:A,x:B\vdash x:(A\Rightarrow B)\vee(A\Rightarrow C)\vee(A\Rightarrow D)}(\vee\mathbf{R})}{x:A\wedge B\vdash x:(A\Rightarrow B)\vee(A\Rightarrow C)\vee(A\Rightarrow D)}(\wedge\mathbf{L})$$

SeqS$_{23}$

CK

$(AX)\ \Gamma,x:P\vdash\Delta,x:P\quad(P\in ATM)$ $\qquad$ $(AX)\ \Gamma,x:\bot\vdash\Delta$ $\qquad$ $(AX)\ \Gamma\vdash\Delta,x:\top$

$$(\neg\mathbf{L})\dfrac{\Gamma\vdash\Delta,x:A}{\Gamma,x:\neg A\vdash\Delta} \qquad (\neg\mathbf{R})\dfrac{\Gamma,x:A\vdash\Delta}{\Gamma\vdash\Delta,x:\neg A} \qquad (\vee\mathbf{L})\dfrac{\Gamma,x:A\vdash\Delta \quad \Gamma,x:B\vdash\Delta}{\Gamma,x:A\vee B\vdash\Delta}$$

$$(\vee\mathbf{R})\dfrac{\Gamma\vdash\Delta,x:A,x:B}{\Gamma\vdash\Delta,x:A\vee B} \qquad (\wedge\mathbf{L})\dfrac{\Gamma,x:A,x:B\vdash\Delta}{\Gamma,x:A\wedge B\vdash\Delta} \qquad (\wedge\mathbf{R})\dfrac{\Gamma\vdash\Delta,x:A \quad \Gamma\vdash\Delta,x:B}{\Gamma\vdash\Delta,x:A\wedge B}$$

$$(\to\mathbf{L})\dfrac{\Gamma\vdash\Delta,x:A \quad \Gamma,x:B\vdash\Delta}{\Gamma,x:A\to B\vdash\Delta} \qquad (\to\mathbf{R})\ \dfrac{\Gamma,x:A\vdash\Delta,x:B}{\Gamma\vdash\Delta,x:A\to B}$$

$$(\mathbf{EQ})\dfrac{u:A\vdash u:B \quad u:B\vdash u:A}{\Gamma,x\xrightarrow{A}y\vdash\Delta,x\xrightarrow{B}y} \qquad (\Rightarrow\mathbf{R})\ \dfrac{\Gamma,x\xrightarrow{A}y\vdash\Delta,x:A\Rightarrow B,y:B}{\Gamma\vdash\Delta,x:A\Rightarrow B}(y\notin\Gamma,\Delta)$$

$$(\Rightarrow\mathbf{L})\ \dfrac{\Gamma,x:A\Rightarrow B\vdash\Delta,x\xrightarrow{A}y \quad \Gamma,x:A\Rightarrow B,y:B\vdash\Delta}{\Gamma,x:A\Rightarrow B\vdash\Delta}$$

extensions

$$(\mathbf{CS})\dfrac{\Gamma\vdash\Delta,x:A\Rightarrow B,x:A \quad \Gamma\vdash\Delta,x:A\Rightarrow B,x:B}{\Gamma\vdash\Delta,x:A\Rightarrow B} \qquad (\mathbf{ID})\dfrac{\Gamma,x\xrightarrow{A}y,y:A\vdash\Delta}{\Gamma,x\xrightarrow{A}y\vdash\Delta}$$

$$(\mathbf{CEM})\dfrac{\Gamma\vdash\Delta,x:A\Rightarrow B,x\xrightarrow{A}y \quad \Gamma\vdash\Delta,x:A\Rightarrow B,y:B}{\Gamma\vdash\Delta,x:A\Rightarrow B} \qquad (\mathbf{MP})\dfrac{\Gamma\vdash\Delta,x\xrightarrow{A}x,x:A}{\Gamma\vdash\Delta,x\xrightarrow{A}x}$$

Figure 1: Rules of sequent calculi SeqS$_{23}$

.

The calculi SeqS$_{23}$ are shown in Figure 1. They satisfy basic structural properties, namely

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:A, x:B \vdash x:A \Rightarrow B, x:A}(AX) \quad \dfrac{}{x:A, x:B \vdash x:A \Rightarrow B, x:B}(AX)}{x:A, x:B \vdash x:A \Rightarrow B}(\mathbf{CS})}{x:A \wedge B \vdash x:A \Rightarrow B}(\wedge\mathbf{L})}{\vdash x:(A \wedge B) \rightarrow (A \Rightarrow B)}(\rightarrow \mathbf{R})$$

Figure 2: A derivation of CS in SeqS$_{23}$.

height-preserving admissibility of weakening, height-preserving invertibility of the rules (with the exception of $(\mathbf{EQ})$), height-preserving admissibility of contraction.

Moreover, the following properties are needed in order to prove the admissibility of the cut rule (see Theorem 7 below), in turn needed to prove the completeness of the calculi:

Proposition 5. *If $\Gamma \vdash \Delta, x \xrightarrow{A} y$ is derivable in SeqS$_{23}$ in systems with the rule $(\mathbf{CS})$, then either (i) $\Gamma \vdash \Delta$ is derivable in SeqS$_{23}$ or (ii) $x = y$ or (iii) there exists $x \xrightarrow{A'} y \in \Gamma$ such that $x \xrightarrow{A'} y \vdash x \xrightarrow{A} y$ is derivable in SeqS$_{23}$.*

Proof. (*sketch*) By induction on the height of the derivation of $\Gamma \vdash \Delta, x \xrightarrow{A} y$. Intuitively, if the transition formula $x \xrightarrow{A} y$ is used in the derivation, then either it is the principal formula in a backward application of (**MP**), then it must be $x = y$ and (ii) holds, otherwise it is the principal formula in a backward application of (**EQ**), therefore there exists another transition formula $x \xrightarrow{A'} y \in \Gamma$ such that $x \xrightarrow{A'} y \vdash x \xrightarrow{A} y$ is derivable in SeqS$_{23}$, thus (iii) holds. On the contrary, if $x \xrightarrow{A} y$ is not used in the derivation, then we have that also $\Gamma \vdash \Delta$ is derivable, thus (i) holds.

■

The following proposition states that, given a derivable sequent $\Gamma \vdash \Delta$, the sequent obtained by replacing in Γ and/or Δ one or more transition formulas $x \xrightarrow{A} y$ with $x \xrightarrow{A'} y$, where A and A' are equivalent[1], is derivable too. For instance, if $\Gamma, x \xrightarrow{A \vee B} y, x \xrightarrow{A \vee B} z \vdash \Delta$ is derivable, so are the sequents $\Gamma, x \xrightarrow{A \vee B} y, x \xrightarrow{\neg A \rightarrow B} z \vdash \Delta$ and $\Gamma, x \xrightarrow{\neg A \rightarrow B} y, x \xrightarrow{\neg A \rightarrow B} z \vdash \Delta$.

Proposition 6. *If $x \xrightarrow{A'} y \vdash x \xrightarrow{A} y$ and $\Gamma \vdash \Delta$ are derivable in SeqS$_{23}$, then $\Gamma' \vdash \Delta$ is also derivable in in SeqS$_{23}$, where Γ' is obtained by replacing in Γ any number of transition formulas $x \xrightarrow{A} y$ with $x \xrightarrow{A'} y$.*

The proof is by induction on the height of the derivation of $\Gamma \vdash \Delta$, is straightforward, therefore left to the reader in order to save space.

As mentioned, we can show that the following *cut* rule is admissible:

[1]As usual, this means that both $A \rightarrow A'$ and $A' \rightarrow A$ are valid.

Theorem 7. *The* cut *rule:*

$$\frac{\Gamma \vdash \Delta, F \qquad\qquad F, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\ (cut)$$

where F is any labelled formula, is admissible in SeqS$_{23}$, i.e. if $\Gamma \vdash \Delta, F$ and $F, \Gamma \vdash \Delta$ are derivable in SeqS$_{23}$, so is $\Gamma \vdash \Delta$.

Proof. We proceed in the standard way by a double induction over the complexity of the cut formula and the sum of the heights of the derivations of the two premises of cut, namely we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations. To save space, we show the most interesting case involving the novel rule $(\mathbf{CS})$ and we left the other cases to the reader. Consider the case in which the proof is ended as follows:

$$\dfrac{\dfrac{\begin{array}{c}\Gamma \vdash \Delta, x : A \Rightarrow B, x : A \\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : B\end{array}}{(**)\ \Gamma \vdash \Delta, x : A \Rightarrow B}\ (\mathbf{CS}) \qquad \dfrac{\begin{array}{c}(*)\ \Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \\ \Gamma, x : A \Rightarrow B, y : B \vdash \Delta\end{array}}{\Gamma, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow L)}{\Gamma \vdash \Delta}\ (cut)$$

Since weakening is admissible and the fact that $(**)$ is derivable, we have that also $(\clubsuit)\ \Gamma \vdash \Delta, x : A \Rightarrow B, x \xrightarrow{A} y$ is derivable. We can apply the inductive hypothesis on the sum of the heights to cut $(\clubsuit)$ and $(*)$ to obtain a derivation of $(\spadesuit)\ \Gamma \vdash \Delta, x \xrightarrow{A} y$. By Proposition 5 we have that either (i) $\Gamma \vdash \Delta$ is derivable, and we are done, or (ii) $x = y$, or (iii) there exists $x \xrightarrow{A'} y \in \Gamma$ such that $x \xrightarrow{A'} y \vdash x \xrightarrow{A} y$ is derivable. In case (ii), the proof is ended as follows:

$$\dfrac{\dfrac{\begin{array}{c}(1)\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : A \\ (2)\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : B\end{array}}{(3)\ \Gamma \vdash \Delta, x : A \Rightarrow B}\ (\mathbf{CS}) \qquad \dfrac{\begin{array}{c}(4)\ \Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} x \\ (5)\ \Gamma, x : A \Rightarrow B, x : B \vdash \Delta\end{array}}{(6)\ \Gamma, x : A \Rightarrow B \vdash \Delta}\ (\Rightarrow L)}{\Gamma \vdash \Delta}\ (cut)$$

Since weakening is height-preserving admissible, since (3) is derivable, so is $(3')\ \Gamma, x : B \vdash \Delta, x : A \Rightarrow B$. We can apply the inductive hypothesis on the sum of the heights of the derivations of the two premises to cut $(3')$ and (5), to obtain a derivation of $(7)\ \Gamma, x : B \vdash \Delta$. Similarly, since (6) is derivable, then so is $(6')\ \Gamma, x : A \Rightarrow B \vdash \Delta, x : B$, and we can apply the inductive hypothesis on the sum of the heights to cut $(6')$ and (2) and obtain a proof of $(8)\ \Gamma \vdash \Delta, x : B$. We can the conclude that $\Gamma \vdash \Delta$ is derivable by applying the inductive hypothesis on the complexity of the cut formula to (7) and (8).

In case (iii), by Proposition 6 and the fact that also $x \xrightarrow{A} y \vdash x \xrightarrow{A'} y$ is derivable, we have that $\Gamma', x \xrightarrow{A} y \vdash \Delta$ is derivable, where $\Gamma = \Gamma', x \xrightarrow{A'} y$. By applying the inductive hypothesis

on the complexity of the cut formula, we conclude by cutting it with ($\spadesuit$) $\Gamma \vdash \Delta, x \xrightarrow{A} y$ and we are done.

■

Theorem 8 (Soundness and completeness). *Given a conditional formula F, it is valid in a conditional logic if and only if it is derivable in the corresponding calculus of SeqS$_{23}$, that is to say $\models F$ if and only if $\vdash x : F$ is derivable in SeqS$_{23}$.*

Proof. Concerning the soundness, we have to prove that, if a sequent $\Gamma \vdash \Delta$ is derivable, then the sequent is valid. This can be done by induction on the height of the derivation of $\Gamma \vdash \Delta$. The basic cases are those corresponding to derivations of height 0, that is to say instances of (AX). It is easy to see that, in all these cases, $\Gamma \vdash \Delta$ is a valid sequent. As an example, consider $\Gamma, x : P \vdash \Delta, x : P$: consider every model $\mathcal{M}$ and every mapping I satisfying all formulas in the left-hand side of the sequent, then also $x : P$. This means that $I(x) \in [P]$, but then we have that $\mathcal{M}$ satisfies via I at least a formula in the right-hand side of the sequent, the same $x : P$. For the inductive step, we proceed by considering each rule of the calculi SeqS$_{23}$ in order to check that, if the premise(s) is (are) valid sequent(s), to which we can apply the inductive hypothesis, so is the conclusion. To save space, we only present the case of the novel $(\mathbf{CS})$, the other ones are left to the reader. Let us consider a derivation ended as follows:

$$\frac{(1)\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : A \qquad (2)\ \Gamma \vdash \Delta, x : A \Rightarrow B, x : B}{(3)\ \Gamma \vdash \Delta, x : A \Rightarrow B}\ (\mathbf{CS})$$

By inductive hypothesis, the sequents (1) and (2) are valid. By absurd, suppose that (3) is not, that is to say there exists a model $\mathcal{M}$ and a mapping I satisfying all formulas in Γ but falsifying all formulas in the right-hand side of the sequent, namely all formulas in Δ and $x : A \Rightarrow B$. Concerning the latter, there exists a world w such that $(*)$ $w \in f(I(x), [A])$ and $(**)$ $w \notin [B]$. By the validity of (1), we have that, since $\mathcal{M}$ satisfies via I all formulas in Γ, at least one formula in the right-hand side of the sequent must be satisfied by $\mathcal{M}$ via I too. Necessarily, we have that $\mathcal{M}$ satisfies $x : A$ (all formulas in Δ are falsified): this means that $I(x) \in [A]$. By the strong centering condition, it turns out that $f(I(x), [A]) \subseteq \{I(x)\}$. Given $(*)$, we have that $I(x) = w$. By the validity of (2), reasoning in the same way we have that $I(x) = w \in [B]$, contradicting $(**)$.

The completeness is an easy consequence of the admissibility of the *cut* rule (Theorem 7): by induction on the complexity of the formulas we can prove that, if a formula F is valid in a conditional logic, then $\vdash x : F$ is derivable in SeqS$_{23}$. It can be shown that axioms are derivable and that the set of derivable formulas is closed under (Modus Ponens), (RCEA), and (RCK). A derivation of **(CS)** is provided in Figure 2. For the other axioms and rules, we remind the reader to [17, 20] in order to save space.

■

The presence of labels and of the rules ($\Rightarrow$ **L**), ($\Rightarrow$ **R**), $(\mathbf{ID})$, $(\mathbf{MP})$, $(\mathbf{CEM})$, and $(\mathbf{CS})$, which increase the complexity of the sequent in a backward proof search, is a potential cause of a

non-terminating proof search. However, with a similar argument to the one proposed in [17], we can define a procedure that can apply such rules in a controlled way and introducing a finite number of labels, ensuring termination. Intuitively, it can be shown that it is useless to apply ($\Rightarrow$ **L**) and ($\Rightarrow$ **R**) on $x : A \Rightarrow B$ by introducing (looking backward) the same transition formula $x \xrightarrow{A} y$ more than once in each branch of a proof tree. Similarly, it is useless to apply (**ID**), (**MP**), (**CEM**), and (**CS**) on the same formula more than once in a backward proof search on each branch of a derivation. This leads to the decidability of the given logics:

Theorem 9 (Decidability). *Conditional Logics CK and all its extensions with axioms ID, MP, CS, CEM and all their combinations are decidable.*

It can be shown that provability in all the Conditional Logics considered is decidable in $O(n^2 \log n)$ space, we omit the proof which is essentially the same as in [17].

4. The Theorem Prover CondLean 4.0

In this section we introduce CondLean 4.0, a Prolog implementation of the calculi SeqS_{23} introduced in the previous section. As already mentioned, the prover improves the existing provers for that logics [19, 18, 21] and is inspired to the "lean" methodology, introduced by Beckert and Posegga in the middle of the 90s [22, 23, 24]: they have proposed a very elegant and extremely efficient first-order theorem prover, called leanT^AP, consisting of only five Prolog clauses. The basic idea of the "lean" methodology is "to achieve maximal efficiency from minimal means" [22] by writing short programs and exploiting the power of Prolog's engine as much as possible. Moreover, it is straightforward to prove soundness and completeness of the theorem prover by exploiting the one to one correspondence between axioms/rules of SeqS_{23} and clauses of CondLean 4.0.

Let us now describe the theorem prover CondLean 4.0 in detail.

Each component of a sequent is implemented by a list of formulas, partitioned into three sub-lists: atomic formulas, transitions and complex formulas. Atomic and complex formulas are implemented by a Prolog list of the form `[x,a]`, where `x` is a Prolog constant and `a` is a formula. A transition formula $x \xrightarrow{A} y$ is implemented by a Prolog list of the form `[x,a,y]`. Labels are implemented by Prolog constants. The sequent calculi are implemented by the predicate

prove(Gamma, Delta, Labels, Rcond, LCond, Tree)

which succeeds if and only if $\Gamma \vdash \Delta$ is derivable in SeqS_{23}, where `Gamma` and `Delta` are the lists implementing the multisets Γ and Δ, respectively, and `Labels` is the list of labels introduced in that branch. Each clause of the `prove` predicate implements one axiom or rule of SeqS_{23}. Further arguments `RCond` and `LCond` are used in order to ensure the termination of the proof search, essentially by restricting the application of rules ($\Rightarrow$ **L**) and ($\Rightarrow$ **R**), copying their principal formulas in both the premises (more details are provided here below). `Tree` is an output term: if the proof search succeeds, it matches a Prolog representation of the derivation found by the theorem prover.

The theorem prover proceeds as follows. First of all, if $\Gamma \vdash \Delta$ is an axiom, then the goal will succeed immediately by using the clauses for the axioms. If it is not, then the first applicable rule is chosen. The ordering of the clauses is such that the application of the branching rules is postponed as much as possible. Concerning the interplay among the rules dealing with conditional formulas on the right-hand side of a sequent, the new rule ($\mathbf{CS}$) rule is applied first: intuitively, this is needed in order to check whether the conditional formula under consideration can be handled by the current world itself, otherwise the rule ($\Rightarrow$ R), which introduces a new label in a backward proof search, it is first applied and, if this does not lead to a derivation, the rule ($\mathbf{CEM}$) – if available – is applied.

As mentioned here above, arguments `RCond` and `LCond` are used in order to ensure the termination of the proof search by controlling the application of the rules ($\Rightarrow \mathbf{L}$) and ($\Rightarrow \mathbf{R}$): indeed, these rules copy the conditional formula $x : A \Rightarrow B$ to which they are applied in their premises, therefore we need to avoid redundant applications that, otherwise, would lead to expand an infinite branch. As an example, `RCond` is a Prolog list containing all the formulas $x : A \Rightarrow B$ to which the rule ($\Rightarrow \mathbf{R}$) has been already applied on the current branch: such a rule will be then applied to $x : A \Rightarrow B$ only if it does not belong to the list `RCond`. A similar mechanism is implemented for extensions of CK, namely further suitable arguments are added to the predicate `prove` to keep track of the information needed to avoid useless and uncontrolled applications of the rules ($\mathbf{MP}$), ($\mathbf{ID}$), ($\mathbf{CEM}$), and ($\mathbf{CS}$), which copy their principal formulas in their premise(s).

Let us now present some clauses of CondLean 4.0. As a first example, the clause for the axiom checking whether the same atomic formula occurs in both the left and the right hand side of a sequent is implemented as follows:

```
prove([LitGamma,_,_],[LitDelta,_,_],_,_,_,tree(ax)) :-
  member(F,LitGamma),member(F,LitDelta),!.
```

It is easy to observe that the rule succeeds when the same labelled formula F belongs to both the right and the left hand side of the sequent under investigation, completing the proof search: indeed, no recursive call to the predicate `prove` is performed, and the output term `Tree` matches a representation of a leaf in the derivation (`tree(ax)`).

As another example, here is the clause implementing ($\Rightarrow \mathbf{L}$):

```
prove([LitGamma,TransGamma,ComplexGamma],[LitDelta,TransDelta,ComplexDelta],
        Labels, RCond, LCond, CS, tree(condL,SubTree1,SubTree2)) :-
  member([X,A => B],ComplexGamma),
  select([[X,A => B],Used],Cond,TempCond),
  member(Y,Labels),                                                    (i)
  \+member([Y,[X,A => B]],LCond),!,                                    (ii)
  put([Y,B],LitGamma,ComplexGamma,NewLitGamma,NewComplexGamma),
  prove([[[X, A => B],[[X,C,Y] | Used]] | TempCond],
    [LitGamma,TransGamma,ComplexGamma],
```

```
    [LitDelta,[[X,A,Y]|TransDelta],ComplexDelta],Labels),
  prove([[[X, A => B],[[X,C,Y] | Used]] | TempCond],
    [NewLitGamma,TransGamma,NewComplexGamma],
    [LitDelta,TransDelta,ComplexDelta],Labels).
```

The predicate `put` is used to put the labelled formula `[Y,A]` in the proper sub-list of the left-hand side of the sequent. The recursive calls to `prove` implement the proof search on the two premises. The output term `Tree` matches a Prolog term `tree(condL,SubTree1,SubTree2)`, representing a tree with two branches, corresponding to the subtrees of the two premises `SubTree1` and `SubTree2` obtained by an application of the rule `condL`.

As mentioned, in order to ensure termination, in lines i and ii the theorem prover checks whether $(\Rightarrow \mathbf{L})$ has been already applied on the current branch to the conditional formula $x : A \Rightarrow B$ by using the label y, in other words by introducing $x \xrightarrow{A} y$ and $y : B$ in the premises. This avoids useless applications of the rule, by adopting the same label.

Let us now show the code of the novel rule $(\mathbf{CS})$:

```
prove(Gamma,[LitDelta,TransDelta,ComplexDelta],
        Labels, RCond, LCond, CS, tree(cs,SubTree1,SubTree2)) :-
  member([X,A => B],ComplexDelta),
  \+member([X,A => B],CS),!,                                        (iii)
  put([X,A],LitDelta,ComplexDelta,LitDelta1,ComplDelta1),
  put([X,B],LitDelta,ComplexDelta,LitDelta2,ComplDelta2),
  prove(Gamma, [LitDelta1,TransDelta,ComplexDelta1],
      Labels, RCond, LCond, [ [X,A => B] | CS], SubTree1),
  prove(Gamma, [LitDelta2,TransDelta,ComplexDelta2],
      Labels, RCond, LCond, [ [X,A => B] | CS], SubTree2).
```

Also for this rule, in order to ensure termination, the theorem prover checks whether $(\mathbf{CS})$ has been already applied on the current branch to the conditional formula $x : A \Rightarrow B$, in order to avoid further – useless – applications (line iii).

In order to search a derivation of a sequent $\Gamma \vdash \Delta$, the theorem prover proceeds as follows. First, if $\Gamma \vdash \Delta$ is an axiom, the goal will succeed immediately by using the clauses for the axioms. If it is not, then the first applicable rule is chosen, e.g. if `ComplexGamma` contains a formula `[X,A and B]`, then the clause implementing the rule ($\wedge$ L) is applied, invoking `prove` on its unique premise. The prover proceeds in a similar way for the other rules. As already mentioned, the ordering of the clauses is such that the application of the branching rules is postponed as much as possible.

In order to check whether a formula is valid in one of the considered systems, one has just to invoke the following auxiliary predicate:

pr(Formula) or **pr(Formula,ProofTree)**

which wraps the `prove` predicate by a suitable initialization of its arguments.

5. Performance of CondLean 4.0

We have tested CondLean 4.0 and we have compared its performance with those of the last version of CondLean [21]. We have tested the theorem prover over both:

- a set of valid formulas
- a set of randomly generated formulas, either valid or not.

We have observed that, over valid formulas, the performances of CondLean 4.0 are improved of $12,35\%$ with respect to its predecessor. As an example, running both the provers over the formula

$$(A \wedge B_1 \wedge B_2 \wedge \ldots \wedge B_{50}) \rightarrow ((A \Rightarrow B_1) \vee (A \Rightarrow B_2) \vee \ldots \vee (A \Rightarrow B_{50}))$$

CondLean 4.0 is able to build a derivation in 27 ms, against the 567 ms needed by the prover adopting label substitution.

Over randomly generated formulas, the statistics are even better: CondLean 4.0 provides an improvement of the performances of about 35%. We can then observe that its performance are promising, especially concerning cases in which it has to provide a negative answer for a not valid formula: this is justified by the fact that the previous implementations have to spend a lot of time in computing the inefficient label substitution mechanism, needed to succeed.

CondLean 4.0 is available for free download at https://gitlab2.educ.di.unito.it/pozzato/condlean-4.0.git, where one can also find Prolog source files used in order to evaluate the performances described in this section.

6. Conclusions, Related Works, Future Issues

In this work we have introduced CondLean 4.0, a theorem prover for Conditional Logics implementing labelled sequent calculi for the basic system CK and the whole cube of extensions with well established axioms ID, MP, CEM, and CS. Concerning the last one, known as *conditional strong centering*, the calculi considered are new, obtained from existing calculi [20] by replacing a rule computing a label substitution with a standard one, and are of their own interest. The performances of CondLean 4.0 seem promising and are better than those of its predecessors [19, 18, 21]: this is quite obvious if we consider that, in the most recent version of the prover, condition (CS) is handled either by a complicated and inefficient label substitution or by the combination of the rules for (CEM) and (MP) in systems allowing both of them, since in these logics (CS) is derivable (Proposition 3). Such better performances witness that avoiding label substitution is a concrete step towards efficient theorem proving for Conditional Logics.

Concerning related works, we mention [17], where combinations of the conditional third excluded middle (CEM) and conditional modus ponens (MP) are neglected, and [25], where strong centering (CS) is replaced by the condition (CSO).

We plan to extend our work in several directions. First, we aim at extending the calculi and the implementation to stronger Conditional Logics. Moreover, we aim at extending the prover by adopting standard refinements, state of the art heuristics, and data structures, as well as by a graphical web interface for it. We also aim at extending the set of formulas adopted in the

performance evaluation, in order to provide a more detailed and structured comparison, for instance parametrizing statistics with respect to the level of nesting of the conditional operator $\Rightarrow$ in formulas. Last, we are currently working on an extension of CondLean 4.0 which is able to show a countermodel in case of a failed proof: in a XAI perspective, it is obviously crucial to provide a derivation showing that a formula/a sequent is valid as our prover currently does, however it is also important to show a counterexample when this is not the case.

References

[1] D. Lewis, Counterfactuals, Basil Blackwell Ltd (1973).

[2] D. Nute, Topics in Conditional Logic, Reidel, Dordrecht, 1980.

[3] R. Stalnaker, A theory of conditionals, in: N. Rescher (Ed.), Studies in Logical Theory, Blackwell, 1968, pp. 98–112.

[4] B. F. Chellas, Basic conditional logics, Journal of Philosophical Logic 4 (1975) 133–153.

[5] J. P. Burgess, Quick completeness proofs for some logics of conditionals, Notre Dame Journal of Formal Logic 22 (1981) 76–84.

[6] S. Kraus, D. Lehmann, M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, Artificial Intelligence 44 (1990) 167–207.

[7] J. P. Delgrande, A first-order conditional logic for prototypical properties, Artificial Intelligence 33 (1987) 105–130.

[8] N. Friedman, J. Y. Halpern, Plausibility measures and default reasoning, Journal of the ACM 48 (2001) 648–685.

[9] L. Giordano, V. Gliozzi, N. Olivetti, G. L. Pozzato, Analytic tableaux for KLM preferential and cumulative logics, in: G. Sutcliffe, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning, 12th International Conference, LPAR 2005, Montego Bay, Jamaica, December 2-6, 2005, Proceedings, volume 3835 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 666–681. URL: https://doi.org/10.1007/11591191_46. doi:10.1007/11591191_46.

[10] G. Grahne, Updates and counterfactuals, Journal of Logic and Computation 8 (1998) 87–117.

[11] L. Giordano, V. Gliozzi, N. Olivetti, Weak agm postulates and strong ramsey test: a logical formalization, Artificial Intelligence 168 (2005) 1–37.

[12] L. Giordano, V. Gliozzi, N. Olivetti, Iterated belief revision and conditional logic, Studia Logica 70 (2002) 23–47.

[13] D. M. Gabbay, L. Giordano, A. Martelli, N. Olivetti, M. L. Sapino, Conditional reasoning in logic programming, Journal of Logic Programming 44 (2000) 37–74.

[14] C. B. Schwind, Causality in action theories, Electronic Transactions on Artificial Intelligence (ETAI) 3 (1999) 27–50.

[15] L. Giordano, C. Schwind, Conditional logic of actions and causation, Artificial Intelligence 157 (2004) 239–279.

[16] V. Genovese, L. Giordano, V. Gliozzi, G. L. Pozzato, Logics in access control: a conditional approach, Journal of Logic and Computation 24 (2014) 705–762. URL: https://doi.org/10.1093/logcom/exs040. doi:10.1093/logcom/exs040.

[17] N. Olivetti, G. L. Pozzato, C. B. Schwind, A Sequent Calculus and a Theorem Prover for Standard Conditional Logics, ACM Transactions on Computational Logics (ToCL) 8 (2007).

[18] N. Olivetti, G. L. Pozzato, Condlean: A theorem prover for conditional logics, in: M. C. Mayer, F. Pirri (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2003, Rome, Italy, September 9-12, 2003. Proceedings, volume 2796 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 264–270. URL: https://doi.org/10.1007/978-3-540-45206-5_23. doi:10.1007/978-3-540-45206-5_23.

[19] N. Olivetti, G. L. Pozzato, Condlean 3.0: Improving condlean for stronger conditional logics, in: B. Beckert (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2005, Koblenz, Germany, September 14-17, 2005, Proceedings, volume 3702 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 328–332. URL: https://doi.org/10.1007/11554554_27. doi:10.1007/11554554_27.

[20] N. Panic, G. L. Pozzato, Efficient theorem proving for conditional logics with conditional excluded middle, in: R. Calegari, G. Ciatto, A. Omicini (Eds.), Proceedings of the 37th Italian Conference on Computational Logic, Bologna, Italy, June 29 - July 1, 2022, volume 3204 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 217–231. URL: http://ceur-ws.org/Vol-3204/paper_22.pdf.

[21] N. Olivetti, N. Panic, G. L. Pozzato, Labelled sequent calculi for conditional logics: Conditional excluded middle and conditional modus ponens finally together, in: A. Dovier, A. Montanari, A. Orlandini (Eds.), AIxIA 2022 - Advances in Artificial Intelligence - XXIst International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 - December 2, 2022, Proceedings, volume 13796 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 345–357. URL: https://doi.org/10.1007/978-3-031-27181-6_24. doi:10.1007/978-3-031-27181-6_24.

[22] B. Beckert, J. Posegga, leantap: Lean tableau-based deduction, Journal of Automated Reasoning 15 (1995) 339–358.

[23] B. Beckert, J. Posegga, Logic programming as a basis for lean automated deduction., Journal of Logic Programming 28 (1996) 231–236.

[24] M. Fitting, leantap revisited, Journal of Logic and Computation 8 (1998) 33–47.

[25] R. Alenda, N. Olivetti, G. L. Pozzato, Nested sequent calculi for normal conditional logics, Journal of Logic and Computation 26 (2016) 7–50. doi:10.1093/logcom/ext034.