

A Prolog implementation of KEM

Alberto Artosi*, **Paola Cattabriga****, **Guido Governatori****

**Dipartimento di Filosofia*

***CIRFID*

Università di Bologna

Università di Bologna

via Zamboni, 38, 40126 Bologna (Italy) *via Galliera, 3, 40121 Bologna (Italy)*

governat@cirfid.unibo.it *paola@cirfid.unibo.it*

Abstract In this paper, we describe a Prolog implementation of a new theorem prover for (normal propositional) modal and multi-modal logics. The theorem prover, which is called *KEM*, arises from the combination of a classical refutation system which incorporates a restricted (“analytic”) version of the cut rule with a label formalism which allows for a specialised, logic-dependent unification algorithm. An essential feature of *KEM* is that it yields a rather simple and efficient proof search procedure which offers many computational advantages over the usual tableau-based proof search methods. This is due partly to the use of linear 2-premise β rules in place of the branching β rules of the standard tableau method, and partly to the crucial role played by the analytic cut (the only branching rule) in eliminating redundancy from the search space. It turns out that *KEM* method of proof search is not only computationally more efficient but also intuitively more natural than other (e.g. resolution-based) methods leading to simple and easily implementable procedures (two *KEM* Theorem Prover-like systems have been implemented: an LPA interpreter on Macintosh, and a Quintus compiler on Sun-Sparcstation) which make it well suited for efficient automated proof search in modal logics.

1 An overview of KEM

KEM [AG94, Gov95] is a tableau-like modal proof system based on D’Agostino and Mondadori’s [DM94] classical refutation system *KE*. The basic feature of *KEM* is that it uses *KE* rules in combination with a label unification scheme constituted of (1) a label formalism, and (2) a specialised, logic-dependent unification algorithm. The label formalism arises from two (non empty) sets $\Phi_C =$

$\{w_1, w_2, \dots\}$ and $\Phi_V = \{W_1, W_2, \dots\}$ respectively of constant and variable world-symbols through the following definition: a world-label is either (i) an element of the set Φ_C , or (ii) an element of the set Φ_V , or (iii) a path term (k', k) where (iiia) $k' \in \Phi_C \cup \Phi_V$ and (iiib) $k \in \Phi_C$ or $k = (m', m)$ where (m', m) is a label. Intuitively, we may think of a label $i \in \Phi_C$ as denoting a (given) world, and a label $i \in \Phi_V$ as denoting a set or worlds (any world) in some Kripke model. A label $i = (k', k)$ may be viewed as representing a path from k to a (set of) world(s) k' accessible from k (according to the appropriate accessibility relation. For any label $i = (k', k)$ we shall call k' the *head* of i , k the *body* of i , and denote them by $h(i)$ and $b(i)$ respectively. Notice that these notions are recursive: if $b(i)$ denotes the body of i , then $b(b(i))$ will denote the body of $b(i)$, $b(b(b(i)))$ will denote the body of $b(b(i))$; and so on. We shall call each of $b(i), b(b(i))$, etc., a *segment* of i . Let $s(i)$ denote any segment of i (obviously, by definition every segment $s(i)$ of a label i is a label); then $h(s(i))$ will denote the head of $s(i)$. For any label i , we shall define the length of i , $l(i)$, as the number of world-symbols in i (obviously $l(s(i))$ will denote the length of $s(i)$). We shall call a label i *restricted* if $h(i) \in \Phi_C$, otherwise we shall call it *unrestricted*.

KEM's label unification scheme involves two kinds of unifications, respectively "high" and "low" unifications. "High" unifications are meant to mirror specific accessibility constraints. They are used to build "low" unifications which account for the full range of conditions governing the appropriate accessibility relation. Let \mathfrak{S} denote the set of labels. A substitution is defined in the usual way as a function $\Phi_V \longrightarrow \mathfrak{S}^-$ where $\mathfrak{S}^- = \mathfrak{S} - \Phi_V$. For two labels i, k and a substitution σ , if σ is a unifier of i and k , then we shall say that i and k are σ -unifiable. We shall (somewhat unconventionally) use $(i, k)\sigma$ to denote both that i and k are σ -unifiable and the result of their unification. On this basis we can define several specialised, logic-dependent notions of σ "high" (or σ^L -) unification. In particular, the notion of two labels i, k being σ^{K-} , σ^{D-} , and σ^T -unifiable is defined in the following way:

$$\begin{aligned}
(i, k)\sigma^K &= (i, k)\sigma \iff \\
&\quad (i) \text{ at least one of } i \text{ and } k \text{ is restricted, and} \\
&\quad (ii) \text{ for every } s(i), s(k), l(s(i)) = l(s(k)), (s(i), s(k))\sigma^K \\
\\
(i, k)\sigma^D &= (i, k)\sigma \\
\\
(i, k)\sigma^T &= (s(i), k)\sigma \iff \\
&\quad l(i) > l(k), \text{ and} \\
&\quad \forall h(s(i)) : l(s(i)) \geq l(k), (h(s(i)), h(k))\sigma = (h(i), h(k))\sigma \text{ or} \\
(i, k)\sigma^T &= (i, s(k))\sigma \iff \\
&\quad l(k) > l(i), \text{ and} \\
&\quad \forall h(s(k)) : l(s(k)) \geq l(i), (h(i), h(s(k)))\sigma = (h(i), h(k))\sigma.
\end{aligned}$$

In what follows we shall concentrate on *KEM* method for dealing with the *B* logics. To deal with these logics we need an appropriate notion of "reduction" of (intuitively something like the deletion of "irrelevant" steps from the path represented by) a label i . Formally, the *B-reduction*, $r_B(i)$, of a label i is defined to be

a function $r_L : \mathfrak{S} \rightarrow \mathfrak{S}$ determined as follows:

$$r_B(i) = \begin{cases} b(b(i)) & i \text{ unrestricted and either } l(i) \leq 3 \text{ or} \\ & b(i) \text{ restricted} \\ (h(i), r_B(b(i))), & i \text{ restricted} \end{cases}$$

The notion of σ “low” (or σ_L -) unification for the B logics ($L = KB, DB, B$) can now be defined as follows:

$$(i, k)\sigma_{KB} = (r_B(i, k))\sigma^K \quad (i, k)\sigma_{DB} = (r_B(i, k))\sigma^D \quad (i, k)\sigma_B = \begin{cases} (r_B(i, k))\sigma^D \\ (r_B(i, k))\sigma^T \end{cases}$$

where $r_B(i, k)$ denotes either $r_B(i)$ or $r_B(k)$ or both.

The full set of *KEM* inference rules is constituted of (i) 1–premise α rules (the familiar linear branch–expansion rules of the tableau method) and the usual ν and π rules for the modal operators (see **Alpha Elimination**, **Ni Elimination** and **Pi Elimination** in the *KEM* Algorithm Representation below); (ii) 2–premise (linear) β rules (see **Beta Elimination** below); and (iii) a 0–premise branching rule called *PB* (for Principle of Bivalence) which plays the role of the cut rule of the sequent calculus (see **PB1** and **PB2** below). Labels are manipulated, according to these rules, in such a way that (1) in all inferences via an α rule the label of the premise carries over unchanged to the conclusion; (2) in all inferences via a ν and π rule the label of premises is “updated” to an extended new (unrestricted or restricted) label; (3) in all inferences via a β rule the labels of the premises must be σ_L -unifiable, so that the conclusion inherits their unification; and (4) for the K logics, *PB* is applied only to already existing restricted labels. Closure of a branch follows from the occurrence of a pair of complementary formulas whose labels are σ_L -unifiable (let us call them σ_L -complementary).

2 Implementation

In this section we will briefly consider two main problems arising from the Prolog implementation of *KEM*. These problems are: (1) *KEM*’s label unification scheme has some idiosyncratic features; for example it does not allow a variable to be substituted to another variable; and (2) *KEM* rules are essentially non–deterministic; in particular, *PB* is not an *analytic* rule.

The well–known difficulty to handle variables in lists and terms in Prolog, on one hand, and the unification theory and the necessity of recursively generating new constants and variables, on the other, have made necessary to define constants and variables as functions of the form $w(N)$ and $vw(N)$. Thus *KEM* Interpreter has $\Phi_C = \{w(1), w(2), w(3), \dots\}$ and $\Phi_V = \{vw(1), vw(2), vw(3), \dots\}$. The labels are defined as binary terms. Let us consider a *KEM* label $(w_4, (W_3, (w_3, (W_2, w_1))))$. Its *KEM* Interpreter equivalent is $i(w(4), i(vw(3), i(w(3), i(vw(2), i(w(1), w(1))))))$. The unification theory is completely redefined without using the built in Prolog predicate “unify”. Labels are treated as binary terms and $(i, k)\sigma$, $(i, k)\sigma^L$, $(i, k)\sigma_L$ and

$r_L(i)$ are defined, using functor and arg, as ternary predicates, where the first and the second argument are i, k and the third is their unification. (For a complete description of *KEM* Prolog implementation see [Cat95]. The Interpreter is ftp available at ftp.cirfid.unibo.it.).

The *KEM*–Prolog Interpreter has been based on the notion of a canonical (deterministic) *KEM*-tree, see [AG94, Gov95]. A *KEM*-tree is said to be *canonical* if it is generated by applying the rules of *KEM* in the following fixed order: first the 1-premise rules, then the 2-premises rules, and finally the 0-premises rule (*PB*). As proved in [AG94, Gov95] a *KEM*-tree is closed iff the corresponding canonical *KEM*-tree is closed, and canonical *KEM*-trees always terminate. Notice that in a canonical *KEM*-tree *PB* is applied only to *unanalysed* or *unfulfilled* β formulas (see **Beta Elimination** below). This allows much of the characteristic redundancy generated by the standard tableau branching rules to be eliminated from the search space.

The basic data structures ([DP94, PC94]) are provided by two sets Δ , Λ of unanalysed and analyzed formulas respectively. In Prolog Δ and Λ are lists. The Interpreter starts with the list Δ of input formulas and $\Lambda = \emptyset$, and simulates the rules of *KEM* by analysing and moving formulas from Δ to Λ . Each rule application produces subformulas which are added to Δ . The rules of *KEM* are applied until an application of the branch–closure rule (see **Closure** below) succeeds or Δ is empty. In the first case Γ is closed and unsatisfiable, in the second Γ is completed and satisfiable. The *KEM* algorithm runs as follows.

KEM Algorithm Representation

Δ is the list of the unanalysed formulas, Λ is the list of the analyzed formulas, “Labeltree” is a list of the generated labels, \times denotes a closed branch

Analyse Literal: $? p \in \Delta \implies \Delta - p, \Lambda \cup p$

Closure: $? X, i$ and $X^C, k \in \Lambda$
 $? (i, k)\sigma_L \implies$
 \times

Ni Elimination: $? \nu, i \in \Delta \implies$
 generate a new unrestricted label (i', i)
 add (i', i) to Labeltree
 $\Delta - \nu, i$
 $\Delta \cup \nu_0, (i', i)$
 $\Lambda \cup \nu, i$

Pi Elimination: $? \pi, i \in \Delta \implies$
 generate a new restricted label (i', i)
 add (i', i) to Labeltree
 $\Delta - \pi, i$
 $\Delta \cup \pi_0, (i', i)$
 $\Lambda \cup \pi, i$

Alpha Elimination: $? \alpha, i \in \Delta \implies$

$$\begin{array}{l}
\Delta - \alpha, i \\
\Delta \cup \alpha_1, i \cup \alpha_2, i \\
\Lambda \cup \alpha, i \\
\text{Beta Elimination: } ? \beta, i \in \Delta \\
? \beta_1^C, k \text{ or } \beta_2^C, k \in \Delta \cup \Lambda \\
?(i, k)\sigma_L \implies \\
\Delta - \beta, i \\
\Delta \cup \beta_2^C, (i, k)\sigma_L \text{ or } \beta_1^C, (i, k)\sigma_L \\
\Lambda \cup \beta, i
\end{array}$$

$$\begin{array}{l}
\text{PB1: } ? \beta, i \in \Delta \\
? \text{ not } (\beta_1^C, k : (i, k)\sigma_L) \in \Delta \cup \Lambda \\
?(i, m)\sigma_L \\
(m \text{ is a restricted label in Labeltree}) \implies \\
\begin{array}{ll}
\text{branch1} & \text{and} & \text{branch2} \\
\Delta - \beta, i & & \Delta - \beta, i \\
\Delta \cup \beta_1, m & & \Delta \cup \beta_1^C, m \cup \beta, i \\
\Lambda \cup \beta, i & &
\end{array}
\end{array}$$

$$\begin{array}{l}
\text{PB2: } ? \beta, i \in \Delta \\
? \text{ not } (\beta_2^C : k, (i, k)\sigma_L) \in \Delta \cup \Lambda \\
?(i, m)\sigma_L \\
(m \text{ is a restricted label in Labeltree}) \implies \\
\begin{array}{ll}
\text{branch1} & \text{and} & \text{branch2} \\
\Delta - \beta, i & & \Delta - \beta, i \\
\Delta \cup \beta_2, m & & \Delta \cup \beta_2^C, m \cup \beta, i \\
\Lambda \cup \beta, i & &
\end{array}
\end{array}$$

$$\begin{array}{l}
\text{Modal Closure: } ? X, i \text{ and } X^c, k \in \Lambda \\
? \text{ not } (i, k)\sigma_L \\
? m \in \text{Labeltree}, (m \text{ is a restricted label}) \\
?(i, m)\sigma_L \\
?(k, m)\sigma_L \implies \\
\times
\end{array}$$

Modal Closure is an “hidden” application of *PB* to the the label which unifies with both the labels of the σ_L -complementary formulas.

We conclude by showing the *KEM* Prolog output of the characteristic axiom of *B*, i.e. $p \rightarrow \Box \Diamond p$.

```

:- kem(b, [~ (p-> $ (@ p))])
[i(w(1), w(1)): ~ (p-> $ (@ p))]
alpha elimination
[i(w(1), w(1)):p, i(w(1), w(1)): ~ ($ (@ p))]
literal
[i(w(1), w(1)): ~ ($ (@ p))]
pi elimination
[i(w(2), i(w(1), w(1))): ~ (@ p)]
ni elimination
[i(vw(1), i(w(2), i(w(1), w(1)))): ~ p]

```

```
literal
i(vw(1),i(w(2),i(w(1),w(1)))):~p, i(w(1),w(1)):p unify in b
unsatisfiable in b in 10 msec
N 1 yes
```

References

- [ACG94] A. Artosi, P. Cattabriga and G. Governatori. An Automated Approach to Deontic Reasoning. In J. Breuker (ed.), *Artificial Normative Reasoning*, Workshop ECAI 1994: 132–145.
- [AG94] A. Artosi and G. Governatori. Labelled Model Modal Logic. In *Proceedings of the CADE-12 Workshop on Automated Model Building*, 1994: 11–17.
- [Cat95] P. Cattabriga. *Sistemi algoritmici indicizzati per il ragionamento giuridico*, PhD. Thesis, University of Bologna, 1995.
- [DM94] M. D’Agostino and M. Mondadori. The Taming of the Cut. *Journal of Logic and Computation*, 4, 1994: 285–319.
- [DP94] M. D’Agostino and J. Pitt. Private Communication, 1994.
- [Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logic*, D. Reidel Publishing Company, Dordrecht, 1983.
- [Gov95] G. Governatori. Labelled Tableaux for Multi-Modal Logics. In Peter Baumgartner, Reiner Hähnle and Joachim Posegga (eds.), *4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Berlin, Springer Verlag, LNAI, 1995, 79–94.
- [PC94] J. Pitt and J. Cunningham. Theorem Proving and Model Building with the Calculus KE. In *MEDLAR II, Esprit Basic research Project 6471*, Deliverable D IV.1.2-5P: 538–553.