

C A Johnson

School of Computing
University of Plymouth
Plymouth, PL4 8AA
England

cjohnson@plymouth.ac.uk

Abstract. A method is presented for computing minimal answers of the form $\bigvee \mathcal{A}$ in disjunctive deductive databases under the disjunctive stable model semantics. Such answers are constructed by repeatedly extending partial answers. Our method is complete (in that every minimal answer can be computed) and does not admit redundancy (in the sense that every partial answer generated can be extended to a minimal answer), thus no non-minimal answer is generated. The method does not (necessarily) require the computation of models of the database in their entirety. Compilation is proposed as a tool by which problems relating to computational efficiency and the non-existence of disjunctive stable models can be overcome. The extension of our method to other semantics is also considered.

Other forms of database pre-processing are also considered, and three transformations are presented mapping a database onto an equivalent positive database, normal database, and set of conditional facts.

Keywords: Disjunctive deductive databases, minimal answers, perfect models, disjunctive stable models, cyclic sets, strong covers, compilation, database pre-processing.

Introduction

A propositional disjunctive deductive database [Gr86, He88, Lb92] consists of logical rules of the form

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

where each A_i, B_j is a predicate. A *query* is an expression of the form $?\bigvee \mathcal{H}$, where \mathcal{H} is a set of predicates, and an *answer* to $?\bigvee \mathcal{H}$ is a disjunction of predicates $\bigvee \mathcal{A}$ (where $\mathcal{A} \subseteq \mathcal{H}$) which is logically implied (under the chosen semantics) by the database.

For example suppose that a company has a number of geographically distributed sites s_1, s_2, \dots, s_k , and a database recording the location (i.e., site) where each resource is located. A query $?located_at(r, x)$ to determine which site resource r is located at, when rewritten in propositional terms, then has the form $?\bigvee_{i \leq k} located_at(r, s_i)$. If the database contains disjunctive (i.e., incomplete) information, then the answer to the query might also be disjunctive. Suppose that the query processor retrieves the answers

$$\begin{aligned} & located_at(r, s_1) \vee located_at(r, s_2), \\ & located_at(r, s_1) \vee located_at(r, s_2) \vee located_at(r, s_3), \\ & located_at(r, s_1) \vee located_at(r, s_2) \vee located_at(r, s_4), \dots, etc. \end{aligned}$$

We may then conclude from the “minimal” answer $located_at(r, s_1) \vee located_at(r, s_2)$ that r is located at either s_1 or s_2 . The other (non-minimal) answers being generated are logically weaker than this, and tell us nothing new, i.e., they are redundant.

In addition to being redundant, a non-minimal answer is potentially misleading. Suppose in our example that the end-user employs the answer $located_at(r, s_1) \vee located_at(r, s_2) \vee located_at(r, s_4)$. They will of course conclude that r is located at either s_1, s_2 or s_4 , but in addition they may well infer that r is *possibly* located at s_4 (the intuitive justification being that if this was not the case, then the query processor would not have presented this answer). This inference may then lead the user to instigate a (manual) search for r at site s_4 . This (pointless) search is of course based upon an invalid inference, which in turn was a direct result of the non-minimality of the answer employed.

Thus in order to provide the user with the best possible information (in response to their query) we wish to provide them with minimal answers. Note however that in a real environment, the set of (all) answers to a query will typically consist of a (relatively) small number of minimal answers, and an *extremely* large number of non-minimal ones. In particular, we certainly cannot expect users to manually extract minimal answers from a larger set of answers, and we therefore wish to present them with only minimal answers.

Of course one obvious way to compute minimal answers is to compute all answers, and then eliminate the non-minimal answers by subsumption. Methods for achieving this are well known. For example, in a positive database we can derive disjunctions of predicates using the hyper-resolution operator:

$$\frac{\bigwedge_{i \leq r} A_i \rightarrow \bigvee Q; A_i \vee \bigvee \mathcal{P}_i (i \leq r)}{\bigvee (Q \cup \bigcup_{i \leq r} \mathcal{P}_i)}$$

and this may be achieved by simple forward application, backward application [Ra89, Lb92], or by a combination of the two [Jo98]. In each case however there is the possibility that non-minimal answers will be generated. For example if our database consists

of the rules $\{A \vee B, A \rightarrow C, B \rightarrow D, C \rightarrow D\}$, then in order to generate the minimal answer D , forward application of the hyper-resolution operator above *must* generate a non-minimal answer, i.e., $A \vee D$, $B \vee D$ or $C \vee D$ (and we will see (in Section 9) that this is a general characteristic of this particular hyper-resolution operator).

In addition, it is evident that any query answering method will always risk generating non-minimal answers if it is based upon the construction of (a structure which in effect encodes) a proof of the answer being generated: clearly the existence of such a proof might not preclude the possibility of a proof of some smaller disjunction using other rules within the database.

Other approaches to query processing are of course well known (e.g., [Bra95, Fe95, Fe95a, Ya96, Ya02]), but in each case non-minimal answers may be generated.

The purpose of this paper is to present a query answering method which generates only minimal answers. Our approach differs from existing approaches in two fundamental ways. Firstly, each answer $A_1 \vee A_2 \vee \dots \vee A_n$ is constructed iteratively as a sequence $A_1, A_1 \vee A_2, A_1 \vee A_2 \vee A_3, \dots, A_1 \vee A_2 \vee \dots \vee A_n$, and secondly, for each disjunction $A_1 \vee A_2 \vee \dots \vee A_r$ generated, we (immediately) verify that $A_1 \vee A_2 \vee \dots \vee A_r$ can indeed be extended to a minimal answer, thus in particular, no non-minimal answers will be generated. Our method is of course complete, in the sense that every minimal answer can be generated.

For the most part we work under the disjunctive stable model semantics [Pr91] (although the extension of our results to other semantics is considered). In this context, the above-mentioned verification entails the construction of a suitable set of *cyclic strong covers* [Jo99a], each of which can be viewed as defining a partial disjunctive stable model.

Our method combines bottom-up and top-down features: our iterative construction of minimal answers has a bottom-up flavour, whilst the construction of individual cyclic strong covers is always top-down.

For the sake of simplicity of presentation we concentrate for the most part on the computation of minimal answers to the query $? \vee \mathcal{L}$, where \mathcal{L} is the set of all predicates. (Minimal answers $\vee \mathcal{A}$ to a query $? \vee \mathcal{H}$, where $\mathcal{H} \subseteq \mathcal{L}$, are then of course simply those minimal answers to $? \vee \mathcal{L}$ for which $\mathcal{A} \subseteq \mathcal{H}$.)

In Section 1 we first review some background and terminology. We also present a result detailing the conditions (in terms of disjunctive stable model membership), under which a disjunction of predicates can be extended to a minimal answer. The computation of disjunctive stable models in their entirety is clearly undesirable, and, under certain circumstances, this can be avoided using cyclic strong covers. This concept was previously developed in [Jo99, Jo99a] in the context of top-down query processing and model generation, and provides the basis for the entirety of the current

paper. Thus in Sections 2.1-2.9 we re-state (from [Jo96, Jo99, Jo99a]) the motivation, definition and fundamental properties of cyclic strong covers that are required for an understanding of the current paper. Cyclic strong covers are to be employed in order to overcome the need to compute disjunctive stable models in their entirety, thus in Sections 2.10-2.13 we present new results which demonstrate that they may be viewed as defining partial disjunctive stable models.

In Section 3 we focus on stratified databases. For such databases it is well known that disjunctive stable models coincide with perfect models, thus our results in Section 2 suggest that cyclic strong covers define partial perfect models. More importantly, it is shown in [Jo99] that cyclic strong covers can always be extended to total cyclic strong covers, and hence provide a characterisation of reasoning under the perfect model semantics. Using this, we show (Sections 3.3-3.8) that partial minimal answers may be characterised in terms of cyclic strong covers, and then derive our method for constructing minimal answers in terms of the repeated extension of cyclic strong covers. Worthy of note is that this can be achieved without necessarily constructing perfect models in their entirety.

In Section 4 we consider unstratified databases. For such databases it is known [Jo99a] that total cyclic strong covers characterise disjunctive stable models, but the possible non-existence of disjunctive stable models means that the partial models defined by (non-total) cyclic strong covers cannot necessarily be extended to full disjunctive stable models. This suggests that partial minimal answers can only be characterised using full disjunctive stable models, and (hence) that a direct extension of the methods of Section 3 to unstratified databases requires the (undesirable) computation of such models in their entirety.

In Section 5 we show that this problem can be addressed by partitioning the database into extensional and intensional components, and pre-processing (*compiling*) the construction of cyclic strong covers within the intension using the concept of *weakly cyclic covers* (originally introduced in [Jo99, Jo99a] in the context of compilation of top-down query processing). This greatly simplifies, and hence reduces the cost of, the run-time computation (which then amounts to minimal model reasoning in the extensional database), this saving being of particular importance given the computational complexity of the problem. An important property of our compilation is that it does not need to be repeated in the event of an update to the extensional database (this being the most common type of database update). Because compilation focuses on the intensional database, it is strongly dependent upon the above-mentioned ability to construct cyclic strong covers in a top-down fashion. In Sections 5.1-5.3 we restate (from [Jo99, Jo99a]) the motivation, definition and properties of weakly cyclic covers. We then detail in Sections 5.4-5.8 how our method for generating minimal answers is

amended and enhanced by the prior application of compilation.

If we wish to compute the minimal answers to a query of the form $?\bigvee \mathcal{H}$ (instead of $?\bigvee \mathcal{L}$), it is clearly more efficient to compute these directly (rather than deriving them from the minimal answers to $?\bigvee \mathcal{L}$). Thus in Section 6 we briefly indicate how our method can be adapted to compute (only) minimal answers to $?\bigvee \mathcal{H}$.

We briefly consider the adaptation of our methods to other semantics in Section 7, and in Section 8 consider the issues that arise when lifting our methods to the first order level. In Section 9 we compare and contrast our results with others from the literature. The concepts of the current paper yield a model generating procedure which constructs (only) models that are disjunctive stable, and we illustrate how these concepts could be used to adapt the model generating procedure employed in the DLV system [Le03] so as to prevent the generation of models that are not disjunctive stable. We also show that three database transformations presented in the literature (transformation to a positive database, a normal database, and a set of conditional facts) can be readily achieved using the tools presented in the current paper. Our conclusions are presented in Section 10. A further example is presented in Appendix A, and some proofs are presented in Appendix C.

Minimal answer computation is also of wider interest as a result of the fact that many questions relating to minimal answers (e.g., determining whether a given predicate belongs to some minimal answer) are Σ_2^P -complete ([Ei93, Jo96, Mi82]), and naturally occurring problems residing at Σ_n^P ($n \geq 2$) in the polynomial-time hierarchy are rare [St77, Wa86, Wr77]. In Appendix B we therefore present some complexity results related to our approach.

§1. Minimal answers

In this section we review some background and terminology. We also present a new result (Theorem 1.6) detailing the conditions (in terms of disjunctive stable model membership), under which a disjunction of predicates can be extended to a minimal answer.

1.1 Notation. Throughout Sections 1-7 we assume that \mathcal{L} is a finite propositional language (i.e., a finite set of predicates). Predicates will be denoted by A, B, C, \dots, P, Q . A *literal* is a predicate (a positive literal) or its negation (a negative literal), and we will use $\mathcal{P}, \mathcal{Q}, \mathcal{R}$ to denote arbitrary sets of literals. $\mathcal{Q}^- = \{P \in \mathcal{L} | \neg P \in \mathcal{Q}\}$, $\mathcal{Q}^+ = \{P \in \mathcal{L} | P \in \mathcal{Q}\}$ and $\overline{\mathcal{Q}} = \{\neg K | K \in \mathcal{Q}\}$. \mathcal{Q} is *total* (in \mathcal{L}) iff $\mathcal{Q}^- \cup \mathcal{Q}^+ = \mathcal{L}$, and *consistent* iff $\mathcal{Q}^- \cap \mathcal{Q}^+ = \emptyset$.

Throughout DB will denote a disjunctive deductive database in \mathcal{L} consisting of rules of the form

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

where each A_i, B_j is a predicate and $k > 0$. We may assume without loss of generality that if $r > 0$, then $h > 0$. We will denote rules using \mathbf{R}, \mathbf{S} , possibly with subscripts. If \mathbf{R} is the above rule, then $\text{antec}(\mathbf{R}) = \{A_1, A_2, \dots, A_h\}$ denotes the set of *antecedents* of \mathbf{R} , $\mathcal{N}(\mathbf{R}) = \{A_{h+1}, A_{h+2}, \dots, A_{h+r}\}$, and $\text{conseq}(\mathbf{R}) = \{B_1, B_2, \dots, B_k\}$ denotes the *consequent* of \mathbf{R} .

DB is said to be *positive* iff $\mathcal{N}(\mathbf{R}) = \emptyset$ for each $\mathbf{R} \in \text{DB}$.

1.2 Definition. A set $M \subseteq \mathcal{L}$ is a *model* of \mathbf{R} (written $M \models \mathbf{R}$) iff $\text{antec}(\mathbf{R}) \subseteq M$ and $M \cap \mathcal{N}(\mathbf{R}) = \emptyset$ implies that $\text{conseq}(\mathbf{R}) \cap M \neq \emptyset$. M is a model of DB (written $M \models \text{DB}$) iff $M \models \mathbf{R}$ for each $\mathbf{R} \in \text{DB}$.

A total consistent set of literals identifies a truth value for each predicate in \mathcal{L} , which in turn allows us to determine truth values for rules. Given a consistent set of literals \mathcal{C} , we let DB/\mathcal{C} denote those rules in DB , all of whose predicates are given a truth value by \mathcal{C} .

1.3 Definition. If \mathcal{C} is a consistent set of literals, let

$$\text{DB}/\mathcal{C} = \{\mathbf{R} \in \text{DB} \mid \text{conseq}(\mathbf{R}) \cup \text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}) \subseteq \mathcal{C}^+ \cup \mathcal{C}^-\}.$$

1.4 Definitions.

- (a) If \mathbf{R} is a rule, let $\text{pos}(\mathbf{R}) = \bigwedge \text{antec}(\mathbf{R}) \rightarrow \bigvee \text{conseq}(\mathbf{R})$, i.e., $\text{pos}(\mathbf{R})$ is formed from \mathbf{R} by removing the negative literals from the body of \mathbf{R} .
- (b) If $\mathcal{K} \subseteq \mathcal{L}$, then the *Gelfond-Lifschitz* transformation [Ge88] is given by

$$\text{DB}|_g \mathcal{K} = \{\text{pos}(\mathbf{R}) \mid \mathbf{R} \in \text{DB}, \mathcal{N}(\mathbf{R}) \cap \mathcal{K} = \emptyset\}.$$

Notice that $\text{DB}|_g \mathcal{K}$ is positive, and can therefore be interpreted straightforwardly using the minimal model semantics.

- (c) If $M \subseteq \mathcal{L}$, then M is a *disjunctive stable model* of DB [Pr91] iff M is a minimal model of $\text{DB}|_g M$.
- (d) If Φ is a formula in \mathcal{L} , then we write $\text{DB} \models \Phi$ iff Φ is true in every disjunctive stable model of DB .

- (e) If $\mathcal{A} \subseteq \mathcal{L}$, then $\bigvee \mathcal{A}$ is a *minimal answer* in DB iff $\text{DB} \models \bigvee \mathcal{A}$ and there is no proper subset $\mathcal{B} \subset \mathcal{A}$ such that $\text{DB} \models \bigvee \mathcal{B}$.

Disjunctive stable models are a straightforward generalisation of stable models for non-disjunctive databases [Ge88]. Note that every disjunctive stable model of DB is a minimal model of DB, and that for positive databases the converse holds.

Note in part (d) that we are using cautious reasoning (i.e., Φ is required to be true in every disjunctive stable model), as opposed to brave reasoning (in which Φ is required to be true in some such model). Clearly $\bigvee \mathcal{P}$ is true in some disjunctive stable model M iff there is some predicate in \mathcal{P} that is true in M , thus under brave reasoning, answer minimality reduces simply to disjunctive stable model membership.

Note also that if DB has no disjunctive stable model, then $\text{DB} \models \Phi$ trivially, in which case the empty disjunction is the only minimal answer.

Given a minimal answer $\bigvee \mathcal{A}$, we will for the sake of brevity also refer to the set \mathcal{A} as a minimal answer. If $\bigvee \mathcal{A}$ is a minimal answer and $A \in \mathcal{A}$, then there must be some disjunctive stable model M of DB such that $M \cap \mathcal{A} = \{A\}$ (else $\mathcal{A} - \{A\}$ intersects every disjunctive stable model and is hence an answer, thus contradicting the minimality of \mathcal{A}). Conversely if A belongs to a disjunctive stable model M , then by the minimality of M we must have that $\text{DB} \models A \vee \bigvee (\mathcal{L} - M)$, thus if $\mathcal{A} \subseteq \{A\} \cup (\mathcal{L} - M)$ is a minimal answer, then $\emptyset \neq M \cap \mathcal{A} \subseteq \{A\}$, and hence $A \in \mathcal{A}$. This then yields the following theorem which is a direct analogue of a result originally presented (for the minimal model semantics) by Minker.

1.5 Theorem [Mi82]. A predicate A belongs to some minimal answer (in DB) iff A belongs to some disjunctive stable model of DB.

If, as suggested in the introduction, we wish to construct minimal answers as a sequence $A_1, A_1 \vee A_2, \dots$, then Theorem 1.5 dictates that each such A_i must belong to some disjunctive stable model M_i . The following theorem indicates that we also need to consider the relationship between the models M_i .

1.6 Theorem. A set of predicates $\{A_j | j \leq r\}$ is contained in a minimal answer iff for each $i \leq r$ we may find a disjunctive stable model M_i of DB such that

- (a) $M_i \cap \{A_j | j \leq r\} = \{A_i\}$, and
- (b) $\text{DB} \models \bigvee_{j \leq r} A_j \vee \bigvee \bigcap_{j \leq r} (\mathcal{L} - M_j)$.

Proof (\rightarrow). Suppose that $A_1 \vee A_2 \vee \dots \vee A_{r+s}$ is a minimal answer, then for each $i \leq r$ we may find a disjunctive stable model M_i such that $M_i \cap \{A_j | j \leq r+s\} = \{A_i\}$ (else

$\{A_j | j \leq r + s, j \neq i\}$ intersects every disjunctive stable model, thus contradicting the minimality of $\{A_j | j \leq r + s\}$. In particular $M_i \cap \{A_j | j \leq r\} = \{A_i\}$.

If M is a disjunctive stable model of DB with $M \not\models \bigvee_{j \leq r} A_j$, then since $M \models \bigvee_{j \leq r+s} A_j$, there is some $k > r$ such that $A_k \in M$. By the chosen properties of the models M_i we have that $A_k \in \bigcap_{j \leq r} (\mathcal{L} - M_j)$. Thus $\text{DB} \models \bigvee_{j \leq r} A_j \vee \bigvee \bigcap_{j \leq r} (\mathcal{L} - M_j)$. (\leftarrow). Let $\mathcal{A} \subseteq \{A_j | j \leq r\} \cup \bigcap_{j \leq r} (\mathcal{L} - M_j)$ be a minimal answer. For each $i \leq r$, $\emptyset \neq M_i \cap \mathcal{A} \subseteq (M_i \cap \{A_j | j \leq r\}) \cup (M_i \cap \bigcap_{j \leq r} (\mathcal{L} - M_j)) = \{A_i\}$, thus $A_i \in \mathcal{A}$. ■

§2. Cyclic strong covers

Theorem 1.6 characterises partial minimal answers in terms of disjunctive stable models. Ideally however we would prefer not to have to construct models of the database in their entirety, and we can achieve this using the notion of cyclic strong covers. These were originally introduced in [Jo99, Jo99a] in the context of top-down model generation and query processing. They provide the basis for the entirety of the current paper, and thus in Sections 2.1-2.9 we re-state (from [Jo96, Jo98, Jo99, Jo99a]) their motivation, definition and fundamental properties.

Cyclic strong covers are to be employed in order to overcome the need to compute disjunctive stable models in their entirety, thus in Sections 2.10-2.13 we present new results which demonstrate that they may be viewed as defining partial disjunctive stable models.

The notion of a strong cover was originally introduced in [Jo97, Jo98] in the context of top-down query processing in positive databases. Suppose for example that DB contains the rules $R_0 = A \wedge B \rightarrow P \vee Q$, $R_1 = D \wedge E \rightarrow A \vee C$, $R_2 = E \vee C$, etc., and that we wish to determine whether $\bigvee \mathcal{P}$ is true (in DB) where $\mathcal{P} = \{P, Q, C\}$. Now $\text{conseq}(R_0) \subseteq \mathcal{P}$, thus if $\bigvee \mathcal{P}$ is false in some model M of DB, then so is $\bigvee \text{conseq}(R_0)$, and hence there must be some predicate in the body of R_0 which is also false in M . Thus $\bigvee \mathcal{P}$ is true (in DB) iff $A \vee \bigvee \mathcal{P}$ and $B \vee \bigvee \mathcal{P}$ are both true (in DB). Applying the same argument using R_1 , we see that $A \vee \bigvee \mathcal{P}$ is true iff $D \vee A \vee \bigvee \mathcal{P}$ and $E \vee A \vee \bigvee \mathcal{P}$ are both true. We cannot iterate this process indefinitely, thus we must eventually generate disjunctions $\bigvee \mathcal{C}$ (with $\mathcal{C} \supseteq \mathcal{P}$) that are either subsumed by a rule in DB whose antecedent is empty (e.g., R_2 subsumes $E \vee A \vee \bigvee \mathcal{P}$), or for which (for all $R \in \text{DB}$) $\text{conseq}(R) \subseteq \mathcal{C} \implies \mathcal{C} \cap \text{antec}(R) \neq \emptyset$ (in which case nothing further can be gained by applying database rules using the above approach).

In the latter case, we refer to \mathcal{C} as a *strong cover* of \mathcal{P} (in DB). Strong covers are clearly related to models, in that (for positive databases) \mathcal{C} is a strong cover (in DB) iff $\mathcal{L} - \mathcal{C}$ is a model of DB (hence in particular $\bigvee \mathcal{P}$ is true iff \mathcal{P} has no strong cover). The

two notions are therefore simply complementary, but the notion of a strong cover gives a clear(er) view of top-down processing, i.e., starting with the goal $\bigvee \mathcal{P}$ and attempting to close \mathcal{P} under the above-mentioned extension operator via backward application of the database rules.

Consider now extending these ideas to non-positive databases. Suppose that $\text{DB} = \{C \wedge \neg A \rightarrow Q \vee D, A \vee B, C \vee D, B \vee D, D \rightarrow P \vee Q, P \rightarrow Q\}$, then (using the last two rules) $\text{DB} \models Q$ iff $\text{DB} \models Q \vee P \vee D$, which (by the first rule) is true iff $\text{DB} \models Q \vee P \vee D \vee C$ and $\text{DB} \models Q \vee P \vee D \vee \neg A$. The first of these is trivially true by virtue of the rule $C \vee D$, and the set identified in the second, $\{Q, P, D, \neg A\}$, is now closed under our extension operator (and we again refer to it as a strong cover). This then yields the following definition. (Recall that $\overline{\mathcal{N}(\mathbf{R})} = \{\neg P \mid P \in \mathcal{N}(\mathbf{R})\}$, thus $\text{antec}(\mathbf{R}) \cup \overline{\mathcal{N}(\mathbf{R})}$ is the set of literals appearing in the body of \mathbf{R} .)

2.1 Definition [Jo99a]. Let \mathcal{Q} be a consistent set of literals in \mathcal{L} . A *strong cover* of \mathcal{Q} (in DB) is a consistent set of literals $\mathcal{C} \supseteq \mathcal{Q}$ such that for each $\mathbf{R} \in \text{DB}$

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

Again there is some complementarity between strong covers and models in that if \mathcal{C} is a strong cover then $\mathcal{L} - \mathcal{C}^+ \models \text{DB}$. We can also view $\overline{\mathcal{C}}$ as a partial model, since Definition 2.1 dictates that if $\bigvee \text{conseq}(\mathbf{R})$ is false in $\overline{\mathcal{C}}$, then some atom in the body of \mathbf{R} is also false in $\overline{\mathcal{C}}$. Conversely, if $M \subseteq \mathcal{L}$ then M is a model of DB iff $\overline{M} \cup (\mathcal{L} - M)$ is a strong cover in DB .

So in the above example $\{Q, P, D, \neg A\}$ is a strong cover of $\{Q\}$, hence we can certainly infer that Q is false in some model (i.e., $\mathcal{L} - \{Q, P, D\} = \{A, B, C\}$) of DB . However strong covers do not address the issue of model minimality (Definition 1.4(c)), so there is no guarantee that our model $\{A, B, C\}$ will be disjunctive stable. Clearly the problem is that we have not expanded the negative literal $\neg A$ generated in the strong cover. In this case, if M is disjunctive stable model of DB , then $A \in M$ iff $B \notin M$, thus $\text{DB} \models Q \vee P \vee D \vee \neg A$ iff $\text{DB} \models Q \vee P \vee D \vee \neg A \vee B$ (which is trivially true by virtue of the rule $B \vee D$). Thus $\text{DB} \models Q$.

Thus the expansion of negative literals requires a tool with which we can handle questions regarding disjunctive stable model membership. For this purpose we will use cyclic trees¹. These were introduced in [Jo96] in order to facilitate reasoning about minimal (and thereby perfect and disjunctive stable) models. We first present

¹Throughout this paper we will use what is referred to in [Jo96] as *unfactored* cyclic trees.

an example to motivate the definition of such trees, and then detail their essential properties.

2.2 Example. Suppose that DB consists of the following rules:

- | | | |
|--|--|--|
| 1. $Q_2 \wedge Q_3 \wedge \neg P_1 \rightarrow Q_1 \vee Q_5$ | 2. $Q_1 \wedge \neg P_2 \rightarrow Q_2$ | 3. $A_2 \wedge \neg P_3 \rightarrow Q_3$ |
| 4. $A_3 \rightarrow Q_1 \vee Q_2 \vee Q_4$ | 5. $A_2 \vee P_5$ | 6. $A_1 \rightarrow Q_3 \vee Q_2$ |
| 7. $A_3 \vee P_4$ | 8. $Q_5 \rightarrow Q_2$ | |

and we wish to determine whether Q_1 lies in some disjunctive stable model of DB.

Suppose that Q_1 lies in a disjunctive stable model M . Since M is a minimal model of $\text{DB}|_g M$, $M - \{Q_1\} \not\models \text{DB}|_g M$, hence we may find a rule $R \in \text{DB}$ such that $\mathcal{N}(R) \cap M = \emptyset$ (i.e., $\text{pos}(R) \in \text{DB}|_g M$) and $M - \{Q_1\} \not\models \text{pos}(R)$, i.e., $\text{antec}(R) \subseteq M - \{Q_1\}$ and $\text{conseq}(R) \cap (M - \{Q_1\}) = \emptyset$. (It will prove useful to note that this constraint can be rewritten as $M \cap (\text{conseq}(R) - \{Q_1\}) = \emptyset$). Since $M \models \text{pos}(R)$, we must have that $M \cap \text{conseq}(R) \neq \emptyset$, and hence that $Q_1 \in \text{conseq}(R)$.

Since $Q_1 \in \text{conseq}(R)$, there are only two possibilities for R , namely rules 1 and 4. Suppose that R is rule 1, then the above constraints dictate that $\{Q_1, Q_2, Q_3\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1\}$. We will represent this application of rule 1 using the “rule node” rn_1 in the tree \mathcal{T}_1 (Figure 2.2(i)). (Only Q_2 and Q_3 (i.e., the antecedents) are depicted, since we wish to examine these predicates further.)

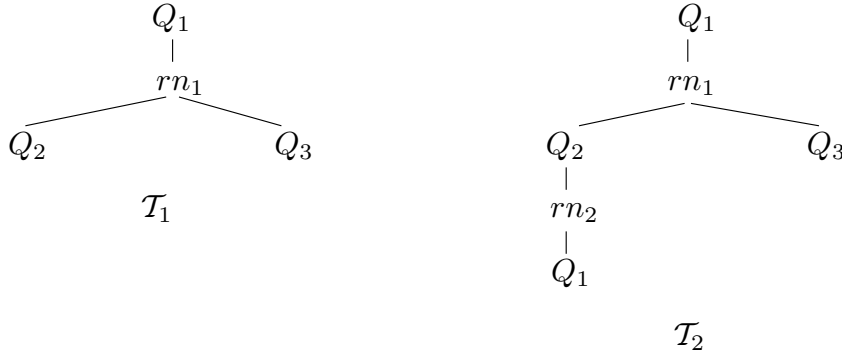


Figure 2.2(i).

Suppose now that we apply the same argument to Q_2 . Let R be a rule in DB such that $\mathcal{N}(R) \cap M = \emptyset$ and $M - \{Q_2\} \not\models \text{pos}(R)$. Again we must have that $Q_2 \in \text{conseq}(R)$ (thus restricting R to rules 2,4,6 or 8), $\text{antec}(R) \subseteq M - \{Q_2\}$ (thus disallowing rule 8 since the existing constraints dictate that $Q_5 \notin M$) and $M \cap (\text{conseq}(R) - \{Q_2\}) = \emptyset$ (thus disallowing rules 4 and 6, since the existing constraints dictate that $\{Q_1, Q_2, Q_3\} \subseteq M$). R *must* therefore be rule 2 (none of the existing constraints disallow

rule 2) and this then yields the tree \mathcal{T}_2 depicted in Figure 2.2(i). The constraint that $\text{antec}(\mathbf{R}) \subseteq M \subseteq \mathcal{L} - \mathcal{N}(\mathbf{R})$ then dictates that $\{Q_1\} \subseteq M \subseteq \mathcal{L} - \{P_2\}$, and $M \cap (\text{conseq}(\mathbf{R}) - \{Q_2\}) = \emptyset$ imposes no further constraint on M . Combining this with the existing constraint on M we have that $\{Q_1, Q_2, Q_3\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1, P_2\}$.

The left hand branch of \mathcal{T}_2 forms a “cycle”. We have already identified rules that witness that $M - \{Q_1\} \not\models \text{DB}|_g M$ and $M - \{Q_2\} \not\models \text{DB}|_g M$, so there is no point in doing so again. We thus look for a rule $\mathbf{R} \in \text{DB}$ such that $\mathcal{N}(\mathbf{R}) \cap M = \emptyset$ and $M - \{Q_1, Q_2\} \not\models \text{pos}(\mathbf{R})$, and as above we can easily show that (given the existing constraints on M) the only candidate is rule 4, thus yielding the new constraint $\{Q_1, Q_2, Q_3, A_3\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1, P_2, Q_4\}$.

Applying the same argument to A_3 , rule 7 is the only rule that can witness that $M - \{A_3\} \not\models \text{DB}|_g M$, which then yields the constraint $\{Q_1, Q_2, Q_3, A_3\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1, P_2, Q_4, P_4\}$, and also terminates the branch, since rule 7 has no antecedents (see \mathcal{T}_3 , Figure 2.2(ii)).

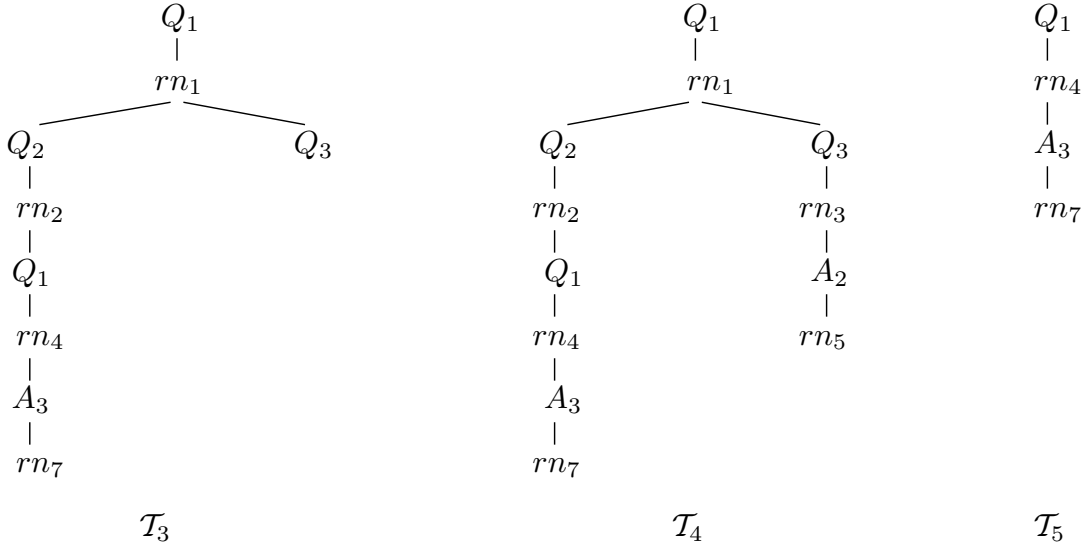


Figure 2.2(ii).

We thus move on to examine Q_3 . If $\mathcal{N}(\mathbf{R}) \cap M = \emptyset$ and $M - \{Q_3\} \not\models \text{pos}(\mathbf{R})$, then as above, $M \cap (\text{conseq}(\mathbf{R}) - \{Q_3\}) = \emptyset$, hence \mathbf{R} cannot be rule 6 (given the existing constraints on M). \mathbf{R} must therefore be rule 3. Rule 5 can then be used to handle A_2 , and again this terminates the branch, yielding \mathcal{T}_4 (Figure 2.2(ii)), and the constraint $\{Q_1, Q_2, Q_3, A_3, A_2\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1, P_2, Q_4, P_4, P_3, P_5\}$.

Note that the only point in the above construction when there was a choice (of rule) was at the very beginning when we chose rule 1 rather than rule 4. If we had instead chosen rule 4, then we would have generated the tree \mathcal{T}_5 (Figure 2.2(ii)), with associated constraint $\{Q_1, A_3\} \subseteq M \subseteq \mathcal{L} - \{Q_2, Q_4, P_4\}$.

Thus if Q_1 belongs to some disjunctive stable model M , then either

- (i) $\{Q_1, Q_2, Q_3, A_3, A_2\} \subseteq M \subseteq \mathcal{L} - \{Q_5, P_1, P_2, Q_4, P_4, P_3, P_5\}$, or
- (ii) $\{Q_1, A_3\} \subseteq M \subseteq \mathcal{L} - \{Q_2, Q_4, P_4\}$.

Moreover we can also show the converse. For example if M is *any* model of DB that is disjoint from $\{Q_5, P_1, P_2, Q_4, P_4, P_3, P_5\}$, then using the rules in \mathcal{T}_4 we may infer that $\{Q_1, Q_2, Q_3, A_3, A_2\} \subseteq M$ (see Theorem 2.4(c) below). Thus for any formula Φ , the following are equivalent:

- $\text{DB} \models \Phi \vee \neg Q_1$,
- $\text{DB} \models \Phi \vee \bigvee \{\neg Q_1, \neg Q_2, \neg Q_3, \neg A_3, \neg A_2\} \vee \bigvee \{Q_5, P_1, P_2, Q_4, P_4, P_3, P_5\}$ and $\text{DB} \models \Phi \vee \bigvee \{\neg Q_1, \neg S_3\} \vee \bigvee \{Q_2, Q_4, P_4\}$,
- $\text{DB} \models \Phi \vee \bigvee \{Q_5, P_1, P_2, Q_4, P_4, P_3, P_5\}$ and $\text{DB} \models \Phi \vee \bigvee \{Q_2, Q_4, P_4\}$.

This then gives us our required means of expanding negative query literals.

Note the following features of the above tree construction. Firstly it is the intention that each predicate lies in the intended model M . Each “rule node” is labelled with a rule $\mathbf{R} \in \text{DB}$ such that $\mathcal{N}(\mathbf{R}) \cap M = \emptyset$, and if the parent node of $rn_{\mathbf{R}}$ is n , then n is labelled with a predicate, and there is a subset \mathcal{P} of the predicates lying on the partial branch from the root down to (and including) n such that $M - \mathcal{P} \not\models \text{pos}(\mathbf{R})$. Thus, $\text{antec}(\mathbf{R}) \subseteq M - \mathcal{P}$, and $\emptyset = \text{conseq}(\mathbf{R}) \cap (M - \mathcal{P}) = M \cap (\text{conseq}(\mathbf{R}) - \mathcal{P})$. Since $M \models \text{pos}(\mathbf{R})$ we must have that $\text{conseq}(\mathbf{R}) \cap \mathcal{P} \neq \emptyset$.

The following definition captures these features, and also makes precise the subset $\mathcal{P} = \text{CYC}(n)$ (the “cycle above n ”) that we wish to work with, this choice being motivated by the properties (Theorem 2.4) of the resulting trees. Informally if m is the top-most (or first) node on the partial branch from the root down to (and including) n which is labelled with the same predicate as n , then $\text{CYC}(n)$ consists of the predicates labelling a node lying between m and n (including n). [Jo96] compares this choice of \mathcal{P} with a number of alternatives.

2.3 Definition [Jo96]. If P is a predicate, then a *cyclic tree* \mathcal{T} for P in DB contains rule nodes and predicate nodes, satisfying conditions (i) - (v) below.

- (i) Each rule node rn is labelled with a rule $\mathbf{R} \in \text{DB}$ (written $rn_{\mathbf{R}}$). Each predicate node n is labelled with a predicate $Q \in \mathcal{L}$ (and we write $\text{lab}(n) = Q$).

We define $\text{Pred}(\mathcal{T}) = \{\text{lab}(n) \mid n \text{ is a predicate node in } \mathcal{T}\}$.

- (ii) The root node is a predicate node labelled with P (and we write $\text{root}(\mathcal{T}) = P$).

We regard the root as being at the “top” of the tree, thus if m and n are any two nodes, we write $m \geq n$ iff m is the root node, or m and n are the same node, or m lies on the partial branch from the root down to n .

- (iii) If n is a predicate node, let $\text{CYC}(n) = \{\text{lab}(n') \mid n' \text{ is a predicate node, } n' \geq n\}$

and there is some predicate node $m \geq n'$ with $lab(m) = lab(n)$.

n has exactly one child node which is a rule node $rn_{\mathbf{R}}$ with $conseq(\mathbf{R}) \cap CYC(n) \neq \emptyset$ and $antec(\mathbf{R}) \cap CYC(n) = \emptyset$.

We define $\mathcal{O}(rn_{\mathbf{R}}) = conseq(\mathbf{R}) - CYC(n)$.

- (iv) If $rn_{\mathbf{R}}$ is a rule node, then for each predicate $A \in antec(\mathbf{R})$, $rn_{\mathbf{R}}$ has a (predicate) child node labelled with A . $rn_{\mathbf{R}}$ has no other child nodes.
- (v) $\mathcal{O}(\mathcal{T}) = \bigcup\{\mathcal{O}(rn_{\mathbf{R}}) \mid rn_{\mathbf{R}} \text{ occurs in } \mathcal{T}\}$ and $\mathcal{N}(\mathcal{T}) = \bigcup\{\mathcal{N}(\mathbf{R}) \mid rn_{\mathbf{R}} \text{ occurs in } \mathcal{T}\}$ are both disjoint from $Pred(\mathcal{T})$.
Let $\mathcal{S}(\mathcal{T}) = \overline{Pred(\mathcal{T})} \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$.

The requirement that $Pred(\mathcal{T})$ and $\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$ be disjoint is a consequence of the intention that $Pred(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$.

Thus in Example 2.2, $Pred(\mathcal{T}_4) = \{Q_1, Q_2, Q_3, A_3, A_2\}$, $\mathcal{O}(\mathcal{T}_4) = \{Q_5, Q_4, P_4, P_5\}$, and $\mathcal{N}(\mathcal{T}) = \{P_1, P_2, P_3\}$. Similarly $Pred(\mathcal{T}_5) = \{Q_1, A_3\}$, $\mathcal{O}(\mathcal{T}_5) = \{Q_2, Q_4, P_4\}$ and $\mathcal{N}(\mathcal{T}_5) = \emptyset$.

Notice that condition (iii) above is inherently top-down. In [Jo96, Section 5], it is shown that every branch through a cyclic tree has length at most $|\mathcal{L}| * (|\mathcal{L}| + 1)/2$. Clearly if \mathcal{T} is a cyclic tree in DB, then \mathcal{T} is also a cyclic tree in any superset of DB. Notice also that if \mathbf{R} is a rule labelling some rule node in \mathcal{T} , then each predicate appearing in \mathbf{R} appears in $Pred(\mathcal{T}) \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$. In the terminology of Definition 1.3, $\mathbf{R} \in DB/\mathcal{S}(\mathcal{T})$.

The following theorem captures the essential properties of cyclic trees.

2.4 Theorem [Jo96, Jo98a, Jo99a].

- (a) If the predicate P belongs to some disjunctive stable model M , then we may find a cyclic tree \mathcal{T} for P in DB such that $Pred(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$ (i.e., $\mathcal{S}(\mathcal{T}) \subseteq \overline{M} \cup (\mathcal{L} - M)$).
- (b) If \mathcal{T} is a cyclic tree in DB and M is any model of $DB|_g(\mathcal{L} - \mathcal{N}(\mathcal{T}))$ with $M \cap \mathcal{O}(\mathcal{T}) = \emptyset$, then $Pred(\mathcal{T}) \subseteq M$.
- (c) If \mathcal{T} is a cyclic tree in DB and M is any model of DB with $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$, then $Pred(\mathcal{T}) \subseteq M$.
- (d) For any formula Φ , $DB \models \Phi \vee \neg P$ iff whenever \mathcal{T} is a cyclic tree for P in T , $DB \models \Phi \vee \bigvee \mathcal{S}(\mathcal{T})$.

The construction presented in Example 2.2 illustrates how a cyclic tree satisfying the properties of Theorem 2.4(a) is generated. In order to further facilitate understanding of cyclic trees, a sketch of the proof for part (c) is presented in Appendix C. Part (d) encapsulates our approach to expanding negative goal literals previously suggested

in Example 2.2.

If M is a disjunctive stable model of DB then (applying Theorem 2.4(a) for each $P \in M$) we can find a set $\{\mathcal{T}_i | i \leq j\}$ of cyclic trees in DB such that $\bigcup_{i \leq j} \text{Pred}(\mathcal{T}_i) = M$ and $M \subseteq \mathcal{L} - \bigcup_{i \leq j} (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$, i.e., $\mathcal{L} - M \supseteq \bigcup_{i \leq j} (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$. $\overline{M} \cup (\mathcal{L} - M)$ thus has the property identified in the following definition.

2.5 Definition [Jo99]. Let \mathcal{C} be a consistent set of literals, then \mathcal{C} is said to be *cyclic* (in DB) iff there is a set of cyclic trees $\{\mathcal{T}_i | i \leq j\}$ in DB such that

- (i) $\mathcal{C}^- = \bigcup_{i \leq j} \text{Pred}(\mathcal{T}_i)$, and
- (ii) $\mathcal{C}^+ \supseteq \bigcup_{i \leq j} (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$.

Again note that if \mathcal{C} is cyclic in DB, then \mathcal{C} is also cyclic in any superset of DB. Where there is no ambiguity (or required emphasis), we will, for the sake of brevity, refer to a cyclic set/strong cover in DB as simply cyclic/a strong cover.

Following on from Theorem 2.4(b) we see for example that if \mathcal{C} is cyclic and M is any model of $\text{DB}|_g(\mathcal{L} - \mathcal{C}^+)$ with $M \cap \mathcal{C}^+ = \emptyset$, then $\mathcal{C}^- \subseteq M$. In particular, if \mathcal{C} is a total cyclic strong cover and $M = \mathcal{C}^- = \mathcal{L} - \mathcal{C}^+$, then (since \mathcal{C} is a strong cover) $M \models \text{DB}$, and hence $M \models \text{DB}|_g M$. Moreover if $M^* \subseteq M$ with $M^* \models \text{DB}|_g M$, then $M^* \models \text{DB}|_g(\mathcal{L} - \mathcal{C}^+)$ and $M^* \cap \mathcal{C}^+ = \emptyset$, thus $\mathcal{C}^- = M \subseteq M^*$. Hence $M = M^*$ and M is a disjunctive stable model of DB. Conversely (as indicated above) if M is a disjunctive stable model, then $\overline{M} \cup (\mathcal{L} - M)$ is a total cyclic strong cover, thus we have part (a) of the following theorem. Part (b) follows immediately from part (a), and part (c) gives us a useful characterisation with which we can manipulate cyclic sets.

2.6 Theorem [Jo99a].

- (a) If $M \subseteq \mathcal{L}$, then M is a disjunctive stable model of DB iff $\overline{M} \cup (\mathcal{L} - M)$ is a cyclic strong cover in DB.
- (b) If \mathcal{Q} is a set of literals, then $\text{DB} \models \bigvee \mathcal{Q}$ iff there is no total cyclic strong cover of \mathcal{Q} in DB.
- (c) A consistent set of literals \mathcal{C} is cyclic in DB iff for each $P \in \mathcal{C}^-$ there is a cyclic tree \mathcal{T} for P in DB such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$.

Notice that the complementarity between cyclicness and model-hood arises again from our top-down view of processing² in which we start with the goal, and develop supersets there-of. For example, if DB contains the rule $\neg P \rightarrow Q$, then $\text{DB} \models Q$ iff $\text{DB} \models Q \vee \neg P$ iff $\text{DB} \models Q \vee \mathcal{S}(\mathcal{T})$ for every cyclic tree \mathcal{T} for P in DB. The superset

²As mentioned earlier, such a top-down view is inherent to covers and cyclic trees, and is important to our later discussion on compilation.

$\{Q\} \cup \mathcal{S}(T)$ of $\{Q\}$ that we have generated is cyclic. We can of course then iterate this process, applying the operator indicated in Definition 2.1, and forming cyclic trees in order to expand negative literals. Ultimately this will generate sets of literals that are either inconsistent, subsumed by a rule in DB whose body is empty, or closed under the two extension operators being employed. In the latter case, the set will be a cyclic strong cover. The top-down construction of cyclic strong covers and top-down query processing are thus closely related.

The top-down construction of cyclic trees and cyclic strong covers is detailed in [Jo96, Jo98, Jo99, Jo99a], representing cyclic strong covers as the branches of an *extended deduction tree*, as illustrated in the following example (taken from [Jo98]).

2.7 Example. Suppose that DB contains the following rules

- | | | |
|---|---|--|
| 1. $E_0 \wedge \neg B_0 \rightarrow A_0 \vee A_1$ | 2. $E_1 \wedge \neg E_3 \rightarrow B_0 \vee B_1$ | 3. $E_1 \wedge \neg E_5 \rightarrow B_0$ |
| 4. $E_4 \wedge E_1 \rightarrow B_2 \vee B_1$ | 5. $E_1 \vee E_2$ | 6. $E_0 \vee E_2$ |

and we wish to construct cyclic strong covers of $\mathcal{P} = \{A_0, A_1, B_2, E_2\}$. Since the consequent of rule 1 is contained in \mathcal{P} , any strong cover of \mathcal{P} must contain either E_0 or $\neg B_0$, this being depicted by the first level of the tree depicted in Figure 2.7. The left hand branch cannot be extended to a strong cover, since the consequent of rule 6 is a subset of the predicates on the branch. If $\neg B_0$ is contained in a cyclic strong cover \mathcal{C} , then by Theorem 2.6(c) there is cyclic tree \mathcal{T} for B_0 such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$. B_0 has two cyclic trees in DB, their \mathcal{S} sets appearing as child nodes of B_0 .

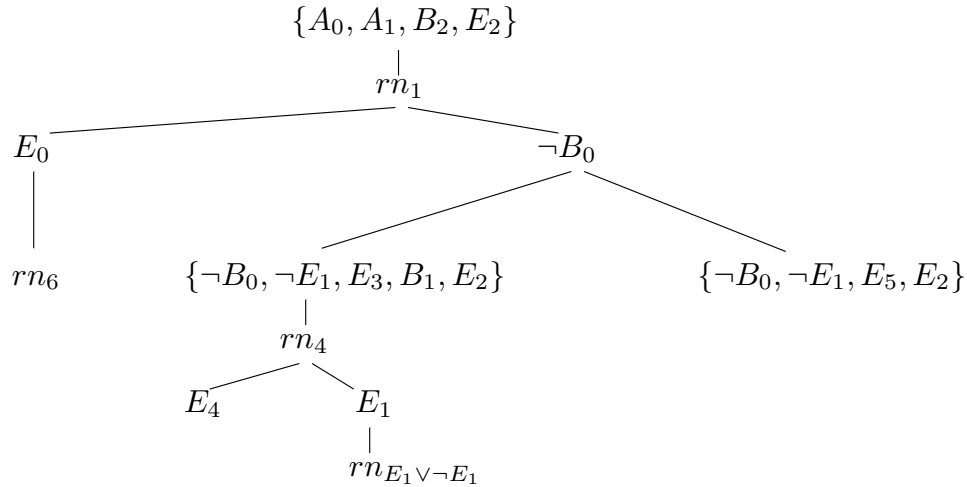


Figure 2.7

The consequent of rule 4 is contained in the branch to the left-hand child of $\neg B_0$. The branch down to E_1 is complementary, and hence cannot be extended to a cyclic

strong cover. The other two branches represent the sets $\mathcal{P} \cup \{\neg B_0, \neg E_1, E_3, B_1, E_2\} \cup \{E_4\}$ and $\mathcal{P} \cup \{\neg B_0, \neg E_1, E_5, E_2\}$. Both of these sets are cyclic, and are easily seen to be strong covers. Moreover, any other cyclic strong cover of \mathcal{P} must contain one of these.

Note the distinction between extended deduction trees and cyclic trees. The former provide a construction of cyclic strong covers, part of which requires the construction of cyclic trees. These are generated when we encounter a negative literal (e.g., $\neg B_0$) in order to restore the cyclicity of the (literals on the) branch. It is in fact possible to prune the cyclic trees needed to achieve this [Jo99a, Theorem 4.1], for example by allowing a predicate nodes n (in the cyclic tree) to be a leaf if $\neg lab(n)$ already appears on the branch in the extended deduction tree. We return to this point in Section 9.5.

The construction of cyclic strong covers can easily be amended to compute total cyclic strong covers (and hence by Theorem 2.6(a) disjunctive stable models). To achieve this we implicitly add the denial rules $\{P \wedge \neg P \rightarrow \text{FALSE} \mid P \in \mathcal{L}\}$ to DB^3 . The disjunctive stable models of the database are unaffected, and all strong covers of $\{\text{FALSE}\}$ are forced to be total. This approach can also be enhanced by the application of the results presented in Theorem 2.12 and Corollary 2.13 below. In Section 9 we compare this approach to the computation of disjunctive stable models with those presented in [Fe95, Fe95a, Le03].

The operators employed to construct extended deduction trees are detailed in Definition 2.8, and using these a simple algorithm (i.e., a tree traversal) for the construction of cyclic strong covers can then be presented, based upon the use of two stacks, and operating in space that is quadratic in $|\mathcal{L}|$ (the size of the language) [Jo98].

2.8 Definition. Suppose that \mathcal{Q} is cyclic, then a strong cover \mathcal{C} of \mathcal{Q} is said to be a *constructible extension* of \mathcal{Q} iff we can find a sequence $\mathcal{Q} = \mathcal{Q}_0 \subseteq \mathcal{Q}_1 \subseteq \dots \subseteq \mathcal{Q}_r = \mathcal{C}$ such that for each $1 \leq i \leq r$ there is a rule $\mathbf{R}_i \in \text{DB}$ such that $\text{conseq}(\mathbf{R}_i) \subseteq \mathcal{Q}_{i-1}$, $\mathcal{Q}_{i-1} \cap (\text{antec}(\mathbf{R}_i) \cup \overline{\mathcal{N}(\mathbf{R}_i)}) = \emptyset$ and either

- (i) $\mathcal{Q}_i = \mathcal{Q}_{i-1} \cup \{A_i\}$, where $A_i \in \text{antec}(\mathbf{R}_i)$, or
- (ii) $\mathcal{Q}_i = \mathcal{Q}_{i-1} \cup \mathcal{S}(\mathcal{T})$, where \mathcal{T} is a cyclic tree for some $A_i \in \mathcal{N}(\mathbf{R}_i)$ in DB.

Note that each \mathcal{Q}_i is (required to be) consistent (since \mathcal{C} is). If $\mathcal{Q} = \mathcal{R} \cup \{\neg A\}$, where \mathcal{R} is cyclic and $A \notin \mathcal{R}^- \cup \mathcal{R}^+$, then a constructible extension of \mathcal{Q} is formed by taking a constructible extension of $\mathcal{Q}' = \mathcal{R} \cup \mathcal{S}(\mathcal{T})$, where \mathcal{T} is a cyclic tree for A in DB such that \mathcal{Q}' is consistent.

³The application of these denial rules within such a top-down construction amounts to the application of an unrestricted splitting rule (e.g., [Ya96]).

Note that Definition 2.8 includes the case when $\mathcal{Q} \subseteq \mathcal{L}$ since \mathcal{Q} is then trivially cyclic. It will not prove necessary to define constructible extensions for sets more complex than those above. Note that constructible extensions are themselves cyclic strong covers (since \mathcal{C} is required to be a strong cover). Theorem 2.6(c) trivially implies the following corollary.

2.9 Corollary. If \mathcal{Q} has the form given in Definition 2.8 and \mathcal{D} is a cyclic strong cover of \mathcal{Q} , then we may find a constructible extension \mathcal{C} of \mathcal{Q} such that $\mathcal{C} \subseteq \mathcal{D}$.

Throughout the rest of this paper we will view (the complement of) cyclic strong covers as partial disjunctive stable models, and in the remainder of this section we therefore present results which support this view. We first show that cyclicity can be characterised in terms of disjunctive stable models (thus providing a “converse” to Theorem 2.6(a)).

2.10 Theorem.

- (a) A consistent set of literals \mathcal{C} is cyclic in DB iff \mathcal{C}^- is a disjunctive stable model of $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$.
- (b) If \mathcal{C} is a strong cover in DB/\mathcal{C} , then \mathcal{C} is cyclic in DB iff \mathcal{C}^- is a disjunctive stable model of DB/\mathcal{C} .

Proof (a) (\rightarrow). Suppose that \mathcal{C} is cyclic in DB. If \mathcal{T} is a cyclic tree with $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$, then (as mentioned earlier) for each rule \mathbf{R} labelling a rule node in \mathcal{T} we have that $\mathbf{R} \in \text{DB}/\mathcal{S}(\mathcal{T}) \subseteq \text{DB}/\mathcal{C}$ and $\mathcal{C}^- \models \mathbf{R}$ (since $\text{conseq}(\mathbf{R}) \cap \text{Pred}(\mathcal{T}) \neq \emptyset$). Thus \mathcal{C} is cyclic in $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$.

Clearly every predicate in $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$ appears in $\mathcal{C}^+ \cup \mathcal{C}^-$, hence \mathcal{C} is total strong cover in $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$. The result then follows from Theorem 2.6(a).

(\leftarrow). Every predicate in $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$ appears in $\mathcal{L}^* = \mathcal{C}^+ \cup \mathcal{C}^-$. Thus by Theorem 2.6(a), $\mathcal{C} = \overline{\mathcal{C}^-} \cup (\mathcal{L}^* - \mathcal{C}^-)$ is cyclic in $\{\mathbf{R} \in \text{DB}/\mathcal{C} \mid \mathcal{C}^- \models \mathbf{R}\}$, and hence cyclic in DB.

(b). If \mathcal{C} is a strong cover in DB/\mathcal{C} , then $\mathcal{C}^- \models \text{DB}/\mathcal{C}$, thus the result follows trivially from part (a). ■

Because of the possible non-existence of disjunctive stable models, it is of course not the case that every cyclic strong cover can be extended to a total cyclic strong cover. We can however show (Theorem 2.12 below) that the disjunctive stable models extending (the complement of) a cyclic strong cover are precisely the disjunctive stable

models of some reduced database, and thus that an inability to extend cyclic strong covers is simply the phenomenon of the non-existence of such models in a different guise. Definition 2.11 below defines our reduced database.

2.11 Definition. If \mathcal{D} is a strong cover in DB and R is rule in DB, then let $R_{\mathcal{D}}$ be formed from R by replacing each predicate in \mathcal{D}^- by **TRUE**, and each predicate in \mathcal{D}^+ by **FALSE**.

Note that if $\text{conseq}(R) \cap \mathcal{D}^- \neq \emptyset$, then $R_{\mathcal{D}}$ reduces to **TRUE**. Also, if $\text{conseq}(R) \subseteq \mathcal{D}^+$ then $\text{conseq}(R_{\mathcal{D}})$ reduces to **FALSE**, but since \mathcal{D} is a strong cover there must be some literal in the body of R which evaluates to **FALSE**, thus $R_{\mathcal{D}}$ as a whole reduces to **TRUE**.

In general $R_{\mathcal{D}}$ reduces to **TRUE** (and we will write $R_{\mathcal{D}} \equiv \text{TRUE}$) iff $\text{conseq}(R) \cap \mathcal{D}^- \neq \emptyset$ or $\text{antec}(R) \cap \mathcal{D}^+ \neq \emptyset$ or $\mathcal{N}(R) \cap \mathcal{D}^- \neq \emptyset$.

Let $\text{DB}_{\mathcal{D}} = \{R_{\mathcal{D}} \mid R \in \text{DB}, R_{\mathcal{D}} \not\equiv \text{TRUE}\}$. Clearly a rule R^* is in $\text{DB}_{\mathcal{D}}$ iff there is a rule $R \in \text{DB}$ such that

- (i) $\text{conseq}(R) \cap \mathcal{D}^- = \emptyset$ and $\text{conseq}(R^*) = \text{conseq}(R) - \mathcal{D}^+$,
- (ii) $\text{antec}(R) \cap \mathcal{D}^+ = \emptyset$ and $\text{antec}(R^*) = \text{antec}(R) - \mathcal{D}^-$, and
- (iii) $\mathcal{N}(R) \cap \mathcal{D}^- = \emptyset$ and $\mathcal{N}(R^*) = \mathcal{N}(R) - \mathcal{D}^+$

and that under these conditions we have that $R^* = R_{\mathcal{D}}$. Notice that only predicates from $\mathcal{L} - (\mathcal{D}^+ \cup \mathcal{D}^-)$ appear in $\text{DB}_{\mathcal{D}}$.

2.12 Theorem. Suppose that \mathcal{D} is a cyclic strong cover in DB and $\mathcal{D}^- \subseteq M \subseteq \mathcal{L} - \mathcal{D}^+$. Then M is a disjunctive stable model of DB iff $M - \mathcal{D}^-$ is a disjunctive stable model of $\text{DB}_{\mathcal{D}}$.

Since cyclic strong covers are themselves characterised by disjunctive stable models of some sub-database, Theorem 2.12 also allows us to characterise when one cyclic strong cover extends another. The proofs of both results are presented in Appendix C.

2.13 Corollary. Suppose that \mathcal{D} is a cyclic strong cover in DB and \mathcal{G} is a consistent set of literals with $\mathcal{G} \supseteq \mathcal{D}$.

- (a) \mathcal{G} is a strong cover in DB iff $\mathcal{G} - \mathcal{D}$ is a strong cover in $\text{DB}_{\mathcal{D}}$.
- (b) \mathcal{G} is a cyclic strong cover in DB iff $\mathcal{G} - \mathcal{D}$ is a cyclic strong cover in $\text{DB}_{\mathcal{D}}$.

§3. Stratified databases

Throughout this section we assume that DB is stratified [Pr88]. In [Jo99] it was shown that for such databases, cyclic strong covers provide a characterisation of reasoning under the perfect model semantics, and that they may always be extended to

total cyclic strong covers. This characterisation forms the basis for the whole of this section, and is therefore re-stated in Theorem 3.2.

Using this we then show (Theorem 3.3) that cyclic strong covers can be used to characterise partial minimal answers. In Sections 3.4-3.8 we introduce the notion of a *cyclic state*, and then present our method for constructing minimal answers in terms of an extension mechanism for cyclic states. As a result of the fact that cyclic strong covers can always be extended to total cyclic strong covers, this can be achieved without necessarily computing perfect models in their entirety.

3.1 Definitions [Pr88]. DB is *stratified* iff there is a level function $\ell : \mathcal{L} \rightarrow \mathbb{N}$ such that for each rule $A_1 \wedge A_2 \wedge \dots \wedge A_h \wedge \neg A_{h+1} \wedge \dots \wedge \neg A_{h+r} \rightarrow B_1 \vee B_2 \vee \dots \vee B_k$ in DB: (i) $\ell(B_j) = \ell(B_1)$ for each $j \leq k$, (ii) $\ell(A_i) \leq \ell(B_1)$ for each $i \leq h$, and (iii) $\ell(A_{h+i}) < \ell(B_1)$ for each $i \leq r$. We define $\ell(\mathbf{R}) = \ell(B_1)$.

If DB is stratified, then a model M of DB is *perfect* iff for each α , there is no set $M' \subset \{P \in M \mid \ell(P) = \alpha\}$ such that $\{P \in M \mid \ell(P) < \alpha\} \cup M' \models \{\mathbf{R} \in \text{DB} \mid \ell(\mathbf{R}) = \alpha\}$.

It is well known [Pr91] that for stratified databases, perfect and disjunctive stable models coincide. Theorem 1.6 dictates the requirements on a disjunction if it is to be extended to a minimal answer. As mentioned earlier however, we certainly do not wish to (have to) compute perfect models in their entirety, and in this respect the following result is key.

3.2 Theorem [Jo99]. Let \mathcal{Q} be a set of literals, then $\text{DB} \models \bigvee \mathcal{Q}$ iff \mathcal{Q} has no cyclic strong cover in DB.

By Theorem 2.6(a), if $M \subseteq \mathcal{L}$, then M is a perfect model of DB iff $\overline{M} \cup (\mathcal{L} - M)$ is a cyclic strong cover in DB. But moreover if \mathcal{C} is a cyclic strong cover in DB then \mathcal{C} is a cyclic strong cover of itself, thus (applying Theorem 3.2 with $\mathcal{Q} = \mathcal{C}$), $\text{DB} \not\models \bigvee \mathcal{C}$ and hence there is a perfect model M of DB such that $M \not\models \bigvee \mathcal{C}$, i.e., $\mathcal{C} \subseteq \overline{M} \cup (\mathcal{L} - M)$. Thus as in Section 2, (the complements of) cyclic strong covers may be regarded as partial perfect models, but crucially in this case they may always be extended to a total cyclic strong cover.

Using this observation we may easily reformulate Theorem 1.6 to give a characterisation of partial minimal answers in terms of cyclic strong covers.

3.3 Theorem. A set of predicates $\{A_j \mid j \leq r\}$ is contained in a minimal answer iff for each $i \leq r$ we may find a cyclic strong cover \mathcal{C}_i of $\{\neg A_i\} \cup \{A_j \mid j \leq r, j \neq i\}$ such that $\text{DB} \models \bigvee_{j \leq r} A_j \vee \bigvee \bigcap_{j \leq r} \mathcal{C}_j^+$.

Proof (\rightarrow). By Theorem 1.6, let $\{M_j | j \leq r\}$ be a set of perfect models of DB such that for each $i \leq r$, $M_i \cap \{A_j | j \leq r\} = \{A_i\}$, and $\text{DB} \models \bigvee_{j \leq r} A_j \vee \bigvee \bigcap_{j \leq r} (\mathcal{L} - M_j)$. If $\mathcal{C}_i = \overline{M_i} \cup (\mathcal{L} - M_i)$, then it is easy to see that \mathcal{C}_i satisfies the required conditions.

(\leftarrow). By Theorem 3.2, we may, for each i , pick a perfect model M_i of DB such that $\mathcal{C}_i \subseteq \overline{M_i} \cup (\mathcal{L} - M_i)$. Clearly $M_i \cap \{A_j | j \leq r\} = \{A_i\}$, and (since $\mathcal{L} - M_i \supseteq \mathcal{C}_i^+$), it is trivially the case that $\text{DB} \models \bigvee_{j \leq r} A_j \vee \bigvee \bigcap_{j \leq r} (\mathcal{L} - M_j)$. Thus the models M_i satisfy the conditions of Theorem 1.6, and the result then follows. ■

We now employ the above result to present our method for the construction of minimal answers. Such answers will be generated iteratively as a sequence $A_1, A_1 \vee A_2, A_1 \vee A_2 \vee A_3, \dots$. At each stage, the disjunction $A_1 \vee A_2 \vee \dots \vee A_r$ will be verified (as representing a partial minimal answer) by a corresponding sequence of cyclic strong covers $(\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_r)$ satisfying the conditions of Theorem 3.3. We will represent the disjunction and accompanying cyclic strong covers as the sequence $((A_i, \mathcal{C}_i) | i \leq r)$, and our method thus consists of repeatedly extending such sequences.

We commence, of course, with a pair of the form $((A, \mathcal{C}))$, where \mathcal{C} is a constructible extension of $\{\neg A\}$ (Definition 2.8), and by Theorem 2.4(c) it is trivially the case that such pairs satisfy the conditions of Theorem 3.3.

Our extension mechanism takes a sequence $((A_i, \mathcal{C}_i) | i \leq r)$ (satisfying the conditions of Theorem 3.3), and from this constructs a sequence $((A_i, \mathcal{G}_i) | i \leq r + 1)$ (also satisfying the conditions of Theorem 3.3) for which $\mathcal{G}_i \supseteq \mathcal{C}_i$ for each $i \leq r$. First note that if $((A_i, \mathcal{G}_i) | i \leq r + 1)$ has these properties, then

- (i) \mathcal{G}_{r+1} is a cyclic strong cover of $\{\neg A_{r+1}\} \cup \{A_i | i \leq r\}$,
- (ii) for $i \leq r$, \mathcal{G}_i is a cyclic strong cover of $\{\neg A_i\} \cup \{A_j | j \leq r + 1, j \neq i\}$ with $\mathcal{G}_i \supseteq \mathcal{C}_i$ (and since $\mathcal{C}_i \supseteq \{\neg A_i\} \cup \{A_j | j \leq r, j \neq i\}$, this is equivalent to \mathcal{G}_i being a cyclic strong cover of $\{A_{r+1}\} \cup \mathcal{C}_i$), and
- (iii) $\text{DB} \models \bigvee_{i \leq r+1} A_i \vee \bigvee \bigcap_{i \leq r+1} \mathcal{G}_i^+$.

Our extension will take place via two distinct phases. In the first phase we (attempt to) extend $((A_i, \mathcal{C}_i) | i \leq r)$ to a sequence $((A_i, \mathcal{F}_i) | i \leq r + 1)$ satisfying conditions (i) and (ii). We therefore pick⁴ A_{r+1} such that $\{\neg A_{r+1}\} \cup \{A_i | i \leq r\}$ has a cyclic strong cover \mathcal{F}_{r+1} , and for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has a cyclic strong cover \mathcal{F}_i . $((A_i, \mathcal{F}_i) | i \leq r + 1)$ then satisfies conditions (i) and (ii), but note that whilst $((A_i, \mathcal{F}_i) | i \leq r)$ continues to satisfy condition (iii) (since each $\mathcal{F}_i \supseteq \mathcal{C}_i$), it is not necessarily the case that $((A_i, \mathcal{F}_i) | i \leq r + 1)$ will satisfy condition (iii).

In the second phase we thus attempt to find (for each $i \leq r + 1$) a cyclic strong cover \mathcal{G}_i of \mathcal{F}_i such that $((A_i, \mathcal{G}_i) | i \leq r + 1)$ satisfies condition (iii) above. Clearly

⁴Methods of restricting the choice of A_{r+1} are presented in Definitions 3.7.3 and 5.7.3.

$((A_i, \mathcal{G}_i)|i \leq r + 1)$ continues to satisfy conditions (i) and (ii).

The existence of the sets $\{\mathcal{G}_i|i \leq r + 1\}$ can be thought of as *verifying* the choice of A_{r+1} and $\{\mathcal{F}_i|i \leq r + 1\}$ made in the first phase. If such verification is not possible, then we *truncate* $((A_i, \mathcal{F}_i)|i \leq r + 1)$ in order to seek alternative extensions of $((A_i, \mathcal{F}_i)|i \leq r)$.

The following definition captures these ideas.

3.4 Definitions.

- (a) A *cyclic state* of length r ($r \geq 1$) is a sequence $\mathcal{S} = ((A_i, \mathcal{C}_i)|i \leq r)$ such that
- (i) each A_i is a predicate,
 - (ii) each \mathcal{C}_i is a cyclic strong cover of $\{\neg A_i\} \cup \{A_j|j \leq r, j \neq i\}$, and
 - (iii) for $1 \leq j < r$, $\text{DB} \models \bigvee_{i \leq j} A_i \vee \bigvee \bigcap_{i \leq j} \mathcal{C}_i^+$.
- (b) A cyclic state $\mathcal{S}^* = ((A_i, \mathcal{D}_i)|i \leq r + k)$ ($k \geq 0$) is an *extension* of $\mathcal{S} = ((A_i, \mathcal{C}_i)|i \leq r)$ iff $\mathcal{C}_i \subseteq \mathcal{D}_i$ for each $i \leq r$.
- (c) A cyclic state $\mathcal{S} = ((A_i, \mathcal{C}_i)|i \leq r)$ is said to be:
- (i) *verified* iff $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee \bigcap_{i \leq r} \mathcal{C}_i^+$ (cf., Theorem 3.3),
 - (ii) *verifiable* iff \mathcal{S} has a verified extension,
 - (iii) *total* iff each \mathcal{C}_i is total, and
 - (iv) *complete* iff $\text{DB} \models \bigvee_{i \leq r} A_i$.

If $((A_i, \mathcal{C}_i)|i \leq r)$ is a cyclic state, note that $A_i \neq A_j$ for $i \neq j$ (since \mathcal{C}_i is consistent). Property (a)(iii) is a minor technical property capturing our intention that a cyclic state will only be extended in length if it is verified. This property also ensures that if $((A_i, \mathcal{C}_i)|i \leq r)$ is a cyclic state of length $r > 1$, then truncation yields a verified cyclic state $((A_i, \mathcal{C}_i)|i \leq r - 1)$. Note the obvious fact that if, for each $i \leq r$, \mathcal{D}_i is a cyclic strong cover of \mathcal{C}_i , then $\mathcal{S}^* = ((A_i, \mathcal{D}_i)|i \leq r)$ is a cyclic state. Moreover if $((A_i, \mathcal{C}_i)|i \leq r)$ is verified, then so is \mathcal{S}^* . The following proposition captures some of the basic properties of cyclic states.

3.5 Proposition.

- (a) If $\text{DB} \models \bigvee \{A_i|i \leq r\}$, and for each $i \leq r$, \mathcal{C}_i is a cyclic strong cover of $\{\neg A_i\} \cup \{A_j|j \leq r, j \neq i\}$, then for any non-empty subset $\{A_{i_1}, A_{i_2}, \dots, A_{i_k}\} \subseteq \{A_i|i \leq r\}$, we have that $((A_{i_h}, \mathcal{C}_{i_h})|h \leq k)$ is a verified cyclic state.
- (b) A set of predicates $\{A_i|i \leq r\}$ is contained in a minimal answer iff there is a verified cyclic state of the form $((A_i, \mathcal{C}_i)|i \leq r)$.
- (c) If $\text{DB} \models \bigvee \{A_i|i \leq r\}$, then $\{A_i|i \leq r\}$ is a minimal answer iff there is a cyclic state of the form $((A_i, \mathcal{C}_i)|i \leq r)$.
- (d) Every cyclic state of length 1 is verified.
- (e) If $\mathcal{S} = ((A_i, \mathcal{C}_i)|i \leq r)$ is verified, then \mathcal{S} has a complete total extension $((A_i, \mathcal{D}_i)|i$

$\leq r + k$) such that $\{A_j | r < j \leq r + k\} \subseteq \bigcap_{i \leq r} \mathcal{C}_i^+$.

(f) \mathcal{S} is verifiable iff \mathcal{S} has a verified extension of the same length iff \mathcal{S} has a complete extension iff \mathcal{S} has a total complete extension.

Proof (a) It is clear that for each $h \leq k$, \mathcal{C}_{i_h} is a cyclic strong cover of $\{\neg A_{i_h}\} \cup \{A_{i_l} | l \leq k, l \neq h\}$. If $1 \leq h \leq k$, then $\{A_j | j \leq r\} \subseteq \{A_{i_l} | l \leq h\} \cup \bigcap_{l \leq h} \mathcal{C}_{i_l}^+$, hence conditions (a)(iii) and (c)(i) of Definition 3.4 are satisfied.

(b) If $\{A_i | i \leq r + k\}$ is a minimal answer, then for each $i \leq r + k$ we may find a total cyclic strong cover \mathcal{C}_i of $\{\neg A_i\} \cup \{A_j | j \leq r + k, j \neq i\}$. By part (a), $((A_i, \mathcal{C}_i) | i \leq r)$ is verified. The converse follows immediately from Theorem 3.3.

(c) The implication from left to right follows from part (b). Conversely if $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a cyclic state, then since $\text{DB} \models \bigvee \{A_i | i \leq r\}$, \mathcal{S} is verified, and hence by part (b), $\{A_i | i \leq r\}$ is contained in a minimal answer. The result then follows from the fact that $\text{DB} \models \bigvee \{A_i | i \leq r\}$.

(d) This follows immediately from Theorem 2.4(c).

(e) Let \mathcal{A} be a minimal answer with $\mathcal{A} \subseteq \{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. For each $i \leq r$, \mathcal{C}_i is a cyclic strong cover of $\{\neg A_i\} \cup (\mathcal{A} - \{A_i\})$, thus $A_i \in \mathcal{A}$ (else \mathcal{C}_i is a cyclic strong cover of \mathcal{A} , thus contradicting the fact that $\text{DB} \models \bigvee \mathcal{A}$). For $i \leq r$, let \mathcal{D}_i be a total cyclic strong cover of \mathcal{C}_i .

If we write $\mathcal{A} = \{A_i | i \leq r + k\}$, and for each $i > r$, let \mathcal{D}_i be a total cyclic strong cover of $\{\neg A_i\} \cup (\mathcal{A} - \{A_i\})$, then $((A_i, \mathcal{D}_i) | i \leq r + k)$ is the required cyclic state.

(f) If $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is verifiable, then we may find a verified extension $\mathcal{S}' = ((A_i, \mathcal{C}'_i) | i \leq r + k)$. If $k > 0$, then by property 3.4(a)(iii), $\mathcal{S}^* = ((A_i, \mathcal{C}'_i) | i \leq r)$ is the required verified extension. By part (d), \mathcal{S}' has a total complete extension.

For the converse, any complete cyclic state is verified, hence if \mathcal{S} has a complete extension, it is verifiable. ■

As mentioned earlier, minimal answers will be constructed by (iteratively) extending cyclic states. Each individual step in this iteration will take an existing cyclic state $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ and form an *immediate extension* of \mathcal{S} . In the case when \mathcal{S} is unverified, the immediate extension will also have length r , the aim being to generate a verified extension of \mathcal{S} (of length r) via a sequence of one or more such extension steps. On the other hand, when \mathcal{S} is verified, the immediate extension of \mathcal{S} will be a (possibly unverified) cyclic state of length $r + 1$.

We will see that the extension of an unverified cyclic state yields information which can then be used to assist subsequent extension steps, and we thus discuss the extension of unverified cyclic states first.

Before doing so, we note that we require our extension mechanism to be *complete*,

meaning that every minimal answer can be generated via a sequence of immediate extension steps⁵. Now every minimal answer is witnessed by a total complete cyclic state, thus in order to achieve completeness we will insist that whenever \mathcal{S}^* is a total complete extension of \mathcal{S} , there is a (proper) immediate extension \mathcal{S}' of \mathcal{S} such that \mathcal{S}^* is an extension of \mathcal{S}' . The minimal answer represented by \mathcal{S}^* can then be constructed from \mathcal{S} via a sequence of immediate extension steps.

3.6 Extending unverified cyclic states.

Suppose now that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is an unverified cyclic state. Our aim is to find cyclic strong covers \mathcal{D}_i of \mathcal{C}_i such that $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee \bigcap_{i \leq r} \mathcal{D}_i^+$. There are broadly two ways in which we can extend the sets \mathcal{C}_i : either by adding some negative literal to some \mathcal{C}_{i_0} (which in turn will probably require the addition of further literals (both positive and negative) in order to re-form a cyclic strong cover); or the addition of some positive literal (i.e., predicate) to some \mathcal{C}_{i_0} (which again would probably require the addition of further literals). However, in order to achieve our desired goal, we specifically need to extend $\bigcap_{i \leq r} \mathcal{C}_i^+$, and this suggests that searching for positive literals which can be used to simultaneously extend all \mathcal{C}_i would be the more fruitful option. This option is also more appealing since it is the more constrained (i.e., has a narrower search space), and also because it allows within it an integral test to check whether or not the current cyclic state is indeed unverified. The following result encapsulates these ideas, and also shows that our approach to extending unverified cyclic states satisfies the completeness criteria mentioned above.

3.6.1 Lemma. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is unverified, and \mathcal{C} is a cyclic strong cover of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. Suppose also that \mathcal{S} is verifiable, and that $((A_i, \mathcal{D}_i) | i \leq r)$ is a verified cyclic state such that for each $i \leq r$, $\mathcal{D}_i \supseteq \mathcal{C}_i$.

Then $\mathcal{F} = \bigcap_{i \leq r} \mathcal{D}_i^+ - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-) \neq \emptyset$, and for each $A \in \mathcal{F}$ and each $i \leq r$ we may find a constructible extension \mathcal{C}_i^* of $\{A\} \cup \mathcal{C}_i$ such that $\mathcal{C}_i^* \subseteq \mathcal{D}_i$.

Proof. First note that $\mathcal{C}^+ \not\supseteq \bigcap_{i \leq r} \mathcal{D}_i^+$, for otherwise \mathcal{C} would be a cyclic strong cover of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{D}_i^+$, thus contradicting the fact that $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee \bigcap_{i \leq r} \mathcal{D}_i^+$. If $A \in \bigcap_{i \leq r} \mathcal{D}_i^+$, then $A \notin \bigcup_{i \leq r} \mathcal{D}_i^- \supseteq \bigcup_{i \leq r} \mathcal{C}_i^-$, and hence $\mathcal{F} \neq \emptyset$.

Finally, if $i \leq r$ then $\{A\} \cup \mathcal{C}_i \subseteq \mathcal{D}_i$, and the existence of \mathcal{C}_i^* is then given by Corollary 2.9. ■

This then allows us to define immediate extensions of unverified cyclic states.

⁵Of course we also require *correctness*, i.e., we should generate only minimal answers, but this is guaranteed by Theorem 3.3 and our use of verification.

3.6.2 Definition.

Suppose that we are given a cyclic state $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$, then an *immediate extension* \mathcal{S}^* of \mathcal{S} is formed as follows. Pick a constructible extension \mathcal{C} of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. (If no such \mathcal{C} exists, i.e., \mathcal{S} is verified, then immediate extensions of \mathcal{S} are as given in Definition 3.7.3 below.)

Pick a predicate $A \in \mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ such that for each $i \leq r$, $\{A\} \cup \mathcal{C}_i$ has a constructible extension \mathcal{C}_i^* , and let $\mathcal{S}^* = ((A_i, \mathcal{C}_i^*) | i \leq r)$.

If no such predicate A exists, i.e., \mathcal{S} is not verifiable, then \mathcal{S} has no immediate extension and the *truncation* of \mathcal{S} is given by $\mathcal{S}^* = ((A_i, \mathcal{C}_i) | i \leq r - 1)$.

Notes.

1. Note that if the predicate A exists, then the extension formed is a proper extension, since $A \in \mathcal{L} - \mathcal{C}^+ \subseteq \mathcal{L} - \bigcap_{i \leq r} \mathcal{C}_i^+$.
2. The constraint that $A \in \mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ clearly allows us to limit the search space, but nevertheless, it is still the case that we are *blindly* picking an element of $\mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$, and then determining whether the sets \mathcal{C}_i^* exist. Note however that if we make an incorrect choice for A then the cost of doing so is the cost of computing some of the sets \mathcal{C}_i^* . In fact this computation is not totally wasted, and we return to this point in the notes following Definition 3.7.3. In Section 5.6 we will see that compilation allows us to largely overcome this need to make a blind choice.
3. Constructible extensions \mathcal{C} of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$ can be computed with no additional effort by computing constructible extensions \mathcal{C}' of $\{A_i | i \leq r\}$, and then computing constructible extensions \mathcal{C} of $\mathcal{C}' \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. This observation is useful for a number of reasons.

Firstly it gives us a useful free test, since (by Theorem 3.2) if no such \mathcal{C}' exists, then $\text{DB} \models \bigvee_{i \leq r} A_i$, thus $\{A_i | i \leq r\}$ is a minimal answer and no further extension steps are required.

Secondly, if $\mathcal{S}^* = ((A_i, \mathcal{C}_i^*) | i \leq r)$ is an immediate extension of \mathcal{S} , then in order to extend \mathcal{S}^* we will (by Definition 3.6.2) need to compute constructible extensions of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} (\mathcal{C}_i^*)^+$, and as mentioned above, these can be computed by extending the (previously computed) constructible extensions of $\{A_i | i \leq r\}$. If no such constructible extension of $\{A_i | i \leq r\} \cup \bigcap_{i \leq r} (\mathcal{C}_i^*)^+$ exists, i.e., \mathcal{S}^* is verified, then again the subsequent extension of \mathcal{S}^* will employ constructible extensions of $\{A_i | i \leq r\}$, as is illustrated in the following section.

3.7 Extending verified cyclic states.

Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is verified (but $\text{DB} \not\models \bigvee_{i \leq r} A_i$, i.e., $\{A_i | i \leq r\}$

has a cyclic strong cover). As indicated in the discussion prior to Definition 3.4, an immediate extension of \mathcal{S} will have the form $\mathcal{S}^* = ((A_i, \mathcal{D}_i) | i \leq r+1)$, where (i) \mathcal{D}_{r+1} is a cyclic strong cover of $\{\neg A_{r+1}\} \cup \{A_i | i \leq r\}$, and (ii) for $i \leq r$, \mathcal{D}_i is a cyclic strong cover of $\{A_{r+1}\} \cup \mathcal{C}_i$.

We could of course blindly pick A_{r+1} , and then attempt to find the sets \mathcal{D}_i (if such exist). However note that cyclic strong covers of $\{\neg A_{r+1}\} \cup \{A_i | i \leq r\}$ can be computed by first computing a constructible extension \mathcal{C} of $\{A_i | i \leq r\}$, and then extending to a constructible extension of $\{\neg A_{r+1}\} \cup \mathcal{C}$. Choosing \mathcal{C} first, and then attempting to find A_{r+1} has three advantages:

- (a) We have already computed the constructible extensions of $\{A_i | i \leq r\}$ in Section 3.6 above (at least in the case when $r > 1$),
- (b) Given \mathcal{C} we can (by Lemma 3.7.1 below) prune the search space by insisting that $A_{r+1} \in \mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ without compromising our completeness criterion identified in the remarks following Proposition 3.5, and
- (c) Given \mathcal{C} , a predicate A_{r+1} and sets \mathcal{D}_i ($i \leq r+1$) are guaranteed to exist (by Lemma 3.7.2 below).

3.7.1 Lemma. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state, and that $((A_i, \mathcal{K}_i) | i \leq r+k)$ is a complete extension of \mathcal{S} with $k > 0$.

Then we may find a constructible extension \mathcal{C} of $\{A_i | i \leq r\}$ such that

- (i) $A_{r+1} \notin \mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-$,
- (ii) $\{\neg A_{r+1}\} \cup \mathcal{C}$ has a constructible extension \mathcal{D}_{r+1} such that $\mathcal{D}_{r+1} \subseteq \mathcal{K}_{r+1}$, and
- (iii) for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has a constructible extension \mathcal{D}_i such that $\mathcal{D}_i \subseteq \mathcal{K}_i$.

Proof. Trivially $A_{r+1} \in \bigcap_{i \leq r} \mathcal{K}_i^+ \subseteq \mathcal{L} - \bigcup_{i \leq r} \mathcal{K}_i^- \subseteq \mathcal{L} - \bigcup_{i \leq r} \mathcal{C}_i^-$. Now \mathcal{K}_{r+1} is a cyclic strong cover of $\{\neg A_{r+1}\} \cup \{A_i | i \leq r\}$, thus by Corollary 2.9 we may find a constructible extension \mathcal{C} of $\{A_i | i \leq r\}$ such that $\mathcal{C} \subseteq \mathcal{K}_{r+1}$. Since $\neg A_{r+1} \in \mathcal{K}_{r+1}$, we must have that $A_{r+1} \notin \mathcal{C}^+$.

Again using Corollary 2.9, parts (ii) and (iii) follow trivially from the facts that \mathcal{K}_{r+1} is a cyclic strong cover of $\{\neg A_{r+1}\} \cup \mathcal{C}$, and (for each $i \leq r$) \mathcal{K}_i is a cyclic strong cover of $\{A_{r+1}\} \cup \mathcal{C}_i$. ■

3.7.2 Lemma. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state, and \mathcal{C} is a cyclic strong cover of $\{A_i | i \leq r\}$. Then we may find a predicate A_{r+1} such that

- (i) $A_{r+1} \notin \mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-$,
- (ii) $\{\neg A_{r+1}\} \cup \mathcal{C}$ has a constructible extension \mathcal{D}_{r+1} , and
- (iii) for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has a constructible extension \mathcal{D}_i .

Proof. For each $i \leq r$, let \mathcal{K}_i be a total cyclic strong cover of \mathcal{C}_i , and let \mathcal{K} be a total cyclic strong cover of \mathcal{C} .

Since $\text{DB} \models \bigvee \{A_i \mid i \leq r\} \vee \bigvee \bigcap_{i \leq r} \mathcal{K}_i^+$ we cannot have that $\mathcal{K} \supseteq \{A_i \mid i \leq r\} \cup \bigcap_{i \leq r} \mathcal{K}_i^+$, thus pick $A_{r+1} \in \{A_i \mid i \leq r\} \cup \bigcap_{i \leq r} \mathcal{K}_i^+ - \mathcal{K}$. But then $A_{r+1} \notin \mathcal{K}^+ \supseteq \mathcal{C}^+$, and for each $i \leq r$, $A_{r+1} \in \mathcal{K}_i^+ \subseteq \mathcal{L} - \mathcal{K}_i^- \subseteq \mathcal{L} - \mathcal{C}_i^-$.

In addition, since \mathcal{K} is total, $\neg A_{r+1} \in \mathcal{K}$. Parts (i) and (ii) then follow from Corollary 2.9 and the facts that $\{\neg A_{r+1}\} \cup \mathcal{C} \subseteq \mathcal{K}$ and $\{A_{r+1}\} \cup \mathcal{C}_i \subseteq \mathcal{K}_i$. ■

Note that $A_{r+1} \notin \{A_i \mid i \leq r\}$, since $A_{r+1} \notin \mathcal{C}$. This then gives us our method of extending verified cyclic states.

3.7.3 Definition.

Let $\mathcal{S} = ((A_i, \mathcal{C}_i) \mid i \leq r)$ be a verified cyclic state, then an *immediate extension* \mathcal{S}^* of \mathcal{S} is formed as follows. Let \mathcal{C} be a constructible extension of $\{A_i \mid i \leq r\}$. (If no such \mathcal{C} exists then $\text{DB} \models \bigvee_{i \leq r} A_i$ and $\bigvee_{i \leq r} A_i$ is a minimal answer.)

Pick $A_{r+1} \in \mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ such that

- (i) $\{\neg A_{r+1}\} \cup \mathcal{C}$ has a constructible extension \mathcal{D}_{r+1} , and
- (ii) for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has a constructible extension \mathcal{D}_i .

Let $\mathcal{S}^* = ((A_i, \mathcal{D}_i) \mid i \leq r + 1)$.

Notes.

1. Note in both Definitions 3.6.2 and 3.7.3, that cyclic strong covers of sets the form $\{K\} \cup \mathcal{D}$ could (by Theorem 2.12) be computed from constructible extensions of $\{K\}$ in the reduced database $\text{DB}_{\mathcal{D}}$. In the case when $K = \neg A_{r+1}$ (Definition 3.7.3(i)) it may be preferable to adopt a different strategy (see 3 below).

2. As was the case in Section 3.6, cyclic strong covers may be computed and then found to be obsolete as far as extending the current cyclic state is concerned. Note however that the computation of any cyclic strong cover \mathcal{C} is not wasted (provided $\mathcal{C}^- \neq \emptyset$) since it can still be employed in the derivation of other minimal answers: for each $A \in \mathcal{C}^-$, $((A, \mathcal{C}))$ is a (verified) cyclic state of length 1.

3. Again we see that the constraint $A_{r+1} \in \mathcal{L} - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ allows us to prune the search space, but that beyond this we are still making a blind choice of A_{r+1} , and then testing whether it satisfies conditions (i) and (ii) above. This blind search is again not so detrimental however, since a failure to satisfy condition (i) still provides useful information as follows. A constructible extension \mathcal{D}_{r+1} of $\{\neg A_{r+1}\} \cup \mathcal{C}$ may be computed by first computing a constructible extension \mathcal{C}' of $\{\neg A_{r+1}\}$, and then computing \mathcal{D}_{r+1} as a constructible extension of $\mathcal{C}' \cup \mathcal{C}$. In the case when no such \mathcal{C}' exists, we can immediately discount A_{r+1} from belonging to any minimal answer.

In the case when \mathcal{C}' exists, but cannot be extended to \mathcal{D}_{r+1} , then as above, for each $A \in \mathcal{C}'^-$, the pair (A, \mathcal{C}') can still be employed in the derivation of other minimal answers, and the computation of \mathcal{C}' is not wasted.

4. If $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is verified, then by Proposition 3.5(e) we may find a minimal answer \mathcal{A} such that $\{A_i | i \leq r\} \subseteq \mathcal{A} \subseteq \{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. Moreover, it is easy to amend the proof of Lemma 3.7.2 to show that given a constructible extension \mathcal{C} of $\{A_i | i \leq r\}$ we can find some A_{r+1} in $\mathcal{A} - \{A_i | i \leq r\}$ (and hence in $\bigcap_{i \leq r} \mathcal{C}_i^+$) satisfying conditions (i) and (ii) of Definition 3.7.3.

Note however that in Definition 3.7.3 we are *not* able to insist that A_{r+1} is chosen from $\bigcap_{i \leq r} \mathcal{C}_i^+$, since this would compromise completeness. For example if $\text{DB} = \{A \vee B, A \rightarrow C, B \rightarrow D\}$, then $C \vee D$ is a minimal answer, but the only constructible extension of $\neg C$ (resp. $\neg D$) is $\{\neg C, \neg A, B\}$ (resp. $\{\neg D, \neg B, A\}$), thus the only cyclic states of length 1 (generated using constructible extensions) representing a sub-answer of $C \vee D$ are $((C, \{\neg C, \neg A, B\}))$ and $((D, \{\neg D, \neg B, A\}))$.

5. We have already mentioned that our method is both complete and correct. Notice that a truncation step is in effect an undo operation, and clearly we wish to prevent circularity by insisting that extension steps following truncation do not redo what has previously been undone. With this proviso, it is then clear that any sequence of cyclic states generated via immediate extension and truncation will eventually generate a complete cyclic state (i.e., a minimal answer).

We will see in Section 5.7 that compilation addresses the problems raised in notes 3 and 4 above.

3.8 Example. Suppose that DB contains the following rules

- | | | |
|---|--|-------------------------------------|
| 1. $Q_2 \wedge Q_3 \wedge \neg P_1 \rightarrow Q_1$ | 2. $Q_1 \wedge \neg P_2 \rightarrow Q_2$ | 3. $A_2 \wedge P_3 \rightarrow Q_3$ |
| 4. $A_1 \rightarrow Q_1 \vee Q_2$ | 5. $A_2 \vee P_2$ | 6. $A_1 \vee P_1$ |
| | | 7. $P_1 \vee P_3$ |

and that we wish to identify minimal answers containing Q_1 . We first search for a constructible extension of $\{\neg Q_1\}$, there being precisely two options, namely $\{\neg Q_1, \neg Q_2, \neg Q_3, P_1, P_2, \neg A_1, \neg P_3, \neg A_2\}$, and $\{\neg Q_1, \neg A_1, Q_2, P_1, \neg P_2, A_2\}$. This then shows that Q_1 does indeed belong to a minimal answer.

Let us consider extending $((Q_1, \mathcal{C}_1))$ using Definition 3.7.3, where $\mathcal{C}_1 = \{\neg Q_1, \neg Q_2, \neg Q_3, P_1, P_2, \neg A_1, \neg P_3, \neg A_2\}$. We first generate a constructible extension of $\{Q_1\}$, there being precisely five: suppose we choose $\mathcal{C} = \{Q_1, \neg P_1, A_1\}$, then again we are required to pick a predicate in $\mathcal{L} - (\mathcal{C}^+ \cup \mathcal{C}_1^-) = \{P_1, P_2\}$, satisfying the conditions of Definition 3.7.3. P_1 is an easy choice, since $\neg P_1 \in \mathcal{C}$ ensures that \mathcal{C} itself is a constructible extension of $\{\neg P_1\} \cup \mathcal{C}$ (Definition 3.7.3(i)), and $P_1 \in \mathcal{C}_1$ ensures that \mathcal{C}_1 itself is a

constructible extension of $\{P_1\} \cup \mathcal{C}_1$ (Definition 3.7.3(ii)). Thus $((Q_1, \mathcal{C}_1), (P_1, \mathcal{C}))$ is an immediate extension of $((Q_1, \mathcal{C}_1))$.

This cyclic state is now extended using Definition 3.6.2. We first check for verification by attempting to generate a constructible extension \mathcal{D} of $\{Q_1, P_1\} \cup (\mathcal{C}_1^+ \cap \mathcal{C}^+) = \{Q_1, P_1\}$ (since $\mathcal{C}_1^+ \cap \mathcal{C}^+ = \emptyset$), there being precisely one, namely $\mathcal{D} = \{Q_1, P_1, Q_3, A_2\}$.

We then pick a predicate in $\mathcal{L} - (\mathcal{D}^+ \cup \mathcal{C}_1^- \cup \mathcal{C}^-) = \{P_2\}$ satisfying the conditions of Definition 3.6.2. Fortunately $\mathcal{C}_1 \cup \{P_2\}$ and $\mathcal{C} \cup \{P_2\}$ are both strong covers, and are clearly cyclic since the addition of a positive literal has no effect on this property. Thus $((Q_1, \mathcal{C}_1 \cup \{P_2\}), (P_1, \mathcal{C} \cup \{P_2\}))$ is an immediate extension of $((Q_1, \mathcal{C}_1), (P_1, \mathcal{C}))$. It is also easy to check that this extension is verified (the only constructible extension of $\{Q_1, P_1\} \cup (\mathcal{C}_1^+ \cap \mathcal{C}^+)$ was noted above to be $\{Q_1, P_1, Q_3, A_2\}$, hence any cyclic strong cover of $\{Q_1, P_1\} \cup ((\mathcal{C}_1 \cup \{P_2\})^+ \cap (\mathcal{C} \cup \{P_2\})^+)$ must contain $\{Q_1, P_1, Q_3, A_2, P_2\}$, no extension of which can be a strong cover by virtue of the rule $A_2 \vee P_2$).

Employing Definition 3.7.3 we search for a constructible extension of $\{Q_1, P_1\}$ (\mathcal{D} being the only choice), and pick a predicate in $\mathcal{L} - (\mathcal{D}^+ \cup (\mathcal{C}_1 \cup \{P_2\})^- \cup (\mathcal{C} \cup \{P_2\})^-) = \{P_2\}$. $\{\neg P_2\} \cup \mathcal{D}$ is a cyclic strong cover (cf., Definition 3.7.3(i)) and P_2 is already a member of $\mathcal{C}_1 \cup \{P_2\}$ and $\mathcal{C} \cup \{P_2\}$ (cf., Definition 3.7.3(ii)), hence $((Q_1, \mathcal{C}_1 \cup \{P_2\}), (P_1, \mathcal{C} \cup \{P_2\}), (P_2, \{\neg P_2\} \cup \mathcal{D}))$ is an immediate extension of $((Q_1, \mathcal{C}_1 \cup \{P_2\}), (P_1, \mathcal{C} \cup \{P_2\}))$. We have already noted above that $\{Q_1, P_1, P_2\}$ has no cyclic strong cover, and hence we may conclude (using Proposition 3.5(c)) that it is a minimal answer.

Note in this example that our first verified cyclic state was extended to an unverified cyclic state, which was in turn extended (in a single step) to a verified cyclic state. This in turn was directly extended to another verified state. In addition, it is clear that an unverified state may be extended to another unverified state (if more than one predicate needs to be added in order to achieve verification).

§4. Unstratified databases

For unstratified databases, cyclic strong covers are not necessarily extendible to total cyclic strong covers, and we are thus (apparently) unable to characterise partial minimal answers without computing *total* cyclic strong covers. In addition, testing verification, i.e., whether $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee \bigcap_{i \leq r} \mathcal{C}_i^+$ (cf., Theorem 3.3) requires the computation of total cyclic strong covers (Theorem 2.6(b)), at least in the case when the cyclic state in question is not verified.

As discussed in Section 2, disjunctive stable models can be generated by (implic-

itly) adding to our database the denial rules $\{P \wedge \neg P \rightarrow \text{FALSE} \mid P \in \mathcal{L}\}$, thus forcing every strong cover of $\{\text{FALSE}\}$ to be total. Cyclic states then encode sequences of disjunctive stable models satisfying the conditions of Theorem 1.6, and the (top-down) construction of cyclic strong covers indicated in Section 2 provides us with a means of testing verification.

As mentioned earlier, the need to compute models in their entirety is undesirable, and in the following section we show how compilation can be employed to overcome this.

§5. Compilation

In this section we show that pre-processing the computation of cyclic strong covers can be employed to greatly simplify (and hence reduce the cost of) the run-time computation. In addition, and as a by-product, pre-processing is shown to resolve the problems raised in Section 4 above.

Pre-processing a computation is based upon two assumptions. Firstly that the computation can itself be computationally expensive - in our case for example deciding whether a given literal belongs to some cyclic strong cover is Σ_2^P -complete in the propositional case (cf., Appendix B). Secondly, that a deductive database is subdivided into a large but transient extensional database (containing facts and/or disjunctive facts) and a more static intensional database (the rule base) where the real expense and complexities of the computation lie.

The expense of the computation suggests that pre-processing may well prove to be important in a practical setting. The fact that the complexities lie in the intensional database, and the transient nature of the extensional database suggests that pre-processing should focus on, and be restricted to the intensional database. Subsequent updates to the intensional database would of course require that the pre-processing phase were repeated, but the more frequent changes to the extensional database would not.

In the current context, this approach therefore requires that the computation of cyclic strong covers be subdivided into a pre-processing phase (against the intensional database), and then a subsequent processing phase. It is also worth noting as an aside that such a partitioning of the overall computation is clearly more feasible for computations that are *top-down*, i.e., which start with the intensional database. As was seen in Section 2, this is the case with the computation of cyclic strong covers. In our case, the subsequent processing phase will employ the extensional database only, in which case pre-processing is usually referred to as *compilation*.

We will also see that the post-compilation construction of minimal answers reduces to minimal model reasoning in $\text{EXT}(\text{DB})$, this simplification being reflected in a reduction in the computational complexity (see Appendix B).

Throughout this section we therefore assume that \mathcal{L} is the disjoint union of $\text{EXT}(\mathcal{L})$ and $\text{INT}(\mathcal{L})$. If \mathcal{Q} is a set of literals, then let $\mathcal{Q}_{ext} = \{K \in \mathcal{Q} \mid K \in \text{EXT}(\mathcal{L}) \text{ or } \neg K \in \text{EXT}(\mathcal{L})\}$, and $\mathcal{Q}_{int} = \{K \in \mathcal{Q} \mid K \in \text{INT}(\mathcal{L}) \text{ or } \neg K \in \text{INT}(\mathcal{L})\} = \mathcal{Q} - \mathcal{Q}_{ext}$. Notice that $(\mathcal{Q}^-)_{ext} = (\mathcal{Q}_{ext})^-$, etc. We also make the (usual) assumption that for each rule \mathbf{R} , either

- (i) $\text{conseq}(\mathbf{R}) \subseteq \text{EXT}(\mathcal{L})$ and $\text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}) = \emptyset$, or
- (ii) $\text{conseq}(\mathbf{R}) \subseteq \text{INT}(\mathcal{L})$ and $\text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}) \neq \emptyset$ (thus $\text{antec}(\mathbf{R}) \neq \emptyset$, cf., Section 1.1).

In case (ii) the assumption that the body of \mathbf{R} is non-empty is of course a technical requirement that can be achieved artificially without loss of generality. We let $\text{EXT}(\text{DB}) = \{\mathbf{R} \in \text{DB} \mid \text{conseq}(\mathbf{R}) \subseteq \text{EXT}(\mathcal{L})\}$, and $\text{INT}(\text{DB}) = \text{DB} - \text{EXT}(\text{DB}) = \{\mathbf{R} \in \text{DB} \mid \text{conseq}(\mathbf{R}) \subseteq \text{INT}(\mathcal{L})\}$.

Note that a rule in $\text{EXT}(\text{DB})$ has the form $\bigvee \mathcal{E}$, where $\mathcal{E} \subseteq \text{EXT}(\mathcal{L})$. In particular, minimal and disjunctive stable models of $\text{EXT}(\text{DB})$ coincide, thus if Φ is a formula in $\text{EXT}(\mathcal{L})$, then $\text{EXT}(\text{DB}) \models \Phi$ iff Φ is true in every minimal model of DB . In particular if $\mathcal{F} \subseteq \text{EXT}(\mathcal{L})$, then $\text{EXT}(\text{DB}) \models \bigvee \mathcal{F}$ iff there is a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $\mathcal{E} \subseteq \mathcal{F}$.

This partitioning of \mathcal{L} can be viewed as a very weak form of stratification, and indeed yields a weakened form of Theorem 3.2 as follows: Let us say that a consistent set of literals \mathcal{Q} is *int-total* iff $\mathcal{Q}^- \cup \mathcal{Q}^+ \supseteq \text{INT}(\mathcal{L})$. If \mathcal{C} is an int-total cyclic strong cover, then $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{C}_{ext}^+$, thus we may find a minimal model $M_0 \subseteq \text{EXT}(\mathcal{L})$ of $\text{EXT}(\text{DB})$ such that $M_0 \cap \mathcal{C}_{ext}^+ = \emptyset$. But then it is easy to show (as in the comments preceding Theorem 2.6) that $\mathcal{C}_{ext}^- \subseteq M_0$, and hence that $\mathcal{C} \cup \overline{M_0} \cup (\text{EXT}(\mathcal{L}) - M_0)$ is a total cyclic strong cover extending \mathcal{C} . This then yields the following result, which is re-stated from [Jo99a].

5.1 Theorem [Jo99a].

- (a) Every int-total cyclic strong cover can be extended to a total cyclic strong cover.
- (b) If \mathcal{Q} is a set of literals, then $\text{DB} \models \bigvee \mathcal{Q}$ iff \mathcal{Q} has no int-total cyclic strong cover.

So query processing can be achieved by the construction of int-total cyclic strong covers (and thus again does not require the computation of disjunctive stable models in their entirety), thus our aim is to partition this construction into a compilation phase against $\text{INT}(\text{DB})$, and then a further step against $\text{EXT}(\text{DB})$. In [Jo99, Jo99a] it was shown that this partitioning can be achieved using the notion of an *int-total weakly cyclic cover*. In Sections 5.2 and 5.3 we therefore briefly restate (from [Jo99,

Jo99a]) the motivation, definition and properties of weakly cyclic covers. We then detail in Sections 5.4-5.8 how our method for generating minimal answers is amended and enhanced by the prior application of compilation.

Before presenting the definition of weakly cyclic covers, let us briefly consider how this partitioning can be achieved. For strong covers this is trivial, since a strong cover in DB is simply a set that is both a strong cover in INT(DB) and a strong cover in EXT(DB). Moreover a consistent set of literals \mathcal{C} is a strong cover in EXT(DB) iff $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{C}_{ext}^+$.

In a cyclic tree, every leaf node is a rule node. Moreover a rule node $rn_{\mathbf{R}}$ is a leaf node iff $\text{antec}(\mathbf{R}) = \emptyset$, which (by assumptions (i) and (ii) above) is the case iff $\mathbf{R} \in \text{EXT}(\text{DB})$. If $rn_{\mathbf{R}}$ is such a leaf node, with parent n , then $\text{conseq}(\mathbf{R}) \cap \text{CYC}(n) \neq \emptyset$, thus $\text{CYC}(n)$ (and hence the branch to n) must contain a predicate node labelled with an extensional predicate. If m is the top-most such predicate node, then $\text{CYC}(m) = \{\text{lab}(m)\}$, thus the child of m must be a rule node of the form $rn_{\mathbf{S}}$, where $\text{lab}(m) \in \text{conseq}(\mathbf{S})$, and hence $\mathbf{S} \in \text{EXT}(\text{DB})$, i.e., $m = n$. Thus predicate nodes are labelled with extensional predicates iff their child is a leaf node.

Let us therefore say that a *partial cyclic tree* satisfies the conditions of Definition 2.3, with the exception that every leaf node is a predicate node labelled with an extensional predicate. Such trees are defined (constructed) entirely within INT(DB). To *complete* a partial cyclic tree in order to form a cyclic tree, we need to extend each such leaf node n with a rule node $rn_{\mathbf{R}}$, where $\mathbf{R} \in \text{EXT}(\text{DB})$. Suppose that $\text{lab}(n) = P$ and that $\mathbf{R} = \bigvee \mathcal{E}$, then in order to ensure that the extended tree continues to satisfy the conditions of Definition 2.3, we must have that $P \in \mathcal{E}$ (since $\text{CYC}(n) = \{P\}$) and $\mathcal{O}(rn_{\mathbf{R}}) = \mathcal{E} - \{P\}$ is disjoint from $\text{Pred}(\mathcal{T})$ (or equivalently disjoint from $\text{Pred}(\mathcal{T})_{ext}$ since $\mathcal{E} \subseteq \text{EXT}(\mathcal{L})$).

The extended tree \mathcal{T}' then has the properties that $\text{Pred}(\mathcal{T}') = \text{Pred}(\mathcal{T})$, $\mathcal{N}(\mathcal{T}') = \mathcal{N}(\mathcal{T})$ and $\mathcal{O}(\mathcal{T}') \cap \text{INT}(\mathcal{L}) = \mathcal{O}(\mathcal{T})$ and $\mathcal{O}(\mathcal{T}') \cap \text{EXT}(\mathcal{L}) = \bigcup \{\mathcal{O}(rn_{\mathbf{R}}) \mid rn_{\mathbf{R}} \text{ is a leaf in } \mathcal{T}'\}$. For example if \mathcal{T} is the tree obtained from \mathcal{T}_4 (Figure 2.2) by removing the leaf nodes rn_7 and rn_5 , then \mathcal{T} is a partial cyclic tree with $\text{Pred}(\mathcal{T}_4) = \text{Pred}(\mathcal{T})$, $\mathcal{N}(\mathcal{T}_4) = \mathcal{N}(\mathcal{T})$ and $\mathcal{O}(\mathcal{T}_4) = \mathcal{O}(\mathcal{T}) \cup \mathcal{O}(rn_7) \cup \mathcal{O}(rn_5)$.

5.2 Definition [Jo99, Jo99a].

(a) A consistent set of literals \mathcal{C} is *weakly cyclic* (in INT(DB)) iff there is a set $\{\mathcal{T}_i \mid i \leq k\}$ of partial cyclic trees such that $\mathcal{C}^- = \bigcup_{i \leq k} \text{Pred}(\mathcal{T}_i)$, and $\mathcal{C}^+ \supseteq \bigcup_{i \leq k} (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$.

Let \mathcal{C} be weakly cyclic in INT(DB) and f be a function $f : \mathcal{C}_{ext}^- \rightarrow \text{EXT}(\text{DB})$ such that for each $P \in \mathcal{C}_{ext}^-$, $f(P) = \bigvee \mathcal{E}_P$, where $P \in \mathcal{E}_P$ and $(\mathcal{E}_P - \{P\}) \cap \mathcal{C}_{ext}^- = \emptyset$. The set $\mathcal{C} \cup \bigcup \{\mathcal{E}_P - \{P\} \mid P \in \mathcal{C}_{ext}^-\}$ is said to be a *completion* of \mathcal{C} in DB.

(b) \mathcal{C} is a *weakly cyclic cover* (in INT(DB)) iff \mathcal{C} is weakly cyclic and a strong cover in

INT(DB).

Thus a completion of a weakly cyclic cover is formed by completing each of the cyclic trees which form the weakly cyclic cover⁶. Note that if \mathcal{D} is a completion of \mathcal{C} , then $\mathcal{D} - \mathcal{C} \subseteq \text{EXT}(\mathcal{L})$. Note also that the computation of weakly cyclic covers takes place entirely in INT(DB), and the computation of completions takes place entirely in EXT(DB).

Note that a partial cyclic tree \mathcal{T} in INT(DB) cannot necessarily be extended to a cyclic tree in DB. However it is easy to observe that \mathcal{T} can always be extended to a cyclic tree in $\text{INT}(\text{DB}) \cup \{E \mid E \in \text{Pred}(\mathcal{T})_{ext}\}$ by appending to each predicate leaf node n the rule node labelled with the unit rule $lab(n)$. This, together with Theorem 2.6(c) then yields part (d) of the following theorem. Parts (a) - (c) capture the trivial relationships between weakly cyclic covers and their completions, and are restated from [Jo99, Jo99a].

5.3 Proposition.

- (a) Let \mathcal{D} be a cyclic strong cover. Then \mathcal{D} is a weakly cyclic cover, and moreover if \mathcal{C} is a weakly cyclic cover with $\mathcal{C} \subseteq \mathcal{D}$, there is a completion \mathcal{C}' of \mathcal{C} such that $\mathcal{C}' \subseteq \mathcal{D}$.
- (b) Let \mathcal{C} be a weakly cyclic cover. If \mathcal{D} is a completion of \mathcal{C} , then \mathcal{D} is a cyclic strong cover iff $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+$.
- (c) If \mathcal{Q} is a set of literals, then $\text{DB} \models \bigvee \mathcal{Q}$ iff whenever \mathcal{C} is an int-total weakly cyclic cover of \mathcal{Q} and \mathcal{D} is a completion of \mathcal{C} , then $\text{EXT}(\text{DB}) \models \bigvee \mathcal{D}_{ext}^+$.
- (d) A consistent set of literals \mathcal{C} is weakly cyclic (in INT(DB)) iff for each $P \in \mathcal{C}^-$ there is a partial cyclic tree \mathcal{T} for P in DB such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$.

Proof (d) By the above remark, \mathcal{C} is its own completion in $\text{DB}^* = \text{INT}(\text{DB}) \cup \{E \mid E \in \mathcal{C}_{ext}^-\}$, and is therefore cyclic in DB^* . Thus by Theorem 2.6(c), if $P \in \mathcal{C}^-$ we may find a cyclic tree \mathcal{T}^* for P in DB^* such that $\mathcal{S}(\mathcal{T}^*) \subseteq \mathcal{C}$. If \mathcal{T} is the tree obtained from \mathcal{T}^* by removing the leaf nodes, then \mathcal{T} is a partial cyclic tree for P in INT(DB) with $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$. The converse is trivial. ■

We note, as an aside, that \mathcal{C} has a completion \mathcal{D} for which $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+$ iff $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{C}_{ext}$ (i.e., there is a minimal model M of EXT(DB) such that $\mathcal{C}_{ext}^- \subseteq M \subseteq \text{EXT}(\mathcal{L}) - \mathcal{C}_{ext}^+$).

⁶Alternatively we might regard a completion of \mathcal{C} as being constructed from \mathcal{C} and EXT(DB) via a hyper-resolution like operator.

Note that since weakly cyclic covers are defined entirely in $\text{INT}(\text{DB})$, we may easily show that if \mathcal{Q} is a consistent set of literals, then \mathcal{C} is a weakly cyclic cover of \mathcal{Q} iff \mathcal{C} has the form $\mathcal{C} = \mathcal{C}' \cup \mathcal{Q}_{ext}$ where \mathcal{C}' is a weakly cyclic cover of \mathcal{Q}_{int} and $\mathcal{C}' \cup \mathcal{Q}_{ext}$ is consistent.

5.4 The compilation process.

Clearly the compilation process entails the computation of int-total weakly cyclic covers. Note that during compilation we can (if desired) apply subsumption to remove redundancy, since if \mathcal{C} and \mathcal{D} are int-total weakly cyclic covers with $\mathcal{C} \subseteq \mathcal{D}$, then any completion of \mathcal{D} contains a completion of \mathcal{C} . We assume therefore that the compilation process generates a set COMP of int-total weakly cyclic covers, such that every int-total weakly cyclic cover is a superset of some element of COMP .

This construction can clearly be achieved using Proposition 5.3(d) and a variant of the approach presented in Section 2 for the construction of cyclic strong covers. We can of course achieve int-totally by the implicit addition of $\{P \wedge \neg P \rightarrow \text{FALSE} \mid P \in \text{INT}(\mathcal{L})\}$ to $\text{INT}(\text{DB})$.

Notice that the set COMP resulting from our compilation process is in general going to be voluminous, and in this sense compilation represents a trade-off between storage and run-time efficiency. In view of the complexities of query processing, and the relative modern-day costs of storage versus user time, we believe that this trade-off is certainly worth considering. We would also argue that our suggested approach compares favourably (with respect to both computational cost and the use of storage) with approaches which generate the full set of disjunctive stable models since the weakly cyclic covers generated are int-total (as opposed to total), and their construction is immune to updates to the extensional database (an immunity not shared by the computation of full models). We return to this issue in Section 9.

Note that int-total cyclic strong covers and disjunctive stable models can be computed easily from COMP as follows.

5.5 Proposition.

- (a) If \mathcal{Q} is a consistent set of literals, and \mathcal{S} is an int-total cyclic strong cover of \mathcal{Q} , then we may find some $\mathcal{C} \in \text{COMP}$ such that $\mathcal{C} \supseteq \mathcal{Q}_{int}$, $\mathcal{C} \cup \mathcal{Q}_{ext}$ is consistent, and a completion \mathcal{D} of $\mathcal{C} \cup \mathcal{Q}_{ext}$ such that $\mathcal{D} \subseteq \mathcal{S}$ (thus $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+$ and \mathcal{D} is an int-total cyclic strong cover of \mathcal{Q}).
- (b) A consistent set of literals \mathcal{D} is an int-total cyclic strong cover iff (i) for each $P \in \mathcal{D}_{ext}^-$ there is a rule $\bigvee \mathcal{E}_P \in \text{EXT}(\text{DB})$ such that $P \in \mathcal{E}_P$ and $\mathcal{E}_P - \{P\} \subseteq \mathcal{D}$, (ii) $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+$, and (iii) there exists a $\mathcal{C} \in \text{COMP}$ such that $\mathcal{C} \subseteq \mathcal{D}$.
- (c) M is a disjunctive stable model of DB iff $M \cap \text{EXT}(\mathcal{L})$ is a minimal model of

EXT(DB), and there exists a $\mathcal{C} \in \text{COMP}$ such that $\overline{\mathcal{C}} \subseteq M \cup (\overline{\mathcal{L} - M})$.

Let us denote by $\text{COMP}(\mathcal{Q})$, the set of int-total cyclic strong covers of \mathcal{Q} obtained using part (a) of the above proposition, i.e., $\text{COMP}(\mathcal{Q}) = \{\mathcal{D} \mid \exists \mathcal{C} \in \text{COMP}, \mathcal{C} \supseteq \mathcal{Q}_{int}, \mathcal{C} \cup \mathcal{Q}_{ext} \text{ is consistent}, \mathcal{D} \text{ is a completion of } \mathcal{C} \cup \mathcal{Q}_{ext}, \text{ and } \text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+\}$.

We now turn our attention to the run-time computation of minimal answers following compilation. As in Section 3, this is again based upon two distinct processes, depending on whether the cyclic state to be extended is verified or not.

5.6 Extending unverified cyclic states.

As in Section 3, our aim is to extend an unverified cyclic state $\mathcal{S} = ((A_i, \mathcal{C}_i) \mid i \leq r)$ to a verified cyclic state $\mathcal{S}^* = ((A_i, \mathcal{D}_i) \mid i \leq r)$, and again we will seek to achieve this by the simultaneous addition of some predicate(s) to each \mathcal{C}_i^+ . In Section 3.6.2 (Note 2) we observed that the search for such a predicate was partially blind. In the current context, each \mathcal{C}_i will already be int-total, thus such predicates must be taken from $\text{EXT}(\mathcal{L})$, and we will see (Theorem 5.6.1) below that the search for such can be driven by a search of $\text{EXT}(\text{DB})$, thus further limiting the search space.

Given that we are restricted to predicates from $\text{EXT}(\mathcal{L})$, it is worth noting that if we extend an int-total cyclic strong cover \mathcal{C} with a set of predicates $\mathcal{B} \subseteq \text{EXT}(\mathcal{L})$, then $\mathcal{C} \cup \mathcal{B}$ is a cyclic strong cover iff $\mathcal{C} \cup \mathcal{B}$ is consistent (i.e., $\mathcal{C}_{ext}^- \cap \mathcal{B} = \emptyset$) and $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{B} \vee \bigvee \mathcal{C}_{ext}^+$.

As in Section 3.6 we first perform a test to ensure that \mathcal{S} is not verified, and then use the results of this test to constrain the subsequent extension operation. The following theorem encapsulates this test and extension operation, and also demonstrates that the required completeness property (given after Proposition 3.5) is satisfied.

5.6.1 Theorem. Let $\mathcal{S} = ((A_i, \mathcal{C}_i) \mid i \leq r)$ be a cyclic state, where each \mathcal{C}_i is int-total, and $\mathcal{V} = \{A_i \mid i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$. Suppose that \mathcal{S} is not verified and that $\mathcal{D} \in \text{COMP}(\mathcal{V})$. Suppose also that \mathcal{S} is verifiable, and that $((A_i, \mathcal{D}_i) \mid i \leq r)$ is a verified cyclic state, with each $\mathcal{D}_i \supseteq \mathcal{C}_i$.

Then we may find a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that for each $i \leq r$, $\mathcal{E} - \mathcal{D} \subseteq \mathcal{D}_i^+ - \mathcal{C}_i^-$, $\mathcal{E} \cap (\mathcal{D} - \mathcal{C}_i) \neq \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{E} - \mathcal{D}) \vee \bigvee (\mathcal{C}_i^+)_{ext}$.

Proof. Let $\mathcal{W} = \{A_i \mid i \leq r\} \cup \bigcap_{i \leq r} \mathcal{D}_i^+$, then $\mathcal{V} \subseteq \mathcal{W}$, $\mathcal{W}_{int} = \mathcal{V}_{int}$ and (hence) $\mathcal{W} - \mathcal{V} \subseteq \text{EXT}(\mathcal{L})$. Now \mathcal{D} is an int-total cyclic strong cover of \mathcal{V} , hence $\mathcal{D} \cup \mathcal{W} = \mathcal{D} \cup \mathcal{W}_{ext} = \mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$.

If $\mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$ is inconsistent, then there is some $P \in \mathcal{D}_{ext}^-$ such that $P \in \mathcal{W}_{ext} - \mathcal{V}_{ext}$, and (by Proposition 5.5(b)) there is some rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that

$P \in \mathcal{E}$ and $\mathcal{E} - \{P\} \subseteq \mathcal{D}$, thus $\mathcal{E} \subseteq \mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$. If $\mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$ is consistent, then there is some $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $\mathcal{E} \subseteq \mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$ (else by the above remark $\mathcal{D} \cup (\mathcal{W}_{ext} - \mathcal{V}_{ext})$ is an int-total cyclic strong cover of \mathcal{W} thus contradicting the fact that $((A_i, \mathcal{D}_i) | i \leq r)$ is verified).

But then $\mathcal{E} - \mathcal{D} \subseteq \mathcal{W}_{ext} - \mathcal{V}_{ext} \subseteq \bigcap_{i \leq r} \mathcal{D}_i^+ \subseteq \mathcal{L} - \bigcup_{i \leq r} \mathcal{D}_i^- \subseteq \mathcal{L} - \bigcup_{i \leq r} \mathcal{C}_i^-$. Suppose that $\mathcal{E} \cap (\mathcal{D} - \mathcal{C}_i) = \emptyset$, then $\mathcal{E} \cap \mathcal{D} \subseteq \mathcal{C}_i$ and therefore $\mathcal{E} = (\mathcal{E} - \mathcal{D}) \cup (\mathcal{E} \cap \mathcal{D}) \subseteq \mathcal{D}_i \cup \mathcal{C}_i \subseteq \mathcal{D}_i$, thus contradicting the fact that \mathcal{D}_i is a strong cover. Similarly $\text{EXT}(\text{DB}) \not\equiv \bigvee (\mathcal{E} - \mathcal{D}) \vee \bigvee (\mathcal{C}_i^+)_{ext}$, since $(\mathcal{E} - \mathcal{D}) \cup \mathcal{C}_i^+ \subseteq \mathcal{D}_i$. ■

5.6.2 Definition.

Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a cyclic state, where each \mathcal{C}_i is int-total. An *immediate extension* $\mathcal{S}^* = ((A_i, \mathcal{C}_i^*) | i \leq r)$ of \mathcal{S} is computed as follows. Pick $\mathcal{D} \in \text{COMP}(\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+)$. By Proposition 5.5(a), if no such \mathcal{D} exists, then \mathcal{S} is verified, in which case immediate extensions of \mathcal{S} are defined in Definition 5.7.3 below.

Pick $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that for each $i \leq r$, $(\mathcal{E} - \mathcal{D}) \cap \mathcal{C}_i^- = \emptyset$, $\mathcal{E} \cap (\mathcal{D} - \mathcal{C}_i) \neq \emptyset$ and $\text{EXT}(\text{DB}) \not\equiv \bigvee (\mathcal{E} - \mathcal{D}) \vee \bigvee (\mathcal{C}_i^+)_{ext}$. We then set $\mathcal{C}_i^* = (\mathcal{E} - \mathcal{D}) \cup \mathcal{C}_i$.

Notice that there must be some i_0 for which $\mathcal{C}_{i_0} \subset \mathcal{C}_{i_0}^*$, for otherwise $\mathcal{E} - \mathcal{D} \subseteq \bigcap_{i \leq r} \mathcal{C}_i^+$, thus $\mathcal{E} \subseteq \mathcal{D} \cup (\mathcal{E} - \mathcal{D}) \subseteq \mathcal{D} \cup \bigcap_{i \leq r} \mathcal{C}_i^+ \subseteq \mathcal{D}$, contradicting the fact that $\text{EXT}(\text{DB}) \not\equiv \bigvee \mathcal{D}_{ext}^+$. Thus \mathcal{S}^* is a proper extension of \mathcal{S} .

If no such rule $\bigvee \mathcal{E}$ exists in $\text{EXT}(\text{DB})$ then \mathcal{S} is not verifiable (by Theorem 5.6.1), \mathcal{S} has no immediate extension, and the truncation of \mathcal{S} is given by $((A_i, \mathcal{C}_i) | i \leq r - 1)$.

5.7 Extending verified cyclic states.

Whilst employing basically the same mechanism as was the case in Section 3.7, the extension of verified cyclic states is also considerably simplified.

We first present two results (whose proofs are virtually identical to those of Lemmas 3.7.1 and 3.7.2) which detail the extension mechanism and show that it satisfies the required completeness property.

5.7.1 Lemma. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state such that each \mathcal{C}_i is int-total, and let $((A_i, \mathcal{K}_i) | i \leq r + k)$ be a complete total extension of \mathcal{S} with $k > 0$. Then we may find a $\mathcal{C} \in \text{COMP}(\{A_i | i \leq r\})$ such that

- (i) $A_{r+1} \notin \mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-$,
- (ii) $\{\neg A_{r+1}\} \cup \mathcal{C}$ has an int-total cyclic strong cover \mathcal{D}_{r+1} such that $\mathcal{D}_{r+1} \subseteq \mathcal{K}_{r+1}$, and
- (iii) for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has an int-total cyclic strong cover \mathcal{D}_i such that $\mathcal{D}_i \subseteq \mathcal{K}_i$.

5.7.2 Lemma. Suppose $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state where each \mathcal{C}_i is int-total, and $\mathcal{C} \in \text{COMP}(\{A_i | i \leq r\})$. Then we may find a predicate A_{r+1} such that

- (i) $A_{r+1} \notin \mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-$,
- (ii) $\{\neg A_{r+1}\} \cup \mathcal{C}$ has an int-total cyclic strong cover \mathcal{D}_{r+1} , and
- (iii) for each $i \leq r$, $\{A_{r+1}\} \cup \mathcal{C}_i$ has an int-total cyclic strong cover \mathcal{D}_i .

In Lemma 5.7.2, note that when $A_{r+1} \in \text{INT}(\mathcal{L})$, we must have that $A_{r+1} \in \mathcal{C}^- \cap \bigcap \mathcal{C}_i^+$ (since \mathcal{C} and each \mathcal{C}_i is int-total), in which case \mathcal{C} itself is an int-total cyclic strong cover of $\{\neg A_{r+1}\} \cup \mathcal{C}$, and for each $i \leq r$, \mathcal{C}_i is itself an int-total cyclic strong cover of $\{A_{r+1}\} \cup \mathcal{C}_i$.

In the case when $A_{r+1} \in \text{EXT}(\mathcal{L})$, condition (ii) can be characterised by the existence of a rule $\bigvee \mathcal{E} \in \text{EXT}(\mathcal{L})$ such that $A_{r+1} \in \mathcal{E}$, $(\mathcal{E} - \{A_{r+1}\}) \cap \mathcal{C}_{ext}^- = \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{E} - \{A_{r+1}\}) \vee \bigvee \mathcal{C}_{ext}^+$, in which case we may take $\mathcal{D}_{r+1} = \{\neg A_{r+1}\} \cup (\mathcal{E} - \{A_{r+1}\}) \cup \mathcal{C}$ (i.e., \mathcal{D}_{r+1} is a completion of $\{\neg A_{r+1}\} \cup \mathcal{C}$). Condition (iii) can be characterised by the condition $\text{EXT}(\text{DB}) \not\models A_{r+1} \vee \bigvee (\mathcal{C}_i^+)_{ext}$, in which case we may take $\mathcal{D}_i = \{A_{r+1}\} \cup \mathcal{C}_i$. (As mentioned earlier, the test against $\text{EXT}(\text{DB})$ simply requires us to check that no rule in $\text{EXT}(\text{DB})$ subsumes the given disjunction.)

Note that in both cases, $A_{r+1} \notin \{A_i | i \leq r\}$ since $A_{r+1} \notin \mathcal{C}^+ \supseteq \{A_i | i \leq r\}$.

5.7.3 Definition.

Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state, where each \mathcal{C}_i is int-total. An *immediate extension* \mathcal{S}^* of \mathcal{S} is computed as follows. Let $\mathcal{C} \in \text{COMP}(\{A_i | i \leq r\})$. (As in Section 3.7.3, if no such \mathcal{C} exists, then $\text{DB} \models \bigvee_{i \leq r} A_i$ and $\bigvee_{i \leq r} A_i$ is a minimal answer.) There are now two cases (at least one of which must apply), depending on whether we (try to) extend with a predicate in $\text{INT}(\mathcal{L})$ or $\text{EXT}(\mathcal{L})$.

- (a) Pick a predicate $A_{r+1} \in \text{INT}(\mathcal{L}) \cap \mathcal{C}^- \cap \bigcap_{i \leq r} \mathcal{C}_i^+$, and let \mathcal{S}^* be formed by extending \mathcal{S} with the pair (A_{r+1}, \mathcal{C}) .
- (b) Pick a predicate $A_{r+1} \in \text{EXT}(\mathcal{L}) - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ such that
 - (i) for each $i \leq r$, $\text{EXT}(\text{DB}) \not\models A_{r+1} \vee \bigvee (\mathcal{C}_i^+)_{ext}$, and
 - (ii) there is a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $A_{r+1} \in \mathcal{E}$, $(\mathcal{E} - \{A_{r+1}\}) \cap \mathcal{C}_{ext}^- = \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{E} - \{A_{r+1}\}) \vee \bigvee \mathcal{C}_{ext}^+$.

\mathcal{S}^* is then formed from \mathcal{S} by replacing each \mathcal{C}_i by $\{A_{r+1}\} \cup \mathcal{C}_i$, and then extending with the pair $(A_{r+1}, \{\neg A_{r+1}\} \cup (\mathcal{E} - \{A_{r+1}\}) \cup \mathcal{C})$.

Thus if we wish to extend using a predicate in $\text{INT}(\mathcal{L})$, we simply need to check that $\text{INT}(\mathcal{L}) \cap \mathcal{C}^- \cap \bigcap_{i \leq r} \mathcal{C}_i^+$ is non-empty, and (if this is the case) any predicate there-in can be employed. In the case when we wish to extend using a predicate in $\text{EXT}(\mathcal{L})$, the search for an appropriate predicate may be driven by a search of $\text{EXT}(\text{DB})$, thus reducing the search space. These comments address the issues identified in Section 3.7.3, Notes 3 and 4.

5.8 Example. Let $\text{EXT}(\mathcal{L}) = \{P_1, P_2, P_3, A_1, A_2, A_3\}$ with $\text{EXT}(\text{DB}) = \{A_3 \vee P_2, P_1 \vee P_2, A_1 \vee A_2 \vee P_3\}$, and let $\text{INT}(\mathcal{L}) = \{Q_1, Q_2, Q_3, Q_4\}$, where $\text{INT}(\text{DB})$ is as given in Appendix A. The set of minimal int-total weakly cyclic covers for $\text{INT}(\text{DB})$ is given in Appendix A, and by completing these in $\text{EXT}(\text{DB})$ we may conclude that DB has the following minimal int-total cyclic strong covers.

$$\begin{aligned}\mathcal{F}_1 &= \{Q_1, Q_4, \neg Q_2, Q_3, \neg P_1, A_2, P_2\} \\ \mathcal{F}_2 &= \{Q_1, Q_4, \neg Q_2, Q_3, \neg A_1, A_2, P_3\} \\ \mathcal{F}_3 &= \{Q_1, Q_4, \neg Q_2, Q_3, \neg A_3, A_2, P_2\} \\ \mathcal{F}_4 &= \{Q_1, Q_4, \neg Q_2, \neg Q_3, \neg P_1, \neg A_2, P_3, P_2, A_1\} \\ \mathcal{F}_5 &= \{Q_1, Q_4, Q_2, \neg Q_3, \neg A_2, A_1, P_3, A_3, P_1\} \\ \mathcal{F}_6 &= \{Q_1, Q_4, Q_2, \neg Q_3, \neg A_1, P_1, A_3, A_2, P_3\}, \text{ and} \\ \mathcal{F}_7 &= \{Q_1, Q_4, Q_2, Q_3, A_3, A_1, P_1, A_2\}\end{aligned}$$

(a) Suppose that we start with the int-total cyclic strong cover \mathcal{F}_5 of $\{\neg Q_3\}$. We then look for an int-total cyclic strong cover of $\{Q_3\}$, say \mathcal{F}_1 . If we wish to extend using a predicate in $\text{INT}(\mathcal{L})$, then we pick such a predicate in $\mathcal{F}_1^- \cap \mathcal{F}_5^+$, the only option being Q_2 .

$((Q_3, \mathcal{F}_5), (Q_2, \mathcal{F}_1))$ is not verified, since (for example) \mathcal{F}_7 is an int-total cyclic strong cover of $\{Q_3, Q_2\} \cup (\mathcal{F}_5^+ \cap \mathcal{F}_1^+) = \{Q_2, Q_3, Q_1, Q_4\}$. If $((Q_3, \mathcal{F}_5), (Q_2, \mathcal{F}_1))$ is verifiable, then (by Theorem 5.6.1) there is a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $(\mathcal{E} - \mathcal{F}_7) \cap (\mathcal{F}_5^- \cup \mathcal{F}_1^-) = \emptyset$, $\mathcal{E} \cap (\mathcal{F}_7 - \mathcal{F}_5) \neq \emptyset$, $\mathcal{E} \cap (\mathcal{F}_7 - \mathcal{F}_1) \neq \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{F}_5^+ \cup (\mathcal{E} - \mathcal{F}_7))_{ext}$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{F}_1^+ \cup (\mathcal{E} - \mathcal{F}_7))_{ext}$. $A_1 \vee A_2 \vee P_3$ is such a rule, and therefore $((Q_3, \mathcal{F}_5 \cup \{P_3\}), (Q_2, \mathcal{F}_1 \cup \{P_3\}))$ is an immediate extension of $((Q_3, \mathcal{F}_5), (Q_2, \mathcal{F}_1))$. It is easy to check that this cyclic state is verified, and may therefore be extended using the methods of Section 5.7.

(b) Suppose again that we start with $((Q_3, \mathcal{F}_5))$, and we employ the int-total cyclic strong cover \mathcal{F}_7 of $\{Q_3\}$. If we attempt to extend $((Q_3, \mathcal{F}_5))$ using a predicate in $\text{EXT}(\mathcal{L})$, then P_3 satisfies the conditions of Section 5.7.3, and $((Q_3, \mathcal{F}_5), (P_3, \mathcal{F}_7))$ is an immediate extension of $((Q_3, \mathcal{F}_5))$.

It is easy to check that $\{Q_3, P_3\} \cup (\mathcal{F}_5^+ \cap \mathcal{F}_7^+) = \{Q_3, P_3, Q_1, Q_2, Q_4, A_1, A_3, P_1\}$ has no int-total cyclic strong cover, and thus $((Q_3, \mathcal{F}_5), (P_3, \mathcal{F}_7))$ is verified.

Again, $\{Q_3, P_3\}$ has an int-total cyclic strong cover, for example \mathcal{F}_2 . If we again try to extend using an extensional predicate, we look for a predicate in $\text{EXT}(\mathcal{L}) - (\mathcal{F}_2^+ \cup \mathcal{F}_5^- \cup \mathcal{F}_7^-) = \{A_1, A_3, P_1, P_2\}$ satisfying the conditions of Section 5.7.3. A_1 satisfies these conditions, and $((Q_3, \mathcal{F}_5), (P_3, \mathcal{F}_7), (A_1, \mathcal{F}_2))$ is an immediate extension of $((Q_3, \mathcal{F}_5), (P_3, \mathcal{F}_7))$. It is easy to check that $\{Q_3, P_3, A_1\}$ has no int-total cyclic strong cover, and hence is a minimal answer.

5.9 Compilation for query processing.

Compilation has been previously studied for query processing under the minimal model [Ya02], perfect model [Jo99], disjunctive stable model [Jo99a], possible model [Jo02], and disjunctive well-founded [Jo03] semantics. Each of these approaches compiles a specific query (as opposed to using compilation to generate all (minimal) answers). For example in order to compile a query $\bigvee Q$ under the disjunctive stable model semantics, the approach presented in [Jo99a] computes int-total weakly cyclic covers \mathcal{C} of Q , and the run-time processing then attempts to find a completion \mathcal{D} of some such \mathcal{C} for which $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{D}_{ext}^+$. (In the terminology of the current section, $\text{DB} \models \bigvee Q$ iff $\text{COMP}(Q) = \emptyset$.) As an aside we also note that this approach provides a means of performing view updates and abductive reasoning [Jo97, Jo99, Jo99a].

Compilation of the GCWA in first order non-recursive positive databases is discussed in [He88]. In such databases, evaluation under the GCWA reduces to testing minimal answer membership, whose compilation is achieved by a combination of resolution and subsumption.

5.10 Extensions.

Throughout Section 5 we have assumed a partitioning of the database into two components, the first of which, $\text{EXT}(\text{DB})$, consists of simple disjunctions. Our crucial result, Theorem 5.1, was dependent on the resulting properties of $\text{EXT}(\text{DB})$, specifically that testing whether $\text{EXT}(\text{DB}) \models \bigvee Q_{ext}$ can be achieved without recourse to cyclic strong covers that are total in $\text{EXT}(\mathcal{L})$. But this property is (by Theorem 3.2) shared by all stratified databases, and we therefore note that the techniques of Section 5 could be extended to the case where the database is suitably partitioned into two components, one of which is stratified. This approach was discussed briefly in [Jo99a, Section 6] in the context of query processing.

§6. Query answers

As mentioned in the introduction, a *query* to a database is an expression of the form $?\bigvee \mathcal{H}$, where $\mathcal{H} \subseteq \mathcal{L}$, and an *answer* (to $?\bigvee \mathcal{H}$) is a set $\mathcal{A} \subseteq \mathcal{H}$ such that $\text{DB} \models \bigvee \mathcal{A}$.

Of course minimal answers to $?\bigvee \mathcal{H}$ can be computed from the minimal answers to $?\bigvee \mathcal{L}$ by simply discarding any answer that is not contained in \mathcal{H} . Such an approach however is clearly wasteful, and the following result shows that we can adapt our methods to directly compute (only) minimal answers to $?\bigvee \mathcal{H}$.

6.1 Theorem. A set $\{A_i | i \leq r\} \subseteq \mathcal{H}$ is contained in a minimal answer to the query

$? \bigvee \mathcal{H}$ iff for each $i \leq r$ we may find a disjunctive stable model M_i of DB such that $M_i \cap \{A_j | j \leq r\} = \{A_i\}$ and $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee \bigcap_{i \leq r} (\mathcal{H} - M_i)$.

So for example, for stratified databases we can then show that $\{A_i | i \leq r\} \subseteq \mathcal{H}$ is contained in a minimal answer to the query $? \bigvee \mathcal{H}$ iff for each $i \leq r$ we may find a cyclic strong cover \mathcal{C}_i of $\{\neg A_i\} \cup \{A_j | j \leq r, j \neq i\}$ such that $\text{DB} \models \bigvee_{i \leq r} A_i \vee \bigvee (\mathcal{H} \cap \bigcap_{i \leq r} \mathcal{C}_i^+)$.

As an aside, we recall ([Jo99, Section 5]) that when $r = 1$, Theorem 6.1 indicates that the relationship between minimal answer membership and disjunctive stable model membership (Theorem 1.5) does not carry over to queries of the form $? \bigvee \mathcal{H}$. Specifically Theorem 6.1 enables us to show that a predicate belongs to some minimal answer to the query $? \bigvee \mathcal{H}$ iff it is contained in some disjunctive stable model M of DB for which there is no disjunctive stable model M' such that $M' \cap \mathcal{H} \subset M \cap \mathcal{H}$.

Finally we note that our characterisations of minimal answers can easily be extended to the case when we consider queries and answers to be disjunctions of literals (as opposed to disjunctions of predicates). We have not however detailed further the extension of our results to this case.

§7. Other semantics

The perfect/disjunctive stable model semantics are of course not the only available semantics. In this section we discuss briefly the relationships between our concepts and methods, and other semantics.

7.1 Possible models.

Whilst minimal models (and semantics based upon these) tend to adopt the view that disjunction is interpreted exclusively, the possible model semantics [Sa94] is based upon inclusive disjunction. Possible models can be characterised by total supported strong covers [Jo02], and using this, the methods of the current paper can be adapted to the possible model semantics. For unstratified databases, possible models are ultimately defined in terms of stable models, whose possible non-existence gives rise to the same issues as found in the current paper. It is also worthy of note that for a certain class of stratified databases, the answers (and hence minimal answers) under the perfect model semantics coincide with those under the possible model semantics [Jo02, Corollary 7.2.7].

7.2 Stationary models.

In [Pr91, Pr91a], Przymusiński introduced the notion of a disjunctive stationary model, these being consistent sets of literals I such that I^+ is a minimal model of $\text{DB}|_g(\mathcal{L} - I^-)$ and $\mathcal{L} - I^-$ is a minimal model of $\text{DB}|_g I^+$. Total disjunctive stationary models clearly coincide with disjunctive stable models, and thus disjunctive stationary models can (also) be regarded as partial disjunctive stable models.

For non-disjunctive databases, stationary models are related to (supported) strong covers [Jo02, Section 4.2], but in the disjunctive case it is doubtful that any relationship exists with cyclic strong covers. For example if $\text{DB} = \{C \vee E, B \rightarrow D, B \rightarrow A, C \wedge \neg E \rightarrow A \vee B, A \wedge C \wedge \neg D \rightarrow B \vee E\}$, then $\{A, C, \neg E\}$ is a disjunctive stationary model, yet there is no cyclic strong cover containing $\neg A$.

Similarly the analogue of Theorem 2.12 does not hold for disjunctive stationary models. For example if $\text{DB} = \{C \vee E, D \rightarrow A, \neg D \rightarrow B, \neg B \rightarrow D, C \wedge \neg E \rightarrow A \vee B\}$, then again $I = \{A, C, \neg E\}$ is a disjunctive stationary model, and indeed $\bar{I} = \{\neg A, \neg C, E\}$ is a strong cover. $\text{DB}_{\bar{I}} = \{\neg D \rightarrow B, \neg B \rightarrow D\}$ (Definition 2.11) has disjunctive stable models $\{B\}$ and $\{D\}$, but $\{A, B, C\}$ is not a disjunctive stable model of DB .

Ultimately, the difference between disjunctive stationary models and cyclic strong covers lies in the motivation for their definition. Cyclicity insists that \mathcal{C}^- is contained in *any* model of $\text{DB}|_g(\mathcal{L} - \mathcal{C}^+) \wedge \neg \bigvee \mathcal{C}^+$, whereas stationarity insists that I^+ is one (of possibly many) minimal models of $\text{DB}|_g(\mathcal{L} - I^-)$.

7.3 The disjunctive well-founded semantics.

For unstratified databases we could alternatively employ the disjunctive well-founded semantics (DWFS) [Bra94]. DWFS can be constructed as the union of an increasing sequence $\emptyset = D_0 \subseteq D_1 \subseteq \dots$ [Bra98], where each $D_{\alpha+1}$ consists of a set $D_{\alpha+1}^+$ of disjunctions of predicates, and a set $D_{\alpha+1}^-$ of negative literals. $D_{\alpha+1}^+$ is constructed from D_{α}^- which in turn is constructed from D_{α}^+ . In [Jo01, Theorem 2.4] it is shown that $D_{\alpha+1}^+ = \{\bigvee \mathcal{P} : \text{DB}|_g(\mathcal{L} - \overline{D_{\alpha}^-}) \models \bigvee \mathcal{P}\}$, and thus the computation of minimal answers *in each* $D_{\alpha+1}^+$ can be achieved by applying the methods of Section 3 to the positive database $\text{DB}|_g(\mathcal{L} - \overline{D_{\alpha}^-})$ (using of course minimal rather than perfect models). However, the minimality of some answer in $D_{\alpha+1}^+$ does not necessarily imply its minimality in $D_{\alpha+2}^+$ or (hence) in DWFS as a whole.

We can overcome this problem by computing the sequence $D_0^- \subseteq D_1^- \subseteq \dots$ independently of the sets D_{α}^+ , since ([Jo01, Corollary 4.10]) $\neg Q \in D_{\alpha+1}^-$ iff for each quasi-cyclic tree \mathcal{T} for Q in DB , either $\text{DB}|_g(\mathcal{L} - \overline{D_{\alpha}^-}) \models \bigvee \mathcal{N}(\mathcal{T})$ or $\text{DB}|_g(\mathcal{L} - \mathcal{N}(\mathcal{T}) \cup \overline{D_{\alpha}^-}) \models \bigvee \mathcal{O}(\mathcal{T})$. Having computed $\text{DWFS}^- = \bigcup_{\alpha} D_{\alpha}^-$, we can then compute $\text{DWFS}^+ = \bigcup_{\alpha} D_{\alpha}^+ = \{\bigvee \mathcal{P} : \text{DB}|_g(\mathcal{L} - \overline{\text{DWFS}^-}) \models \bigvee \mathcal{P}\}$ [Jo01, Theorem 2.6], and in particular we can apply the techniques of Section 3 to the positive database $\text{DB}|_g(\mathcal{L} - \overline{\text{DWFS}^-})$

(again using minimal rather than perfect models) in order to compute minimal answers in $DWFS^+$. In this respect it is shown in [Jo01, Lemma 1.5] that only predicates in $\mathcal{L} - \overline{DWFS^-}$ can appear in a minimal answer in $DWFS^+$.

Despite the fact that $DWFS$ can be characterised using the variant notion of quasi-cyclic trees, we note that again there appears to be no direct relationship between $DWFS$ and cyclic strong covers. For example if $DB = \{A \vee D, B \vee E, C \vee F, \neg A \rightarrow B, \neg B \rightarrow C, \neg C \rightarrow A, \neg D \rightarrow E, \neg E \rightarrow F, \neg F \rightarrow D\}$, then there is no non-empty cyclic strong cover, yet $DWFS = \{A \vee D, B \vee E, C \vee F\}$.

As an aside we recall that in [Jo01, Section 2] we showed that if I is a disjunctive stationary model, then $I^+ \models DWFS^+$ and $I^- \supseteq \overline{DWFS^-}$, which in turn suggested the following question: “Given a minimal model M of $DWFS^+$, is it the case that $M \cup DWFS^-$ can always be extended to a disjunctive stationary model?” The above database, having no disjunctive stationary models, shows the answer to this question to be negative.

§8. The first order level

We have presented a new, and somewhat novel solution to the problem of computing minimal answers for propositional databases. In view of the relationship with the wider issues of computational complexity we consider these results to be of interest in themselves.

From the viewpoint of deductive databases however, “real” databases are of course first order (but function free). The rules in such databases represent the set of their ground instances (this defining issues such as logical inference), but for the sake of computational efficiency we require methods that apply directly to the first order level. In particular, grounding the entire database, and then subsequently applying a propositional query answering method would be considered inappropriate, since the resulting ground database would typically be very large. However first order methods are usually developed initially for the propositional level, and then lifted to the first order level, hence our results presented in Sections 3-5 should be considered a first step.

We commence in Section 8.1 with some terminology, and in Section 8.2 we re-state some basic facts from [Jo96, Jo98, Jo00] concerning cyclic and partial cyclic trees (and their construction) at the first order level. In Section 8.3 we consider how our minimal answer generating method applies to first order databases, and show that it can be supported by a lifting of the approach to computing cyclic strong covers presented in Sections 2.7-2.8.

Clearly processing is more costly at the first order level, thus suggesting that

there is an even greater need here for the use of pre-processing (e.g., compilation). In Section 8.4 we therefore present a first order compilation process. This entails the (top-down) construction of a first order “compilation tree” whose instantiated branches will encode int-total weakly cyclic covers in the grounded database. We also briefly detail how the compilation tree is employed in subsequent (post-compilation) minimal answer generation.

One of the difficulties with top-down first order processing is termination, and in Section 8.5 we present a set of database constraints which significantly reduce the cost of guaranteeing the termination of the compilation process. In Section 8.6 we illustrate that our lifted method overcomes a problem encountered with the top-down first order query processing method presented in [Jo98, Jo98a], and finally in Section 8.7 we briefly discuss issues surrounding the full first order level (i.e., with function symbols).

8.1 Terminology.

Throughout Section 8 we assume that \mathcal{L} is a first order language containing a finite set of predicates $\{P_i | i \leq n\}$, a finite set of constants $\{a_i | i \leq r\}$, and of course a countable set of variables, to be denoted by x, y, z, u, v, w, \dots . A *term* is a constant or a variable, and an *atom* is a formula of the form $P(t_1, t_2, \dots, t_k)$ or $\neg P(t_1, t_2, \dots, t_k)$, where each t_i is a term and P is a predicate in \mathcal{L} of arity k . We will employ $\mathbf{t}_1, \mathbf{t}_2, \dots$ to denote sequences of terms, and $\mathbf{a}, \mathbf{b}, \dots$ to denote sequences of constants.

DB will denote a first order database, and the set of ground instances of rules in DB is denoted by $gr(\text{DB})$. As in Section 5 we assume that the predicates in \mathcal{L} are either extensional or intensional (and we will use $\text{EXT}(\mathcal{L})$ etc. to denote the set of positive atoms whose predicate is extensional), and that

- (i) if $\text{conseq}(\mathbf{R}) \cap \text{EXT}(\mathcal{L}) \neq \emptyset$, then $\text{conseq}(\mathbf{R}) \subseteq \text{EXT}(\mathcal{L})$ and $\text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}) = \emptyset$,
- (ii) if $\text{conseq}(\mathbf{R}) \cap \text{INT}(\mathcal{L}) \neq \emptyset$, then $\text{conseq}(\mathbf{R}) \subseteq \text{INT}(\mathcal{L})$ and $\text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}) \neq \emptyset$ (thus $\text{antec}(\mathbf{R}) \neq \emptyset$, cf., Section 1.1).

We make the usual assumption that rules are *range restricted*, in the sense that every variable appearing in a rule \mathbf{R} appears in $\text{antec}(\mathbf{R})$. In particular, rules in $\text{EXT}(\text{DB})$ are disjunctions of ground extensional positive atoms.

Our compilation process will also employ constraints, and for this purpose we employ a further distinguished predicate $=$, not appearing in \mathcal{L} (or hence in DB), using \neq to represent its negation. We then define an *eq-constraint* to be a formula of the form $\forall x_1 \forall x_2 \dots \forall x_r \Phi$ ($r \geq 0$), where Φ is constructed from atoms of the form $t_1 = t_2$ (where each t_i is a term) using \neg , \wedge and \vee , and x_1, x_2, \dots, x_r are variables appearing in Φ . If η is a ground instantiation of the free variables in $\forall x_1 \forall x_2 \dots \forall x_r \Phi$ (i.e., variables in Φ other than x_1, x_2, \dots, x_r), then η *satisfies* $\forall x_1 \forall x_2 \dots \forall x_r \Phi$ iff $\forall x_1 \forall x_2 \dots \forall x_r \Phi \eta$ is true

(in $\{a = a \mid a \in \mathcal{L}\} \cup \{a \neq b \mid a, b \in \mathcal{L}, a \neq b\}$). As in earlier sections, we also employ an additional distinguished predicate **FALSE**.

If the rule **R** has the form $P(w, y) \wedge R(w) \rightarrow Q(w)$, with the variable y appearing in the body of **R**, but not in its consequent, then **R** may be rewritten as $\forall w((\exists y P(w, y)) \wedge R(w) \rightarrow Q(w))$. Thus y can be regarded as existentially quantified, and we will henceforth refer to such variables as \exists - *variables*.

8.2 Cyclic trees.

The definition and (top-down) construction of cyclic trees for first order (function free) databases is discussed at length in [Jo96, Jo98, Jo98a, Jo00], and in particular we note [Jo96, Theorem 6.4.7] that even for first order databases, such trees are still ground (i.e., all labels on the tree, and in particular $Pred(\mathcal{T}) \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$, are ground). This grounding occurs naturally as a result of the fact that rules within the tree are linked to the cycle above (cf., Definition 2.3(iii)) by a unifying instantiation, and in particular the fact that every branch terminates with a (ground) disjunction from the extensional database, some atom of which must have been unified with the extensional atom lying just above the leaf.

Partial cyclic trees on the other hand do not have these extensional rule nodes at the leaves (and do not therefore have the instantiation that results from appending these leaves), and as a result, partial cyclic trees are not necessarily ground. We can still show however that if \mathcal{T} is a partial cyclic tree, then every variable appearing in $Pred(\mathcal{T}) \cup \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$ appears in an extensional predicate leaf node (i.e., in $\mathcal{S}(\mathcal{T})_{ext}^-$).

The construction of partial cyclic trees at the first order level may be achieved using *complete pairs*, whose definition and construction is discussed in [Jo00]. When employed to construct partial cyclic trees, a *complete pair* for a first order positive atom $Q(\mathbf{t})$ consists of a first order partial cyclic tree \mathcal{T} for some (possibly first order) instance of $Q(\mathbf{t})$, and a set of constraints \mathcal{I} (satisfying certain conditions) of the form $\mathbf{t}_1 \neq \mathbf{t}_2$, where $\mathbf{t}_1, \mathbf{t}_2$ are sequences of terms from \mathcal{T} . We then have the following theorem, which is taken from [Jo00].

8.2.1 Theorem [Jo00].

(a) If $Q(\mathbf{a})$ is a ground instance of $Q(\mathbf{t})$, then \mathcal{T}^* is a partial cyclic tree for $Q(\mathbf{a})$ in $gr(INT(DB))$ iff there is a complete pair $(\mathcal{T}, \mathcal{I})$ for $Q(\mathbf{t})$ and a ground instantiation θ of \mathcal{T} such that θ satisfies $\bigwedge \mathcal{I}$ and $\mathcal{T}\theta = \mathcal{T}^*$.

(b) If $(\mathcal{T}, \mathcal{I})$ is a complete pair and θ is a ground instantiation of \mathcal{T} which satisfies $\bigwedge \mathcal{I}$ then $Pred(\mathcal{T}\theta) = Pred(\mathcal{T})\theta$, $\mathcal{N}(\mathcal{T}\theta) = \mathcal{N}(\mathcal{T})\theta$, $\mathcal{O}(\mathcal{T}\theta) = \mathcal{O}(\mathcal{T})\theta$ and $\mathcal{S}(\mathcal{T}\theta) = \mathcal{S}(\mathcal{T})\theta$.

Notice in part (a) that if $root(\mathcal{T}) = Q(\mathbf{t}_0)$, then $\mathbf{t}_0\theta = a$. In Section 5 we showed that cyclic strong covers can be computed by forming the completions of weakly cyclic covers. The following definition provides the analogous concept for complete pairs.

8.2.2 Definition. Let $(\mathcal{T}, \mathcal{I})$ be a complete pair, and θ be a ground instantiation of $\mathcal{S}(\mathcal{T})_{ext}^-$ such that

- (i) θ satisfies $\bigwedge \mathcal{I}$,
- (ii) for each $P(\mathbf{t}) \in \mathcal{S}(\mathcal{T})_{ext}^-$ there is a rule $f(P(\mathbf{t})) = \bigvee \mathcal{E}_{P(\mathbf{t})} \in \text{EXT}(\text{DB})$ such that $P(\mathbf{t}\theta) \in \mathcal{E}_{P(\mathbf{t})}$ and $\mathcal{S}(\mathcal{T})\theta \cup \bigcup \{\mathcal{E}_{P(\mathbf{t})} - \{P(\mathbf{t}\theta)\} \mid P(\mathbf{t}) \in \mathcal{S}(\mathcal{T})_{ext}^-\}$ is consistent.

Then the set $\mathcal{S}(\mathcal{T})\theta \cup \bigcup \{\mathcal{E}_{P(\mathbf{t})} - \{P(\mathbf{t}\theta)\} \mid P(\mathbf{t}) \in \mathcal{S}(\mathcal{T})_{ext}^-\}$ is a *completion* of $(\mathcal{T}, \mathcal{I})$.

By the above remark, a ground instantiation of $\mathcal{S}(\mathcal{T})_{ext}^-$ will (conveniently) also have the effect of grounding $\mathcal{S}(\mathcal{T})$. Finally we note that having constructed a complete pair $(\mathcal{T}, \mathcal{I})$ we do not then need to store the entire tree - future processing simply requires access to the triple $(root(\mathcal{T}), \mathcal{S}(\mathcal{T}), \mathcal{I})$.

8.3 Computing minimal answers.

A minimal answer in DB is simply a set of ground positive atoms that is a minimal answer in $gr(\text{DB})$. Moreover a set of ground positive atoms is witnessed as being a partial minimal answer by the existence of a verified cyclic state, in which each of the cyclic strong covers is int-total and *ground*.

Thus in order to extend our minimal answer generating method to first order databases, we need to be able to generate the int-total cyclic strong covers (in $gr(\text{DB})$) needed for the construction of immediate extensions: In Section 5.6.2 we need to construct an int-total cyclic strong cover of $\{A_i \mid i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+$, and in Section 5.7.3 we need to compute an int-total cyclic strong cover of $\{A_i \mid i \leq r\}$. The additional processing required to compute the immediate extension (see Sections 5.6.2 and 5.7.3) is then carried out trivially at the ground level (since $\text{EXT}(\text{DB})$ is ground).

In this sub-section we show that such int-total cyclic strong covers can be computed by amending the approach presented in Sections 2.7-2.8. For this purpose we assume (without loss of generality) that for each rule $\mathbf{R} \in \text{INT}(\text{DB})$ there is a distinguished atom $P^{\mathbf{R}}$ in $\text{antec}(\mathbf{R})$ which is the only atom on the body of \mathbf{R} which may contain \exists -variables.

If K is an atom, then let $gr(K)$ denote the set of ground instances of K in \mathcal{L} .

8.3.1 Definition. Let \mathcal{Q} be a set of positive ground atoms, then a strong cover \mathcal{C} of \mathcal{Q} in $gr(\text{DB})$ is a *constructible extension* of \mathcal{Q} in DB iff we can find a sequence $\mathcal{Q} = \mathcal{Q}_0 \subseteq \mathcal{Q}_1 \subseteq \dots \subseteq \mathcal{Q}_r = \mathcal{C}$ such that for each $1 \leq i \leq r$ there is a rule $\mathbf{R}_i \in \text{DB}$ and

a (ground) instantiation θ_i with $\text{domain}(\theta_i) = \{x \mid x \text{ appears in } \text{conseq}(\mathbf{R}_i)\}$ such that

- (i) $\text{conseq}(\mathbf{R}_i\theta_i) \subseteq \mathcal{Q}_{i-1}$,
- (ii) \mathcal{Q}_{i-1} is disjoint from $(\text{antec}(\mathbf{R}_i) - \{P^{\mathbf{R}_i}\})\theta_i \cup \overline{\mathcal{N}(\mathbf{R}_i\theta_i)}$,
- (iii) $gr(P^{\mathbf{R}_i}\theta_i) \not\subseteq \mathcal{Q}_{i-1}$, and
- (iv) $\mathcal{Q}_i = \mathcal{Q}_{i-1} \cup \{A_i\theta_i\}$, for some $A_i \in \text{antec}(\mathbf{R}_i) - \{P^{\mathbf{R}_i}\}$, or
 $\mathcal{Q}_i = \mathcal{Q}_{i-1} \cup gr(P^{\mathbf{R}_i}\theta_i)$, or
 $\mathcal{Q}_i = \mathcal{Q}_{i-1} \cup \mathcal{S}(\mathcal{T})$, where \mathcal{T} is a cyclic tree for some $A_i\theta_i \in \mathcal{N}(\mathbf{R}_i\theta_i)$ in DB.

Note that each \mathcal{Q}_i is ground, consistent (since \mathcal{C} is), and cyclic (by the third clause of condition (iv)). The groundedness of θ_i results naturally from a unification of $\text{conseq}(\mathbf{R}_i)$ with a subset of \mathcal{Q}_{i-1} (cf., condition (i)) as opposed to a mass grounding of DB. The finiteness of the Herbrand base clearly implies that there is no infinite sequence $\mathcal{Q} = \mathcal{Q}_0 \subseteq \mathcal{Q}_1 \subseteq \dots \subseteq \mathcal{Q}_k \subseteq \dots$ satisfying the above conditions.

Note that if we expand \mathcal{Q}_{i-1} using clause 2 of condition (iv), then we take every instantiation of the \exists -variables (in $P^{\mathbf{R}_i}\theta_i$). In top-down query processing we would certainly consider the mass generation of ground instances of a non-ground atom to be undesirable, but here our aim is specifically to compute (ground) cyclic strong covers, and in this context our handling of $P^{\mathbf{R}_i}\theta_i$ is in fact an efficient one, whose correctness is proven in the following theorem.

8.3.2 Theorem. Let \mathcal{Q} be a set of positive ground atoms.

- (a) If \mathcal{C} is a constructible extension of \mathcal{Q} then \mathcal{C} is a cyclic strong cover of \mathcal{Q} in $gr(\text{DB})$.
- (b) If \mathcal{D} is a cyclic strong cover of \mathcal{Q} in $gr(\text{DB})$, then there is a constructible extension \mathcal{C} of \mathcal{Q} in DB such that $\mathcal{C} \subseteq \mathcal{D}$.

Proof (a). \mathcal{C} is by definition a strong cover, and is cyclic by the above remark.

(b). Suppose that $\mathcal{Q} = \mathcal{Q}_0 \subseteq \mathcal{Q}_1 \subseteq \dots \subseteq \mathcal{Q}_s$ is a sequence satisfying the conditions of Definition 8.3.1 with $\mathcal{Q}_s \subseteq \mathcal{D}$. If \mathcal{Q}_s is not a strong cover in $gr(\text{DB})$ then we may find a rule $\mathbf{R} \in \text{DB}$ and a ground instantiation $\mathbf{R}\phi$ of \mathbf{R} such that $\text{conseq}(\mathbf{R}\phi) \subseteq \mathcal{Q}_s$ and $\mathcal{Q}_s \cap [\text{antec}(\mathbf{R}\phi) \cup \overline{\mathcal{N}(\mathbf{R}\phi)}] = \emptyset$. Let θ be the restriction of ϕ to the variables in $\text{conseq}(\mathbf{R})$, then $\mathbf{R}\theta$ satisfies the conditions of Definition 8.3.1.

If there exists $K \in \mathcal{D} \cap (\text{antec}(\mathbf{R}) - \{P^{\mathbf{R}}\})\theta$, then we may set $\mathcal{Q}_{s+1} = \mathcal{Q}_s \cup \{K\}$. If there exists a $K \in \mathcal{D}^- \cap \mathcal{N}(\mathbf{R}\theta)$, then (since \mathcal{D} is cyclic) we may set $\mathcal{Q}_{s+1} = \mathcal{Q}_s \cup \mathcal{S}(\mathcal{T})$ for some cyclic tree \mathcal{T} for K in DB such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{D}$.

If no such K exists, then (since \mathcal{D} is a strong cover) we must have that $gr(P^{\mathbf{R}}\theta) \subseteq \mathcal{D}$, and we may therefore set $\mathcal{Q}_{s+1} = \mathcal{Q}_s \cup gr(P^{\mathbf{R}}\theta)$.

As noted above, the sequence $(\mathcal{Q}_i \mid i = 0, 1, 2, \dots)$ must be finite, and this then proves the theorem. ■

As noted in Section 2 we can of course force all constructible extensions to be int-total by the implicit addition of $P(\mathbf{a}) \wedge \neg P(\mathbf{a}) \rightarrow \text{FALSE}$, for each ground intensional atom $P(\mathbf{a})$.

8.3.3 Factorisation.

As in Section 2 we may represent the construction of constructible extensions as the branches through a tree. In [Jo98, Jo98a] we saw that extended deduction trees are amenable to various forms of pruning, and in this section we illustrate the particular applicability of one such pruning mechanism to the current scenario.

Suppose for example that we encountered the subtree depicted in Figure 8.3(i), then *factorisation* (e.g., [Bi81, Jo98]) dictates that we can discard the child of $rn_{\mathbf{S}_\eta}$ labelled with W , since any cyclic strong cover generated by extending this branch will contain a cyclic strong cover resulting from the extension of the right-hand child of $rn_{\mathbf{R}\theta}$ (labelled with W).

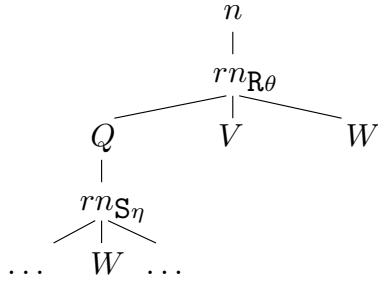


Figure 8.3(i)

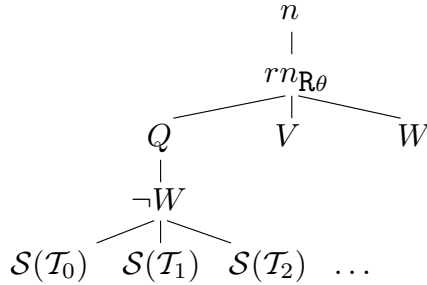


Figure 8.3(ii)

An interesting strengthening of factorisation is possible when W is intensional (and we intend to compute int-total sets). Since we are not intending to extend any branch through Q with W , we must eventually add $\neg W$ to (all) such branches. We can save on the duplication of doing so, by doing so immediately. This then yields the subtree depicted in Figure 8.3(ii).

8.3.4 Pre-processing. The above computation of int-total cyclic strong covers can obviously be supported by pre-computing the complete pairs for $P(x_1, x_2, \dots, x_k)$ (for each intensional predicate P). The run-time computation of cyclic strong covers is then required to form their completion in order to expand negative literals (cf., clause 3 of Definition 8.3.1(iv)). This is of course a limited form of pre-processing; full scale compilation is considered in the following section.

8.4 Compilation.

In this section we present the (top-down) construction of a first order “compilation” tree, whose (instantiated) branches encode int-total weakly cyclic covers in $gr(DB)$. In Section 8.5 we indicate briefly how the compilation process can be made more efficient.

We have already noted that the construction of cyclic strong covers can be achieved by traversing an extended deduction tree, and it is not surprising therefore that our compilation tree has a similar structure. As a result (cf., Example 2.7), we can immediately identify the following types of “goal” nodes:

- (i) *set* nodes, which are labelled with a set of atoms,
- (ii) *positive atom* nodes, which are labelled with a positive atom, and
- (iii) *negative atom* nodes, which are labelled with a negative atom.

We will of course need rule nodes (in this case labelled with instances of rules in $INT(DB)$), and in order to ensure that our tree construction generates int-total sets, we implicitly add the splitting rules $P(x_1, x_2, \dots, x_k) \wedge \neg P(x_1, x_2, \dots, x_k) \rightarrow \text{FALSE}$ (for each intensional predicate P) to $INT(DB)$. (The root node of our compilation tree will be labelled with **FALSE**.) We will also need *constraint* nodes, which will be labelled with an eq-constraint. Given any node N in the compilation tree, we let \mathcal{B}_N denote the (partial) branch through the compilation tree, from the root down to (and including) N . Let $atom(N)$ denote the set of atoms from \mathcal{L} appearing on any goal node on \mathcal{B}_N .

8.4.1 Expanding negative atom nodes

By Theorem 8.2.1 and Proposition 5.3(d), negative atom nodes $\neg Q(\mathbf{t})$ call for the generation of complete pairs. Thus for each complete pair $(\mathcal{T}, \mathcal{I})$ for $Q(\mathbf{t})$, $\neg Q(\mathbf{t})$ has a child node labelled with the eq-constraint $(\mathbf{t}_0 = \mathbf{t}) \wedge \bigwedge \mathcal{I}$ (where $root(\mathcal{T}) = Q(\mathbf{t}_0)$), which in turn has a single (set node) child of the form $\mathcal{S}(\mathcal{T})$.

8.4.2 Expanding set and positive atom nodes

Let n be a set or positive atom node. As was seen in Example 2.7, our intention is to attack n with a rule from $INT(DB)$. Before we can do so however we need to split the branch for the purposes of unification and ancestor-pruning, as is described in the following two sections.

Unification splitting

Suppose that $atom(n) = \{Q(x, y), R(y, x)\}$, and that \mathbf{R} is a rule of the form $P(u, v) \rightarrow Q(u, v) \vee R(v, u)$. We can apply \mathbf{R} to n via the substitution $\lambda = \{u \rightarrow x, v \rightarrow y\}$, yielding the sub-tree depicted in Figure 8.4(i).

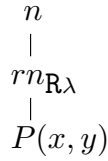


Figure 8.4(i)

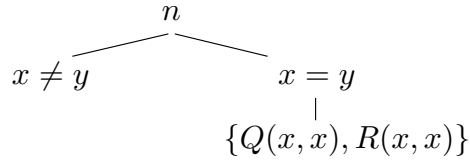


Figure 8.4(ii)

Suppose instead however that \mathbf{R} had the form $P(u, u) \rightarrow Q(u, u) \vee R(u, u)$. If we were to apply \mathbf{R} directly to $atom(n)$ it would result in the unification of x and y , which in turn might result in a weakly cyclic cover in $gr(\text{DB})$ being over-looked. To prevent this we split the branch using the constraint that defines the required instantiation of $atom(n)$, in this case $x = y$ (Figure 8.4(ii)). We (implicitly) append the set of instantiated atoms, and \mathbf{R} can then be applied directly to the right-hand branch via the substitution $\{u \rightarrow x\}$. We will need to distinguish between the two branches of the split, thus we refer to the child $x = y$ (of the split) as the *positive* child.

Thus the application of a rule $\mathbf{R} \in \text{INT}(\text{DB})$ to n requires that we find an instance $\mathbf{R}\theta$ for which $\text{conseq}(\mathbf{R}\theta) \subseteq atom(n)$. No instantiation of \mathcal{B}_n results from the application of $\mathbf{R}\theta$: if unification is required, it is achieved by the prior use of a unification split as illustrated above. (If unification is not required, then of course forming a unification split is pointless, since the positive branch would be labelled with the empty (null) eq-constraint, e.g., $x = x$, and the negative branch would be labelled with an unsatisfiable eq-constraint, e.g., $x \neq x$.)

For clarity, we will insist that no variable appearing in the compilation tree appears in any rule in DB . Thus we will label rule nodes with a triple of the form $(\mathbf{R}, \theta, \delta)$, where θ maps every variable appearing in $\text{conseq}(\mathbf{R})$ onto a term in \mathcal{B}_n (where n is the parent node), and δ renames the \exists -variables in \mathbf{R} to be disjoint from the variables already appearing in the tree or in DB . Clearly $(\mathbf{R}, \theta, \delta)$ has a child node of the form $K\theta\delta$ for every atom K in the body of \mathbf{R} . When δ is empty we write $-$.

Ancestor-pruning splitting

At the propositional level, we can force termination of an extended deduction tree construction simply by employing *ancestor pruning*, i.e., each new atom (node) that is added to a branch as a child of a rule node must not already appear on the branch. At the first order level we mimic this approach as follows. Suppose that we have identified a rule \mathbf{R} that we wish to apply to n , and have previously applied unification splitting (if necessary), so that $\text{conseq}(\mathbf{R}\theta) \subseteq atom(n)$.

For example suppose that $atom(n) = \{P(w), P(z), Q(x)\}$, and $\mathbf{R}\theta = P(x) \rightarrow Q(x)$.

database rule application, i.e., splitting is *only* applied as a preparatory step for the subsequent application of a database rule. A negative child of a split is extended in the same way as a set/positive atom node;

- (ii) If N is the positive child of a split, then \mathcal{B}_N is consistent;
- (iii) If \mathcal{B}_N is inconsistent, then N is a leaf;
- (iv) If $\forall u \neg\Phi$ is the negative child of an ancestor-pruning split, and $(\mathbf{R}, \theta, \delta)$ is the child of the corresponding positive sibling, then the subtree below $\forall u \neg\Phi$ is prohibited from containing a rule node of the form $(\mathbf{R}, \theta, \gamma)$;
- (v) The tree is extended maximally subject to the above constraints.

The restrictions placed on the compilation tree are shown in Theorem 8.4.4(a) below to guarantee its finiteness. Parts (b) and (c) show that the satisfying instantiations of branches through the tree capture int-total weakly cyclic covers in $\text{INT}(gr(\text{DB}))$.

8.4.4 Theorem.

- (a) The compilation tree is finite.
- (b) If \mathcal{B} is a branch through the compilation tree, and η is a satisfying instantiation of \mathcal{B} , then $atom(\mathcal{B})\eta$ is an int-total weakly cyclic cover in $\text{INT}(gr(\text{DB}))$.
- (c) If \mathcal{C} is an int-total weakly cyclic cover in $\text{INT}(gr(\text{DB}))$, then there is a branch \mathcal{B} through the compilation tree, and a satisfying instantiation η of \mathcal{B} such that $atom(\mathcal{B})\eta \subseteq \mathcal{C}$.

Proof (a). Suppose that \mathcal{B} is an infinite branch. For every node N on \mathcal{B} , \mathcal{B}_N must be consistent (else it would not be extended). If η is a satisfying instantiation of \mathcal{B}_N , then for each rule node $(\mathbf{R}, \theta, \delta)$ on \mathcal{B}_N , the child of $(\mathbf{R}, \theta, \delta)$ on \mathcal{B} introduces a new atom into $atom(\mathcal{B})\eta$. Since the Herbrand base is finite, \mathcal{B} must pass through only a finite number of rules nodes.

But then we may find a node N^* on \mathcal{B} such that below N^* , \mathcal{B} passes only through segments of the form

- (i) the negative branch of a unification split, or
- (ii) the positive branch of a unification split, followed by the negative branch of the following ancestor-pruning split⁷.

If N lies below N^* on \mathcal{B} , let $\mu_1(N)$ denote the number of ground instantiations of \mathcal{B}_N that satisfy \mathcal{B}_N . Let $\mu_2(N)$ denote the number of pairs of the form (\mathbf{R}, θ) , where $\mathbf{R} \in \text{INT}(\text{DB})$ and θ is a mapping of the variables in $\text{conseq}(\mathbf{R})$ onto either a constant

⁷The case where there is no unification split can be viewed as a unification split in which the negative child is unsatisfiable, thus this is also covered by (ii) above.

or a variable in $atom(N)$ such that triples of the form $(\mathbf{R}, \theta, \gamma)$ are not prohibited (by Definition 8.4.3(b)(iv)) from the subtree below \mathcal{B}_N . Let $\mu(N) = \mu_1(N) + \mu_2(N)$.

Suppose now that we traverse a segment of the form given in option (i): N_0 is extended via a unification split, and N_1 is the negative child of this split. No new variables are introduced at N_1 , and the corresponding positive branch is required to have at least one satisfying instantiation, thus $\mu_1(N_1) < \mu_1(N_0)$. Again because no new variables are introduced into $atom(N_1)$, $\mu_2(N_1) \leq \mu_2(N_0)$. Thus $\mu(N_1) < \mu(N_0)$.

Suppose now that we traverse a segment of the form given in option (ii): N_0 is extended via a unification split and N_1 is the positive child of this split. N_1 is then extended with an ancestor pruning split, and N_2 is the negative child of this split. Since no new free variables are being introduced, $\mu_1(N_2) \leq \mu_1(N_1) \leq \mu_1(N_0)$, and since no new variables are introduced into $atom(N_2)$ we again have that $\mu_2(N_2) \leq \mu_2(N_1) \leq \mu_2(N_0)$. If $(\mathbf{R}, \theta, \delta)$ is the child of N_2 's positive sibling, then (\mathbf{R}, θ) could not have been prohibited below N_0 or N_1 , but is prohibited below N_2 . Thus $\mu_2(N_2) < \mu_2(N_1)$ and hence $\mu(N_2) < \mu(N_1) \leq \mu(N_0)$.

Thus \mathcal{B} cannot be infinite.

(b). Suppose that $\mathbf{R}\sigma$ is a ground instance of $\mathbf{R} \in \text{INT}(\text{DB})$ with $\text{conseq}(\mathbf{R}\sigma) \subseteq atom(\mathcal{B})\eta$ and $(\text{antec}(\mathbf{R}\sigma) \cup \overline{\mathcal{N}(\mathbf{R}\sigma)}) \cap atom(\mathcal{B})\eta = \emptyset$. But then we can extend \mathcal{B} with a rule node of the form $(\mathbf{R}, \theta, \delta)$ (preceeded of course by the appropriate splits), thus contradicting \mathcal{B} 's maximality.

The same argument can be applied to any of the splitting rules $(P(\mathbf{x}) \wedge \neg P(\mathbf{x}) \rightarrow \text{FALSE})$, thus showing that $atom(\mathcal{B})\eta$ is int-total.

Suppose that $\neg Q(\mathbf{t})$ is a negative atom node on \mathcal{B} . Let $(\mathcal{T}, \mathcal{I})$ be the complete pair for which $(\mathbf{t}_0 = \mathbf{t}) \wedge \bigwedge \mathcal{I}$ (where $root(\mathcal{T}) = Q(\mathbf{t}_0)$) and $\mathcal{S}(\mathcal{T})$ lie on \mathcal{B} below $\neg Q(\mathbf{t})$. Since η satisfies $(\mathbf{t}_0 = \mathbf{t}) \wedge \bigwedge \mathcal{I}$ and $\mathcal{S}(\mathcal{T})\eta \subseteq atom(\mathcal{B})\eta$, Theorem 8.2.1 then dictates that $atom(\mathcal{B})\eta$ is weakly cyclic (cf., Definition 5.2(a)).

(c). We traverse a path through the compilation tree, constructing a satisfying instantiation with the required properties as follows. Suppose that n is a node and η is a satisfying instantiation of \mathcal{B}_n such that $atom(n)\eta \subseteq \mathcal{C}$.

Case 1: n is a set/positive atom node or the negative child of a split.

If necessary n is extended via a unification split, in which case we may traverse to the child m of the split for which \mathcal{B}_m is satisfied by η .

If m is the positive child of the split, we then encounter an ancestor-pruning split of the form $\forall \mathbf{u} \neg \Phi$ and Φ . If $\forall \mathbf{u} \neg \Phi \eta$ is true, we of course traverse to the negative child $\forall \mathbf{u} \neg \Phi$ of the split.

If on the other hand $\exists \mathbf{u} \Phi \eta$ is true, then we traverse to the positive child Φ of the split, extending η to a substitution which satisfies Φ . On encountering the subsequent rule node $(\mathbf{R}, \theta, \delta)$, η is then already defined on all variables in $\mathbf{R}\theta\delta$, and we can of course

use the fact that \mathcal{C} is a strong cover in INT(DB) to traverse to a child of $(\mathbf{R}, \theta, \delta)$ labelled with an atom K (from the body of $\mathbf{R}\theta\delta$) for which $K\eta \in \mathcal{C}$.

Case 2: n is a negative atom node, say labelled with $\neg Q(\mathbf{t})$.

By Theorem 8.2.1 and Proposition 5.3(d) we may pick a complete pair $(\mathcal{T}, \mathcal{I})$ for $Q(\mathbf{t})$ and a ground instantiation θ of \mathcal{T} such that $Q(\mathbf{t})\eta = \text{root}(\mathcal{T})\theta$, θ satisfies $\bigwedge \mathcal{I}$ and $\mathcal{S}(\mathcal{T})\theta \subseteq \mathcal{C}$. We may thus traverse to $\mathcal{S}(\mathcal{T})$ (cf., Section 8.4.1), and extend η via θ . This then completes the construction. ■

8.4.5 Minimal answer generation using the compilation tree.

As noted at the beginning of Section 8.3, the computation of minimal answers requires us to construct int-total cyclic strong covers (in $gr(\text{DB})$) of a set of ground (positive) atoms \mathcal{P} . To achieve this we traverse a branch through the compilation tree (as in the proof of Theorem 8.4.4(c)) at each point maintaining (i) a satisfying instantiation η of the current partial branch \mathcal{B}_n , and (ii) an additional (variable) set of ground atoms \mathcal{V} such that $\text{atom}(\mathcal{B}_n)\eta \subseteq \mathcal{V}$. Upon encountering a leaf, \mathcal{V} will yield an int-total cyclic strong cover of \mathcal{P} . We initially set $\mathcal{V} = \mathcal{P}$.

Case 1: n is set/positive atom node, or the negative child of a split.

We traverse the splits immediately below n in the same manner as in the proof of Theorem 8.4.4(c). If we traverse the positive branch of the ancestor pruning split to the resulting rule node $(\mathbf{R}, \theta, \delta)$, then we may traverse to a child node of $(\mathbf{R}, \theta, \delta)$ labelled with K (where K is in the body of $\mathbf{R}\theta\delta$) provided $\neg K\eta \notin \mathcal{V}$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{V} \cup \{K\eta\})_{ext}^+$, in which case we clearly update $\mathcal{V} \leftarrow \mathcal{V} \cup \{K\eta\}$.

Case 2: n is a negative atom node labelled with $\neg Q(\mathbf{t})$.

As in the proof of Section 8.4.4(c), we wish to traverse to a (grand)child $\mathcal{S}(\mathcal{T})$ of $\neg Q(\mathbf{t})$ whilst extending the current instantiation to the new variables in $\mathcal{S}(\mathcal{T})$. However our aim now is to compute cyclic strong covers (as opposed to weakly cyclic covers), thus we also need to employ rules in $\text{EXT}(\text{DB})$ to form the completion.

We thus pick a complete pair $(\mathcal{T}, \mathcal{I})$ (for $Q(\mathbf{t})$), and a completion $\mathcal{B} = \mathcal{S}(\mathcal{T})\theta \cup \bigcup \{\mathcal{E}_{P(\mathbf{t})} - \{P(\mathbf{t}\theta)\} \mid P(\mathbf{t}) \in \mathcal{S}(\mathcal{T})_{ext}^-\}$ of $(\mathcal{T}, \mathcal{I})$ (cf., Definition 8.2.2) such that $Q(\mathbf{t}\eta) = \text{root}(\mathcal{T})\theta$, $\mathcal{V} \cup \mathcal{B}$ is consistent, and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{V} \cup \mathcal{B})_{ext}^+$.

We may then traverse to $\mathcal{S}(\mathcal{T})$, extend η with θ , and update $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{B}$.

8.5 Reducing the cost of compilation.

In Section 8.4 we employed the notion of branch consistency to guarantee termination of our compilation tree construction. This is clearly an expensive test to perform, and although it is carried out only during the compilation (i.e., pre-processing) phase, it might still be considered a somewhat “brute-force” approach. In [Jo96, Jo97, Jo98] we have observed that database constraints can often be employed to guarantee prop-

erties of top-down computation (such as termination). In this section we note that database constraints can be used to overcome the need to employ branch consistency, whilst still allowing a significant amount of pre-processing.

Specifically we may develop, for each intensional predicate P , a P -*compilation tree*. This has as its root, the negative atom $\neg P(x_1, x_2, \dots, x_k)$, and is such that its instantiated branches encode weakly cyclic covers of ground instances of $\neg P(x_1, x_2, \dots, x_k)$. In this case however, these weakly cyclic covers will not necessarily be int-total, as a result of the fact that we are not now assuming the inclusion of splitting rules (since they violate our constraints below).

In order to compute our P -compilation tree without the use of branch consistency, we note that at the propositional level, ancestor pruning coupled with the fact that there are only a finite number of literals is of course sufficient to guarantee termination. We could employ this argument at the first order level *if* we were able to control the introduction of new variables, there being two points at which this might occur.

Firstly, when we apply a rule whose body contains an \exists -variable, then (a renaming of) the \exists -variable is introduced into the tree (as a new variable). We can ensure that these new variables play no part in the subsequent extension of the tree, by assuming that “for each rule R which contains an \exists -variable, there is a distinguished *extensional* atom P^R in $\text{antec}(R)$ which is the only atom in the body of R containing \exists - variables.”

The second point at which new variables may be introduced is in the construction of complete pairs. We can prevent this by using *semi-definite* predicates [Jo97], where (informally) a predicate P is semi-definite iff whenever P appears in the consequent of a rule R , then R is definite (i.e., $|\text{conseq}(R)| = 1$) and every intensional predicate appearing in the body of R is semi-definite. Under the above-mentioned assumption on \exists -variables we may then easily show that if P is semi-definite and $(\mathcal{T}, \mathcal{I})$ is a complete pair for $P(\mathbf{t})$, then every intensional atom in $\mathcal{S}(\mathcal{T})$ contains only variables appearing in \mathbf{t} . (As an aside we also recall [Jo98a] that such trees can be assumed to be linear in the sense given in Section 9.7.) Thus if we assume that for each rule $R \in \text{DB}$, every predicate in $\mathcal{N}(R)$ is either semi-definite or extensional, then the new variables introduced during the construction of complete pairs occur only in (negative) extensional atoms and thus do not endanger termination.

We may then compute our P -compilation tree in a similar fashion to Section 8.4, but here we do not employ ancestor-pruning splitting or branch consistency. In extending a node n via (R, θ, δ) , we insist that $\text{conseq}(R\theta) \subseteq \text{atom}(n)$ (by the prior application of unification splitting if necessary), and that no atom in the body of $R\theta\delta$ matches that of any atom in $\text{atom}(n)$. The sub-tree below (R, θ, δ) may be prohibited from containing a rule node of the form (R, θ, γ) (by ancestor pruning), and the sub-tree below the negative child of the unification split (if it exists) may be prohibited from

containing a repetition of the same split. Since the number of variables appearing in intensional atoms is bounded (in fact equal to the arity of P), there can only be a finite number of unification splits, and these constraints therefore guarantee the termination of the construction⁸.

The subsequent computation of int-total cyclic strong covers can then proceed as in Section 8.3, with the exception that if \mathcal{Q}_i is to be extended with a negative atom $\neg P(a_1, a_2, \dots, a_k)$ (clause 3 of Definition 8.3.1(iv)), then \mathcal{Q}_{i+1} is a cyclic strong cover of $\mathcal{Q}_i \cup \{\neg P(a_1, a_2, \dots, a_k)\}$, which may be computed from the P -compilation tree as in Section 8.4.5 (with the exception that clause 2 of Definition 8.3.1(iv) is used to handle the distinguished extensional atoms $P^{\mathbf{R}}$).

A natural question to ask is whether we can weaken the constraint imposed on \exists -variables. Firstly it is easy to show that we may extend the above argument to the case when \exists - variables in $\text{antec}(\mathbf{R})$ are restricted to a set of extensional positive atoms rather than a single such atom. When \exists - variables are not restricted to extensional atoms however, we face a much more difficult problem. For example given a rule \mathbf{R} of the form $P(x, y) \rightarrow Q(x)$, if P is intensional, then repeated application of \mathbf{R} might be necessary to allow the subsequent application of disjunctive rules whose consequent has the form $P(a, u) \vee P(a, v)$ say. The difficulty is then to limit the number of applications of \mathbf{R} without disallowing some such subsequent rule application. In the context of computing strong covers in positive databases, the approach adopted in [Jo97] was to control \exists - variables by limiting them to (stratified) semi-definite predicates, and allowing them to become instantiated only in response to a specific demand from a potential rule application. This then allowed strong covers in $gr(\text{INT}(\text{DB}))$ to be represented by ground instances of a first order tree (although not by individual branches through the tree). A similar approach is adopted in [Ba97] in the context of hyper-tableaux, where the application of additional variants of a rule are generated in a demand driven way, according to the requirements of potential rule applications. It is certainly worth considering whether such approaches could be employed within the context of compilation.

8.6 Answer size.

In [Jo98, Jo98a], we presented a top-down query processing method applicable to first order (stratified) databases. In this sub-section we show that the approach presented in the present paper overcomes a problem found in [Jo98, Jo98a].

Suppose for example that we have a unary predicate Q , then the query $?Q(x)$

⁸We conjecture that there are other constraints that we could impose upon the tree construction that would have the effect of forcing termination more quickly.

represents the query $? \bigvee \{Q(a) \mid a \text{ is a constant in } \mathcal{L}\}$. An answer of the form $\bigvee_{i \leq r} Q(a_i)$ is witnessed as being an answer by the absence of a cyclic strong cover (of $\{Q(a_i) \mid i \leq r\}$) which in turn is represented by an extended deduction tree whose root is labelled with $\{Q(a_i) \mid i \leq r\}$. The difficulty in this approach is deciding when to fix the size of the answer that we are seeking.

Conceptually the simplest approach is to start with the goal $? \bigvee_{i \leq m} Q(x_i)$, where each x_i is a variable, and m is the number of constants in \mathcal{L} . As an extended deduction tree is generated, the variables x_i become instantiated, and eventually yield an answer. Of course m will typically be very large, and this approach is therefore extremely inefficient (allowing massive redundancy) in the (usual) case when the answers actually being generated are much smaller than m . For non-unary predicates the potential redundancy is even worse.

The converse approach would be to first compute answers of size 1 (with initial goal $?Q(x_1)$), then answers of size 2 (with initial goal $?Q(x_1) \vee Q(x_2)$), and so on. The difficulty with this approach is knowing when the process can halt, since the computation of no new answers of size k implies nothing about the existence of minimal answers of size $k+1$. (Indeed it is shown in [Jo96] that for propositional databases, the problem of determining whether a minimal answer of size $> k$ exists is Σ_2^P -complete.)

The approach presented in the present paper of course overcomes this problem.

8.7 Disjunctive logic programs.

The insistence that deductive databases are function free is of course motivated by the desire that queries should have a finite number of answers (and in the disjunctive case, that answers should themselves be finite). Disjunctive logic programs on the other hand allow rules with function symbols (i.e., they are full first order), and hence query processing encounters the problems of first order undecidability. This is also true of minimal answer generation and testing [Ba97a, In02].

Limitations would therefore need to be imposed on the use of function symbols to allow an extension of our methods. Restrictions on the use of function symbols which yield a fragment of clausal logic decidable by hyper-resolution are presented in [Ge02], and we conjecture that similar results could be obtained in the present context. In addition, some preliminary results on the computation of cyclic trees in infinite propositional languages are to be found in [Jo01, Section 4].

It is also worth noting that at the first order level, subsumption can be employed to provide alternative definitions of the notion of a minimal answer ([Ba97, In02]). In [In02] it is shown that under such definitions, the problem of infinite answer sets can occur even in the function free case.

§9. Related approaches

In this section we briefly describe some related approaches, and compare and contrast their features with those of the method presented in previous sections.

9.1 Hyper-resolution.

Let us first consider the hyper-resolution operator

$$\frac{\bigwedge_{i \leq r} A_i \rightarrow \bigvee Q; A_i \vee \bigvee P_i (i \leq r)}{\bigvee(Q \cup \bigcup_{i \leq r} P_i)} \quad \dots (\diamond)$$

mentioned in the introduction. When employed as a bottom-up means of generating answers, the rule can be applied to (positive) propositional and first order databases alike. In the latter case range restriction implies that all disjunctions generated are ground, and of course the difficult problems of guaranteeing termination encountered in top-down processing (cf., Section 8) do not arise. However, as mentioned earlier, bottom-up approaches are not obviously amenable to compilation or pre-processing, and without such, a modification to the extensional database would require the recomputation of answers from scratch.

The second disadvantage is that non-minimal answers may clearly be generated, as was illustrated in the example in the introduction. To see that this is an instance of a general phenomenon, first note that a proof of $\bigvee \mathcal{R}_n$ using (\diamond) consists of a sequence $(\bigvee \mathcal{R}_j | j \leq n)$ of disjunctions, where each $\bigvee \mathcal{R}_j$ is generated from a subset of $\{\bigvee \mathcal{R}_l | l < j\}$ and some rule in DB using (\diamond) . Also we may clearly assume that for each $j < n$, $\bigvee \mathcal{R}_j$ is used as an input (via (\diamond)) to the generation of some $\bigvee \mathcal{R}_k$, where $j < k \leq n$. But then if some \mathcal{R}_j is such that $|\mathcal{R}_j| > |\mathcal{R}_n|$, the above proof must generate a pair $j_0 < j_1 \leq r$ such that $\mathcal{R}_{j_0} \supset \mathcal{R}_{j_1}$. To see this let j_0 be the largest integer for which $|\mathcal{R}_{j_0}| > |\mathcal{R}_n|$, and let $\bigvee \mathcal{R}_{j_0}$ be an input into the generation of $\bigvee \mathcal{R}_{j_1}$ ($j_1 > j_0$). But then by the maximality of j_0 , $|\mathcal{R}_{j_0}| > |\mathcal{R}_{j_1}|$ and by the definition of (\diamond) , $\mathcal{R}_{j_0} - \{A\} \subseteq \mathcal{R}_{j_1}$ for some $A \in \mathcal{R}_{j_0}$. Hence $\mathcal{R}_{j_1} = \mathcal{R}_{j_0} - \{A\}$. We will see below that this scenario (in which one of the input disjunctions is subsumed by the output disjunction) is not the only case where subsumption occurs.

Note also that if a database rule $\bigwedge \mathcal{A} \rightarrow \bigvee Q$ is employed in (\diamond) , then the disjunction generated contains Q . Thus for example if the predicate P is not derivable from the definite rules in DB (i.e., from $\{R \in \text{DB} : |\text{conseq}(R)| = 1\}$), then any proof of P from DB via (\diamond) encounters a non-minimal answer, as described above.

Of course whilst employing (\diamond) we can discard any disjunction that is found to be non-minimal. To see this suppose that $A_{i_0} \vee \bigvee P_{i_0}$ ($i_0 \leq r$) is an input to the

instance of (\diamond) given above, yielding the disjunction $\bigvee(\mathcal{Q} \cup \bigcup_{i \leq r} \mathcal{P}_i)$. If $\bigvee \mathcal{A}$ subsumes $A_{i_0} \vee \bigvee \mathcal{P}_{i_0}$, then either $A_{i_0} \notin \mathcal{A}$, in which case $\mathcal{A} \subseteq \mathcal{P}_{i_0} \subseteq \mathcal{Q} \cup \bigcup_{i \leq r} \mathcal{P}_i$, or $A_{i_0} \in \mathcal{A}$, in which case the instance of (\diamond) that employs $\bigvee \mathcal{A}$ instead of $A_{i_0} \vee \bigvee \mathcal{P}_{i_0}$ generates $\bigvee(\mathcal{Q} \cup (\mathcal{A} - \{A_{i_0}\}) \cup \bigcup_{i \leq r, i \neq i_0} \mathcal{P}_i)$ which clearly subsumes $\bigvee(\mathcal{Q} \cup \bigcup_{i \leq r} \mathcal{P}_i)$. This then shows that the instance of (\diamond) employing $A_{i_0} \vee \bigvee \mathcal{P}_{i_0}$ is redundant (even if the two outputs are equal, since then the same output can be generated by another instance of (\diamond)).

Of course a non-minimal answer cannot be discarded until we have generated a subsuming answer. Worse still, the very same argument that shows that we can (upon discovering a subsuming answer) remove the non-minimal answer also shows that any application of (\diamond) in which the non-minimal answer acted as an input (prior to its removal) will have generated a redundant (and possibly non-minimal) output. This suggests that not only can we eliminate non-minimal answers (as soon as they are detected), but that we should. However to do so requires that whenever we generate a new answer via (\diamond) we must test whether it is subsumed by any existing answer, and, if this is not the case, test whether it subsumes any existing answer (this requiring a comparison with every existing answer). Clearly this is highly costly.

Of course if we do employ subsumption as described above, and we ensure that no instance of (\diamond) is repeated, then we will eventually generate a set Δ of answers that contains all minimal answers, and no non-minimal answers (i.e., exactly the required set). Having generated Δ however, we will only be in a position to verify that it is indeed the set of minimal answers when we have checked that for every instance of (\diamond) employing a database rule and input disjunctions from Δ , either the instance has already been employed or it generates an answer that is subsumed by an answer in Δ . Needless to say the requirement to conduct, and test the output from, every such instance is extremely costly. Each of the instances of (\diamond) that we have to exercise following the generation of Δ is of course redundant, achieving nothing more than providing confirmation of Δ 's status. This is a further example of where the computation of minimal answers using (\diamond) is forced to generate redundant/non-minimal answers.

In addition, the relative size of the search spaces for minimal and non-minimal answers should not be under-estimated. If we take the database given in Appendix A.1, then there are precisely 3 minimal answers ($P_1 \vee A_3$, $Q_2 \vee Q_4$ and $Q_2 \vee A_3$), but 509 non-minimal answers. This example is of course artificial in that the language is very small (containing only 10 predicates), and the extensional database is very small (which in turn limits the number of minimal answers). But clearly for every minimal answer there are typically a very large number of non-minimal answers (i.e., all its supersets), thus if the number of minimal answers increases, then in general terms we would expect the same to be true of the number of non-minimal answers. For example

(again in our language of size 10) if we had a fourth minimal answer, say $Q_2 \vee A_1$, there would then be 572 non-minimal answers. Of course larger minimal answers would have fewer supersets, but in real examples we would expect the size of minimal answers to be very considerably smaller than the size of the underlying language.

If the language itself increases in size, then the number of disjunctions of predicates increases exponentially, and we would certainly expect therefore that the number of non-minimal answers will increase. For example if we had our 4 minimal answers above, but in an underlying language of size 11, there would then be 1148 non-minimal answers.

There is no suggestion that (\diamond) will generate all non-minimal answers, but the diversion into the much larger space of non-minimal answers is costly, and the point at which a non-minimal answer will become subsumed is unpredictable. This point highlights one of the novelties of our approach presented in Sections 3 and 5: here the search space is carefully constrained by “immediate verification” of chosen alternatives to ensure that we do not stray into the space of non-minimal answers.

Of course in real examples the number of minimal answers will typically be very large (although very small relative to the number of non-minimal answers), and there is no suggestion that the computation of (only) minimal answers is inexpensive. However, if we do wish to compute answers, then (as our example in the introduction illustrated) there is clearly no alternative but to compute (at least) the set of minimal answers. Moreover our approach using compilation allows a very large proportion of this computation to be conducted in the pre-processing phase.

Finally the large size of the set of minimal answers also points to the need to be able to compute only minimal answers to a specific query $?\bigvee \mathcal{H}$ (as in Section 6) instead of all minimal answers when this is required. It is unclear how a purely bottom-up application of (\diamond) could handle this, since there is no obvious way of directing the search. This is of course a common problem with bottom-up methods in general. A partial solution to this problem can be achieved using the approach presented in [Jo98], where top-down applications of (\diamond) are used to identify a bottom-up (\diamond) derivation of a subset of \mathcal{H} . Again answer minimality is not addressed by this approach.

9.2 Conditional facts.

The approach presented in [Bra95] to query processing in non-positive databases is to apply the following variant of (\diamond)

$$\frac{\bigwedge_{i \leq r} A_i \wedge \bigwedge \bar{\mathcal{N}} \rightarrow \bigvee \mathcal{Q}; \bigwedge \bar{\mathcal{N}}_i \rightarrow A_i \vee \bigvee \mathcal{P}_i (i \leq r)}{\bigwedge (\bar{\mathcal{N}} \cup \bigcup_{i \leq r} \bar{\mathcal{N}}_i) \rightarrow \bigvee (\mathcal{Q} \cup \bigcup_{i \leq r} \mathcal{P}_i)}$$

where \mathcal{N} and each \mathcal{N}_i are sets of predicates. A *conditional fact* is a rule \mathbf{R} for which $\text{antec}(\mathbf{R}) = \emptyset$. When applied at the first order level, the input conditional facts $\overline{\mathcal{N}_i} \rightarrow A_i \vee \bigvee \mathcal{P}_i$ are assumed ground, and hence again by range restriction the resulting conditional fact is also ground.

Repeated application transforms a non-positive database into a set of ground conditional facts which is equivalent to the original database under a variety of semantics including the disjunctive stable model semantics. The set of answers can then be generated from the transformed database using hyper-resolution, but again non-minimal answers may be generated and are to be removed using subsumption [Bra95, Definition 19]. In view of the similarity with (\diamond), the comments made in Section 9.1 apply equally well here.

We note that cyclic trees also yield a transformation into an equivalent set of conditional facts. If \mathcal{T} is a cyclic tree, then let $\rho(\mathcal{T}) = \bigwedge \{\neg P \mid P \in \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})\}$.

9.2.1 Theorem. Given DB, let $\text{DB}^* = \text{DB}_1 \cup \text{DB}_2$, where $\text{DB}_1 = \{\rho(\mathcal{T}) \rightarrow \text{root}(\mathcal{T}) \mid \mathcal{T} \text{ is a cyclic tree in DB}\}$, and $\text{DB}_2 = \{\bigwedge_{i \leq k} \rho(\mathcal{T}_i) \wedge \bigwedge \overline{\mathcal{N}(\mathbf{R})} \rightarrow \bigvee \text{conseq}(\mathbf{R}) \mid \mathbf{R} \in \text{DB}, \text{antec}(\mathbf{R}) = \{A_i \mid i \leq k\} \text{ and for each } i \leq k, \mathcal{T}_i \text{ is a cyclic tree for } A_i \text{ in DB such that } \text{Pred}(\mathcal{T}_i) \cap \text{conseq}(\mathbf{R}) = \emptyset\}$.

Then DB and DB^* have the same disjunctive stable models.

Proof (\rightarrow). Suppose that M is a disjunctive stable model of DB. Then $M \models \text{DB}$, and hence by Theorem 2.4(c), $M \models \text{DB}^*$. Thus $M \models \text{DB}^*|_g M$.

Suppose that $N \subset M$ with $N \models \text{DB}^*|_g M$. Let $P \in M - N$, then we may find a cyclic tree \mathcal{T} for P in DB such that $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$. But then the unit rule P is in $\text{DB}^*|_g M$, hence $P \in N$ yielding a contradiction. Thus M is a disjunctive stable model of DB^* .

(\leftarrow). Let M be a disjunctive stable model of DB^* . The result is proven by Theorem 2.6(a) and the following 3 lemmata.

Lemma 1. If \mathcal{T} is a cyclic tree in DB with $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$, then $\text{Pred}(\mathcal{T}) \subseteq M$.

Proof of Lemma 1. Now $\mathcal{S}(\mathcal{T})$ is cyclic in DB, thus if $P \in \text{Pred}(\mathcal{T})$, then by Theorem 2.6(c), we may find a cyclic tree \mathcal{V} for P in DB such that $\mathcal{S}(\mathcal{V}) \subseteq \mathcal{S}(\mathcal{T})$. In particular, $\mathcal{O}(\mathcal{V}) \cup \mathcal{N}(\mathcal{V}) \subseteq \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) \subseteq \mathcal{L} - M$. But then the unit rule P is in $\text{DB}^*|_g M$, hence $P \in M$. This then completes the proof of Lemma 1.

Lemma 2. $\overline{M} \cup (\mathcal{L} - M)$ is cyclic in DB.

Proof of Lemma 2. Let $A \in M$. By Theorem 2.6(c) it suffices to construct a cyclic tree \mathcal{T} for A in DB such that $\text{Pred}(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$.

Since $A \in M$ we may find a cyclic tree \mathcal{U} for A in DB^* such that $M \cap (\mathcal{O}(\mathcal{U}) \cup \mathcal{N}(\mathcal{U})) = \emptyset$. But now since every rule in DB^* is a conditional fact, \mathcal{U} must have the form depicted in Figure 9.2.1(i), where of course $\mathbf{S} \in \text{DB}^*$.

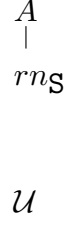


Figure 9.2.1(i)

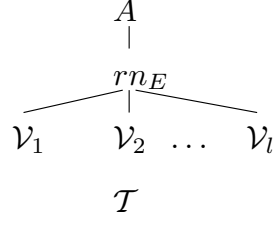


Figure 9.2.1(ii)

If $\mathbf{S} \in \text{DB}_1$, then \mathbf{S} has the form $\rho(\mathcal{T}) \rightarrow A$, where \mathcal{T} is a cyclic tree for A in DB . But then $\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}) = \mathcal{N}(\mathbf{S}) = \mathcal{N}(\mathcal{U}) \subseteq \mathcal{L} - M$, and by Lemma 1, $\text{Pred}(\mathcal{T}) \subseteq M$.

If $\mathbf{S} \in \text{DB}_2$, then \mathbf{S} has the form $\bigwedge_{j \leq l} \rho(\mathcal{V}_j) \wedge \bigwedge \overline{\mathcal{N}(\mathbf{E})} \rightarrow \bigvee \text{conseq}(\mathbf{E})$, where $\mathbf{E} \in \text{DB}$. Let \mathcal{T} be the tree depicted in Figure 9.2.1(ii): $\text{root}(\mathcal{T}) = A$ has $\text{rn}_{\mathbf{E}}$ as its child, and for each predicate in $\text{antec}(\mathbf{E})$ (i.e., each child of $\text{rn}_{\mathbf{E}}$), \mathcal{V}_j is appended as a child sub-tree below $\text{rn}_{\mathbf{E}}$.

Now $A \in \text{conseq}(\mathbf{S}) = \text{conseq}(\mathbf{E})$, which is disjoint from $\bigcup_{j \leq l} \text{Pred}(\mathcal{V}_j)$ by the definition of DB_2 . Thus for nodes in \mathcal{V}_j , the cycles and \mathcal{O} sets computed in \mathcal{T} are as computed in \mathcal{V}_j . Now $\mathcal{O}(\mathcal{V}_j) \cup \mathcal{N}(\mathcal{V}_j) \subseteq \mathcal{N}(\mathbf{S}) = \mathcal{N}(\mathcal{U}) \subseteq \mathcal{L} - M$, thus by Lemma 1, $\text{Pred}(\mathcal{V}_j) \subseteq M$, and hence $\text{Pred}(\mathcal{T}) = \{A\} \cup \bigcup_{j \leq l} \text{Pred}(\mathcal{V}_j) \subseteq M$. In addition, $\mathcal{O}(\mathcal{T}) = (\text{conseq}(\mathbf{E}) - \{A\}) \cup \bigcup_{j \leq l} \mathcal{O}(\mathcal{V}_j) = (\text{conseq}(\mathbf{S}) - \{A\}) \cup \bigcup_{j \leq l} \mathcal{O}(\mathcal{V}_j) = \mathcal{O}(\mathcal{U}) \cup \bigcup_{j \leq l} \mathcal{O}(\mathcal{V}_j) \subseteq \mathcal{L} - M$, and $\mathcal{N}(\mathcal{T}) = \mathcal{N}(\mathbf{E}) \cup \bigcup_{j \leq l} \mathcal{N}(\mathcal{V}_j) \subseteq \mathcal{N}(\mathbf{S}) = \mathcal{N}(\mathcal{U}) \subseteq \mathcal{L} - M$. Since $\text{Pred}(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$, it is clear that \mathcal{T} is indeed a cyclic tree for A in DB , and this then completes the proof of Lemma 2.

Lemma 3. $M \models \text{DB}$ (i.e., $\overline{M} \cup (\mathcal{L} - M)$ is a strong cover in DB).

Proof of Lemma 3. Let $\mathbf{R} \in \text{DB}$ with $\text{antec}(\mathbf{R}) = \{A_i \mid i \leq k\} \subseteq M \subseteq \mathcal{L} - (\text{conseq}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R}))$. By Lemma 2, for each $A_i \in \text{antec}(\mathbf{R})$ let \mathcal{T}_i be a cyclic tree for A_i in DB such that $\text{Pred}(\mathcal{T}_i) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$. But then $\text{Pred}(\mathcal{T}_i) \cap \text{conseq}(\mathbf{R}) = \emptyset$, hence $\bigwedge_{i \leq k} \rho(\mathcal{T}_i) \wedge \bigwedge \overline{\mathcal{N}(\mathbf{R})} \rightarrow \bigvee \text{conseq}(\mathbf{R}) \in \text{DB}_2$, but is not true in M . ■

Note that our transformation differs from that of [Bra95] in that it is not iterative. If we assume a partitioning of \mathcal{L} into $\text{EXT}(\mathcal{L})$ and $\text{INT}(\mathcal{L})$, then a transformation to conditional facts will *not* be immune to updates to the extensional database. However we can easily provide immunity by employing partial cyclic trees, for which we define $\rho(\mathcal{T}) = \bigwedge \{Q \mid Q \in \text{Pred}(\mathcal{T}) \cap \text{EXT}(\mathcal{L})\} \wedge \bigwedge \{-P \mid P \in \mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})\}$, and thereby transform DB into an equivalent database whose rules \mathbf{R} are such that $\text{antec}(\mathbf{R}) \subseteq \text{EXT}(\mathcal{L})$.

An extensional update against DB can then be applied verbatim to DB_2 (note that $EXT(DB) \subseteq DB_2$) thus providing the transformation with immunity to such updates.

Note also that the above transformations are directly applicable to first order databases. As mentioned in Section 8, cyclic trees are always ground, hence their use might be considered inappropriate. However partial cyclic trees are not ground, and this, coupled with the immunity to extensional updates makes them a more preferable transformation tool at the first order level.

9.3 Pre-processing using stratified and positive databases.

In [Fe95] it is shown that the disjunctive stable models of DB can be computed from the perfect models of the *stratified evidential transformation* of DB. Given the advantages that stratified databases enjoy, this too could be considered a pre-processing of the database, with a view to making subsequent query processing more efficient.

Using cyclic trees we can in fact transform a non-positive database to a positive database. For $P \in \mathcal{L}$, let $\phi(P) = \bigwedge \{ \neg \rho(\mathcal{T}) \mid \mathcal{T} \text{ is a cyclic tree for } P \text{ in } DB \}$, where of course $\neg \rho(\mathcal{T}) = \bigvee (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$. Let FALSE be a distinguished predicate not in \mathcal{L} , and in particular not appearing in DB. Then we have the following theorem.

9.3.1 Theorem. Let

$$DB^* = \{ \bigwedge \text{antec}(\mathbf{R}) \rightarrow \bigvee (\text{conseq}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R})) \mid \mathbf{R} \in DB \} \cup \{ P \wedge \phi(P) \rightarrow \text{FALSE} \mid P \in \mathcal{L} \}$$

Then

- (a) If $M \subseteq \mathcal{L}$, then M is a disjunctive stable model of DB iff M is a model of DB^* iff M is a minimal model of DB^* .
- (b) DB has a minimal model that is not disjunctive stable iff $DB^* \not\models \neg \text{FALSE}$.
- (c) DB has a disjunctive stable model iff $DB^* \not\models \text{FALSE}$.
- (d) If Φ is a formula in \mathcal{L} then $DB \models \Phi$ iff $DB^* \models \text{FALSE} \vee \Phi$.
- (e) If $\mathcal{A} \subseteq \mathcal{L}$, then \mathcal{A} is contained in a minimal answer in DB iff for each $P \in \mathcal{A}$ we may find a strong cover $\mathcal{C}_P \subseteq \mathcal{L} \cup \{ \text{FALSE} \}$ of $\{ \text{FALSE} \} \cup (\mathcal{A} - \{ P \})$ in DB^* such that $DB^* \models \bigvee \mathcal{A} \vee \bigvee \bigcap \{ \mathcal{C}_P \mid P \in \mathcal{A} \}$.

Proof (a). Let $M \subseteq \mathcal{L}$. If M is a disjunctive stable model of DB, then by Theorem 2.4(a), for each $P \in M$ we may find a cyclic tree \mathcal{T} for P in DB such that $M \models \rho(\mathcal{T})$. In particular $M \models \neg \phi(P)$, hence $M \models DB^*$.

Suppose that $M \models DB^*$, then clearly $M \models DB$ (and hence $\overline{M} \cup (\mathcal{L} - M)$ is a strong cover in DB). Since $M \subseteq \mathcal{L}$, we have that $\text{FALSE} \notin M$, hence for each $P \in M$ there is some cyclic tree \mathcal{T} for P in DB such that $M \models \rho(\mathcal{T})$. Since $M \models DB$ we have (by

Theorem 2.4(c)) that $Pred(\mathcal{T}) \subseteq M$, hence by Theorem 2.6(c), $\overline{M} \cup (\mathcal{L} - M)$ is cyclic, and by Theorem 2.6(a), M is disjunctive stable.

Finally M must be a minimal model of DB^* for if $N \subset M$ is a model of DB^* , then N is (also) a disjunctive stable model of DB , contradicting the minimality of disjunctive stable models.

(b) Suppose that $M \subseteq \mathcal{L}$ is a minimal model of DB that is not disjunctive stable. Clearly $M \cup \{\mathbf{FALSE}\} \models DB^*$. We show that $M \cup \{\mathbf{FALSE}\}$ is a minimal model of DB^* . Suppose that $M^* \subset M \cup \{\mathbf{FALSE}\}$ is a model of DB^* , then $M^* \cap \mathcal{L} \models DB$, hence by the minimality of M we must have that $M^* \cap \mathcal{L} = M$ and hence $M^* = M$. But then by part (a), M is disjunctive stable.

Conversely suppose that $M^* \subseteq \mathcal{L} \cup \{\mathbf{FALSE}\}$ is a minimal model of DB^* with $\mathbf{FALSE} \in M^*$. Let $M = M^* \cap \mathcal{L}$, then $M \models DB$. Moreover M is a minimal model of DB (for if $N \subset M$ is a model of DB then $N \cup \{\mathbf{FALSE}\}$ is a model of DB^*). By the minimality of M^* , M is not a model of DB^* , hence by part (a), M is not disjunctive stable.

Parts (c) and (d) follow trivially from part (a).

(e) If \mathcal{A} is contained in a minimal answer, then by Theorem 1.6, for each $P \in \mathcal{A}$ we may find a disjunctive stable model M_P of DB such that $M_P \cap \mathcal{A} = \{P\}$ and $DB \models \bigvee \mathcal{A} \vee \bigvee \bigcap \{\mathcal{L} - M_P \mid P \in \mathcal{A}\}$. By parts (a) and (d), $\mathcal{C}_P = \{\mathbf{FALSE}\} \cup (\mathcal{L} - M_P)$ is then the required strong cover.

Conversely, for each $P \in \mathcal{A}$, $M_P = (\mathcal{L} \cup \{\mathbf{FALSE}\}) - \mathcal{C}_P$ is a model of DB^* with $M_P \subseteq \mathcal{L}$ (since \mathcal{C}_P is a strong cover of $\{\mathbf{FALSE}\}$ in DB^*), hence by part (a), M_P is a disjunctive stable model of DB . Now $M_P \cap (\mathcal{A} - \{P\}) \subseteq M_P \cap \mathcal{C}_P = \emptyset$ and (since $M_P \models DB^*$) we have that $\emptyset \neq M_P \cap (\mathcal{A} \cup \bigcap \{\mathcal{C}_Q \mid Q \in \mathcal{A}\}) \subseteq \{P\}$, i.e., $P \in M_P$. Finally by hypothesis, $\bigvee \mathcal{A} \vee \bigvee \bigcap \{\mathcal{C}_P \mid P \in \mathcal{A}\} = \bigvee \mathcal{A} \vee \mathbf{FALSE} \vee \bigvee \bigcap \{\mathcal{L} - M_P \mid P \in \mathcal{A}\}$ is true in DB^* , hence by part (d), $DB \models \bigvee \mathcal{A} \vee \bigvee \bigcap \{\mathcal{L} - M_P \mid P \in \mathcal{A}\}$. The result then follows from Theorem 1.6. ■

Note that the database DB^* is not normalised. This can clearly be achieved (if desired) by applying distributivity to the denial rules $P \wedge \phi(P) \rightarrow \mathbf{FALSE}$, but this of course results in a far less compact representation. Note also that, in contrast to the approach presented in [Fe95], our transformed database is (with the exception of \mathbf{FALSE}) still in the original language⁹. As in Section 9.2 our transformation is not immune to extensional updates, which could again be achieved using partial cyclic trees.

Part (e) indicates that having constructed cyclic trees in DB (in order to construct DB^*), we can then construct minimal answers in DB by computing strong covers as opposed to cyclic strong covers in DB^* . Notice that the above un-normalised form of DB facilitates the construction of such strong covers, since a denial rule of the form

⁹We could give a similar, but less elegant characterisation without using \mathbf{FALSE} .

$P \wedge \phi(P) \rightarrow \text{FALSE}$ indicates that any strong cover of $\{\text{FALSE}\}$ should contain either P or $\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})$ for some cyclic tree \mathcal{T} for P in DB. (The reader may note that this is almost the same as the effect of a denial rule of the form $P \wedge \neg P \rightarrow \text{FALSE}$ when constructing cyclic strong covers using the approach given in Section 2).

Part (a) is interesting in that *all* models of $\text{DB}^* \cup \{\neg \text{FALSE}\}$ correspond to disjunctive stable models of DB. More generally the proof of part (a) indicates that the disjunctive stable models of DB equal the models of $\text{DB}^+ \cup \{P \wedge \phi(P) \rightarrow \text{FALSE} \mid P \in \mathcal{L}\} \cup \{\neg \text{FALSE}\}$ where DB^+ is any database in \mathcal{L} that shares the same models as DB.

As given, we do not have a direct mapping between minimal answers of DB and those of DB^* . However this can easily be achieved by adding FALSE to the consequent of each rule in DB^* , and making the trivial assumption that some rule in DB has an empty antecedent. This ensures that $\{\text{FALSE}\}$ is a minimal model of DB^* , and hence that $\mathcal{A} \subseteq \mathcal{L}$ is a minimal answer in DB iff $\mathcal{A} \cup \{\text{FALSE}\}$ is a minimal answer in DB^* .

9.4 Model trees.

We can of course generate answers by first computing models of the database. For example in [Ya94, Fe95, Fe95a], bottom-up methods for computing minimal, perfect and (using the stratified evidential transformation) disjunctive stable models are presented using the notion of a *model tree*. The suggested approach to computing answers is to then pick a predicate from each of the branches (i.e., models) in the tree [Fe95a], although subsumption would again be required to compute minimal answers.

The need for subsumption could of course be overcome by employing the full set of models in the method presented in Section 5 (which would then reduce to repeated application of the trivial operator defined in Definition 5.7.3(a)). Irrespective of this however the approach still has two serious drawbacks. Firstly the full set of (disjunctive stable) models will typically be (very) large. For example if DB is the database illustrated in Example 5.8, then with only an extremely small extensional database $\{A_3 \vee P_2, P_1 \vee P_2, A_1 \vee A_2 \vee P_3\}$ there are 7 disjunctive stable models. Secondly, their computation is not immune to changes in the extensional database, so for example an update to say $\{A_3 \vee P_2, P_1 \vee A_1, A_2 \vee P_3\}$ results in 11 disjunctive stable models (all of which would need to be recomputed from scratch).

The computation of int-total weakly cyclic covers on the other hand can clearly be viewed as a partial computation of (disjunctive stable) models, which, as a result of its immunity to extensional update, can be carried out in a pre-processing stage. The subsequent (post-compilation) generation of minimal answers is then relatively trivial (amounting to minimal model computation in $\text{EXT}(\text{DB})$), and moreover does not require the computation of disjunctive stable models in their entirety since the cyclic strong covers are only required to be int-total (which in turn occurs automatically from the

fact that the cyclic strong covers are formed by extending the int-total weakly cyclic covers computed during compilation). These features considerably reduce the run-time cost for the end-user (cf., Sections 5.5-5.8 and Appendix B).

[Aside: By Proposition 5.5(c), given an int-total weakly cyclic cover \mathcal{C} in $\text{INT}(\text{DB})$, $\bar{\mathcal{C}}$ can be extended to a disjunctive stable model of DB if (and only if) $\text{EXT}(\text{DB}) \not\models \bigvee \mathcal{C}_{ext}$. Thus \mathcal{C} plays no part in the generation of minimal answers *for* DB if $\text{EXT}(\text{DB}) \models \bigvee \mathcal{C}_{ext}$. This is clearly a consequence of the fact that the set of int-total weakly cyclic covers generated by our compilation process has to work with any extension (in order to provide immunity to extensional updates).

Even if $\bigvee \mathcal{C}_{ext}$ is entailed by the current extension, there will of course be (many) other extensions in which it is not. Indeed given any total consistent interpretation \mathcal{I} containing $\bar{\mathcal{C}}$ there will always be at least one (and usually many) extensional databases $\text{EXT}(\text{DB}^*)$ for which \mathcal{I}^+ is a disjunctive stable model of $\text{INT}(\text{DB}) \cup \text{EXT}(\text{DB}^*)$, e.g., $\text{EXT}(\text{DB}^*) = \{P \mid P \in \mathcal{I}_{ext}^+\}$.

Moreover if $\text{EXT}(\text{DB}^*) \not\models \bigvee \mathcal{C}_{ext}$, we would typically expect (particularly in the case when $\text{EXT}(\text{DB}^*)$ is large) there to be many disjunctive stable models of $\text{EXT}(\text{DB}^*) \cup \text{INT}(\text{DB})$ containing $\bar{\mathcal{C}}$, since the literals in \mathcal{C}_{ext} would typically not imply truth values for all other predicates in $\text{EXT}(\mathcal{L})$.]

An alternative approach to query processing, again using model trees, is presented in [Ya96, Ya02], where it is observed that top-down query processing for positive databases can be achieved by the application of a model generating procedure to the dual database. It is not clear that this approach can be extended to non-positive databases, and as noted in [Ya96], answer minimality is not addressed.

9.5 DLV.

DLV [Le03] is a knowledge representation system based upon (function free) disjunctive logic programming (augmented with true negation and weak constraints).

At its core, DLV has a model generator and model checker. The model generator constructs in the first instance an interpretation $\mathcal{W}_{\text{DB}}^\omega(\emptyset)$ as a fixed point of the operator \mathcal{W}_{DB} [Le97, Le03], this being an adaptation (to the disjunctive case) of unfoundedness [VG91]. $\mathcal{W}_{\text{DB}}^\omega(\emptyset)$ is contained in any disjunctive stable model, but $\mathcal{W}_{\text{DB}}^\omega(\emptyset)$ is not necessarily total (and clearly could not be total in the case when there are multiple disjunctive stable models). Thus in order to generate total interpretations, $\mathcal{W}_{\text{DB}}^\omega(\emptyset)$ is extended using a second operator that is based upon the notion of a *possibly true set* of literals [Le97, Le03]. If \mathcal{I} is a consistent set of literals and \mathbf{R} is a rule in DB with $\text{antec}(\mathbf{R}) \subseteq \mathcal{I}^+$ and $(\text{conseq}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R})) \cap \mathcal{I}^+ = \emptyset$, then for each $P \in \text{conseq}(\mathbf{R})$, $\{P\} \cup \overline{\mathcal{N}(\mathbf{R})}$ is a *possibly true set* of literals for \mathcal{I} , and the extension of \mathcal{I} using this possibly true set would yield $\mathcal{I} \cup \{\overline{P} \mid P \in \text{conseq}(\mathbf{R})\}$.

If \mathcal{I} is an interpretation and M is a disjunctive stable model with $\mathcal{I}^+ \subset M \subseteq \mathcal{L} - \mathcal{I}^-$, then $\mathcal{I}^+ \not\models \text{DB}|_g M$, and hence there is some rule $\mathbf{R} \in \text{DB}$ such that $\mathcal{N}(\mathbf{R}) \cap \mathcal{I}^+ \subseteq \mathcal{N}(\mathbf{R}) \cap M = \emptyset$, $\text{antec}(\mathbf{R}) \subseteq \mathcal{I}^+$ and $\text{conseq}(\mathbf{R}) \cap \mathcal{I}^+ = \emptyset$. Since $M \models \mathbf{R}$ we must have that $M \cap \text{conseq}(\mathbf{R}) \neq \emptyset$, and if $P \in \text{conseq}(\mathbf{R}) \cap M$, then $\{P\} \cup \overline{\mathcal{N}(\mathbf{R})}$ is a possibly true set for \mathcal{I} with $\{P\} \cup \overline{\mathcal{N}(\mathbf{R})} \subseteq M \cup \overline{(\mathcal{L} - M)}$. Thus every disjunctive stable model can be generated by a sequence of such extension steps, and the approach is therefore *complete*.

The above argument is of course reminiscent of that employed in Example 2.2: here the predicate P is being supported (in $\mathcal{I} \cup \{P\} \cup \overline{\mathcal{N}(\mathbf{R})}$) by the presence of $\overline{\text{antec}(\mathbf{R}) \cup \mathcal{N}(\mathbf{R})}$. Note however that the possibly true set has not included $\overline{\text{conseq}(\mathbf{R}) - \{P\}}$, and as a result, the support being provided to the predicate P (as a member of the intended model) can easily be undone (if subsequent extension steps add predicates from $\overline{\text{conseq}(\mathbf{R}) - \{P\}}$). As a result of this, repeated extension via possibly true sets may generate models that are not disjunctive stable (and partial interpretations that cannot be extended to disjunctive stable models) (e.g., if $\text{DB} = \{A \vee B, \neg C \rightarrow B\}$, then $\mathcal{I} = \{A\}$ is a possibly true set in \emptyset , and the subsequent extension to $\{A, B, \neg C\}$ undoes the support for A), hence the requirement for a separate model checker to test for stability¹⁰. (In addition, it may generate total interpretations that are not models, although it is probably the case that any closure operation would encounter this problem, e.g., if $\text{DB} = \{\neg A \wedge \neg B \rightarrow C, A \vee B\}$, then $\mathcal{I} = \{\neg A, \neg B, C\}$ is a possibly true set in \emptyset .)

In order to try to force every model generated to be stable, we might attempt to re-define the possibly true set to include $\overline{\text{conseq}(\mathbf{R}) - \{P\}}$, but then we lose completeness: for example if $\mathcal{I} = \emptyset$ and $\text{DB} = \{A \vee B, A \rightarrow B, B \rightarrow A\}$, then there is no rule $\mathbf{R} \in \text{DB}$ and predicate $P \in \text{conseq}(\mathbf{R})$ for which $\text{antec}(\mathbf{R}) \subseteq \mathcal{I}^+$ and $\{P\} \cup \overline{\mathcal{N}(\mathbf{R})} \cup \overline{(\text{conseq}(\mathbf{R}) - \{P\})}$ is true in the disjunctive stable model $\{A, B\}$. This difficulty is due to the fact that the inherent support relationships within a disjunctive stable model require more than the simple support provided by single rules.

The approach to (disjunctive) stable model generation suggested in Example 2.7 is based upon the repeated extension of (partial) branches in an extended deduction tree using the operators described in Definition 2.8. This approach is complete (in the sense given above), in that if \mathcal{C} is a total cyclic strong cover properly containing the literals on a partial branch \mathcal{B} , then \mathcal{B} can be extended and \mathcal{C} must contain the literals found at one of the child nodes of \mathcal{B} .

¹⁰DLV also employs additional look ahead techniques to try to limit these problems. In this respect it is worth noting that determining whether a partial interpretation is contained in a disjunctive stable model is Σ_2^P - complete, and determining whether a model is disjunctive stable is Π_1^P - complete [Ei97].

The operator defined in Definition 2.8 clearly has two purposes. Firstly it extends Definition 2.1 which in turn tries to ensure that the literals on the branch eventually form a strong cover, and hence yield a model. Secondly it insists that the literals on the branch are always cyclic, i.e., if we add a negative literal $\neg P$, then we must also add $\mathcal{S}(\mathcal{T})$ for some cyclic tree \mathcal{T} for P in DB. Theorem 2.4(c) then ensures that P is supported in *any* subsequent extension, i.e., the support cannot be undone.

As with extension via possibly true sets, this process may obviously generate partial cyclic strong covers that cannot be further extended (if they are contained in no total cyclic strong cover), and again it is clearly not possible to prevent an extension step from generating a total set of literals that is not a strong cover (i.e., does not correspond to a model of DB). On the other hand, if neither of these two options occurs, then the resulting set of literals is a total cyclic strong cover (and hence by Theorem 2.6(a) yields a disjunctive stable model). In particular, no separate model checking is required to test for stability: we have forced any resulting model to be stable by insisting that we maintain cyclicness.

To see the relationship between the two approaches, first observe that the proposed extension of \mathcal{I} to $\mathcal{I} \cup \{P\} \cup \overline{\mathcal{N}(\mathbf{R})} \cup \overline{(\text{conseq}(\mathbf{R}) - \{P\})}$ can be viewed as employing a tree \mathcal{T} such that $\text{root}(\mathcal{T}) = P$ has child $rn_{\mathbf{R}}$ which in turn has a predicate child Q for each $Q \in \text{antec}(\mathbf{R})$. \mathcal{T} satisfies the conditions of a cyclic tree (Definition 2.3) except that we allow predicate leaf nodes. Each predicate node at the leaf of \mathcal{T} has a label in $\text{antec}(\mathbf{R})$ (and hence in \mathcal{I}^+) and $\mathcal{I} \cup \{P\} \cup \overline{\mathcal{N}(\mathbf{R})} \cup \overline{(\text{conseq}(\mathbf{R}) - \{P\})} = \mathcal{I} \cup \overline{\mathcal{S}(\mathcal{T})}$. This form of extension compromises completeness because the trees being used are too trivial (being based upon a single rule) to capture to required support relationships.

We can overcome this by allowing the use of larger trees. Specifically, suppose that given a consistent interpretation \mathcal{I} we allow \mathcal{I} to be extended to $\mathcal{I} \cup \overline{\mathcal{S}(\mathcal{T})}$ if \mathcal{T} is a tree for some $P \in \mathcal{L} - (\mathcal{I}^+ \cup \mathcal{I}^-)$ in DB (satisfying the conditions of Definition 2.3, with the exception that it may contain predicate leaf nodes) such that $\{\text{lab}(n) \mid n \text{ is a predicate leaf node in } \mathcal{T}\} \subseteq \mathcal{I}^+$ and $\mathcal{I} \cup \overline{\mathcal{S}(\mathcal{T})}$ is consistent. Theorem 2.4(a) then dictates that this extension operator is complete (in the above sense), and clearly if \mathcal{I} is closed under this extension operator, then \mathcal{I}^+ is a model of DB. But moreover we can show that such an approach would generate only models that are disjunctive stable, since if $\overline{\mathcal{I}}$ is cyclic and \mathcal{T} is a tree satisfying the above conditions, then $\overline{\mathcal{I}} \cup \mathcal{S}(\mathcal{T})$ is cyclic [Jo99a].

This modified approach now has some resemblance to the approach presented in Section 2 (see the remarks following Example 2.7). The outstanding difference lies in the fact that Definition 2.8 is purely top-down, employing the predicates designated to fall outside the eventual model (i.e., \mathcal{Q}^+ in the terminology of Definition 2.8) in order to try to ensure we generate a strong cover. This was also our primary mechanism by which new members of the (intended) model are generated (cf., Definition 2.8(ii)).

In contrast the definition of a possibly true sets makes no mention of \mathcal{I}^- , and the approach of [Le03] is bottom-up using look-ahead to identify potential new members of the intended model (although the suggested use of larger trees would introduce a top-down element).

Finally we note that the above-mentioned differences between the method presented in Section 2 and that of [Le03] again reflects one of the fundamental features of the approach being presented in the current paper, i.e., “immediate verification”. With regard to the generation of disjunctive stable models, we immediately restore the cyclic property (upon generating a new negative literal), and this in turn prevents the generation of non-stable models. Similarly in Sections 3-5, the verification of cyclic states prevents the generation of non-minimal answers.

9.6 Unfolding disjunction.

In common with [Fe95] (cf., Section 9.3 above), the idea of generating a transformed database from which the disjunctive stable models of the original database can be computed has also been considered in [JN00]. Specifically it is shown that given a disjunctive database DB , we may compute a non-disjunctive database $Gen(DB)$ (in an extended language) from which the disjunctive stable models of DB may be computed by a two step process. Firstly stable models M of $Gen(DB)$ are computed as candidate disjunctive stable models of DB . Given such a candidate model M , $M \cap \mathcal{L}$ is then a disjunctive stable model of DB iff a further transformed (non-disjunctive) database $Test(DB, M \cap \mathcal{L})$ has no stable model. In particular, the computation of disjunctive stable models of disjunctive databases can be achieved by computing stable models of non-disjunctive databases, the approach suggested (in [JN00]) for the latter being smodels [Si00, Si02, Si03].

It is suggested in [JN00] that the performance of this approach suffers (in relation to the performance of DLV), partly as a result of the fact that it requires the cooperation of two stable model generators. This in turn suggests that a direct transformation to a non-disjunctive database might be beneficial (in addition to being of general interest). A key idea in the transformation from DB to $Gen(DB)$ is that every predicate in a disjunctive stable model must have a rule which supports its presence in the model. However we have observed that predicates must have an entire cyclic tree (rather than just a single rule) to support their presence, and using this we can provide the required transformation (which now takes place in the original language). Such a transformation could again be considered a pre-processing step prior to answer generation. This and similar results (e.g., Theorem 9.2.1, Theorem 9.3.1, and those presented in [Jo99, Section 4]) reinforces our view that cyclic trees provide the “correct” form of support

for minimal (and related) models of disjunctive databases. Let

$$shift(DB) = \{\bigwedge[\text{antec}(\mathbf{R}) \cup \overline{\mathcal{N}(\mathbf{R})} \cup \overline{(\text{conseq}(\mathbf{R}) - \{P\})}] \rightarrow P \mid \mathbf{R} \in DB, P \in \text{conseq}(\mathbf{R})\}.$$

It is well known that every stable model of $shift(DB)$ is a disjunctive stable model of DB , but that the converse does not hold. This results from the more complex support relationships found in disjunctive databases (which are captured by cyclic trees).

9.6.1 Theorem. M is a disjunctive stable model of DB iff M is a stable model of

$$DB^* = shift(DB) \cup \{\rho(\mathcal{T}) \rightarrow \text{root}(\mathcal{T}) \mid \mathcal{T} \text{ is a cyclic tree in } DB\}.$$

Proof (\rightarrow). The proof is exactly the same as that of Theorem 9.2.1 (\rightarrow).

(\leftarrow). Let M be a stable model of DB^* , then $M \models DB^*$, thus it is clear that $M \models DB$, and in particular $M \models DB|_g M$. Suppose that $N \subset M$ with $N \models DB|_g M$. We show that $N \models DB^*|_g M$, thus contradicting the stability of M .

Suppose that $\mathbf{S} = \rho(\mathcal{T}) \rightarrow P$ with $\text{pos}(\mathbf{S}) \in DB^*|_g M$, i.e., $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$. But then $DB|_g M \supseteq DB|_g(\mathcal{L} - \mathcal{N}(\mathcal{T}))$, and since $N \cap \mathcal{O}(\mathcal{T}) = \emptyset$, Theorem 2.4(b) dictates that $P \in \text{Pred}(\mathcal{T}) \subseteq N$.

Suppose that $\mathbf{S} = \bigwedge[\text{antec}(\mathbf{R}) \cup \overline{\mathcal{N}(\mathbf{R})} \cup \overline{(\text{conseq}(\mathbf{R}) - \{P\})}] \rightarrow P$ is in $shift(DB)$, where $\mathbf{R} \in DB$, with $\text{pos}(\mathbf{S}) \in DB^*|_g M$, and $\text{antec}(\mathbf{S}) = \text{antec}(\mathbf{R}) \subseteq N$. Then $\text{pos}(\mathbf{R}) \in DB|_g M$ with $\text{antec}(\text{pos}(\mathbf{R})) = \text{antec}(\mathbf{R}) \subseteq N$, thus $N \cap \text{conseq}(\mathbf{R}) \neq \emptyset$. Since $(\text{conseq}(\mathbf{R}) - \{P\}) \cap N \subseteq (\text{conseq}(\mathbf{R}) - \{P\}) \cap M = \emptyset$ we must have that $P \in N$. ■

As in Section 9.1 we can employ partial cyclic trees to provide virtual immunity to extensional update. A modification to $\text{EXT}(DB)$ then simply requires the corresponding trivial modification to the subset of $shift(DB)$ relating to $\text{EXT}(DB)$, i.e., $\{\bigwedge \overline{(\text{conseq}(\mathbf{R}) - \{P\})} \rightarrow P \mid \mathbf{R} \in \text{EXT}(DB), P \in \text{conseq}(\mathbf{R})\}$. As in Section 9.1, partial cyclic trees also provide a first order transformation.

As noted above, $shift(DB)$ alone is not always equivalent to DB . For example ([Le03]) if $DB = \{A \rightarrow B, B \rightarrow A, A \vee B\}$, then $shift(DB) = \{A \rightarrow B, B \rightarrow A, \neg A \rightarrow B, \neg B \rightarrow A\}$ (which has no disjunctive stable model). In this case the transformed database given in Theorem 9.6.1 has the form $\{A \rightarrow B, B \rightarrow A, \neg B \rightarrow A, \neg A \rightarrow B, A, B\}$, the unit rule A (say) being obtained from the cyclic tree \mathcal{T} (for A) constructed (top to bottom) using $B \rightarrow A$, $A \rightarrow B$, and then $A \vee B$ (to handle the cycle), since $\rho(\mathcal{T})$ is then the conjunct of the empty set.

We also note that an alternative mapping to a non-disjunctive database can be obtained using the database $shift(DB) \cup \{P \wedge \phi(P) \rightarrow \text{FALSE} \mid P \in \mathcal{L}\}$ (cf., the penultimate paragraph in Section 9.3).

9.7 Head-cycle free databases.

DB is said to be *head-cycle free* (HCF) [Be94] iff there is a level function $\ell : \mathcal{L} \rightarrow \mathbb{N}$ such that for each rule $A_1 \wedge A_2 \wedge \dots \wedge A_h \wedge \neg A_{h+1} \wedge \dots \wedge \neg A_{h+r} \rightarrow B_1 \vee B_2 \vee \dots \vee B_k$ in DB: (i) $\ell(A_i) \leq \ell(B_j)$ for each $i \leq h, j \leq k$, and (ii) $\ell(B_{j_0}) \neq \ell(B_{j_1})$ for each $1 \leq j_0 < j_1 \leq k$.

HCF databases have been widely studied, partly because the computational complexity of many problems is reduced [Be94]. It is also well known ([Be94]) that the (disjunctive) stable models of an HCF database DB coincide with the stable models of $shift(DB)$, and this then allows us to easily prove the following results.

9.7.1 Theorem. Suppose that DB is HCF. Then:

- (a) \mathcal{C} is cyclic in DB iff \mathcal{C} is cyclic in $shift(DB)$.
- (b) If M is a disjunctive stable model of DB and $P \in M$, then we may find a cyclic tree \mathcal{T} for P in DB such that
 - (i) $Pred(\mathcal{T}) \subseteq M \subseteq \mathcal{L} - (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T}))$, and
 - (ii) for each predicate node n in \mathcal{T} , there is no predicate node $m > n$ such that $lab(m) = lab(n)$.

Thus the cyclic tree in part (b) has no non-trivial cycles (and in particular $CYC(n)$ is always equal to $\{lab(n)\}$), and is linear in the sense that if $rn_{\mathbf{R}}$ is the child of n , then $conseq(\mathbf{R}) \cap \{lab(m) \mid m \geq n\} = \{lab(n)\}$. In view of the equivalence of DB and $shift(DB)$, this result correlates with those of [Jo98a] where it is shown (for non-HCF databases) that this form of linearity can always be adopted during the construction of cyclic trees should processing pass into the definite part of the database.

More generally we would argue that the HCF concept and cyclic trees have opposing motivations, in that the former concept disallows positive cycles (thereby allowing simpler processing and reducing the computational complexity), whereas the latter provides a means of tackling such cycles.

§10. Conclusions

We have presented a method of computing (only) minimal answers of the form $\bigvee \mathcal{A}$ in disjunctive deductive databases (and to the best of our knowledge this is the first such method). The method achieves this by generating (and extending) partial minimal answers, with “immediate verification” being employed at each stage to ensure that a new predicate used to extend a partial answer has the required properties.

We have also discussed the problems inherent in extending the method to unstratified databases under the disjunctive stable model semantics. The possible absence of disjunctive stable models would seem to imply the need to force the generation of disjunctive stable models in their entirety by the addition of denial rules of the form $P \wedge \neg P \rightarrow \mathbf{FALSE}$. Compilation (in this context, computing int-total weakly cyclic covers in a pre-processing phase) has been proposed as a solution to this problem, and also as a means of simplifying and improving the efficiency of the run-time computation, which then effectively amounts to minimal model reasoning in the extensional database. In addition, compilation has the significant advantage of being immune to updates to the extensional database. We have also presented a number of (other) forms of database transformation which might be employed as pre-processing steps prior to answer generation (as well as being of general interest), these results being based upon the strong form of support provided by cyclic trees.

We have compared our approach to other answer generating methods in the literature. These methods (typically) either generate answers directly (as in the hyper-resolution operator mentioned in the introduction), or generate models as an intermediate step. In either case non-minimal answers may be generated (which then need to be removed by subsumption), and (significantly) the lack of immunity to extensional update means that all answers would need to be recomputed from scratch in the event of such an update. It is also unclear how such techniques might take advantage of the situation in which we wish to compute only answers to a specific query $?\bigvee \mathcal{H}$ (as opposed to all answers). In the context of generating answers, we believe that constructing models in their entirety is undesirable in view of their size, number, and lack of immunity to extensional update, and our computation of int-total weakly cyclic covers can be viewed as a partial (and immune) computation of such models.

In spite of these comments the approach presented in the current paper is not entirely inconsistent with other methods appearing in the literature.

Firstly it is clear that computing minimal answers to a query of the form $?\bigvee \mathcal{H}$ is more efficient than computing minimal answers to $?\bigvee \mathcal{L}$, since the search space is constrained by \mathcal{H} . To take advantage of this, we might first employ an (arbitrary) answer generator to yield answers $\bigvee \mathcal{Q}$ (some of which might be non-minimal), and then apply the techniques of the current paper to find minimal answers within each of the queries $?\bigvee \mathcal{Q}$. It is however not obvious that such an approach is advantageous. If the first answer generator lacks immunity to extensional update, then so does the whole approach. Moreover every minimal answer $\bigvee \mathcal{A}$ (to $?\bigvee \mathcal{L}$) must be contained in at least one of the answers $\bigvee \mathcal{Q}$, and if it is contained in several, we risk duplicating the computation of $\bigvee \mathcal{A}$.

A second possibility might be to employ (or modify) an arbitrary (disjunctive

stable) model generator to construct cyclic sets (cf., Theorem 2.10). Surprisingly we can (with a little extra work) also employ an arbitrary model generator to construct the int-total weakly cyclic covers required for compilation (see Appendix D). We do however believe that the application of a *bottom-up* model generator for this purpose would result in a computation that is highly prone to redundancy. Moreover it is unclear whether this approach could be adapted to the first order level.

A natural question arising from our results is whether there is a weaker property than stratification which guarantees that every cyclic strong cover may be extended to a total cyclic strong cover. In the non-disjunctive case it is well-known that a database DB is guaranteed to have a stable model if every cycle (in DB's predicate dependency graph) contains an even number of negative edges [Du92]. It is clear that every such cycle in $DB_{\mathcal{D}}$ (Definition 2.11) is also a cycle in DB, thus if DB has the above mentioned property, then so does $DB_{\mathcal{D}}$, and hence (by Theorem 2.12) every cyclic strong cover may be extended to a total cyclic strong cover.

Acknowledgement. The author wishes to thank the two anonymous referees for their many helpful comments.

References

- [Ba97] P. Baumgartner, Hyper-tableaux - The next generation, Technical Report 32/97, Institut für Informatik, Universität Koblenz-Landau (1997).
- [Ba97a] P. Baumgartner, U. Furbach and F. Stolzenburg, Computing answers with model elimination, *Artificial Intelligence*, vol. 90 (1997), 135-176.
- [Be94] R. Ben-Eliyahu and R. Dechter, Propositional semantics for disjunctive logic programs, *Annals of Mathematics and Artificial Intelligence*, vol. 12 (1994), 53-87.
- [Bi81] W. Bibel, On matrices with connections, *J. Association for Computing Machinery*, vol. 28 (1981), 633-645.
- [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, A general approach to bottom-up computation of disjunctive semantics, in J. Dix, L. Pereira, T. Przymusiński (eds.): *Non-Monotonic Extensions of Logic Programming*, Springer Lecture Notes in Artificial Intelligence, vol. 927 (Springer, Berlin, 1995), 127-155.
- [Bra98] S. Brass and J. Dix, Characterisations of the disjunctive well-founded semantics: confluent calculi and iterated GCWA, *J. Automated Reasoning*, vol. 20 (1998),

- [Du92] P.M. Dung, On the relations between stable and well-founded semantics of logic programs, *Theoretical Computer Science*, vol. 105 (1992), 7-25.
- [Ei93] T. Eiter and G. Gottlob, Propositional circumscription and extended closed world reasoning are \prod_2^P - complete, *Theoretical Computer Science*, vol. 114 (1993)
- [Ei97] T. Eiter, G. Gottlob and H. Mannila, Disjunctive Datalog, *ACM Transactions on Database Systems*, vol. 22 (1997), 364-418.
- [Fe95] J. Fernández, J. Minker and A. Yahya, Computing perfect and stable models using ordered model trees, *Computational Intelligence*, vol. 11 (1995), 89-112.
- [Fe95a] J. Fernández and J. Minker, Bottom-up computation of perfect models for disjunctive theories, *Journal of Logic Programming*, vol. 25 (1995), 33-51.
- [Ge88] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski and K. Bowen (eds.), *Proc. 5th International Conference on Logic Programming*, Seattle (1988), 1070-1080.
- [Ge02] L. Georgieva, U. Hustadt and R.A. Schmidt, A new clausal class decidable by hyperresolution, University of Manchester Department of Computer Science technical report CSPP-18. Available May, 2003: www.cs.man.ac.uk/~schmidt/.
- [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.
- [He88] L.J. Henschen and H. Park, Compiling the GCWA in indefinite databases, in J. Minker, ed., *Foundations of Deductive Databases*, pp 395-438 (Morgan Kaufmann, Washington), 1988.
- [In02] K. Inoue and K. Iwanuma, Minimal Answer Computation and SOL, in: S. Greco and N. Leone (eds.), *Logics in Artificial Intelligence: Proceedings of the Eighth European Conference*, (Springer Lecture Notes in Artificial Intelligence, vol. 2424, 2002), 245-257.
- [JN00] T. Janhunen, I. Niemelä, D. Seipel, P. Simons and J.H. You, Unfolding partiality and disjunctions in stable model semantics, in: A.G. Cohn, F. Giunchiglia and B. Selman (eds.): *Proceedings of the 7th International Conference on the Principles of Knowledge Representation and Reasoning*, (Morgan Kaufmann, San Francisco, 2000), 411-419.
- [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*, vol. 19 (1997), 31-85.
- [Jo98] C.A. Johnson, Top down query processing in indefinite deductive databases, *Data and Knowledge Engineering*, vol. 26 (1998), 1-36.
- [Jo98a] C.A. Johnson, Extended deduction trees and query processing, Keele University

technical report TR98-07. Available May, 2003: www.tech.plym.ac.uk/soc/staff/chrisjohnson/tr95.html

- [Jo99] C.A. Johnson, On cyclic covers and perfect models, *Data and Knowledge Engineering*, vol. 31 (1999), 25-65.
- [Jo99a] C.A. Johnson, Processing deductive databases under the disjunctive stable model semantics, *Fundamenta Informaticae*, vol. 40 (1999), 31-51.
- [Jo00] C.A. Johnson, Top-down query processing in first order deductive databases under the DWFS, in Z. Ras and S. Ohsuga (eds), *Foundations of Intelligent Systems, 12th International Symposium on Methodologies for Intelligent Systems, Charlotte, USA* (Springer, 2000), 377-388.
- [Jo01] C.A. Johnson, On the computation of the disjunctive well-founded semantics, *J. Automated Reasoning*, vol. 26 (2001), 333-356.
- [Jo02] C.A. Johnson, Processing indefinite deductive databases under the possible model semantics, *Fundamenta Informaticae*, vol. 49 (2002), 325-347.
- [Jo03] C.A. Johnson, Query compilation under the disjunctive well-founded semantics, pre-print. www.tech.plym.ac.uk/soc/staff/chrisjohnson/compilation/compilation.pdf (available May, 2003)
- [Lb92] J. Lobo, J. Minker and A. Rajasekar, *Foundations of Disjunctive Logic Programming*, (MIT Press, Cambridge, Massachusetts, 1992).
- [Le97] N. Leone, P. Rullo, and F. Scarcello, Disjunctive stable models: Unfounded sets, fixpoint semantics and computation, *Information and Computation*, vol. 135 (1997), 69-112.
- [Le03] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. *The DLV System for Knowledge Representation and Reasoning*, arxiv.org/abs/cs.AI/0211004 (available October 2003).
- [Mi82] J. Minker, On indefinite databases and the closed world assumption, in D.W. Loveland (ed.): *Proceedings of the 6th Conference on Automated Deduction*, Springer Lecture Notes in Computer Science, vol. 138 (Spring, Berlin, 1982), 292-308.
- [Pa84] C. Papadimitriou and M. Yannakakis, The complexity of facets (and some facets of complexity), *Journal of Computer and Systems Sci.*, vol. 28 (1984) 244 - 259.
- [Pr88] T. Przymusinski, On the declarative semantics of deductive databases and programs, in: J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, (Morgan Kauffman, Washington, 1988).
- [Pr91] T. Przymusinski, Stable semantics for disjunctive databases, *New Generation Computing*, vol. 9 (1991), 401-424.
- [Pr91a] T. Przymusinski, Stationary semantics for normal and disjunctive logic programs, in: C. Delobel, M. Kifer and Y. Masunaga (eds.), *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases DOOD'91*,

Munich (Springer Verlag, 1991), 85-107.

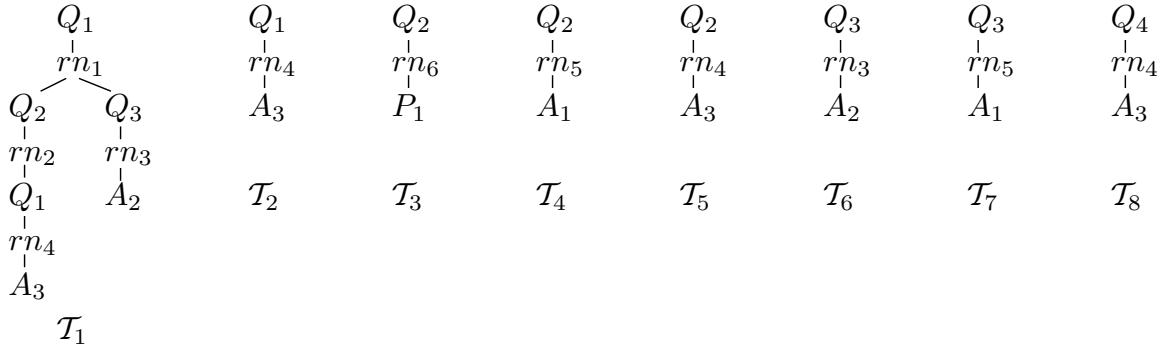
- [Ra89] A. Rajasekar, Semantics for Disjunctive Logic Programs, PhD thesis, University of Maryland (1989).
- [Sa94] C. Sakama and K. Inoue, An alternative approach to the semantics of disjunctive logic programs and deductive databases, *J. Automated Reasoning*, vol. 13 (1994), 145-172.
- [Si00] P. Simons, Extending and Implementing the Stable Model Semantics, Doctoral dissertation, Research Report 58, Helsinki University of Technology, Helsinki (2000).
- [Si02] P. Simons, I. Niemelä and T. Soinen, Extending and implementing the stable model semantics, *Artificial Intelligence*, vol. 138 (2002), 181-234.
- [Si03] P. Simons and T. Syrjänen, SMOBELS and LPARSE - a solver and a grounder for normal logic programs, www.tcs.hut.fi/Software/smodels/ (available October 2003).
- [St77] L. Stockmeyer, The polynomial-time hierarchy, *Theoretical Computer Science*, vol. 3 (1977), 1-22.
- [VG91] A. Van Gelder, K. Ross and J. S. Schlipf, The well-founded semantics for general logic programs, *J. Association of Computing Machinery*, vol. 38, (1991), 620-650.
- [Wa86] K. Wagner and G. Wechsung, *Computational Complexity* (Reidel, 1986).
- [Wr77] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoretical Computer Science*, vol. 3 (1977), 23-33.
- [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.
- [Ya96] A. Yahya, A goal-driven approach to efficient query processing in disjunctive databases, Technical Report PMS-FB-1996-12, Institut für Informatik, Ludwig Maximilians Universität, München, Germany.
- [Ya02] A. Yahya, Duality for efficient query processing in disjunctive deductive databases, *J. Automated Reasoning*, vol. 28 (2002).

Appendix A: Worked example.

Let $\text{INT}(\mathcal{L}) = \{Q_1, Q_2, Q_3, Q_4\}$ and $\text{INT}(\text{DB})$ consist of the following rules

- | | | | | | |
|----|---|----|---------------------------------------|----|---------------------------------------|
| 1. | $Q_2 \wedge Q_3 \wedge \neg P_1 \rightarrow Q_1 \vee Q_4$ | 2. | $Q_1 \wedge \neg P_2 \rightarrow Q_2$ | 3. | $A_2 \wedge \neg P_3 \rightarrow Q_3$ |
| 4. | $A_3 \rightarrow Q_1 \vee Q_2 \vee Q_4$ | 5. | $A_1 \rightarrow Q_2 \vee Q_3$ | 6. | $P_1 \rightarrow Q_2$ |

then the partial cyclic trees (cf., Section 5) in INT(DB) are as follows.



There is one further tree (for Q_2) which yields the same $\mathcal{S}(\mathcal{T})$ set as \mathcal{T}_1 , thus we omit it. Below we present, for each tree, the set $\mathcal{S}(\mathcal{T})$.

1. $\mathcal{T}_1 : \{\neg Q_1, \neg Q_2, \neg Q_3, Q_4, P_1, P_2, P_3, \neg A_2, \neg A_3\}$
2. $\mathcal{T}_2 : \{\neg Q_1, \neg A_3, Q_2, Q_4\}$
3. $\mathcal{T}_3 : \{\neg Q_2, \neg P_1\}$
4. $\mathcal{T}_4 : \{\neg Q_2, \neg A_1, Q_3\}$
5. $\mathcal{T}_5 : \{\neg Q_2, \neg A_3, Q_1, Q_4\}$
6. $\mathcal{T}_6 : \{\neg Q_3, \neg A_2, P_3\}$
7. $\mathcal{T}_7 : \{\neg Q_3, \neg A_1, Q_2\}$
8. $\mathcal{T}_8 : \{\neg Q_4, \neg A_3, Q_1, Q_2\}$.

By Proposition 5.3(d), if \mathcal{C} is a weakly cyclic cover containing $\neg Q_i$, then there must be some partial cyclic tree \mathcal{T} for P such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}$. (By the above remark, \mathcal{T}_1 must also count as a tree for Q_2 .)

As indicated in Section 2, we can depict the generation of weakly cyclic covers using (the analogues) of the operations presented in Section 2.8: in particular, negative atoms are expanded with the relevant $\mathcal{S}(\mathcal{T})$ sets. For the sake of brevity in this example, we first compute the minimal strong covers of the above $\mathcal{S}(\mathcal{T})$ sets in INT(DB):

1. $\mathcal{C}_1 = \{\neg Q_1, \neg Q_2, \neg Q_3, Q_4, P_1, P_2, P_3, \neg A_2, \neg A_3\}$
2. $\mathcal{C}_2 = \{\neg Q_1, \neg A_3, Q_2, Q_4, \neg P_2, P_1\}$
3. $\mathcal{C}_3 = \{\neg Q_2, \neg P_1\}$
4. $\mathcal{C}_{4a} = \{\neg Q_2, \neg A_1, Q_3, A_2\}$ and
 $\mathcal{C}_{4b} = \{\neg Q_2, \neg A_1, Q_3, \neg P_3\}$
5. $\mathcal{C}_{5a} = \{\neg Q_2, \neg A_3, Q_1, Q_4, Q_3, A_2\}$,
 $\mathcal{C}_{5b} = \{\neg Q_2, \neg A_3, Q_1, Q_4, Q_3, \neg P_3\}$,
 $\mathcal{C}_{5c} = \{\neg Q_2, \neg A_3, Q_1, Q_4, \neg P_1\}$
6. $\mathcal{C}_6 = \{\neg Q_3, \neg A_2, P_3\}$
7. $\mathcal{C}_{7a} = \{\neg Q_3, \neg A_1, Q_2, Q_1, P_1\}$,
 $\mathcal{C}_{7b} = \{\neg Q_3, \neg A_1, Q_2, \neg P_2, P_1\}$

8. $\mathcal{C}_8 = \{\neg Q_4, \neg A_3, Q_1, Q_2, P_1\}$.

Thus if \mathcal{C} is a weakly cyclic cover containing $\neg Q_i$, then it must contain one of (i) $\mathcal{C}_1, \mathcal{C}_2$, when $i = 1$; (ii) $\mathcal{C}_1, \mathcal{C}_3, \mathcal{C}_{4a}, \mathcal{C}_{4b}, \mathcal{C}_{5a}, \mathcal{C}_{5b}, \mathcal{C}_{5c}$, when $i = 2$; (iii) $\mathcal{C}_6, \mathcal{C}_{7a}, \mathcal{C}_{7b}$ when $i = 3$; and (iv) \mathcal{C}_8 when $i = 4$.

We represent the minimal int-total weakly cyclic covers as the branches through the tree depicted in Figure A.1. Note that the path to \mathcal{C}_{5c} is not int-total, but has not been expanded since the set of literals along the path contains the set of literals along the path to the sibling \mathcal{C}_3 , the extensions of which have already been generated. Notice also that in expanding $\neg Q_2$ we have not employed \mathcal{C}_1 , since it would generate an inconsistent branch. Although such trees allow us to represent weakly cyclic covers, it is unclear whether they provide a means of *choosing* cyclic covers during the computation of minimal answers.

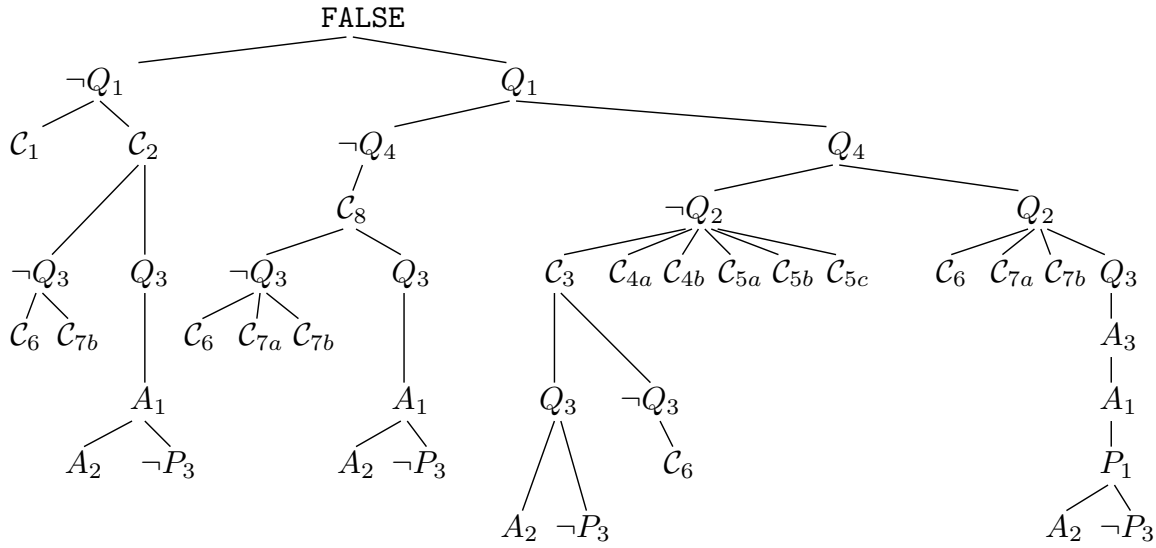


Figure A.1

Below we enumerate the int-total weakly cyclic covers identified. The reader will note that in some of these sets we have (for the sake of readability) left duplicates in. The notes on the right indicate the branch of the above tree.

Case A. $\neg Q_1 \in \mathcal{C}$

- | | |
|--|--------------------------------------|
| 1. $\{\neg Q_1, \neg Q_2, \neg Q_3, Q_4, P_1, P_2, P_3, \neg A_2, \neg A_3\}$ | \mathcal{C}_1 |
| 2. $\{\neg Q_1, \neg A_3, Q_2, Q_4, \neg P_2, P_1, \neg Q_3, \neg A_2, P_3\}$ | $\mathcal{C}_2, \mathcal{C}_6$ |
| 3. $\{\neg Q_1, \neg A_3, Q_2, Q_4, \neg P_2, P_1, \neg Q_3, \neg A_1, Q_2, \neg P_2, P_1\}$ | $\mathcal{C}_2, \mathcal{C}_{7b}$ |
| 4. $\{\neg Q_1, \neg A_3, Q_2, Q_4, \neg P_2, P_1, Q_3, A_1, A_2\}$ | $\mathcal{C}_2, Q_3 \in \mathcal{C}$ |
| 5. $\{\neg Q_1, \neg A_3, Q_2, Q_4, \neg P_2, P_1, Q_3, A_1, \neg P_3\}$ | $\mathcal{C}_2, Q_3 \in \mathcal{C}$ |

Case B. $Q_1 \in \mathcal{C}$ and $\neg Q_4 \in \mathcal{C}$

- | | |
|--|--------------------------------------|
| 6. $\{Q_1, \neg Q_4, \neg A_3, Q_1, Q_2, P_1, \neg Q_3, \neg A_2, P_3\}$ | $\mathcal{C}_8, \mathcal{C}_6$ |
| 7. $\{Q_1, \neg Q_4, \neg A_3, Q_1, Q_2, P_1, \neg Q_3, \neg A_1, Q_2, Q_1, P_1\}$ | $\mathcal{C}_8, \mathcal{C}_{7a}$ |
| 8. $\{Q_1, \neg Q_4, \neg A_3, Q_1, Q_2, P_1, \neg Q_3, \neg A_1, Q_2, \neg P_2, P_1\}$ | $\mathcal{C}_8, \mathcal{C}_{7b}$ |
| 9. $\{Q_1, \neg Q_4, \neg A_3, Q_1, Q_2, P_1, Q_3, A_1, A_2\}$ | $\mathcal{C}_8, Q_3 \in \mathcal{C}$ |
| 10. $\{Q_1, \neg Q_4, \neg A_3, Q_1, Q_2, P_1, Q_3, A_1, \neg P_3\}$ | $\mathcal{C}_8, Q_3 \in \mathcal{C}$ |
| Case C. $Q_1 \in \mathcal{C}, Q_4 \in \mathcal{C}$ and $\neg Q_2 \in \mathcal{C}$ | |
| 11. $\{Q_1, Q_4, \neg Q_2, \neg P_1, Q_3, A_2\}$ | $\mathcal{C}_3, Q_3 \in \mathcal{C}$ |
| 12. $\{Q_1, Q_4, \neg Q_2, \neg P_1, Q_3, \neg P_3\}$ | $\mathcal{C}_3, Q_3 \in \mathcal{C}$ |
| 13. $\{Q_1, Q_4, \neg Q_2, \neg P_1, \neg Q_3, \neg A_2, P_3\}$ | $\mathcal{C}_3, \mathcal{C}_6$ |
| 14. $\{Q_1, Q_4, \neg Q_2, \neg A_1, Q_3, A_2\}$ | \mathcal{C}_{4a} |
| 15. $\{Q_1, Q_4, \neg Q_2, \neg A_1, Q_3, \neg P_3\}$ | \mathcal{C}_{4b} |
| 16. $\{Q_1, Q_4, \neg Q_2, \neg A_3, Q_1, Q_4, Q_3, A_2\}$ | \mathcal{C}_{5a} |
| 17. $\{Q_1, Q_4, \neg Q_2, \neg A_3, Q_1, Q_4, Q_3, \neg P_3\}$ | \mathcal{C}_{5b} |
| 18. $\{Q_1, Q_4, \neg Q_2, \neg A_3, Q_1, Q_4, \neg P_1\}$ | \mathcal{C}_{5c} |
| Case D. $Q_1 \in \mathcal{C}, Q_4 \in \mathcal{C}$ and $Q_2 \in \mathcal{C}$ | |
| 19. $\{Q_1, Q_4, Q_2, \neg Q_3, \neg A_2, P_3, A_3, P_1\}$ | \mathcal{C}_6 |
| 20. $\{Q_1, Q_4, Q_2, \neg Q_3, \neg A_1, Q_2, Q_1, P_1, A_3\}$ | \mathcal{C}_{7a} |
| 21. $\{Q_1, Q_4, Q_2, \neg Q_3, \neg A_1, Q_2, \neg P_2, P_1, A_3\}$ | \mathcal{C}_{7b} |
| 22. $\{Q_1, Q_4, Q_2, Q_3, A_3, A_1, P_1, A_2\}$ | $Q_3 \in \mathcal{C}$ |
| 23. $\{Q_1, Q_4, Q_2, Q_3, A_3, A_1, P_1, \neg P_3\}$ | $Q_3 \in \mathcal{C}$ |

A.1 Example. Suppose now that $\text{EXT}(\text{DB})$ contains a single rule $P_1 \vee A_3$. There are then precisely 3 minimal int-total cyclic strong covers:

- | | |
|--|---------|
| $\mathcal{D}_1 = \{Q_1, \neg Q_4, \neg A_3, P_1, Q_2, Q_3, A_1, A_2\}$ | from 9 |
| $\mathcal{D}_2 = \{Q_1, Q_4, \neg Q_2, \neg P_1, A_3, Q_3, A_2\}$ | from 11 |
| $\mathcal{D}_3 = \{Q_1, Q_4, \neg Q_2, \neg A_3, P_1, Q_3, A_2\}$ | from 16 |

[Note that $\mathcal{D} = \{Q_1, Q_4, Q_2, Q_3, A_3, A_1, P_1, A_2\}$ is a completion of (22), but it is not a strong cover since $\text{EXT}(\text{DB}) \models \bigvee \mathcal{D}_{ext}^+$.]

Note that Q_1 and Q_3 do not therefore belong to any minimal answer. $((Q_2, \mathcal{D}_2))$ is a (verified) cyclic state of length 1, thus let us consider its extension using the approach detailed in Section 5.7.3. We pick an int-total cyclic strong cover \mathcal{C} of $\{Q_2\}$, the only choice being $\mathcal{C} = \mathcal{D}_1$.

Suppose first that we intend to (try to) extend using a predicate Q in $\text{INT}(\mathcal{L})$, then we require that $Q \in \mathcal{D}_1^- \cap \mathcal{D}_2^+$, and (not surprisingly) the only option is Q_4 . We may then extend the cyclic state to $((Q_2, \mathcal{D}_2), (Q_4, \mathcal{D}_1))$. There are no int-total cyclic strong covers of $\{Q_2, Q_4\}$, thus $Q_2 \vee Q_4$ is a minimal answer (and clearly is the only minimal answer contained in $\text{INT}(\mathcal{L})$).

Suppose that we now try to extend $((Q_2, \mathcal{D}_2))$ with a predicate E in $\text{EXT}(\mathcal{L})$. Section 5.7.3 demands that $E \notin \mathcal{D}_1^+ \cup \mathcal{D}_2^-$, and also that there exists a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $E \in \mathcal{E}$ and $\text{EXT}(\text{DB}) \not\models \bigvee (\mathcal{E} - \{E\}) \vee \bigvee (\mathcal{D}_1)_{ext}^+$. A_3 satisfies the conditions of Definition 5.7.3, with $(\mathcal{E} - \{E\}) \cup \mathcal{D}_1 = \mathcal{D}_1$, and $((Q_2, \mathcal{D}_2), (A_3, \mathcal{D}_1))$ is an immediate extension of $((Q_2, \mathcal{D}_2))$. $\{Q_2, A_3\}$ has no int-total cyclic strong cover, thus $Q_2 \vee A_3$ is a minimal answer.

It is clear that P_1 and A_3 are the only extensional predicates that belong to a minimal model of $\text{EXT}(\text{DB})$ (and hence to a minimal answer). Moreover $Q_2 \vee P_1, Q_4 \vee P_1$ and $Q_4 \vee A_3$ are not answers (they all have an int-total cyclic strong cover), hence there are just 3 minimal answers, namely $P_1 \vee A_3, Q_2 \vee Q_4$ and $Q_2 \vee A_3$.

Appendix B: Complexity.

In this section we present a number of complexity results relating to the issues of the current paper.

It is well known that many questions relating to the disjunctive stable model semantics are Σ_2^P - complete (e.g., determining whether a given predicate belongs to some disjunctive stable model; determining whether a given predicate belongs to some minimal answer). Similarly, given a disjunction $\bigvee_{i \leq n} A_i$, the problem of determining whether $\text{DB} \not\models \bigvee_{i \leq n} A_i$ is Σ_2^P - complete (e.g., [Ei97, Le03]).

A conjunction of a Π_n formula and a Σ_n formula is sometimes referred to as D_n [Pa84].

B.1 Theorem. Given DB and $\mathcal{P} \subseteq \mathcal{L}$, the problem of determining whether $\bigvee \mathcal{P}$ is a minimal answer in DB is D_2^P -complete.

Proof.

D_2^P - *hardness*. First recall that if DB is a positive disjunctive database, then the problem of determining whether a predicate P belongs to some minimal model of DB (i.e., $\text{DB} \not\models \neg P$) is Σ_2^P - complete [Ei93]. A D_2^P problem can therefore be reduced to checking whether, given two positive disjunctive databases DB_0 and DB_1 , and two predicates P_0 and P_1 , it is the case that $\text{DB}_0 \not\models \neg P_0$ and $\text{DB}_1 \models \neg P_1$.

We may assume that DB_0 and DB_1 have no predicates in common, and that (for each i) P_i appears in DB_i . Let E_0, D_0, D_1, Q_0, Q_1 be predicates not appearing in either DB_i . Let DB_0^* be formed from DB_0 by adding D_0 to the antecedent and E_0 to the consequent of every rule (in DB_0). Let DB_1^* be formed from DB_1 by adding D_1 to the antecedent of every rule (in DB_1). Let $\text{DB}^* = \text{DB}_0^* \cup \text{DB}_1^* \cup \{D_0 \vee D_1, D_0 \wedge \neg P_0 \rightarrow Q_0, D_1 \wedge \neg P_1 \rightarrow Q_1\}$.

We show that $\text{DB}_0 \not\models \neg P_0$ and $\text{DB}_1 \models \neg P_1$ iff $Q_1 \vee P_0 \vee Q_0$ is a minimal answer in DB^* .

(\rightarrow .) Suppose that M_0 is a minimal model of DB_0 such that $P_0 \in M_0$ and $\text{DB}_1 \models \neg P_1$.

We first show that $\text{DB}^* \models Q_1 \vee P_0 \vee Q_0$; suppose that M is a stable model of DB^* . If $D_0 \in M$ then $P_0 \vee Q_0$ is true in M . If $D_1 \in M$ then M restricted to the predicates in DB_1 forms a minimal model of DB_1 , and therefore $P_1 \notin M$. Thus $Q_1 \in M$.

Note that $M_0 \cup \{D_0\}$ is a stable model of DB^* which does not intersect $\{Q_0, Q_1\}$. If M_1 is a minimal model of DB_1 , then $M_1 \cup \{D_1, Q_1\}$ is a stable model of DB^* which does not intersect $\{P_0, Q_0\}$. Finally $\{D_0, E_0, Q_0\}$ contains a stable model of DB^* which does not intersect $\{Q_1, P_0\}$.

(\leftarrow .) Conversely suppose that $Q_1 \vee P_0 \vee Q_0$ is a minimal answer in DB^* . Suppose that M_1 is a minimal model of DB_1 with $P_1 \in M_1$, then $M_1 \cup \{D_1\}$ is a stable model of DB^* in which $Q_1 \vee P_0 \vee Q_0$ is false.

Let M be a stable model of DB^* containing P_0 but not containing Q_0 or Q_1 . But then we must have that $E_0 \notin M$ and $D_0 \in M$, whence M restricted to the predicates in DB_0 is a minimal model of DB_0 containing P_0 .

D_2^P -completeness. $\bigvee_{i \leq n} A_i$ is a minimal answer in DB iff $\text{DB} \models \bigvee_{i \leq n} A_i$, and for each $j \leq n$, $\text{DB} \not\models \bigvee_{i \leq n, i \neq j} A_i$.

The first of these problems is Π_2^P -complete (e.g., [Ei97, Le03]), and the second is therefore Σ_2^P -complete. The problem of testing whether $\bigvee_{i \leq n} A_i$ is a minimal answer can therefore be reduced to testing the truth of a D_2 formula. ■

A D_1^P -complete problem relating to minimal answers is discussed briefly in [Jo96, Section 3.5]. Similarly we may prove similar results for cyclic strong cover membership.

B.2 Theorem. Suppose that we are given DB and a literal K . The problem of determining whether K belongs to some cyclic strong cover in DB is Σ_2^P -complete.

Proof.

Σ_2^P -hardness. If DB is a positive database, then $\neg P$ belongs to a cyclic strong cover in DB iff P belongs to some minimal model of DB (cf., Theorem 3.2) (and the latter problem is known to be Σ_2^P -complete).

Similarly for positive literals, if $\text{DB}^* = \text{DB} \cup \{\neg P \rightarrow Q\}$, where Q is a predicate not appearing in DB , then Q belongs to some cyclic strong cover in DB^* iff P belongs to some minimal model of DB .

Σ_2^P -completeness. Theorem 2.10(b) can easily be rephrased as follows: K belongs to some cyclic strong cover of DB iff there exists a consistent set of literals \mathcal{C} such that $K \in \mathcal{C}$, \mathcal{C} is a strong cover in DB , and for all $\mathcal{N} \subseteq \mathcal{L}$, if $\mathcal{N} \subset \mathcal{C}^-$ then $\mathcal{N} \not\models (\text{DB}/\mathcal{C})|_g \mathcal{C}^-$.

This condition can clearly be expressed as a Σ_2 formula, thus demonstrating Σ_2^P - completeness. ■

For HCF databases (Section 9.7), the complexity is of course reduced to NP-completeness, since the same is true of (disjunctive) stable model membership [Be94].

As regards the computational complexity of post-compilation processing (Definitions 5.6.2 and 5.7.3), the inputs to this processing are the int-total weakly cyclic covers generated during compilation $\text{COMP} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n\}$ and $\text{EXT}(\text{DB})$. Both definitions (5.6.2 and 5.7.3) demand that we compute an element of $\text{COMP}(\mathcal{A})$ (for a given set of predicates \mathcal{A}). For each $i \leq n$, \mathcal{C}_i has a completion in $\text{COMP}(\mathcal{A})$ iff $\mathcal{A} \cup \mathcal{C}_i$ is consistent, and $\text{EXT}(\text{DB})$ has a minimal model in which $\bigvee \mathcal{A}_{ext} \vee \bigvee (\mathcal{C}_i)_{ext}$ is false. Determining whether such a minimal model exists is easily shown to be NP - complete.

B.3 Theorem. Given $\text{EXT}(\text{DB})$ and a set of literals \mathcal{K} (from $\text{EXT}(\mathcal{L})$), the problem of determining whether $\text{EXT}(\text{DB})$ has a minimal model in which $\bigvee \mathcal{K}$ is false, is NP-complete.

Proof.

NP-hardness. Suppose that Φ is a set of clauses in \mathcal{L} (where a clause is a disjunction of literals). For each proposition $P \in \mathcal{L}$, introduce two new propositions P' and P^* , and let Φ' be the result of replacing each negative literal (in Φ) $\neg Q$ by Q' . Let $\Phi^* = \Phi' \cup \{P \vee P^* \mid P \in \mathcal{L}\} \cup \{P' \vee P^* \mid P \in \mathcal{L}\}$. Then Φ has a model iff Φ^* has a minimal model in which $\bigvee \{\neg P^* \mid P \in \mathcal{L}\}$ is false.

NP-completeness. Suppose that we are given $\text{EXT}(\text{DB})$ and a set of literals \mathcal{K} (from $\text{EXT}(\mathcal{L})$). For each clause $C \in \text{EXT}(\text{DB})$, let $Q(C)$ denote a new proposition. For each $P \in \mathcal{K}^-$, let $C_1^P, C_2^P, \dots, C_{r_P}^P$ ($r_P \geq 0$) denote those rules in $\text{EXT}(\text{DB})$ containing P . Φ is formed from $\text{EXT}(\text{DB})$ by adding, for each $P \in \mathcal{K}^-$, a new clause $Q(C_1^P) \vee Q(C_2^P) \vee \dots \vee Q(C_{r_P}^P)$, for each $i \leq r_P$, the clauses $\{\neg R \vee \neg Q(C_i^P) \mid R \in C_i^P - \{P\}\}$, and then finally adding the unit clauses $\{\neg P \mid P \in \mathcal{K}^+\}$.

We may then easily see that Φ has a model iff $\text{EXT}(\text{DB})$ has a minimal model in which $\bigvee \mathcal{K}$ is false. ■

B.4 Theorem. Given a cyclic state \mathcal{S} and a $\mathcal{C} \in \text{COMP}(\mathcal{A})$, the extension mechanisms detailed in Definitions 5.6.2 and 5.7.3 can be conducted in time that is polynomial in the size of $\text{EXT}(\text{DB})$ and \mathcal{L} .

Proof. Let $\sigma = \Sigma\{|C| : C \in \text{EXT}(\text{DB})\} + |\mathcal{L}|$. We show that both of the extension mechanisms can be performed in time that is polynomial in σ .

Definition 5.6.2. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a cyclic state, where each \mathcal{C}_i is int-total, and that $\mathcal{D} \in \text{COMP}(\{A_i | i \leq r\} \cup \bigcap_{i \leq r} \mathcal{C}_i^+)$. We need to pick a $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that for each $i \leq r$, $(\mathcal{E} - \mathcal{D}) \cap \mathcal{C}_i^- = \emptyset$, $\mathcal{E} \cap (\mathcal{D} - \mathcal{C}_i) \neq \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee(\mathcal{E} - \mathcal{D}) \vee \bigvee(\mathcal{C}_i^+)_{ext}$.

This requires a linear search of $\text{EXT}(\text{DB})$, within which we have an iteration for each $i \leq r$ (note that $r \leq |\mathcal{L}|$). The test $\text{EXT}(\text{DB}) \not\models \bigvee(\mathcal{E} - \mathcal{D}) \vee \bigvee(\mathcal{C}_i^+)_{ext}$, then requires that no rule $\bigvee \mathcal{G} \in \text{EXT}(\text{DB})$ subsumes $\bigvee(\mathcal{E} - \mathcal{D}) \vee \bigvee(\mathcal{C}_i^+)_{ext}$ (thus requiring a further iteration through $\text{EXT}(\text{DB})$).

Definition 5.7.3. Suppose that $\mathcal{S} = ((A_i, \mathcal{C}_i) | i \leq r)$ is a verified cyclic state, where each \mathcal{C}_i is int-total and $\mathcal{C} \in \text{COMP}(\{A_i | i \leq r\})$. There are two cases:

- (a) In this case we pick a predicate $A_{r+1} \in \text{INT}(\mathcal{L}) \cap \mathcal{C}^- \cap \bigcap_{i \leq r} \mathcal{C}_i^+$. Clearly this check can be performed in polynomial time (in σ).
- (b) In this case we pick a predicate $A_{r+1} \in \text{EXT}(\mathcal{L}) - (\mathcal{C}^+ \cup \bigcup_{i \leq r} \mathcal{C}_i^-)$ such that
 - (i) for each $i \leq r$, $\text{EXT}(\text{DB}) \not\models A_{r+1} \vee \bigvee(\mathcal{C}_i^+)_{ext}$, and
 - (ii) there is a rule $\bigvee \mathcal{E} \in \text{EXT}(\text{DB})$ such that $A_{r+1} \in \mathcal{E}$, $(\mathcal{E} - \{A_{r+1}\}) \cap \mathcal{C}_{ext}^- = \emptyset$ and $\text{EXT}(\text{DB}) \not\models \bigvee(\mathcal{E} - \{A_{r+1}\}) \vee \bigvee \mathcal{C}_{ext}^+$.

As above, each of these tests requires an iteration through each proposition in \mathcal{L} , each $i \leq r$ or each clause in $\text{EXT}(\text{DB})$. The search can therefore be performed in time that is polynomial in σ . ■

Theorem B.3 (and the preceding comment) indicate that the computational impact of compilation is to reduce stable model reasoning in DB to minimal model reasoning in $\text{EXT}(\text{DB})$. It is clear that the latter is a proper subset of the former (since a stable model must be a minimal model of $\text{EXT}(\text{DB})$), and moreover that the real complexities of the former lie in the intensional database. It should therefore be of no surprise that we see a real reduction in the theoretical computational complexity.

We can see this reduction in complexity in other problems too. For example, as above, $\bigvee_{i \leq n} A_i$ is a minimal answer in DB iff $\text{DB} \models \bigvee_{i \leq n} A_i$, and for each $j \leq n$, $\text{DB} \not\models \bigvee_{i \leq n, i \neq j} A_i$. Following compilation these problems are reduced to the first level of the polynomial time hierarchy; for example $\text{DB} \not\models \bigvee \mathcal{A}$ iff for some $i \leq n$, $\mathcal{C}_i \cup \mathcal{A}$ is consistent and $\text{EXT}(\text{DB})$ has a minimal model in which $\bigvee \mathcal{A}_{ext} \vee \bigvee(\mathcal{C}_i)_{ext}$ is false (and we have already seen in Theorem B.3 that this problem is NP-complete).

Similarly, for example, we can show that testing stable model membership is reduced to NP-completeness: given DB and proposition P , P belongs to a stable model of DB iff there is a $\mathcal{C}_i \in \text{COMP}$ such that $\mathcal{C}_i \cup \{\neg P\}$ is consistent and a minimal model of $\text{EXT}(\text{DB})$ in which $\bigvee(\mathcal{C}_i \cup \{\neg P\})_{ext}$ is false.

Appendix C: Proofs

In this section we present the proofs of the following theorems.

2.4(c) Theorem. If \mathcal{T} is a cyclic tree in DB and M is any model of DB with $M \cap (\mathcal{O}(\mathcal{T}) \cup \mathcal{N}(\mathcal{T})) = \emptyset$, then $Pred(\mathcal{T}) \subseteq M$.

Proof. Suppose that $Pred(\mathcal{T}) \not\subseteq M$. Firstly pick a predicate node m for which $lab(m) \notin M$ and such that for each predicate node $m^* > m$, $lab(m^*) \in M$. Next traverse the tree downwards from m , at each stage picking predicate nodes whose label is not in M . Eventually we must reach a predicate node $n \leq m$ for which $lab(n) \notin M$, and whose child, $rn_{\mathbf{R}}$ has child nodes all of which are labelled with a predicate in M , i.e., $antec(\mathbf{R}) \subseteq M$. (Ultimately the rule nodes at the leaves must have this property since they have no child nodes.) But then no predicate node lying strictly above m can have the same label as n , hence $CYC(n)$ is a subset of the predicates lying between m and n , and in particular $CYC(n)$ is disjoint from M . But now $conseq(\mathbf{R}) = (conseq(\mathbf{R}) - CYC(n)) \cup (conseq(\mathbf{R}) \cap CYC(n)) \subseteq \mathcal{O}(rn_{\mathbf{R}}) \cup CYC(n) \subseteq \mathcal{O}(\mathcal{T}) \cup CYC(n) \subseteq \mathcal{L} - M$. Since $antec(\mathbf{R}) \subseteq M \subseteq \mathcal{L} - \mathcal{N}(\mathcal{T}) \subseteq \mathcal{L} - \mathcal{N}(\mathbf{R})$, this contradicts the fact that $M \models \text{DB}$. ■

2.12 Theorem. Let \mathcal{D} be a cyclic strong cover in DB and $\mathcal{D}^- \subseteq M \subseteq \mathcal{L} - \mathcal{D}^+$. Then M is a disjunctive stable model of DB iff $M - \mathcal{D}^-$ is a disjunctive stable model of $\text{DB}_{\mathcal{D}}$.

Proof. Let $N = M - \mathcal{D}^-$, then $M = N \cup \mathcal{D}^-$.

(\rightarrow). We first show that $N \models \text{DB}_{\mathcal{D}}|_g N$. Suppose that $\mathbf{R} \in \text{DB}, \mathbf{R}_{\mathcal{D}} \neq \text{TRUE}$ with $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap N = \emptyset$ and $antec(\mathbf{R}_{\mathcal{D}}) \subseteq N$. Then $\mathcal{N}(\mathbf{R}) \cap M \subseteq (\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^+) \cap M = \mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap M = \mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap N = \emptyset$ (since $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap \mathcal{D}^- = \emptyset$). Also, $antec(\mathbf{R}) \subseteq antec(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^- \subseteq N \cup \mathcal{D}^- = M$. Thus $\emptyset \neq conseq(\mathbf{R}) \cap M \subseteq (conseq(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^+) \cap M = conseq(\mathbf{R}_{\mathcal{D}}) \cap N$ (since $M \cap \mathcal{D}^+ = \emptyset$ and $conseq(\mathbf{R}_{\mathcal{D}}) \cap \mathcal{D}^- = \emptyset$).

Suppose that $N' \subset N$ with $N' \models \text{DB}_{\mathcal{D}}|_g N$. We show that $\mathcal{D}^- \cup N' \models \text{DB}|_g M$, thus contradicting the minimality of M . Let $\mathbf{R} \in \text{DB}$ with $\mathcal{N}(\mathbf{R}) \cap M = \emptyset$, $antec(\mathbf{R}) \subseteq \mathcal{D}^- \cup N'$ and $conseq(\mathbf{R}) \cap \mathcal{D}^- = \emptyset$.

Since $\mathcal{N}(\mathbf{R}) \cap \mathcal{D}^- = \emptyset$ and $antec(\mathbf{R}) \cap \mathcal{D}^+ = \emptyset$ we have that $\mathbf{R}_{\mathcal{D}} \neq \text{TRUE}$, and $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap N \subseteq \mathcal{N}(\mathbf{R}) \cap M = \emptyset$, thus $pos(\mathbf{R}_{\mathcal{D}}) \in \text{DB}_{\mathcal{D}}|_g N$ with $antec(\mathbf{R}_{\mathcal{D}}) = antec(\mathbf{R}) - \mathcal{D}^- \subseteq N'$. Since $N' \models \text{DB}_{\mathcal{D}}|_g N$ we then have that $\emptyset \neq conseq(\mathbf{R}_{\mathcal{D}}) \cap N' \subseteq conseq(\mathbf{R}) \cap N'$.

(\leftarrow). We first show that $M \models \text{DB}|_g M$. Suppose that $\mathbf{R} \in \text{DB}$ with $\mathcal{N}(\mathbf{R}) \cap M = \emptyset$,

$\text{antec}(\mathbf{R}) \subseteq M$, and $\text{conseq}(\mathbf{R}) \cap M = \emptyset$. Since $\mathcal{D}^- \subseteq M \subseteq \mathcal{L} - \mathcal{D}^+$ we then have that $\mathbf{R}_{\mathcal{D}} \not\equiv \text{TRUE}$, with $\text{antec}(\mathbf{R}_{\mathcal{D}}) = \text{antec}(\mathbf{R}) - \mathcal{D}^- \subseteq N$ and $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap N \subseteq \mathcal{N}(\mathbf{R}) \cap M = \emptyset$. Since $N \models \text{DB}_{\mathcal{D}}|_{\mathcal{G}}N$, we have that $\emptyset \neq \text{conseq}(\mathbf{R}_{\mathcal{D}}) \cap N \subseteq \text{conseq}(\mathbf{R}) \cap M$.

Suppose that $M' \subset M$ with $M' \models \text{DB}|_{\mathcal{G}}M$. Now $M \subseteq \mathcal{L} - \mathcal{D}^+$, hence $\text{DB}|_{\mathcal{G}}(\mathcal{L} - \mathcal{D}^+) \subseteq \text{DB}|_{\mathcal{G}}M$ and $M' \models \text{DB}|_{\mathcal{G}}(\mathcal{L} - \mathcal{D}^+) \wedge \neg \bigvee \mathcal{D}^+$. Thus $\mathcal{D}^- \subseteq M'$ (by the remark following Definition 2.5), and in particular $M' - \mathcal{D}^- \subset N$.

We show that $M' - \mathcal{D}^- \models \text{DB}_{\mathcal{D}}|_{\mathcal{G}}N$ (thus contradicting the minimality of N). Suppose that $\mathbf{R} \in \text{DB}$, $\mathbf{R}_{\mathcal{D}} \not\equiv \text{TRUE}$, $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap N = \emptyset$ and $\text{antec}(\mathbf{R}_{\mathcal{D}}) \subseteq M' - \mathcal{D}^-$. But then $\text{antec}(\mathbf{R}) \subseteq \text{antec}(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^- \subseteq M'$ and $\mathcal{N}(\mathbf{R}) \cap M = (\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^+) \cap (N \cup \mathcal{D}^-) = \emptyset$ (since $\mathcal{N}(\mathbf{R}_{\mathcal{D}}) \cap \mathcal{D}^- = \emptyset = N \cap \mathcal{D}^+$). Since $M' \models \text{DB}|_{\mathcal{G}}M$ we must then have that $\emptyset \neq \text{conseq}(\mathbf{R}) \cap M' \subseteq (\text{conseq}(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^+) \cap [(M' - \mathcal{D}^-) \cup \mathcal{D}^-] = \text{conseq}(\mathbf{R}_{\mathcal{D}}) \cap (M' - \mathcal{D}^-)$ (since $\text{conseq}(\mathbf{R}_{\mathcal{D}}) \cap \mathcal{D}^- = \emptyset$ and $M' \cap \mathcal{D}^+ = \emptyset$). ■

2.13 Corollary. Suppose that \mathcal{D} is a cyclic strong cover in DB and \mathcal{G} is a consistent set of literals with $\mathcal{G} \supseteq \mathcal{D}$.

(a) \mathcal{G} is a strong cover in DB iff $\mathcal{G} - \mathcal{D}$ is a strong cover in $\text{DB}_{\mathcal{D}}$.

(b) \mathcal{G} is a cyclic strong cover in DB iff $\mathcal{G} - \mathcal{D}$ is a cyclic strong cover in $\text{DB}_{\mathcal{D}}$.

Proof (a). Suppose that \mathcal{G} is a strong cover in DB, and that $\mathbf{R}_{\mathcal{D}} \not\equiv \text{TRUE}$ with $\text{conseq}(\mathbf{R}_{\mathcal{D}}) \subseteq \mathcal{G} - \mathcal{D}$, then $\text{conseq}(\mathbf{R}) \subseteq \text{conseq}(\mathbf{R}_{\mathcal{D}}) \cup \mathcal{D}^+ \subseteq \mathcal{G}$. Suppose that $A \in \text{antec}(\mathbf{R}) \cap \mathcal{G}$, then $A \notin \mathcal{D}^-$, and since $\text{antec}(\mathbf{R}) \cap \mathcal{D}^+ = \emptyset$, we have that $A \in (\mathcal{G} - \mathcal{D}) \cap \text{antec}(\mathbf{R}_{\mathcal{D}})$. If $A \in \mathcal{N}(\mathbf{R}) \cap \mathcal{G}^-$, then $A \notin \mathcal{D}^+$, and since $\mathcal{N}(\mathbf{R}) \cap \mathcal{D}^- = \emptyset$, we have that $A \in (\mathcal{G} - \mathcal{D})^- \cap \mathcal{N}(\mathbf{R}_{\mathcal{D}})$.

Conversely suppose $\mathcal{G} - \mathcal{D}$ is a strong cover in $\text{DB}_{\mathcal{D}}$, and that $\mathbf{R} \in \text{DB}$ with $\text{conseq}(\mathbf{R}) \subseteq \mathcal{G}$. Firstly note that $\text{conseq}(\mathbf{R}) \cap \mathcal{D}^- = \emptyset$. Suppose that $\mathbf{R}_{\mathcal{D}} \equiv \text{TRUE}$, then either $\text{antec}(\mathbf{R}) \cap \mathcal{D}^+ \neq \emptyset$ or $\mathcal{N}(\mathbf{R}) \cap \mathcal{D}^- \neq \emptyset$. Thus the body of \mathbf{R} intersects \mathcal{D} and hence \mathcal{G} .

Suppose that $\mathbf{R}_{\mathcal{D}} \not\equiv \text{TRUE}$, then $\text{conseq}(\mathbf{R}_{\mathcal{D}}) \subseteq \mathcal{G} - \mathcal{D}$. But then there is a literal in the body of $\mathbf{R}_{\mathcal{D}}$ which is contained in $\mathcal{G} - \mathcal{D}$, and hence a literal in the body of \mathbf{R} which is contained in \mathcal{G} .

(b) By Theorem 2.10(b), \mathcal{G} is cyclic in DB iff \mathcal{G}^- is a disjunctive stable model in DB/\mathcal{G} . Now \mathcal{D} is a cyclic strong cover in DB/\mathcal{G} , thus (by Theorem 2.12) \mathcal{G}^- is a disjunctive stable model in DB/\mathcal{G} iff $\mathcal{G}^- - \mathcal{D}^- = (\mathcal{G} - \mathcal{D})^-$ is a disjunctive stable model of $(\text{DB}/\mathcal{G})_{\mathcal{D}} = \text{DB}_{\mathcal{D}}/(\mathcal{G} - \mathcal{D})$. By Theorem 2.10(b) again, this is the case iff $\mathcal{G} - \mathcal{D}$ is cyclic in $\text{DB}_{\mathcal{D}}$. ■

Appendix D: Constructing int-total weakly cyclic covers from disjunctive stable models.

Suppose that we are given $\text{INT}(\text{DB})$. For each rule $\mathbf{R} \in \text{INT}(\text{DB})$ let $P_{\mathbf{R}}, P_{\mathbf{R}}^*$ be two new predicates not appearing in \mathcal{L} . If \mathbf{R} contains one or more extensional atoms in its body, let \mathbf{R}^* be formed from \mathbf{R} by replacing these by $P_{\mathbf{R}}$ (else define $\mathbf{R}^* = \mathbf{R}$). Let $\text{DB}^* = \{\mathbf{R}^* \mid \mathbf{R} \in \text{INT}(\text{DB})\} \cup \{P_{\mathbf{R}} \vee P_{\mathbf{R}}^* \mid \mathbf{R} \in \text{INT}(\text{DB})\}$.

If \mathcal{C} is a total cyclic strong cover in DB^* , let \mathcal{C}^* be formed from \mathcal{C} by (i) replacing every occurrence of $\neg P_{\mathbf{R}}$ by $\{\neg K \mid K \text{ is an extension atom in the body of } \mathbf{R}\}$, and (ii) removing any atom of the form $P_{\mathbf{S}}^*, \neg P_{\mathbf{S}}^*, P_{\mathbf{S}}$. Notice that \mathcal{C} and \mathcal{C}^* are int-total (in \mathcal{L}).

Let $\mathbb{W} = \{\mathcal{C}^* \mid \mathcal{C} \text{ is a total cyclic strong cover in } \text{DB}^*\}$, then we have the following theorem.

D.1 Theorem (a). If $\mathcal{C}^* \in \mathbb{W}$ then any strong cover of \mathcal{C}^* in $\text{INT}(\text{DB})$ is an int-total weakly cyclic cover in $\text{INT}(\text{DB})$.

(b) If \mathcal{G} is an int-total weakly cyclic cover in $\text{INT}(\text{DB})$, then there is a $\mathcal{C}^* \in \mathbb{W}$ and a strong cover \mathcal{D} of \mathcal{C}^* in $\text{INT}(\text{DB})$ such that $\mathcal{D} \subseteq \mathcal{G}$.

Proof (Sketch) (a). Let \mathcal{D} be a strong cover of \mathcal{C}^* in $\text{INT}(\text{DB})$. Firstly note that since \mathcal{C}^* is already int-total and \mathcal{D} is consistent, we must have that $\mathcal{D} - \mathcal{C}$ contains only extensional atoms.

Let $P \in \text{INT}(\mathcal{L}) \cap \mathcal{D}^-$, then $P \in \mathcal{C}^-$, hence there is a cyclic tree \mathcal{T}^* for P in DB^* such that $\mathcal{S}(\mathcal{T}^*) \subseteq \mathcal{C}$. If we delete every leaf node, and then replace each remaining rule node $rn_{\mathbf{R}^*}$ by $rn_{\mathbf{R}}$ (modifying the child nodes appropriately), we may form a partial cyclic tree \mathcal{T} for P in DB such that $\mathcal{S}(\mathcal{T}) \subseteq \mathcal{C}^* \subseteq \mathcal{D}$. The result then follows from Proposition 5.3(d).

(b) Let $\{\mathcal{T}_i \mid i \leq k\}$ be partial cyclic trees in $\text{INT}(\text{DB})$ such that $\mathcal{G}^- = \bigcup_{i \leq k} \text{Pred}(\mathcal{T}_i)$ and $\mathcal{G}^+ \supseteq \bigcup_{i \leq k} (\mathcal{O}(\mathcal{T}_i) \cup \mathcal{N}(\mathcal{T}_i))$.

If \mathcal{T}_i^* is formed from \mathcal{T}_i by replacing each rule node $rn_{\mathbf{R}}$ by $rn_{\mathbf{R}^*}$ (modifying the child nodes appropriately), then $\{\mathcal{T}_i^* \mid i \leq k\}$ witnesses that $\mathcal{C} = \mathcal{G}_{\text{int}} \cup \bigcup \{\{\neg P_{\mathbf{R}}, P_{\mathbf{R}}^*\} \mid \mathbf{R} \text{ labels a rule node in some } \mathcal{T}_i\} \cup \bigcup \{\{P_{\mathbf{R}}, \neg P_{\mathbf{R}}^*\} \mid \mathbf{R} \text{ does not label a rule node in some } \mathcal{T}_i\}$ is weakly cyclic in DB^* . Clearly \mathcal{C} is its own completion in DB^* , and is easily seen to be a strong cover (in DB^*). Finally $\mathcal{C}^* \subseteq \mathcal{G}$, and we may therefore take $\mathcal{D} = \mathcal{G}$. ■