

ON CYCLIC COVERS AND PERFECT MODELS

C. A. JOHNSON

Computer Science Department
University of Keele
Staffordshire, ST5 5BG
England
email : chrisj@cs.keele.ac.uk

Abstract. The relationship between cyclic covers and perfect models of indefinite stratified deductive databases is discussed with respect to query answering, view updates and partial evaluation.

The notion of a conjunctive answer to a database query is introduced, and such answers are shown to have a number of advantages over (the more commonly studied) “disjunctive” answers. The problem of computing conjunctive answers is shown to be a special case of the view update problem.

Cyclic covers are shown to characterise perfect models, conjunctive answers and view updates, and using this characterisation a method is presented for solving the view update problem, and hence for computing conjunctive answers. In addition, cyclic covers are shown to yield methods of partially evaluating (ie., pre-processing) the intensional database, for instance to facilitate query compilation, the removal of recursion and the removal of positive or negative intensional sub-goals. Such pre-processing is immune to future changes in the extensional database.

Keywords: Indefinite data, stratified deductive databases, cyclic trees, cyclic covers, query answering, conjunctive answers, view updates, query compilation, partial deduction, perfect models.

INTRODUCTION.

A deductive database allows the representation of data and data relationships using logical formulae of the form

$$A_1(\mathbf{x}_1) \wedge \dots \wedge A_h(\mathbf{x}_h) \wedge \neg A_{h+1}(\mathbf{x}_{h+1}) \wedge \dots \wedge \neg A_{h+r}(\mathbf{x}_{h+r}) \rightarrow B_1(\mathbf{y}_1) \vee \dots \vee B_k(\mathbf{y}_k)$$

where each $A_i(\mathbf{x}_i), B_j(\mathbf{y}_j)$ is a first order (positive) atom.

Given such a representation, the meaning of the information in the database is given by the *declarative semantics*. In this paper we restrict our attention to stratified (indefinite) databases, and interpret these using (what is often regarded as) the most natural semantics defined by perfect models, thus a statement Φ is true in a database

T iff it is true in every perfect model of T , and we write $T \models \Phi$. We examine the relationship between perfect models and cyclic covers with respect to query answering, view updates and partial evaluation.

A query against a database T is typically an expression of the form $?Q(\mathbf{x})$, representing a request for information about those instances (or combinations of instances) of $Q(\mathbf{x})$ that are true in T . In indefinite databases it is normal to consider an answer to be a set \mathcal{D} of ground instances of $Q(\mathbf{x})$ such that $T \models \bigvee \mathcal{D}$, and we refer to such answers as *disjunctive answers*.

The problem with disjunctive answers is twofold. Firstly, if $\mathcal{D}' \subset \mathcal{D}$ are both disjunctive answers to the query $?Q(\mathbf{x})$, then the pair of answers $\mathcal{D}', \mathcal{D}$ gives us no new information about the query beyond that which is provided by the subsuming answer \mathcal{D}' , ie., the non-minimal answer is completely redundant. Secondly, minimal disjunctive answers are very difficult to compute in an efficient manner, the only known strategy being to generate disjunctive answers, and then remove redundant answers by subsumption. As indicated above, the computation employed in producing a non-minimal answer is completely wasted.

In this paper we address these issues by considering an alternative form of answer which we refer to as *conjunctive answers*. Specifically a set \mathcal{A} of ground instances of $Q(\mathbf{x})$ is a *conjunctive answer* to the query $?Q(\mathbf{x})$ iff there is some perfect model M of the database such that \mathcal{A} is the set of (ground) instances of $Q(\mathbf{x})$ that hold in M . Intuitively, the perfect models of T represent the “possible interpretations” of T , and \mathcal{A} is thus a conjunctive answer if it is one of those sets of ground instances of $Q(\mathbf{x})$ which could occur (in some such possible interpretation).

We argue (Section 2) that non-minimal conjunctive answers are not redundant, in that if $\mathcal{A}' \subset \mathcal{A}$ are both conjunctive answers, then \mathcal{A} provides information about the query that is *not* provided by \mathcal{A}' . We also show that conjunctive answers provide more information about a query than do their disjunctive counterparts. Specifically the set of disjunctive answers can be readily computed from the set of conjunctive answers, whereas the converse is not the case. Our arguments in Section 2 are not restricted to perfect models, and could equally well be made for other model-theoretic semantics.

The view update problem is concerned with indirectly modifying a view (ie., a derived or intensional predicate) by an appropriate modification of the extensional database. We show, somewhat surprisingly, that the problem of computing conjunctive answers can be readily reduced to the problem of inserting tuples into a view.

In Section 3 we review the basic definitions and properties of cyclic trees and cyclic covers. Cyclic trees were introduced in [Jo96] as a means of testing minimal and perfect model membership, and cyclic covers were introduced in [Jo95] as a means of processing queries under the perfect model semantics. We show that cyclic covers characterise perfect models, conjunctive answers and the view update problem. Methods for constructing cyclic covers are presented in [Jo95], and we illustrate an adaptation of these methods for performing view insertions (and hence for computing conjunctive answers).

Query compilation involves a pre-processing of the intensional database, so that query processing subsequently requires a manipulation of the extensional database only.

Partial evaluation involves a partial pre-processing of the (intensional) database so that query processing is simplified in some manner (and in this sense, compilation can be regarded as a query specific form of partial evaluation). The relationship between cyclic covers and various forms of intensional pre-processing is discussed in Section 4. In particular, we show that cyclic covers yield methods of compiling queries, identifying redundant rules, eliminating recursion and eliminating positive and/or negative intensional sub-goals from the intensional database. The general structure of a deductive database is generally expected to be a large transient extension, and a relatively smaller, static intension. We therefore insist that our pre-processing does not employ the extension and in particular is immune to future changes in the extensional database.

For the most part we consider only the ground level (without variables), although in Section 5 we discuss the problems involved in lifting our results and methods to the first order level. Section 5 also contains some concluding remarks. Section 6 compares the results of the current paper with other related work from the literature.

§1. TERMINOLOGY.

1.1 Language and rules.

Throughout \mathcal{L} will denote a finite propositional language $\mathcal{L} = \{P_1, P_2, \dots, P_n\}$. A *positive literal* is a predicate, and a *negative literal* is the negation of a predicate. We assume the existence of a level function $\ell : \{P_1, P_2, \dots, P_n\} \rightarrow \{0, 1, 2, \dots, n\}$. For each predicate P we define $\ell(\neg P) = \ell(P)$. If \mathcal{P} is a set of literals and $\alpha \leq n$, then we define $\mathcal{P}_\alpha = \{K \in \mathcal{P} \mid \ell(K) = \alpha\}$, $\mathcal{P}_{\leq \alpha} = \{K \in \mathcal{P} \mid \ell(K) \leq \alpha\}$ and $\mathcal{P}_{< \alpha} = \{K \in \mathcal{P} \mid \ell(K) < \alpha\}$, etc. $\text{EXT}(\mathcal{L}) = \mathcal{L}_0$ denotes the set of *extensional* predicates, and $\text{INT}(\mathcal{L}) = \mathcal{L}_{>0}$ is the set of *intensional* predicates. If \mathcal{C} is a set of literals, then $\text{EXT}(\mathcal{C})$ denotes the set of extensional positive literals in \mathcal{C} , ie., $\text{EXT}(\mathcal{C}) = \text{EXT}(\mathcal{L}) \cap \mathcal{C}$.

Stratification requires that recursion cannot proceed via negation, this requirement being expressed in the following constraints on database rules.

1.1.1 Definition. A *rule* C is a formula of the form

$$A_1 \wedge A_2 \wedge \dots \wedge A_h \wedge \neg A_{h+1} \wedge \neg A_{h+2} \wedge \dots \wedge \neg A_{h+r} \rightarrow B_1 \vee B_2 \vee \dots \vee B_k$$

such that:

- (i) each A_i and each B_j is a predicate in \mathcal{L} and $k > 0$,
- (ii) for each $j \leq k$, $\ell(B_j) = \ell(B_1)$,
- (iii) for each $i \leq h$, $\ell(A_i) \leq \ell(B_1)$,
- (iv) for $1 \leq i \leq r$, $\ell(A_{h+i}) < \ell(B_1)$,
- (v) $h = r = 0$ iff $\ell(B_1) = 0$.

We define $\text{antec}(C) = \{A_1, A_2, \dots, A_h\}$, $\mathcal{N}(C) = \{A_{h+1}, A_{h+2}, \dots, A_{h+r}\}$, $\overline{\mathcal{N}}(C) = \{\neg A_{h+1}, \neg A_{h+2}, \dots, \neg A_{h+r}\}$ and $\text{conseq}(C) = \{B_1, B_2, \dots, B_k\}$. Let $\ell(C) = \ell(B_1)$.

A *deductive database* is a set of rules, which throughout will be denoted by T . T is said to be *positive* iff $\mathcal{N}(C) = \emptyset$ for each $C \in T$.

We also define $T_\alpha = \{C \in T \mid \ell(C) = \alpha\}$, etc. The *extension* of T is given by $\text{EXT}(T) = T_0$ and the *intension* by $\text{INT}(T) = T_{>0}$. If $C = B_1 \vee B_2 \vee \dots \vee B_k \in \text{EXT}(T)$, then we may confuse C with the set of its constituent predicates $C = \{B_1, B_2, \dots, B_k\}$.

1.1.2 Definition. Let C be a rule and $M \subseteq \mathcal{L}$. Then M *models* C (written $M \models C$) iff $\text{antec}(C) \subseteq M \implies M \cap (\text{conseq}(C) \cup \mathcal{N}(C)) \neq \emptyset$.

M *models* T (written $M \models T$) iff $M \models C$ for each $C \in T$.

1.1.3 Definition [Pr88]. A model M of T is *perfect* iff whenever $\alpha \leq n (= |\mathcal{L}|)$ and $M' \subset M_\alpha$, then $M_{<\alpha} \cup M' \not\models T_\alpha$.

Przymusiński's original definition of perfectness used the notion of preferability, although for the purposes of the current paper, the above is all that is needed. Thus a perfect model M is formed by iteratively minimizing the predicates in the model, and thus can be viewed as a stratified form of minimalism. Specifically M_0 is a minimal model of T_0 , and given $M_{<\alpha}$, M_α is a minimal subset of \mathcal{L}_α such that $M_{\leq\alpha} \models T_\alpha$. It is easy to show [Pr88] that every perfect model is minimal, and if T is positive, then the converse holds.

1.1.4 Theorem. Let $M \subseteq \mathcal{L}$, then the following are equivalent.

- (i) M is a perfect model of T ,
- (ii) for each $\alpha \leq n$, $M_{\leq\alpha}$ is a perfect model of $T_{\leq\alpha}$,
- (iii) for each $\alpha \leq n$, $M_{\leq\alpha}$ is a minimal model of $T_{\leq\alpha}$.

1.1.5 Definition. If Φ is a formula in \mathcal{L} , then $T \models \Phi$ iff Φ is true in every perfect model of T .

Two databases are regarded as equivalent iff they have the same perfect models.

1.1.6 Theorem. If $T \subseteq T'$, then T and T' are equivalent iff each perfect model of T is a model of T' , ie., $T \models T'$.

Proof (\leftarrow). Let M be a perfect model of T , then by hypothesis, M is a model of T' . If M is not a perfect model of T' , then we may find an $\alpha \leq n$ and an $M' \subset M_\alpha$ such that $M_{<\alpha} \cup M' \models T'_\alpha$. But then $M_{<\alpha} \cup M' \models T_\alpha$, contradicting the fact that M is a perfect model of T .

If M is a perfect model of T' , then M is a model of T . If M is not a perfect model of T , then we may find an $\alpha \leq n$, an $M' \subset M_\alpha$ and an $M^* \subseteq \mathcal{L}_{>\alpha}$ such that $M_{<\alpha} \cup M' \cup M^*$ is a perfect model of T . But then $M_{<\alpha} \cup M' \cup M^*$ is a model of T' , whence $M_{<\alpha} \cup M' \models T'_\alpha$, contradicting the fact that M is a perfect model of T' .

(\rightarrow). This direction is trivial.

§2. QUERY ANSWERS.

In this section we introduce the concept of a *conjunctive answer* to a database query. We show that all such answers (including those that are non-minimal) provide useful information about the query, and that disjunctive answers are readily computed from the set of conjunctive answers (whilst the converse is not the case). We also attempt to give informal criteria indicating when it might be computationally preferable to compute conjunctive rather than disjunctive answers. Finally we show that the problem of computing conjunctive answers can be reduced to the problem of performing view insertions.

2.1 Conjunctive answers.

2.1.1 Definition (a). A query is an expression of the form $?Q$, where $Q \subseteq \mathcal{L}$.

(b). A *disjunctive answer* of $?Q$ is a set $\mathcal{D} \subseteq Q$ such that $T \models \bigvee \mathcal{D}$. A disjunctive answer is said to be *minimal* iff it properly contains no other disjunctive answer.

(c). A set $\mathcal{A} \subseteq Q$ is a *conjunctive answer* of $?Q$ iff there is a perfect model M of T such that $M \cap Q = \mathcal{A}$. A conjunctive answer is *minimal* iff it properly contains no other conjunctive answer.

Note (1). $T \models \bigvee Q$ iff each conjunctive answer is non-empty. Moreover if $T \not\models \bigvee Q$, then computing disjunctive answers tells us nothing about Q (beyond the fact that $T \not\models \bigvee Q$).

(2). The above definition is consistent with the first order level, since a first order query $?Q(\mathbf{x})$ can be regarded as $? \{Q(\mathbf{a}) \mid \mathbf{a} \text{ is a ground instance of } \mathbf{x}\}$.

(3). We will see in Section 3 that in order to compute the set of conjunctive answers, we do not need to compute perfect models in their entirety. This is important since perfect models would typically be large and numerous. Moreover many perfect models might well yield the same conjunctive answer. Theorem 3.3.1 indicates that for each conjunctive answer \mathcal{A} , we need to generate a cyclic cover (see Definitions 3.2.1/2) which witnesses that \mathcal{A} is the intersection of Q with some perfect model.

We first illustrate that *every* (possibly non-minimal) conjunctive answer can be regarded as providing useful information about the query $?Q$.

2.1.2 Example. Suppose that T consists of the following rules:

$$\begin{array}{llll} P \vee R \vee S & P \rightarrow Q_3 & P \rightarrow Q_1 \vee Q_2 & Q_1 \rightarrow Q_2 \\ S \rightarrow Q_1 & R \rightarrow Q_1 & R \rightarrow Q_2 & R \rightarrow Q_3 \end{array}$$

and we have the query $?Q$, where $Q = \{Q_1, Q_2, Q_3\}$.

Clearly the perfect/minimal models of T are $\{P, Q_2, Q_3\}$, $\{S, Q_1, Q_2\}$, and $\{R, Q_1, Q_2, Q_3\}$. Hence the minimal disjunctive answers of $?Q$ are $\{Q_2\}$ and $\{Q_1, Q_3\}$. The conjunctive answers are $\{Q_2, Q_3\}$, $\{Q_1, Q_2\}$ and $\{Q_1, Q_2, Q_3\}$.

What information about the query do these answers give us? The fact that $\{Q_2\}$ is a disjunctive answer tells us that Q_2 is true in all perfect models, and the fact that

$\{Q_1, Q_3\}$ is a *minimal* disjunctive answer tells us that $Q_1 \wedge \neg Q_3$ is true in some perfect model, as is $\neg Q_1 \wedge Q_3$.

Neither the disjunctive answers, nor the conjunctive answers of size 2 (ie., $\{Q_2, Q_3\}$ and $\{Q_1, Q_2\}$) tell us whether (or not) there is a perfect model in which $Q_1 \wedge Q_2 \wedge Q_3$ is true (ie., whether $Q_1 \wedge Q_2 \wedge Q_3$ *might* be true).

This information is obviously given by the last of our conjunctive answers, ie., $\{Q_1, Q_2, Q_3\}$, thus showing that this non-minimal answer is not redundant as regards providing useful information about $?Q$.

2.1.3 Example. Suppose that T consists of the following rules:

$$P \vee S \qquad P \rightarrow Q_3 \qquad P \rightarrow Q_1 \vee Q_2 \qquad Q_1 \rightarrow Q_2 \qquad S \rightarrow Q_1$$

and we have the query $?Q$, where $Q = \{Q_1, Q_2, Q_3\}$. Again the minimal disjunctive answers of $?Q$ are $\{Q_2\}$ and $\{Q_1, Q_3\}$, whereas in this case the conjunctive answers are $\{Q_2, Q_3\}$ and $\{Q_1, Q_2\}$ only.

The following theorems show that disjunctive answers can be computed from conjunctive answers. As illustrated in the above examples, the converse is not the case, thus conjunctive answers provide more information about a query than do their disjunctive counterparts.

2.1.4 Theorem. Let $?Q$ and $\mathcal{D} \subseteq Q$ be given, then the following are equivalent.

- (a) \mathcal{D} is a disjunctive answer,
- (b) \mathcal{D} intersects every conjunctive answer,
- (c) \mathcal{D} intersects every minimal conjunctive answer.

Proof (a) \rightarrow (b). Suppose that $\mathcal{D} \subseteq Q$ is a disjunctive answer, and \mathcal{A} is a conjunctive answer with associated perfect model M . Then $\mathcal{D} \cap \mathcal{A} = \mathcal{D} \cap M \cap Q = \mathcal{D} \cap M \neq \emptyset$ (since $\mathcal{D} \subseteq Q$ and $M \models \bigvee \mathcal{D}$).

(b) \rightarrow (c) is trivial.

(c) \rightarrow (a). Suppose that M is a perfect model of T . Let \mathcal{A} be a minimal conjunctive answer such that $\mathcal{A} \subseteq M \cap Q$, then since $\mathcal{A} \cap \mathcal{D} \neq \emptyset$, we must have that $M \cap \mathcal{D} \neq \emptyset$. ■

2.1.5 Corollary. Let $\mathcal{D} \subseteq Q$ be a disjunctive answer to the query $?Q$, then the following are equivalent.

- (a) \mathcal{D} is minimal,
- (b) for each $Q \in \mathcal{D}$ there is a minimal conjunctive answer \mathcal{A} such that $\mathcal{D} \cap \mathcal{A} = \{Q\}$,
- (c) for each $Q \in \mathcal{D}$ there is a conjunctive answer \mathcal{A} such that $\mathcal{D} \cap \mathcal{A} = \{Q\}$.

Proof. Firstly note that \mathcal{D} is minimal iff for each $Q \in \mathcal{D}$ there is a perfect model M such that $M \cap \mathcal{D} = \{Q\}$.

(a) \rightarrow (b). Suppose that \mathcal{D} is minimal, $Q \in \mathcal{D}$, and there is no minimal conjunctive answer \mathcal{A} for which $\mathcal{D} \cap \mathcal{A} = \{Q\}$. Let \mathcal{A} be a minimal conjunctive answer, then by

Theorem 2.1.4, $\emptyset \neq \mathcal{D} \cap \mathcal{A} \neq \{Q\}$, thus $(\mathcal{D} - \{Q\}) \cap \mathcal{A} \neq \emptyset$. Since $\mathcal{D} - \{Q\}$ intersects every minimal conjunctive answer, $\mathcal{D} - \{Q\}$ is (by Theorem 2.1.4) a disjunctive answer, thus contradicting the minimality of \mathcal{D} .

(b) \rightarrow (c) is trivial.

(c) \rightarrow (a). Suppose that $\mathcal{D}' \subset \mathcal{D}$ and $Q \in \mathcal{D} - \mathcal{D}'$. Let \mathcal{A} be a conjunctive answer such that $\mathcal{D} \cap \mathcal{A} = \{Q\}$, then $\mathcal{D}' \cap \mathcal{A} = \emptyset$, whence by Theorem 2.1.4, $T \not\models \bigvee \mathcal{D}'$. ■

2.1.6 Corollary. Given $?Q$ and $Q_0 \in \mathcal{Q}$, then Q_0 belongs to some minimal disjunctive answer of $?Q$ iff Q_0 belongs to some minimal conjunctive answer of $?Q$.

Proof (\leftarrow). Let \mathcal{A} be a minimal conjunctive answer with $Q_0 \in \mathcal{A}$, and let M_0 be the associated perfect model. Let $(M_i \mid 1 \leq i \leq m)$ list all other perfect models of T , then for each $i > 0$ we have $M_i \cap \mathcal{Q} \not\subseteq \mathcal{A}$, by the minimality of \mathcal{A} . For each $i > 0$, pick $Q_i \in M_i \cap \mathcal{Q}$ such that $Q_i = Q_0$ if $M_i \cap \mathcal{Q} = \mathcal{A}$; $Q_i \in M_i \cap \mathcal{Q} - \mathcal{A}$ otherwise.

Clearly $T \models \bigvee_{i=0}^m Q_i$. If \mathcal{D} is a minimal disjunctive answer such that $\mathcal{D} \subseteq \{Q_i \mid 0 \leq i \leq m\}$, then since $M_0 \models \bigvee \mathcal{D}$, we have $\emptyset \neq M_0 \cap \mathcal{D} \subseteq M_0 \cap \{Q_i \mid 0 \leq i \leq m\} = \{Q_0\}$, whence $Q_0 \in \mathcal{D}$.

(\rightarrow). This follows trivially from the preceding corollary. ■

It is well known that minimal (and perfect) model membership can be characterised syntactically, ie., that Q_0 belongs to some perfect model of T iff there is a disjunct $Q_0 \vee A_1 \vee A_2 \vee \dots \vee A_n$ such that $T \models Q_0 \vee A_1 \vee A_2 \vee \dots \vee A_n$ but $T \not\models \bigvee \mathcal{D}$ for any $\mathcal{D} \subset \{Q_0, A_1, A_2, \dots, A_n\}$. (In the terminology of Definition 2.1.1, $\{Q_0, A_1, A_2, \dots, A_n\}$ is a minimal disjunctive answer of the query $?L$.) However, perfect model membership is *not* sufficient to guarantee membership of a minimal disjunctive answer *of the query* $?Q$. For example if $T = \{Q_0 \vee Q, R \vee S\}$ and $\mathcal{Q} = \{Q_0, R, S\}$, then Q_0 belongs to some minimal model of T , and hence to some conjunctive answer (eg., $\{Q_0, R\}$), but Q_0 does not belong to any minimal disjunctive answer of $?Q$. By the above corollary, this is because Q_0 does not belong to a minimal conjunctive answer of $?Q$.

2.2 Conjunctive vs. disjunctive answers.

When might it be more appropriate to compute conjunctive rather than disjunctive answers? We have already made the point that conjunctive answers do provide useful information about a query, and so from a logical perspective we might argue that they should always be computed. What, however if we look at the issue from computational grounds? Consider the following examples.

2.2.1 Example. Suppose that $T = \{Q_{2i} \vee Q_{2i+1} \mid 0 \leq i \leq n\}$, and the query under consideration is $?Q$ where $\mathcal{Q} = \{Q_i \mid 0 \leq i \leq 2n+1\}$.

Clearly the minimal disjunctive answers to $?Q$ are those disjuncts to be found in T . On the other hand there are 2^{n+1} conjunctive answers, and clearly computing all of these would be impossible for large n .

2.2.2 Example. Suppose that $T = \{\bigvee_{i=0}^n Q_{3i}\} \cup \{Q_{3i} \rightarrow Q_{3i+1} \vee Q_{3i+2} \mid 0 \leq i \leq n\}$, then the conjunctive answers to the query $? \{Q_i \mid 0 \leq i \leq 3n+2\}$ are of the form $\{Q_{3i}, Q_{3i+k}\}$ where $i \leq n, k \in \{1, 2\}$.

There are however 2^{n+1} minimal disjunctive answers, these being of the form

$$\{Q_{3i} \mid i \in I\} \cup \{Q_{3i+k} \mid i \notin I, k \in \{1, 2\}\}$$

for any $I \subseteq \{0, 1, 2, \dots, n\}$.

Needless to say, we can produce examples in which both forms of answer suffer from such computational problems.

2.2.3 Example. Suppose that T is the database

$$\{Q_{2i} \vee Q_{2i+1} \mid 0 \leq i \leq n\} \cup \{Q_0 \rightarrow \bigvee_{j=0}^m P_{3j}\} \cup \{P_{3j} \rightarrow P_{3j+1} \vee P_{3j+2} \mid 0 \leq j \leq m\}$$

then the minimal disjunctive answers to the query $? \{Q_i \mid 0 \leq i \leq 2n+1\} \cup \{P_j \mid 0 \leq j \leq 3m+2\}$ are of the form

$$\{Q_{2i}, Q_{2i+1}\}$$

for $0 \leq i \leq n$, and

$$\{Q_1\} \cup \{P_{3j} \mid j \in I\} \cup \{P_{3j+k} \mid j \notin I, k \in \{1, 2\}\}$$

for any $I \subseteq \{0, 1, 2, \dots, m\}$. The conjunctive answers are of the form

$$\{Q_1\} \cup \{Q_{2i+g(i)} \mid 1 \leq i \leq n\}$$

and

$$\{Q_0\} \cup \{Q_{2i+g(i)} \mid 1 \leq i \leq n\} \cup \{P_{3h}, P_{3h+k}\}$$

for any $g : \{1, 2, \dots, n\} \rightarrow \{0, 1\}$, $0 \leq h \leq m$ and $k \in \{1, 2\}$.

What general guidelines can we come to? Needless to say, conjunctive answers are computationally preferable when there are substantially fewer conjunctive answers than disjunctive answers. The above examples suggest that this will be the case when there are strong interdependencies between the elements of \mathcal{Q} within perfect models.

For example, in Example 2.2.1, the only dependencies which hold within perfect/minimal models are $Q_{2i} \leftrightarrow \neg Q_{2i+1}$, whereas in Example 2.2.2, the dependencies which hold are $Q_{3i} \rightarrow \neg Q_{3j}$ ($j \neq i$), $Q_{3i} \leftrightarrow Q_{3i+1} \vee Q_{3i+2}$, and $Q_{3i} \rightarrow (Q_{3i+1} \leftrightarrow \neg Q_{3i+2})$.

In practice it seems unlikely that we would be able to compute a measure of the strength of such interdependencies. However, the possibility of using some higher level

meta-knowledge as a guide seems a reasonable possibility. The following example illustrates the point.

2.2.4 Example. Suppose that we have a database containing spatial information about points, line segments, etc. Locations in the plane are represented by a pair of integers (x, y) , each lying in some given (large) range. $start(l, x, y)$ gives the (x, y) co-ordinates of the start of the line segment l , and the end-point is given by $end(l, x, y)$. The derived relation $on(l, x, y)$ indicates that the point (x, y) lies on l (perhaps to within some small tolerance). Suppose that our database contains the facts

$$start(l_1, 0, 0) \vee start(l_1, 50000, 0) \text{ and } end(l_1, 0, 100000)$$

and also suppose that we have a number of points of interest p_i, q_j , together with their locations

$$loc(p_1, 0, 1000), loc(p_2, 0, 34543), loc(p_3, 0, 78988), loc(p_4, 0, 98080), \\ loc(q_1, 40000, 20000), loc(q_2, 25000, 50000), loc(q_3, 10000, 80000).$$

Suppose that we ask which of these points lies on l_1 , then the query

$$?loc(p, x, y) \wedge on(l_1, x, y) \wedge (p = p_1 \vee p = p_2 \vee p = p_3 \vee p = p_4 \vee p = q_1 \vee p = q_2 \vee p = q_3)$$

has 12 minimal disjunctive answers of the form

$$(loc(p_i, 0, a_i) \wedge on(l_1, 0, a_i)) \vee (loc(q_j, b_j, c_j) \wedge on(l_1, b_j, c_j)).$$

Informally, for each $i \leq 4$ and each $j \leq 3$, either p_i or q_j lies on l_1 . Such answers are numerous (and in addition are not particularly helpful in visualising the topology). This is to be expected since the values that satisfy the query within a given model are highly interdependent. Specifically, if we know that the position of one of the points of interest is on (or off) the line, then the positioning of the other points (ie., on or off the line) is immediately determined.

On the other hand, there are just two conjunctive answers, namely

$$\{loc(p_i, 0, a_i) \wedge on(l_1, 0, a_i) \mid 1 \leq i \leq 4\} \quad \text{and}$$

$$\{loc(q_j, b_j, c_j) \wedge on(l_1, b_j, c_j) \mid 1 \leq j \leq 3\}.$$

Such answers would be far more useful in understanding the topology. Note also that a user requesting that (the various possibilities for) l_1 be drawn, is in effect requesting that the various conjunctive answers to the query $?on(l_1, x, y)$ be computed (and depicted graphically).

2.2.5 Example. Suppose that our database contains the following information about employees' ages.

$$\begin{aligned}
& age(Chris, 30) \vee age(Chris, 31) \vee age(Chris, 32) \\
& age(Sam, 58) \vee age(Sam, 61) \vee age(Sam, 62) \\
& age(Fred, 62) \vee age(Fred, 63) \vee age(Fred, 64) \\
& age(John, 63) \vee age(John, 64)
\end{aligned}$$

and we ask which of these employees are close to retirement age, specifically

$$?age(r, x) \wedge x \geq 62 \wedge (r = Chris \vee r = Sam \vee r = Fred \vee r = John).$$

Clearly there are 12 conjunctive answers but only 2 minimal disjunctive answers, namely

$$\begin{aligned}
& age(Fred, 62) \vee age(Fred, 63) \vee age(Fred, 64), \quad \text{and} \\
& age(John, 63) \vee age(John, 64).
\end{aligned}$$

This is to be expected, since the ages of the different employees are clearly independent of each other. We do however see (again) that the conjunctive answers provide more information about the query, specifically that *Sam* might be close to the retirement age (since *Sam* appears in one of the conjunctive answers).

Many naturally occurring queries $?Q$ satisfy the constraint that any given element from \mathcal{Q} excludes all others - formally $T \models Q \rightarrow \neg Q'$ for all distinct $Q, Q' \in \mathcal{Q}$ - in which case it is clear that each conjunctive answer is (at most) a singleton set. This feature is most commonly seen when functionally dependent attributes are involved, as in the query $?age(Chris, x)$.

The following corollary shows that under such conditions (in fact under the slightly weaker conditions seen in (a) or (b)), disjunctive and conjunctive answers effectively coincide.

2.2.6 Corollary. Let $?Q$ is given, then the following are equivalent.

- (a) For each conjunctive answer \mathcal{A} there is some $Q \in \mathcal{A}$ such that $T \models Q \rightarrow \bigvee\{Q' \mid Q' \in \mathcal{Q} - \{Q\}\}$.
- (b) Each minimal conjunctive answer of $?Q$ is a singleton set.
- (c) $\{Q \in \mathcal{Q} \mid \{Q\} \text{ is a minimal conjunctive answer}\}$ is the unique minimal disjunctive answer.
- (d) There is a unique minimal disjunctive answer.
- (e) $\{Q \in \mathcal{Q} \mid \{Q\} \text{ is a minimal conjunctive answer}\}$ is a disjunctive answer.

Proof. (a) \rightarrow (b) is trivial, (b) \rightarrow (c) follows trivially from Theorem 2.1.4, and (c) \rightarrow (d) is trivial.

(d) \rightarrow (e). Let \mathcal{D}' be the unique minimal disjunctive answer, then we need to show that $\mathcal{D} = \{Q \in \mathcal{Q} \mid \{Q\} \text{ is a minimal conjunctive answer}\} \supseteq \mathcal{D}'$. Suppose $Q \in \mathcal{D}'$, then by Corollary 2.1.5, we may find a minimal conjunctive answer \mathcal{A} such that $\mathcal{A} \cap \mathcal{D}' = \{Q\}$.

If $Q' \in \mathcal{A}$, then by Corollary 2.1.6, $Q' \in \mathcal{D}'$, whence $Q' = Q$. Thus $\mathcal{A} = \{Q\}$ and $Q \in \mathcal{D}$.

(e) \rightarrow (a). Suppose that \mathcal{A} is a conjunctive answer. If $\mathcal{D} = \{Q \in \mathcal{Q} \mid \{Q\} \text{ is a minimal conjunctive answer}\}$ is a disjunctive answer, then by Theorem 2.1.4, $\mathcal{D} \cap \mathcal{A} \neq \emptyset$. But if $Q \in \mathcal{D} \cap \mathcal{A}$, then $\{Q\}$ is a conjunctive answer, whence $T \not\models Q \rightarrow \bigvee \{Q' \mid Q' \in \mathcal{Q} - \{Q\}\}$.

■

2.3 Conjunctive answers and view insertions.

Somewhat surprisingly, the problem of computing conjunctive answers can be reduced to that of performing view insertions. Specifically we show that if $?Q$ is a query of T , then we may find a database T^* such that the conjunctive answers to $?Q$ in T are generated by solving a particular view insertion problem in T^* .

2.3.1 View insertions. Suppose that $\mathcal{Q} \subseteq \mathcal{L}$, then ([Fa83, Gr93, Jo97]) the problem of “inserting” $\bigvee \mathcal{Q}$ into T is to find a strengthening of T of the form $T' = T \cup \{\bigvee S_i \mid i \leq n\}$ such that

- (i) for each $i \leq n$, $S_i \subseteq \text{EXT}(\mathcal{L})$ and $\text{EXT}(T') - \{\bigvee S_i\} \not\models \bigvee S_i$,
- (ii) $T' \models \bigvee \mathcal{Q}$, and
- (iii) if $T \cup \{\bigvee E_j \mid j \leq m\} \models \bigvee \mathcal{Q}$ (where each $E_j \subseteq \text{EXT}(\mathcal{L})$), then

$$T \cup \{\bigvee E_j \mid j \leq m\} \models T'.$$

The second constraint in (i) can obviously be forgone if we are willing to accept some redundancy in T' . Condition (iii) says that T' is the minimal strengthening of T (of the required form) that satisfies $\bigvee \mathcal{Q}$. Quite clearly T' is unique if it exists.

Aside: It should be noted that the above “definition” of inserting a view tuple may not necessarily be the most natural approach, especially in the case of non-positive databases. We return to this issue at the end of Section 3.4.

2.3.2 Definition. Let $\mathcal{Q} \subseteq \mathcal{L}$ be given.

- (a) For each $P \in \mathcal{Q}$, let P^* and P' be new predicates not appearing in \mathcal{L} . In addition, let E and \mathbb{F} be distinguished predicates not appearing in $\mathcal{L} \cup \{P' \mid P \in \mathcal{Q}\} \cup \{P^* \mid P \in \mathcal{Q}\}$, and let $\mathcal{L}^* = \mathcal{L} \cup \{P' \mid P \in \mathcal{Q}\} \cup \{P^* \mid P \in \mathcal{Q}\} \cup \{E, \mathbb{F}\}$.
- (b) For each $\mathcal{A} \subseteq \mathcal{Q}$, let

$$\mathcal{A}^* = \{P' \mid P \in \mathcal{Q} - \mathcal{A}\} \cup \{P^* \mid P \in \mathcal{A}\}.$$

- (c) We stratify \mathcal{L}^* via the function ℓ^+ defined as

$$\ell^+(R) = \begin{cases} 0, & \text{if } R \in \{E\} \cup \{P^* \mid P \in \mathcal{Q}\} \cup \{P' \mid P \in \mathcal{Q}\}; \\ \ell(R) + 1, & \text{if } R \in \mathcal{L}; \\ n + 2, & \text{if } R = \mathbb{F}. \end{cases}$$

(where $n = |\mathcal{L}|$).

(d) Let T be given, then define

$$T^* = \{E\} \cup \{E \rightarrow C \mid C \in \text{EXT}(T)\} \cup \text{INT}(T) \cup \{P' \wedge \neg P \rightarrow \mathbb{F} \mid P \in \mathcal{Q}\} \cup \{P^* \wedge P \rightarrow \mathbb{F} \mid P \in \mathcal{Q}\}.$$

Our view update problem is to insert \mathbb{F} into T^* . The following three results show that $T^* \cup \{\bigvee \mathcal{A}^* \mid \mathcal{A} \text{ is a conjunctive answer of } ?\mathcal{Q} \text{ in } T\}$ is the (unique) database that allows the insertion of \mathbb{F} into T^* according to the conditions of Section 2.3.1.

It is clear that if \mathcal{A} and \mathcal{B} are distinct subsets of \mathcal{Q} , then $\mathcal{A}^* \not\subseteq \mathcal{B}^*$, whence we have the following result.

2.3.3 Proposition. If \mathcal{B} is a conjunctive answer of $?\mathcal{Q}$, then $\bigvee \mathcal{B}^*$ is not redundant in $T^* \cup \{\bigvee \mathcal{A}^* \mid \mathcal{A} \text{ is a conjunctive answer of } ?\mathcal{Q} \text{ in } T\}$.

2.3.4 Theorem. $T^* \cup \{\bigvee \mathcal{A}^* \mid \mathcal{A} \text{ is a conjunctive answer of } ?\mathcal{Q} \text{ in } T\} \models \mathbb{F}$.

Proof. Suppose that M is a perfect model of $T^* \cup \{\bigvee \mathcal{A}^* \mid \mathcal{A} \text{ is a conjunctive answer}\}$. $M \cap \mathcal{L}$ is a perfect model of T , and let $\mathcal{B} = M \cap \mathcal{Q}$ be the corresponding conjunctive answer. Since $M \models \bigvee \mathcal{B}^*$, we have that $M \cap (\{P' \mid P \in \mathcal{Q} - \mathcal{B}\} \cup \{P^* \mid P \in \mathcal{B}\}) \neq \emptyset$. If $P' \in M$ (where $P \in \mathcal{Q} - \mathcal{B}$), then $M \models P' \wedge \neg P$. If $P^* \in M$ (where $P \in \mathcal{B}$), then $M \models P \wedge P^*$. In both cases, $M \models \mathbb{F}$. ■

The following result shows that the addition of $\{\bigvee \mathcal{A}^* \mid \mathcal{A} \text{ is a conjunctive answer of } ?\mathcal{Q} \text{ in } T\}$ to T^* does not strengthen T^* any more than is necessary.

2.3.5 Theorem. Suppose that $T^+ = T^* \cup \{\bigvee E_j \mid j \leq m\} \models \mathbb{F}$, where each $E_j \subseteq \text{EXT}(\mathcal{L}^*)$. Let \mathcal{A} be a conjunctive answer of $?\mathcal{Q}$ in T , then $T^+ \models \bigvee \mathcal{A}^*$.

Proof. Suppose not, then for $j \leq m$, $E_j \not\subseteq \mathcal{A}^*$. Let M_0 be a minimal model of $\text{EXT}(T^+)$ such that $M_0 \cap \mathcal{A}^* = \emptyset$. Let $M' \subseteq \mathcal{L}$ be the perfect model of T corresponding to \mathcal{A} , then clearly $M_0 \cup M'$ is a perfect model of $T_{\leq n+1}^+$.

If $P \in \mathcal{Q} - M'$, then $P \notin \mathcal{A}$, whence $P' \in \mathcal{A}^*$ and $P' \notin M_0$. If $P \in \mathcal{Q} \cap M'$, then $P \in \mathcal{A}$, whence $P^* \in \mathcal{A}^*$ and $P^* \notin M_0$.

Thus for each $P \in \mathcal{Q}$, $M_0 \cup M' \not\models (P' \wedge \neg P) \vee (P^* \wedge P)$, whence $M_0 \cup M'$ is a perfect model of $T_{\leq n+2}^+ = T^+$, contradicting the fact that $T^+ \models \mathbb{F}$. ■

The preceding reduction is so direct that we might anticipate that a method of performing view insertions could be amended to compute conjunctive answers. In fact in Section 4, we show that a query processing method (from [Jo95]) can be adapted to the problem of performing view insertions.

The converse reduction, ie., reducing the problem of view insertions to that of computing conjunctive answers is trivial for positive databases. The dual T^* of T is

formed by discarding $\text{EXT}(T)$ and replacing each rule

$$A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B_1 \vee \dots \vee B_r$$

in $\text{INT}(T)$ by the rule

$$B_1 \wedge B_2 \wedge \dots \wedge B_r \rightarrow A_1 \vee A_2 \vee \dots \vee A_n.$$

It is easy to show (cf., Definition 3.4.1) that the problem of inserting $\bigvee \mathcal{Q}$ into T is solved by adding to T the conjunctive answers of $\text{?EXT}(\mathcal{L})$ in $T^* \cup \{Q \mid Q \in \mathcal{Q}\}$.

For non-positive databases, a similar reduction can be obtained using the methods of Section 4.5, although the reduction is somewhat indirect.

§3. CYCLIC COVERS.

Cyclic trees were introduced in [Jo96] as a means of testing minimal and perfect model membership, and cyclic covers were introduced in [Jo95] as a means of processing queries under the perfect model semantics.

In this section we show that cyclic covers can be used to characterise perfect models (Section 3.2), conjunctive answers (Section 3.3), and view insertions (Section 3.4). We also use our results to illustrate a method (Example 3.4.12) for performing view insertions (and hence for computing conjunctive answers). Sections 3.1 and 3.2 review the definitions of cyclic trees and cyclic covers.

It should be emphasised that the results below are presented in their full generality (at least as far as the ground level is concerned). In order for our results to yield efficient methods, it may be necessary to impose certain constraints on the database, and this issue is discussed in Section 5.

3.1 Cyclic trees.

Suppose that $M \subseteq \mathcal{L}_{\leq \alpha}$ is a model of $T_{\leq \alpha}$ and $M_{< \alpha}$ is a perfect model of $T_{< \alpha}$. By Definition 1.1.3, $M_{\leq \alpha}$ is a perfect model of $T_{\leq \alpha}$ iff for each non-empty subset $\mathcal{S} \subseteq M_\alpha$ there is a $C \in T_\alpha$ such that $M_{< \alpha} \cup (M_\alpha - \mathcal{S}) \not\models C$ (and following the terminology of [Jo96] we say that C *witnesses* \mathcal{S}). Notice that C , \mathcal{S} and M must have the following properties: $\text{antec}(C) \subseteq M - \mathcal{S}$, $M \cap \mathcal{N}(C) = M_{< \alpha} \cap \mathcal{N}(C) = \emptyset$ (since $\mathcal{N}(C) \subseteq \mathcal{L}_{< \alpha}$) and $M \cap (\text{conseq}(C) - \mathcal{S}) = (M_\alpha - \mathcal{S}) \cap \text{conseq}(C) = \emptyset$ (since $\text{conseq}(C) \subseteq \mathcal{L}_\alpha$). Also, $M \models T_{\leq \alpha}$, hence we must have that $M_\alpha \cap \text{conseq}(C) \cap \mathcal{S} \neq \emptyset$.

Thus, in order to check the perfectness of $M_{\leq \alpha}$ it suffices to check that each non-empty subset of M_α is witnessed. Needless to say, such checking can be made more efficient if the number of subsets that need to be checked can be reduced.

Cyclic trees were introduced in [Jo96, Jo95] as a means of testing minimal and perfect model membership. Their structure is intended to identify a witnessing rule for certain subsets (the cycles within the tree) of an intended perfect model. The following example (taken from [Jo95, Jo96]) illustrates and motivates this structure.

3.1.1 Example. Suppose that T consists of the following rules:

- | | | |
|--|--|--|
| 1. $Q_2 \wedge Q_3 \wedge \neg R_1 \rightarrow Q_1 \vee Q_5$ | 2. $Q_1 \wedge \neg R_2 \rightarrow Q_2$ | 3. $S_2 \wedge \neg R_3 \rightarrow Q_3$ |
| 4. $S_3 \rightarrow Q_1 \vee Q_2 \vee Q_6$ | 5. $S_2 \vee R_5$ | 6. $S_1 \rightarrow Q_3 \vee Q_2$ |
| 7. $S_3 \vee R_7$ | 8. $R_1 \rightarrow Q_2$ | |

with $\ell(Q_i) = 1$, $\ell(S_i) = \ell(R_j) = 0$, and we wished to determine whether Q_1 lies in some perfect model of T .

Suppose that Q_1 lies in the perfect model M of T . $\{Q_1\}$ can only be witnessed by either rule 1 or rule 4. Suppose we *guess* that rule 1 witnesses $\{Q_1\}$ in M , then M must satisfy the constraint $\{Q_2, Q_3, Q_1\} \subseteq M \subseteq \mathcal{L} - \{R_1, Q_5\}$. We represent this using the tree \mathcal{T}_1 depicted in Figure 3.1.1(i). RN_1 indicates that rule 1 has been “applied”. Only Q_2, Q_3 , ie., the antecedents, are depicted since the constraint dictates that these predicates are in M .

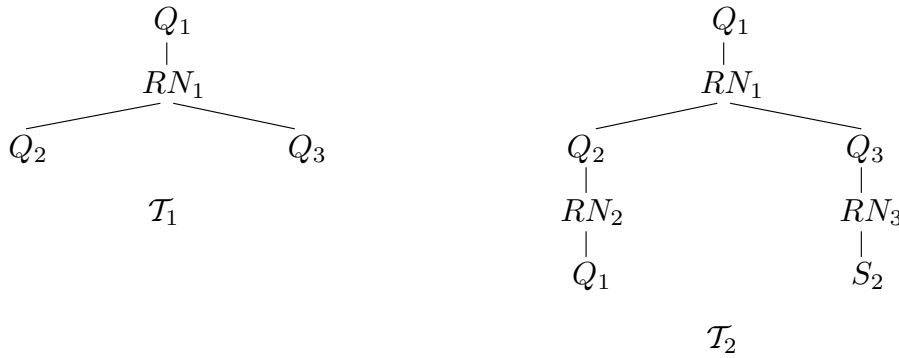


Figure 3.1.1(i).

We now look for a witnessing rule for $\{Q_2\}$. Given that M satisfies the above constraint, then $\{Q_2\}$ can only be witnessed in M by rule 2. Similarly, $\{Q_3\}$ can only be witnessed by rule 3, yielding the new constraint $\{S_2, Q_2, Q_3, Q_1\} \subseteq M \subseteq \mathcal{L} - \{R_3, R_2, R_1, Q_5\}$, and the new tree \mathcal{T}_2 (Figure 3.1.1(i)).

The left hand branch forms a “cycle”. There is no point working with Q_1 or Q_2 alone (since we have already done so), thus we look for a rule that witnesses $\{Q_1, Q_2\}$, the only candidate being rule 4 (whence $S_3 \in M$ and $Q_6 \notin M$), and the only possible witness for $\{S_3\}$ is rule 7 (whence $R_7 \notin M$). Along the right hand branch, the only possible witness for $\{S_2\}$ is rule 5 (whence $R_5 \notin M$). The final tree is depicted as \mathcal{T}_3 (Figure 3.1.1(ii)), and the final constraint is $\{S_3, S_2, Q_2, Q_3, Q_1\} \subseteq M \subseteq \mathcal{L} - \{R_5, R_7, Q_6, R_3, R_2, R_1, Q_5\}$.

Note however that (using the rules in the tree) any model of T that is disjoint from $\{R_5, R_7, Q_6, R_3, R_2, R_1, Q_5\}$ must contain $\{S_3, S_2, Q_2, Q_3, Q_1\}$.

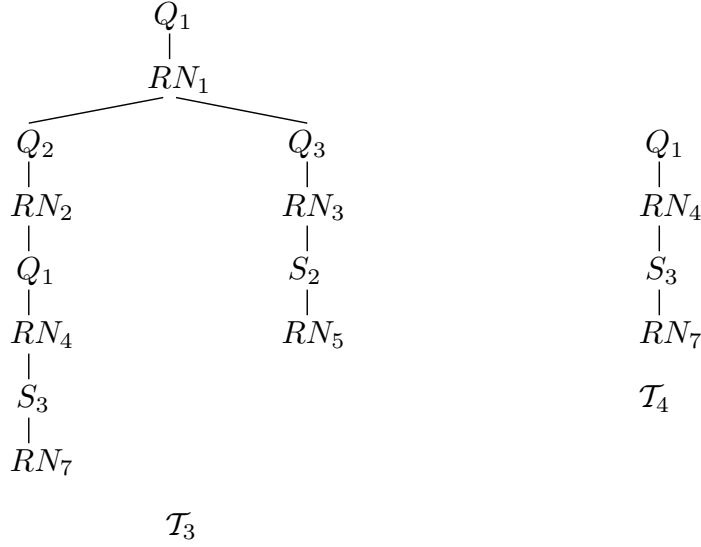


Figure 3.1.1(ii).

The only stage in the above argument in which there was an alternative witnessing rule was the first stage. If instead we had guessed that rule 4 was the witnessing rule for $\{Q_1\}$ in M , we would have obtained the tree \mathcal{T}_4 (Figure 3.1.1(ii)), and the constraint $\{S_3, Q_1\} \subseteq M \subseteq \mathcal{L} - \{R_7, Q_2, Q_6\}$. Again, any model of T that is disjoint from $\{R_7, Q_2, Q_6\}$ must contain $\{S_3, Q_1\}$.

Thus we see that Q_1 lies in some perfect model of T iff either $T \not\models \bigvee\{R_7, Q_2, Q_6\}$ or $T \not\models \bigvee\{R_5, R_7, Q_6, R_3, R_2, R_1, Q_5\}$.

The following definition captures the basic structure of the trees depicted above. The motivation is that if \mathcal{T} is such a tree and M is a perfect model containing the predicates in the tree, then for each predicate (node) N in \mathcal{T} , the rule (node) below N witnesses the “cycle above N ” (denoted by $\text{CYC}(N)$). The reason for using the cycle (rather than some other subset of the branch) is to ensure that the construction required to prove Theorem 3.1.5 (below) terminates, and also to ensure that Theorem 3.1.6 (below) holds.

3.1.2 Definition [Jo95, Jo96]. Suppose that \mathcal{T} is a tree containing two types of nodes, rule nodes and predicate nodes, satisfying conditions (i) - (iv) below.

- (i) Each rule node RN is labelled with a rule $C \in T$ (and we write RN_C). Each predicate node N is labelled with a predicate $R \in \mathcal{L}$ (and we write $\text{lab}(N) = R$).
- (ii) The root node $\text{root}(\mathcal{T})$ (at the top of the tree) is a predicate node.
- (iii) If RN_C is a rule node, then for each predicate $R \in \text{antec}(C)$, RN_C has a child node labelled with R . RN_C has no other child nodes.
- (iv) If N is a predicate node (other than the root) with $\text{lab}(N) = R$, then its parent node is a rule node RN_D (with $R \in \text{antec}(D)$). A predicate node can have at most one child node. If the child exists, then it must be a rule node.

Then we make the following definitions.

- (a) If node1 is the parent of node2, then we write $\text{node1} > \text{node2}$, the $>$ relationship being transitive. Thus $\text{root}(\mathcal{T}) \geq N$ for every node $N \in \mathcal{T}$.
- (b) If N is a predicate node, then the *cycle* of N is defined via

$$\text{CYC}(N) = \{\text{lab}(M) \mid \exists N'(N' \geq M \geq N, \text{lab}(N') = \text{lab}(N))\}.$$

If RN_C is the child of N , then let $\mathcal{O}(RN_C) = \text{conseq}(C) - \text{CYC}(N)$.

- (c) $\text{Pred}(\mathcal{T}) = \{\text{lab}(N) \mid N \text{ is a predicate node in } \mathcal{T}\}$,
 $\mathcal{O}(\mathcal{T}) = \bigcup \{\mathcal{O}(RN_C) \mid RN_C \text{ occurs in } \mathcal{T}\}$, and
 $\mathcal{N}(\mathcal{T}) = \bigcup \{\mathcal{N}(C) \mid RN_C \text{ occurs in } \mathcal{T}\}$.

If we rephrase the conditions preceding Example 3.1.1, we see that $\text{Pred}(\mathcal{T}) \subseteq M$, $M \cap \mathcal{N}(\mathcal{T}) = \emptyset$, and $M \cap \mathcal{O}(\mathcal{T}) = \emptyset$. Also if RN_C has parent node N , then $\text{antec}(C) \cap \text{CYC}(N) = \emptyset$, and $\text{conseq}(C) \cap \text{CYC}(N) \neq \emptyset$.

Hence we have the following definition.

3.1.3 Definition. Let \mathcal{T} be a tree as given in Definition 3.1.2, with $\text{lab}(\text{root}(\mathcal{T})) = P$. Then \mathcal{T} is said to be a *cyclic tree* for P (in T) iff

- (i) $\text{Pred}(\mathcal{T}) \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$, and
- (ii) for each rule node RN_C in \mathcal{T} , if RN_C has parent node N then
 - $\text{conseq}(C) \cap \text{CYC}(N) \neq \emptyset$, and
 - $\text{antec}(C) \cap \text{CYC}(N) = \emptyset$.

3.1.4 Definition. A cyclic tree \mathcal{T} is *unfactored* iff \mathcal{T} contains no predicate leaf node.

Thus in Example 3.1.1, both \mathcal{T}_3 and \mathcal{T}_4 are unfactored with $\text{Pred}(\mathcal{T}_3) = \{Q_1, Q_2, Q_3, S_3, S_2\}$, $\mathcal{O}(\mathcal{T}_3) = \{Q_5, Q_6, R_7, R_5\}$, and $\mathcal{N}(\mathcal{T}_3) = \{R_1, R_2, R_3\}$. $\text{Pred}(\mathcal{T}_4) = \{Q_1, S_3\}$, $\mathcal{O}(\mathcal{T}_4) = \{Q_2, Q_6, R_7\}$ and $\mathcal{N}(\mathcal{T}_4) = \emptyset$.

3.1.5 Theorem [Jo96]. If M is a perfect model of T , then for each $P \in M$ there is an unfactored cyclic tree \mathcal{T} for P in T such that $\text{Pred}(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$.

3.1.6 Theorem [Jo96]. Let \mathcal{T} be a cyclic tree in T , and suppose that $M \subseteq \mathcal{L}_{\leq \alpha}$ is a model of $T_{\leq \alpha}$ such that $\{\text{lab}(N) \mid N \text{ is a predicate leaf node in } \mathcal{T}\} \cap \mathcal{L}_{\leq \alpha} \subseteq M \subseteq \mathcal{L}_{\leq \alpha} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$. Then $\text{Pred}(\mathcal{T}) \cap \mathcal{L}_{\leq \alpha} \subseteq M$.

The following (trivial) technical lemma will also be needed.

3.1.7 Lemma [Jo96, Proposition 5.6.1]. Let \mathcal{T} be a cyclic tree, and N a predicate node in \mathcal{T} with $\ell(\text{lab}(N)) = 0$. Then either

- (i) N is a leaf node, or
- (ii) the child node RN_C of N is a leaf node with $C \in \text{EXT}(T)$, $\text{lab}(N) \in C$, and $\mathcal{O}(RN_C) = C - \{\text{lab}(N)\}$.

3.2 Cyclic covers.

Covers were introduced in [Jo97] as a tool for investigating query processing and view updates in positive databases. The following definition extends the concept to non-positive databases.

3.2.1 Definition [Jo95]. Let $\mathcal{Q} \subseteq \mathcal{L}$. A *cover* of \mathcal{Q} in T is a non-complementary set of literals \mathcal{C} such that $\mathcal{Q} \subseteq \mathcal{C}$ and for each $C \in \text{INT}(T)$

$$\text{conseq}(C) \subseteq \mathcal{C} \implies (\text{antec}(C) \cup \overline{\mathcal{N}}(C)) \cap \mathcal{C} \neq \emptyset.$$

Notice that $\text{EXT}(T)$ plays no part in the definition of a cover.

3.2.2 Definition [Jo95]. Let \mathcal{C} be a non-complementary set of literals, then \mathcal{C} is said to be *cyclic* (in T) iff there is a set of unfactored cyclic trees $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ in T such that

- (i) $\{P \in \mathcal{L} \mid \neg P \in \mathcal{C}\} = \bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i)$, and
- (ii) $\{P \in \mathcal{L} \mid P \in \mathcal{C}\} \supseteq \bigcup_{i=0}^m (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$.

In line with Definition 3.2.1, if \mathcal{P} is a set of literals, and \mathcal{C} is a cyclic cover with $\mathcal{P} \subseteq \mathcal{C}$, then we say that \mathcal{C} is a cyclic cover of \mathcal{P} . We first show that cyclic covers characterise perfect models (Proposition 3.2.3(b)) and query processing under the perfect model semantics (Corollary 3.2.5).

3.2.3 Proposition (a). If \mathcal{C} is cyclic in T , and $M \subseteq \mathcal{L}_{\leq \alpha}$ is a model of $T_{\leq \alpha}$ with $M \cap \{P \in \mathcal{L}_{\leq \alpha} \mid P \in \mathcal{C}\} = \emptyset$, then $\{P \in \mathcal{L}_{\leq \alpha} \mid \neg P \in \mathcal{C}\} \subseteq M$.

(b). If $M \subseteq \mathcal{L}$, then M is a perfect model of T iff $\mathcal{C} = \{\neg P \mid P \in M\} \cup \{P \in \mathcal{L} \mid P \notin M\}$ is a cyclic cover in T and $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$.

Proof (a). This follows trivially from Theorem 3.1.6.

(b). Suppose that $\mathcal{C} = \{\neg P \mid P \in M\} \cup \{P \in \mathcal{L} \mid P \notin M\}$ is a cyclic cover with $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$.

Clearly $M \models \text{EXT}(T)$, since $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$. If $C \in \text{INT}(T)$ with $M \cap \text{conseq}(C) = \emptyset$, then $\text{conseq}(C) \subseteq \mathcal{C}$, whence $(\text{antec}(C) \cup \overline{\mathcal{N}}(C)) \cap \mathcal{C} \neq \emptyset$. If $\text{antec}(C) \cap \mathcal{C} \neq \emptyset$, then $\text{antec}(C) \not\subseteq M$. If $\overline{\mathcal{N}}(C) \cap \mathcal{C} \neq \emptyset$, then $M \cap \mathcal{N}(C) \neq \emptyset$. In either case $M \not\models C$.

Suppose that M is not perfect, then we may find an $\alpha \leq n$ and a model of T of the form $M^+ = M_{< \alpha} \cup M' \cup M^*$, where $M' \subset M_\alpha$ and $M^* \subseteq \mathcal{L}_{> \alpha}$. But now $M^+ \cap \{P \in \mathcal{L}_{\leq \alpha} \mid P \in \mathcal{C}\} \subseteq M \cap \{P \in \mathcal{L}_{\leq \alpha} \mid P \in \mathcal{C}\} = \emptyset$, whence by part (a), $\{P \in \mathcal{L}_{\leq \alpha} \mid \neg P \in \mathcal{C}\} \subseteq M_{\leq \alpha}^+$. However, $\{P \in \mathcal{L}_{\leq \alpha} \mid \neg P \in \mathcal{C}\} = M \cap \mathcal{L}_{\leq \alpha} = M_{< \alpha} \cup M_\alpha \supset M_{< \alpha} \cup M' = M_{\leq \alpha}^+$, a contradiction.

The implication from left to right is trivial (using Theorem 3.1.6). ■

3.2.4 Theorem [Jo95, Theorem 9.1.2]. Let \mathcal{Q} be a non-complementary set of literals, then $T \not\models \bigvee \mathcal{Q}$ iff there is a set $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ of unfactored cyclic trees in T and a cover \mathcal{C} in T such that

- (a) $\{P \in \mathcal{L} \mid \neg P \in \mathcal{Q} \cup \mathcal{C}\} \subseteq \bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i) \subseteq \mathcal{L} - \mathcal{C}$,
- (b) $\{P \in \mathcal{L} \mid P \in \mathcal{Q}\} \cup \bigcup_{i=0}^m (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i)) \subseteq \mathcal{C}$, and
- (c) $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$.

3.2.5 Corollary. Let \mathcal{Q} be a set of literals, then the following are equivalent.

- (a) $T \models \bigvee \mathcal{Q}$
- (b) for each cyclic cover \mathcal{C} of \mathcal{Q} in T , $\text{EXT}(T) \models \bigvee \text{EXT}(\mathcal{C})$.

Proof. Firstly note that the result is trivial if \mathcal{Q} is complementary (since \mathcal{Q} then has no covers).

(a) \rightarrow (b). Let \mathcal{C} be a cyclic cover of \mathcal{Q} with associated unfactored cyclic trees $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$, then \mathcal{C} and $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ satisfy conditions (a) and (b) of the preceding theorem, whence $\text{EXT}(T) \models \bigvee \text{EXT}(\mathcal{C})$.

(b) \rightarrow (a). Suppose that $T \not\models \bigvee \mathcal{Q}$, and that $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ and \mathcal{C} are as given in the preceding theorem. But then $\mathcal{D} = \{\neg P \mid P \in \bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i)\} \cup \{P \in \mathcal{L} \mid P \in \mathcal{C}\}$ is a cyclic cover of \mathcal{Q} with $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{D})$. ■

3.2.6 Corollary. If \mathcal{C} is a cyclic cover in T , then

$$T \models \bigvee \mathcal{C} \iff \text{EXT}(T) \models \bigvee \text{EXT}(\mathcal{C}).$$

In fact the proof of [Jo95, Theorem 9.1.2] shows that a minimal model M_0 of $\text{EXT}(T)$ for which $M_0 \models \neg \bigvee \text{EXT}(\mathcal{C})$ can be extended to a perfect model $M_0 \cup M_{>0}$ of T such that $M_0 \cup M_{>0} \models \neg \bigvee \mathcal{C}$.

The following theorem is fundamental to our method of constructing cyclic covers, illustrated in Example 3.4.12.

3.2.7 Theorem. Let \mathcal{C} be a cover in T , then the following are equivalent.

- (a) \mathcal{C} is cyclic
- (b) for each $P \in \mathcal{L}$, if $\neg P \in \mathcal{C}$ then there is an unfactored cyclic tree \mathcal{T} for P in T such that $\{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{C}$.

Proof (\rightarrow). Let $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ be a sequence of unfactored cyclic trees in T such that $\bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i) = \{R \in \mathcal{L} \mid \neg R \in \mathcal{C}\}$ and $\bigcup_{i=0}^m (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i)) \subseteq \mathcal{C}$.

Let $T' = \{C \in T \mid \exists i \leq m (C \text{ labels some rule node in } \mathcal{T}_i)\}$, then clearly \mathcal{C} is a cyclic cover in T' .

If $E \in \text{EXT}(T')$, then there is some $R \in E$ such that $R \in \bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i)$, whence $\neg R \in \mathcal{C}$, and $R \notin \mathcal{C}$. In particular, $\text{EXT}(T') \not\models \bigvee \text{EXT}(\mathcal{C})$, thus by Corollary 3.2.6, we may find a perfect model M of T' such that $\{R \in \mathcal{L} \mid \neg R \in \mathcal{C}\} \subseteq M \subseteq \{R \in \mathcal{L} \mid R \notin \mathcal{C}\}$.

Now $\neg P \in \mathcal{C}$, thus by Theorem 3.1.5, we may find an unfactored cyclic tree \mathcal{T} for P in T' such that $\text{Pred}(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$. Clearly \mathcal{T} is cyclic in T .

We first show that $\{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \subseteq \mathcal{C}$. Let N be a predicate node in \mathcal{T} . If $N = \text{root}(\mathcal{T})$, then $\neg \text{lab}(N) = \neg P \in \mathcal{C}$. Suppose that N is not the root, and that

RN_C is the parent of N . Since RN_C appears in some \mathcal{T}_{i_0} we have immediately that $\neg lab(N) \in \{\neg R \mid R \in \text{antec}(C)\} \subseteq \{\neg R \mid R \in \text{Pred}(\mathcal{T}_{i_0})\} \subseteq \mathcal{C}$.

To see that $\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{C}$, let RN_D be a rule node in \mathcal{T} . Again RN_D appears in some \mathcal{T}_{i_1} , whence $\mathcal{N}(D) \subseteq \mathcal{N}(\mathcal{T}_{i_1}) \subseteq \mathcal{C}$.

Suppose that N' is the parent node of RN_D in \mathcal{T}_{i_1} , then $\text{conseq}(D) = \mathcal{O}(\mathcal{T}_{i_1}, RN_D) \cup (\text{conseq}(D) \cap \text{CYC}(\mathcal{T}_{i_1}, N'))$, where $\text{CYC}(\mathcal{T}_{i_1}, N') \subseteq \text{Pred}(\mathcal{T}_{i_1})$ is $\text{CYC}(N')$ computed in \mathcal{T}_{i_1} , etc. If $R \in \mathcal{O}(\mathcal{T}, RN_D) \subseteq \text{conseq}(D) \cap \mathcal{O}(\mathcal{T})$, then either $R \in \mathcal{O}(\mathcal{T}_{i_1}, RN_D) \subseteq \mathcal{O}(\mathcal{T}_{i_1}) \subseteq \mathcal{C}$, or $R \in \text{Pred}(\mathcal{T}_{i_1})$, in which case $\neg R \in \mathcal{C}$ and thus $R \in M$ (contradicting the fact that $M \cap \mathcal{O}(\mathcal{T}) = \emptyset$).

(\leftarrow). This is trivial. ■

3.3 Cyclic covers and conjunctive answers.

Given that cyclic covers characterise perfect models, it is of no surprise that they also characterise conjunctive answers.

3.3.1 Theorem. A set $\mathcal{A} \subseteq \mathcal{Q}$ is a conjunctive answer of $?Q$ iff there is a cyclic cover \mathcal{C} of $\{\neg P \mid P \in \mathcal{A}\} \cup (\mathcal{Q} - \mathcal{A})$ such that $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$.

Proof (\rightarrow). Let \mathcal{A} be a conjunctive answer, and M a perfect model of T such that $M \cap \mathcal{Q} = \mathcal{A}$. By Proposition 3.2.3(b), $\mathcal{C} = \{\neg P \mid P \in M\} \cup \{P \in \mathcal{L} \mid P \notin M\}$ is the required cyclic cover.

(\leftarrow). Let \mathcal{C} be a cyclic cover satisfying the given conditions, then by Corollary 3.2.6 we may find a perfect model M of T such that $\{P \in \mathcal{L} \mid \neg P \in \mathcal{C}\} \subseteq M \subseteq \{P \in \mathcal{L} \mid P \notin \mathcal{C}\}$. Trivially $\mathcal{A} = M \cap \mathcal{Q}$. ■

Taking $\mathcal{Q} = \{P\}$ gives the following corollary.

3.3.2 Corollary. If $P \in \mathcal{L}$, then P belongs to some perfect model of T iff there is a cyclic cover \mathcal{C} such that $\neg P \in \mathcal{C}$ and $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$.

3.4 Cyclic covers and view insertions.

Suppose that we wish to insert $\bigvee \mathcal{Q}$ into T . Corollary 3.2.5 provides us with the information about what needs to be inserted into $\text{EXT}(T)$ in order to ensure that $\bigvee \mathcal{Q}$ is inferred.

3.4.1 Definition (a). Let $cl(T, \mathcal{Q})$ be the minimal superset of T such that

(i) $\text{INT}(cl(T, \mathcal{Q})) = \text{INT}(T)$, and

(ii) whenever \mathcal{C} is a cyclic cover of \mathcal{Q} in $cl(T, \mathcal{Q})$, then $\bigvee \text{EXT}(\mathcal{C}) \in cl(T, \mathcal{Q})$.

(b). A database T' is a *closure* of T with respect to \mathcal{Q} iff $T \subseteq T' \subseteq cl(T, \mathcal{Q})$ and whenever \mathcal{C} is a cyclic cover of \mathcal{Q} in T' , then $\text{EXT}(T') \models \bigvee \text{EXT}(\mathcal{C})$.

It is easy to see that $cl(T, \mathcal{Q})$ is unique and itself a closure of T . It follows immediately from Corollary 3.2.5 that if $T \subseteq T' \subseteq cl(T, \mathcal{Q})$, then T' is a closure of T with respect to \mathcal{Q} iff $T' \models \bigvee \mathcal{Q}$. Thus in order to strengthen T to a database T' such that $T' \models \bigvee \mathcal{Q}$ (ie., condition (ii) of Section 2.3.1), it suffices to ensure that T' is a closure of T with respect to \mathcal{Q} . We will also show (Theorem 3.4.3 below) that the constraint $T' \subseteq cl(T, \mathcal{Q})$ ensures that T' is the weakest such strengthening (condition (iii) of Section 2.3.1).

3.4.2 Lemma. Suppose that $\text{INT}(T^+) = \text{INT}(T^*)$ and $T^+ \models T^*$. If \mathcal{C} is a cyclic cover in T^* , then either $\text{EXT}(T^+) \models \bigvee \text{EXT}(\mathcal{C})$, or \mathcal{C} is a cyclic cover in T^+ .

Proof. Suppose that $\text{EXT}(T^+) \not\models \bigvee \text{EXT}(\mathcal{C})$. Clearly \mathcal{C} is a cover in T^+ . To show that \mathcal{C} is cyclic in T^+ we employ Theorem 3.2.7. Suppose that $\neg P \in \mathcal{C}$, then by Theorem 3.2.7 we may find a cyclic tree \mathcal{T} (in T^*) with $\{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{C}$.

Let RN_E be a leaf node in \mathcal{T} (with $E \in \text{EXT}(T^*)$) and parent N , then $\mathcal{O}(RN_E) = E - \{lab(N)\} \subseteq \mathcal{C}$ (by Lemma 3.1.7). Since $T^+ \models T^*$ we may find a rule $h(E) \in \text{EXT}(T^+)$ such that $h(E) \subseteq E$, and since $\text{EXT}(T^+) \not\models \bigvee \text{EXT}(\mathcal{C})$, we must have that $lab(N) \in h(E)$. Let \mathcal{T}' be formed from \mathcal{T} by replacing each leaf node RN_E by $RN_{h(E)}$, then \mathcal{T}' is cyclic in T^+ with $\text{Pred}(\mathcal{T}') = \text{Pred}(\mathcal{T})$, $\mathcal{N}(\mathcal{T}') = \mathcal{N}(\mathcal{T})$, and $\mathcal{O}(\mathcal{T}') \subseteq \mathcal{O}(\mathcal{T})$. In particular, $\{\neg R \mid R \in \text{Pred}(\mathcal{T}')\} \cup \mathcal{O}(\mathcal{T}') \cup \mathcal{N}(\mathcal{T}') \subseteq \mathcal{C}$. ■

3.4.3 Theorem. If $T^+ = T \cup \{\bigvee E_j \mid j \leq m\} \models \bigvee \mathcal{Q}$ with each $E_j \subseteq \text{EXT}(\mathcal{L})$, then $\text{EXT}(T^+) \models \text{EXT}(cl(T, \mathcal{Q}))$.

Proof. Let $T^* = T \cup \{C \in \text{EXT}(cl(T, \mathcal{Q})) \mid \text{EXT}(T^+) \models C\}$. If \mathcal{C} is a cyclic cover of \mathcal{Q} in T^* , then by the preceding lemma and Corollary 3.2.5, $\text{EXT}(T^+) \models \bigvee \text{EXT}(\mathcal{C})$, whence $\bigvee \text{EXT}(\mathcal{C}) \in T^*$. By the minimality of $cl(T, \mathcal{Q})$, we have that $T^* = cl(T, \mathcal{Q})$. ■

It thus follows trivially that any two closures of T with respect to \mathcal{Q} are equivalent, and that T has a unique minimal closure with respect to \mathcal{Q} . In addition, if T' is a closure of T , then T' contains no redundant new facts (condition (i) of Section 2.3.1) iff T' is the minimal closure.

In order to close T in the manner suggested by Definition 3.4.1, we need to compute cyclic covers in $T = T^0$ to form T^1 ; then compute cyclic covers in T^1 to form T^2 , etc. At each stage, the intension is unchanged, and moreover the processing of this intension is identical - thus the processing of the intension is redundant in all but the first stage. With this in mind, the following definitions and results enable us to process the intension just once, and then use the output from this processing to close the extension. We first introduce weakly cyclic sets, whose definition (3.4.4 and 3.4.5 below) is completely independent of the extension, and which will thus also prove useful when we present our results on pre-processing the intensional database in Section 4.

If \mathcal{T} is a cyclic tree in $\text{INT}(T)$, then \mathcal{T} can have no rule leaf nodes and (by Lemma 3.1.7) if N is a predicate node with $\ell(lab(N)) = 0$, then N is a leaf. We will be interested

in trees in which all leaf nodes are of this form.

3.4.4 Definition. For $\alpha \leq n + 1$ (where $n = |\mathcal{L}|$), let $\mathcal{U}_{<\alpha}(T)$ denote the set of cyclic trees \mathcal{T} in T such that

for each predicate node $N \in \mathcal{T}$ (N is a leaf node $\iff \ell(\text{lab}(N)) < \alpha$).

Notice that if $\mathcal{T} \in \mathcal{U}_{<\alpha}(T)$, then \mathcal{T} is actually a cyclic tree in $T_{\geq\alpha}$. In particular if $\alpha > 0$, then \mathcal{T} is a cyclic tree in $\text{INT}(T)$ and \mathcal{T} can have no rule leaf nodes.

3.4.5 Definition. A non-complementary set of literals \mathcal{P} is said to be *weakly cyclic* in T iff there is a set $\{\mathcal{T}_i \mid 0 \leq i \leq m\} \subseteq \mathcal{U}_{<1}(T)$ such that

- (i) $\{P \in \mathcal{L} \mid \neg P \in \mathcal{P}\} = \bigcup_{i=0}^m \text{Pred}(\mathcal{T}_i)$, and
- (ii) $\{P \in \mathcal{L} \mid P \in \mathcal{P}\} \supseteq \bigcup_{i=0}^m (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$.

With regard to computing cyclic covers, we obviously do not wish to compute weakly cyclic covers if they cannot be extended to a fully cyclic cover. Without a detailed manipulation of the extension, we cannot precisely predict which weakly cyclic covers will be extendable in this manner.

We note however that (by Lemma 3.1.7) if $P \in \text{EXT}(\mathcal{L})$ labels a node in an unfactored cyclic tree, then the child node of N must be of the form RN_C , where $C \in \text{EXT}(T)$ and $P \in C$. In the terminology below, $P \in \bigcup \text{EXT}(T)$.

3.4.6 Definition (a). $\bigcup \text{EXT}(T)$ consists of those predicates that appear in some disjunct in $\text{EXT}(T)$, ie., $\bigcup \text{EXT}(T) = \{P \in \text{EXT}(\mathcal{L}) \mid \exists C \in \text{EXT}(T), P \in C\}$.

(b). A non-complementary set of literals \mathcal{P} is said to be *partially cyclic* in T iff \mathcal{P} is weakly cyclic in T and $\{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\} \subseteq \bigcup \text{EXT}(T)$.

Quite clearly, \mathcal{P} is partially cyclic in T iff there is a set $\{\mathcal{T}_i \mid 0 \leq i \leq m\}$ of cyclic trees satisfying the conditions of Definition 3.4.5 such that

$$\{\text{lab}(N) \mid N \text{ is a leaf node in some } \mathcal{T}_i\} \subseteq \bigcup \text{EXT}(T).$$

Alternatively, \mathcal{P} is partially cyclic in T iff \mathcal{P} is weakly cyclic in $\{C \in \text{INT}(T) \mid \text{antec}(C) \cap \text{EXT}(\mathcal{L}) \subseteq \bigcup \text{EXT}(T)\}$.

In order to extend a partially cyclic set \mathcal{P} to a cyclic set \mathcal{C} , we need to extend each of the trees \mathcal{T}_i to an unfactored tree \mathcal{T}_i^* . To do this, each of the leaf nodes N in \mathcal{T}_i is extended via some rule node $RN_{f(N)}$. The function f must have the following properties.

- (i) $\text{lab}(N) \in f(N)$ with $\mathcal{O}(\mathcal{T}_i^*, RN_{f(N)}) = f(N) - \{\text{lab}(N)\}$ (by Lemma 3.1.7).
 - (ii) $\{\text{lab}(N') \mid \exists j \leq m (N' \text{ is a leaf node in } \mathcal{T}_j)\} = \{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\}$ is disjoint from $f(N) - \{N\}$
- (since $\{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\} \subseteq \{P \in \mathcal{L} \mid \neg P \in \mathcal{C}\}$ and $f(N) - \{N\} = \mathcal{O}(\mathcal{T}_i^*, RN_{f(N)}) \subseteq \mathcal{O}(\mathcal{T}_i^*) \subseteq \mathcal{C}$ which need to be disjoint for \mathcal{C} to be non-complementary).

We thus have the following definition.

3.4.7 Definition. Let \mathcal{P} be partially cyclic in T and let f be a function $f : \{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\} \rightarrow \text{EXT}(T)$ such that for each $P \in \text{domain}(f)$

- (i) $P \in f(P)$, and
- (ii) $(f(P) - \{P\}) \cap \text{domain}(f) = \emptyset$.

The set $\mathcal{P} \cup \bigcup \{f(P) - \{P\} \mid P \in \text{domain}(f)\}$ is said to be a *completion* of \mathcal{P} in T .

Clearly if \mathcal{C} is cyclic, then \mathcal{C} is partially cyclic, and \mathcal{C} is a completion of itself. Conversely, if \mathcal{P} is partially cyclic, then any completion of \mathcal{P} is cyclic. Note also that if \mathcal{C} is a completion of \mathcal{P} , then $\mathcal{C} - \mathcal{P} \subseteq \text{EXT}(\mathcal{L})$, and hence if \mathcal{P} is a cover, then so is \mathcal{C} .

3.4.8 Theorem. Let \mathcal{Q} be a set of literals, then $T \models \bigvee \mathcal{Q}$ iff whenever \mathcal{P} is a partially cyclic cover of \mathcal{Q} in T , then $\text{EXT}(T) \models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$.

Proof. Firstly recall that $T \models \Phi$ iff Φ is true in every perfect model of T . Thus $\text{EXT}(T) \models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$ iff $\bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$ is true in every minimal model of $\text{EXT}(T)$.

(\rightarrow). Let \mathcal{P} be a partially cyclic cover of \mathcal{Q} , and suppose that M_0 is a minimal model of $\text{EXT}(T)$ such that $M_0 \not\models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$. If $P \in \text{EXT}(\mathcal{L})$ with $\neg P \in \mathcal{P}$, then $P \in M_0$ and hence we may find an $f(P) \in \text{EXT}(T)$ such that $M_0 \cap (f(P) - \{P\}) = \emptyset$.

Thus $\mathcal{C} = \mathcal{P} \cup \bigcup \{f(P) - \{P\} \mid P \in \text{EXT}(\mathcal{L}), \neg P \in \mathcal{P}\}$ is a completion of \mathcal{P} such that $M_0 \not\models \bigvee \text{EXT}(\mathcal{C})$, whence $\text{EXT}(T) \not\models \bigvee \text{EXT}(\mathcal{C})$. However, $\mathcal{Q} \subseteq \mathcal{P} \subseteq \mathcal{C}$, thus contradicting the fact that $T \models \bigvee \mathcal{Q}$ (by Corollary 3.2.5).

(\leftarrow). If $T \not\models \bigvee \mathcal{Q}$, then we may find a cyclic cover \mathcal{C} of \mathcal{Q} and a perfect model M of T such that $M \models \neg \bigvee \mathcal{C}$. But now, \mathcal{C} is partially cyclic, and $M \cap \text{EXT}(\mathcal{L})$ is a minimal model of T such that $M \cap \text{EXT}(\mathcal{L}) \not\models \bigvee \{K \in \mathcal{C} \mid \ell(K) = 0\}$. ■

3.4.9 Corollary. If \mathcal{P} is a partially cyclic cover in T , then

$$T \models \bigvee \mathcal{P} \iff \text{EXT}(T) \models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}.$$

We now return to the issue of computing closures of T . Note that if $T \subseteq T' \subseteq \text{cl}(T, \mathcal{Q})$, then $\text{INT}(T) = \text{INT}(T')$ and $\bigcup \text{EXT}(T) = \bigcup \text{EXT}(T')$. In particular, T and T' share the same partially cyclic covers. We thus have the following corollary.

3.4.10 Corollary. Let $T \subseteq T' \subseteq \text{cl}(T, \mathcal{Q})$, then the following are equivalent.

- (a) T' is a closure of T with respect to \mathcal{Q} ,
- (b) whenever \mathcal{P} is a partially cyclic cover of \mathcal{Q} in T , and \mathcal{C} is a completion of \mathcal{P} in T' , then $\text{EXT}(T') \models \bigvee \text{EXT}(\mathcal{C})$.

In order to close T with respect to \mathcal{Q} , we thus need to compute partially cyclic covers in T , and then iteratively add their completions to the database. (Of course

a given partially cyclic cover might not have any completions, in which case there is nothing to add.)

When computing partially cyclic covers in T , we can ignore subsumed partially cyclic covers, for if $\mathcal{P} \subset \mathcal{P}'$ are both partially cyclic covers, then for any completion \mathcal{C}' of \mathcal{P}' in T , there is a completion \mathcal{C} of \mathcal{P} in T' such that $\mathcal{C} \subseteq \mathcal{C}'$. Clearly we can also ignore subsumed completions.

Note that the constraint $T' \subseteq cl(T, \mathcal{Q})$ is automatically satisfied, for if $T' \subseteq cl(T, \mathcal{Q})$, \mathcal{P} is a partially cyclic cover in T and \mathcal{C} is a completion of \mathcal{P} in T' , then $\bigvee \text{EXT}(\mathcal{C}) \in cl(T, \mathcal{Q})$. We can ensure that the closure produced is minimal by simply eliminating redundant facts en-route.

The following result is instrumental in computing weakly (and hence partially) cyclic covers.

3.4.11 Theorem. Suppose that \mathcal{P} is a cover, then the following are equivalent.

- (a) \mathcal{P} is weakly cyclic,
- (b) for each $P \in \text{INT}(\mathcal{L})$, if $\neg P \in \mathcal{P}$, then there is a cyclic tree \mathcal{T} for P in $\mathcal{U}_{<1}(T)$ such that $\{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{P}$.

Proof (a) \rightarrow (b). Let X be a new predicate not appearing in \mathcal{L} , and let $T^* = T \cup \{X \vee Q \mid Q \in \text{EXT}(\mathcal{L})\}$. Then \mathcal{P} is partially cyclic in T^* and $\mathcal{P} \cup \{X\}$ is a completion of \mathcal{P} in T^* . Thus by Theorem 3.2.7 there is an unfactored cyclic tree \mathcal{T}^* for P in T^* such that $\{\neg R \mid R \in \text{Pred}(\mathcal{T}^*)\} \cup \mathcal{O}(\mathcal{T}^*) \cup \mathcal{N}(\mathcal{T}^*) \subseteq \mathcal{P} \cup \{X\}$. Removing all leaf nodes from \mathcal{T}^* yields the required tree.

The converse is trivial. ■

The following example informally illustrates a method for performing view insertions using the preceding results.

3.4.12 Example. Suppose that T contains the following rules.

- | | | |
|--|---|--|
| 1. $B_1 \wedge E_0 \wedge \neg B_0 \rightarrow A_0 \vee A_1$ | 2. $E_1 \wedge \neg E_3 \rightarrow B_0 \vee B_3$ | 3. $E_6 \wedge \neg E_1 \rightarrow B_0$ |
| 4. $E_4 \wedge E_5 \rightarrow B_2 \vee B_1$ | 5. $E_4 \vee E_3$ | 6. $E_0 \vee E_2$ |
| 7. $E_1 \vee E_8$ | 8. $E_6 \vee E_7$ | 9. $E_3 \vee E_8$ |

where $\ell(A_i) = 2$, $\ell(B_i) = 1$ and $\ell(E_i) = 0$. Suppose that we wish to add $\bigvee \mathcal{Q}$ to T , where $\mathcal{Q} = \{A_0, A_1, B_2, E_2\}$. We first set about generating partially cyclic covers of \mathcal{Q} in T . In order to do this we will use a tree structure to represent the construction. At any given stage, the tree will have the property that if \mathcal{P} is a partially cyclic cover of \mathcal{Q} in T , then there is some branch through the tree that subsumes \mathcal{P} .

Since the consequent of rule 1 is contained in \mathcal{P} , any cover must contain either B_1 , E_0 or $\neg B_0$. We illustrate this via the tree depicted in Figure 3.4.12(i). RN_1 denotes the ‘‘application’’ of rule 1.

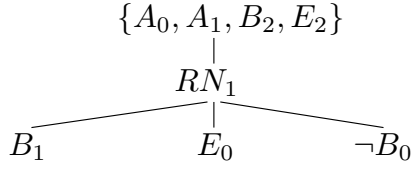


Figure 3.4.12(i).

The same reasoning may be applied to the left-hand branch using rule 4 : any cover containing B_1 and B_2 must contain either E_4 or E_5 . We thus see that both

- (i) $\{A_0, A_1, B_2, E_2, B_1, E_4\}$, and
- (ii) $\{A_0, A_1, B_2, E_2, B_1, E_5\}$

are minimal partially cyclic covers of \mathcal{P} .

We now move on to the node E_0 . Clearly if $\mathcal{P} \supseteq \{E_0, E_2\}$ is a partially cyclic cover, and \mathcal{C} is a completion of \mathcal{P} , then $\text{EXT}(T) \models \bigvee \text{EXT}(\mathcal{C})$ by rule 6. Thus any such cover is redundant, and we may therefore ignore partially cyclic covers extending $\{E_0, E_2\}$. This then yields the tree depicted in Figure 3.4.12(ii).

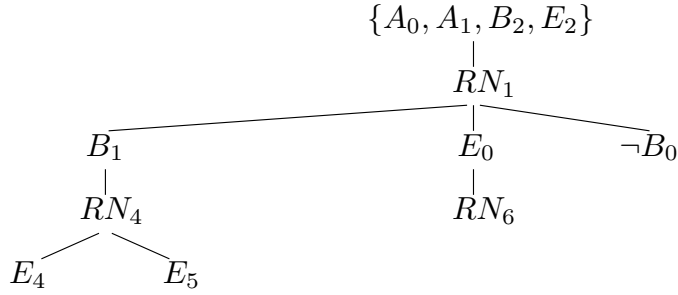


Figure 3.4.12(ii).

Returning to $\neg B_0$, if $\neg B_0$ is contained in some partially cyclic cover \mathcal{P} , then there is a cyclic tree \mathcal{T} for B_0 in $\mathcal{U}_{<1}(T)$ satisfying the conditions of Theorem 3.4.11. The possible cyclic trees for B_0 are shown in Figure 3.4.12(iii).



Figure 3.4.12(iii).

Thus, $\text{Pred}(\mathcal{T}_1) = \{B_0, E_1\}$, $\mathcal{O}(\mathcal{T}_1) = \{B_3\}$ and $\mathcal{N}(\mathcal{T}_1) = \{E_3\}$. For \mathcal{T}_2 , we have $\text{Pred}(\mathcal{T}_2) = \{B_0, E_6\}$, $\mathcal{O}(\mathcal{T}_2) = \emptyset$ and $\mathcal{N}(\mathcal{T}_2) = \{E_1\}$. Appending the corresponding

new nodes below $\neg B_0$ yields the tree depicted in Figure 3.4.12(iv). The “rule node” $RN_{\rightarrow \neg B_0}$ simply denotes the generation of cyclic trees to process the “sub-goal” $\neg B_0$.

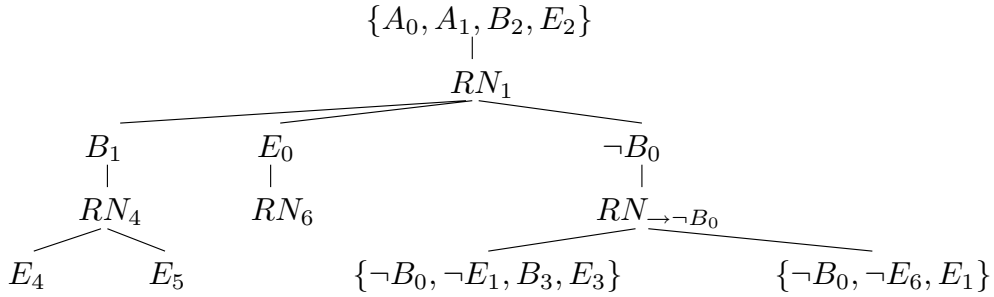


Figure 3.4.12(iv).

We thus have two other minimal partially cyclic covers, namely

(iii) $\{A_0, A_1, B_2, E_2, \neg B_0, \neg E_1, B_3, E_3\}$, and

(iv) $\{A_0, A_1, B_2, E_2, \neg B_0, \neg E_6, E_1\}$.

Now we compute completions of the partially cyclic covers. Firstly (i) and (ii) are their own completions, thus we need to add to $\text{EXT}(T)$ the facts $E_2 \vee E_4$ and $E_2 \vee E_5$.

From the completion of (iii) using rule 7 we obtain $E_2 \vee E_3 \vee E_8$, although by rule 9 this completion is redundant. The completion of (iv) using rule 8 yields $E_2 \vee E_1 \vee E_7$. We can also take the completion of (iii) with $E_2 \vee E_1 \vee E_7$ yielding $E_2 \vee E_3 \vee E_7$, and this then completes the required additions to $\text{EXT}(T)$. The updated database is

$$T \cup \{E_2 \vee E_4, E_2 \vee E_5, E_2 \vee E_1 \vee E_7, E_2 \vee E_3 \vee E_7\}.$$

The trees depicted in Figures 3.4.12(i) - (iv) are slight variants of *extended deduction trees*. Such trees were introduced and studied extensively in [Jo95, Jo95a], where methods for constructing them (at the propositional and first order levels) are presented. The methods of [Jo95] are easily amended to enable the generation of the variant trees required above.

In Section 2.3 we mentioned that the definition of view insertion given there-in may not be the most natural, especially for non-positive databases. The following examples illustrate the point.

3.4.13 Example. Suppose that $T = \{P \vee V, R, S, S \wedge \neg P \rightarrow Q, S \wedge \neg R \rightarrow Q\}$, and we wish to insert Q into T . The only partially cyclic cover for which $\text{EXT}(T) \not\models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$ is $\mathcal{P} = \{Q, \neg P, \neg R\}$, whence by Theorem 3.4.8, we need to amend T so as to make $\neg P \vee \neg R$ true.

The only completion of \mathcal{P} in T is $\{Q, \neg P, \neg R, V\}$, whence the definition of Section 2.3 would require us to add the unit clause V to T . Notice that this has the effect of rendering P false in T , and in this sense the method has adopted a biased treatment of the “sub-goal” $\neg P \vee \neg R$.

It would surely have been equally as valid to delete the fact R from the database.

Computing partially cyclic covers provides useful information about how to perform a view update. However, the above example illustrates that such information can be used in a number of ways other than that suggested by the definition of Section 2.3. Probably the most sensible approach (discussed in [Jo97]) is to regard making $\bigvee\{K \in \mathcal{P} \mid \ell(K) = 0\}$ true in T (for each partially cyclic cover \mathcal{P}) as a sub-goal, and then perform some kind of disambiguation. For instance in the above example, \mathcal{P} yields the sub-goal $\neg P \vee \neg R$. We might then for instance ([At91]) ask the user whether to replace $P \vee V$ by V or to delete R .

Various other methods for disambiguation have been suggested (some of which are applicable only to first order databases). [Ro89, Ka90, Jo97] employ null values, [Bry90] uses knowledge about the domain of variables, [De90] uses ground terms that appear elsewhere in the “derivation” and [Ka90, p656] suggests the possibility of using a type of minimality criteria. [Ke86] discusses the idea of specifying an (unambiguous) view update translator at view definition time, and [To88] suggests a similar idea. The computing of partially cyclic covers, provides the input into such a method, and the results of this paper are independent of the particular disambiguation policy employed.

3.4.14 Example. Suppose that $T = \{R, S, S \wedge \neg R \rightarrow Q\}$, and we wish to insert Q . Again the only relevant partially cyclic cover of $\mathcal{P} = \{Q, \neg R\}$, which tells us that we need to make $\neg R$ true - and in this case it is self-evident how to do this.

However, if we follow the definition of Section 2.3, the only possible way of inserting Q into T is to add the empty clause!

As regards the problem of view deletions, if $T \models \bigvee \mathcal{Q}$, and we wish to amend $\text{EXT}(T)$ so that $\bigvee \mathcal{Q}$ is not inferred from the updated database, then we need to ensure that $\bigvee \text{EXT}(\mathcal{C})$ is not inferred for some cyclic cover \mathcal{C} of \mathcal{Q} . Methods for achieving this are discussed in [Jo97], although in general there is no single update which achieves the desired effect. (Conditions are given in [Jo97] which characterise when there is an unambiguous extensional update for a view deletion.) Hence for deletions, there is an even greater case for employing disambiguation methods such as those discussed above. Partially cyclic covers again give us the relevant information needed to feed into the disambiguation method.

View insertion asks what modification must be made to the extension in order to make $\bigvee \mathcal{Q}$ true, and to some extent this can be regarded as a process of abduction [Bry90, Ka90, Ri91]. Alternatively we could view abduction as a process of inferring statements about the extension from statements about the intension.

3.4.15 Theorem. Let $P \in \text{INT}(\mathcal{L})$ and Φ be a propositional formula in $\text{EXT}(\mathcal{L})$. Then $T \models P \rightarrow \Phi$ iff whenever \mathcal{C} is a cyclic cover of $\{\neg P\}$, then $\text{EXT}(T) \models \Phi \vee \bigvee \text{EXT}(\mathcal{C})$.

Proof (\rightarrow). Suppose that \mathcal{C} is a cyclic cover of $\{\neg P\}$, and M_0 is a minimal model

of $\text{EXT}(T)$ such that $M_0 \models \neg\Phi \wedge \neg\bigvee \text{EXT}(\mathcal{C})$. But then (cf., the remarks following Corollary 3.2.6), we may extend M_0 to a perfect model $M_0 \cup M_{>0}$ of T such that $M_0 \cup M_{>0} \models \neg\bigvee \mathcal{C}$. In particular, $P \in M_{>0}$, and clearly $M_0 \cup M_{>0} \models \neg\Phi$, thus contradicting the fact that $T \models P \rightarrow \Phi$.

(\leftarrow). Suppose that M is a perfect model of T such that $P \in M$. By Proposition 4.2.4, $\mathcal{C} = \{\neg R \mid R \in M\} \cup \{P \in \mathcal{L} \mid P \notin M\}$ is a cyclic cover of $\{\neg P\}$, and $M \cap \text{EXT}(\mathcal{L})$ is a minimal model of $\text{EXT}(T)$ disjoint from $\text{EXT}(\mathcal{C})$, whence by hypothesis, $M \cap \text{EXT}(\mathcal{L}) \models \Phi$. Thus clearly, $M \models \Phi$. ■

§4. PRE-PROCESSING THE INTENSIONAL DATABASE.

The general structure of a deductive database is expected to be a large and transient extension, and a much smaller, relatively static intension. This suggests the idea of pre-processing the intension so that query processing is subsequently more efficient. In this section we show that cyclic trees and covers provide means of performing query compilation, identifying redundant intensional rules, eliminating recursion and removing positive and/or negative intensional sub-goals.

4.1 Compiling queries.

Suppose that we wish to compute the conjunctive answers of the query $?Q$. The results of the preceding section enable us to *compile* the query. Specifically we can pre-process $\text{INT}(T)$ so that the processing of the subsequent query $?Q$ then requires a manipulation of $\text{EXT}(T)$ only. Since this pre-processing must not use $\text{EXT}(T)$ we need to employ weakly (as opposed to partially) cyclic covers.

4.1.1 Theorem. \mathcal{A} is a conjunctive answer to $?Q$ iff there is a weakly cyclic cover \mathcal{P} of $\{\neg P \mid P \in \mathcal{A}\} \cup (Q - \mathcal{A})$ in $\text{INT}(T)$ such that

- (i) $\{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\} \subseteq \bigcup \text{EXT}(T)$, and
- (ii) $\text{EXT}(T) \not\models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$.

Proof (\rightarrow). Let \mathcal{C} be a cyclic cover satisfying the conditions of Theorem 3.3.1, then $\mathcal{P} = \mathcal{C}$ satisfies conditions (i) and (ii) above.

(\leftarrow). By Corollary 3.4.9, $T \not\models \bigvee \mathcal{P}$. If M is a perfect model of T such that $M \models \neg\bigvee \mathcal{P}$, then $M \cap \mathcal{Q} = \mathcal{A}$. ■

4.1.2 Compilation.

The compilation process thus consists of computing weakly cyclic covers \mathcal{P} in $\text{INT}(T)$ such that $\mathcal{Q} \subseteq \mathcal{P} \cup \{P \in \mathcal{L} \mid \neg P \in \mathcal{P}\}$. Notice that such a compilation process does not require access to $\text{EXT}(T)$.

4.1.3 Query processing. By Theorem 4.1.1, \mathcal{A} is a conjunctive answer to the query $?Q$ iff there is a weakly cyclic cover \mathcal{P} (such that $\mathcal{Q} \subseteq \mathcal{P} \cup \{P \in \mathcal{L} \mid \neg P \in \mathcal{P}\}$) with

- (i) $\mathcal{A} = \{P \in \mathcal{Q} \mid \neg P \in \mathcal{P}\}$,
- (ii) $\{P \in \mathcal{L} \mid \neg P \in \mathcal{P}\} \subseteq \bigcup \text{EXT}(T)$ and
- (iii) $\text{EXT}(T) \not\models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$.

Having computing the weakly cyclic covers in the compilation phase, this then requires access to $\text{EXT}(T)$ only.

Recall that $T \models \bigvee \mathcal{Q}$ iff every conjunctive answer to the query $?Q$ is non-empty. We can thus also compile a yes/no query of the form $?T \models \bigvee \mathcal{Q}$ by the above process with $\mathcal{A} = \emptyset$.

4.1.4 Example. Suppose that $\text{INT}(T)$ consists of the following rules.

1. $B_1 \wedge E_0 \wedge \neg B_0 \rightarrow A_0 \vee A_1$
2. $E_1 \wedge \neg E_3 \rightarrow B_0 \vee B_3$
3. $E_6 \wedge \neg E_1 \rightarrow B_0$
4. $E_4 \wedge E_5 \rightarrow B_2 \vee B_1$

where $\ell(A_i) = 2$, $\ell(B_i) = 1$ and $\ell(E_i) = 0$. Suppose that we wish to compile the yes/no query $?T \models \bigvee \mathcal{Q}$, where $\mathcal{Q} = \{A_0, A_1, B_2, E_2\}$. By the above remarks, $T \models \bigvee \mathcal{Q}$ iff $\mathcal{A} = \emptyset$ is a conjunctive answer to $?Q$ iff there is a weakly cyclic cover \mathcal{P} of \mathcal{Q} satisfying the conditions of Theorem 4.1.1.

As in Example 3.4.12, the minimal weakly cyclic covers of \mathcal{Q} in $\text{INT}(T)$ are

- (i) $\{A_0, A_1, B_2, E_2, B_1, E_4\}$,
- (ii) $\{A_0, A_1, B_2, E_2, B_1, E_5\}$,
- (iii) $\{A_0, A_1, B_2, E_2, \neg B_0, \neg E_1, B_3, E_3\}$, and
- (iv) $\{A_0, A_1, B_2, E_2, \neg B_0, \neg E_6, E_1\}$.

Thus $T \models \bigvee \mathcal{Q}$ iff

- (i) $\text{EXT}(T) \models E_2 \vee E_4$, and
- (ii) $\text{EXT}(T) \models E_2 \vee E_5$, and
- (iii) $E_1 \notin \bigcup \text{EXT}(T)$ or $\text{EXT}(T) \models E_2 \vee \neg E_1 \vee E_3$, and
- (iv) $E_6 \notin \bigcup \text{EXT}(T)$ or $\text{EXT}(T) \models E_2 \vee \neg E_6 \vee E_1$.

4.2 Eliminating redundant rules.

4.2.1 Definition. Let $C \in \text{INT}(T)$, then we say that C is *redundant* in $\text{INT}(T)$ iff for every database T_1 , $\text{EXT}(T_1) \cup \text{INT}(T)$ and $\text{EXT}(T_1) \cup \text{INT}(T) - \{C\}$ are equivalent.

4.2.2 Theorem. C is redundant in $\text{INT}(T)$ iff there is no weakly cyclic cover \mathcal{P} of $\mathcal{A} = \{\neg A \mid A \in \text{antec}(C)\} \cup \mathcal{N}(C) \cup \text{conseq}(C)$ in $\text{INT}(T) - \{C\}$.

Proof. Firstly note that by Theorem 1.1.6, $\text{EXT}(T_1) \cup \text{INT}(T)$ and $\text{EXT}(T_1) \cup \text{INT}(T) - \{C\}$ are equivalent iff $\text{EXT}(T_1) \cup \text{INT}(T) - \{C\} \models \bigvee \mathcal{A}$.

(\rightarrow). Suppose that \mathcal{P} is a weakly cyclic cover of \mathcal{A} in $\text{INT}(T) - \{C\}$, and let T_1 be the extensional database consisting of unit clauses of the form $T_1 = \{P \in \text{EXT}(\mathcal{L}) \mid \neg P \in \mathcal{P}\}$. But then \mathcal{P} is a partially cyclic cover of \mathcal{A} in $T_1 \cup \text{INT}(T) - \{C\}$ with $T_1 \not\models \bigvee \{K \in \mathcal{P} \mid \ell(K) = 0\}$. Hence by Theorem 3.4.8, $T_1 \cup \text{INT}(T) - \{C\} \not\models \bigvee \mathcal{A}$.

(\leftarrow). Suppose that $\text{EXT}(T_1) \cup \text{INT}(T) - \{C\} \not\models \bigvee \mathcal{A}$, then by Theorem 3.4.8, we may find a partially cyclic cover \mathcal{P} of \mathcal{A} in $\text{EXT}(T_1) \cup \text{INT}(T) - \{C\}$. But then \mathcal{P} is weakly cyclic in $\text{INT}(T) - \{C\}$. ■

4.3 Eliminating recursion.

Partial evaluation is an optimisation technique introduced into logic programming by Komorowski [Ko81]. In the context of deductive databases, it involves the transformation of a database T into an equivalent database T' (ie., T and T' have the same perfect models) such that T' has some desirable property. Because we expect the extensional database to be transient, we also wish to ensure that our transformation is immune to future changes in the extension. Clearly compilation can be regarded as a query specific form of partial evaluation.

In this section we show how the results of Section 3 can be used to eliminate recursion. Recall that T is stratified via the level function ℓ .

4.3.1 Definition. Let C be a rule, then C is recursion-free (with respect to ℓ) iff $\ell(A) < \ell(C)$ for each $A \in \text{antec}(C)$.

We aim to transform T into an equivalent database T' such that each rule in T' is recursion free. Suppose that $C \in \text{INT}(T)$ and $A \in \text{antec}(C)$ with $\ell(A) = \ell(C) = \alpha$.

4.3.2 Definition. Let \mathcal{T} be a cyclic tree for A , then define $C_{\mathcal{T}}$ to be the rule

$$\bigwedge (\text{antec}(C) - \{A\}) \wedge \bigwedge \overline{\mathcal{N}}(C) \wedge \bigwedge \{lab(N) \mid N \text{ is a predicate leaf node in } \mathcal{T}\} \wedge \\ \bigwedge \{\neg Q \mid Q \in \mathcal{N}(\mathcal{T})\} \wedge \bigwedge \{\neg Q \mid Q \in \mathcal{O}(\mathcal{T}) \cap \mathcal{L}_{<\alpha}\} \rightarrow \bigvee \text{conseq}(C) \vee \bigvee (\mathcal{O}(\mathcal{T}) \cap \mathcal{L}_{\alpha}).$$

Note that for any such cyclic tree, $\mathcal{N}(\mathcal{T}) \subseteq \mathcal{L}_{<\alpha}$ and $\mathcal{O}(\mathcal{T}) \subseteq \mathcal{L}_{\leq\alpha}$, hence $C_{\mathcal{T}}$ is a well-defined rule. Note also that if $\mathcal{T} \in \mathcal{U}_{<\alpha}$, then $\{lab(N) \mid N \text{ is a predicate leaf node in } \mathcal{T}\} \subseteq \mathcal{L}_{<\alpha}$, and hence the “degree” of recursion in $C_{\mathcal{T}}$ is less than in C , and each leaf node in \mathcal{T} is a predicate node.

4.3.3 Theorem. If \mathcal{T} is a cyclic tree for A in T , then $T \models C_{\mathcal{T}}$.

Proof. If $T \not\models C_{\mathcal{T}}$, then we may find a perfect model M of T such that $\text{antec}(C_{\mathcal{T}}) \subseteq M \subseteq \mathcal{L} - (\mathcal{N}(C_{\mathcal{T}}) \cup \text{conseq}(C_{\mathcal{T}}))$. In particular, $\{lab(N) \mid N \text{ is a predicate leaf node in } \mathcal{T}\} \subseteq M$ and M is disjoint from $\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$, whence by Theorem 3.1.6, $A \in M$. In addition, $\text{antec}(C) - \{A\} \subseteq M \subseteq \mathcal{L} - (\mathcal{N}(C) \cup \text{conseq}(C))$, which then contradicts the fact that $M \models C$. ■

Let T^{++} be the (unique) minimal superset of T such that whenever \mathcal{T} is a cyclic tree for A in $\mathcal{U}_{<\alpha}(T^{++} - \{C\})$, then $C_{\mathcal{T}} \in T^{++}$. It follows immediately from Theorems 1.1.6 and 4.3.3 that T and T^{++} are equivalent.

Let $T \subseteq T^+ \subseteq T^{++}$ be such that whenever \mathcal{T} is a cyclic tree for A in $\mathcal{U}_{<\alpha}(T^+ - \{C\})$, then there is a rule in T^+ which subsumes $C_{\mathcal{T}}$.

4.3.4 Theorem. T^+ and $T^+ - \{C\}$ are equivalent.

Proof. Let M be a perfect model of $T^+ - \{C\}$, and suppose that $\text{antec}(C) \subseteq M$ and $M \cap (\mathcal{N}(C) \cup \text{conseq}(C)) = \emptyset$. By Theorem 3.1.5 we may find a cyclic tree \mathcal{T} for A in $\mathcal{U}_{<\alpha}(T^+ - \{C\})$ such that $\text{Pred}(\mathcal{T}) \subseteq M$ and $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$, whence $M \not\models C_{\mathcal{T}}$, and $T^+ - \{C\} \not\models C_{\mathcal{T}}$.

However, by the closure of T^+ , there is some rule $D \in T^+$ which subsumes $C_{\mathcal{T}}$. Clearly C does not subsume $C_{\mathcal{T}}$ (since $A \in \text{antec}(C) - \text{antec}(C_{\mathcal{T}})$), thus $D \in T^+ - \{C\}$, whence $T^+ - \{C\} \models C_{\mathcal{T}}$, a contradiction. ■

Thus by iterating the above procedure over all appropriate rules C and all appropriate antecedants of C , we may transform T into an equivalent non-recursive database. Notice that $\text{EXT}(T)$ plays no part in this transformation, which is thus immune to future changes in the extension.

4.3.5 Example. Let T contain the following rules.

1. $A_1 \wedge \neg E_3 \rightarrow B_1 \vee B_3$
2. $B_3 \wedge A_1 \wedge \neg E_0 \rightarrow B_1 \vee B_2$
3. $B_1 \wedge A_2 \rightarrow B_3$
4. $E_1 \wedge E_2 \rightarrow A_1$
5. $E_1 \rightarrow A_2 \vee A_3$
6. $E_0 \vee E_1$
7. $E_2 \vee E_3$

with $\ell(E_i) = 0$, $\ell(A_i) = 1$ and $\ell(B_i) = 2$.

Let us first start out by eliminating $C = \text{rule 3}$. The only cyclic tree for B_1 in $\mathcal{U}_{<2}(T - \{C\})$ is \mathcal{T}_1 (Figure 4.3.5) with $C_{\mathcal{T}_1} = A_1 \wedge A_2 \wedge \neg E_3 \rightarrow B_3$.

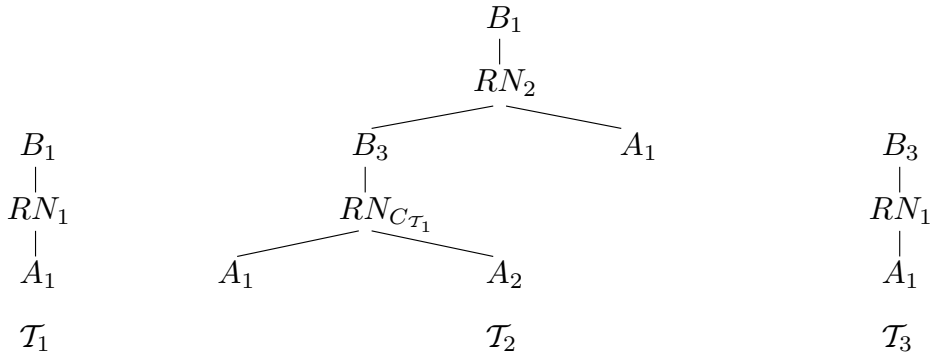


Figure 4.3.5.

We can also form a further cyclic tree, \mathcal{T}_2 for B_1 in $\mathcal{U}_{<2}((T - \{C\}) \cup \{C_{\mathcal{T}_1}\})$, but $C_{\mathcal{T}_2} = A_1 \wedge A_2 \wedge \neg E_3 \wedge \neg E_0 \rightarrow B_3 \vee B_2$ is subsumed by $C_{\mathcal{T}_1}$. Thus, $T \cup \{C_{\mathcal{T}_1}\}$ satisfies the appropriate closure property, and we may therefore replace C by $C_{\mathcal{T}_1}$.

We now move on to consider $D = \text{rule 2}$. In this case the cyclic tree \mathcal{T}_3 (Figure 4.3.5) yields the formula $D_{\mathcal{T}_3} = A_1 \wedge \neg E_0 \wedge \neg E_3 \rightarrow B_1 \vee B_2$. We could also employ the formula $C_{\mathcal{T}_1}$ instead of rule 1 in \mathcal{T}_3 , but the formula so generated (namely $A_1 \wedge A_2 \wedge \neg E_3 \wedge \neg E_0 \rightarrow B_1 \vee B_2$) is subsumed by $D_{\mathcal{T}_3}$. Thus the addition of $D_{\mathcal{T}_3}$ to $(T - \{C\}) \cup \{C_{\mathcal{T}_1}\}$ allows us to remove D .

It is easy to check that the perfect models of both T and $(T - \{C, D\}) \cup \{C_{\mathcal{T}_1}, D_{\mathcal{T}_3}\}$ are

$$\begin{aligned} &\{E_0, E_2\}, \{E_0, E_3\}, \{E_1, E_2, A_1, A_2, B_3, B_1\}, \\ &\{E_1, E_2, A_1, A_2, B_3, B_2\}, \{E_1, E_2, A_1, A_3, B_1\}, \\ &\{E_1, E_2, A_1, A_3, B_3, B_2\}, \{E_1, E_3, A_2\}, \text{ and } \{E_1, E_3, A_3\}. \end{aligned}$$

Notice that in this example, T did not increase in size during the transformation. In general we would expect an increase in size, as processing non-recursive databases is computationally simpler [Jo96] than processing recursive ones.

4.3.6 Example. Let T contain the following rules.

1. $A_1 \wedge \neg E_3 \rightarrow B_1 \vee B_4$
2. $B_4 \wedge A_1 \wedge \neg E_0 \rightarrow B_1 \vee B_2$
3. $B_1 \wedge A_2 \rightarrow B_3$
4. $E_1 \wedge E_2 \rightarrow A_1$
5. $E_1 \rightarrow A_2 \vee A_3$
6. $E_0 \vee E_1$
7. $E_2 \vee E_3$

with $\ell(E_i) = 0$, $\ell(A_i) = 1$ and $\ell(B_i) = 2$.

Suppose that we set out to eliminate $C = \text{rule 3}$, then the only cyclic tree for B_1 in $\mathcal{U}_{<2}(T - \{C\})$ is \mathcal{T}_1 (Figure 4.3.6) with $C_{\mathcal{T}_1} = A_1 \wedge A_2 \wedge \neg E_3 \rightarrow B_3 \vee B_4$.

The addition of $C_{\mathcal{T}_1}$ to T yields the further cyclic tree \mathcal{T}_2 (Figure 4.3.5), where $C_{\mathcal{T}_2} = A_1 \wedge A_2 \wedge \neg E_0 \wedge \neg E_3 \rightarrow B_2 \vee B_3$, and we may then replace C by $C_{\mathcal{T}_1}$ and $C_{\mathcal{T}_2}$.

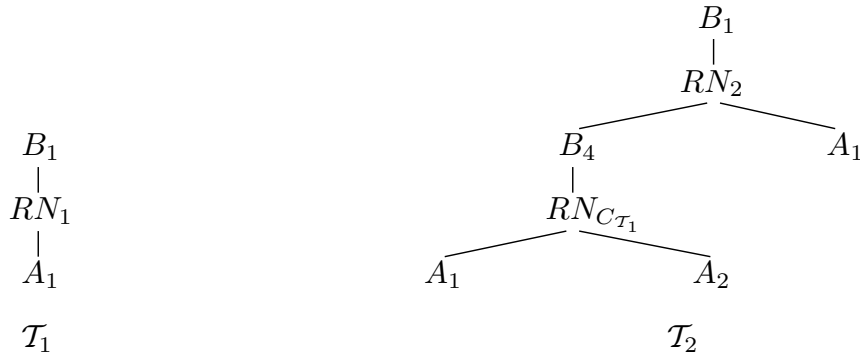


Figure 4.3.6.

4.4 Eliminating positive intensional sub-goals.

In this section we employ the results of Section 3 to illustrate the removal of positive sub-goals. Firstly it is worth pointing out the trivial fact that *extensional* positive sub-goals cannot be removed via a transformation which is immune to future changes in the extensional database. We thus concentrate on the removal of *intensional* positive sub-goals.

By the results of Section 4.3 we may assume that T is recursion free. Let C be a rule in T with $\ell(C) = \alpha$ such that $\text{antec}(C)$ contains the intensional predicate A . Let T^+ be formed from T by adding the rule $C_{\mathcal{T}}$:

$$\bigwedge (\text{antec}(C) - \{A\}) \wedge \bigwedge \overline{\mathcal{N}}(C) \wedge \bigwedge \{lab(N) \mid N \text{ is a leaf node in } \mathcal{T}\} \wedge$$

$$\bigwedge \{\neg Q \mid Q \in \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})\} \rightarrow \bigvee \text{conseq}(C)$$

for each cyclic tree \mathcal{T} for A in $\mathcal{U}_{<1}(T)$. Notice that since T is assumed recursion free, we have that $A \in \mathcal{L}_{<\alpha}$ and hence if D labels some rule node in \mathcal{T} , then $\ell(D) < \alpha$. In particular, $\text{Pred}(\mathcal{T}) \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{L}_{<\alpha}$.

As in Theorem 4.3.3, $T \models C_{\mathcal{T}}$, whence by Theorem 1.1.6, T and T^+ are equivalent.

4.4.1 Proposition. T^+ and $T^+ - \{C\}$ are equivalent.

Proof. By Theorem 1.1.6, it suffices to show that $T^+ - \{C\} \models C$. Let M be a perfect model of $T^+ - \{C\}$ such that $\text{antec}(C) \subseteq M \subseteq \mathcal{L} - \mathcal{N}(C)$. Let \mathcal{T} be a cyclic tree for A in $\mathcal{U}_{<1}(T^+ - \{C\})$ such that $\text{Pred}(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$.

Notice that $\ell(D) = \alpha$ for each rule in $D \in T^+ - T$, and hence since $\ell(A) < \alpha$, no rule node in \mathcal{T} is labelled with a rule in $T^+ - T$. Thus $\mathcal{T} \in \mathcal{U}_{<1}(T)$ and in particular $C_{\mathcal{T}} \in T^+ - \{C\}$.

But then $M \models C_{\mathcal{T}}$, whence $M \models \bigvee \text{conseq}(C)$. ■

4.4.2 Example. Let T be the following database (which resulted from the transformation in Example 4.3.5).

- | | | |
|---|---|-----------------------------------|
| 1. $A_1 \wedge \neg E_3 \rightarrow B_1 \vee B_3$ | 2. $A_1 \wedge \neg E_0 \wedge \neg E_3 \rightarrow B_1 \vee B_2$ | |
| 3. $A_1 \wedge A_2 \wedge \neg E_3 \rightarrow B_3$ | 4. $E_1 \wedge E_2 \rightarrow A_1$ | 5. $E_1 \rightarrow A_2 \vee A_3$ |
| 6. $E_0 \vee E_1$ | 7. $E_2 \vee E_3$ | |

with $\ell(E_i) = 0$, $\ell(A_i) = 1$ and $\ell(B_i) = 2$.

The only cyclic tree \mathcal{T} for A_1 in $\mathcal{U}_{<1}(T)$ has $\text{Pred}(\mathcal{T}) = \{E_1, E_2\}$, and $\mathcal{O}(\mathcal{T}) = \mathcal{N}(\mathcal{T}) = \emptyset$. We may thus replace each occurrence of A_1 in an antecedant by $E_1 \wedge E_2$.

Similarly we may replace each occurrence of A_2 by $E_1 \wedge \neg A_3$. The transformed database now has the form

- | | | |
|---|--|-------------------|
| 1. $E_1 \wedge E_2 \wedge \neg E_3 \rightarrow B_1 \vee B_3$ | 2. $E_1 \wedge E_2 \wedge \neg E_0 \wedge \neg E_3 \rightarrow B_1 \vee B_2$ | |
| 3. $E_1 \wedge E_2 \wedge \neg A_3 \wedge \neg E_3 \rightarrow B_3$ | 4. $E_1 \wedge E_2 \rightarrow A_1$ | |
| 5. $E_1 \rightarrow A_2 \vee A_3$ | 6. $E_0 \vee E_1$ | 7. $E_2 \vee E_3$ |

4.5 Eliminating negative intensional sub-goals.

In this section we employ the results of Section 3 to illustrate the removal of negative sub-goals. As in the preceding section, we can only eliminate intensional sub-goals if our transformation is to be immune to future changes in the extension.

Notice that if T^* contains no negative intensional sub-goals, then a model M of T^* is perfect iff M is minimal, and M_0 is a minimal model of T_0^* .

We approach this problem in two ways - firstly via a transformation directly in \mathcal{L} , and secondly by extending the language. The latter option does not cause the database to increase in size to the same degree as the former.

4.5.1 Definition. For each $P \in \text{INT}(\mathcal{L})$, let $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k$ ($k \geq 0$) list all cyclic trees for P in $\mathcal{U}_{<1}(T)$, then a *replacement* for P (in T) is a conjunct of the form $R_1 \wedge R_2 \wedge \dots \wedge R_k$ such that each $R_i \in \{\neg R \mid R \in \text{Pred}(\mathcal{T}_i) \cap \text{EXT}(\mathcal{L})\} \cup \mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i)$.

Note that if $k = 0$, then P has a single replacement - the empty conjunct - which is trivially true. The following lemma follows trivially from Theorems 3.1.5 and 3.1.6.

4.5.2 Lemma. Suppose that M is a perfect model of T , then for each $P \in \text{INT}(\mathcal{L})$, $P \in M$ iff for each replacement $R_1 \wedge R_2 \wedge \dots \wedge R_k$ for P in T we have that $M \not\models R_1 \wedge R_2 \wedge \dots \wedge R_k$.

4.5.3 Definition. Let $C \in \text{INT}(T)$, then a *replacement* for C is formed from C by replacing each intensional negative subgoal $\neg P$ in the body of C by a replacement for P .

Let T^* be the database formed from T by replacing each rule $C \in \text{INT}(T)$ by the set of its replacements. Notice that T^* contains no intensional negative sub-goals.

4.5.4 Theorem. T and T^* are equivalent.

Proof. In order to show that T and T^* are equivalent, we show that both T and T^* are equivalent to $T \cup T^*$. By Theorem 1.1.6 it suffices to show that (i) $T \models T^*$ and (ii) for each α , $T_{<\alpha}^* \models T_{\leq\alpha}$.

(i). Let M be a perfect model of T and suppose that $C^* \in T^* - T$ with $M \not\models C^*$. Let C^* be a replacement of the rule $C \in \text{INT}(T)$. But then $\text{antec}(C) \subseteq \text{antec}(C^*) \subseteq M$ and $\mathcal{N}(C) \cap \text{EXT}(\mathcal{L}) \subseteq \mathcal{N}(C^*)$ is disjoint from M as is $\text{conseq}(C) = \text{conseq}(C^*)$.

Suppose that $P \in \mathcal{N}(C) \cap \text{INT}(\mathcal{L})$. Since C^* is a replacement of C , there is a replacement of P contained within the body of C^* , and moreover, this replacement is true in M , whence by Lemma 4.5.2, $P \notin M$. Hence $M \cap \mathcal{N}(C) \cap \text{INT}(\mathcal{L}) = \emptyset$, and $M \not\models C$, a contradiction.

(ii). We proceed by induction on α . Clearly $T_{\leq 0}^* \models T_{\leq 0}$ (since $T_{\leq 0} = T_{\leq 0}^*$), thus suppose that $T_{<\alpha}^* \models T_{<\alpha}$. By (i) above, $T_{<\alpha} \models T_{<\alpha}^*$ and hence by Theorem 1.1.6, $T_{<\alpha}^*$ and $T_{<\alpha}$ are both equivalent (to $T_{<\alpha}^* \cup T_{<\alpha}$).

Suppose that $C \in T_\alpha$, and that M is a perfect model of $T_{\leq \alpha}^*$ such that $M \not\models C$. Firstly note that $M_{< \alpha}$ is a perfect model of $T_{< \alpha}^*$, and hence of $T_{< \alpha}$.

Let $P \in \mathcal{N}(C)$, then $\ell(P) < \ell(C) = \alpha$, and in particular, if K appears in some replacement for P then $\ell(K) < \alpha$. Since $P \notin M$ we may (by Lemma 4.5.2) find a replacement \mathcal{R}_P for P in $T_{< \alpha}$ such that \mathcal{R}_P is true in $M_{< \alpha}$ (and hence true in M).

Thus we may find a replacement C^* for C such that $M \models C^*$. ■

We now turn to our second option - a transformation in an extended language.

For each $P \in \text{INT}(\mathcal{L})$, introduce a new predicate P^* , with $\ell(P^*) = \ell(P)$, and define $\mathcal{L}^* = \{P^* \mid P \in \text{INT}(\mathcal{L})\}$. For each rule $C \in T$

$$A_1 \wedge A_2 \wedge \dots \wedge A_h \wedge \neg A_{h+1} \wedge \neg A_{h+2} \wedge \dots \wedge \neg A_{h+r} \rightarrow B_1 \vee B_2 \vee \dots \vee B_k$$

let C^* be the rule obtained by replacing each $\neg A_{h+i}$, where A_{h+i} is intensional, by A_{h+i}^* .

Let $P \in \text{INT}(\mathcal{L})$ with $\ell(P) = \alpha$. For each cyclic tree \mathcal{T} for P in $\mathcal{U}_{< 1}(T)$, introduce a new predicate $Q_{\mathcal{T}}$ with $\ell(Q_{\mathcal{T}}) = \alpha$.

Let $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_k$ ($k \geq 0$) list all cyclic trees for P in $\mathcal{U}_{< 1}(T)$, then C_P denotes the rule

$$Q_{\mathcal{T}_1} \wedge Q_{\mathcal{T}_2} \wedge \dots \wedge Q_{\mathcal{T}_k} \rightarrow P^*.$$

Let $T^* = \{C^* \mid C \in T\} \cup T'$ where

$$\begin{aligned} T' = & \{C_P \mid P \in \text{INT}(\mathcal{L})\} \cup \\ & \{K \rightarrow Q_{\mathcal{T}} \mid \mathcal{T} \in \mathcal{U}_{< 1}(T), \text{lab}(\text{root}(\mathcal{T})) \in \text{INT}(\mathcal{L}), \\ & K \in \{\neg R \mid R \in \text{Pred}(T) \cap \text{EXT}(\mathcal{L})\} \cup \mathcal{O}(T) \cup \mathcal{N}(T)\}. \end{aligned}$$

We aim to show that T and T^* are equivalent in the sense that (Theorem 4.5.8) for every perfect model $M \subseteq \mathcal{L}$ of T , there is a perfect model M^* of T^* such that $M^* \cap \mathcal{L} = M$, and conversely (Theorem 4.5.9) that if M^* is a perfect model of T^* , then $M^* \cap \mathcal{L}$ is a perfect model of T .

4.5.6 Definition. If $M \subseteq \mathcal{L}$, then let $cl_{\leq \alpha}(M)$ ($cl_{< \alpha}(M)$) denote the closure of M under $T'_{\leq \alpha}$ ($T'_{< \alpha}$).

i.e., $cl_{\leq \alpha}(M) \cap \mathcal{L} = M$, and if $R \notin \mathcal{L}$ with $\ell(R) \leq \alpha$, then $R \in cl_{\leq \alpha}(M)$ iff $T'_{\leq \alpha} \cup \{P \in \mathcal{L} \mid P \in M\} \cup \{\neg Q \mid Q \in \text{EXT}(\mathcal{L}) - M\} \models R$.

Note (1). If $\ell(Q_{\mathcal{T}}) \leq \alpha$, then $Q_{\mathcal{T}} \in cl_{\leq \alpha}(M)$ iff $M \models \bigvee \{\neg R \mid R \in \text{Pred}(T) \cap \text{EXT}(\mathcal{L})\} \vee \bigvee \mathcal{O}(T) \vee \bigvee \mathcal{N}(T)$.

(2). If $\beta \leq \alpha$, then $(cl_{\leq \alpha}(M))_{\leq \beta} = cl_{\leq \beta}(M_{\leq \beta})$.

(3). $cl_{\leq 0}(M) = M$ (since $T'_0 = \emptyset$).

4.5.7 Lemma. If $M \subseteq \mathcal{L}_{\leq \alpha}$ is a perfect model of $T_{\leq \alpha}$, then $cl_{\leq \alpha}(M) \cap \mathcal{L}^* = \{P^* \mid P \in \text{INT}(\mathcal{L})_{\leq \alpha} - M\}$.

Proof (\rightarrow). If $P \in \text{INT}(\mathcal{L})_{\leq \alpha} \cap M$, then by Theorem 3.1.5, there is a cyclic tree $\mathcal{T} \in \mathcal{U}_{<1}(T)$ for P such that $M \not\models \bigvee \{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$, whence $P^* \notin \text{cl}_{\leq \alpha}(M)$.

(\leftarrow). If $P \in \text{INT}(\mathcal{L})_{\leq \alpha} - M$, then for each unfactored cyclic tree $\mathcal{T} \in \mathcal{U}_{<1}(T)$ for P we must have (by Theorem 3.1.6) that $M \models \bigvee \{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \vee \bigvee \mathcal{O}(\mathcal{T}) \vee \bigvee \mathcal{N}(\mathcal{T})$, whence $P^* \in \text{cl}_{\leq \alpha}(M)$. ■

4.5.8 Theorem. If $M \subseteq \mathcal{L}_{\leq \alpha}$ is a perfect model of $T_{\leq \alpha}$, then $\text{cl}_{\leq \alpha}(M)$ is a perfect model of $T_{\leq \alpha}^*$.

Proof. The result is trivially true for $\alpha = 0$, thus we proceed by induction. Suppose that the result holds for all $\beta < \alpha$, thus in particular, $(\text{cl}_{\leq \alpha}(M))_{< \alpha} = \text{cl}_{< \alpha}(M_{< \alpha})$ is a perfect model of $T_{< \alpha}^*$.

Firstly it is easy to check that $\text{cl}_{\leq \alpha}(M) \models T_{\leq \alpha}^*$ by virtue of Lemma 4.5.7. If $\text{cl}_{\leq \alpha}(M)$ is not perfect, then (by the perfectness of $(\text{cl}_{\leq \alpha}(M))_{< \alpha}$) we may find an $M' \subset \text{cl}_{\leq \alpha}(M)$ such that $M' \models T_{\leq \alpha}^*$, and $M'_{< \alpha} = (\text{cl}_{\leq \alpha}(M))_{< \alpha}$.

Firstly note that $M' \cap \mathcal{L} \subset M$ (for if $M' \cap \mathcal{L} = M$, then $\text{cl}_{\leq \alpha}(M) = \text{cl}_{\leq \alpha}(M' \cap \mathcal{L}) \subseteq M' \subset \text{cl}_{\leq \alpha}(M)$), and $M' \cap \mathcal{L}_{< \alpha} = M \cap \mathcal{L}_{< \alpha}$.

Pick $P \in M_{\alpha} - M'$, then there is a cyclic tree $\mathcal{T} \in \mathcal{U}_{<1}(T)$ for P in $T_{\leq \alpha}$ such that $M \not\models \bigvee \{\neg R \mid R \in \text{Pred}(\mathcal{T})\} \vee \bigvee \mathcal{O}(\mathcal{T}) \vee \bigvee \mathcal{N}(\mathcal{T})$.

We transform \mathcal{T} into a cyclic tree in $T_{\leq \alpha}^*$ as follows. If RN_C appears in \mathcal{T} , then replace RN_C by RN_{C^*} . If A^* is a child node of RN_{C^*} , then $A \in \mathcal{N}(\mathcal{T}) \subseteq \mathcal{L}_{< \alpha} - M$. The child node of A^* is labelled with C_A . If $Q_{\mathcal{T}_i}$ labels some child node of C_A , then (by Theorem 3.1.6) there is some $K \in \{\neg R \mid R \in \text{Pred}(\mathcal{T}) \cap \text{EXT}(\mathcal{L})\} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$ such that $M_{< \beta} \models K$ (whence $M' \models K$), and we thus label the child node of $Q_{\mathcal{T}_i}$ with $K \rightarrow Q_{\mathcal{T}_i}$.

Thus $\mathcal{O}(\mathcal{T}^*) \cup \mathcal{N}(\mathcal{T}^*) = \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{L} - M \subseteq \mathcal{L} - M'$, and $\{lab(N) \mid N \text{ is a leaf node in } \mathcal{T}^*\} \subseteq M'$. By Theorem 3.1.6, $P \in M'$, a contradiction. ■

4.5.9 Theorem. If M^* is a perfect model of $T_{\leq \alpha}^*$, then

- (i) for each $P \in (\text{INT}(\mathcal{L}))_{\leq \alpha}$, $|M^* \cap \{P, P^*\}| = 1$, and
- (ii) $M^* \cap \mathcal{L}$ is a perfect model of $T_{\leq \alpha}$.

Proof. Again the result is trivial for $\alpha = 0$, thus we proceed by induction. Suppose that the result holds for all $\beta < \alpha$.

If $C \in T_{\alpha}$ with $\text{antec}(C) \subseteq M^* \cap \mathcal{L}$ and $\mathcal{N}(C) \cap M^* \cap \mathcal{L} = \emptyset$, then by inductive hypothesis, $P^* \in M^*$ for each $P \in \mathcal{N}(C) \cap \text{INT}(\mathcal{L})$. But then since $M^* \models T_{\leq \alpha}^*$ we must have that $M^* \cap \text{conseq}(C) \neq \emptyset$. Thus $M^* \cap \mathcal{L} \models T_{\alpha}$.

If $M^* \cap \mathcal{L}$ is not perfect, then (by inductive hypothesis) we may find an $M' \subset M^* \cap \mathcal{L}$ such that $M'_{< \alpha} = M^* \cap \mathcal{L}_{< \alpha}$ and M' is a perfect model of $T_{\leq \alpha}$. But then $\text{cl}_{\leq \alpha}(M')$ is a perfect model of $T_{\leq \alpha}^*$ with $\text{cl}_{\leq \alpha}(M') \subset \text{cl}_{\leq \alpha}(M^* \cap \mathcal{L}) \subseteq M^*$, contradicting the minimality of M^* .

Thus $M^* = \text{cl}_{\leq \alpha}(M^* \cap \mathcal{L})$. By Lemma 4.5.7, for each $P \in \mathcal{L}$, $P^* \in M^*$ iff $P \notin M^*$. ■

4.5.10 Example. Let T contain the following rules.

1. $A_2 \wedge \neg E_5 \rightarrow B_1 \vee B_4$
2. $B_4 \wedge \neg A_1 \rightarrow B_1 \vee B_2$
3. $B_1 \wedge A_2 \rightarrow B_3$
4. $A_2 \wedge E_1 \wedge E_2 \rightarrow A_1$
5. $E_1 \wedge \neg E_5 \rightarrow A_2 \vee A_3$
6. $E_0 \vee E_1$
7. $E_2 \vee E_3$

The only cyclic tree in $\mathcal{U}_{<1}(T)$ for A_1 is \mathcal{T}_1 (Figure 4.5.10) with $\mathcal{O}(\mathcal{T}_1) = \{A_3\}$ and $\mathcal{N}(\mathcal{T}_1) = \{E_5\}$.

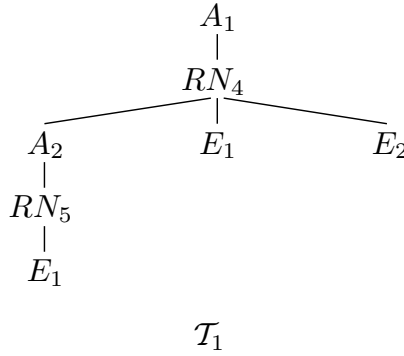


Figure 4.5.10.

We may thus replace rule 2 by the four rules $B_4 \wedge \neg E_2 \rightarrow B_1 \vee B_2$, $B_4 \wedge \neg E_1 \rightarrow B_1 \vee B_2$, $B_4 \wedge A_3 \rightarrow B_1 \vee B_2$, and $B_4 \wedge E_5 \rightarrow B_1 \vee B_2$.

Let T^* be the transformed database. It is clear that if \mathcal{T} is a cyclic tree for B_4 in $\mathcal{U}_{<1}(T^*)$, then $\text{Pred}(\mathcal{T}) = \{B_4, E_1, A_2\}$, $\mathcal{O}(\mathcal{T}) = \{B_1, A_3\}$ and $\mathcal{N}(\mathcal{T}) = \{E_5\}$. In particular, if \mathcal{P} is a weakly cyclic cover in T^* containing $\neg B_4$, then $\{\neg E_1, A_3, E_5\} \subseteq \mathcal{P}$. Thus, by Theorem 4.2.2, each of the last three of these rules is redundant.

We finally remark again that the transformation proposed in this section is immune to future changes in $\text{EXT}(T)$. In order to eliminate all negative sub-goals, we could employ a similar procedure to that detailed above, using unfactored cyclic tree rather than trees in $\mathcal{U}_{<1}(T)$. Such a transformation could not be immune to future changes in $\text{EXT}(T)$.

§5. CONCLUSIONS AND FUTURE RESEARCH.

We have introduced the notion of a conjunctive answer to a deductive database query, and shown that such answers provide more information about a query than do their disjunctive counterparts. We have also argued that such answers do not suffer from some of the pitfalls encountered with disjunctive answers. Somewhat surprisingly the

problem of computing conjunctive answers is found to be a special case of performing view insertions.

We have also discussed the rather technical relationship between cyclic covers and conjunctive answers, view insertions and pre-processing of the intensional database. Methods exist for the construction of the former [Jo95], and such methods can be amended to the problems at hand.

As regards open problems and future research, the following issues are worthy of note.

(1) *When is it appropriate to compute conjunctive answers?*

The examples of Section 2.2 lead to the intuition that conjunctive answers are computationally preferable when strong interdependencies hold within the models of the database. Higher level meta-knowledge might be employed to give guidance as to which type of answer is preferable, and this was illustrated in Examples 2.2.4 and 2.2.5.

By the results of Section 2, conjunctive answers provide more information about a query than do their disjunctive counterparts. To what extent is it desirable or feasible to extend this process in order to extract more complex data relationships from the database?

(2) *What is the best way to compute conjunctive answers?*

Example 3.4.12 does suggest a method of computing view insertions and hence conjunctive answers, although for the latter the method suggested is slightly indirect. For conjunctive answers (to the query $?Q$) we need (Theorem 3.3.1) to compute cyclic covers \mathcal{C} such that $Q \subseteq \{R \in \mathcal{L} \mid \neg R \in \mathcal{C}\} \cup \mathcal{C}$. Generating cyclic covers for elements of Q can be achieved (at the propositional and first order levels) using the results of [Jo95]. The main problem however is guiding the search. Whilst constructing such a \mathcal{C} , how do we decide which elements of Q should go in $\{R \in \mathcal{L} \mid \neg R \in \mathcal{C}\}$ and which in \mathcal{C} (other than trying both options)? Indeed if we apply the method suggested in Example 3.4.12 to compute conjunctive answers using the reduction given in Section 2.3, then both options are considered.

(3) *The first order level.*

Of course “real” deductive databases are not variable free. This then raises the question as to how we extend the results of the preceding sections to include databases containing variables.

Firstly it should be noted that as in [Jo96, Theorem 6.4.7] we can easily show that if \mathcal{T} is a cyclic tree in $\mathcal{U}_{<1}(T)$ with $\{lab(N) \mid N \text{ is a predicate leaf node in } \mathcal{T}\} \subseteq \bigcup \text{EXT}(T)$ (cf. Definition 3.4.6), then \mathcal{T} is ground (even for first order databases) as a result of the fact that $\text{EXT}(T)$ is ground.

The methods of [Jo95] (for computing extended deduction trees) and [Jo97] (for computing covers) easily yield methods of computing cyclic covers at the first order level provided that the database satisfies constraints such as those given in 4(a) below or the more general constraints given in [Jo97, Section 4].

(4) *Restrictions on T .*

It may well be worthwhile examining restrictions on T which make the computation more efficient. Various such restrictions are considered in [Jo95, Jo96, Jo97], some of which are (for each $C \in T$):

- (a) $\text{antec}(C) \cup \mathcal{N}(C)$ and $\text{conseq}(C)$ share the same set of variables.
- (b) If C is intensional, then $|\text{conseq}(C)| = 1$.
- (c) If $R(\mathbf{t}) \in \mathcal{N}(C)$, then R is either extensional or definite.

§6. RELATED WORK.

Perfect models were introduced by Przymusinski [Pr88, Pr89] and obviously extend minimal models [Gr86, He88] to the stratified case, where they are generally regarded as providing one of the most natural model-theoretic declarative semantics, assuming minimalist reasoning.

Whilst we believe that the vast majority of naturally occurring deductive databases are indeed stratified, there is currently much research interest in unstratified databases (and logic programs) to which the semantics defined by weakly perfect models [Pr95], stable models [Ge88, Pr90, Pr94], stationary (otherwise known as partial or 3-valued stable) models [Pr94], and well-founded models [Pr90, Pr94, VanG91] apply. Each of these coincides with the perfect model semantics in the stratified case [Pr94, Pr95, VanG91]. These concepts were extended to the disjunctive setting via the disjunctive stable semantics [Pr91], stationary semantics [Pr90a, Pr91a], and the static semantics [Pr96, Bra96b].

Constructing perfect models and computing answers under such semantics is computationally hard [Jo96]. Bottom-up methods for computing perfect models of ground databases are presented in [Ya94, Fe95a, Fe95b] using the notion of a model-tree; top down methods for first order databases appear in [Jo95, Jo96]. In particular, the notion of a cyclic tree is introduced in [Jo96] as a means of computing perfect model membership, the notion of a cover was introduced in [Jo97] as a means of attacking the view update problem for positive databases, and the notion of a cyclic cover was introduced in [Jo95] as a means of query processing under the perfect model semantics. Methods for constructing cyclic trees and cyclic covers are presented in [Jo95, Jo96].

The view update problem is a well established topic of research in the context of databases [Ab85, Ab88, Ab93, Ba81, Da82, Go88, Ke91, Sc91], deductive databases [At91, At92, Bry90, De90, Fa83, Gr93, Kr92, Ma88, Ro89, To88], and more general logical databases [Wi90].

Our study of the view update problem was inspired by [Gr93], where the notion of a restricted SLD - tree is used to attack the view update problem at the propositional level in the case when the intensional database contains only stratified definite rules. Restricted SLD - trees deliver the required update in disjunctive (rather than conjunctive) normal form, and thus a further transformation is necessary before insertion into the extension. Another disadvantageous feature of restricted SLD - trees is that they may contain infinite branches at the first order level.

The methods of [Gr93] are extended in [Fe96] to yields methods for performing view updates in ground stratified databases with definite rules, integrity constraints and denial rules under the perfect model semantics. The methods presented there-in have bottom up features, requiring the computation of all minimal models of the extension. [Fe96] also comment that their methods are extendable to first order databases satisfying the condition give in Section 5, part 4(a). The current paper makes no requirement that we compute perfect models (or indeed minimal models of the extension) in their entirety. We employ purely top-down methods, and make no requirement that database rules should be definite, although we do not discuss integrity constraints or denial rules. The results presented in [Jo97] for view insertions are subsumed by the results of the present paper for ground databases. [Jo97] does however presents methods the first order databases (and in particular for databases more general than those prescribed in the condition given in Section 5, part 4(a)), which we do not attempt in any depth here.

Partial evaluation was introduced into logic programming in [Ko81], and has been extensively studied in that setting (eg., [Ll87]). In particular we mention the work of Brass and Dix [Bra94, Bra95, Bra95a] who consider bottom-up partial evaluation using an unfolding operation, the “Generalised Principle of Partial Evaluation” (GPPE) to transform the database into a “residual program” containing only ground conditional facts of the form $\bigwedge_i \neg C_i \rightarrow \bigvee_j A_j$ (in which positive sub-goals are completely absent). The residual program is shown to be equivalent to the original program under a variety of semantics including the perfect (for stratified databases), disjunctive stable and well-founded semantics [Bra96, Bra96a].

The GPPE differs from our own methods in that it is not immune to extensional database updates - which thus necessitate an expensive re-transformation - although it could easily be amended to take into account this requirement. When applied to the examples given in 4.3.5, 4.3.6 and 4.4.2, the GPPE yields the same result as that given by our methods. We have been unable to show however that this is always the case, or to deduce any inherent relationship between the GPPE and our own methods.

Query compilation is discussed in the deductive setting in for example [He84, He88, Jo96], although these results do not cover non-positive indefinite databases.

References

- [Ab85] S. Abiteboul and G. Grahne, Update semantics for incomplete databases, in : Proc 11th VLDB, (1985), 1-12.
- [Ab88] S. Abiteboul, Updates, a new frontier, Proc. ICDT '88, Bruges, Springer LNCS vol. 326, (Springer, Berlin, 1988), 1-18.
- [Ab93] S. Abiteboul, S. Cluet and T. Milo, Querying and updating the file, in : Proc. 19th VLDB, (1993), 73-84.
- [At91] P. Atzeni and R. Torlone, Solving ambiguities in updating deductive databases, in : Mathematical Foundations of Database Systems, Lecture Notes in Computer Science, vol. 495 (Springer, Berlin, 1991), 104-118.
- [At92] P. Atzeni and R. Torlone, Updating intensional predicates in datalog, Data and

Knowledge Engineering, vol. 8 (1992), 1-17.

- [Ba81] F. Bancilhon and N. Spyrtos, Update semantics of relational views, *ACM Transactions on Database Systems*, vol. 6 (1981), 557 - 575.
- [Bra94] S Brass and J. Dix, A disjunctive semantics based upon unfolding and bottom-up evaluation, in : B Wolfinger, (ed.), *Innovationen bei Rechen- und Kommunikationssystemen (IFIP- Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases)*, (Springer, 1994), 83-91.
- [Bra95] S. Brass and J. Dix, Disjunctive semantics based upon partial and bottom-up evaluation, in : L. Sterling (ed.), *Proc. 12th Int'l Conf. on Logic Programming*, Tokyo (MIT Press).
- [Bra95a] S. Brass and J. Dix, A general approach to bottom-up computation of disjunctive semantics, in : *Non-monotonic Extensions of Logic Programming*, Springer Lecture Notes in Artificial Intelligence, vol. 927 (1995), 127-155.
- [Bra96] S. Brass and J. Dix, Characterisations of the disjunctive stable semantics by partial evaluation, *J. Logic Programming* (to appear).
- [Bra96a] S. Brass and J. Dix, Characterisations of the disjunctive well-founded semantics : confluent calculi and iterated GCWA, in L M Pereira, J J Alferes and E Orłowska (eds), *Logics in Artificial Intelligence*, Springer Lecture Notes in Computer Science, vol. 1126 (1996), 268-283.
- [Bra96b] S. Brass, J. Dix, I Niemelä and T Przymusiński, Comparison and efficient computation of the static and disjunctive well-founded semantics, preprint.
- [Bry90] F. Bry, Intensional updates : abduction via deduction, in : *Proc. 7th Int'l Conference on Logic Programming* (1990), 561-578
- [Da82] U. Dayal and P. Bernstein, On the correct translation of update operations on relational view, *ACM Transactions on Database Systems*, vol. 8 (1982), 382 - 416.
- [De90] H. Decker, Drawing updates from derivations, in : S. Abiteboul and P.C. Kanellakis (Eds.), *Proceedings of the 3rd Int'l Conference on Database Theory*, Paris, Lecture Notes in Computer Science, vol. 470 (Springer, Berlin, 1990), 437-451.
- [Fa83] R. Fagin, J. Ullman and M. Vardi, On the semantics of updates in databases, in *Proceedings of the 2nd ACM Symposium on the Principles of Database Systems*, (1983), 352-365.
- [Fe95a] J. Fernández and J. Minker, Computing perfect models of disjunctive stratified databases, *J. Logic Programming*, vol. 25 (1995), 33-51.
- [Fe95b] J. Fernández and J. Minker, Computing perfect and stable models using ordered model trees, *Computational Intelligence*, vol. 11 (1995), 89-112.
- [Fe96] J. Fernández, J. Grant and J. Minker, Model theoretic approach to view updates in deductive databases, *J. Automated Reasoning*, vol. 17 (1996), 171-197.
- [Ge88] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, in : R. Kowalski and K. Bowen (eds.), *Proc. 5th Int'l Conference on Logic Programming*, Seattle (1988), 1070-1080.
- [Gr86] J. Grant and J. Minker, Answering queries in indefinite databases and the null value problem, *Advances in Computing Research*, vol. 3 (1986), 247-267.
- [Gr93] J. Grant, J. Horty, J. Lobo and J. Minker, View updates in stratified disjunctive databases, *J. Automated Reasoning*, vol. 11 (1993), 249-267.

- [Go88] G. Gottlob, P. Paolini and R. Zicari, Properties and update semantics of consistent views, *ACM Transactions of Database Systems*, vol. 13 (1988), 486 - 524.
- [He84] L. Henschen and S. Naqvi, On compiling queries in recursive first-order databases, *J. Association of Computing Machinery*, vol. 31 (1984), 47-85.
- [He88] L. Henschen and H. Park, Compiling the GCWA in indefinite databases, in : J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, (Morgan Kaufman, Washington, 1988).
- [Jo95] C. A. Johnson, Query processing in indefinite stratified databases, Computer Science technical report TR95-14 (Keele University). Available at <http://www.keele.ac.uk/depts/cs/cshome.html>
- [Jo95a] C. A. Johnson, Top down query processing in indefinite stratified databases, submitted, *Data and Knowledge Engineering*.
- [Jo96] C. A. Johnson, On computing minimal and perfect model membership, *Data and Knowledge Engineering*, vol. 18 (1996), 225-276.
- [Jo97] C. A. Johnson, Deduction trees and the view update problem in indefinite deductive databases, *J. Automated Reasoning*, to appear.
- [Jo97a] C. A. Johnson, On cyclic covers and perfect models, Computer Science technical report TR97-04 (Keele University). Available at <http://www.keele.ac.uk/depts/cs/cshome.html>
- [Ka90] A. Kakas and P. Mancarella, Database updates through abduction, in : *Proc. 16th VLDB*, (1990), 650 - 661.
- [Ke86] A. Keller, Choosing a view update translator by dialog at view definition time, in : *Proc. 12th VLDB*, (1986), 467 - 474.
- [Ke91] A. Keller, T. Barsalou, N. Siambela and G. Wiederhold, Updating relational databases through object-based views, in : *Proc. ACM SIGMOD '91* (1991), 248 - 257.
- [Ko81] A Specification of an Abstract Prolog Machine and tis Application to Partial Evaluation, Ph. D. thesis, University of Linkoping, 1981.
- [Kr92] M. Kramer, G. Lausen and G. Saake, Updates in a rule-based language for objects, in : *Proc. 18th VLDB*, Vancouver (1992), 251-262.
- [Lb92] J. Lobo, J. Minker, and A. Rajasekar, *Foundations of Disjunctive Logic Programming*, (MIT Press, Cambridge, Massachusetts, 1992).
- [Ll87] J. Lloyd and J. Shepherdson, Partial evaluation in logic programming, *Journal of Logic Programming*, vol. 11 (1991), 217-242.
- [Ma88] S. Manchanda and D. Warren, A logic based language for database updates, in : J. Minker, ed., *Foundations of Deductive Databases*, 363-394, (Morgan Kaufmann, Washington, 1988).
- [Pr88] T. Przymusinski, On the declarative semantics of deductive databases and logic programs, in: J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, (Morgan Kaufman, Washington, 1988).
- [Pr89] T. C. Przymusinski, On the declarative and procedural semantics of logic programs, *J. Automated Reasoning*, vol. 5 (1989), 167-205.
- [Pr90] T. C. Przymusinski, The well-founded semantics coincides with the three-valued stable semantics, *Fundamenta Informaticae*, vol. 13 (1990), 445-464.

- [Pr90a] T. C. Przymusiński, Stationary semantics for disjunctive logic programs and deductive databases, in : S Debray and M Hermenegildo (eds), Proc. North American Logic Programming Conference, Austin, Texas (MIT Press, 1990), 40-59.
- [Pr91] T. C. Przymusiński, Stable semantics for disjunctive programs, *New Generation Computing*, vol. 9 (1991), 401-424.
- [Pr91a] T. C. Przymusiński, Semantics of disjunctive logic programs and deductive databases, in Proc. 2nd Int'l Conf. on Deductive and Object-Oriented Databases, DOOD'91, Munich, (Springer, 1991).
- [Pr94] T. C. Przymusiński, Well-founded and stationary models of logic programs, *Annals of Mathematics and Artificial Intelligence*, vol. 12 (1994), 141-187.
- [Pr95] T. C. Przymusiński and H. Przymusińska, Weakly stratified logic programs, preprint. (An extended abstract appears in : R. Kowalski and K. Bowen (eds.), Proc. 5th Int'l Conference on Logic Programming, Seattle (1988), 1106-1120.)
- [Pr96] T. C. Przymusiński, Static semantics for normal and disjunctive logic programs, pre-print.
- [Ra90] A. Rajasekar and J. Minker, On stratified disjunctive programs, *Annals of Mathematics and Artificial Intelligence*, vol. 1 (1990), 339-357.
- [Ri91] E. Rich and K. Knight, *Artificial Intelligence*, (McGraw Hill, 1991).
- [Ro89] F. Rossi and S. Naqvi, Contributions to the view update problem, in : Proceedings of the 6th Int'l Conference on Logic Programming, (1989), 398-415.
- [Sc91] M. Scholl, C. Laasch and M. Tresch, Updatable views in object oriented database systems, in Proc. 2nd Int'l Conference on Deductive and Object Oriented Databases, Munich, Lecture Notes in Computer Science, vol. 566 (Springer, Berlin, 1991), 189-207.
- [To88] A. Tomasic, View update annotation in definite deductive databases, Proc ICDT '88 (1988), 338 - 352.
- [VanG91] A. van Gelder, K. Ross and J. Schlipf, The well-founded semantics for general logic programs, *J. Assoc. for Computing Machinery*, vol. 38 (1991), 620-650.
- [Wi90] M. Winslett, *Updating Logical Databases*, Cambridge Tracts in Theoretical Computer Science, vol. 9 (Cambridge University Press, 1990).
- [Ya94] A. Yahya, J. Fernández and J. Minker, Ordered model trees : A normal form for disjunctive deductive databases, *J. Automated Reasoning*, vol. 13 (1994), 117-143.