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@uninet.uu.net

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 $\neg q$ [1]. Moore's procedure for AEL is restricted to a propositional language [12], just as Niemela'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 $\text{apply} : 2^U \rightarrow 2^U$ be a monotonic and compact operator:

1. If $X \subseteq Y$ then $\text{apply}(X) \subseteq \text{apply}(Y)$
2. If $q \in \text{apply}(X)$ then $\exists X \subseteq Y : X$ is finite and $q \in \text{apply}(X)$

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

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

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

\textbf{Lemma 1} Let $\text{apply} : 2^U \rightarrow 2^U$ be a monotonic and compact operator. The following conditions are equivalent:

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

Very often, we consider sets being closed w.r.t. two operators $\text{apply}_1$ and $\text{apply}_2$. For this purpose, we write $(\text{apply}_1 \cup \text{apply}_2)(X)$ for $\text{apply}_1(X) \cup \text{apply}_2(X)$.

\textbf{Definition 2' (Reiter)}: Let $\Delta = (D, W)$ be a closed default theory. A first-order set $T$ is an extension of $\Delta$ 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 $\Gamma$ 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 $\text{apply}_{D,T}$ checks $\neg b_i \notin T$ for every default $(a : b_1, \ldots, b_k/c)$ 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 $L_D := P_D \cup E_D$.

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

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

Hence, if $T$ is an extension then $T \cap L_D$ is an extension base. Conversely, if $B$ is an extension base then $\Gamma(B) = \Gamma(B) \cap L_D = \Gamma(B)$ 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 $\Delta$ to the set of extension bases of $\Delta$.

There is at most one inconsistent extension of $\Delta$. If $B$ is the extension base of an inconsistent extension then $\perp \in \Gamma(B)$ and $B = L_D$ since an inconsistency implies every formula. We can check $\perp \in \Gamma(L_D)$ to detect an inconsistent extension base.

Now, we simplify $\Gamma(B) \cap L_D$ 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 la 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:

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

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

$$\Gamma(B) \cap L_D = \left((\lambda(X). Th((X \cap C_D) \cup W) \cap L_D) \circ apply_{N,R}\right)^\ast(\emptyset) \cap L_D \quad (5)$$

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 $(\text{in}(I), \text{out}(O) \rightarrow c)$ 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:

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

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

if $X \subseteq Y$ 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. $apply_{j,Y}$:

Definition 4 Let $\nu = (N, J)$ be a justification network. $T$ is an extension of $\nu$ iff $T = apply_{j,Y}(\emptyset)$.

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 $apply_M,^\ast(X)$ which is equal to $apply_M,^\ast(X)$ for all $T$. Then we obtain

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

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 $\nu$, i.e. $T = apply_{j,Y}(\emptyset)$.

2. $apply_{j,Y}(T) \subseteq T$ and $\forall q \in T \exists q_1, \ldots, q_k : q = q_k$, and $q_i \in apply_{j,Y}((\{q_1, \ldots, q_i\})$ for $i = 1, \ldots, k$.

3. $T = (apply_M \circ apply_{NM,T,Y})^\ast(\emptyset)$

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 $U := CD$ and $R := L_D = P_D \cup E_D$.

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 $Q$).

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 $(\text{in}(Q) \rightarrow q)$. Let $M_W(U, R)$ be the set of all justifications obtained by this method:

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

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 $C_L$ (e.g., clauses with a finite herbrand universe). However, the source of the non-semidecidability of DL is the non-decidability of classical logic, not the infinite fixed points.

\footnote{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 \((a : b_1, \ldots, b_k/c)\) of \(D\) into a non-monotonic justification \((\text{in}(a), \text{out}(\neg b_1, \ldots, \neg b_k) \rightarrow c)\). This justification is applied if and only if the default is applied. We put all these justifications into a set \(NM\) and get

\[
\text{apply}_{NM,T}(x) = \text{apply}_{D,T}(x)
\]

Furthermore, we supply enough monotonic justifications for first-order derivations. As pointed out in the previous section, we take \(\text{IL} := n/\Gamma,\Delta\) and \(\text{NM} := \{(\text{in}(a), \text{out}(\neg b_1, \ldots, \neg b_k) \rightarrow c) \mid (a : b_1, \ldots, b_k/c) \in D\}\).

Now we concatenate \(\text{apply}_M\) and \(\text{apply}_{NM,T}\) and relate them to \(\Gamma\) using equation 5:

\[
(\text{apply}_M \circ \text{apply}_{NM,T})^*(\theta) = (\lambda(X).(X \cup (Th((X \cap \Gamma_D) \cup U) \land L_D)))^*(\lambda)(\theta)
\]

\[
(\text{apply}_M \circ \text{apply}_{NM,T})^*(\theta) \land L_D = \Gamma(T) \land L_D
\]

\[
\text{apply}_{NM,T} = \text{apply}_{NM,T} \cap L_D = \text{apply}_{NM,B}, \text{and } T\text{ is an extension of } \Delta.
\]

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

This is the main result. We discuss some examples.

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

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

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

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 \((\text{in}(Q) \rightarrow q)\) 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 \(\{(\text{in}(Q) \rightarrow q) \mid q \in R\}\) by a single justification \((\text{in}(Q) \rightarrow \bot)\). 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 the ories, 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 \(\mathcal{L}_{a\mathcal{E}l}\). If \(q\) is a closed formula of \(\mathcal{L}_{a\mathcal{E}l}\) the modal literal \(Lq\) is also in \(\mathcal{L}_{a\mathcal{E}l}\). We use some abbreviations from [8]. Two operators apply \(L\) and \(\neg -\) to each element of a set; \(T_0\) contains the ordinary formulae of \(T\); and \(\overline{T}\) is an extension of \(T\):

\[
L^T = \{Lq \mid q \in T\}
\]

\[
\neg L^T = \{\neg Lq \mid q \in T\}
\]

\[
T_0 = T \cap L_0
\]

\[
\overline{T} = (\mathcal{L}_{a\mathcal{E}l} \setminus T)
\]

We must be careful in using these terms: \(T_0\) means \((\mathcal{L}_{a\mathcal{E}l} \setminus T) \cap L_0\), but not \(\mathcal{L}_{a\mathcal{E}l} \setminus T_0\). There is a normal form for sentences of \(\mathcal{L}_{a\mathcal{E}l}\), 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 \(P \subseteq \mathcal{L}_{a\mathcal{E}l}\) are given by:

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

\(K45\) 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 \(U = \{q \in \mathcal{L}_0 \mid P \cup LU \cup \neg L\overline{U} \models q\}\), 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:

\[
\mathcal{L}_P := \{q \mid Lq \text{ occurs in a formula of } P\}
\]
Since $P$ is in normal form every $Lq$ in $P$ refers only to ordinary formulas $q$. Hence, $L_P$ is a subset of $L_\emptyset$. Furthermore, if $P$ is finite, $L_P$ is also finite. We define $\Gamma_P$ for a set $\Gamma$ as $\Gamma_P := \Gamma \cap L_P$. 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 $L_P$ is an extension base of $P$ iff $B = \{ q \in L_P \mid P \cup LB_P \cup \neg LB_P \models \neg q \}$

If $T$ is an extension then $T_P$ is an extension base. We also regain an extension from each extension base $B$: Consider $U = \{ q \in L_\emptyset \mid P \cup LB_P \cup \neg LB_P \models \neg q \}$ which is fo-closed and satisfies $U_P = B$. Then $U = \{ q \in L_\emptyset \mid P \cup LU_0 \cup \neg LU_0 \models \neg K_45 q \}$ and $U$ is the kernel of the extension $\{ q \in L_{act} \mid P \cup LU_0 \cup \neg LU_0 \models \neg K_45 q \}$.

**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 extensions of $P$.

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

- $B = \emptyset$
- $B = \{ a \}
- $B = \{ b \}
- $B = \{ a, b \}$

Finally, we briefly discuss inconsistent extension bases. Suppose a subset $B$ of $L_P$ satisfies $P \cup LB_P \cup \neg LB_P \models \bot$. Since an inconsistency implies all formulas, $B$ is an extension base if and only if $B = L_P$. Hence, it is easy to find out whether $P$ has an inconsistent extension: just check whether $P \cup L_P$ 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 $La \wedge \neg Lb_1 \ldots c \in P$, we supply a set of nonmonotonic justifications $\{ in(a), out(b_1, \ldots, b_n) \rightarrow c \}$. We also give a set of monotonic justifications over $L_P$ taking consequents of modal sentences as input.

**Definition 7** Let $P$ be a subset of $L_{act}$ in normal form. The TMS-translation of $P$ is $v_P = (L_P \cup C_P, M \cup NM)$ where $M := M_{\emptyset}(P, L_P)$, $NM := \{ (in(a), out(b_1, \ldots, b_n) \rightarrow c) \mid La \wedge \neg Lb_1 \ldots a \supset c \in P \}$ and $C_P := \{ c \mid Lb \wedge \neg Lb_1 \ldots a \supset c \in P \}$. 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 $L_P^+ \subseteq L_P$ be the set of ordinary sentences $\{ a \mid La \wedge \neg Lb_1 \ldots a \supset c \in P, c \in P \}$, i.e., only the arguments of negative occurrences of modal atoms. Define a new monotonic, compact operator:

\[
apply_{+,Y}(X) := apply_M(X) \cup apply_{NM,Y}(X \cup (Y \cap L_P^+))
\]

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

**Theorem 8** Let $P$ be a subset of $L_{act}$ in normal form and $v_P$ its TMS-translation. Then there exists a bijective mapping of the set of consistent extension bases of $P$ to the set of +-extensions of $v_P$.

We discuss some examples for autoepistemic logic:

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

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

- 3) $P = \{ \neg La \supset b, \neg Lb \supset a \}$ is a more complex theory. Here, $L_P = \{ a, b \}$, $L_P^+ = \emptyset$. The resulting justification network ($(out(a) \rightarrow b)$ and $(out(b) \rightarrow a)$) 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 $Ext_N(I, O, U) \equiv$
1. if $I \cap O \neq \emptyset$ then {$I$} else
2. if $U = \emptyset$ then {$I$}
3. if $U \neq \emptyset$ then
   (a) if $U \cap apply_N(I - U) = \emptyset$ then $Ext_N(I, O \cup U, \emptyset)$
   (b) if $\exists q \in U \cap apply_N(I - U)$ then $Ext_N(I \cup \{q\}, O, U - \{q\})$
   (c) if $\exists q \in U - apply_N(I - U)$ then $Ext_N(I, O \cup \{q\}, U - \{q\})$
   (d) if $\exists q \in U - (I \cup O)$ then $Ext_N(I, O \cup \{q\}, U \cup Ext_N(I \cup \{q\}, O, U)$

The algorithm terminates if it is supplied with a finite set of nodes. Calling $Ext_N(\emptyset, \emptyset, N)$ 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