A Goal-Directed Calculus for Standard Conditional Logics

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

[1] LSIS - UMR CNRS 6168 Université Paul Cézanne (Aix-Marseille 3), Avenue Escadrille Normandie-Niemen 13397 Marseille Cedex 20 - France
nicola.olivetti@univ-cezanne.fr
[2] Dipartimento di Informatica - Università degli Studi di Torino, corso Svizzera 185 - 10149 Turin - Italy
pozzato@di.unito.it

Abstract. In this paper we focus on proof methods for conditional logics. We present $\mathcal{U}$S', a goal-directed calculus for the basic normal conditional logic CK and its standard extensions ID, MP, and ID+MP. $\mathcal{U}$S' is derived from some labelled sequent calculi, called SeqS', and it is based on the notion of uniform proofs. We also introduce GoalD$\mathcal{U}$CK, a simple implementation of $\mathcal{U}$S' written in SICStus Prolog[1].

1 Introduction

Conditional logics have a long history. They have been studied first by Lewis [26, 28, 3, 36] in order to formalize a kind of hypothetical reasoning (if A were the case then B), that cannot be captured by classical logic with material implication. In particular they were introduced to capture *counterfactual sentences*, i.e. conditionals of the form "if A were the case then B would be the case", where A is false. If we interpret the *if...then* in the above sentence as a classical implication, we obtain that all counterfactuals are trivially true. Nonetheless one may want to reason about counterfactuals sentences and hence be capable of distinguishing among true and false ones [4].

In the last years, there has been a considerable amount of work on applications of conditional logics to various areas of artificial intelligence and knowledge representation such as non-monotonic reasoning, hypothetical reasoning, belief revision, and even diagnosis.

The application of conditional logics to nonmonotonic reasoning was firstly investigated by Delgrande [8] who proposed a conditional logic for prototypical reasoning; the understanding of a conditional $A \Rightarrow B$ in his logic is "the A's have typically the property B". For instance, one could have:

$$\forall x(Penguin(x) \rightarrow Bird(x)), \forall x(Penguin(x) \rightarrow \neg Fly(x)),$$
$$\forall x(Bird(x) \Rightarrow Fly(x))$$

[1] This research has been partially supported by "*Progetto Lagrange - Fondazione CRT*" and by the projects "*MIUR PRIN05: Specification and verification of agent interaction protocols*" and *"GALILEO 2006: Interazione e coordinazione nei sistemi multi-agenti"*.

The last sentence states that birds typically fly. Observe that replacing $\Rightarrow$ with the classical implication $\rightarrow$, the above knowledge base is consistent only if there are no penguins. The study of the relations between conditional logics and nonmonotonic reasoning has gone much further since the seminal work by Kraus, Lehmann, and Magidor [24] (KLM framework), who proposed a formalization of the properties of a nonmonotonic consequence relation: their system comprises nonmonotonic assertions of the form $A \mathrel{|\!\sim} B$, interpreted as "B is a plausible conclusion of A". It turns out that all forms of inference studied in KLM framework are particular cases of well-known conditional axioms [6]. In this respect the KLM language is just a fragment of conditional logics. We refer to [10] for a broader discussion on the application of conditional logics to nonmonotonic reasoning.

Conditional logics have also been used to formalize knowledge update and revision. For instance, Grahne presents a conditional logic (a variant of Lewis' VCU) to formalize knowledge-update as defined by Katsuno and Mendelzon [23]. More recently, in [16] and [15], it has been shown a tight correspondence between AGM revision systems [1] and a specific conditional logic, called BCR. The connection between revision/update and conditional logics can be intuitively explained in terms of the so-called Ramsey Test (RT): the idea is that $A \Rightarrow B$ "holds" in a knowledge base K if and only if B "holds" in the knowledge base K revised/updated with A; this can be expressed by (RT) $K \vdash A \Rightarrow B$ iff $K \circ A \vdash B$, where $\circ$ denotes a revision/update operator.

Conditional logics have also been used to model hypothetical queries in deductive databases and logic programming; the conditional logic CK+ID is the basis of the logic programming language CondLP defined in [12]. In that language one can have hypothetical goals of the form (quoting the old Yale's Shooting problem) $Load\_gun \Rightarrow (Shoot \Rightarrow Dead)$ and the idea is that the hypothetical goal succeeds if $Dead$ succeeds in the state "revised" first by $Load\_gun$ and then by $Shoot$[2].

In a related context, conditional logics have been used to model causal inference and reasoning about action execution in planning [35, 22]. In the causal interpretation the conditional $A \Rightarrow B$ is interpreted as "A causes B"; observe that identity (i.e. $A \Rightarrow A$) is not assumed to hold.

Moreover, conditional logics have found some applications in diagnosis, where they can be used to reason counterfactually about the expected functioning of system components in face of the observed faults [29].

In spite of their significance, very few proof systems have been proposed for conditional logics. One possible reason of the underdevelopment of proof-methods for conditional logics is the lack of a universally accepted semantics for them. This is in sharp contrast to modal and temporal logics which have a consolidated semantics based on a standard kind of Kripke structures.

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

[2] The language CondLP comprises a nonmonotonic mechanism of revision to preserve the consistency of a program potentially violated by an hypothetical assumption.

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. The two most popular semantics for conditional logics are the so-called *sphere semantics* [26] and the *selection function semantics* [28]. Both are possible-world semantics, but are based on different (though related) algebraic notions. Here we adopt the selection function semantics, which is more general and considerably simpler than the sphere semantics.

With the selection function semantics, truth values are assigned to formulas depending on a world; intuitively, the selection function f selects, for a world w and a formula A, the set of worlds $f(w, A)$ which are "most-similar to w" or "closer to w" given the information A. In *normal* conditional logics, the function 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 (normality condition). A conditional sentence $A \Rightarrow B$ 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 [28]; it has the same role as the system K (from which it derives its name) in modal logic: CK-valid formulas are exactly those ones that are valid in every selection function model.

In this paper we first present a sequent calculus for CK and for some of its standard extensions, namely {ID, MP, ID+MP}. This calculus is called SeqS', and it makes use of labels following the line of [37] and [11]. Two types of formulas are involved in the rules of the calculi: world formulas of the form $x : A$ representing that A holds at world x and transition formulas of the form $x \xrightarrow{A} y$ representing that $y \in f(x, A)$. The rules manipulate both kinds of formulas.

We then show that SeqS' calculi can be the starting point to develop goal-directed proof procedures, according to the paradigm of Uniform Proofs by Miller and others [27, 13]. Calculi of these kind are suitable for logic programming applications. The basic idea of goal-directed proof search is as follows: given a sequent $\Gamma \vdash G$, one can interpret Γ as the program or database, and G as a goal whose proof is searched. The backward proof search of the sequent is driven by the goal G, in the sense that the goal is stepwise decomposed according to its logical structure. In a goal-directed proof method, the logical connectives can be interpreted operationally as simple and fixed search instructions. A proof of this sort is called a *uniform proof*. Given a sequent calculus for a logic L, in general, not every provable sequent admits a uniform proof: one must identify a significant fragment of L that allows uniform proofs. Usually, this fragment (in the propositional case) is alike to the Harrop-fragment of intuitionistic logic (see [27]). To specify this fragment one distinguish between the formulas which can occur in the database (D-formulas) and the formulas that can be asked as goals (G-formulas).

We present goal-directed proof procedures for a selected fragment of CK and its extensions with axioms ID and MP. We call these calculi $\mathcal{U}$S', where S' stands for {CK, ID, MP, ID+MP}.

Finally, we introduce GoalD$\mathcal{U}$CK, a very simple SICStus Prolog implementation of the calculi $\mathcal{U}$S'. As far as we know, no other goal-directed theorem

prover for the above standard conditional logics has been previously described in the literature.

The plan of the paper is as follows. In section 2 we introduce the conditional systems we consider, then we present the sequent calculi SeqS' for conditional logics in section 3. In section 4 we present the goal-directed proof procedure $\mathcal{U}$S' derived from SeqS'. In section 5 we present the theorem prover GoalD$\mathcal{U}$CK. In section 6 we discuss some related work and possible future research.

2 Conditional Logics

Conditional logics are extensions of classical logic by the conditional operator $\Rightarrow$. We restrict our concern to propositional conditional logics.

A propositional conditional language $\mathcal{L}$ is defined from: a set of propositional variables ATM; the symbols of *false* $\bot$ and *true* $\top$; a set of connectives $\neg$, $\rightarrow$, $\vee$, $\wedge$, $\Rightarrow$. We define formulas of $\mathcal{L}$ as follows:

- $\bot$, $\top$, and the propositional variables of ATM are *atomic formulas*;
- if A is a formula, then $\neg A$ is a *complex formula*;
- if A and B are formulas, $A \rightarrow B$, $A \vee B$, $A \wedge B$, and $A \Rightarrow B$ are *complex formulas*.

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. The two most popular semantics for conditional logics are the so-called *sphere semantics* [26] and the *selection function semantics* [28]. Both are possible-world semantics, but are based on different (though related) algebraic notions. In this work, we adopt the more general solution of the selection function semantics. We consider a non-empty set of possible worlds $\mathcal{W}$. Intuitively, the selection function f selects, for a world w and a formula A, the set of worlds of $\mathcal{W}$ which are *closer* to w given 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*.

Definition 1. *A selection function model is a triple $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$ where:*

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

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

Notice that we have defined f taking $[A]$ rather than A; 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 normal conditional system*, called CK. An axiomatization of the CK system is given by:

- any axiomatization of classical propositional logic
- (Modus Ponens) $\dfrac{A \quad A \rightarrow B}{B}$
- (RCEA) $\dfrac{A \leftrightarrow B}{(A \Rightarrow C) \leftrightarrow (B \Rightarrow C)}$
- (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 in modal logic, extensions of the basic system CK are obtained by assuming further properties on the selection function. Here we consider the following standard extensions:

Name	Axiom	Model condition
ID	$A \Rightarrow A$	$f(w, [A]) \subseteq [A]$
MP	$(A \Rightarrow B) \rightarrow (A \rightarrow B)$	if $w \in [A]$ then $w \in f(w, [A])$

3 Sequent Calculi SeqS'

In this section we present some sequent calculi for conditional logics. These calculi are called SeqS', where S' stands for {CK,ID,MP,ID+MP}[3], and they are inspired to labelled deductive systems introduced in [11] and in [37].

In Figure 1 we present SeqS'; the calculi make use of *labelled formulas*, where the labels are drawn from a denumerable set $\mathcal{A}$; there are two kinds of formulas:

- *world formulas*, denoted by $x : A$, where $x \in \mathcal{A}$ and $A \in \mathcal{L}$;
- *transition formulas*, denoted by $x \xrightarrow{A} y$, where $x,\ y \in \mathcal{A}$ and $A \in \mathcal{L}$.

A world formula $x : A$ is used to represent that A holds in the possible world represented by the label x; 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). The rigorous definition of sequent validity is stated as follows:

[3] In [33] sequent calculi SeqS' are also extended to the normal conditional logics allowing axioms CS and CEM. These calculi are called SeqS.

Definition 2 (Sequent validity). *Given a model* $\mathcal{M} = \langle \mathcal{W}, f, [\ ] \rangle$ *for* $\mathcal{L}$*, and a label alphabet* $\mathcal{A}$*, we consider any* mapping $I : \mathcal{A} \to \mathcal{W}$.

Let F *be a labelled formula, we define* $\mathcal{M} \models_I F$ *as follows:*

- $\mathcal{M} \models_I x{:}\, A$ iff $I(x) \in [A]$
- $\mathcal{M} \models_I x \xrightarrow{A} y$ iff $I(y) \in f(I(x), [A])$

We say that $\Gamma \vdash \Delta$ *is* valid *in* $\mathcal{M}$ *if for every mapping* $I : \mathcal{A} \to \mathcal{W}$*, 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 one of its extensions) if it is valid in every* $\mathcal{M}$ *satisfying the specific conditions for that system (if any).*

$(\mathbf{AX})\ \Gamma, x : P \vdash \Delta, x : P \quad (P \in ATM)$

$(\mathbf{A}\bot)\ \Gamma, x : \bot \vdash \Delta$ $\qquad$ $(\mathbf{A}\top)\ \Gamma \vdash \Delta, x : \top$

$$(\neg L)\ \frac{\Gamma \vdash \Delta, x : A}{\Gamma, x : \neg A \vdash \Delta} \qquad (\neg R)\ \frac{\Gamma, x : A \vdash \Delta}{\Gamma \vdash \Delta, x : \neg A}$$

$$(\wedge L)\ \frac{\Gamma, x : A, x : B \vdash \Delta}{\Gamma, x : A \wedge B \vdash \Delta} \qquad (\wedge R)\ \frac{\Gamma \vdash \Delta, x : A \qquad \Gamma \vdash \Delta, x : B}{\Gamma \vdash \Delta, x : A \wedge B}$$

$$(\vee L)\ \frac{\Gamma, x : A \vdash \Delta \qquad \Gamma, x : B \vdash \Delta}{\Gamma, x : A \vee B \vdash \Delta} \qquad (\vee R)\ \frac{\Gamma \vdash \Delta, x : A, x : B}{\Gamma \vdash \Delta, x : A \vee B}$$

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

$$(\Rightarrow L)\ \frac{\Gamma, x : A \Rightarrow B \vdash \Delta, x \xrightarrow{A} y \qquad \Gamma, x : A \Rightarrow B, y : B \vdash \Delta}{\Gamma, x : A \Rightarrow B \vdash \Delta} \qquad (\Rightarrow R)\ \frac{\Gamma, x \xrightarrow{A} y \vdash \Delta, y : B}{\Gamma \vdash \Delta, x : A \Rightarrow B}$$

$$(EQ)\ \frac{u : A \vdash u : B \qquad u : B \vdash u : A}{\Gamma, x \xrightarrow{A} y \vdash \Delta, x \xrightarrow{B} y}$$

$$(MP)\ \frac{\Gamma \vdash \Delta, x \xrightarrow{A} x, x : A}{\Gamma \vdash \Delta, x \xrightarrow{A} x} \qquad (ID)\ \frac{\Gamma, x \xrightarrow{A} y, y : A \vdash \Delta}{\Gamma, x \xrightarrow{A} y \vdash \Delta}$$

Fig. 1. Sequent calculi SeqS'. Rules (ID) and (MP) are only used in corresponding extensions of the basic system SeqCK.

Systems combining two or more semantic conditions are characterized by all rules capturing those conditions.

As an example, in Figure 2 we present a derivation in SeqID of the valid sequent $\vdash x : (P \to Q) \Rightarrow (P \to Q)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{x \overset{P\to Q}{\longrightarrow} y, y : P \vdash y : Q, y : P \qquad x \overset{P\to Q}{\longrightarrow} y, y : Q, y :, P \vdash y : Q}{x \overset{P\to Q}{\longrightarrow} y, y : P \to Q, y : P \vdash y : Q}\,(\to L)}{x \overset{P\to Q}{\longrightarrow} y, y : P \vdash y : Q}\,(ID)}{x \overset{P\to Q}{\longrightarrow} y \vdash y : P \to Q}\,(\to R)}{\vdash x : (P \to Q) \Rightarrow (P \to Q)}\,(\Rightarrow R)$$

Fig. 2. A derivation in SeqID for $\vdash x : (P \to Q) \Rightarrow (P \to Q)$.

SeqS' calculi are sound and complete with respect to the semantics:

Theorem 1 (Soundness and completeness [33]). A sequent $\Gamma \vdash \Delta$ is valid if and only if $\Gamma \vdash \Delta$ is derivable in SeqS'.

4 Goal-directed Proof Procedure for Conditional Logics

In this section we investigate how SeqS' calculi can be used in order to develop goal-directed proof procedures for conditional logics, following the paradigm of Uniform Proof by Miller and others [27, 13].

The paradigm of uniform proof can be seen as a generalization of conventional logic programming. Given a sequent $\Gamma \vdash G$, one can interpret Γ as the *program* or the *knowledge base* or the *database*, whereas G can be seen as a *goal* whose proof is searched. Intuitively, the basic idea is that the backward proof of $\Gamma \vdash G$ is driven by the goal G; roughly speaking, the goal G is stepwise decomposed according to its logical structure by the rules of the calculus, until its constituents are reached. The connectives in G can be interpreted operationally as search instructions. To prove an atomic goal Q, the proof search mechanism checks if Γ contains a "clause" whose head matches with Q and then tries to prove the "body" of the clause.

Given a sequent calculus, not every valid sequent admits a uniform proof of this kind; in order to describe a goal-directed proof search one must identify a fragment of the corresponding logic that allows uniform proofs.

Here we present a simple goal-directed calculus $\mathcal{U}$S', where S' stands for {CK, ID, MP, ID+MP}. First of all, we specify the fragment of the conditional language $\mathcal{L}$ we consider. We distinguish between the formulas which can occurs in the program (or knowledge base or database), called D-formulas, and the formulas that can be asked as goals, called G-formulas.

Definition 3 (Language for uniform proofs). *We consider the fragment of the conditional language $\mathcal{L}$, called $\mathcal{LU}$S', comprising:*

- database formulas, *denoted with* D
- goal formulas, *denoted with* G
- transition formulas *of the form* $x \xrightarrow{A} y$

defined as follows ($Q \in ATM$)*:*

$$D = G \rightarrow Q \mid A \Rightarrow D$$
$$G = Q \mid \top \mid G \wedge G \mid G \vee G \mid A \Rightarrow G$$
$$A = Q \mid A \wedge A \mid A \vee A$$

We define a database Γ *as a set of D-formulas and transition formulas.*

It is worth noticing that formulas of kind A can be either atomic formulas or combinations of conjunctions and disjunctions of atomic formulas.

The rules of the calculi $\mathcal{U}$S' are presented in Figure 3. $\mathcal{U}$S' 's rules are able to prove a goal which is either a formula $x : G$ or a transition formula $x \xrightarrow{A} y$; from now on we use γ to denote a goal of both kinds. Given a goal γ whose proof is searched from a database Γ, we call $\Gamma \vdash_{GD} \gamma$ a *query*. We write $\Gamma \vdash_{GD} \gamma \Rrightarrow \Gamma_i \vdash_{GD} \gamma_i$ to denote that the sequent $\Gamma \vdash_{GD} \gamma$ is reduced to sequents $\Gamma_i \vdash_{GD} \gamma_i$. The rule ($\mathcal{U}$ **prop**) is used when D-formulas have the form $A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q)$, where G could be $\top$.

($\mathcal{U}\top$) $\quad \Gamma \vdash_{GD} x : \top$

($\mathcal{U}$ ax) $\quad \Gamma \vdash_{GD} x : Q \quad$ if $\quad x : Q \in \Gamma$

($\mathcal{U}$ prop) $\quad \Gamma \vdash_{GD} x : Q \quad \Rrightarrow \quad \Gamma \vdash_{GD} y \xrightarrow{A_1} x_1, \Gamma \vdash_{GD} x_1 \xrightarrow{A_2} x_2, \ldots, \Gamma \vdash_{GD} x_{n-1} \xrightarrow{A_n} x, \Gamma \vdash_{GD} x : G$
if $\quad y : A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n \Rightarrow (G \rightarrow Q) \in \Gamma$

($\mathcal{U} \wedge$) $\Gamma \vdash_{GD} x : G_1 \wedge G_2 \quad \Rrightarrow \quad \Gamma \vdash_{GD} x : G_1 \quad$ and $\quad \Gamma \vdash_{GD} x : G_2$

($\mathcal{U} \vee$) $\Gamma \vdash_{GD} x : G_1 \vee G_2 \quad \Rrightarrow \quad \Gamma \vdash_{GD} x : G_1 \quad$ or $\quad \Gamma \vdash_{GD} x : G_2$

($\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ $\Gamma \vdash_{GD} x : A \Rightarrow G \quad \Rrightarrow \quad \Gamma, x \xrightarrow{A} y \vdash_{GD} y : G \quad (y$ new$)$

($\mathcal{U}$ trans) $\Gamma \vdash_{GD} x \xrightarrow{A} y \quad \Rrightarrow \quad \mathsf{Flat}(u : A') \vdash_{GD} u : A \quad$ and $\quad \mathsf{Flat}(u : A) \vdash_{GD} u : A' \quad$ if $\quad x \xrightarrow{A'} y \in \Gamma$ and $x \neq y$

($\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ $\Gamma \vdash_{GD} x : A \Rightarrow G \quad \Rrightarrow \quad \Gamma, x \xrightarrow{A} y, \mathsf{Flat}(y : A) \vdash_{GD} y : G \quad (y$ new$)$

($\mathcal{U}$ **trans**)$_{\mathbf{MP}}$ $\Gamma \vdash_{GD} x \xrightarrow{A} x \quad \Rrightarrow \quad \Gamma \vdash_{GD} x : A$

Fig. 3. Rules for the calculi $\mathcal{U}$S'.

The rule $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ belongs to $\mathcal{U}$CK and $\mathcal{U}$MP only, whereas $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ is used in $\mathcal{U}$ID and $\mathcal{U}$ID+MP. The rule $(\mathcal{U}\ \mathbf{trans})_{\mathbf{MP}}$ only belongs to the calculi $\mathcal{U}$MP and $\mathcal{U}$ID+MP.

A *proof* is a tree whose branches are sequences of nodes $\Gamma_i \vdash_{GD} \gamma_i$, where γ_i is a goal (i.e. either a formula $x : G$ or a transition formula $x \xrightarrow{A} y$). Each node $\Gamma_i \vdash_{GD} \gamma_i$ is obtained by its immediate predecessor $\Gamma_{i-1} \vdash_{GD} \gamma_{i-1}$ by applying a rule of $\mathcal{U}$S', i.e. $\Gamma_{i-1} \vdash_{GD} \gamma_{i-1} \Rrightarrow \Gamma_i \vdash_{GD} \gamma_i$. A branch is closed if one of its nodes is an instance of $(\mathcal{U}\ \top)$ or of $(\mathcal{U}$ ax), otherwise it is open. A *derivation* is a proof whose branches are all closed. We give the following (standard) definition:

Definition 4. *If Γ is a database and γ is a goal, then $\Gamma \vdash_{GD} \gamma$ is derivable in $\mathcal{U}$S' if there is a derivation in $\mathcal{U}$S' starting with $\Gamma \vdash_{GD} \gamma$.*

Given a formula of type A, i.e. either an atomic formula or a boolean combination of conjunctions and disjunctions of atomic formulas, the operation $\mathsf{Flat}(x : A)$ has the effect of *flatten* the conjunction/disjunction into sets of atomic formulas:

Definition 5 (Flat Operation).

- $\mathsf{Flat}(x : Q) = \{\{x : Q\}\}$, *with* $Q \in ATM$;
- $\mathsf{Flat}(x : F \vee G) = \mathsf{Flat}(x : F) \cup \mathsf{Flat}(x : G)$
- $\mathsf{Flat}(x : F \wedge G) = \{S_F \cup S_G \mid S_F \in \mathsf{Flat}(x : F)$ *and* $S_G \in \mathsf{Flat}(x : G)\}$

This operation is needed when *rules introducing a formula of type A in the database* are applied, since an A-formula might not be a D-formula. These rules, namely $(\mathcal{U}\ \mathbf{trans})$ and $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$, introduce a formula of type A in the left hand side of the sequent, i.e. a formula possibly being a combination of conjunctions and disjunctions of atoms. This formula needs to be decomposed in its atomic components. Indeed, in order to search a derivation for a transition formula $x \xrightarrow{A} y$, when $(\mathcal{U}\ \ \mathbf{trans})$ is applied by considering a transition $x \xrightarrow{A'} y$ in the program (or database), then the calculus leads to search a derivation for both $u : A' \vdash_{GD} u : A$ and $u : A \vdash_{GD} u : A'$; intuitively, this step corresponds to an application of the (EQ) rule in SeqS'. As an example, suppose that A has the form $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n$; in this case, in order to prove $u : A \vdash_{GD} u : A'$, we need to flat the database, thus proving the following sequent: $u : Q_1, u : Q_2, \ldots, u : Q_n \vdash_{GD} u : A'$. In case A has the form $Q_1 \vee Q_2$, then we need to prove both $u : Q_1 \vdash u : A'$ and $u : Q_2 \vdash u : A'$. The same happens when $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$ is applied to $\Gamma \vdash_{GD} x : A \Rightarrow B$, and the computation steps to prove $\Gamma, x \xrightarrow{A} y, y : A \vdash_{GD} y : B$, and $y : A$ needs to be decomposed as defined here above.

As another example, suppose that the formula $x : A$ is introduced in the database, and that $A = P \vee (Q \wedge (R \vee S))$. The Flat operation returns the following databases: $\mathsf{Flat}(x : A) = \{\{P\}, \{Q, R\}, \{Q, S\}\}$.

Moreover, we write $\Gamma, \mathsf{Flat}(x : A) \vdash_{GD} \gamma$ to denote that the goal γ can be derived from *all* the databases obtained by applying Flat to $x : A$ (and adding formulas in Γ), that is to say:

Definition 6. *Given a database Γ, a formula A and a goal G, let* $\mathsf{Flat}(x : A) = \{S_1, S_2, \ldots, S_n\}$. *We write* Γ, $\mathsf{Flat}(x : A) \vdash_{GD} \gamma$ *if and only if* $\forall i = 1, 2, \ldots, n$ *we have that* $\Gamma, S_i \vdash_{GD} \gamma$.

The goal-directed calculi $\mathcal{U}$S' are sound and complete wrt to the semantics, that is to say it can be shown that:

Theorem 2 (Soundness and Completeness of $\mathcal{U}$S'). *If Γ is a database, γ is a goal (i.e. either a formula $x : G$ or a transition formula $x \xrightarrow{A} y$), then $\Gamma \vdash_{GD} \gamma$ is derivable in $\mathcal{U}S'$ if and only if $\Gamma \vdash \gamma$ is derivable in the corresponding system SeqS'.*

In order to save space, we omit the proof, which is similar to the one of Theorem 6.5 presented in [33]. Soundness and completeness wrt the semantics immediately follow from the fact that SeqS' calculi are sound and complete wrt selection function models [33].

As an example of usage of the goal-directed proof procedures $\mathcal{U}$S', consider the following knowledge base Γ, representing the functioning of a simple DVD recorder:

- (1) $x : seton \Rightarrow ((pressRecButton \Rightarrow recording) \rightarrow readyToRec)$
- (2) $x : seton \Rightarrow pressRecButton \Rightarrow (sourceSelected \rightarrow recording)$
- (3) $x : seton \Rightarrow pressRecButton \Rightarrow (\top \rightarrow sourceSelected)$

We should interpret a conditional formula $A \Rightarrow B$ as "if the current state is updated with A then B holds" or, if A were an action, as "as an effect of A, B holds", or "having performed A, B holds". In this respect, clauses in Γ can be interpreted as: (1) "having set the device on, it is ready to record whenever it starts to record after pressing the REC button"; (2) "having set the device on and then having pressed the REC button, if the registration source has been selected, then the device will start to record"; (3) "having set the device on and then having pressed the REC button, it results that the registration source has been selected (the default source)". For a broader discussion on plausible interpretations of conditionals, we remind the reader to [35, 22, 12]; here we just observe that they have been widely used to express update/action/causation.

We show that the goal "Having set the DVD recorder on, is it ready to record?", formalized as $x : seton \Rightarrow readyToRec$ derives from Γ in $\mathcal{U}$CK. The derivation is presented in Figure 4.

5 The Theorem Prover GoalD$\mathcal{U}$CK

In this section we present GoalD$\mathcal{U}$CK, a very simple implementation of the calculi $\mathcal{U}$S'. GoalD$\mathcal{U}$CK is a SICStus Prolog program consisting of only eight clauses (resp. nine clauses in systems allowing MP), each one of them implementing a rule of the calculus[4], with the addiction of some auxiliary predicates and of

[4] For technical reasons, GoalD$\mathcal{U}$CK splits the rule ($\mathcal{U}$ **prop**) in two clauses, one taking care of applying it to the specific case of a goal $x : Q$ by using a clause $x : G \rightarrow Q$.

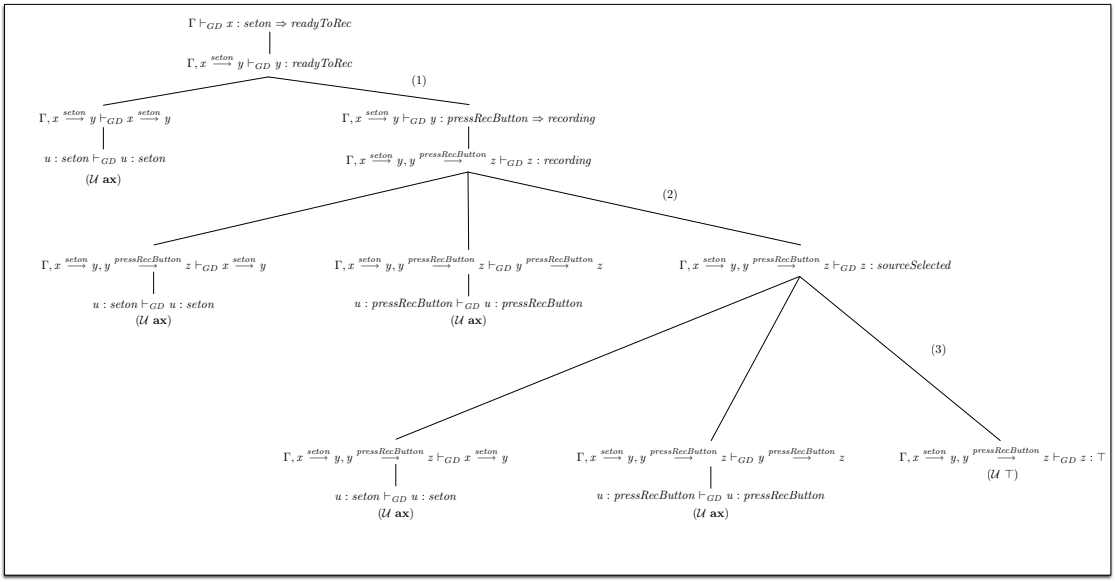

Fig. 4. A derivation of the goal $x : seton \Rightarrow readyToRec$. When ($\mathcal{U}$ **prop**) is applied, the corresponding edge is labelled with the number of the formula of the initial database used.

a predicate implementing the Flat operation. The SICStus Prolog program only consists of 52 lines of code.

The goal-directed proof search is implemented by a predicate

prove(Goal,Gamma,Trans,Labels).

which succeeds if and only if the goal G, represented by `Goal`, derives from the knowledge base Γ, represented by the list of world formulas `Gamma` and by the list of transition formulas `Trans`. Labelled formulas (both world and transition formulas) are represented by Prolog lists `[X,A]` and `[X,A,Y]`, respectively. `Labels` is the list of labels occurring in the current branch of the proof.

For instance, to prove if $x : A \Rightarrow B$ derives from the database $x : A \Rightarrow C \Rightarrow (\top \rightarrow B), x : B \Rightarrow ((A \Rightarrow B) \rightarrow C)$, one queries GoalD$\mathcal{U}$CK with the following goal: `prove([x,a => b],[[x, a => c => (true -> b)], [x,b => ((a => b) -> c)]],[],[x]).`

The predicate `prove` is also equipped with an additional argument `Tree`, instantiated by a functor `tree` and used to store the steps of a derivation and then give a justification of the proof. As mentioned above, each clause of the predicate `prove` implements an axiom or rule of the calculus $\mathcal{U}$S'. Here below we present the clauses implementing ($\mathcal{U} \Rightarrow$)$_{\mathbf{CK}}$, ($\mathcal{U}$ **prop**), and ($\mathcal{U}$ **trans**) as examples:

```
prove([X,A => G],Gamma,Trans,Labels):-
    generateLabel(Y,Labels),
    prove([Y,G],Gamma,[[X,A,Y]|Trans],[Y|Labels]).
```

```
prove([X,Q],Gamma,Trans,Labels):-
   member([Y,F=>(G->Q)],Gamma),atom(Q),
   prove([X,G],Gamma,Trans,Labels),extract(F,List),
   proveList(X,Y,List,Gamma,Trans,Labels).

prove([X,A,Y],_,Trans,_):-
   member([X,AP,Y],Trans),flat(A,FlattenedA),flat(AP,FlattenedAP),
   proveFlat([x,A],[],[],[x],x,FlattenedAP),
   proveFlat([x,AP],[],[],[x],x,FlattenedA).
```

The clause implementing $(\mathcal{U} \Rightarrow)_{\mathbf{CK}}$ is very intuitive: if $A \Rightarrow G$ is the current goal, then the `generateLabel` predicate introduces a new label `Y`, then the predicate `prove` is called to prove the goal $y : G$ from a knowledge base enriched by the transition $x \xrightarrow{A} y$. Obviously, in the versions of GoalD$\mathcal{U}$CK supporting CK+ID{+MP}, i.e. implementing the goal-directed calculi $\mathcal{U}$ID{+MP}, this clause is replaced by the one implementing the rule $(\mathcal{U} \Rightarrow)_{\mathbf{ID}}$.
The clause implementing ($\mathcal{U}$ **prop**) proceeds as follows: first, it searches for a clause $y : F \Rightarrow (G \rightarrow Q)$ in the database Γ, then it checks if Q is an atom, i.e. if $Q \in ATM$; second, it makes a recursive call to `prove` in order to find a derivation for the goal $x : G$; finally, it invokes two auxiliary predicates, `extract` and `proveList`, having the following functions:

- `extract` builds a list of the form `[`A_1`,` A_2`, ...,` A_n`]`, where $F = A_1 \Rightarrow A_2 \Rightarrow \ldots \Rightarrow A_n$;
- `proveList` invokes recursively the predicate `prove` in order to find a uniform proof for each goal $x_i \xrightarrow{A_{i+1}} x_{i+1}$ such that A_{i+1} belongs to the list generated by `extract`.

Given a goal `[X,A,Y]`, the clause implementing ($\mathcal{U}$ **trans**) is invoked. It first searches for a transition formula `[X,AP,Y]` in the database (i.e. in the list `Trans`), then it calls the predicate `flat` on both formulas `A` and `AP`; this predicate implements the Flat operation, building a *list of databases* obtained by flattening `A` (resp. `AP`) as defined in Definition 5. In order to prove $\mathsf{Flat}(u : A) \vdash_{GD} u : AP$ and $\mathsf{Flat}(u : AP) \vdash_{GD} u : A$, another auxiliary predicate, called `proveFlat`, is finally invoked by the clause implementing ($\mathcal{U}$ **prop**); `proveFlat` recursively invokes the predicate `prove` by using *all different databases* built by `flat`.

The performances of GoalD$\mathcal{U}$CK are promising. We have tested its performances and compared them with CondLean's. CondLean is a theorem prover implementing SeqS' calculi written in SICStus Prolog [30, 31]. We have implemented a SICStus Prolog program testing both GoalD$\mathcal{U}$CK and CondLean, versions for CK, in order to compare their performances. This program randomly generates a database containing a specific set of formulas, obtained by combining a fixed set of propositional variables ATM. Moreover, the program builds a set of goals, whose cardinality is specified as a parameter, that can be either derivable or not from the generated database. Each goal is obtained from a set ATM° of variables which is a *subset* of the set ATM of variables in the database, in order to study situations in which several formulas in the database

are *useless* to prove a goal. GoalD$\mathcal{U}$CK seems to offer better performances than CondLean. We have tested the two theorem provers over 100 goals, obtaining that GoalD$\mathcal{U}$CK is able to prove 81 goals, whereas CondLean answers positively in 72 cases. Moreover, GoalD$\mathcal{U}$CK concludes its work in 97 cases over 100 within the fixed time limit of 1 ms, even with a finite failure in 16 cases, whereas CondLean results in a time out in 28 cases.

We conclude this section by remarking that goal-directed proof methods usually do not ensure a terminating proof search. GoalD$\mathcal{U}$CK does not ensure termination too. Indeed, given a database Γ and a goal $x : G$, it could happen that, after several steps, the goal-driven computation leads to search a derivation for the same goal $x : G$ from a database Γ' such that $\Gamma' \supseteq \Gamma$ (that is to say: Γ' either corresponds to Γ or it is obtained from Γ by adding new facts). This problem is also well known in standard logic programming. As an example, consider the database containing the fact $x : (Q \wedge \top) \rightarrow Q$: querying GoalD$\mathcal{U}$CK with the goal $x : Q$, one can observe that the computation does not terminate: GoalD$\mathcal{U}$CK tries to apply ($\mathcal{U}$ **prop**) by using the only clause of the program (database), then it searches a derivation of the two subgoals $x : \top$ (and the computation succeeds) and $x : Q$. In order to prove this goal, GoalD$\mathcal{U}$CK repeats the above steps, then incurring in a loop.

6 Conclusions and Future Works

In this paper we have provided goal-directed calculi for normal conditional logics called $\mathcal{U}$S'. These calculi are sound and complete wrt to the semantics, if the language considered is restricted to a specific fragment allowing uniform proofs. Moreover, we have presented GoalD$\mathcal{U}$CK, a SICStus Prolog implementation of the calculi $\mathcal{U}$S'. GoalD$\mathcal{U}$CK is a simple program, consisting of only seven clauses. Each clause of GoalD$\mathcal{U}$CK implements an axiom or rule of the calculi $\mathcal{U}$S'. To the best of our knowledge, no other goal-directed calculi/theorem provers for conditional logics have been previously proposed in the literature. Further investigation could lead to the development of extensions of logic programming based on conditional logics. Related works on proof methods for conditional logics in the literature have concentrated on extensions of CK and do not present implementations of the deductive systems introduced.

De Swart [7] and Gent [14] give sequent/tableau calculi for the strong conditional logics VC and VCS. Their proof systems are based on the entrenchment connective $\leq$, from which the conditional operator can be defined.

Crocco and Fariñas [5] give sequent calculi for some conditional logics including CK, CEM, CO and others. Their calculi comprise two levels of sequents: principal sequents with $\vdash_P$ correspond to the basic deduction relation, whereas auxiliary sequents with $\vdash_a$ correspond to the conditional operator: thus the constituents of $\Gamma \vdash_P \Delta$ are sequents of the form $X \vdash_a Y$, where X, Y are sets of formulas.

Artosi, Governatori, and Rotolo [2] develop labelled tableau for the *first-degree* fragment (i.e. without nested conditionals) of the conditional logic CU

that corresponds to cumulative non-monotonic logics. In their work they use labels similarly to SeqS'. Formulas are labelled by path of worlds containing also variable worlds.

Lamarre [25] presents tableau systems for the conditional logics V, VN, VC, and VW. Lamarre's method is a consistency-checking procedure which tries to build a system of sphere falsifying the input formulas.

Groeneboer and Delgrande [9] have developed a tableau method for the conditional logic VN which is based on the translation of this logic into the modal logic S4.3.

In [19] and [20] a labelled tableau calculus for the logic CE and some of its extensions is presented. The flat fragment of CE corresponds to the non-monotonic preferential logic P and admits a semantics in terms of preferential structures (possible worlds together with a family of preference relations). The tableau calculus makes use of pseudo-formulas, that are modalities in a hybrid language indexed on worlds.

In [17, 18, 34] tableau calculi for nonmonotonic KLM logics are presented. These calculi, called $\mathcal{T}S^T$, are obtained by introducing suitable modalities to interpret conditional assertions. Moreover, these calculi have been implemented in SICStus Prolog: the program, called KLMLean, is inspired by the "*lean*" methodology and follows the style of CondLean [32, 21].

In future research we aim to develop goal-directed calculi for other conditional logics, in order to extend GoalD$\mathcal{U}$CK to support all of them. We also intend to investigate a broader language allowing uniform proofs.

Finally, we intend to evaluate the performances of GoalD$\mathcal{U}$CK by executing some other tests. As a consequence, in our future research we plan to increase its performances by experimenting standard refinements and heuristics.

References

1. C. E. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
2. A. Artosi, G. Governatori, and A. Rotolo. Labelled tableaux for non-monotonic reasoning: Cumulative consequence relations. *Journal of Logic and Computation*, 12(6):1027–1060, 2002.
3. B. F. Chellas. Basic conditional logics. *Journal of Philosophical Logic*, 4:133–153, 1975.
4. T. Costello and J. McCarthy. Useful counterfactuals. *ETAI (Electronic Transactions on Artificial Intelligence)*, 3:Section A, 1999.
5. G. Crocco and L. Fariñas del Cerro. Structure, consequence relation and logic, volume 4. *In D. M. Gabbay (ed.), What is a Logical System, Oxford University Press*, pages 239–259, 1995.
6. G. Crocco and P. Lamarre. On the connection between non-monotonic inference systems and conditional logics. *In B. Nebel and E. Sandewall editors, Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 565–571, 1992.

7. H. C. M. de Swart. A gentzen-or beth-type system, a practical decision procedure and a constructive completeness proof for the counterfactual logics vc and vcs. *Journal of Symbolic Logic*, 48(1):1–20, 1983.
8. J. P. Delgrande. A first-order conditional logic for prototypical properties. *Artificial Intelligence*, 33(1):105–130, 1987.
9. J. P. Delgrande and C. Groeneboer. A general approach for determining the validity of commonsense assertions using conditional logics. *International Journal of Intelligent Systems*, 5(5):505–520, 1990.
10. N. Friedman and J. Y. Halpern. Plausibility measures and default reasoning. *Journal of the ACM*, 48(4):648–685, 2001.
11. D. M. Gabbay. Labelled deductive systems (vol i). *Oxford Logic Guides, Oxford University Press*, 1996.
12. D. M. Gabbay, L. Giordano, A. Martelli, N. Olivetti, and M. L. Sapino. Conditional reasoning in logic programming. *Journal of Logic Programming*, 44(1-3):37–74, 2000.
13. Dov M. Gabbay and Nicola Olivetti. *Goal-directed Proof Theory*. Kluwer Academic Publishers, 2000.
14. I. P. Gent. A sequent or tableaux-style system for lewis's counterfactual logic vc. *Notre Dame Journal of Formal Logic*, 33(3):369–382, 1992.
15. L. Giordano, V. Gliozzi, and N. Olivetti. Iterated belief revision and conditional logic. *Studia Logica*, 70(1):23–47, 2002.
16. L. Giordano, V. Gliozzi, and N. Olivetti. Weak agm postulates and strong ramsey test: a logical formalization. *Artificial Intelligence*, 168(1-2):1–37, 2005.
17. L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux for KLM Preferential and Cumulative Logics. In Geoff Sutcliffe and Andrei Voronkov, editors, *Prooceedings of LPAR 2005 (12th Conference on Logic for Programming, Artificial Intelligence, and Reasoning)*, volume 3835 of *LNAI*, pages 666–681, Montego Bay, Jamaica, December 2005. Springer-Verlag.
18. L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Analytic Tableaux Calculi for KLM Rational Logic R. In *Proceedings of JELIA 2006*, volume 4160 of *LNAI*, pages 190–202. Springer-Verlag, September 2006.
19. L. Giordano, V. Gliozzi, N. Olivetti, and C. Schwind. Tableau calculi for preference-based conditional logics. *In Proceedings of TABLEAUX 2003 (Automated Reasoning with Analytic Tableaux and Related Methods), volume 2796 of LNAI, Springer*, pages 81–101, 2003.
20. L. Giordano, V. Gliozzi, N. Olivetti, and C.B. Schwind. Extensions of tableau calculi for preference-based conditional logics. In *Proceedings of M4M-4*, pages 220–234. Informatik-Bericht 194, December 2005.
21. L. Giordano, V. Gliozzi, and G. L. Pozzato. KLMLean 2.0: a Theorem Prover for KLM Logics of Nonmonotonic Reasoning. In *Proceedings of TABLEAUX 2007*, volume to appear of *LNAI*. Springer-Verlag, 2007.
22. L. Giordano and C. Schwind. Conditional logic of actions and causation. *Artificial Intelligence*, 157(1-2):239–279, 2004.
23. G. Grahne. Updates and counterfactuals. *Journal of Logic and Computation*, 8(1):87–117, 1998.
24. S. Kraus, D. Lehmann, and M. Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.
25. P. Lamarre. A tableaux prover for conditional logics. *In Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference*, pages 572–580, 1993.

26. D. Lewis. Counterfactuals. *Basil Blackwell Ltd*, 1973.
27. D. Miller, G. Nadathur, F. Pfenning, and A. Scedrov. Uniform proofs as a foundation for logic programming. *In Annal of Pure and Applied Logic*, 51(1-2):125–157, 1991.
28. D. Nute. Topics in conditional logic. *Reidel, Dordrecht*, 1980.
29. N. Obeid. Model-based diagnosis and conditional logic. *Applied Intelligence*, 14:213–230, 2001.
30. N. Olivetti and G. L. Pozzato. CondLean: A Theorem Prover for Conditional Logics. In *Proceedings of TABLEAUX 2003*, volume 2796 of *LNAI*, pages 264–270. Springer, September 2003.
31. N. Olivetti and G. L. Pozzato. CondLean 3.0: Improving Condlean for Stronger Conditional Logics. In *Proceedings of TABLEAUX 2005*, volume 3702 of *LNAI*, pages 328–332. Springer-Verlag, September 2005.
32. N. Olivetti and G. L. Pozzato. KLMLean 1.0: a Theorem Prover for Logics of Default Reasoning. In *Proceedings of M4M-4*, pages 235–245. Informatik-Bericht 194, December 2005.
33. N. Olivetti, G. L. Pozzato, and C. B. Schwind. A Sequent Calculus and a Theorem Prover for Standard Conditional Logics. *ACM Transactions on Computational Logics (TOCL)*, to appear, 2007.
34. G. L. Pozzato. *Proof Methods for Conditional and Preferential Logics*. PhD thesis, 2007.
35. C. B. Schwind. Causality in action theories. *Electronic Transactions on Artificial Intelligence (ETAI)*, 3(A):27–50, 1999.
36. R. Stalnaker. A theory of conditionals. *In N. Rescher (ed.), Studies in Logical Theory, American Philosophical Quarterly, Monograph Series no.2, Blackwell, Oxford*, pages 98–112, 1968.
37. L. Viganò. Labelled non-classical logics. *Kluwer Academic Publishers, Dordrecht*, 2000.