A Modal Computational Framework for Default Reasoning

Alberto Artosi[1], Paola Cattabriga[1], and Guido Governatori[2]

[1] Dipartimento di Filosofia, Università di Bologna,
via Zamboni 28, 40126 Bologna, Italy, Fax +39(0)51-258326
[2] Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ
E-mail: {artosi,paola,governat}@cirfid.unibo.it

Usually a default rule $A : B/C$ is intended to mean that if A holds in a state of affairs a B is consistent, then C follows by default. However, C is not a necessary conclusion: different states of affairs are possible (conceivable). According to this view, Meyer and van der Hoek [MvH92] developed a multimodal logic, called $S5P_{(n)}$, for treating non-monotonic reasoning in a monotonic setting. In this paper we shall describe a proof search algorithm for $S5P_{(n)}$ which has been implemented as a Prolog Interpreter.

$S5P_{(n)}$ arises as a combination of $S5$ with n distinct $K45$ "preference" modalities P_i $(1 \leq i \leq n)$ characterized by the following axioms:

1.$\Box P_i A \equiv P_i A$	3.$\neg P_i \bot \to (P_i \Box A \equiv \Box A)$
2.$\neg P_i \bot \to (P_i P_j A \equiv P_j A)$	4.$\Box A \to P_i A (1 \leq i \leq n)$.

The semantics for $S5P_{(n)}$ is given in terms of clusters of preferred worlds.

To "simulate" defeault reasoning in $S5P_{(n)}$ we simply have to translate the usual default rules in the $S5P_{(n)}$ language. The $S5P_{(n)}$ version of Reiter's rule is $A \wedge \Diamond B \to P_i C$ meaning that if A is true and B is considered possible then C is preferred. Similarly, normal defaults can be expressed as $A \wedge \Diamond B \to P_i B$ and multiple defaults as $A_1 \wedge \Diamond B_1 \to P_1 C_1$, $A_2 \wedge \Diamond B_2 \to P_2 C_2 \ldots$ where P_1 and P_2 are preference operators associated with distinct preferred sets.

To compute inferences in $S5P_{(n)}$ we need the following label formalism. Let $\Phi_C^i = \{w_1^i, w_2^i, \ldots\}$ and $\Phi_V^i = \{W_1^i, W_2^i, \ldots\}$ $(0 \leq i \leq n)$ be (nonempty) sets respectively of constants and variable "world" symbols. An element of the set $\Im$ of "world" labels (henceforth labels) is either (i) an element of Φ_C^i, or (ii) an element of Φ_V^i, or (iii) a path term (k', k) where (iiia) $k' \in \Phi_C^i \cup \Phi_V^i$ and (iiib) $k \in \Phi_C^i$ or $k = (m', m)$ where (m', m) is a label. Intuitively we may think of a label $i \in \Phi_C^i$ as denoting a world, and a label $i \in \Phi_V^i$ as denoting a set of worlds (any world) in cluster of preferred i-worlds. A label $i = (k', k)$ may be viewed as representing a path from k to a (set of) world(s) k' accessibile from k. From now on we shall use $i, j, k \ldots$ to denote arbitrary labels. For any label $i = (k', k)$ we call k' the head of i, k the body of i, and denote them by $h(i)$ and $b(i)$ respectivelly. Notice that these notions are recursive: if $b(i)$ denotes the body of im 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 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 $(s(i))$. We call a label i restricted if $h(i) \in \Phi_C$, otherwise we call it unrestricted. We shall say that a label k is *i-preferred* iff $k \in \Im^i$ where $\Im^i = \{k \in \Im : h(k)$ is either w^i_m or W^i_m, $1 \leq i \leq n\}$, and that a label k is *i-ground* $(1 \leq i \leq n)$ iff: 1) $\forall s(k) : h(s(k)) \notin \Phi^i_V$, and 2) if $\exists s^m(k) : h(s^m(k)) \in \Phi^i_V$, then $\exists s^j(k), j < m : h(s^j(k)) \in \Phi^i_C$.

The formalism just described alows labels to be manipulated in a way closed related to the semantics of modal operators and "matched" using a specilaized (logic-dependent) unification algorithm. For two labels i, k and a substitution σ we shall use $(i,k)\sigma$ to denote both that i and k are σ-unifiable and the result of their unification. On this basis we may go on to define the notion of two labels i, k being $\sigma^{S5P_{(n)}}$-unifiable in the following way:

$$\begin{array}{ll} \sigma^* : \Phi^0_V \longrightarrow \Im^- \Phi^i_V, (1 \leq i \leq n) & \sigma^{S5P_{(n)}} : \Phi_V \longrightarrow \Im^- \\ \quad : \Phi^i_V \longrightarrow \Phi^i_C, (1 \leq i \leq n) & \qquad\qquad : \Phi^i_V \longrightarrow \Im^i, (1 \leq i \leq n). \end{array}$$

The corresponding PTP ("PROLOG Theorem Prover" [ACG95,Cat95]) clauses are:

```
unifypn(vw(N),vw(N1),vw(N2)):- (N >= N1, N2 = N); N1 =N2.
unifypn(w(N),vw(N1),w(N)).
unifypn(vw(N1),w(N),w(N)).
unifypn(w(N),w(N),w(N)).
unifypn(vw(N1),w(J,N),w(J,N)).
unifypn(w(J,N),vw(N1),w(J,N)).
unifypn(vw(J,N),vw(J,N1),vw(J,N2)):- (N >= N1, N2 = N); N1 =N2.
unifypn(w(J,N),vw(J,N1),w(J,N)).
unifypn(vw(J,N1),w(J,N),w(J,N)).
unifypn(w(J,N),w(J,N),w(J,N)).
unifypn(i(A,B),i(C,D),i(E,G)):- functor(i(A,B),F,N),
    functor(i(C,D),F,N), unifyargspn(N,i(A,B),i(C,D),i(E,G)).
unifyargspn(N,X,Y,T):- N>0, unifyargpn(N,X,Y,AT), N1 is N - 1,
    functor(T,i,2), arg(N,T,AT), unifyargspn(N1,X,Y,T).
unifyargspn(0,X,Y,T).
unifyargpn(N,X,Y,AT):- arg(N,X,AX), arg(N,Y,AY), unifypn(AX,AY,AT).
```

We are now able to define the notion of $\sigma_{S5P_{(n)}}$-unification as follows:

$$\begin{aligned} (i,k)\sigma_{S5P_{(n)}} = {} & (h(i),h(k))\sigma^* \text{ if} \\ & i,k \text{ are } i\text{-ground}, 1 \leq i \leq n, \text{ or} \\ & \exists s(i), s(k) : h(s(i)), h(s(k)) \in \Phi^i, \text{ and } (h(s(i)), h(s(k))\sigma^{S5P_{(n)}} \end{aligned}$$

PTP clauses:

```
unifydefault(T1,T2,T3):- iground(T1), iground(T2),
    arg(1,T1,H1), arg(1,T2,H2), unifypn(H1,H2,T3), !.
unifydefault(T1,T2,T3):- isegment(T1,i(H1,B1)),
    isegment(T2,i(H2,B2)), unifydefault(H1,H2,T3).
isegment(I,S):- (subterm(i(w(J,N),K),I), i(w(J,N),K)=S;
    subterm(i(vw(D,M),H),I), i(vw(D,M),H)=S),!.
iground(I):- ( compound(I), I =.. [F], not memb(vw(A,B),F); ig(F);
    (subterm(i(vw(H,M),K),I), subterm(w(C,D),K))),! .
ig([]):- !.
ig([T|B]):- (T = w(H); T = vw(G); T = w(A,B)), ig(B).
```

In contrast with the usual branch-expansion rules of the tableau method, all the rules involved in the following proof search algorithm are linear. Their application generates a one-branch refutation tree (thus eliminating redundancy from the search space). Splitting occurs only as a result of applying the "cut rule" in steps 9, 10 below. The algorithm works with formulas of the form X, i called *labelled formulas* (ℓ-formulas). Formulas will be expressed in Smullyan-Fitting's "α, β, ν, π" notationwith the following addition: formulas of the forms P_iA and $\neg P_iA$ will be classified, in analogy with ν and π type formulas, as being of type $p_i\nu$ and $p_i\pi$ respectively. As usual X^C will be used to denote the conjugate of X (i.e. $\neg Z$ if $X = Z$, and *viceversa*). The algorithm is displayed in its most general formulation, with "L" to be replaced by "$S5P_{(n)}$" or by any other logic anong those treated in [AG94,Gov95] (to which the reader is also referred for all details). The procedure is based on *canonical* trees. A tree is canonical iff it is generated by applying the inference rules in the following fixed order: first the 1-premise rules (see steps 3,4,5,6,7), then the 2-premise rules (see step 8), and finally the 0-premise (cut) rule. An essential property of canonical trees is that they always terminate, thus providing a computable algorithm.

Preliminary definitions. Two ℓ-formulas X, i and X^C, k, such that $(i, k)\sigma_L$ are called σ_L-complementary. An ℓ-formula is said to be *E-analysed in a branch* τ if either (i) X is of type α and both α_1, i and α_2, i occur in τ; or (ii) X is of type β and the following condition is satisfied: if β_1^C, k (resp. β_2^C, k)occurs in τ and $(i, k)\sigma_L$, then also $\beta_2, (i, k\sigma_L)$ (resp. $\beta_1, (i, k)\sigma_L)$ occurs in τ; or (iii) X is of type ν and $\nu_0, (i', i)$ occurs in τ for some $i' \in \Phi_V$ not previously occurring in τ, or (iv) X is of type π and $\pi_0, (i', i)$ occurs in τ for some $i' \in \Phi_C$ not previously occurring in τ, similarly if X is of type $p_i\nu$ or $p_i\pi$. A branch τ is said to be *E-completed* if every ℓ-formula in it is E-analysed and there are no complementary formulas which are not σ_L-complementary. We say that a branch τ is *completed* if it is E-completed and all the ℓ-formulas of type β in it are either analysed or cannot be analysed. We call a tree *completed* if every branch is completed. Finally, a branch τ is σ_L-closed if it contains a pair of σ_L-complementary ℓ-formulas, and a tree is σ_L-closed if all its branches are σ_L-closed.

Let Λ, Δ denote sets of analysed and unalysed ℓ-formulas respectively, and $\mathcal{L}$ the set of generated labels. To prove a formula X of L start the following algorithm with X^C, i (where i is an arbitray constant label) in Δ, and i is in $\mathcal{L}$.
STEP 1 . If a pair of σ_L-complementary ℓ-formulas occurs in Δ, then the tree is σ_L-closed. A is a theorem of L.
STEP 2. If Δ is empty, then the tree is completed. Every literal is deleted from Δ, and added to Λ.
STEPS 3, 4. For each ℓ-formula ν, i (π, i) in Δ, (i) generate a new unrestricted (restricted) label (i',i) and add it to $\mathcal{L}$; (ii) delete ν,i $(\pi$,$i)$ from Δ; (iii) add ν_0,$(i'$,$i)$ $(\pi_0$, $(i'$,$i))$ to Δ; and (iv) add ν,i $(\pi$,$i)$ to Λ.
STEPS 5, 6. For each ℓ-formula $p_i\nu, k$, $(\neg p_i\nu, k)$ in Δ, (i) generate a new unrestricted (restricted) label (m^i, k) and add it to $\mathcal{L}$; (ii) delete $p_i\nu, k$ $(\neg p_i\nu, k)$ from Δ; (iii) add $p_i\nu_0, (m^i, k)$ $(\neg p_i\nu_0, (m^i, k))$ to Δ; and (iv) add $p_i\nu, k$ $(\neg p_i\nu, k)$ to Λ.

STEP 7. For each ℓ-formula α, i in Δ, (i) add α_1, i, and α_2, i to Δ; (ii) delete α, i from Δ; and (iii) add α, i to Λ.
STEP 8. For each ℓ-formula β, i in Δ, such that either β_1^C,k or β_2^C,k is in $\Delta \cup \Lambda$ and $(i,k)\sigma_L$ for some label k, (i) add $\beta_2(i,k)\sigma_L$ or $\beta_1(i,k)\sigma_L$ to Δ; (ii) delete β,i from Δ; and (iii) add the labels resulting from the σ_L-unification to $\mathcal{L}$; and (iv) add β,i to Λ.
STEP 9. For each ℓ-formula β,i in Δ, if $\Delta \cup \Lambda$ does not contains formulas β_1^C,k such that i, k are not σ_L-unifiable, then form sets $\Delta_1 = \Delta \cup \beta_1,m$, $\Lambda_1 = \Lambda \cup \beta_i$, $\Delta_2 = \Delta \cup \beta_1^C, m \cup \beta, i$ where $(i,m)\sigma_L$, and m is a given restricted label, and $\Lambda_2 = \Lambda$.
STEP 10. For each ℓ-formula β, i in Δ, if $\Delta \cup \Lambda$ does not contains formulas β_2^C,k such that i, k are not σ_L-unifiable, then form sets $\Delta_1 = \Delta \cup \beta_2,m$, $\Lambda_1 = \Lambda \cup \beta_i$, $\Delta_2 = \Delta \cup \beta_2^C, m \cup \beta, i$ where $(i,m)\sigma_L$, and m is a given restricted label, and $\Lambda_2 = \Lambda$.
STEP 11, 12. If Λ contains two complementary formulas which are not σ_L-complementary ℓ-formulas, searc in $\mathcal{L}$ for restricted labels which σ_L-unify with both the labels of the complametary formulas; if we find (do not find) such labels then the tree is σ_L-closed (completed). A is (is not) a theorem of L.

In this paper we have presented a proof system for computing default reasoning in a monotonic setting. The above algorithm can be used to verify whether a conclusion C is implied by a (multiple) default D (where D denotes the conjunction of the $S5P_{(n)}$ translation of the default(s)) and, thanks to the distinctive features of the label formalism it uses, it yields a countermodel similar to the state of affairs corresponding to the default(s).

References

[ACG95] A.Artosi, P. Cattabriga and G. Governatori. A Prolog implementation of KEM. In M. Alpuente and M. I. Sessa (eds.), *Proceedings of the GULP-PRODE'95 Joint Conference on Declarative Programming. Marina di Vietri, 11–14 september 1995*, Università degli Studi di Salerno, 1995: 395–400.

[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] cattabriga P. *Sistemi algoritmici indicizzati per il ragionamento giuridico.* PhD Thesis, University of Bologna, 1996.

[Gov95] Governatori G.. Labelled Tableaux For Multi-Modal Logics. In P. Baumgartner, R. Hähnle, and J. Posegga (eds.), *Theorem Proving with Analytic Tableaux and Related Methods*, Lecture Notes in Computer Science, Springer-Verlag, 1995: 79–84.

[MvH92] J.J.Ch. Mayer and W. van der Hoeck. A Modal Logic for Nonmonotonic Reasoning. In W. van der Hoeck,J.J.Ch. Mayer, Y. H. Tan and C. Witteveen (ed.),*Non-Monotonic Reasoning and Partial Semantics.* Ellis Horwood: N.Y.. 1992: 37–77.