KLMLean 2.0: a Theorem Prover for KLM Logics of Nonmonotonic Reasoning

Laura Giordano*, Valentina Gliozzi°, and Gian Luca Pozzato°

* Dip. di Informatica - Univ. del Piemonte Orientale "A. Avogadro" - Alessandria
° Dip. di Informatica - Università degli Studi di Torino

Abstract. We present KLMLean 2.0, a theorem prover for propositional KLM logics of nonmonotonic reasoning. KLMLean 2.0 implements some analytic tableaux calculi for these logics recently introduced. KLMLean 2.0 is inspired by the "lean" methodology, it is implemented in SICStus Prolog and it also comprises a graphical interface written in Java.

1 Introduction

In the early 90s Kraus, Lehmann and Magidor (from now on KLM) proposed a formalization of nonmonotonic reasoning that has become a point of reference [**?**]. According to KLM framework, a defeasible knowledge base is represented by a (finite) set of nonmonotonic conditionals of the form $A \mathrel{|\!\sim} B$, whose reading is *normally (or typically) the A's are B's*. The operator "$\mathrel{|\!\sim}$" is nonmonotonic, in the sense that $A \mathrel{|\!\sim} B$ does not imply $A \wedge C \mathrel{|\!\sim} B$. For instance, a knowledge base K may contain *football_lover* $\mathrel{|\!\sim}$ *bet*, *football_player* $\mathrel{|\!\sim}$ *football_lover*, *football_player* $\mathrel{|\!\sim} \neg$*bet*, whose meaning is that people loving football typically bet on the result of a match, football players typically love football but they typically do not bet (especially on matches they are going to play...). If $\mathrel{|\!\sim}$ were interpreted as classical implication, one would get *football_player* $\mathrel{|\!\sim} \bot$, i.e. typically there are not football players, thereby obtaining a trivial knowledge base. In KLM framework, the set of adopted inference rules defines some fundamental types of inference systems, namely, from the strongest to the weakest: Rational (**R**), Preferential (**P**), Loop-Cumulative (**CL**), and Cumulative (**C**) logic. In all these systems one can infer new assertions without incurring in the trivializing conclusions of classical logic: in the above example, in none of the systems, one can infer *football_player* $\mathrel{|\!\sim}$ *bet*. In cumulative logics (both **C** and **CL**) one can infer *football_lover* $\wedge$ *football_player* $\mathrel{|\!\sim} \neg$*bet*, giving preference to more specific information; in Preferential logic **P** one can also infer that *football_lover* $\mathrel{|\!\sim} \neg$*football_player*; in the rational case **R**, if one further knows that $\neg$(*football_lover* $\mathrel{|\!\sim}$ *rich*), that is to say it is not the case that football lovers are typically rich persons, one can also infer that *football_lover* $\wedge \neg$*rich* $\mathrel{|\!\sim}$ *bet*.

In [**?**,**?**,**?**] analytic tableaux procedures $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ for propositional KLM logics are introduced. In this work we describe an implementation of $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ in SICStus Prolog called **KLMLean 2.0**: it is inspired by the "lean" methodology [**?**], and it also comprises a graphical interface written in Java. For the rational logic **R**, KLMLean 2.0 implements the *labelled* calculus $\mathcal{T}\mathbf{R}^{\mathbf{T}}$ introduced in [**?**], and offers two different versions: 1. a simple version, where Prolog *constants* are used to represent $\mathcal{T}\mathbf{R}^{\mathbf{T}}$'s labels; 2. a more efficient one, where labels are represented by Prolog *variables*, inspired by the free-variable tableau presented in [**?**]. To the best of our knowledge, KLMLean 2.0 is the first theorem prover for KLM logics.

2 KLM Logics and Their Tableau Calculi

We consider a propositional language $\mathcal{L}$ defined from a set of propositional variables ATM, the boolean connectives and the conditional operator $\mathrel{|\!\sim}$. We use $A, B, C, ...$ to denote propositional formulas, whereas $F, G, ...$ are used to denote all formulas (including conditionals). 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, $A \mathrel{|\!\sim} B \in \mathcal{L}$; if F is a boolean combination of formulas of $\mathcal{L}$, $F \in \mathcal{L}$.

In general, the semantics of KLM logics is defined by considering possible world (or possible states) structures with a *preference relation* $w < w'$ among worlds (or states), whose meaning is that w is preferred to w'. $A \mathrel{|\!\sim} B$ holds in a model $\mathcal{M}$ if B holds in all *minimal worlds* (*states*) where A holds. This definition makes sense provided minimal worlds for A exist whenever there are A-worlds (A-states): this is ensured by the *smoothness condition* defined below. We recall the semantics of KLM logics [**?**] from the strongest **R** to the weakest **C**. A *rational* 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, transitive and *modular* relation on $\mathcal{W}$, and V is a valuation function $V : \mathcal{W} \longmapsto 2^{ATM}$. The truth conditions for a formula F are as follows: - if F is a boolean combination of formulas, $\mathcal{M}, w \models F$ is defined as for propositional logic; - let A be a propositional formula; we define $Min_<(A) = \{w \in \mathcal{W} \mid \mathcal{M}, w \models A \text{ and } \forall w',\ w' < w \text{ implies } \mathcal{M}, w' \not\models A\}$; - $\mathcal{M}, w \models A \mathrel{|\!\sim} B$ if for all w', if $w' \in Min_<(A)$ then $\mathcal{M}, w' \models B$. We also define the *smoothness condition* on the preference relation: if $\mathcal{M}, w \models A$, then $w \in Min_<(A)$ or $\exists w' \in Min_<(A)$ s.t. $w' < w$. Validity and satisfiability of a formula are defined as usual. A *preferential* model is defined as the rational model, with the only difference that the preference relation $<$ is no longer assumed to be modular. Models for (loop-)cumulative logics also comprise states. A *(loop-)cumulative* model is a tuple $\mathcal{M} = \langle S, \mathcal{W}, l, <, V \rangle$, where S is a set of states and $l : S \mapsto 2^{\mathcal{W}}$ is a function that labels every state with a nonempty set of worlds; $<$ is defined on S, it satisfies the smoothness condition and it is irreflexive and transitive in **CL**, whereas it is only irreflexive in **C**. A propositional formula holds in a state s if it holds in *all* the worlds $w \in l(s)$; a conditional $A \mathrel{|\!\sim} B$ holds in a model if B holds in all minimal states where A holds.

In Figure **??** we present the tableaux calculi $\mathcal{T}\mathbf{S^T}$ for KLM logics, where S stands for {**R**, **P**, **CL**, **C**}. The basic idea is simply to interpret the preference relation as an accessibility relation. The calculi for **R** and **P** implement a sort of run-time translation into (extensions of) Gödel-Löb modal logic of provability G. This is motivated by the fact that we assume the smoothness condition, which ensures that minimal A-worlds exist whenever there are A-worlds, by preventing infinitely descending chains of worlds. This condition therefore corresponds to the finite-chain condition on the accessibility relation (as in modal logic G). This approach is extended to the cases of **CL** and **C** by using a second modality L which takes care of states. The rules of the calculi manipulate sets of formulas Γ. We write Γ, F as a shorthand for $\Gamma \cup \{F\}$. Moreover, given Γ we define the following sets: $\Gamma^{\Box} = \{\Box\neg A \mid \Box\neg A \in \Gamma\}$; $\Gamma^{\Box^{\downarrow}} = \{\neg A \mid \Box\neg A \in \Gamma\}$; $\Gamma^{\mathrel{|\!\sim}^{\pm}} = \{A \mathrel{|\!\sim} B \mid A \mathrel{|\!\sim} B \in \Gamma\} \cup \{\neg(A \mathrel{|\!\sim} B) \mid \neg(A \mathrel{|\!\sim} B) \in \Gamma\}$; $\Gamma^{L^{\downarrow}} = \{A \mid LA \in \Gamma\}$.

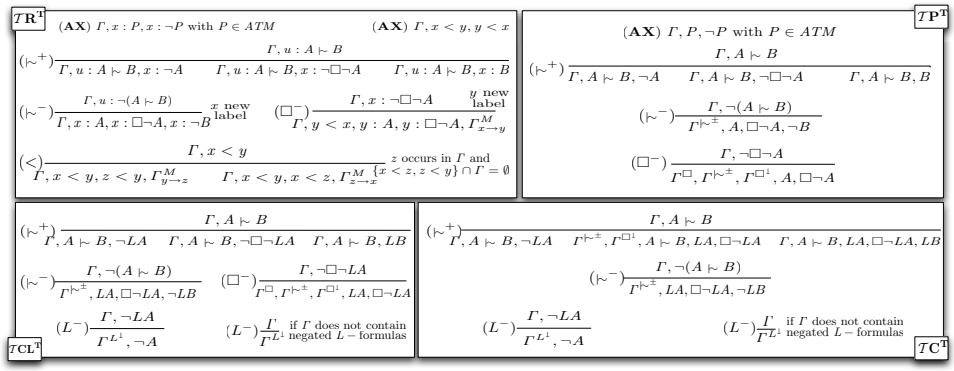

Fig. 1. Tableau systems $\mathcal{T}\mathbf{S}^{\mathbf{T}}$. To save space, we omit the standard rules for boolean connectives. For $\mathcal{T}\mathbf{CL}^{\mathbf{T}}$ and $\mathcal{T}\mathbf{C}^{\mathbf{T}}$ the axiom (**AX**) is as in $\mathcal{T}\mathbf{P}^{\mathbf{T}}$.

As mentioned, the calculus for rational logic **R** makes use of *labelled* formulas, where the labels are drawn from a denumerable set $\mathcal{A}$; there are two kinds of formulas: 1. *world formulas*, denoted by $x : F$, where $x \in \mathcal{A}$ and $F \in \mathcal{L}$; 2. *relation formulas*, denoted by $x < y$, where $x, y \in \mathcal{A}$, representing the preference relation. We define $\Gamma^{M}_{x\to y} = \{y : \neg A, y : \Box\neg A \mid x : \Box\neg A \in \Gamma\}$.

The calculi $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ are sound and complete wrt the semantics, i.e. given a set of formulas Γ of $\mathcal{L}$, it is unsatisfiable if and only if there is a closed tableau in $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ having Γ as a root [**?**,**?**,**?**]. In order to ensure termination, we have to control the application of the $(\mathrel{|\!\sim}^{+})$ rule, which can otherwise be applied without any control since it copies its principal formula $A \mathrel{|\!\sim} B$ in all its conclusions, then the conclusions have a higher complexity than the premise. In [**?**,**?**,**?**], it is shown that it is useless to apply $(\mathrel{|\!\sim}^{+})$ *more than once in the same world*, therefore the calculi $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ keep track of positive conditionals already considered in a world by moving them in an additional set Σ in the conclusions of $(\mathrel{|\!\sim}^{+})$, and restrict the application of this rule to unused conditionals only. The dynamic rules $(\mathrel{|\!\sim}^{-})$ and $(\Box^{-})$, whose conclusions represent a *different* world wrt the corresponding premise, re-introduce formulas from Σ in order to allow further applications of $(\mathrel{|\!\sim}^{+})$ in the other worlds. This machinery is standard. Concerning the labelled calculus $\mathcal{T}\mathbf{R}^{\mathbf{T}}$, the same mechanism is applied by equipping each positive conditional with the list L of worlds-labels in which $(\mathrel{|\!\sim}^{+})$ has already been applied, and restricting its application by using worlds not belonging to L. In [**?**,**?**,**?**] it is shown that no other machinery is needed to ensure termination, except for $\mathcal{T}\mathbf{C}^{\mathbf{T}}$, which needs a further standard loop-checking machinery.

3 Design of KLMLean 2.0

We describe an implementation of $\mathcal{T}\mathbf{S}^{\mathbf{T}}$ calculi in SICStus Prolog. The program, called KLMLean 2.0, is inspired by the "lean" methodology [**?**] (even if it does not fit its style in a rigorous manner): the Prolog program consists in a set of clauses, each one representing a tableau 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. KLMLean 2.0 can be downloaded at `http://www.di.unito.it/~pozzato/klmlean 2.0`.

Let us first describe the implementation of non-labelled calculi for **P**, **CL**, and **C**. We represent each node of a proof tree (i.e. set of formulas) by a Prolog list. 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 list `Gamma`, is unsatisfiable. `Sigma` is the list representing the set Σ of *used conditionals*, and it is used in order to control the application of the ($\mid\!\sim^{+}$) rule, as described in the previous section. When `prove` succeeds, `Tree` contains a representation of a closed tableau. For instance, to prove that $A \mid\!\sim B \wedge C, \neg(A \mid\!\sim C)$ is unsatisfiable in **P**, one queries KLMLean 2.0 with the following goal: `prove([a => (b and c), neg (a => c)],[ ],Tree).` The string "`=>`" is used to represent the conditional operator $\mid\!\sim$, "`and`" is used to denote $\wedge$, and so on. Each clause of `prove` implements one axiom or rule of the tableaux calculi; for example, the clauses implementing (**AX**) and ($\mid\!\sim^{-}$) are as follows:

```
prove(Gamma,_,tree(...)):-member(F,Gamma),member(neg F,Gamma),!.
prove(Gamma,Sigma,tree(...)):-select(neg (A => B),Gamma,NewGamma),conditionals(NewGamma,Cond),
   append(Cond,Sigma,DefGamma),prove([neg B|[box neg A|[A|DefGamma]]],[],...).
```

The clause for ($\mid\!\sim^{-}$) is applied when a formula $\neg(A \mid\!\sim B)$ belongs to Γ. The predicate `select` removes $\neg(A \mid\!\sim B)$ from `Gamma`, then the auxiliary predicate `conditionals` is invoked to compute the set $\Gamma^{\mid\!\sim\pm}$ on the resulting list `NewGamma`; finally, the predicate `prove` is recursively invoked on the only conclusion of the rule. Notice that, since ($\mid\!\sim^{-}$) is a dynamic rule, the conditionals belonging to Σ move to Γ in the conclusion (execution of `append`), in order to allow further applications of ($\mid\!\sim^{+}$). To search a derivation of a set of formulas Γ, KLMLean 2.0 proceeds as follows: first of all, if Γ is an instance of (**AX**), the goal will succeed immediately by using the clauses for the axioms. If it is not, then the first applicable rule will be chosen, e.g. if `Gamma` contains a formula `neg(neg F)`, then the clause for ($\neg$) rule will be used, invoking `prove` on its unique conclusion. KLMLean 2.0 proceeds in a similar way for the other rules. The ordering of the clauses is such that the boolean rules are applied before the other ones. In the case of cumulative logic **C**, KLMLean 2.0 implements a loop-checking machinery by equipping the `prove` predicate with an additional argument, called `Analyzed`, representing the list of sets of formulas already considered in the current branch. Clauses implementing $\mathcal{T}\mathbf{C}^{\mathbf{T}}$ are invoked only if the current set of formulas has not yet been considered, i.e. if it does not belong to `Analyzed`.

The theorem prover for rational logic **R** implements *labelled* tableau calculi $\mathcal{T}\mathbf{R}^{\mathbf{T}}$. It makes use of Prolog constants to represent labels: world formulas $x : A$ are represented by a Prolog list `[x,a]`, and relation formulas $x < y$ are represented by a list `[x,<,y]`. As for the other systems, each clause of the predicate `prove` implements a tableau rule or axiom. As an example, here is the clause implementing the rule ($<$), capturing the modularity of the preference relation:

```
prove(Gamma,Labels,Cond,tree(...)):-
   member([X,<,Y],Gamma),member(Z,Labels),X\=Z, Y\=Z,\+member([X,<,Z],Gamma),
   \+member([Z,<,Y],Gamma),!,gammaM(Gamma,Y,Z,ResLeft),gammaM(Gamma,Z,X,ResRight),
   append(ResLeft,Gamma,LeftConcl),append(ResRight,Gamma,RightConcl),
   prove([[Z,<,Y]|LeftConcl],Labels,Cond,...),!,prove([[X,<,Z]|RightConcl],Labels,Cond,...).
```

The predicate `gammaM(Gamma,X,Y,...)` computes the set $\Gamma^M_{x\to y}$ defined in the previous section. In this system, `Cond` is used in order to control the application of the $(\mid\!\sim^+)$ rule: it is a list whose elements have the form `[x,a => b]`, representing that $(\mid\!\sim^+)$ has been applied to $A \mid\!\sim B$ in the current branch by using the label x. In order to increase its performances, KLMLean for **R** adopts a heuristic approach (not very "lean") to implement the crucial $(\mid\!\sim^+)$ rule: the predicate `prove` *chooses the "best" positive conditional* to which apply the rule, and the *"best" label to use*. Roughly speaking, an application of $(\mid\!\sim^+)$ is considered to be better than another one if it leads to an immediate closure of a major number of conclusions. Even if $(\mid\!\sim^+)$ is invertible, choosing the right label in the application of $(\mid\!\sim^+)$ is highly critical for the performances of the thorem prover. To postpone this choice as much as possible, for the logic **R** we have defined a **free-variables** version of the prover, inspired by the system introduced in [**?**]. It makes use of *Prolog variables* to represent all the labels that can be used in a single application of the $(\mid\!\sim^+)$ rule. This version represents labels by integers starting from 1; by using integers we can easily express constraints on the range of the variable-labels. To this regard, the library `clpfd` is used to manage free-variable domains. In order to prove $\Gamma, u : A \mid\!\sim B$, KLMLean 2.0 will call `prove` on the following conclusions: $\Gamma, u : A \mid\!\sim B, Y : \neg A$; $\Gamma, u : A \mid\!\sim B, Y : \neg\Box\neg A$; $\Gamma, u : A \mid\!\sim B, Y : B$, where Y is a Prolog variable. Y will then be instantiated by Prolog's pattern matching to close a branch with an axiom. Predicate `Y in 1..Max` is used to define the domain of Y, where `Max` is the maximal integer occurring in the branch (i.e. the last label introduced). The list `Cond` here contains elements of the form `[a => b,Used]`, where `Used` is the list of free variables already introduced to apply $(\mid\!\sim^+)$ in the current branch. In order to ensure termination, the clause implementing $(\mid\!\sim^+)$ is applied *only if* |`Used`|< `Max`; the predicate `all_different([Y|Used])` is then invoked to ensure that all variables used to apply $(\mid\!\sim^+)$ on the same conditional will assume different values. On the unsatisfiable set $\neg(A \vee D \mid\!\sim F \vee \neg C \vee \neg B \vee A), (A \wedge B) \vee (C \wedge D) \mid\!\sim E \wedge F, \neg(P \mid\!\sim E \wedge F), \neg((A \wedge B) \vee (C \wedge D) \mid\!\sim G)$, the free-variables version succeeds in less than 2 ms, whereas the "standard" version requires 1.9 s. The performances of KLMLean 2.0 are promising. We have tested the implementation for **R** over 300 sets of formulas: it terminates its computation in 236 cases in less than 2.5 s (204 in less than 100 ms). In future research we intend to increase the performances of KLMLean 2.0 by experimenting standard refinements and heuristics.

References

1. S.Kraus, D.Lehmann, and M.Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44(1-2):167–207, 1990.
2. G.Pozzato. *Proof Methods for Conditional and Preferential Logics*. PhD Thesis.2007.
3. L.Giordano, V.Gliozzi, N.Olivetti, G. Pozzato. Analytic Tableaux for KLM Preferential and Cumulative Logics. In *Proc. of LPAR 2005*, *LNAI* 3835, 666–681. 2005.
4. L.Giordano, V.Gliozzi, N.Olivetti, and G.L. Pozzato. Analytic Tableaux Calculi for KLM Rational Logic R. In *Proc. of JELIA 2006*, *LNAI* 4160, pp. 190–202. 2006.
5. B.Beckert and J.Posegga. leantap: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.
6. B.Beckert and R.Goré. Free variable tableaux for propositional modal logics. *In Proc. of TABLEAUX 1997, LNAI 1227, Springer-Verlag*, pages 91–106, 1997.