Computer Science Department Technical Report
Cognitive Systems Laboratory
University of California
Los Angeles, CA 90024-1596

TRANSLATING A CYCLIC DEFAULT THEORY INTO AN ACYCLIC DEFAULT THEORY

R. Ben-Eliyahu
R. Dechter

September 1991
CSD-910063

TECHNICAL REPORT
R-169
July, 1991

Translating a Cyclic Default Theory into an Acyclic Default Theory

Rachel Ben-Eliyahu
< rachel@cs.ucla.edu >

Cognitive Systems Laboratory
Computer Science Department
University of California
Los-Angeles, California 90024

Rina Dechter
< dechter@ics.uci.edu >

Information & Computer Science
University of California
Irvine, California, 92717

July 29, 1991

Abstract

We present an algorithm that translates any cyclic propositional disjunction free default theory (PDFD) into an equivalent acyclic PDFD over the same alphabet.

1 Reiter's Default Logic

Following is a brief introduction to Reiter's default logic [Rei80].

Let $\mathcal{L}$ be a first order language. A *default theory* is a pair (D, W), where D is a set of defaults and W is a set of closed wffs (well formed formulas) in $\mathcal{L}$. A *default* is a rule of the form $\alpha : \beta_1, ..., \beta_n/\gamma$, where $\alpha, \beta_1, ...\beta_n$ and γ are formulas in $\mathcal{L}$. The intuitive meaning of a default can be : If I believe in α and I have no reason to believe that one of the β_i is false, then I can believe also in γ. A default is *semi-normal* if it is in the form $\alpha : \beta \wedge \gamma/\gamma$. A default theory is *closed* if all the first order formulas in D and W are closed.

The set of defaults D induce an extension of the set of formulas in W. Intuitively, an extension is a maximal set of formulas that can be deduced from W using the defaults in D.

Formally, let $Th(E)$ denote the logical closure of E in $\mathcal{L}$. We use the following definition of an extension:

Definition 1.1 *([Rei80],theorem 2.1) Let $E \subseteq \mathcal{L}$ be a set of closed wffs, and let (D, W) be a closed default theory.*

Define

$E_0 = W$

and for $i \geq 0$

$E_{i+1} = Th(E_i) \bigcup \{\gamma | \alpha : \beta_1, ..., \beta_n/\gamma \in D \textit{ where } \alpha \in E_i \textit{ and } \neg\beta_1, ... \neg\beta_n \notin E\}$

Then E is an extension for (D, W) iff $E = \bigcup_{i=0}^{\infty} E_i$.

(Note the appearance of E in the formula for E_{i+1}).

In this paper we restrict our attention to a subset of propositional default theories where formulas in D and W are disjunction free. We will assume that W is consistent and that no default has a contradiction as a justification. Since if W is inconsistent, only one extension exists (which is inconsistent, as Reiter shows), and a default posessing a contradictory justification can be eliminated.

We call this subclass PDFD (Propositional, Disjunction-Free Default theories).

2 Definitions and Preliminaries

In this section we present notations , definitions and lemmas that will be used throughout the paper.

We denote propositional symbols by upper case letters $P, Q, R...$, propositional literals (i.e. $P, \neg P$) by lower case letters $p, q, r...$ and conjunctions of literals by $\alpha, \beta....$ Sometimes we will regard a conjunction of literals as a set of these literals.

Given a set of formulas S, we denote by S^* the logical closure of S and call S a *logical kernel* of S^*. It is clear that when dealing with disjunction free propositional default logics, every extension E^* has a logical kernel consists only of literals.

For convinience and without loss of generality we will also assume that the consequent in each rule is a single literal.

The *dependency graph* of a PDSD (D, W), $G_{(D,W)}$, is a directed graph built as follows : Each literal p appearing in a rule in D or belonging to W is associated with a node. There is a directed edge from p to r iff there is a default in D where p appears in its prerequisite and r is its consequent.

An acyclic PDSD is one whose dependency graph is acyclic. Note that acyclicity of a directed graph can be tested in linear time , (see [Tar72]), thus yielding a test for acyclicity of PDSD wich is linear with the size of D.

We will sometimes need to identify the "strongly connected components" of the dependency graph . The *strongly-connected components* of a directed graph is a partition of its set of nodes to a maximal disjoint subsets such that for each subset C, and for each $x, y \in C$, there is a directed path from x to y and a directed path from y to x in G (see also [Eve79], section 3.4).

Tarjan ([Tar72]) showed also a linear time algorithm which identifies the strongly connected components of a graph.

We will sometime call "strongly connected components" simply components, and a directed path simply a "path".

Definition 2.1 *Let δ be a default and let E be a set of literals. We will say that E* satisfies the preconditions of *δ (precond(δ)) iff pre(δ) $\in E$ and for each $q \in$ just(δ) $\sim q \notin E$* [1]. *We will say that E* satisfies *δ iff it does not*

[1]Note that since we are dealing with PDSDs, if α is not a contradiction, the negation of one of its conjuncts is in the extension iff the negation of α is there too.

satisfy the preconditions of δ or else if it satisfies both the preconditions of δ and the conclusion of δ.

Definition 2.2 *Let (D, W) be a DF propositional default logic, E a set of propositional formulas and p a literal. A* proof of p in E *is a locally acyclic sequence of rules $\delta_1, ..., \delta_n$ such that the following conditions hold :*

- *$concl(\delta_n) = p$.*
- *for all $1 \leq i \leq n$ and for each $q \in just(\delta_i)$, $\neg q \notin E$*
- *for all $1 \leq i \leq n$ $pre(\delta_i) \subseteq W \bigcup \{concl(\delta_1), ..., concl(\delta_{i-1})\}$.*

The following lemma is instrumental in our theorems:

Lemma 2.3 *Let (D, W) be a PDSD. Then E^* is an extension of (D, W) iff E^* is a logical closure of a set of literals E that satisfy :*

1. *$W \subseteq E$*
2. *E satisfies each rule in D.*
3. *For each $p \in E$, there is a proof of p in E .*
4. *E^* is the logical closure of E .* □

3 Translating a Cyclic PDFD to an Acyclic PDFD

We will say that two default theories are *equivalent* iff every extension of one of them is an extension of the other , and vice versa. We will show now that for each cyclic PDFD there is an equivalent acyclic PDFD over the same alphabet. Thus, an alternative approach for translating a cyclic PDFD to a set of propositional formulas would be to first translate it to an acyclic PDFD and then translate its equivalent acyclic PDFD to propositional logic.

We will need the following definition :

Definition 3.1 *Let $\delta = \alpha : \beta/\gamma$ and $\delta' = \alpha' : \beta'/\gamma'$ be two defaults. We will say that δ* subsumes *δ' iff $\gamma = \gamma'$, $\alpha \subseteq \alpha'$ and $\beta \subseteq \beta'$.*

The following algorithm, we claim, translates a cyclic PDFD to an acyclic one . For the sake of simplicity, it assumes that in each cyclic rule δ , there is only one literal $p \in pre(\delta)$ such that p is on a cycle with concl(δ), and this literal appears first in the prerequisite part of the rule. The generalization is easy.

The algorithm works separately on each component of the dependency graph. If p and q are in the same components and q is in a prerequisite of a rule δ which derives p, then for each acyclic rule of q we replace q in pre(δ) with the prerequisite of that acyclic default , and we also add its justification part to the justification of δ. So we get a new acyclic rule for p, that in its turn can be used to derive more acyclic rules for other literals in the component by replacing each occurrence of p in their cyclic rule using the acyclic rule we have just derived for p. This process might be infinite unless we take care not to add a default if a rule that subsumes it is already in the set. If we do this, we reach a fixed point in $k+1$ iterations, where k is the length of the longest acyclic path in any component.

Here is an outline of the algorithm :

1. Draw the dependency graph of of $(D, W)(G_{(D,W)})$.
2. Identify the strongly connected components of $G_{(D,W)}(C_1, ..., C_m)$.
3. For each component C, compute the following until a fixed point is reached (i.e. until $R_j(C) = R_{j+1}(C)$):

Let $p_1, p_2, ..., p_n$ be all the literals in C.

For each $p_i \in C$, $\text{acyc}^0_{p_i} = \{$ all acyclic rules from D with p as a consequence $\}$.
$\text{acyc}^{j+1}_{p_i} = \text{acyc}^j_{p_i} \bigcup_* \{ q_{1_1} \wedge q_{1_2} ... \wedge q_{1_h} \wedge q_2 ... \wedge q_t : r_{1_1} \wedge r_{1_2} ... \wedge r_{1_k} \wedge r_1 \wedge r_2 ... \wedge r_l / p_i$
$\mid \exists \delta \in D,\ \delta = q_1 \wedge q_2 ... \wedge q_t : r_1 \wedge r_2 ... \wedge r_l / p_i,$
$q_1 \in C$ and $q_{1_1} \wedge q_{1_2} ... \wedge q_{1_h} : r_{1_1} \wedge r_{1_2} ... \wedge r_{1_k} / q_1 \in \text{acyc}^j_{q_1} \}$
$R_j(C) = \bigcup_{i=1...n} \text{acyc}^j_{p_i}$

The operator $\bigcup_*$ denotes a union of two sets of defaults with the additional condition that in the union there are NO two rules such that one subsumes the other. If , when computing the union , two such rules appear, only the shorter one will stay in the union. That is necessary in order to be able to reach a fixed point in a finite number of iterations.

For each component C, Let $R^*(C)$ denote the fixed point of the operator $\bigcup^*$ w.r.t. C.

4. Let $D' = \bigcup_{i=1,...m} R^*(C_i)$. (D', W) is an acyclic PDFD equivalent to (D, W) . □

Proposition 3.2 *For each component, step 3 of the algorithm will be done in at most $k+1$ iterations, where k is the length of the longest directed acyclic path in any component.*

Example 3.3 *Consider the following default theory:*
$D = t_1 : p_1/p_1, t_2 : p_2/p_2, p_1 : p_2/p_2, p_2 : p_1/p_1$
$W = \emptyset$
The strongly connected components of its dependency graph are : $C_1 = \{t_1\}$, $C_2 = \{p_1, p_2\}$, $C_3 = \{t_2\}$.
To compute D' we do the following :

$acyc^0_{t_1} = acyc^0_{t_2} = \emptyset$, *So* $R^*(C_1) = R^*(C_3) = \emptyset$
$acyc^0_{p_1} = \{t_1 : p_1/p_1\}$
$acyc^0_{p_2} = \{t_2 : p_2/p_2\}$
$R_0(C_2) = acyc^0_{p_1} \bigcup acyc^0_{p_2}$
$acyc^1_{p_1} = \{t_1 : p_1/p_1\} \bigcup \{t_2 : p_2 \wedge p_1/p_1\}$
$acyc^1_{p_2} = \{t_2 : p_2/p_2\} \bigcup \{t_1 : p_1 \wedge p_2/p_2\}$

$$
\begin{aligned}
R_1(C_2) &= acyc^1_{p_1} \bigcup acyc^1_{p_2} \\
acyc^2_{p_1} &= acyc^1_{p_1} \\
&\quad \bigcup_* \{t_2 : p_2 \wedge p_1/p_1, t_1 : p_1 \wedge p_2 \wedge p_1/p_1\} \\
&= acyc^1_{p_1} \\
acyc^2_{p_2} &= acyc^1_{p_2} \\
&\quad \bigcup_* \{t_1 : p_1 \wedge p_2/p_2, t_2 : p_2 \wedge p_1 \wedge p_2/p_2\} \\
&= acyc^1_{p_2} \\
D' = R_1(C_2) &= R_2(C_2) \\
&= \{t_1 : p_1/p_1, t_2 : p_2 \wedge p_1/p_1, t_2 : p_2/p_2, t_1 : p_1 \wedge p_2/p_2\}
\end{aligned}
$$

The main result of this section is summerized in the following theorem:

Theorem 3.4 *For every PDFD there is an equivalent* acyclic *PDFD.*

In the following proposition, Let r denote the maximum possible number of literals in the prerequisite of a rule which appear in the same component as its consequent. a - the maximum number of *acyclic* rules in D which has the same literal as a consequent, c - the maximum number of *cyclic* rules in D which has the same literal as a consequent, k - the maximal size of an acyclic path in any strongly connected component in the dependency graph of $(D, W)$and n the number of literals in (D, W)which do not appear in W.

Proposition 3.5 *Suppose $(D, W)$was transformed to (D', W')using the above algorithm. Then*

- *If $r = 1$, the algorithm will run in $O(k * n * a * c^k)$ and this will also be the order of $|D'|$.*
- *If $r > 1$ then the algorithm will run in $O(k * n * (a * c)^{r^k})$ and so will be the order of $|D'|$.*

note that $O(|D|) \leq O(n * (a + c))$ (assuming no rule has a literal from W as a consequence). Also note that step 3 can be executed in parallel for each component.

As mentioned above, the algorithm presented here assumes that $r \leq 1$, to save the reader and us tedious notations. To generalize it, when computing $acyc^{j+1}_p$, if we encounter a rule δ such that concl$(\delta) = p$ and $q_1, \ldots q_m$ are in the same component as p and appear in pre(δ), then for each combination of rules one from each $acyc^j_{q_i}$ $(i = 1 \ldots m)$ we create a new acyclic rule for p. All theorems and propositions in this subsection remain the same for the case $r > 1$.

4 Proofs

Proof of proposition 3.2

Proposition 4.1 *Suppose that $\delta \in D'$ was first introduced in $R_k(C)$ (i.e. $\delta \in R_k$, $\delta \notin R_{k-1}$) for some k, C Then, there must be a series of defaults $\delta_1, ..., \delta_k$ in D and literals $p_0, p_1, ...p_k$ such that the following hold :*

- *$concl(\delta_i) = p_i$, $\delta_k = \delta$.*
- *$p_0, ...p_k$ are in the same strongly connected component.*
- *$concl(\delta_i) \in pre(\delta_{i+1})$.*
- *If $i \neq j$ then $p_i \neq p_j$.*

Proof: By induction on k. All the p_i's are different because we do not allow in R_j a rule which has already a rule that subsumes it in R_{j-1}. □

Suppose that in the $k+1$'s iteration a new default was introduced to D'. Then, by proposition 4.1, there must be a path of length $k+1$ in some component. A contradiction. □

Proof of theorem 3.4 The following proposition is quite obvious:

Proposition 4.2 *If q and p are in the same component, no literal in α is in the same component with p, $q \wedge \alpha : \beta/p$ is in D and $\alpha' : \beta'/q$ is a (acyclic) rule in D', then a default that subsumes $\alpha' \wedge \alpha : \beta \wedge \beta'/p$ is in D'.*

Let (D, W)be an arbitrary PDFD, and let (D', W)be the output of the above algorithm. We will show that $(D, W)$and $(D', W)$are equivalent.

- Let E^*be an extension of (D, W). We will show that $E^*$is also an extension of (D', W). Let E be the logical kernel of E^* which satisfies the conditions of lemma 2.3. We will show that E satisfies the conditions of lemma 2.3 also with respect to (D', W). Condition 1 is clearly satisfied.

For condition 2 , suppose that for an arbitrary $\delta \in D'$, E satisfies precond(δ). We want to show that concl(δ) $\in E$. Suppose that concl(δ) $\in C$ for some component C of $G_{(D,W)}$. The proof is by induction on the lowest i such that $\delta \in R_i(C)$. If $i = 0$, then the assertion clearly holds since δ belongs also to D. Now, suppose that $\delta = q_1 \wedge ...q_n : r_1 \wedge ...r_m/p$ appears for the first time in $R_{i+1}(C)$,$\delta \notin D$, $q_1, ..., q_n$ are in E and $\sim r_1, ..., \sim r_m$ are not in E. So there must be a default $\delta_i = q_1 \wedge ... \wedge q_k : r_1 \wedge ... \wedge r_l/q$ for some $k \leq n$, $l \leq m$ in $R_i(C)$ and a default $\delta_D = q \wedge q_{k+1} ... \wedge q_n : r_{l+1} \wedge ... \wedge r_m/p$ in D, where $q \in C$. Since the preconditions of δ_i are satisfied by E, by the induction hypothesis $q \in E$, So the preconditions of δ_D are in E, and since δ_D is in D, p is in E as well.

For condition 3, Let $p \in E$, we want to show a proof of p in E with respect to the theory (D', W). By induction on the length n of a minimal proof of p in E w.r.t. (D, W): If $n = 0$, then $p \in W$.
Let $\delta_1, ..., \delta_{n+1} = q_1 \wedge ... \wedge q_n : r_1 \wedge ... \wedge r_m/p$ be a proof of p in E using rules from D. If δ_{n+1} is acyclic, we are done using the induction hypothesis. If δ_{n+1} is cyclic, then suppose WLG that q_1 is on the same component as p. Using the induction hypothesis, let $\delta'_1, ..., \delta'_k = \alpha : \beta/q_1$ be a proof of q_1 in E using rules from D'. By proposition 4.2 above then, there must be a default $\delta \in D'$ such that δ subsumes $\alpha \wedge q_2 ... \wedge q_n : \beta \wedge r_1 \wedge ... \wedge r_m/p$. From here it is easy to see how we construct a proof of p in E using rules from D'.

- Let E^* be an extension of (D', W). We will show that E^* is also an extension of (D, W). Let E be the logical kernel of E^* which satisfies the conditions of lemma 2.3. We will show that E satisfies the conditions of lemma 2.3 also with respect to (D, W). Condition 1 is clearly satisfied.

  For condition 2 , suppose that for an arbitrary $\delta \in D$, E satisfies precond(δ). We want to show that concl(δ) $\in E$. If δ is acyclic, then it belongs to D', and we are done. Suppose then that $\delta = q \wedge \alpha : \beta/p$ for some q which is on the same component as p. Since $q \in E$, there is a proof $\delta_1, ...\delta_n = \alpha' : \beta'/q$ of q in E using rules from D'. By proposition 4.2 above then, a rule δ' which subsumes the default $\alpha' \wedge \alpha : \beta' \wedge \beta/p$ is in D'. Since the preconditions of δ' are satisfied, p must be in E.

For condition 3, Let $p \in E$, we want to show a proof of p in E with respect to the theory (D, W). We will need the following proposition:

Proposition 4.3 *For each default $\delta \in D'$ there is a series of defaults $s(\delta) = \delta_1, ..., \delta_n$ in D such that :*

- *for each $1 \leq i \leq n$ $pre(\delta_i) \subseteq \bigcup_{1 \leq j < i} concl(\delta_j) \bigcup W \bigcup pre(\delta)$.*
- *$concl(\delta_n) = concl(\delta)$.*
- *For each $1 \leq i \leq n$, $just(\delta_i) \subseteq just(\delta)$.*

Proof: Let $\delta \in D'$ be a default. We want to show a series in D that have the above properties. We will do it by induction on the minimal j such that $\delta \in R_j$. If $j = 0$, then δ itself is in D. Suppose $\delta = q_1 \wedge ... \wedge q_m : r_1 \wedge ... \wedge r_k/p$ appears for the first time in R_{j+1}. So there must be a default $\delta_j = q_1 \wedge ... \wedge q_l : r_1 \wedge ... \wedge r_h/q$ for some $l \leq m$ and $h \leq k$ in R_j, and default $\delta_D = q \wedge q_{l+1} \wedge ... \wedge q_m : r_{h+1} \wedge ... \wedge r_k/p$ in D. By the induction hypothesis, δ_j can be replaced by series of rules $s(\delta)$ having the properties above. Clearly, the series $< s(\delta), \delta_D >$ will have the required properties w.r.t. δ. □

We will find the proof for p wrt D using induction on the length n of the proof of p in E w.r.t. (D', W): If $n = 0$, then $p \in W$.

Let $\delta_1, ..., \delta_{n+1}$ be a proof of p in E using rules from D'. By the induction hypothesis, for each $q \in$ pre(δ_{n+1}) there is a proof in E using rules from D. Those proofs followed by $s(\delta_{n+1})$ are a proof of p in E using rules from D.

□

Proof of proposition 3.5

s will denote the maximal size a strongly connected component in the dependency graph of (D, W).

d will denote the number of components in the dependency graph of (D, W).

Also, denote by a_j the upper bound of $|\mathrm{acyc}_p^j|$, for any p.

Clearly, if $r = 0$ then $D' = D$ ((D, W)itself is acyclic), so we assume $r \geq 1$.

We get :

$a_0 \leq a$
$a_{j+1} \leq {a_j}^r * c$
OR :
$a_0 \leq a$
$a_1 \leq a^r * c$
$a_2 \leq (a^r * c)^r * c = a^{r^2} * c^{r+1}$
$a_3 \leq (a^{r^2} * c^{r+1})^r * c = a^{r^3} * c^{r^2+r+1}$
.
.
.
So if $r = 1$, $a_k \leq a * c^k$.
If $r > 1$, since $1 + r + r^2 + ... + r^{k-1} = r^k - 1/r - 1 \leq r^k$),
$a_k \leq a^{r^k} * c^{r^k} = (a * c)^{r^k}$
We get that for each component:
$|R_i| \leq |R_{i-1}| + s * a_i \leq s * a_i * (i + 1)$
and
$|D'| \leq d * R_k \leq d * s * a_k * (k + 1)$
Since a better aproximation to $d * s$ is n, we get:
$|D'| \leq n * a * c^k * (k + 1)$ if $r = 1$
$|D'| \leq n * (a * c)^{r^k}(k + 1)$ if $r > 1$

□

References

[Eve79] Shimon Even. *Graph Algorithms.* Computer Science Press, 1979.

[Rei80] Ray Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal of Computing*, 1(2), June 1972.