KLMLean 1.0: a Theorem Prover for Logics of Default Reasoning

Nicola Olivetti[⋆] and Gian Luca Pozzato[⋆⋆]

Abstract. In this paper we present KLMLean 1.0, a theorem prover for some logics of default reasoning, namely Preferential logic **P** and Loop-Cumulative logic **CL** introduced by Kraus, Lehmann, and Magidor. KLMLean 1.0 implements some tableaux calculi for these logics recently introduced. It is implemented in SICStus Prolog and also comprises a graphical user interface written in Java. KLMLean 1.0 is available for free download at `http://www.di.unito.it/~pozzato/klmlean1/`

1 Introduction

In the early 90' [6, 7] Kraus, Lehmann and Magidor (from now on KLM) proposed a formalization of non-monotonic reasoning that was early recognized as a landmark. Their work stemmed from two sources: the theory of nonmonotonic consequence relations initiated by Gabbay [4] and the preferential semantics proposed by Shoham [11] as a generalization of Circumscription. Their works lead to a classification of nonmonotonic consequence relations, determining a hierarchy of stronger and stronger systems.

According to the KLM framework, defeasible knowledge is represented by a (finite) set of nonmonotonic conditionals or assertions of the form $A \mid\!\sim B$ whose reading is *normally (or typically) the A's are B's.* The operator "$\mid\!\sim$" is nonmonotonic, in the sense that $A \mid\!\sim B$ does not imply $A \wedge C \mid\!\sim B$. For instance, a knowledge base K may contain the following set of conditionals: $adult \mid\!\sim work, retired \mid\!\sim adult, retired \mid\!\sim \neg work$, whose meaning is that adults typically work, retired people are typically adults but they typically do not work. Observe that if $\mid\!\sim$ were interpreted as classical (or intuitionistic) implication, we simply would get $retired \mid\!\sim \bot$, i.e. typically there are not retired people, thereby obtaining a trivial knowledge base. One can derive new conditional assertions from the knowledge base by means of a set of inference rules.

In KLM framework, the set of adopted inference rules defines some fundamental types of inference systems, namely, from the weakest to the strongest: Cumulative (**C**) , Loop-Cumulative (**CL**), Preferential (**P**) and Rational logic (**R**). All these systems allow one to infer new assertions from K without incurring in the trivialising conclusions of classical logic: regarding our example, in

[⋆] LSIS - UMR CNRS 6168 Université Paul Cézanne (Aix-Marseille 3) Avenue Escadrille Normandie-Niemen 13397 Marseille Cedex 20 - France - E-mail: `nicola.olivetti@univ.u-3mrs.fr`

[⋆⋆] Dipartimento di Informatica - Universitá degli Studi di Torino - corso Svizzera 185 - 10149 Torino - Italia - e-mail: `pozzato@di.unito.it` none of them, one can infer $retired \mid\sim work$. In cumulative logics (both **C** and **CL**) one can infer $adult \land retired \mid\sim \neg work$ (giving preference to more specific information), in Preferential logic **P** one can also infer that $adult \mid\sim \neg retired$ (i.e. typical adults are not retired). In the rational case **R**, if one further knows that $adult \not\mid\sim \neg married$ (i.e. it is not the case the adults are typically unmarried), one can also infer that $adult \land married \mid\sim work$.

From a semantic point of view, to the each logic (**C**, **CL**, **P**, **R**) corresponds one kind of models, namely, possible-world structures equipped with a preference relation among worlds or states. More precisely, for **P** we have models with a preference relation (an irreflexive and transitive relation) on worlds. For the stronger **R** the preference relation is further assumed to be *modular*. For the weaker logic **CL**, the preference relation is defined on *states*, where a state can be identified, intuitively, with a set of worlds. In the weakest case of **C**, the preference relation is on states, as for **CL**, but it is no longer assumed to be transitive. In all cases, the meaning of a conditional assertion $A \mid\sim B$ is that B holds in the *most preferred* worlds/states where A holds.

Even if KLM was born as an inferential approach to nonmonotonic reasoning, very few proof systems have been proposed for these logics. In [5] analytic tableaux $\mathcal{T}S^{\mathbf{T}}$ for logics **P** and **CL** have been introduced, where S stands for $\{\mathbf{P}, \mathbf{CL}\}$. The calculi $\mathcal{T}S^{\mathbf{T}}$ are based on a novel interpretation of **P** into modal logic G. The idea is simply to interpret the preference relation as an accessibility relation: a conditional $A \mid\sim B$ holds in a model if B is true in all A-worlds w, i.e. worlds w where A holds, that are minimal. An A-world is minimal if all smaller worlds are not A-worlds. The relation with modal logic G is motivated by the fact that we assume, following KLM, the so-called *smoothness condition*, which is related to the well-known *limit assumption*. This condition ensures indeed that A-minimal worlds exist, by preventing an infinitely descending chain of worlds. This condition is therefore ensured by the finite-chain condition on the accessibility relation (as in modal logic G).

In this work we describe an implementation of $\mathcal{T}S^{\mathbf{T}}$ calculi in SICStus Prolog. The program, called KLMLean 1.0, is implemented following the methodology of leanTAP, inspired to [1] and [2]. As far as we know, KLMLean 1.0 is the first theorem prover for these logics. KLMLean 1.0 also comprises a graphical user interface written in Java.

The plan of this paper is as follows: in section 2 we briefly recall KLM logics **P** and **CL** and the tableaux calculi $\mathcal{T}S^{\mathbf{T}}$. In section 3 we describe KLMLean 1.0. In section 4 we discuss some related works and suggest some future developments.

2 KLM Logics and the Tableaux Calculi $\mathcal{T}S^{\mathbf{T}}$

2.1 KLM Preferential Logic P

We consider a propositional language $\mathcal{L}$ over a set ATM of propositional variables. Formulas of $\mathcal{L}$ are built from propositional variables by means of the boolean operators $\rightarrow$, $\land$, $\lor$, $\neg$ and the conditional operator $\mid\sim$. The language $\mathcal{L}$

consists just of conditional assertions $A \mid\sim B$, where A and B are propositional formulas, and of boolean combinations of them. More precisely, the formulas of $\mathcal{L}$ are defined as follows: if A is a propositional formula, $A \in \mathcal{L}$; if A and B are propositional formulas, $A \mid\sim B \in \mathcal{L}$; if F is a boolean combination of formulas of $\mathcal{L}$, $F \in \mathcal{L}$.

The semantics of **P** is defined by considering possible world structures with a preference relation (a strict partial order) $w < w^{'}$ whose meaning is that w is preferred to $w^{'}$. We have that $A \mid\sim B$ holds in a model $\mathcal{M}$ if B holds in all *minimal worlds* (with respect to the relation $<$) where A holds. This definition makes sense provided minimal worlds for A exist whenever there are A-worlds. This is ensured by the *smoothness condition* in the next definition.

Definition 1 (Semantics of P [6]). *A preferential model is a triple $\mathcal{M} = \langle \mathcal{W}, <, V \rangle$ where: $\mathcal{W}$ is a non-empty set of items called worlds; $<$ is an irreflexive and transitive relation on $\mathcal{W}$; V is a function $V : \mathcal{W} \longmapsto pow(ATM)$, which assigns to every world w the set of atoms holding in that world. We define the truth conditions for a formula F as follows:*

- *If F is a boolean combination of formulas, $\mathcal{M}, w \models F$ is defined in the obvious way;*
- *Let A be a propositional formula; we define $Min_<(A) = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A \text{ and } \forall w^{'}, w^{'} < w \quad \mathcal{M}, w^{'} \not\models A\}$;*
- *$\mathcal{M}, w \models A \mid\sim B$ if for all $w^{'} \in Min_<(A)$, then $\mathcal{M}, w^{'} \models B$.*

The relation $<$ satisfies the following condition, called smoothness*: if $\mathcal{M}, w \models A$ then $w \in Min_<(A)$ or $\exists w^{'} \in Min_<(A)$ such that $w^{'} < w$.*
We say that a formula F is valid *in a model $\mathcal{M}$, denoted with $\mathcal{M} \models F$, if $\mathcal{M}, w \models F$ for every $w \in \mathcal{W}$. A formula is* valid *if it is valid in every model $\mathcal{M}$.*

Notice that the truth conditions for conditional formulas are given with respect to single possible worlds for uniformity sake. Since the truth value of a conditional only depends on global properties of $\mathcal{M}$, we have that: $\mathcal{M}, w \models A \mid\sim B$ iff $\mathcal{M} \models A \mid\sim B$.

Now we introduce the language $\mathcal{L}_P$ of the calculus $\mathcal{T}\mathbf{P}^{\mathbf{T}}$. $\mathcal{L}_P$ extends $\mathcal{L}$ by formulas of the form $\Box A$, where A is a propositional formula, whose intuitive meaning is as follows: $\Box A$ holds in a world w if A holds in all the worlds $w^{'}$ such that $w^{'} < w$:

Definition 2 (Truth condition of modality $\Box$, [5]). *We define the truth condition of a boxed formula as follows:*

$$\mathcal{M}, w \models \Box A \textit{ if for every } w^{'} \in \mathcal{W} \textit{ if } w^{'} < w \textit{ then } \mathcal{M}, w^{'} \models A$$

It is easy to see that $\Box$ has the properties of the modal system G: the accessibility relation (defined as xRy iff $y < x$) is transitive and does not have infinite ascending chains. By Definitions 1 and 2, it follows that for any formula A, $w \in Min_<(A)$ iff $\mathcal{M}, w \models A \wedge \Box\neg A$.

2.2 KLM Loop Cumulative Logic CL

The next KLM logic we consider is **CL**, weaker than **P**.

Definition 3 (Loop-cumulative models, [6]). *A loop-cumulative model is a tuple $\mathcal{M} = \langle S, l, <, V \rangle$. S is a set, whose elements are called states. Given a set $\mathcal{U}$ of possible worlds, $l : S \mapsto 2^{\mathcal{U}}$ is a function that labels every state with a nonempty set of worlds. $<$ is an irreflexive and transitive relation on S. V is a valuation function $V : \mathcal{U} \longmapsto pow(ATM)$, which assigns to every world w the atoms holding in that world. For $s \in S$ and A propositional formula, we let $s \models\equiv A$ if $\forall w \in l(s)$, $w \models A$. Let $Min_<(A)$ be the set of minimal states s such that $s \models\equiv A$. We define $\mathcal{M}, s \models\equiv A \mid\!\sim B$ if $\forall s' \in Min_<(A)$, $s' \models\equiv B$. We assume that $<$ satisfies the smoothness condition.*

In [5] it is shown that one can map loop-cumulative models into preferential models extended with an additional accessibility relation R. These models are called CL-Preferential structures and are defined below. The idea is to represent states as sets of possible worlds related by R: thus a world plays also a role of a state. By this correspondence we have that a formula is satisfied in a state s just in case it is satisfied in all possible worlds w' accessible from its corresponding w via R. The syntactic counterpart of the extra accessibility relation R is a modality L. Given a loop-cumulative model $\mathcal{M}$ and the corresponding CL-structure $\mathcal{M}'$, $\mathcal{M}, s \models\equiv A$ iff for its corresponding w, $\mathcal{M}', w \models LA$. This mapping allows us to use a variant of the tableaux calculus for **P** to deal with system **CL**. As for **P**, the tableaux calculus for **CL** will use boxed formulas. Thus, the formulas that appear in the tableaux for **CL** belong to the language $\mathcal{L}_L$ obtained from $\mathcal{L}$ as follows: (i) if A is a propositional formula, then $A \in \mathcal{L}_L$; $LA \in \mathcal{L}_L$; $\Box\neg LA \in \mathcal{L}_L$; ($ii$) if A, B are propositional formulas, then $A \mid\!\sim B \in \mathcal{L}_L$; ($iii$) if F is a boolean combination of formulas of $\mathcal{L}_L$, then $F \in \mathcal{L}_L$.

Definition 4 (CL-preferential structures, [5]). *A model has the form $\mathcal{M} = \langle \mathcal{W}, R, <, V \rangle$ where: $\mathcal{W}$, $<$, and V are defined as in Definition 1, and R is a* serial *accessibility relation (i.e. $\forall w \exists w'\ Rww'$). We add to the truth conditions for preferential models in Definition 1 the following clause:*

$$\mathcal{M}, w \models LA \text{ if for all } w' \text{ if } wRw' \text{ then } \mathcal{M}, w' \models A$$

The truth condition for conditional formulas is changed as follows: $\mathcal{M}, w \models A \mid\!\sim B$ if for all $w' \in Min_<(LA)$ then $\mathcal{M}, w' \models LB$.

Proposition 1 ([5]). *A set of conditional formulas $\{(\neg)A_1 \mid\!\sim B_1, \ldots, (\neg)A_n \mid\!\sim B_n\}$ is satisfiable in a loop-cumulative model $\langle S, l, <, V \rangle$ iff it is satisfiable in a CL-preferential model $\langle W, R, <, V \rangle$.*

2.3 The Tableaux Calculi $\mathcal{T}S^{\mathbf{T}}$

In Figures 1 and 2 we present the tableaux calculi $\mathcal{T}\mathbf{P}^{\mathbf{T}}$ and $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ introduced in [5]. In order to obtain terminating calculi, the $(\mid\!\sim^+)$ rule must be applied under some restrictions that for clarity are not displayed in the figures below, but are explained later in this section.

The rules of the calculi manipulate sets of formulas Γ. We write Γ, F to denote $\Gamma \cup \{F\}$. Moreover, we consider the following sets:

- $\Gamma^{\Box} = \{\Box A \mid \Box A \in \Gamma\}$
- $\Gamma^{\Box^{\downarrow}} = \{A \mid \Box A \in \Gamma\}$
- $\Gamma^{\mid\!\sim^{+}} = \{A \mid\!\sim B \mid A \mid\!\sim B \in \Gamma\}$
- $\Gamma^{\mid\!\sim^{-}} = \{\neg(A \mid\!\sim B) \mid \neg(A \mid\!\sim B) \in \Gamma\}$
- $\Gamma^{\mid\!\sim^{\pm}} = \Gamma^{\mid\!\sim^{+}} \cup \Gamma^{\mid\!\sim^{-}}$
- $\Gamma^{L^{\downarrow}} = \{A \mid LA \in \Gamma\}$

$$(\mathbf{AX})\ \Gamma, F, \neg F \qquad\qquad (\neg)\ \frac{\Gamma, \neg\neg F}{\Gamma, F}$$

$$(\mid\!\sim^{+})\ \frac{\Gamma, A \mid\!\sim B}{\Gamma, \neg\Box\neg A, A \mid\!\sim B \qquad \Gamma, A \rightarrow B, A \mid\!\sim B}$$

$$(\mid\!\sim^{-})\ \frac{\Gamma, \neg(A \mid\!\sim B)}{A, \Box\neg A, \neg B, \Gamma^{\mid\!\sim^{\pm}}} \qquad\qquad (\Box^{-})\ \frac{\Gamma, \neg\Box\neg A}{\Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\!\sim^{\pm}}, A, \Box\neg A}$$

$$(\wedge^{+})\ \frac{\Gamma, F \wedge G}{\Gamma, F, G} \qquad\qquad (\wedge^{-})\ \frac{\Gamma, \neg(F \wedge G)}{\Gamma, \neg F \qquad \Gamma, \neg G}$$

Fig. 1. Tableaux system $\mathcal{T}\mathbf{P^T}$. To save space, we omit rules for the other boolean connectives.

Theorem 1 (Soundness and completeness of $\mathcal{T}\mathbf{P^T}$ and $\mathcal{T}\mathbf{CL^T}$, [5]). *The systems $\mathcal{T}\mathbf{P^T}$ and $\mathcal{T}\mathbf{CL^T}$ are sound and complete with respect to the semantics, i.e. there is a closed tableaux for a set Γ iff Γ is unsatisfiable.*

In order to obtain a decision procedure we have to control the application of the $(\mid\!\sim^{+})$ rule, since it copies its principal formula in its conclusions. To this regard we have the following result:

Theorem 2 ([5]). *$\mathcal{T}\mathbf{P^T}$ and $\mathcal{T}\mathbf{CL^T}$ are sound and complete even if $(\mid\!\sim^{+})$ is applied at most once on the same conditional $A \mid\!\sim B$ in the same world. More*

$$(\mid\!\sim^{+})\ \frac{\Gamma, A \mid\!\sim B}{\Gamma, \neg\Box\neg LA, A \mid\!\sim B \qquad \Gamma, LA \rightarrow LB, A \mid\!\sim B} \qquad (\mid\!\sim^{-})\ \frac{\Gamma, \neg(A \mid\!\sim B)}{LA, \Box\neg LA, \neg LB, \Gamma^{\mid\sim^{\pm}}}$$

$$(\mathbf{L}^{-})\ \frac{\Gamma, \neg LA}{\Gamma^{L^{\downarrow}}, \neg A} \text{ where either } \{\neg LA\} \neq \emptyset \text{ or } \Gamma^{L^{\downarrow}} \neq \emptyset \quad (\Box^{-})\ \frac{\Gamma, \neg\Box\neg LA}{\Gamma^{\Box}, \Gamma^{\Box^{\downarrow}}, \Gamma^{\mid\sim^{\pm}}, LA, \Box\neg LA}$$

Fig. 2. Tableaux system $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$. If $\neg LA$ is not in the premise of (L^{-}) (i.e. $\{\neg LA\} = \emptyset$) the rule allows to step from Γ to $\Gamma^{L^{\downarrow}}$. The boolean rules are omitted.

precisely, when $(\mid\!\sim^{+})$ *is applied to* $A \mid\!\sim B$*,* $A \mid\!\sim B$ *is marked as* used*; the applications of* $(\mid\!\sim^{+})$ *are restricted to unused conditionals only; when* $(\Box^{-})$ *or* $(\mid\!\sim^{-})$ *is applied, all conditionals become unused*[1]*.*

The generation of infinite branches due to the interplay between $(\mid\!\sim^{+})$ and $(\Box^{-})$ cannot occur, because of the properties of $\Box$ in modal logic G (see [5] for details). These results give a constructive proof of decidability of Preferential and Loop-Cumulative logics.

3 Design of KLMLean 1.0

In this section we present an implementation of the tableaux calculi $\mathcal{T}S^{\mathbf{T}}$ called **KLMLean 1.0**; it is a SICStus Prolog program inspired by leanTAP ([1], [2]) and follows the line of CondLean, a theorem prover for other conditional logics introduced in [9] and [10]. The idea of *lean deduction* [1] is ”to achieve maximal efficiency from minimal means”, that is to say to write short programs and exploit the power of Prolog's engine as much as possible. In [1] authors remark that another important argument for lean methodology is safety: to verify soundness and completeness of a theorem prover consisting of a couple of lines is easier than doing the same for thousands of lines of C code. Moreover, in [1] it is observed that programs following the lean methodology offer high performances on simple to moderately complex problems, and that they offer a high degree of adaptability to applications.

It is worth noticing that KLMLean 3.1 is only *inspired* by leanTAP, however it does not follow strictly the style of leanTAP. For instance, KLMLean 3.1 makes use of auxiliary Prolog's predicates like `member` and `select`, whereas leanTAP only relies on Prolog's clause indexing scheme and backtracking.

[1] The premise and the conclusion of $(\Box^{-})$ (resp. $(\mid\!\sim^{-})$) represent two different worlds. When $(\Box^{-})$ (resp. $(\mid\!\sim^{-})$) is applied, then all conditional formulas used in the world represented by the premise become unused in the world represented by the conclusion.

The program KLMLean 1.0 comprises a set of clauses, each one of them represents a tableaux rule or axiom. The proof search is provided for free by the mere depth-first search mechanism of Prolog, without any additional ad hoc mechanism. We represent each node of a proof tree by a *list* of formulas. The tableaux calculi are implemented by the predicate **prove(Gamma,Sigma,Tree).** which succeeds if and only if the set of formulas Γ, represented by the Prolog list `Gamma`, is unsatisfiable. `Sigma` is the list representing the set Σ of *used conditionals*, and it is needed in order to control the application of the $(\mid\!\sim^{+})$ rule only to unused conditional, as stated by Theorem 2 above. If the predicate succeeds, `Tree` contains the closed proof tree found by the theorem prover.

For instance, to prove that $A \mid\!\sim B \land C, \neg(A \mid\!\sim C)$ is unsatisfiable in $\mathbf{P}$, one queries KLMLean 1.0 with the following goal:

```
prove([a => (b and c), neg (a => c)],[  ],Res)).
```

The string `=>` is used to represent the conditional operator $\mid\!\sim$, `and` is used to denote $\land$, and so on. Each clause of `prove` implements one axiom or rule of $\mathcal{TS}^{\mathbf{T}}$; for example, we present the clause implementing the axiom:

```
prove(Gamma,_,tree(ax,F)):-
   member(neg F,Gamma),
   member(F, Gamma),!.
```

If the formula F and its negation $\neg F$ both belong to the set of formulas Γ, then the predicate `prove` succeeds by the above clause. As mentioned above, the third argument of the predicate `prove` contains the proof tree found by the theorem prover; in this case, it corresponds to a leaf represented by the functor `tree(ax,F)`, to keep trace that the tree is closed by the presence of both formulas `F` and `neg F`.

As another example, the clause implementing $(\mid\!\sim^{+})$ is as follows:

```
prove(Gamma,Sigma,tree(entP,A,B,ST1,ST2)):-
   select(A => B,Gamma,NewGamma),!,
   prove([neg box neg A|NewGamma],[A => B|Sigma],ST1),!,
   prove([A -> B|NewGamma],[A => B|Sigma],ST2).
```

The predicate `select(X,List,NewList)` succeeds if and only if an item `X` belongs to `List`, returning the list `NewList` in which `X` has been deleted from `List`. Therefore, if a formula $A \mid\!\sim B$ belongs to Γ, then KLMLean 1.0 applies the $(\mid\!\sim^{+})$ rule, by invoking recursively the predicate `prove` on its two conclusions. In the first one, the formula $\neg\Box\neg A$ is introduced; in the other one, $A \rightarrow B$ is added to the set of formulas to analyze. In both the recursive calls to the predicate `prove` the principal formula $A \mid\!\sim B$ is copied into the auxiliary list `Sigma`, in order to avoid further applications of $(\mid\!\sim^{+})$ in the same world.
The functor `tree(entP,A,B,ST1,ST2)` represents the proof tree obtained by applying the above clause. More in detail, its terms represent that the $(\mid\!\sim^{+})$ rule

has been applied (`entP`) to a conditional formula `A => B`; `ST1` and `ST2` are the closed subtrees obtained by invoking `prove` on the two conclusions of the rule.

To search a closed tree of a set Γ of formulas of $\mathcal{L}$, KLMLean 1.0 proceeds as follows. First of all, if Γ is an axiom, then the goal will succeed immediately by using the clause for the axiom presented above. If it is not, then the first applicable rule will be chosen, e.g. if `Gamma` contains a formula `[A and B]`, then the clause for the $(\wedge^+)$ rule will be used, invoking `prove` on the unique conclusion of $(\wedge^+)$. KLMLean 1.0 proceeds in a similar way for the other rules. The ordering of the clauses is such that the application of the branching rules and of the conditional and box rules is postponed as much as possible. In the above example, it is clear how KLMLean implement the controlled application of $(\mid\!\sim^+)$ on a conditional formula $A \mid\!\sim B$ (Theorem 2): $A \mid\!\sim B$ passes from Γ to the set Σ of used conditionals, therefore the above clause will not further apply to it (it works only on conditionals belonging to Γ).

KLMLean has also a graphical user interface (GUI) implemented in Java. The GUI interacts with the SICStus Prolog implementation by means of the package `se.sics.jasper`. Thanks to the GUI, one does not need to know how to call the predicate `prove`: one just introduces

- formulas of a knowledge base K (or the set of formulas to prove to be unsatisfiable);
- a formula F, in order to prove if one can infer F from K, corresponding to verify if $K \cup \{\neg F\}$ is unsatisfiable

in a text box and searches a derivation by clicking a button. Moreover, one can choose the intended system of conditional logic, namely **P** or **CL**. When the submitted set $K \cup \{\neg F\}$ of formulas is unsatisfiable, KLMLean 1.0 offers these options:

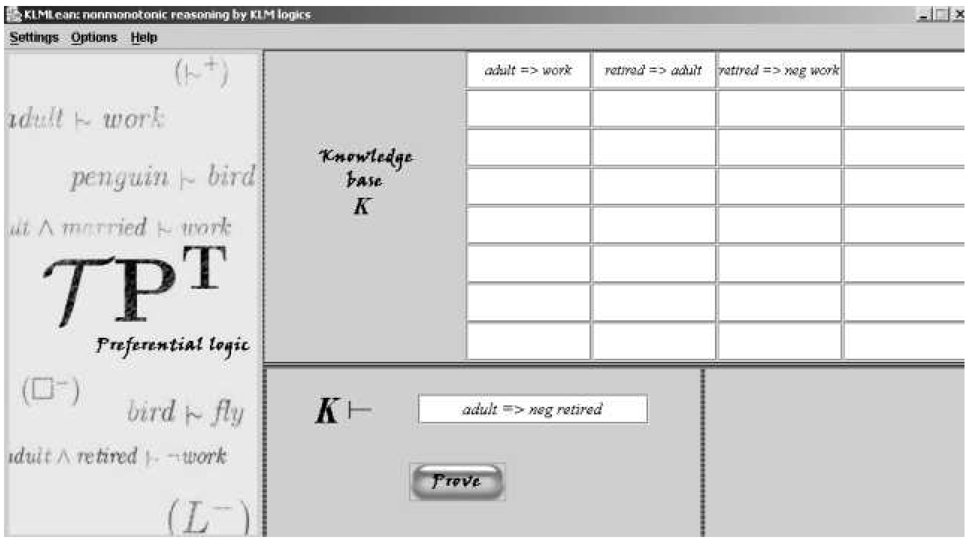
- display a proof tree of the set in a special window;
- build a latex file containing the same proof tree: compiling this file with Latex, one can obtain the closed tree in a pdf file, or ps, or dvi, and then print it.

Some pictures of KLMLean 1.0 are presented in Figures 3, 4, and 5.

4 Conclusions, Comparison with Other Works, and Future Work

In this work we have presented KLMLean 1.0, a theorem prover for KLM Preferential and Loop-Cumulative logics. To the best of our knowledge, this is the first theorem prover for logics of the KLM framework. KLMLean 1.0 is a SICStus Prolog implementation of the analytic tableaux calculi introduced in [5], and follows the leanTAP methodology of [1] and [2].

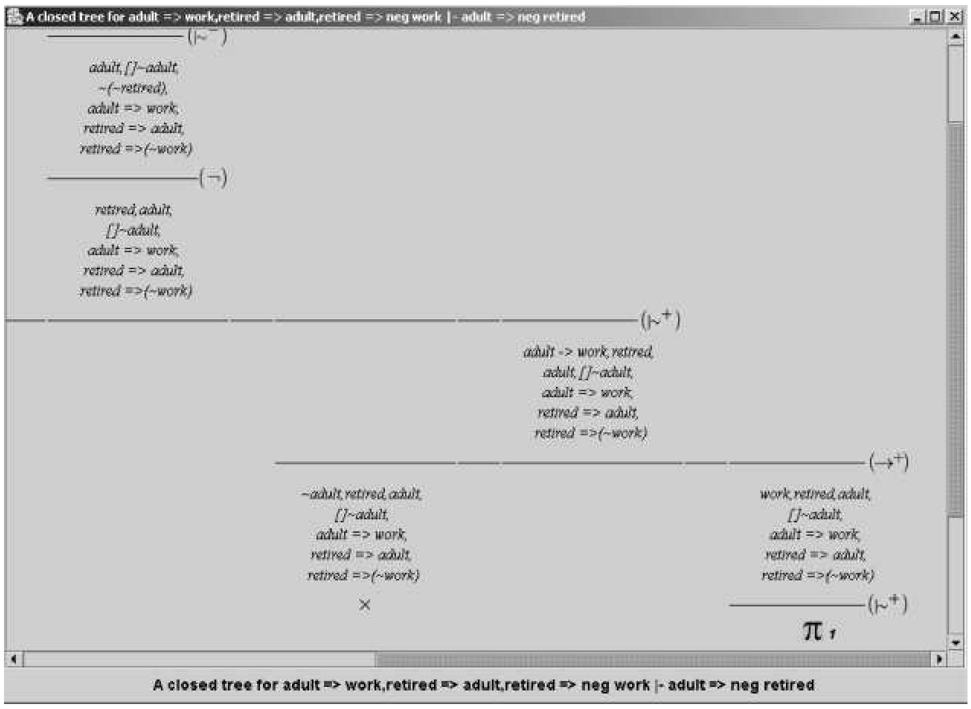
We briefly remark on some related works. We just mention the program CondLean, a theorem prover for some standard conditional logics ([9], [10]).


KLMLean: nonmonotonic reasoning by KLM logics
Settings
Options
Help
$(\mid\!\sim^+)$
adult $\mid\!\sim$ work
penguin $\mid\!\sim$ bird
adult $\wedge$ married $\mid\!\sim$ work
TP^T
Preferential logic
$(\Box^-)$
bird $\mid\!\sim$ fly
adult $\wedge$ retired $\mid\!\sim$ $\neg$work
(L^-)
Knowledge base K
Prove

adult => work	retired => adult	retired => neg work

$K \vdash$ adult => neg retired

Fig. 3. The main window of KLMLean 1.0.



A closed tree for adult => work,retired => adult,retired => neg work |- adult => neg retired

```
                                    (|~⁻)
adult, []~adult,
~(~retired),
adult => work,
retired => adult,
retired =>(~work)
                                    (¬)
retired, adult,
[]~adult,
adult => work,
retired => adult,
retired =>(~work)
                                    (|~⁺)
adult -> work, retired,
adult, []~adult,
adult => work,
retired => adult,
retired =>(~work)
                                    (→⁺)
~adult, retired, adult,          work, retired, adult,
[]~adult,                        []~adult,
adult => work,                   adult => work,
retired => adult,                retired => adult,
retired =>(~work)                retired =>(~work)
×                                                    (|~⁺)
                                 π₁
```

A closed tree for adult => work,retired => adult,retired => neg work |- adult => neg retired

Fig. 4. A proof tree in a special window.

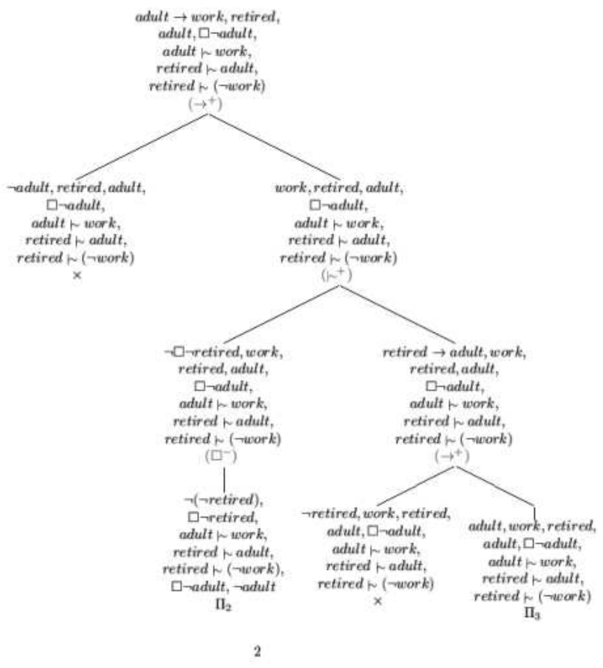

adult → work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
(→+)
¬adult, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
×
work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
(|~+)
¬□¬retired, work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
(□−)
retired → adult, work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
(→+)
¬(¬retired), □¬retired, adult |~ work, retired |~ adult, retired |~ (¬work), □¬adult, ¬adult
Π2
¬retired, work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
×
adult, work, retired, adult, □¬adult, adult |~ work, retired |~ adult, retired |~ (¬work)
Π3
2

Fig. 5. The proof tree printed in a .dvi file.

CondLean is a SICStus Prolog implementation of some labelled sequent calculi for conditional logics CK, ID, MP, CS, CEM, and some combinations of them. It follows the leanTAP methodology, too. The main difference between CondLean and KLMLean is that they offer a deductive mechanism for different conditional logics: CondLean is a theorem prover for conditional logics whose semantics is based on the so called *selection function* [8], whereas KLMLean is a theorem prover for KLM logics.

In future research, we intend to extend KLMLean to the other KLM conditional logics, namely the weaker Cumulative logic **C** and the stronger Rational logic **R**. Moreover, we intend to make a detailed study on the performances of our theorem prover, and experiment some standard refinements and heuristics to increase its efficiency. Furthermore, we are improving KLMLean 1.0 in such a way that, given a satisfiable set of formulas in input, it shows a model of that satisfiable set.
A further extension is to develop a theorem prover for the first order version of these logics, which have recently received a renewed attention thanks to the results given in [3].

References

1. Beckert, B. and J. Posegga, *leantap: Lean tableau-based deduction*, Journal of Automated Reasoning, 15(3) (1995), pp. 339–358.
2. Fitting, M., *leantap revisited*, Journal of Logic and Computation, 8(1) (1998), pp. 33–47.
3. Friedman, N., J. Y. Halpern and D. Koller, *First-order conditional logic for default reasoning revisited*, ACM TOCL, ACM Press **1** (2000), pp. 175–207.
4. Gabbay, D., *Theoretical foundations for non-monotonic reasoning in expert systems*, Logics and models of concurrent systems, Springer (1985), pp. 439–457.
5. Giordano, L., V. Gliozzi, N. Olivetti and G. L. Pozzato, *Analytic tableaux for klm preferential and cumulative logics*, In Proc. of 12th Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2005), volume 3835 of LNAI, Springer-Verlag (2005), pp. 666–681.
6. Kraus, S., D. Lehmann and M. Magidor, *Nonmonotonic reasoning, preferential models and cumulative logics*, Artificial Intelligence, 44(1-2) (1990), pp. 167–207.
7. Lehmann, D. and M. Magidor, *What does a conditional knowledge base entail?*, Artificial Intelligence, Elsevier Science Publishers Ltd. **55** (1992), pp. 1–60.
8. Nute, D., *Topics in conditional logic*, Reidel (1980).
9. Olivetti, N. and G. L. Pozzato, *Condlean: A theorem prover for conditional logics*, In Proc. of TABLEAUX 2003, vol. 2796 of LNAI, Springer (2003), pp. 264–270.
10. Olivetti, N. and G. L. Pozzato, *Condlean 3.0: Improving condlean for stronger conditional logics*, In Proc. of TABLEAUX 2005, vol. 3702 of LNAI, Springer (2005), pp. 328–332.
11. Shoham, Y., *A semantical approach to nonmonotonic logics*, In Proceedings of Logics in Computer Science (1987), pp. 275–279.