Nested Sequent Calculi and Theorem Proving for Normal Conditional Logics

Nicola Olivetti[1] and Gian Luca Pozzato[2]

[1] Aix-Marseille Université, CNRS, LSIS UMR 7296 - France - nicola.olivetti@univ-amu.fr
[2] Dipartimento di Informatica - Università di Torino - Italy gianluca.pozzato@unito.it

Abstract. In this paper we focus on proof methods and theorem proving for normal conditional logics, by describing nested sequent calculi as well as a theorem prover for them. Nested sequent calculi are a useful generalization of ordinary sequent calculi, where sequents are allowed to occur within sequents. Nested sequent calculi have been profitably employed in the area of (multi)-modal logic to obtain analytic and modular proof systems for these logics. In this work, we describe nested sequent calculi recently introduced for the basic conditional logic CK and some of its significant extensions. We also provide a calculus for Kraus Lehman Magidor cumulative logic C. The calculi are internal (a sequent can be directly translated into a formula), cut-free and analytic. Moreover, they can be used to design (sometimes optimal) decision procedures for the respective logics, and to obtain complexity upper bounds. Our calculi are an argument in favour of nested sequent calculi for modal logics and alike, showing their versatility and power. We also describe NESCOND, a Prolog implementation of nested sequent calculi mentioned above. NESCOND (NESted sequent calculi for propositional CONDitional logics) is inspired by the methodolody of leanT^AP. The paper also shows some experimental results, witnessing that the performances of NESCOND are promising. NESCOND is available at http://www.di.unito.it/∼pozzato/nescond/

1 Introduction

Conditional logics extend classical logic by means of a conditional operator $\Rightarrow$. They can be seen as a generalization of (multi)modal logics, where the modality $\Rightarrow$ is indexed by a formula of the same language. Conditional logics have a long history: Lewis [15, 16] introduced them in order to formalize a kind of hypothetical reasoning: the conditional formula $A \Rightarrow B$ is used to formalize a sentence like "if A were the case then B" that cannot be captured by classical logic with material implication. One original motivation was to formalize *counterfactual sentences*, i.e. conditionals of the form "if A were the case then B would be the case", where A is false.

Over the years, conditional logics firmly established themselves in various fields of artificial intelligence and knowledge representation. Just to mention a few, they have been used[3] to reason about prototypical properties [10] and to model belief change [12, 11]. Moreover, they can provide an axiomatic foundation of nonmonotonic reasoning [7, 14]: in detail, a conditional $A \Rightarrow B$ is read as "in normal circumstances, if A then B". Recently, constructive conditional logics have been applied to reason about *access control* policies [9, 8]: the statement A **says** B, intuitively meaning that a user/program

[3] We refer to [20, 1] for a complete bibliography about conditional logics.

A *asserts* B to hold in the system, can be naturally expressed by a conditional $A \Rightarrow B$. Finally, a kind of (multi)-conditional logics [3, 6] have been used to formalize epistemic change in a multi-agent setting and in some kind of epistemic "games", each conditional operator expresses the "conditional beliefs" of an agent.

All conditional logics enjoy a possible world semantics, with the intuition that a conditional $A \Rightarrow B$ is true in a world x if B is true in the set of worlds where A is true and that are most similar to/closest to/"as normal as" x. Since there are different ways of formalizing "the set of worlds similar/closest/..." to a given world, there are expectedly rather different semantics for conditional logics, from the most general selection function semantics to the stronger sphere semantics.

However, from the point of view of proof-theory and automated deduction, conditional logics have not achieved a state of the art comparable with, say, the one of modal logics, where there are well-established calculi, whose proof-theoretical and computational properties are well-understood. In this work we first describe *nested sequent calculi*, called $\mathcal{NS}$, for propositional conditional logics, recently introduced in [1, 2]. Nested sequent calculi, introduced by Kashima in [13] for classical modal logics, are a natural generalization of ordinary sequent calculi where sequents are allowed to occur within sequents. However, a nested sequent always corresponds to a formula of the language, so that we can think of the rules as operating "inside a formula", combining subformulas rather than just combining outer occurrences of formulas as in ordinary sequent calculi.

We will consider the basic normal conditional logic CK and its extensions with ID, MP and CEM, as well as the cumulative logic **C** introduced in [14] which corresponds to the *flat* fragment (i.e., without nested conditionals) of CK+CSO+ID. The calculi are rather natural, all rules have a fixed number of premises. Completeness is established by cut-elimination, whose peculiarity is that it must take into account the substitution of equivalent antecedents of conditionals (a condition corresponding to *normality*). The calculi can be used to obtain a decision procedure for the respective logics by imposing some restrictions preventing redundant applications of rules. In all cases, we get a PSPACE upper bound, a bound that for CK and its extensions with ID and MP is optimal (but not for CK+CEM that is known to be coNP). For flat CK+CSO+ID = cumulative logic **C** we also get a PSPACE bound, we are not aware of a better upper bound for this logic (although we may suspect that it is not optimal).

Furthermore, we describe an implementation of $\mathcal{NS}$ calculi in Prolog. The program, called NESCOND, gives a PSPACE decision procedure for the respective logics, and it is inspired by the methodology introduced by the system leanT^AP [4], even if it does not fit its style in a rigorous manner. The basic idea is that each axiom or rule of the nested sequent calculi is implemented by a Prolog clause of the program. The resulting code is therefore simple and compact: the implementation of NESCOND for CK consists of only 6 predicates, 24 clauses and 34 lines of code. We also provide some experimental results to show that the performances of NESCOND are promising, especially compared to the ones of CondLean [17, 18] and GOALD$\mathcal{U}$CK [19], to the best of our knowledge the only existing provers for conditional logics. This shows that nested sequent calculi are not only a proof theoretical tool, but they can be the basis of efficient theorem proving for conditional logics.

2 Normal Conditional Logics

We consider a propositional conditional language $\mathcal{L}$ over a set ATM of propositional variables. Formulas of $\mathcal{L}$ are built as usual: $\bot$, $\top$ and the propositional variables in ATM are *atomic formulas*; if A and B are formulas, then $\neg A$ and $A \otimes B$ are *compound formulae*, where $\otimes \in \{\wedge, \vee, \rightarrow, \Rightarrow\}$. We adopt the *selection function semantics*.

Definition 1 (Selection function semantics). *A model is a triple* $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$*:*
- $\mathcal{W}$ *is a non empty set of worlds;*
- $f : \mathcal{W} \times 2^{\mathcal{W}} \longmapsto 2^{\mathcal{W}}$ *is the* selection function*;*
- $[\,]$ *is the* evaluation function*, which assigns to an atom* $P \in ATM$ *the set of worlds where* P *is true, and is extended to boolean formulas as follows:*
  - $[\top] = \mathcal{W}$*;*
  - $[\bot] = \emptyset$*;*
  - $[\neg A] = \mathcal{W} - [A]$*;*
  - $[A \wedge B] = [A] \cap [B]$*;*
  - $[A \vee B] = [A] \cup [B]$*;*
  - $[A \rightarrow B] = [B] \cup (\mathcal{W} - [A])$*;*
  - $[A \Rightarrow B] = \{w \in \mathcal{W} \mid f(w, [A]) \subseteq [B]\}$.

A formula $F \in \mathcal{L}$ *is valid in a model* $\mathcal{M} = \langle \mathcal{W}, f, [\,] \rangle$*, and we write* $\mathcal{M} \models F$*, if* $[F] = \mathcal{W}$*. A formula* $F \in \mathcal{L}$ *is valid, and we write* $\models F$*, if it is valid in every model, that is to say* $\mathcal{M} \models F$ *for every* $\mathcal{M}$*.*

The semantics above characterizes the *basic conditional system*, called CK, where no specific properties of the selection function are assumed. An axiomatization of CK is given by ($\vdash$ denotes provability in the axiom system):

- any axiomatization of the classical propositional calculus (prop)
- If $\vdash A$ and $\vdash A \rightarrow B$, then $\vdash B$ (Modus Ponens)
- If $\vdash A \leftrightarrow B$ then $\vdash (A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ (RCEA)
- If $\vdash (A_1 \wedge \cdots \wedge A_n) \rightarrow B$ then $\vdash (C \Rightarrow A_1 \wedge \cdots \wedge C \Rightarrow A_n) \rightarrow (C \Rightarrow B)$ (RCK)

Moreover, we consider the following standard extensions of the basic system CK:

System	Axiom	Model condition
ID	$A \Rightarrow A$	$f(w, [A]) \subseteq [A]$
CEM	$(A \Rightarrow B) \vee (A \Rightarrow \neg B)$	$\mid f(w, [A]) \mid \leq 1$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	$w \in [A]$ implies $w \in f(w, [A])$
CSO	$(A \Rightarrow B) \wedge (B \Rightarrow A) \rightarrow ((A \Rightarrow C) \rightarrow (B \Rightarrow C))$	$f(w, [A]) \subseteq [B]$ and $f(w, [B]) \subseteq [A]$ implies $f(w, [A]) = f(w, [B])$

3 Nested Sequent Calculi $\mathcal{N}S$ for Conditional Logics

In this section we recall nested sequent calculi $\mathcal{N}S$ introduced in [1, 2], where S is an abbreviation for CK{+X}, with X $\in$ {CEM, ID, MP, ID+MP, CEM+ID}. We are able to deal with the basic normal conditional logic CK and its extensions with axioms ID, MP and CEM. We are also able to deal with some combinations of them, namely the systems allowing ID with either MP or CEM. The problem of extending $\mathcal{N}S$ to the conditional

$$\Gamma(P, \neg P)\ \underset{P \in ATM}{(AX)} \qquad \Gamma(\top)\ (AX_\top) \qquad \Gamma(\neg\bot)\ (AX_\bot) \qquad \frac{\Gamma(A)}{\Gamma(\neg\neg A)}(\neg)$$

$$\frac{\Gamma(A) \qquad \Gamma(B)}{\Gamma(A \wedge B)}(\wedge^+) \qquad \frac{\Gamma(\neg A, \neg B)}{\Gamma(\neg(A \wedge B))}(\wedge^-) \qquad \frac{\Gamma(A, B)}{\Gamma(A \vee B)}(\vee^+) \qquad \frac{\Gamma(\neg A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \vee B))}(\vee^-)$$

$$\frac{\Gamma(\neg A, B)}{\Gamma(A \to B)}(\to^+) \qquad \frac{\Gamma(A) \qquad \Gamma(\neg B)}{\Gamma(\neg(A \to B))}(\to^-) \qquad \frac{\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \qquad A, \neg A' \qquad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}(\Rightarrow^-)$$

$$\frac{\Gamma([A : B])}{\Gamma(A \Rightarrow B)}(\Rightarrow^+) \qquad \frac{\Gamma(\neg(A \Rightarrow B), A) \qquad \Gamma(\neg(A \Rightarrow B), \neg B)}{\Gamma(\neg(A \Rightarrow B))}(MP) \qquad \frac{\Gamma([A : \Delta, \Sigma], [B : \Sigma]) \quad A, \neg B \quad B, \neg A}{\Gamma([A : \Delta], [B : \Sigma])}(CEM)$$

$$\frac{\Gamma([A : \Delta, \neg A])}{\Gamma([A : \Delta])}(ID) \qquad \frac{\Gamma, \neg(A \Rightarrow B), [A' : \Delta, \neg B] \qquad \Gamma, \neg(A \Rightarrow B), [A : A'] \qquad \Gamma, \neg(A \Rightarrow B), [A' : A]}{\Gamma, \neg(A \Rightarrow B), [A' : \Delta]}(CSO)$$

Fig. 1. The nested sequent calculi $\mathcal{N}S$.

logics allowing both MP and CEM is open at present. As usual, the completeness of the calculi is an easy consequence of the admissibility of cut. We are also able to turn $\mathcal{N}S$ into a terminating calculus, which gives us a decision procedure for the respective conditional logics.
A nested sequent Γ is defined inductively as follows:

- a formula of $\mathcal{L}$ is a nested sequent;
- if A is a formula and Γ is a nested sequent, then $[A : \Gamma]$ is a nested sequent;
- a finite multiset of nested sequents is a nested sequent.

A nested sequent can be displayed as

$$A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n],$$

where $n, m \geq 0$, $A_1, \ldots, A_m, B_1, \ldots, B_n$ are formulas and $\Gamma_1, \ldots, \Gamma_n$ are nested sequents.

A nested sequent can be directly interpreted as a formula, just replace "," by $\vee$ and ":" by $\Rightarrow$. More explicitly, the interpretation of a nested sequent $A_1, \ldots, A_m, [B_1 : \Gamma_1], \ldots, [B_n : \Gamma_n]$ is inductively defined by the formula

$$\mathcal{F}(\Gamma) = A_1 \vee \ldots \vee A_m \vee (B_1 \Rightarrow \mathcal{F}(\Gamma_1)) \vee \ldots \vee (B_n \Rightarrow \mathcal{F}(\Gamma_n)).$$

The calculi $\mathcal{N}S$ are shown in Figure 1. As usual, we say that a nested sequent Γ is *derivable* in $\mathcal{N}S$ if it admits a *derivation*. A derivation is a tree whose nodes are nested sequents. A branch is a sequence of nodes $\Gamma_1, \Gamma_2, \ldots, \Gamma_n, \ldots$ such that each node Γ_i is obtained from its immediate successor Γ_{i-1} by applying *backward* a rule of $\mathcal{N}S$, having Γ_{i-1} as the conclusion and Γ_i as one of its premises. A branch is closed if one of its nodes is an instance of axioms (AX), $(AX_\top)$, $(AX_\bot)$, otherwise it is open. We say that a tree is closed if all its branches are closed. A nested sequent Γ has a derivation in $\mathcal{N}S$ if there is a closed tree having Γ as the root. As an example, Figure 2 shows a derivation in the calculus $\mathcal{N}$CK+ID of an instance of the axiom ID.

$$\dfrac{\dfrac{\dfrac{}{[P : P, \neg P]}\,(AX)}{[P : P]}\,(ID)}{P \Rightarrow P}\,(\Rightarrow^+)$$

Fig. 2. A derivation of an instance of the axiom ID in $\mathcal{N}$CK+ID.

We have also provided a nested sequent calculus for the flat fragment, i.e. without nested conditionals, of CK+CSO+ID, corresponding to KLM logic **C**. The rules of the calculus, called $\mathcal{N}\mathbf{C}_{\mathrm{KLM}}$, are those ones of $\mathcal{N}$CK+ID (restricted to the flat fragment) where the rule $(\Rightarrow^-)$ is replaced by the rule (CSO).

The specificity of nested sequent calculi is to allow inferences that apply within sequence. In order to introduce the rules of the calculus, we need the notion of context. Intuitively a context denotes a "hole", a *unique* empty position, within a sequent that can be filled by a sequent. We use the symbol ( ) to denote the empty context. A context is defined inductively as follows: $\Gamma(\ ) = \Lambda, (\ )$ is a context; if $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Lambda, [A : \Sigma(\ )]$ is a context. Finally, we define the result of filling "the hole" of a context by a sequent. Let $\Gamma(\ )$ be a context and Δ be a sequent, then the sequent obtained by filling the context by Δ, denoted by $\Gamma(\Delta)$ is defined as follows: if $\Gamma(\ ) = \Lambda, (\ )$, then $\Gamma(\Delta) = \Lambda, \Delta$; if $\Gamma(\ ) = \Lambda, [A : \Sigma(\ )]$ then $\Gamma(\Delta) = \Lambda, [A : \Sigma(\Delta)]$. The notions of derivation and of derivable sequent are defined as usual.

Nested sequent calculi $\mathcal{N}S$ are sound and complete with respect to the semantics for the respective logics.

Theorem 1. *The nested sequent calculi $\mathcal{N}S$ are sound and complete for the respective logics, i.e. a formula F of $\mathcal{L}$ is valid in CK+X if and only if it is derivable in $\mathcal{N}$CK+X.*

Proof. (*Soundness*): If Γ is derivable in $\mathcal{N}S$, then Γ is valid. To improve readability, we slightly abuse the notation identifying a sequent Γ with its interpreting formula $\mathcal{F}(\Gamma)$, thus we shall write $A \Rightarrow \Delta, \Gamma \wedge \Delta$, etc. instead of $A \Rightarrow \mathcal{F}(\Delta), \mathcal{F}(\Gamma) \wedge \mathcal{F}(\Delta)$. First, we prove that nested inference is sound, that is to say: let $\Gamma(\ )$ be any context. If the formula $A_1 \wedge \ldots \wedge A_n \rightarrow B$, with $n \geq 0$, is (CK{+X})-valid, then also $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is (CK{+X}) valid. The proof is by induction on the depth of a context $\Gamma(\ )$, defined as follows:

- $\Gamma(\ ) = \Delta, (\ )$ is a context with depth $d(\Gamma(\ )) = 0$;
- if $\Sigma(\ )$ is a context, $\Gamma(\ ) = \Delta, [A : \Sigma(\ )]$ is a context with depth $d(\Gamma(\ )) = 1 + d(\Sigma(\ ))$.

Let $d(\Gamma(\ )) = 0$, then $\Gamma = \Lambda, (\ )$. Since $A_1 \wedge \ldots \wedge A_n \rightarrow B$ is valid, by propositional reasoning, we have that also $(\Lambda \vee A_1) \wedge \ldots (\Lambda \vee A_n) \rightarrow (\Lambda \vee B)$ is valid, that is $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is valid. Let $d(\Gamma(\ )) > 0$, then $\Gamma(\ ) = \Delta, [C : \Sigma(\ )]$. By inductive hypothesis, we have that $\Sigma(A_1) \wedge \ldots \wedge \Sigma(A_n) \rightarrow \Sigma(B)$ is valid. By (RCK), we obtain that also $(C \Rightarrow \Sigma(A_1)) \wedge \ldots \wedge (C \Rightarrow \Sigma(A_n)) \rightarrow (C \Rightarrow \Sigma(B))$ is valid. Then, we get that $(\Lambda \vee (C \Rightarrow \Sigma(A_1))) \wedge \ldots \wedge (\Lambda \vee (C \Rightarrow \Sigma(A_n))) \rightarrow (\Lambda \vee (C \Rightarrow \Sigma(B)))$ is also valid, that is $\Gamma(A_1) \wedge \ldots \wedge \Gamma(A_n) \rightarrow \Gamma(B)$ is valid.

Now we can prove the soundness of the calculi, namely if Γ is derivable in $\mathcal{N}S$, then Γ is valid. By induction on the height of the derivation of Γ. In the base case, Γ is an axiom, that is $\Gamma = \Gamma(P, \neg P)$; then, trivially $P \vee \neg P$ is valid, then also $\Gamma(P, \neg P)$ is valid by the fact that nested inference is sound. Similarly for $\Gamma(\top)$ and $\Gamma(\neg\bot)$. For the inductive step, in order to save space, we only show the most interesting case of $(\Rightarrow^-)$: $\Gamma = \Gamma(\neg(A \Rightarrow B), [A' : \Delta])$ is derived from (i) $\Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B])$, (ii) $\neg A, A'$, (iii) $\neg A', A$. By inductive hypothesis we have that $A \leftrightarrow A'$ is valid. We show that also $(*)$ $[\neg(A \Rightarrow B) \vee (A' \Rightarrow (\Delta \vee \neg B))] \rightarrow [\neg(A \Rightarrow B) \vee (A' \Rightarrow \Delta)]$ is valid, then we conclude since nested inference is sound and by applying the inductive hypothesis. To prove (*), by (RCK) we have that the following is valid: $[(A' \Rightarrow B) \wedge (A' \Rightarrow (\Delta \vee \neg B))] \rightarrow (A' \Rightarrow \Delta)$. Since $A \leftrightarrow A'$ is valid, by (RCEA) we get that $(A \Rightarrow B) \rightarrow (A' \Rightarrow B)$ is valid, so that also $(A \Rightarrow B) \rightarrow ((A' \Rightarrow (\Delta \vee \neg B)) \rightarrow (A' \Rightarrow \Delta))$ is valid, then we conclude by propositional reasoning.

(*Completeness*): If Γ is valid, then Γ has a derivation in $\mathcal{N}S$. First of all, it can be shown that weakening and contraction are height-preserving admissible in $\mathcal{N}S$, that is to say: (weakening) if $\Gamma(\Delta)$ is derivable in $\mathcal{N}S$ with a derivation of height h, then also $\Gamma(\Delta, \Sigma)$ is derivable in $\mathcal{N}S$ with a proof of height $h' \leq h$, where Δ and Σ are nested sequents; (contraction) given a nested sequent Δ, if $\Gamma(\Delta, \Delta)$ has a derivation of height h, then also $\Gamma(\Delta)$ has a derivation of height $h' \leq h$. Moreover, the derivation of the contracted sequent $\Gamma(\Delta)$ does not add any rule application to the initial derivation.

Completeness is an easy consequence of the admissibility of the following rule *cut*:

$$\frac{\Gamma(F) \qquad\qquad \Gamma(\neg F)}{\Gamma(\emptyset)} \; (cut)$$

where F is a formula. The standard proof of admissibility of cut proceeds by a double induction over the complexity of F and the sum of the heights of the derivations of the two premises of (cut), in the sense that we replace one cut by one or several cuts on formulas of smaller complexity, or on sequents derived by shorter derivations. We only show the case of systems without MP and CSO, reminding to [2] for all the other details. However, in $\mathcal{N}S$ the standard proof does not work in the following case, in which the cut formula F is a conditional formula $A \Rightarrow B$:

$$\frac{\dfrac{(1)\ \Gamma([A : B], [A' : \Delta])}{(3)\ \Gamma(A \Rightarrow B, [A' : \Delta])}\,(\Rightarrow^+) \qquad \dfrac{(2)\ \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad A, \neg A' \quad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}\,(\Rightarrow^-)}{\Gamma([A' : \Delta])}\,(cut)$$

Indeed, even if we apply the inductive hypothesis on the heights of the derivations of the premises to cut (2) and (3), obtaining (modulo weakening, which is admissible) a derivation of $(2')$ $\Gamma([A' : \Delta, \neg B], [A' : \Delta])$, we cannot apply the inductive hypothesis on the complexity of the cut formula to $(2')$ and $(1')$ $\Gamma([A : \Delta, B], [A' : \Delta])$ (obtained from (1) again by weakening). Such an application would be needed in order to obtain a derivation of $\Gamma([A' : \Delta], [A' : \Delta])$ and then to conclude $\Gamma([A' : \Delta])$ since contraction is admissible: indeed, the two contexts are different, we have $[A' : \Delta, \neg B]$ in $(2')$ whereas we have $[A : \Delta, B]$ in $(1')$.

In order to tackle this problem, we need to prove another property, namely that if A and A' are equivalent, if a sequent $\Gamma([A : \Delta])$ is derivable in $\mathcal{N}S$, then also $\Gamma([A' : \Delta])$, obtained by replacing A with the equivalent formula A', is also derivable. In turn, we need (cut) to prove this property, therefore we prove both the properties (admissibility of (cut) and "substitution" of A with A') by mutual induction, namely:

In $\mathcal{N}S$, the following propositions hold:

– *(A)* If $\Gamma(F)$ and $\Gamma(\neg F)$ are derivable, so is $\Gamma(\emptyset)$, i.e. (cut) is admissible in $\mathcal{N}S$;
– *(B)* if (I) $\Gamma([A : \Delta])$, (II) $A, \neg A'$ and (III) $A', \neg A$ are derivable, then $\Gamma([A' : \Delta])$ is derivable.

Let us first consider (A). We have the following cases:

– (at least) one of the premises of (cut) is an instance of the axioms. Suppose that the left premise is an instance of (AX). In case it has the form $\Gamma(P, \neg P, F)$, then also $\Gamma(P, \neg P)$ is an instance of (AX) and we are done. Otherwise, we have $\Gamma(F, \neg F)$. In this case, the right premise of (cut) has the form $\Gamma(\neg F, \neg F)$, whereas the conclusion is $\Gamma(\neg F)$: since contraction is admissible, we conclude that $\Gamma(\neg F)$ is derivable and we are done. The other cases are symmetric. The cases in which one premise of (cut) is an instance of either $(AX_\top)$ or $(AX_\bot)$ are easy and left to the reader;
– the last step of *one* of the two premises is obtained by a rule (**R**) in which F is *not* the principal formula. This case is standard, we can permute (**R**) over the cut, i.e. we cut the premise(s) of (**R**) and then we apply (**R**) to the result of cut.
– F is the principal formula in the last step of *both* derivations of the premises of the cut inference. There are several subcases, to save space we only consider the above mentioned case in which the standard proof does not work, where the derivation is as follows:

$$\dfrac{\dfrac{(1)\ \Gamma(\neg(A \Rightarrow B), [A' : \Delta, \neg B]) \quad A, \neg A' \quad A', \neg A}{\Gamma(\neg(A \Rightarrow B), [A' : \Delta])}(\Rightarrow^-) \quad \dfrac{(2)\ \Gamma([A : B], [A' : \Delta])}{(3)\ \Gamma(A \Rightarrow B, [A' : \Delta])}(\Rightarrow^+)}{\Gamma([A' : \Delta])}(cut)$$

First of all, since we have proofs for $A, \neg A'$ and for $A', \neg A$ and the complexity of A is lower than the one of $A \Rightarrow B$, we can apply the inductive hypothesis for (B) to (2), obtaining a derivation of $(2')$ $\Gamma([A' : B], [A' : \Delta])$. Since weakening is admissible, from (3) we obtain a derivation of at most the same height of $(3')$ $\Gamma(A \Rightarrow B, [A' : \Delta, \neg B])$. We can then conclude as follows: we first apply the inductive hypothesis on the height for (A) to cut (1) and $(3')$, obtaining a derivation of (4) $\Gamma([A' : \Delta, \neg B])$. By weakening, we have also a derivation of $(4')$ $\Gamma([A' : \Delta, \neg B], [A' : \Delta])$. Again by weakening, from $(2')$ we obtain a derivation of $(2'')$ $\Gamma([A' : \Delta, B], [A' : \Delta])$. We then apply the inductive hypothesis on the complexity of the cut formula to cut $(2'')$ and $(4')$, obtaining a derivation of $\Gamma([A' : \Delta], [A' : \Delta])$, from which we conclude since contraction is admissible.

Concerning (B), we proceed by induction on the height h of the premise (I), as follows:

- Base case: $\Gamma([A : \Delta])$ is an instance of the axioms, that is to say either $\Delta = \Lambda(P, \neg P)$ or $\Delta = \Lambda(\top)$ or $\Delta = \Lambda(\neg\bot)$: we immediately conclude that also $\Gamma([A' : \Delta])$ is an instance of the axioms, and we are done;
- Inductive step: we have to consider all possible rules ending (looking forward) the derivation of $\Gamma([A : \Delta])$. We only show the most interesting case, when $(\Rightarrow^-)$ is applied by using $[A : \Delta]$ as principal formula. The derivation ends as follows:

$$\frac{(1)\ \Gamma(\neg(C \Rightarrow D), [A : \Delta, \neg D]) \qquad (2)\ C, \neg A \qquad (3)\ A, \neg C}{\Gamma(\neg(C \Rightarrow D), [A : \Delta])}\,(\Rightarrow^-)$$

We can apply the inductive hypothesis to (1) to obtain a derivation of $(1')$ $\Gamma(\neg(C \Rightarrow D), [A' : \Delta, \neg D])$. Since weakening is admissible, from (II) we obtain a derivation of (II') $C, A, \neg A'$, from (III) we obtain a derivation of (III') $A', \neg A, \neg C$. Again by weakening, from (2) and (3) we obtain derivations of $(2')$ $C, \neg A, \neg A'$ and $(3')$ $A', A, \neg C$, respectively. We apply the inductive hypothesis of (A) that is that cut holds for the formula A (of a given complexity c) and conclude as follows:

$$\cfrac{(1')\ \Gamma(\neg(C \Rightarrow D), [A' : \Delta, \neg D]) \quad \cfrac{\begin{array}{c}(II')\ C, A, \neg A' \\ (2')\ C, \neg A, \neg A'\end{array}}{C, \neg A'}\,(cut) \quad \cfrac{\begin{array}{c}(III')\ A', \neg A, \neg C \\ (3')\ A', A, \neg C\end{array}}{A', \neg C}\,(cut)}{\Gamma(\neg(C \Rightarrow D), [A' : \Delta])}\,(\Rightarrow^-)$$

With the admissibility of cut at hand, we can easily conclude the proof of completeness of $\mathcal{NS}$ by showing that the axioms are derivable and that the set of derivable formulas is closed under (Modus Ponens), (RCEA), and (RCK). A derivation of an instance of ID has been shown in Figure 2. A derivation of an instance of MP is as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{\neg(A \Rightarrow B), \neg A, B, A}\,(AX) \quad \cfrac{}{\neg(A \Rightarrow B), \neg A, B, \neg B}\,(AX)}{\neg(A \Rightarrow B), \neg A, B}\,(MP)}{\neg(A \Rightarrow B), A \to B}\,(\to^+)}{(A \Rightarrow B) \to (A \to B)}\,(\to^+)$$

Here is a derivation of an instance of CEM:

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{[A : B, \neg B], [A : \neg B]}\,(AX) \quad \cfrac{}{A, \neg A}\,(AX) \quad \cfrac{}{\neg A, A}\,(AX)}{[A : B], [A : \neg B]}\,(CEM)}{[A : B], A \Rightarrow \neg B}\,(\Rightarrow^+)}{A \Rightarrow B, A \Rightarrow \neg B}\,(\Rightarrow^+)}{(A \Rightarrow B) \vee (A \Rightarrow \neg B)}\,(\vee^+)$$

For (Modus Ponens), we have to show that, if (1) $A \to B$ ad (2) A are derivable, then also B is derivable. Since weakening is admissible, we have also derivations for $(1')$ $A \to B, B, \neg A$ and $(2')$ A, B. Furthermore, observe that (3) $A, B, \neg A$ and

(4) $\neg B, B, \neg A$ are both instances of (AX). Since cut is admissible (statement A above), the following derivation shows that B is derivable:

$$\dfrac{(2')\ A, B \qquad \dfrac{(1')\ A \to B, B, \neg A \qquad \dfrac{(3)\ A, B, \neg A \qquad (4)\ \neg B, B, \neg A}{\neg(A \to B), B, \neg A}\ (\to^-)}{B, \neg A}\ (cut)}{B}\ (cut)$$

For (RCEA), we have to show that if $A \leftrightarrow B$ is derivable, then also $(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)$ is derivable. As usual, $A \leftrightarrow B$ is an abbreviation for $(A \to B) \wedge (B \to A)$. Since $A \leftrightarrow B$ is derivable, and since $(\wedge^+)$ and $(\to^+)$ are invertible, we have a derivation for $A \to B$, then for (1) $\neg A, B$, and for $B \to A$, then for (2) $A, \neg B$. We derive $(A \Rightarrow C) \to (B \Rightarrow C)$ (the other half is symmetric) as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \Rightarrow C), [B : C, \neg C]}\ (AX) \qquad (1)\ \neg A, B \qquad (2)\ A, \neg B}{\neg(A \Rightarrow C), [B : C]}\ (\Rightarrow^-)}{\neg(A \Rightarrow C), B \Rightarrow C}\ (\Rightarrow^+)}{(A \Rightarrow C) \to (B \Rightarrow C)}\ (\to^+)$$

For (RCK), suppose that we have a derivation in $\mathcal{NS}$ of $(A_1 \wedge \ldots \wedge A_n) \to B$. Since $(\to^+)$ is invertible, we have also a derivation of $B, \neg(A_1 \wedge \ldots \wedge A_n)$. Since $(\wedge^-)$ is also invertible, then we have a derivation of $B, \neg A_1, \ldots, \neg A_n$ and, by weakening, of (1) $\neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2, \ldots, \neg A_n]$, from which we conclude as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{matrix}(1)\ \neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2, \ldots, \neg A_n] \\ \vdots\end{matrix}}{\neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B, \neg A_1, \neg A_2]}\ (\Rightarrow^-) \quad \dfrac{}{C, \neg C}\ (AX) \quad \dfrac{}{\neg C, C}\ (AX)}{\neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B, \neg A_1]}\ (\Rightarrow^-) \quad \dfrac{}{C, \neg C}\ (AX) \quad \dfrac{}{\neg C, C}\ (AX)}{\neg(C \Rightarrow A_1), \ldots, \neg(C \Rightarrow A_n), [C : B]}\ (\Rightarrow^-)}{\neg(C \Rightarrow A_1 \wedge \ldots \wedge C \Rightarrow A_n), [C : B]}\ (\wedge^-)}{\neg(C \Rightarrow A_1 \wedge \ldots \wedge C \Rightarrow A_n), C \Rightarrow B}\ (\Rightarrow^+)}{(C \Rightarrow A_1 \wedge \ldots \wedge C \Rightarrow A_n) \to (C \Rightarrow B)}\ (\to^+)$$

■

As usual, in order to obtain a decision procedure for the conditional logics under consideration, we have to control the application of the rules $(\Rightarrow^-)$/(CSO), (MP), (CEM), and (ID) that otherwise may be applied infinitely often in a backward proof search, since their principal formula is copied into the respective premise(s). In detail, we obtain a sound, complete and terminating calculus if we restrict the applications of these rules as follows [1, 2]: $(\Rightarrow^-)$ can be applied only once to each formula $\neg(A \Rightarrow B)$ with a context $[A' : \Delta]$ in each branch - the same for (CSO) in the system CK+CSO+ID; (ID) can be applied only once to each context $[A : \Delta]$ in each branch; (MP) can be applied only once to each formula $\neg(A \Rightarrow B)$ in each branch. For systems with (CEM), we need a more complicated mechanism: due to space limitations, we refer to [1] for this case. These results give a PSPACE decision procedure for their respective logics.

4 Design of NESCOND

In this section we present a Prolog implementation of the nested sequent calculi $\mathcal{N}S$. The program, called NESCOND (NESted sequent calculi for CONDitional logics), is inspired by the "lean" methodology of lean$T^A P$, even if it does not follow its style in a rigorous manner. The program comprises a set of clauses, each one of which implements a sequent rule or axiom of $\mathcal{N}S$. The proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism.

NESCOND represents a nested sequent with a Prolog list of the form:

```
              [F_1, F_2, ..., F_m,[[A_1,Gamma_1],AppliedConditionals_1],
 [[A_2,Gamma_2],AppliedConditionals_2], ..., [[A_n,Gamma_n],AppliedConditionals_n]] ]
```

In detail, elements of a nested sequent can be either formulas `F` or contexts. A context is represented by a pair `[Context,AppliedConditionals]` where:

– `Context` is also a pair of the form `[F,Gamma]`, where `F` is a formula of $\mathcal{L}$ and `Gamma` is a Prolog list representing a nested sequent;
– `AppliedConditionals` is a Prolog list `[A_1=>B_1,A_2=>B_2,...,A_k=>B_k]`, keeping track of the negated conditionals to which the rule $(\Rightarrow^-)$ has been already applied by using `Context` in the current branch. This is used in order to implement the restriction on the application of the rule $(\Rightarrow^-)$ in order to ensure termination.

Symbols $\top$ and $\bot$ are represented by constants `true` and `false`, respectively, whereas connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\Rightarrow$ are represented by `!`, `^`, `v`, `->`, and `=>`.

As an example, the Prolog list

```
[p, q, !(p => q), [[p, [q v !p, [[p,[p => r]],[]], !r]],[p => q]], [[q, [p, !p]],[]]]
```

represents the nested sequent

$$P, Q, \neg(P \Rightarrow Q), [P : Q \vee \neg P, [P : P \Rightarrow R], \neg R], [Q : P, \neg P].$$

Furthermore, the list `[p => q]` in the leftmost context is used to represent the fact that, in a backward proof search, the rule $(\Rightarrow^-)$ has already been applied to $\neg(P \Rightarrow Q)$ by using $[P : Q \vee \neg P, [P : P \Rightarrow R], \neg R]$.

Let us now discuss more in detail the implementation of NESCOND, starting from the description of some auxiliary predicates, and then distinguishing among the different conditional logics considered.

4.1 Auxiliary predicates

In order to manipulate formulas "inside" a sequent, NESCOND makes use of the three following auxiliary predicates:

– `deepMember(+Formulas,+NS)` succeeds if and only if either (i) the nested sequent `NS` representing a nested sequent Γ contains all the fomulas in the list `Formulas` or (ii) there exists a context `[[A,Delta],AppliedConditionals]` in `NS` such that `deepMember(Formulas,Delta)` succeeds, that is to say there is a nested sequent occurring in `NS` containing all the formulas of `Formulas`.
– `deepSelect(+Formulas,+NS,-NewNS)` operates exactly as `deepMember`, however it removes the formulas of the list `Formulas` by replacing them with a placeholder `hole`; the output term `NewNS` matches the resulting sequent.
– `fillTheHole(+NS,+Formulas,-NewNS)` replaces the placeholder `hole` in `NS` with the formulas in the list `Formulas`. The resulting sequent matches the output term `NewNS`.

4.2 NESCOND for CK

The calculi $\mathcal{N}S$ are implemented by the predicate

```
prove(+NS,-ProofTree).
```

This predicate succeeds if and only if the nested sequent represented by the list `NS` is derivable. When the predicate succeeds, then the output term `ProofTree` matches with a representation of the derivation found by the prover, used in order to display the proof tree. For instance, in order to prove that the formula $(A \Rightarrow (B \wedge C)) \rightarrow (A \Rightarrow B)$ is valid in CK, one queries NESCOND with the goal: `prove([(a => b ^ c) -> (a => b)],ProofTree).` Each clause of the `prove` predicate implements an axiom or rule of $\mathcal{N}S$. To search a derivation of a nested sequent Γ, NESCOND proceeds as follows. First of all, if Γ is an axiom, the goal will succeed immediately by using one of the following clauses for the axioms:

```
prove(NS,tree(ax)):-deepMember([P,!P],NS),!.
prove(NS,tree(axt)):-deepMember([top],NS),!.
prove(NS,tree(axb)):-deepMember([!bot],NS),!.
```

implementing (AX), $(AX_\top)$ and $(AX_\bot)$, respectively. If Γ is not an instance of the axioms, then the first applicable rule will be chosen, e.g. if a nested sequent in Γ contains a formula `A v B` then the clause implementing the $(\vee^+)$ rule will be chosen, and NESCOND will be recursively invoked on the unique premise of $(\vee^+)$. NESCOND 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 $(\Rightarrow^-)$ is as follows:

```
1.  prove(NS,tree(condn,A,B,Sub1,Sub2,Sub3)):-
2.      deepSelect([!(A => B),[[C,Delta],AppliedConditionals]],
           NS,NewNS),
3.      \+member(!(A => B),AppliedConditionals),!,
4.      prove([A,!C],Sub2),
5.      prove([C,!A],Sub3),
6.      fillTheHole(NewNS,[!(A => B),
            [[C,[!B|Delta]],[!(A => B)|AppliedConditionals]]],DefNS),
7.      prove(DefNS,Sub1).
```

In line 2, the auxiliary predicate `deepSelect` is invoked in order to find both a negated conditional $\neg(A \Rightarrow B)$ and a context $[C : \Delta]$ in the sequent (even in a nested subsequent). In this case, such formulas are replaced by the placeholder `hole`. Line 3 implements the restriction on the application of $(\Rightarrow^-)$ in order to guarantee termination: the rule is applied only if $\neg(A \Rightarrow B)$ does not belong to the list `AppliedConditionals` of the selected context. Since the rules of $\mathcal{NS}$ are invertible[4], a cut `!` is introduced in order to avoid useless backtrackings in the choice of the rule to apply. In lines 4, 5 and 7, NESCOND is recursively invoked on the three premises of the rule. In line 7, NESCOND is invoked on the premise in which the context $[C : \Delta]$ is replaced by $[C : \Delta, \neg B]$. To this aim, in line 6 the auxiliary predicate `fillTheHole(+NewNS,+Formulas,-DefNS)` is invoked to replace the `hole` in `NewNS`, introduced by `deepSelect`, with the negated conditional $\neg(A \Rightarrow B)$, which is copied into the premise, and the context $[C : \Delta, \neg B]$, whose list of `AppliedConditionals` is updated by adding the formula $\neg(A \Rightarrow B)$ itself.

4.3 NESCOND for extensions of CK

The implementation of the calculi for extensions of CK with axioms ID and MP are very similar. For systems allowing ID, contexts are triples `[Context, AppliedConditionals, AllowID]`. The third element `AllowID` is a flag used in order to implement the restriction on the application of the rule (ID), namely the rule is applied to a context only if `AllowID=true`:

```
prove(NS,tree(id,A,SubTree)):-
   deepSelect([[[A,Delta],AppliedConditionals,true]]],NS,NewNS),!,
   fillTheHole(NewNS,[[[A,[!A|Delta]],AppliedConditionals,false]]],DefNS),
   prove(DefNS,SubTree).
```

When (ID) is applied to `[Context, AppliedConditionals, true]` then the predicate `prove` is invoked on the unique premise of the rule `DefNS`, and the flag is set to `false` in order to avoid multiple applications in a backward proof search.

The restriction on the application of the rule (MP) is implemented by equipping the predicate `prove` by a third argument, `AppliedMP`, a Prolog list keeping track of the negated conditionals to which the rule has already been applied in the current branch. The clause of `prove` implementing (MP) is as follows:

[4] The rule $(\Rightarrow^-)$, as well as (CEM), are only "weakly" invertible, in the sense that only the leftmost premise of these rules can be obtained by weakening from the respective conclusions (see [2] for details).

```
1.  prove(NS,AppliedMP,tree(mp,A,B,Sub1,Sub2)):-
2.     deepSelect([!(A => B)],NS,NewNS),
3.     \+member(A => B,AppliedMP),!,
4.     fillTheHole(NewNS,[A,!(A => B)],NS1),
5.     fillTheHole(NewNS,[!B,!(A => B)],NS2),
6.     prove(NS1,[A => B|AppliedMP],Sub1),
7.     prove(NS2,[A => B|AppliedMP],Sub2).
```

The rule is applicable to a formula $\neg(A \Rightarrow B)$ only if `[A => B]` does not belong to `AppliedMP` (line 3). When (MP) is applied, then `[A => B]` is added to `AppliedMP` in the recursive calls of `prove` on the premises of the rule (lines 6 and 7).

The implemetation of the calculus for the flat fragment of CK+CSO+ID, corresponding to KLM cumulative logic **C**, is very similar to the one for CK+ID; the only difference is that the rule $(\Rightarrow^-)$ is replaced by (CSO). This does not make use of the predicate `deepSelect` to "look inside" a sequent to find the principal formulas $\neg(A \Rightarrow B)$ and $[C : \Delta]$: since the calculus only deals with the *flat* fragment of the logic under consideration, such principal formulas are directly selected from the current sequent by easy membership tests (standard Prolog predicates `member` and `select`), without searching inside other contexts.

As mentioned, NESCOND also deals with extensions with CEM; the clause implementing the rule (CEM) is as follows:

```
1. prove(NS,tree(cem,A,B,Sub1,Sub2,Sub3)):-
2.    deepSelect([[[A,Delta],ApplCond1],[[B,Sigma],ApplCond2]],
                                                  NS,NewNS),
3.    notSequentIncluded(Delta,Sigma),!,
4.    prove([A,!B],Sub2),
5.    prove([B,!A],Sub3),
6.    append(Delta,Sigma,ResDelta),
7.    fillTheHole(NewNS,[[[A,ResDelta],ApplCond1],
                    [[B,Sigma],ApplCond2]],DefNS),
8.   prove(DefNS,Sub1).
```

In line 3 the predicate `notSequentIncluded` is invoked: this predicate implements the restriction on the application of the rule described in the previous section in order to ensure termination.

5 Performances of NESCOND

The performances of NESCOND are promising. We have tested it by running SICStus Prolog 4.0.2 on an Apple MacBook Pro, 3.06 GHz Intel Core 2 Duo, 4GB RAM machine. We have performed two kind of tests:

– we have compared the performances of NESCOND for CK with the ones of two other provers for conditional logics, namely CondLean, implementing labelled sequent calculi [17, 18], and the goal-directed procedure GoalD$\mathcal{U}$CK [19]. To this aim, we have tested the three theorem provers on randomly generated sequents, obtaining the results shown in Figure 3. It is easy to conclude that the performances of NESCOND are better in all cases: in particular, concerning sequents containing formulas with a high level of conditional nesting (15), with a 1ms time limit, GOALD$\mathcal{U}$CK is not able to answer in the 41% of cases, whereas CondLean finds all the valid sequents but it is not able to find *any* non-valid ones. NESCOND answers correctly for the 100% of the generated formulas, discovering that all the missing answers of CondLean are non-valid ones;

– we have tested NESCOND over a set of 88 CK valid formulas obtained by translating K valid formulas[5] provided by Heuerding and used to test the theorem prover ModLeanTAP [5] by Beckert and Goré. Also in this case, the results, shown in Figure 4, are encouraging: NESCOND is not able to give an answer in less than 1 ms only in 16 cases over 88; the number of timeouts drops to 7 if we extend the time limit to 5 seconds.

number of formulas : 30	Prop. vars: 5	Depth: 3	Timeout: 1 ms
	yes	no	timeout
UCK	73,00%	21,00%	6,00%
CondLean	74,00%	0,00%	26,00%
Nested sequents	74,00%	26,00%	0,00%

number of formulas: 10	Prop. vars: 3	Depth: 15	Timeout: 1ms
	yes	no	timeout
UCK	55,00%	4,00%	41,00%
CondLean	69,00%	0,00%	31,00%
Nested sequents	69,00%	31,00%	0,00%

Fig. 3. NESCOND vs CondLean vs GOALD$\mathcal{U}$CK

Time limit (ms)	1	100	1000	5000	30000
NESCOND	16	13	10	7	5
CondLean	22	16	14	10	9

Fig. 4. Number of timeouts of NESCOND and CondLean over 88 CK valid formulas.

Time limit (ms)	1	50	100	5000
Percentage of timeouts	30%	28%	27%	20%

Fig. 5. Percentage of timeouts of NESCOND for extensions of CK.

These results show that the performances of NESCOND are encouraging, probably better than the ones of the other existing provers for conditional logics (we are currently completing the comparative statistics). Figure 5 shows that this also holds for extensions of CK: in the 70% of the tests (all of them are valid formulas), NESCOND gives an answer in less than 1ms. The remaining 30% concerns the case of CEM(+ID), where the performances worsen because of the overhead of the termination mechanism. We note in passim that it does not exist a set of acknowledged benchmarks for conditional logics. We are currently building a set of meaningful tests, they can be found at http://www.di.unito.it/∼pozzato/nescond/.

6 Conclusions and Future Issues

In this work we have presented NESCOND, a theorem prover for conditional logics implementing nested sequent calculi introduced in [1]. The performances described in the previous section show that nested sequent calculi do not only provide elegant and

[5] $\Box A$ is replaced by $\top \Rightarrow A$, whereas $\diamond A$ is replaced by $\neg(\top \Rightarrow \neg A)$.

natural calculi for conditional logics, but they are also significant for developing efficient theorem provers for them. In future research we aim to extend NESCOND to other systems of conditional logics. To this regard, we strongly conjecture that adding a rule for (CS) will be enough to cover the whole cube of the extensions of CK generated by axioms (ID), (MP), (CEM) and (CS). This will be object of subsequent research.

References

1. Alenda, R., Olivetti, N., Pozzato, G.L.: Nested Sequent Calculi for Conditional Logics. In: Luis Fariñas del Cerro, Andreas Herzig, J.M. (ed.) Logics in Artificial Intelligence - 13th European Conference, JELIA 2012. Lecture Notes in Artificial Intelligence (LNAI), vol. 7519, pp. 14–27. Springer-Verlag, Toulouse, France (Sptember 2012)
2. Alenda, R., Olivetti, N., Pozzato, G.L.: Nested Sequents Calculi for Normal Conditional Logics. Journal of Logic and Computation to appear, 1–48 (2013)
3. Baltag, A., Smets, S.: The logic of conditional doxastic actions. Texts in Logic and Games, Special Issue on New Perspectives on Games and Interaction 4, 9–31 (2008)
4. Beckert, B., Posegga, J.: leanT^AP: Lean tableau-based deduction. Journal of Automated Reasoning 15(3), 339–358 (1995)
5. Beckert, B., Goré, R.: Free variable tableaux for propositional modal logics. In: Proceedings of Tableaux 97. LNCS, vol. 1227, pp. 91–106. Springer, Pont-a-Mousson, France (1997)
6. Board, O.: Dynamic interactive epistemology. Games and Economic Behavior 49(1), 49–80 (2004)
7. Boutilier, C.: Conditional logics of normality: a modal approach. Artificial Intelligence 68(1), 87–154 (1994)
8. Genovese, V., Giordano, L., Gliozzi, V., Pozzato, G.L.: A conditional constructive logic for access control and its sequent calculus. In: Brünnler, K., Metcalfe, G. (eds.) Tableaux 2011. Lecture Notes in Artificial Intelligence (LNAI), vol. 6793, pp. 164–179. Springer-Verlag, Bern, Switzerland (July 2011)
9. Genovese, V., Giordano, L., Gliozzi, V., Pozzato, G.L.: Logics in access control: A conditional approach. Journal of Logic and Computation p. to appear (2012)
10. Ginsberg, M.L.: Counterfactuals. Artificial Intelligence 30(1), 35–79 (1986)
11. Giordano, L., Gliozzi, V., Olivetti, N.: Weak AGM postulates and strong ramsey test: A logical formalization. Artificial Intelligence 168(1-2), 1–37 (2005)
12. Grahne, G.: Updates and counterfactuals. Journal of Logic and Computation 8(1), 87–117 (1998)
13. Kashima, R.: Cut-free sequent calculi for some tense logics. Studia Logica 53(1), 119–136 (1994)
14. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artificial Intelligence 44(1-2), 167–207 (1990)
15. Lewis, D.: Counterfactuals. Basil Blackwell Ltd (1973)
16. Nute, D.: Topics in conditional logic. Reidel, Dordrecht (1980)
17. Olivetti, N., Pozzato, G.L.: CondLean: A Theorem Prover for Conditional Logics. In: Cialdea Meyer, M., Pirri, F. (eds.) Tableaux 2003. LNAI, vol. 2796, pp. 264–270. Springer, Roma, Italy (September 2003)
18. Olivetti, N., Pozzato, G.L.: CondLean 3.0: Improving Condlean for Stronger Conditional Logics. In: Beckert, B. (ed.) Proceedings of Tableaux 2005. LNAI, vol. 3702, pp. 328–332. Springer-Verlag, Koblenz, Germany (September 2005)
19. Olivetti, N., Pozzato, G.L.: Theorem Proving for Conditional Logics: CondLean and Goal-Duck. Journal of Applied Non-Classical Logics (JANCL) 18(4), 427–473 (2008)
20. Olivetti, N., Pozzato, G.L., Schwind, C.B.: A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. ACM Transactions on Computational Logic (ToCL) 8(4) (2007)