Computing the Extensions of Autoepistemic and Default Logics with a Truth Maintenance System

Ulrich Junker
GMD, F3-HIS
Postfach 1240
5205 St. Augustin 1, F.R.G.
unido!gmdzi!junker@uunet.uu.net

Kurt Konolige
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025/USA
konolige@ai.sri.com

Abstract

In this paper we develop a proof procedure for autoepistemic (AEL) and default logics (DL), based on translating them into a Truth Maintenance System (TMS). The translation is decidable if the theory consists of a finite number of defaults and premises and classical derivability for the base language is decidable. To determine all extensions of a network, we develop variants of Doyle's labelling algorithms.

Introduction

We seek proof methods for autoepistemic logic (AEL) [11] and default logic (DL) [14]. Most existing techniques are too restrictive in terms of the language allowed. Reiter's original proof procedure is restricted to closed normal defaults [14]. Etherington handles only ordered network theories, a special class of general defaults which are suitable for inheritance [4]. Kautz and Selman are mainly interested in complexity results and restrict their attention to disjunction-free default theories [7]. Brewka's prover handles modal implications having no preconditions of the form Lq [1]. Moore's procedure for AEL is restricted to a propositional language [12], just as Niemelä's work [10]. Closer to our approach is the work of Levesque on quantified AEL [9], which has a proof procedure under the same conditions as ours; however Levesque's proof theory is a Hilbert-style axiomatization, without an efficient implementation.

We want to develop a more general proof procedure for AEL and DL which profits from efficient truth maintenance (TMS) techniques. To this end, we translate an AEL or DL theory into a TMS network, and relate the admissible labeling of the network to extensions of the original theory. It is already known from the work of Reinfrank, Dressler, and Brewka [13] that the TMS can be given a semantics by mapping to a restricted form of AEL or DL; here we give the inverse transformation. Our approach has only two limits: 1) We do not handle infinite sets of defaults and premises. 2) Derivability has to be decidable for the base language. If both requirements are satisfied our procedure is able to determine all extensions and to detect incoherence, i.e., non-existence of an extension. To achieve this goal we tackle three problems:

1. Extensions of autoepistemic and default theories are defined as infinite fixed points of monotonic operators, while the TMS is finite. Are there computable, finite conditions for extensions?

2. Doyle's TMS lacks complete first-order derivability. What first-order derivations must be encoded by justifications to get all needed conclusions?

3. TMS labeling algorithms (cf. [2], [5]) are incomplete because they compute only one extension and can stop without a definite result if there are odd loops. How can we get all extensions?

In outline the strategy of the paper is as follows. We concentrate initially on DL because its fixed-point condition is close to that of the TMS. First we identify all formulas which are relevant for the fixed point conditions, and develop alternative operators whose fixed points consist of relevant formulas. This set of formulas is finite for a finite number of defaults and allows reconstruction of every extension. The appropriate TMS can then be constructed by considering only the relevant formulas as nodes, with justifications coming from the defaults and from appropriate first-order proofs relating the relevant formulas. Admissible labelings of the TMS provably correspond to extensions of the DL theory.

For AEL the story is slightly more complicated. Because of the relation between DL and strongly-grounded extensions of AEL, the above method can be transferred directly. However, in the case of weakly-grounded extensions, the definition of admissible labeling must be changed to allow circular support in the TMS.

Finally, we sketch a complete version of Doyle's labeling algorithm. As in [6], it introduces choice points for loops and performs backtracking if it runs into an odd loop.

Closures of operators

In this paper, we use extensively operators that apply defaults or justifications. Here we give notation and properties for these operators. Let U be a domain and $apply : 2^U \mapsto 2^U$ be a monotonic and compact operator:

$$\begin{array}{ll} \text{if } X \subseteq Y \text{ then } apply(X) \subseteq apply(Y) & \\ \text{if } q \in apply(Y) \text{ then } \exists X \subseteq Y : X \text{ is finite and} & (1) \\ \quad q \in apply(X) & \end{array}$$

Repeated application of *apply* leads to an operator $apply^i$ which is also monotonic and compact. If we apply it to the empty set it satisfies $apply^i(\emptyset) \subseteq apply^{i+k}(\emptyset)$. Now we define the transitive and reflexive closure of *apply*:

Definition 1 *Let apply* : $2^U \mapsto 2^U$ *be a monotonic and compact operator. The closure* $apply^*(X)$ *is the minimal set that contains* X *and is closed w.r.t. apply, i.e.* $apply(apply^*(X)) \subseteq apply^*(X)$.

We mention some alternative characterizations of $apply^*$ in Lemma 1. Minimality of $apply^*$ can be ensured by iterating *apply* or by finite derivation chains.

Lemma 1 *Let apply* : $2^U \mapsto 2^U$ *be a monotonic, and compact operator. The following conditions are equivalent:*

1. $T = apply^*(\emptyset)$.
2. $apply(T) = T \subseteq apply^*(\emptyset)$.
3. $T = \bigcup_{i=0}^{\infty} apply^i(\emptyset)$
4. $apply(T) \subseteq T$ *and* $\forall q \in T\ \exists q_1, \ldots, q_k : q = q_k$ *and* $q_i \in apply(\{q_1, \ldots, q_{i-1}\})$ *for* $i = 1, \cdots, k$.

Very often, we consider sets being closed w.r.t. two operators $apply_1$ and $apply_2$. For this purpose, we write $(apply_1 \cup apply_2)(X)$ for $apply_1(X) \cup apply_2(X)$. Consider $(apply_1 \cup apply_2)^*(\emptyset)$. Here, both operators are applied in parallel. Sometimes, a sequential view yields more insights: We first use $apply_2$ and then determine the closure w.r.t. $apply_1$. Thus, we get a concatenated operator $apply_1^* \circ apply_2$.

Lemma 2 *Let* $apply_1, apply_2 : 2^U \mapsto 2^U$ *be two monotonic and compact operators. Then* $(apply_1 \cup apply_2)^*(\emptyset) = (apply_1^* \circ apply_2)^*(\emptyset)$.

A reduced fixed point condition for DL

In this section we reduce the infinite fixed-point extensions of DL to finite ones, which are called the *extension bases*. Let $\mathcal{L}_0$ be a first-order language containing a contradiction formula $\perp$. A default theory Δ is a pair (D, W) consisting of a set D of defaults and a set of first-order premises $W \subseteq \mathcal{L}_0$. Each default has the form $(a : b_1, \cdots, b_k/c)$ where $a, b_1, \cdots, b_k, c$ are ordinary formulas. Informally, we say that the default is *applied* if its prerequisite a is provable and $b_1, \cdots, b_k$ are consistent, i.e. all $\neg b_i$ are not provable. We abbreviate the set of prerequisites, the set of negated consistency assumptions (i.e. exceptions), and the set of conclusions by

$$\begin{array}{lcll} P_D & = & \{a \mid (a : b_1, \cdots, b_k/c) \in D\} & \\ E_D & = & \{\neg b_i \mid (a : b_1, \cdots, b_k/c) \in D, i \in \{1, \ldots, k\}\} & \\ C_D & = & \{c \mid (a : b_1, \cdots, b_k/c) \in D\} & (2) \end{array}$$

Let $Th(X)$ be the set of first-order conclusions of X. We also introduce a special operator that applies defaults. We use different sets X, Y to check both kinds of preconditions. Our operator applies a default $(a : b_1, \cdots, b_k/c)$ if $a \in X$ and $\neg b_i \notin Y$:

$$\begin{array}{l} apply_{D,Y}(X) := \{c \mid (a : b_1, \cdots, b_k/c) \in D, a \in X, \\ \qquad \text{and } \neg b_i \notin Y \text{ for } i = 1, \cdots, k\} \end{array} \qquad (3)$$

To be more precise, we have defined a unary operator for every Y. These operators have nice properties: they are monotonic and compact and we can apply the results of the previous section in reformulating Reiter's operator Γ. $\Gamma(T)$ is the smallest set that contains W and is closed w.r.t. the first-order closure Th and our operator $apply_{D,T}$. Hence, $\Gamma(T) = (Th \cup apply_{D,T})^*(W)$. An extension is defined using Γ:

Definition 2 *(Reiter): Let* $\Delta = (D, W)$ *be a closed default theory. A first-order set* T *is an extension of* Δ *iff* $\Gamma(T) = T$.

We now discuss two problems:

1. What are the relevant formulas to establish the fixed point condition?
2. How can we reduce the argument of Γ to a finite domain?

Every extension is an infinite set and it is difficult to find a set T satisfying the fixed point condition. What part of T is really needed for determining extensions? The operator $apply_{D,T}$ checks $\neg b_i \notin T$ for every default $(a : b_1, \cdots, b_k/c)$ and ignores the rest of T. In addition to these negated consistency assumptions, the prerequisites of defaults are relevant because derived prerequisites are needed to apply further defaults. Thus our test domain is defined by

$$\mathcal{L}_D := P_D \cup E_D.$$

The relevant part of a set T is $B = T \cap \mathcal{L}_D$ and satisfies following properties: $apply_{D,T} = apply_{D,B}$ and $\Gamma(T) = \Gamma(B)$. If T is an extension then $B = T \cap \mathcal{L}_D = \Gamma(T) \cap \mathcal{L}_D = \Gamma(B) \cap L_D$. Thus, we have obtained a fixed point condition for a test domain $\mathcal{L}_D$ which is finite if there are a finite number of defaults. We can now define a reduced extension, called the extension base, as follows.

Definition 3 *Let* $\Delta = (D, W)$ *be a closed default theory.* B *is an extension base of* Δ *iff* $B = \Gamma(B) \cap \mathcal{L}_D$.

Hence, if T is an extension then $T \cap \mathcal{L}_D$ is an extension base. Conversely, if B is an extension base then $\Gamma(B) = \Gamma(\Gamma(B) \cap \mathcal{L}_D) = \Gamma(\Gamma(B))$ and $\Gamma(B)$ is an extension, giving the following theorem.

Theorem 3 *Let $\Delta = (D, W)$ be a closed default theory. There exists a bijective mapping of the set of extensions of Δ to the set of extension bases of Δ.*

There is at most one inconsistent extension of Δ. If B is the extension base of an inconsistent extension then $\bot \in \Gamma(B)$ and $B = \mathcal{L}_D$ since an inconsistency implies every formula. We can check $\bot \in \Gamma(\mathcal{L}_D)$ to detect an inconsistent extension base.

Now, we simplify $\Gamma(B) \cap \mathcal{L}_D$ which is needed in Definition 3. We want to reduce the necessary first-order derivations as much as possible. First, we introduce a new operator that inserts W and then derives first-order consequences. We write it down using a lambda-expression, namely $\lambda(X).Th(X \cup W)$. Then $\Gamma(B)$ is the minimal set that is closed w.r.t. $\lambda(X).Th(X \cup W)$ and $apply_{D,B}$. By Lemma 2, we get:

$$\Gamma(B) = ((\lambda(X).Th(X \cup W)) \circ apply_{D,B})^*(\emptyset) \quad (4)$$

Since $apply_{D,B}$ expects only prerequisites which are in $\mathcal{L}_D$ and $\lambda(X).Th(X \cup W)$ is only supplied with consequents which are in C_D we can show

$$\begin{aligned}\Gamma(B) \cap \mathcal{L}_D = \\ ((\lambda(X).(X \cup (Th((X \cap C_D) \cup W) \cap \mathcal{L}_D))) \circ \\ apply_{D,B})^*(\emptyset) \cap \mathcal{L}_D\end{aligned} \quad (5)$$

consistency assumptions from consequents and premises. Equation 5 allows a simple translation to TMS.

Compact fixed point conditions for the TMS

Doyle's TMS maintains a network $\nu := (N, J)$ consisting of a set N of nodes and a set J of justifications. A justification $\langle in(I), out(O) \rightarrow c\rangle$ for a node c consists of a finite in-list $I \subseteq N$, a finite out-list $O \subseteq N$, and a consequent c. A justification is applied if all members of its in-list are believed, but no member of its out-list. As for default logic, we introduce an operator $apply_{J,Y}$ which is monotonic and compact:

$$\begin{aligned}apply_{J,Y}(X) := \quad \{c \mid \langle in(I), out(O) \rightarrow c\rangle \in J, \\ I \subseteq X \text{ and } O \cap Y = \emptyset\}\end{aligned} \quad (6)$$

Another property of $apply_{J,Y}$ is important for computing extensions, namely:

$$\text{if } X \subseteq Y \text{ then } apply_{J,X}(T) \supseteq apply_{J,Y}(T) \quad (7)$$

T is an extension of a justification network if T is the minimal set closed w.r.t. $apply_{J,T}$:

Definition 4 *Let $\nu = (N, J)$ be a justification network. T is an extension of ν iff $T = apply^*_{J,T}(\emptyset)$.*

Using the results of the section on operators, we find alternatives to definition 4. We split the set J of all justifications into a set M of monotonic justifications and a set NM of non-monotonic justifications. Monotonic justifications have an empty out-list and we write $apply_M(X)$ for $apply_{M,\emptyset}(X)$ which is equal to $apply_{M,T}(X)$ for all T. Then we obtain

$$apply_{J,T}(X) = apply_{NM,T}(X) \cup apply_M(X). \quad (8)$$

Lemma 4 *Let $\nu = (N, J)$ be a justification network, $M \subseteq J$ be a set of monotonic justifications and $NM := J - M$. The following sentences are equivalent:*

1. *T is an extension of ν, i.e. $T = apply^*_{J,T}(\emptyset)$.*
2. *$apply_{J,T}(T) \subseteq T$ and $\forall q \in T\ \exists q_1, \ldots, q_k : q = q_k$, and $q_i \in apply_{J,T}(\{q_1, \ldots, q_{i-1}\})$ for $i = 1, \cdots, k$.*
3. *$T = (apply^*_M \circ apply_{NM,T})^*(\emptyset)$.*

Property (2) agrees with usual definitions of extensions as closed and grounded sets (cf. [3], [13]). The third alternative allows a clear translation of AEL and DL to TMS. We map first-order derivations to monotonic justifications and defaults or modal formulas to non-monotonic justifications.

First-order derivations

We need to derive TMS justifications for the first-order operator $\lambda(X).Th(X \cup W)$. To limit the generated justifications, we use subsets of X which are in a domain U and we are interested in theorems contained in a range R. For default logic, we choose $U := C_D$ and $R := \mathcal{L}_D = P_D \cup E_D$.

To realize $Th((X \cap U) \cup W) \cap R$, we try to find all minimal proofs for elements of the range R. More precisely, we determine all minimal subsets Q of U inferring a $q \in R$ in conjunction with W (i.e. $Q \cup W \models q$ and $Q' \cup W \not\models q$ for every proper subset Q' of Q).[1] For this purpose, we apply a proof method that returns a goal q as well as the premises Q used to prove q; LR-resolution would work here. The proof is translated into a monotonic justification $\langle in(Q) \rightarrow q\rangle$. Let $M_W(U, R)$ be the set of all justifications obtained by this method:

$$\begin{aligned}M_W(U, R) := \quad \{\langle in(Q) \rightarrow q\rangle \mid Q \text{ is a minimal subset} \\ \text{of } U \text{ s.t. } Q \cup W \models q \text{ for } q \in R\}\end{aligned} \quad (9)$$

in question and we get:

$$apply_{M_W(U,R)}(X) = Th((X \cap U) \cup W) \cap R \quad (10)$$

w.r.t. $apply_{M_W(U,R)}$. Hence we obtain the closure of $apply_{M_W(U,R)}$ if we add the input:

$$apply^*_{M_W(U,R)}(X) = X \cup (Th((X \cap U) \cup W) \cap R) \quad (11)$$

Here, we translated complete proofs into justifications. Other encodings of first-order derivations including intermediate steps are conceivable.

Finding all minimal proofs is not computable in general. Therefore, our approach works only when W is a decidable subset of $\mathcal{L}_0$ (e.g., clauses with a finite herbrand universe). However, the source of the non-semi-decidability of DL is the non-decidability of classical logic, not the infinite fixed points.

[1] In fact, it is not necessary to determine minimal subsets: we can use any subset that infers q. But the minimal subsets give the minimal sufficient set of justification rules.

Translating default theories to a TMS

We are now in a position to translate both the defaults and first-order part of a default theory into a TMS network. We translate each default $(a : b_1, \cdots, b_k/c)$ of D into a non-monotonic justification $\langle in(a), out(\neg b_1, \ldots, \neg b_k) \rightarrow c\rangle$. This justification is applied if and only if the default is applied. We put all these justifications into a set NM and get

$$apply_{NM,T}(X) = apply_{D,T}(X) \tag{12}$$

Furthermore, we supply enough monotonic justifications for first-order derivations. As pointed out in the previous section, we take $M := M_W(C_D, \mathcal{L}_D)$.

Definition 5 *Let $\Delta = (D, W)$ be a default theory. Its TMS-translation is $\nu_\Delta := (\mathcal{L}_D \cup C_D, NM \cup M)$ where $M := M_W(C_D, \mathcal{L}_D)$ and $NM := \{\langle in(a), out(\neg b_1, \ldots, \neg b_k) \rightarrow c\rangle \mid (a : b_1, \cdots, b_k/c) \in D\}$.*

Now we concatenate $apply^*_M$ and $apply_{NM,T}$ and relate them to Γ using equation 5:

$$\begin{aligned}(apply^*_M \circ apply_{NM,T})^*(\emptyset) = \\ (\lambda(X).(X \cup (Th((X \cap C_D) \cup W) \cap \mathcal{L}_D)))\circ \\ apply_{D,T})^*(\emptyset)\end{aligned} \tag{13}$$

$$(apply^*_M \circ apply_{NM,T})^*(\emptyset) \cap \mathcal{L}_D = \Gamma(T) \cap \mathcal{L}_D \tag{14}$$

If T is an extension of ν_Δ then $T \cap \mathcal{L}_D = \Gamma(T) \cap \mathcal{L}_D = \Gamma(T \cap \mathcal{L}_D) \cap \mathcal{L}_D$ and $T \cap \mathcal{L}_D$ is an extension base of Δ. Conversely, if B is an extension base of Δ consider $T := (apply^*_M \circ apply_{NM,B})^*(\emptyset)$. Then $T \cap \mathcal{L}_D = \Gamma(B) \cap \mathcal{L}_D = B$ and $apply_{NM,T} = apply_{NM,T\cap\mathcal{L}_D} = apply_{NM,B}$, and T is an extension of Δ.

Theorem 5 *Let $\Delta = (D, W)$ be a default theory and ν_Δ its TMS-translation. There exists a bijective mapping of the set of extension bases of Δ to the set of extensions of ν_Δ.*

This is the main result. We discuss some examples.

1) If $D = \{(a : /a)\}$ and $W = \emptyset$ then the test domain $\mathcal{L}_D = \{a\}$ consists of a single element. There are no interesting first-order derivations, and the network consists simply of a single justification, $\langle in(a) \rightarrow a\rangle$. Its single extension is empty.

2) If $D = \{(: a/b)\}$ and $W = \{b \supset \neg a\}$ then the test domain $\mathcal{L}_D = \{\neg a\}$ again consists of a single element. There is one minimal first-order derivation from the domain $C_D = \{b\}$ to the range $\mathcal{L}_D$, namely $b \cup W \models \neg a$. The network consists of a two justifications ($\langle out(\neg a) \rightarrow b\rangle$ and $\langle in(b) \rightarrow \neg a\rangle$), and has no extensions because of the odd loop.

3) $D = \{(: \neg a/b), (: \neg b/a)\}$, $W = \{a \supset b\}$ is a more complex theory. Here, $\mathcal{L}_D = \{a, b\}$. There is one minimal first-order derivation, $a \cup W \models b$. The resulting justification network ($\langle out(a) \rightarrow b\rangle$, $\langle out(b) \rightarrow a\rangle$, and $\langle in(a) \rightarrow b\rangle$) has a single extension, $\{b\}$.

Thus, we have translated default theories into finite justification networks. Using this translation, we get a proper criterion for incoherence: if a theory has no extension then its TMS-translation contains an odd loop.

Our translation makes no special use of a contradiction node. Therefore, standard labeling routines are also sufficient to determine inconsistent extensions. However, a high price is paid because a lot of justifications are needed for this purpose. If a subset Q of the domain U is inconsistent then $\langle in(Q) \rightarrow q\rangle$ is included for every element q of our range R. If we handle inconsistent extensions separately we can reduce this effort and replace the set $\{\langle in(Q) \rightarrow q\rangle \mid q \in R\}$ by a single justification $\langle in(Q) \rightarrow \bot\rangle$. Consistent extensions of this modified network correspond to consistent extension bases of the theory.

Autoepistemic logic

We now turn to AEL. The results of the previous section carry over to strongly-grounded extensions of AEL theories, by first translating to DL, and then to a TMS. In this section we treat weakly-grounded extensions. The development is similar to that of the previous two sections. The only real difference is in the definition of extension for TMS networks: we must allow circular justifications for some nodes.

We discuss autoepistemic logic using the terminology and concepts in [8]. Hence, we have a first-order language $\mathcal{L}_0$ containing a contradiction formula $\bot$. Extending $\mathcal{L}_0$ by a modal operator L leads to a modal language $\mathcal{L}_{ael}$. If q is a closed formula of $\mathcal{L}_{ael}$ the modal literal Lq is also in $\mathcal{L}_{ael}$. We use some abbreviations from [8]. Two operators apply L and $\neg L$ to each element of a set; T_0 contains the ordinary formulas of T; and $\bar{T}$ is the complement of T:

$$\begin{aligned} LT &= \{Lq \mid q \in T\} \\ \neg LT &= \{\neg Lq \mid q \in T\} \\ T_0 &= T \cap \mathcal{L}_0 \\ \bar{T} &= (\mathcal{L}_{ael} - T) \end{aligned}$$

We must be careful in using these terms: $\bar{T}_0$ means $(\mathcal{L}_{ael} - T) \cap \mathcal{L}_0$, but not $\mathcal{L}_{ael} - T_0$. There is a normal form for sentences of $\mathcal{L}_{ael}$, in which there are no nested modal operators. From now on we assume that every sentence is in this form.

From [8], the extensions of a premise set $P \subset \mathcal{L}_{ael}$ are given by:

Theorem 6 *(Konolige): A subset T of $\mathcal{L}_{ael}$ is an extension of P iff $T = \{q \in \mathcal{L}_{ael} \mid P \cup LT_0 \cup \neg L\bar{T}_0 \models_{K45} q\}$.*

$K45$ is the modal logic of weak $S5$. By this theorem, the original fixed point condition is reduced to ordinary sets. From this, we can show that a set $U \subseteq \mathcal{L}_0$ is the kernel of an extension of P if and only if it satisfies $U = \{q \in \mathcal{L}_0 \mid P \cup LU \cup \neg L\bar{U} \models q\}$, so the fixed-point condition is over first-order implication.

We now concentrate on the problem of reducing U in this fixed-point equation to a finite set. Define as a test domain:

$$\mathcal{L}_P := \{q \mid Lq \text{ occurs in a formula of } P\} \tag{15}$$

Since P is in normal form every Lq in P refers only to ordinary formulas q. Hence, $\mathcal{L}_P$ is a subset of $\mathcal{L}_0$. Furthermore, if P is finite $\mathcal{L}_P$ is also finite. We define Γ_P for a set Γ as $\Gamma_P := \Gamma \cap \mathcal{L}_P$. For subsets of the test domain we can derive a reduced fix-point form, the extension base:

Definition 6 *Let P be in normal form. A subset B of $\mathcal{L}_P$ is an extension base of P iff $B = \{q \in \mathcal{L}_P \mid P \cup LB_P \cup \neg L\bar{B}_P \models q\}$*

If T is an extension then T_P is an extension base. We also regain an extension from each extension base B: Consider $U = \{q \in \mathcal{L}_0 \mid P \cup LB_P \cup \neg L\bar{B}_P \models q\}$ which is fo-closed and satisfies $U_P = B_P$. Then $U = \{q \in \mathcal{L}_0 \mid P \cup LU_0 \cup \neg L\bar{U}_0 \models_{K45} q\}$ and U is the kernel of the extension $\{q \in \mathcal{L}_{ael} \mid P \cup LU_0 \cup \neg L\bar{U}_0 \models_{K45} q\}$.

Theorem 7 *Let P be in normal form. Then there is a bijective mapping of the set of extensions of P to the set of extension bases of P.*

As an example, consider $P = \{\neg La \supset b, \neg Lb \supset a\}$. Our test domain $\mathcal{L}_P = \{a, b\}$ has four subsets two of which are extension bases:

if $B = \emptyset$
 then $\{q \in \mathcal{L}_P \mid P \cup \{\neg La, \neg Lb\} \models q\} = \{a, b\} \neq B$
if $B = \{a\}$
 then $\{q \in \mathcal{L}_P \mid P \cup \{La, \neg Lb\} \models q\} = \{a\} = B$
if $B = \{b\}$
 then $\{q \in \mathcal{L}_P \mid P \cup \{\neg La, Lb\} \models q\} = \{b\} = B$
if $B = \{a, b\}$
 then $\{q \in \mathcal{L}_P \mid P \cup \{La, Lb\} \models q\} = \emptyset \neq B$

Finally, we briefly discuss inconsistent extension bases. Suppose a subset B of $\mathcal{L}_P$ satisfies $P \cup LB_P \cup \neg L\bar{B}_P \models \perp$. Since an inconsistency implies all formulas, B is an extension base if and only if $B = \mathcal{L}_P$. Hence, it is easy to find out whether P has an inconsistent extension: just check whether $P \cup L\mathcal{L}_P$ is inconsistent. In the sequel we consider just the consistent extensions of P.

The translation to a TMS is similar to that of DL theories. For every modal sentence $La \wedge \neg Lb_1 \cdots, \neg Lb_n \supset c$ of P, we supply a set of nonmonotonic justifications $\langle in(a), out(b_1, \cdots, b_n) \rightarrow c\rangle$. We also give a set of monotonic justifications over $\mathcal{L}_P$ taking consequents of modal sentences as input.

Definition 7 *Let P be a subset of $\mathcal{L}_{ael}$ in normal form. The TMS-translation of P is $\nu_P = (\mathcal{L}_P \cup C_P, M \cup NM)$ where $M := M_{P \cap \mathcal{L}_0}(C_P, \mathcal{L}_P)$, $NM := \{\langle in(a), out(b_1, \cdots, b_n) \rightarrow c\rangle \mid La \wedge \neg Lb_1 \cdots, \neg Lb_n \supset c \in P - \mathcal{L}_0\}$ and $C_P := \{c \mid La \wedge \neg Lb_1 \cdots \supset c \in P - \mathcal{L}_0\}$.*

Unlike in the strongly-grounded case, the translated TMS network does not behave in the same way as the extensions of P, because P may have expressions like $La \supset a$ that generate circular arguments. However, we must still avoid circular first-order proofs. So we modify the definition of extensions of the TMS. Let $\mathcal{L}_P^+ \subseteq \mathcal{L}_P$ be the set of ordinary sentences $\{a \mid La \wedge \neg Lb_1 \cdots \supset c \in P\}$, i.e., only the arguments of *negative* occurrences of modal atoms. Define a new monotonic, compact operator:

$$apply{+}_{J,Y}(X) := apply_M(X) \cup apply_{NM,Y}(X \cup (Y \cap \mathcal{L}_P^+)) \quad (16)$$

All the results for *apply* hold as well for *apply+*. We call consistent network labelings (those where $\perp$ is not believed) defined by the new operator +-extensions.

Theorem 8 *Let P be a subset of $\mathcal{L}_{ael}$ in normal form and ν_P its TMS-translation. Then there exists a bijective mapping of the set of consistent extension bases of P to the set of +-extensions of ν_P.*

We discuss some examples for autoepistemic logic:

1) If $P = \{La\}$ (that is, $\neg La \supset \perp$ in normal form) our test domain $\mathcal{L}_P = \{a\}$ consists of a single element, and $\mathcal{L}_P^+$ is empty. There are no interesting first-order proofs. The network contains one justification, $\langle out(a) \rightarrow \perp\rangle$, which leads to an inconsistency. It has no +-extension.

2) The premise set $P = \{\neg La\}$ (in normal form, $La \supset \perp$) is similar. However, we get $\mathcal{L}_P^+ = \{a\}$. There is one justification, $\langle in(a) \rightarrow \perp\rangle$, and one +-extension, $\{a\}$.

3) $P = \{\neg La \supset b, \neg Lb \supset a\}$ is a more complex theory. Here, $\mathcal{L}_P = \{a, b\}$, $\mathcal{L}_P^+ = \emptyset$. The resulting justification network ($\langle out(a) \rightarrow b\rangle$ and $\langle out(b) \rightarrow a\rangle$) contains an even loop and has two extensions $\{a\}$, $\{b\}$.

We get the same TMS network for weakly and strongly grounded extensions. The major difference between them is the inclusion of circular justifications in the definition of *apply+*. Moderately-grounded extensions have so far resisted analysis in terms of the *apply* operators, although we are still working on the problem.

Computing TMS-extensions

After translating autoepistemic and default theories into justification networks a single algorithm, with slight variations, is sufficient to compute both extensions and +-extensions. We use a complete variant of Doyle's TMS to compute all extensions of a justification network. It additionally introduces choice points and considers an OUT-label, as well as an IN-label for a node of every loop. This algorithm is also able to detect incoherence, i.e. non-existence of an extension. The algorithm is NP-complete, and we have alternatively been experimenting with stochastic constraint-satisfaction methods to achieve good average time complexity.

We use two disjoint sets, I and O, to denote all nodes that are labeled IN, or OUT, respectively. Above, we mentioned that nodes in a loop may have a chosen label which is unconfirmed and may fail in presence of odd loops. For this purpose, we introduce a third set U containing all unlabeled and unconfirmed nodes. For strongly grounded derivations, we may only use elements of $I - U$.

Similar to TMS, our algorithm uses two rules to determine an IN-label and an OUT-label of a node. The first rule looks for a justification whose in-list members have confirmed IN-labels and whose out-list members are labeled with OUT, i.e. are not contained in any solution. If there is such a justification its consequent gets a confirmed IN-label. The second rule checks whether there is still a justification which may be applied if further unlabeled nodes get an IN-label. If there is none the consequent gets a confirmed OUT-label. If the algorithm cannot proceed with propagation it chooses an arbitrary node contained in a loop and tries an IN-label, as well as an OUT-label. However, these labels may fail and therefore are unconfirmed. A failure occurs if a node is added to I and O. Finally, there is a termination rule. If the algorithm cannot find any justification whose in-list members have a confirmed IN-label then it stops and adds all remaining nodes to O. The precise algorithm is listed below:

Algorithm 1 $Ext_J(I, O, U) \equiv$

```
1. if $I \cap O \neq \emptyset$ then $\{\}$ else
2. if $U = \emptyset$ then $\{I\}$
3. if $U \neq \emptyset$ then
   (a) if $U \cap apply_{J,I}(I - U) = \emptyset$ then $Ext_J(I, O \cup U, \emptyset)$
   (b) if $\exists q \in U \cap apply_{J,N-O}(I - U)$ then $Ext_J(I \cup \{q\}, O, U - \{q\})$
   (c) if $\exists q \in U - apply_{J,I}(N - O)$ then $Ext_J(I, O \cup \{q\}, U - \{q\})$
   (d) if $\exists q \in U - (I \cup O)$ then $Ext_J(I, O \cup \{q\}, U) \cup Ext_J(I \cup \{q\}, O, U)$
```

The algorithm terminates if it is supplied with a finite set of nodes. Calling $Ext_J(\emptyset, \emptyset, N)$ results in the set of all extensions of $\nu = (N, J)$. A slight variation of the algorithm is sufficient to get $+$-extensions, too. Justification in NM may be applied to nodes in I even if their label is not confirmed. This change affects lines 3a and 3b.

Conclusion

We have developed a translation of AEL and DL to the finite constraint networks of the TMS. The advantage of doing so is that efficient constraint-satisfaction methods can be exploited in the TMS to produce the extensions of the logics. There are two areas in which we wish to develop further results. The first is in the translation process, which unfortunately is correct only for finite sets of defaults and a decidable base language. The extension to infinite sets of defaults seems attainable, but the base language complexity must remain a problem, because the TMS labelling is always decidable. Rather, we can view the TMS as an *approximation* to the original (undecidable) AEL or DL theory. The question then becomes, under what conditions are we certain to have added enough justifications to the TMS so that its extensions are extensions of the original theory?

The second area is the development of constraint-satisfaction methods for the TMS that allow the efficient computation of whether a node is in all extensions of the network or not. As mentioned above, we are currently interested in stochastic methods, in which the answer is approximate, but improves with time.

Acknowledgements

Ulrich Junker was supported by the West German Ministry for Research and Technology within project TASSO (Grant No. ITW8900A7); Kurt Konolige was supported by the Office of Naval Research Contract No. N00014-89-C-0095.

References

[1] Brewka, G., Nonmonotonic Reasoning: From Theoretical Foundation Towards Efficient Computation. Dissertation, University of Hamburg (1989)

[2] Doyle, J., A Truth Maintenance System. Artificial Intelligence 12 (1979)

[3] Doyle, J., The Ins and Outs of Reason Maintenance. Proc. IJCAI 83

[4] Etherington, D.W., Formalizing Nonmonotonic Reasoning Systems. Artificial Intelligence 31 (1987)

[5] Goodwin, J.W., A Theory and System for Non-Monotonic Reasoning. PhD Dissertation, University of Linköping 1987

[6] Junker, U., A Correct Non-Monotonic ATMS. Proc. IJCAI 89

[7] Kautz, H., Selman, B., Hard Problems for Simple Default Logics. Proc. KR'89

[8] Konolige, K., On the Relation between Default and Autoepistemic Logic. Artificial Intelligence 35 (1988)

[9] Levesque, H. J., All I Know: an Abridged Report. Proc. AAAI87.

[10] Niemelä, I., Decision Procedure for Autoepistemic Logic. Conference on Automated Deduction (1988)

[11] Moore, R.C., Semantical Considerations on Nonmonotonic Logic. Artificial Intelligence 25 (1985)

[12] Moore, R.C., Autoepistemic Logic, in: Smets, P.,Mamdani, E.H., Dubois, D., Prade, H. (eds), Non-Standard Logics for Automated Reasoning, Academic Press (1988)

[13] Reinfrank, M., Dressler, O., Brewka, G., On the Relation between Truth Maintenance and Autoepistemic Logic. Proc. IJCAI 89

[14] Reiter, R., A Logic for Default Reasoning. Artificial Intelligence 13 (1980)