Journal of Automated Reasoning **15:** 95–165, 1995.

A New Methodology for Query Answering in Default Logics via Structure-Oriented Theorem Proving

T. SCHAUB
IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France
e-mail: torsten@irisa.fr

(Received: 30 March 1994; in final form: 1 September 1994)

Abstract. We present a new approach to query answering in default logics. The basic idea is to treat default rules as classical implications along with some qualifying conditions restricting the use of such rules while query answering. We accomplish this by taking advantage of the conception of structure-oriented theorem proving provided by Bibel's connection method. We show that the structure-sensitive nature of the connection method allows for an elegant characterization of proofs in default logic. After introducing our basic method for query answering in default logics, we present a corresponding algorithm and describe its implementation. Both the algorithm and its implementation are obtained by slightly modifying an existing algorithm and an existing implementation of the standard connection method. In turn, we give a couple of refinements of the basic method that lead to conceptually different algorithms. The approach turns out to be extraordinarily qualified for implementations by means of existing automated theorem proving techniques. We substantiate this claim by presenting implementations of the various algorithms along with some experimental analysis.
Even though our method has a general nature, we introduce it in the first part of this paper with the example of constrained default logic. This default logic is tantamount to a variant due to Brewka, and it coincides with Reiter's default logic and a variant due to Łukaszewicz on a large fragment of default logic. Accordingly, our exposition applies to these instances of default logic without any modifications.

Key words: default logics, query answering, credulous reasoning, theorem proving, connection method.

AMS Subject Classification: 68T15, 68T27

1. Introduction

Reasoning in the absence of complete information constitutes one of the most important facets of commonsense reasoning. This form of reasoning is frequently accomplished by making default assumptions or simply by *default reasoning*. A versatile approach to this is Reiter's default logic [30]. Since its introduction, it has proven to be extremely valuable for formalizing default reasoning in various domains. Among others, it has been applied to diagnosis [31], natural language [23], inheritance networks [16], terminological logics [1], and databases [8]. In particular, it provides semantics for truth maintenance systems [6] and diverse forms of logic programming [17]. Hence, default logic is very expressive and thus of theoretical importance. But expressiveness has its costs. Even though default logic captures many practical approaches, it is hardly implementable in full generality. The major cause for this is that regular default logic lacks several properties that are indispensable for reasonable proof procedures, as discussed in one of the following sections.

So far, this difficulty has been addressed in two different ways. First, it has led to algorithmic approaches dealing with restricted subclasses of default logic, which enjoy desirable computational properties [30, 2, 41]. Yet there are only few computational approaches to full-fledged default logic [19, 42]. Second, it has led to variants of default logic overcoming several shortcomings encountered in the original approach [22, 5, 11]. Although these variants are more easily 'implementable' in full generality, this line has been rarely pursued [32].

In this paper, we address the aforementioned difficulty from a strictly different point of view, namely, the one given by existing automated theorem provers for classical logic. Our approach is driven by the desire to obtain a simple yet powerful method for default theorem proving that is easily adaptable by existing implementations of automated theorem provers.

So, the key question is how classical theorem proving differs from default theorem proving. In default logic, classical logic is augmented by so-called default rules. These rules can be seen as rules of conjecture whose role is to augment an underlying incomplete first-order theory. They differ from standard inference rules in sanctioning inferences that rely upon given as well as absent information. Hence, a default rule $(\alpha : \beta)/\gamma$ has two types of antecedent: A *prerequisite* α, which is established if α is derivable, and a *justification* β, which is established if β is consistent in a certain way. If both conditions hold, the *consequent* γ is concluded by default. A set of conclusions sanctioned by a given set of default rules and by means of classical logic is called an *extension* of an initial set of facts.

Now, automated theorem provers handle classical logic extremely well. However, there are no means for dealing with default rules. Thus, the difference between classical theorem proving and default theorem proving rests on the notion of a default rule, like $(\alpha : \beta)/\gamma$. In contrast to such rules, automated theorem provers deal with classical implications, like $\alpha \rightarrow \gamma$, or their clausal form. Accordingly, the previously raised question reduces to the one upon the difference between implications and default rules. Roughtly speaking, this difference boils down to that between sentential operators and inference rules on the one hand, and an additional condition given by the consistency check on the other hand. The last two notions strongly affect the application and the use of default rules as opposed to classical implications.

As an example, consider the default rule $(A : \neg S)/E$, saying that adults (A) are typically employed (E) unless they are students (S), along with its sentential counterpart $A \rightarrow E$. Of course, given an adult A (and nothing else), both rules allow us to conclude E. However, given an unemployed person $\neg E$, the implication allows us to conclude $\neg A$ (by contraposition) while this is impossible with the default rule, since an inference rule cannot be applied in reverse order. Also, we can derive E from A and S with the implication $A \rightarrow E$ while this is not possible with the default rule, since its justification $\neg S$ is inconsistent with the premises.

The basic idea of our approach is the following one. To allow for default theorem proving based on classical automated theorem provers, we treat default rules as classical implications along with some qualifying conditions restricting the use of such rules. In concrete terms, this leads to two restrictions on classical proofs: First, we restrict admissible proofs to those that are structured in a certain way in order to account for the concept of an inference rule. Second, we impose a condition of proofs ensuring the compatible use of default rules preserving their consistency conditions.

In what follows, we develop a new approach to theorem proving in default logics based on the connection method [3]. We have chosen this method because it relies on analyzing the structure of formulas and thus allows for structure-oriented theorem proving. Unlike resolution-based methods that decompose formulas in order to derive a contradiction, the connection method analyzes the structure of formulas for proving their unsatisfiability. This structure-sensitive nature allows for an elegant characterization of the two aforementioned restrictions on classical proofs. As a consequence, we obtain a homogeneous characterization of default proofs at the level of the calculus.

In general, there are two approaches to query answering in default logics. In the *credulous* approach, we accept a query if it belongs to one extension of a considered default theory, whereas in the *skeptical* approach, we accept a query if it belongs to all extension of the default theory. In the sequel, we exclusively deal with the more basic approach, namely, credulous default reasoning. The given approach is extended to skeptical reasoning in [45, 40].

Even though our method has a general nature, we introduce it in this paper with the example of constrained default logic [10, 35, 11].[1] Afterwards, we discuss in turn how our approach applies to other variants of default logic. For a complement, we detail in [39] how our method applies to a prioritized version of default logic, recently proposed by Brewka in [7]. Our initial exemplar, constrained default logic, enjoys several desirable computational properties needed for reasonable proof procedures. Moreover, it has recently been shown in [12] that in certain fragments of constrained default logic reasoning is significantly easier than in Reiter's default logic – even though general goal-directed reasoning remains exponential. All this renders our exemplar a prime candidate for computational purposes. In general, however, credulous reasoning is Σ_2^P-complete, while skeptical reasoning is Π_2^P-complete [18].

The paper is organized as follows. After some formal preliminaries accounting for default logics in Section 2, we smooth the way for our approach by providing computational characterizations of extensions, queries, and default theories in Section 3. We introduce our basic method for query answering in default logics in Section 4. In the subsequent section, we present a corresponding algorithm and sketch an existing implementation obtained by carefully modifying an existing connection method theorem prover. This endeavor is driven by our initial desire to obtain a simple yet powerful method for default theorem proving which is easily adaptable by existing implementations of automated theorem provers. To this end, the latter implementation provides a case study in how far an existing theorem prover for the connection method has to be modified in order to allow for query answering in default logics.

We introduce in Section 6 an equivalent but conceptually different approach to query answering in default logics. This results in an algorithm that is orthogonal to the one introduced in the first part of the paper. Section 7 gives an intermediate summary and contrasts our approach with other computational approaches to default logic. We provide prototypical implementations of the different versions of our approach in Section 8. These prototypes provide us with some experimental results as well as some implementation techniques needed for implementing our approach. Section 9 describes several enhancements and extensions of our approach. Among others, we show how our approach applies to other variants of default logic and how it can be enriched by lemma handling. The proofs of all subsequent theorems are given in the Appendix.

2. Default Logics

This section gives some basic definitions dealing with default logic. Since our approach is initially applied to constrained default logic, most of the formal preliminaries account for this variant of default logic. However, we will try to be as general as possible and indicate each special reference to this specific variant. Nevertheless, constrained default logic coincides with other default logics, like Reiter's [30] and Łukaszewicz' [22], on the fragment of so-called normal default theories (see below). Also, it is tantamount to a variant due to Brewka [5] when neglecting representational issues (see [37, 11] for details). Consequently, the following exposition applies to these instances of default logic as well. We discuss the adaptation of our approach to the latter variants of default logic in Section 9.

In what follows, we deal with a propositional language $\mathcal{L}_\Sigma$ over a finite alphabet Σ. Arguably, the restriction to a decidable logic is a necessary one. Otherwise the resulting system would not even be semi-decidable, given the reference to consistency while deriving formulas in default logic (cf. [30]).

As mentioned in the introduction, the central concepts in default logic are default rules along with their induced extensions of an initial set of facts. In default logics, knowledge is represented by *default theories* (D, W) consisting of a consistent[2] set of formulas W and a set of default rules D. A *normal default* *theory* is restricted to *normal default rules* whose justification in equivalent to the consequent. In any default logic, default rules induce one or more extensions of an initial set of facts: Given a set of facts W and a set of default rules D, any such extension E is a deductively closed set of formulas containing W such that, for any $(\alpha : \beta)/\gamma \in D$, if $\alpha \in E$ and $\neg\beta \notin E$, then $\gamma \in E$.

Now, let us make all this more precise and look at our exemplary variant, constrained default logic [11]. This variant enjoys several desirable computational properties that are given only for restricted default theories in Reiter's default logic. One such desirable property is the *existence of extensions*. Another even more important property for query answering is that of *semi-monotonicity*, since it allows us to restrict our attention to default rules relevant for proving a query.[3] Moreover, semi-monotonicity implies the existence of extensions.

In constrained default logic, an extension, E, comes with an underlying set of constraints, C, which is used for accumulating the set of justifications of the applied default rules. Formally, this amounts to the usual fixed-point definition given for extensions in default logics:

DEFINITION 2.1. Let (D, W) be a default theory. For any set of formulas T let $\Upsilon(T)$ be the pair of smallest sets of formulas (S', T') such that

(1) $W \subseteq S' \subseteq T'$,

(2) $S' = \mathrm{Th}(S')$ and $T' = \mathrm{Th}(T')$,

(3) For any $(\alpha : \beta)/\gamma \in D$, if $\alpha \in S'$ and $T \cup \{\beta\} \cup \{\gamma\} \nvdash \bot$ then $\gamma \in S'$ and $\beta \wedge \gamma \in T'$.

A pair of sets of formulas (E, C) is a constrained extension of (D, W) iff $\Upsilon(C) = (E, C)$.

As an example, consider the statements ‘students are typically adults’, ‘adults usually drive a car’, and ‘adults are typically employed unless they are students’, along with a student S. The corresponding default theory is the following one.

$$\left(\left\{\frac{S : A}{A}, \frac{A : C}{C}, \frac{A : \neg S}{E}\right\}, \{S\}\right). \tag{2.1}$$

In both Reiter's and constrained default logic, this default theory yields a unique extension $\mathrm{Th}(\{S, A, C\})$ in which a student is an adult driving a car. In this simple example, the constraints in constrained default logic coincide with the actual extension. It is instructive to verify that the constraints differ from the extension obtained when substituting the fact S by A. In this case, we obtain in both default logics the extension $\mathrm{Th}(\{A, C, E\})$, which is supplemented with constraints, $\mathrm{Th}(\{A, C, E, \neg S\})$, in constrained default logic. Apart from supplementing constraints, the difference between both approaches rests on the different interpretation of consistency.[4] In Reiter's approach, the consistency of a justification β is checked wrt the extension E by $\neg\beta \notin E$, whereas in constrained default logic the same is done wrt the constraints C by $\neg(\beta \wedge \gamma) \notin C$ (where γ is the consequent of the considered default rule).[5] The former condition ensures that each justification β is individually consistent with a final extension, while the latter enforces the joint consistency of all justifications (of all applying default rules) with a final extension.

3. The Fundamental Basis

In this section, we provide the fundamental basis for our approach to query answering in default logics. To this end, let us first turn to the two features distinguishing default rules from classical implications, namely, the character of an inference rule and the additional consistency check. While the latter is handled in the usual manner by testing satisfiability, the former needs a more subtle treatment. In fact, the character of an inference rule can be captured by the notion of *groundedness*.[6] This fundamental concept is common to all existing default logics. We call a set of default rules D *grounded* in a set of facts W iff there exists an enumeration $\langle \delta_i \rangle_{i \in I}$ of D such that for $i \in I$,

$$W \cup \text{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \text{Prereq}(\delta_i). \tag{3.2}$$

For convenience, we denote the prerequisite of a default rule δ by Prereq(δ), its justification by Justif(δ), and its consequent by Conseq(δ).[7]

In particular, each set of default rules 'generating' an extension is grounded in the set of facts.[8] In the above example (2.1), the extension Th($\{S, A, C\}$) is generated by the first two default rules. This results in the enumeration $\langle (S : A)/A, (A : C)/C \rangle$, whose defaults are obviously grounded in $\{S\}$. In general, groundedness distinguishes default rules from classical implications. For instance, the above default rule $(A : \neg S)/E$ is (trivially) not grounded in the set of facts $\{\neg E\}$ so that reasoning by contraposition becomes impossible. That is, $\neg A$ is not derivable from $\neg E$. Moreover, groundedness prevents circular chains of reasoning. Consider the default rules $(A : C)/C$ and $(C : A)/A$ and no facts. In this case, neither A nor C is derivable since there is no nonempty grounded sequence of default rules.

So, from the perspective of the introductory section, groundedness and consistency constitute the two qualifying conditions for the application and the use of default rules. In particular, these two notions allow for characterizing extensions in a considerably simpler way. As a first result, we obtain a nonfixed point characterization of constrained extensions, which is indispensable for computational purposes:

THEOREM 3.1. *Let (D, W) be a default theory, and let E and C be sets of formulas. Then, (E, C) is a constrained extension of (D, W) iff*

$$\begin{aligned} E &= \text{Th}(W \cup \text{Conseq}(D')), \\ C &= \text{Th}(W \cup \text{Justif}(D') \cup \text{Conseq}(D')) \end{aligned}$$

for a maximal $D' \subseteq D$ such that D' is grounded in W and $W \cup$ Justif(D') $\cup$ Conseq(D') is consistent.

That is, an extension is characterized as the deductive closure of the set of facts and the consequents of a maximal set of default rules which is grounded and preserves consistency. Accordingly, the computation of a constrained extension boils down to classical deduction along with enforcement of groundedness and consistency.

This suggests the following approach to query answering in default logics. To verify whether a formula φ is in some extension E of a default theory (D, W), we have to find a subset of D that allows for deriving φ and complies with the above requirements. As already noticed in [30], this can be accomplished in a reasonable way only if we can confine ourselves to default rules relevant for deriving φ. The formal counterpart of this observation is given by the property of *semi-monotonicity* – on which also our approach relies. Formally, semi-monotonicity stipulates that if $D' \subseteq D$ for two sets of default rules, then if E' is an extension of (D', W), there is an extension E of (D, W) such that $E' \subseteq E$. Given this property, it is sufficient to consider a relevant subset of default rules while answering a query, since applying other default rules would only enlarge or preserve the partial extension at hand.

Semi-monotonicity holds only for restricted fragments of Reiter's default logic, whereas it is enjoyed by constrained default logic in its full generality. This is one of the reason we have chosen constrained default logic as an illustration of our method.

Anyway, this property leads us to the following corollary to Theorem 3.1 providing a formal characterization of query answering in constrained default logic.

COROLLARY 3.2. *Let (D, W) be a default theory. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff*

$$W \cup \text{Conseq}(D') \vdash \varphi$$

for some $D' \subseteq D$ such that D' is grounded in W and $W \cup \text{Justif}(D') \cup \text{Conseq}(D')$ is consistent.

That is, for verifying whether φ is in some extension of a default theory (D, W), it is enough to determine a grounded and consistent set of default rules $D' \subseteq D$ that allows for proving φ from the facts in W and all default rules in D'.

Theorem 3.1 and Corollary 3.2 provide the fundamental basis for our approach to query answering in (constrained) default logic. They are strongly rooted in the basic concepts of groundedness and consistency. Observe that in both specifications the latter concepts constitute rather separate constraints on the default rules under consideration. We will stepwisely refine this approach in the two following sections. In fact, Section 5 strongly relies on the possibility of separating these concepts for implementing our approach by using existing automated theorem provers.

Another salient feature of the previous specifications is the formation of sequences of default rules. This will come to the fore more and more in the subsequent sections, in particular, in Section 6, where we provide an alternative approach by meshing together the concepts of groundedness and consistency for forming sequences of default rules. Moreover, we have shown in [39] that such a combination is very useful for implementing priorities.

We now turn to the issue of default theorem proving using conventional theorem provers. As argued in the introductory section, classical theorem provers cannot deal with default rules but conventional clauses only. As a first step, we thus shift information from the default part into the classical part of a default theory in order to facilitate the treatment of default theories. To this end, we transform default theories by substituting default rules by so-called atomic default rules consisting of new atomic propositions and by extending the facts with a set of implications relating these propositions to the constituents of the original default rules: For a default theory (D, W) in $\mathcal{L}_\Sigma$, let $\mathcal{L}_{\Sigma'}$ be the language obtained by adding the new propositions $\alpha_\delta, \beta_\delta, \gamma_\delta$ for each $\delta \in D$. The function τ maps a default theory (D, W) in $\mathcal{L}_\Sigma$ into a default theory (D', W') in $\mathcal{L}_{\Sigma'}$, where

$$D' = \left\{ \frac{\alpha_\delta : \beta_\delta}{\gamma_\delta} \,\middle|\, \delta \in D \right\},$$

$$W' = W \cup \{\text{Prereq}(\delta) \rightarrow \alpha_\delta, \beta_\delta \rightarrow \text{Justif}(\delta), \gamma_\delta \rightarrow \text{Conseq}(\delta) \mid \delta \in D\}.$$

The resulting default theory (D', W') is called the *atomic format* of the original default theory (D, W). That is, (D', W') contains only atomic default rules.

Consider the default rule $(S : A)/A$ (for short δ_1) in default theory (2.1). Applying τ to this theory yields for δ_1 the default rule $(S_{\delta_1} : A_{\delta_1})/A_{\delta_1}$ (where S_{δ_1} and A_{δ_1} are new propositional letters)[9] along with the implications $S \rightarrow S_{\delta_1}$ and $A_{\delta_1} \rightarrow A$.

The transformation of default theories into their atomic format does not affect the computation of queries to the original default theory, as shown in [33].[10]

THEOREM 3.3 (33). *Let (D, W) be a default theory in $\mathcal{L}_\Sigma$. Let E, C be sets of formulas in $\mathcal{L}_\Sigma$ and E', C' be sets of formulas in $\mathcal{L}_{\Sigma'}$ such that $E = E' \cap \mathcal{L}_\Sigma$ and $C = C' \cap \mathcal{L}_\Sigma$. Then, (E, C) is a constrained extension of (D, W) iff (E', C') is a constrained extension of $\tau(D, W)$.*

The major advantage of atomic default rules over arbitrary ones is that the constituents of default rules are not spread over several clauses while transforming them into clausal format. Rather, each atomic default rule can be represented as a single binary clause, as we will see in the next section. Strictly speaking, this is not absolutely necessary but it simplifies matters dramatically. This concerns the formal presentation of the approach and moreover its implementation by existing automated theorem provers. With the above transformation, we can (and will) therefore confine ourselves to default theories with atomic default rules only (without losing generality).

4. A Method for Query Answering in Default Logics

In this section, we develop a method for query answering in default logics based on the connection method [3]. The connection method allows for testing the unsatisfiability of formulas in conjunctive normal form (CNF). Unlike resolution-based methods that decompose formulas in order to derive a contradiction, the connection method analyzes the structure of formulas for proving their unsatisfiability. This structure-sensitive nature allows for an elegant characterization of proofs in default logic, as we will see below.

4.1. The connection method

In the connection method, formulas in CNF are displayed two-dimensionally in the form of *matrices*. A matrix is a set of sets of literals (literal occurrences, to be precise).[11] Such a matrix is given in (4.3) below. Each column of a matrix represents a *clause* of the CNF of the formula. In order to show that a sentence φ is entailed by a sentence W, we prove that $W \wedge \neg\varphi$ is unsatisfiable. In the connection method this is accomplished by path checking: A *path* through a matrix is a set of literals, one from each clause. A *connection* is an unordered pair of literals which are identical except for the negation sign (and possible indexes). A *mating* is a set of connections. A mating *spans* a matrix if each path through the matrix contains a connection from the mating. Finally, a formula, like $W \wedge \neg\varphi$, is unsatisfiable iff there is a spanning mating for its matrix.

Let us briefly illustrate this by verifying whether C is entailed by

$$S \wedge (S \rightarrow A) \wedge (A \rightarrow C).$$

For this, we prove that conjoining the negated query $\neg C$ to the latter formula yields an unsatisfiable formula. Transforming the resulting formula into its CNF yields

$$S \wedge (\neg S \vee A) \wedge (\neg A \vee C) \wedge \neg C$$

whose two-dimensional representation is the following one (by ignoring the arcs).

$$\left[\begin{tikzcd} & \neg S & \neg A & \neg C \\ S \arrow[ur, bend left, dash] & A \arrow[ur, bend left, dash] & C \arrow[ur, bend left, dash] & \end{tikzcd}\right] \qquad (4.3)$$

This matrix has a spanning mating whose connections are represented by arcs linking the respective literals. This is so because matrix (4.3) contains four paths, like $\{S, A, \neg A, \neg C\}$, all of which contain at least one connection, like $\{A, \neg A\}$. In this way, we have shown that C is entailed by $S \wedge (S \rightarrow A) \wedge (A \rightarrow C)$.

In the sequel, we sometimes refer to certain submatrices or supermatrices of a given matrix. We call a matrix M' a submatrix of a matrix M if M' is obtainable from M by deleting literals or even clauses in M. The definition of supermatrices is analogous. We say that a path is *complementary* or *closed* if it contains a connection from a given mating. Otherwise, we say that the path is *noncomplementary* or *open*. Finally, we call a matrix *complementary* if it has a spanning mating or, in other words, if each path through the matrix contains a connection from a mating at hand.

4.2. COMPLEMENTARITY

In this section, we describe how to turn default theories into matrices and how to verify the complementarity of the resulting matrices.

Our approach relies on the idea that a default rule can be decomposed into a classical implication along with two qualifying conditions, one accounting for the character of an inference rule and another one enforcing the respective consistency condition.[12] The computational counterparts of these qualifying conditions are given by the proof-oriented concepts of *admissibility* and *compatibility*, which we will introduce in the following two subsections.

To find out whether a formula φ is contained in some extension of a default theory (D, W), we proceed as follows. First, we transform the atomic default rules in D into their sentential counterparts. This yields a set of indexed implications

$$W_D = \left\{ \alpha_\delta \rightarrow \gamma_\delta \,\middle|\, \frac{\alpha_\delta : \beta_\delta}{\gamma_\delta} \in D \right\}.$$

In what follows, we adopt this notation and write $W_{D'} = \{\alpha_\delta \rightarrow \gamma_\delta | (\alpha_\delta : \beta_\delta)/\gamma_\delta \in D'\}$ for any subset D' of D. Second, we transform both W and W_D into their clausal forms, C_W and C_D. The clauses in C_D, like $\{\neg\alpha_\delta, \gamma_\delta\}$, are called *$\delta$-clauses*; all other clauses like those in C_W are refered to as *ω-clauses*. Now, we are ready for query answering. That is, a query φ is derivable from (D, W) iff there is a spanning mating for the matrix $C_W \cup C_D \cup \{\neg\varphi\}$ agreeing with the concepts of admissibility and compatibility.[13]

Consider our student example. The encoding of the set of default rules yields the following set, W_D, of implications:

$$\{S_{\delta_1} \rightarrow A_{\delta_1}, A_{\delta_2} \rightarrow C_{\delta_2}, A_{\delta_3} \rightarrow E_{\delta_3}\}.$$

The indexes denote the respective default rules in default theory (2.1) from left to right. In order to verify that a student drives a car, C, we first have to transform the fact S (in default theory (2.1)) and the implications in W_D into their clausal form. The resulting clauses are given two-dimensionally as the first four columns of the matrix in (4.4). The full matrix is obtained by adding the clause containing the negated query, $\neg C$. In fact, the matrix has a spanning mating, $\{\{S, \neg S_{\delta_1}\}, \{A_{\delta_1}, \neg A_{\delta_2}\}, C_{\delta_2}, \neg C\}\}$. As above, we have indicated these connections in (4.4) as arcs linking the respective literals.

$$\begin{tikzcd}
 & \neg S_{\delta_1} & \neg A_{\delta_2} & \neg A_{\delta_3} & \neg C \\
S \arrow[ur, no head, bend left] & A_{\delta_1} \arrow[ur, no head, bend left] & C_{\delta_2} \arrow[urr, no head, bend left] & E_{\delta_3} &
\end{tikzcd} \tag{4.4}$$

For simplicity, we have refrained from transforming default theory (2.1) into atomic format because it already consists of atomic formulas. In such a case, let us rather adopt the following two conventions. First, let us agree on simply labeling components of a default rule and allowing for connections between complementary literals having different indexes (if any at all). Second, let us assume that we can always distinguish between the prerequisite and the consequent in a *δ-clause*. Observe that both conventions are obsolete as soon as we enforce default theories in atomic format by transformation τ (cf. Section 2). First, we obtain in atomic format two standard connections, rather than a 'mixed' connection between an indexed and unindexed literal. For instance, instead of two clauses $\{S\}$ and $\{\neg S_{\delta_1}, A_{\delta_1}\}$ (from S, $(S : A)/A$) along with the 'mixed' connection $\{S, \neg S_{\delta_1}\}$, we would obtain three clauses $\{S\}, \{\neg S, S_{\delta_1}\}$, and $\{\neg S_{\delta_1}, A_{\delta_1}\}$ (from $S, S \rightarrow S_{\delta_1}, (S_{\delta_1} : A_{\delta_1})/A_{\delta_1}$) along with two standard connections $\{S, \neg S\}$ and $\{S_{\delta_1}, \neg S_{\delta_1}\}$. The same applies to the remaining clauses in matrix (4.4). Second, observe that in atomic format the distinction between prerequisites and consequents of δ-clause is trivial. This is so because the prerequisite is given by the negative literal in the δ-clause and the consequent by the positive literal. We support this in two-dimensional notation by stacking prerequisites over consequents.

The above matrix illustrates yet another point: Not all of the clauses are necessarily involved in providing a spanning mating for a matrix. A useful concept is then that of a *core* of a matrix M wrt a mating Π, which allows for isolating the clauses relevant to the underlying proof. We define the core of M wrt Π as follows.[14]

DEFINITION 4.1. Let Π be a mating for the matrix M. Then, we define the core of M wrt Π as

$$\kappa(M, \Pi) = \{c \in M \mid \exists \pi \in \Pi \,.\, c \cap \pi \neq \varnothing\}.$$

For instance, the core of the preceding matrix relative to the drawn mating is given by the first three and the last clauses.

So far, it might seem that classical theorem proving with ω- and δ-clauses suffices for querying default theories. To see that this is not enough, consider again the default rule $(A : \neg S)/E$ along with the fact $\neg E$. In default logics, there is no way to derive $\neg A$. However, the resulting matrix, given in (4.5), has a spanning mating, which amounts to deriving $\neg A$ by contraposition.

$$\left[\begin{tikzcd} & \neg A_{\delta_3} \arrow[r, dash, bend left] & A \\ \neg E \arrow[r, dash, bend left] & E_{\delta_3} & \end{tikzcd}\right] \tag{4.5}$$

This example shows that pure deduction with δ-clauses cannot account for the inference rule character of the original default rules.

4.3. ADMISSIBILITY

In default logics, the nature of an inference rule is reflected by the property of groundedness, which relies on forming sequences of default rules. In fact, the connection method allows for imposing a similar restriction on the clausal counterparts of default rules. This leads us to our first qualifying condition on proofs given by the concept of admissibility.

DEFINITION 4.2 (Admissibility). Let C_W be a set of ω-clauses and C_D be a set of δ-clauses, and let Π be a mating for $C_W \cup C_D$. Then, $(C_W \cup C_D, \Pi)$ is admissible iff there is an enumeration $\langle\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i\in I}$ of $\kappa(C_D, \Pi)$ such that for $i \in I, \Pi$ is a spanning mating for

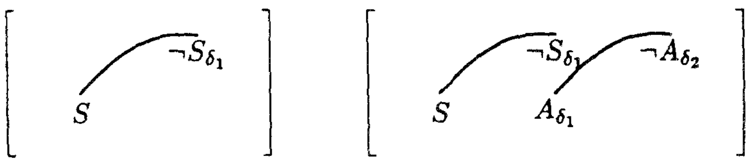
$$C_W \cup \left(\bigcup_{j=0}^{i-1}\{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}. \tag{4.6}$$

Note that normally not all connections in Π are needed for showing the unsatisfiability of the submatrices in (4.6). We say that $(C_W \cup C_D, \Pi)$ is admissible at i in an index set I if (4.6) holds for $i \in I$. Moreover, we say that $(C_W \cup C_D, \Pi)$ is admissible wrt I if it is admissible at all $i \in I$.

The previous definition may be nicely illustrated by the proof in our student example given in (4.4). There, we obtain the enumeration

$$\langle\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}\rangle,$$

which in turn leads to the following matrices, each representing a set of clauses as specified in (4.6):



(4.7)

Observe that the preceding matrices are in fact submatrices of matrix (4.4). Clearly, each of these submatrices has a spanning mating, so that the original matrix along with its mating, given in (4.4), constitute an admissible proof. Observe that the proof in the example involving contraposition violates admissibility. This is so because there is no spanning mating for the submatrix $\{\{\neg E\}, \{\neg A_{\delta_3}\}\}$ of matrix (4.5).

In the remainder of this subsection, we provide an incremental approach to admissibility. This is made precise in the following theorem.

THEOREM 4.1. *Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let Π be a mating for $C_W \cup C_D$ such that $(C_W \cup C_D, \Pi)$ is admissible wrt I. Let $\{\neg\alpha_\delta, \gamma_\delta\}$ be a δ-clause. Then, $(C_W \cup C_D \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}, \Pi)$ is admissible iff Π is a spanning mating for $C_W \cup \bigcup_{i \in I}\{\{\gamma_{\delta_i}\}\} \cup \{\{\neg\alpha_\delta\}\}$.*

Informally, this theorem allows us to discard paths through 'prerequisites of admissible δ-clauses' while verifying admissibility. Hence, for verifying the admissibility of the proof given in (4.4), we can proceed as follows. For illustration, consider also the two submatrices in (4.7). We start with the set of open paths through all ω-clauses. There is only one such path in our example, $\{S\}$. For verifying the admissibility of[15] $\{\neg S_{\delta_1}, A_{\delta_1}\}$, we have to check whether all such open paths contain a literal complementary to $\neg S_{\delta_1}$. Since this is the case, we can proceed by verifying the admissibility of $\{\neg A_{\delta_2}, C_{\delta_2}\}$. For this, we can discard all paths through $\neg S_{\delta_1}$. Thus, we can restrict ourselves to all open paths obtained by adding A_{δ_1} to all open paths through all ω-clauses. There is only one such path in our example, $\{S, A_{\delta_1}\}$. As above, this path has to contain a literal complementary to $\neg A_{\delta_2}$ for confirming the admissibility of the second δ-clause. Clearly, the path $\{S, A_{\delta_1}\} \cup \{\neg A_{\delta_2}\}$ is closed, so that admissibility is confirmed.

Moreover, the theorem shows that admissibility is in fact the proof-theoretic counterpart of groundedness. That is, if C_W is the clausal representation of W, then there is a spanning mating for $C_W \cup \bigcup_{i=0}^{n}\{\{\gamma_{\delta_i}\}\} \cup \{\{\neg\alpha_\delta\}\}$ iff $W \cup \text{Conseq}(\{\delta_0, \ldots, \delta_n\}) \vdash \text{Prereq}(\delta)$, where $\gamma_{\delta_i} = \text{Conseq}(\delta_i)$. Observe that the latter corresponds to the condition given for groundedness in (3.2).

4.4. Compatibility

The second qualifying condition for proofs is given by the concept of compatibility; it relies on the notion of consistency specific to constrained default logic.

DEFINITION 4.3 (Compatibility). Let C_W be a set of ω-clauses and C_D be a set of δ-clauses, and let Π be a mating for $C_W \cup C_D$. Then, $(C_W \cup C_D, \Pi)$ is compatible iff there is no spanning mating for $C_W \cup C_J$, where $C_J = \{\{\beta_\delta\}, \{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in \kappa(C_D, \Pi), \beta_\delta = \text{Justif}(\delta)\}$.

Notably, this is the first place where we refer to a notion specific to constrained default logic; the entire preceding exposition involving the concept of admissibility applies to any (semi-monotonic) default logic.

Consider again our student example. For compatibility, we have to verify that the matrix $\{\{S\}\} \cup \{\{A_{\delta_1}\}, \{C_{\delta_2}\}\}$ or two-dimensionally

$$\begin{bmatrix} S & A_{\delta_1} & C_{\delta_2} \end{bmatrix} \tag{4.8}$$

has no spanning mating. This matrix is formed by the facts $\{S\}$ and the justifications A_{δ_1} and C_{δ_2} of the first two default rules in (2.1). Obviously, matrix (4.8) has no spanning mating, since it has a noncomplementary path, $\{S, A_{\delta_2}, E_{\delta_3}\}$. We thus obtain an admissible and compatible proof for the original query, S, asking whether a student drives a car. Note that an open path gives a model of the considered formula.

In order to give an example for an incompatible proof, we consider the matrix

$$\{\{S\}\} \cup \{\{A\}, \{\neg S\}, \{E\}\}$$

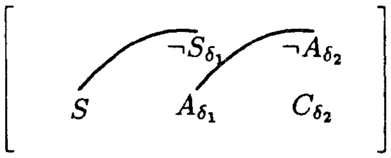
whose compatibility is verified while answering the query E from the fact S and the default rules $(S : A)/A$ and $(A : \neg S)/E$. This matrix has a spanning mating $\{\{S, \neg S\}\}$ indicating an incompatible use of default rules.

In principle, compatibility is separate from admissibility. However, the next theorem shows that compatibility can be verified on (almost) the same matrices as used for verifying complementarity and admissibility.

THEOREM 4.2. *Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let Π be a mating for $C_W \cup C_D$ such that $(C_W \cup C_D, \Pi)$ is admissible. Then, Π is a spanning mating for $C_W \cup C_D \cup \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D, \beta_\delta = \text{Justif}(\delta)\}$ iff Π is a spanning mating for $C_W \cup C_J$, where $C_J = \{\{\beta_\delta\}, \{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D, \beta_\delta = \text{Justif}(\delta)\}$.*

This theorem offers the computational advantage of structure *and* information sharing while query answering. Observe that a simpler formulation is obtained for normal default theories. Then, Π is a spanning mating for $C_W \cup C_D$ iff Π is a spanning mating for $C_W \cup \{\{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D\}$.

Above, we have verified the compatibility of the proof obtained in our student example by regarding the matrix given in (4.8). Theorem 4.2 tells us that this is equivalent to checking whether the following admissible supermatrix of (4.8) has no spanning mating.[16]



(4.9)

Even though the latter matrix is larger than the one in (4.8), it shares the structure of the matrices used for verifying complementarity and admissibility. In fact, it is at the same time a supermatrix of the largest matrix used for checking admissibility in (4.7) and a submatrix of the actual matrix used for proving the query C in (4.4). That is, matrix (4.9) is obtained by adding C_{δ_2} to the rightmost clause of the right matrix in (4.7). Analogously, we obtain the proof for C in (4.4) by adding the query clause $\{\neg C\}$ to matrix (4.9). We will take up these ideas in Section 6.

4.5. Characterizing Default Proofs

In Section 3, we have decomposed default theorem proving in default logic into classical deduction along with the concepts of groundedness and consistency. In the preceding subsections, we have carefully mapped these notions onto the connection method. We have accomplished this by identifying the concepts complementarity, admissibility, and compatibility as the proof-theoretic counterparts of classical deduction, groundedness, and consistency, respectively.

As a result, we obtain the following theorem showing that our method is correct and complete for constrained default logic:

THEOREM 4.3. *Let (D, W) be a default theory in atomic format and φ an atomic formula. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff there is a spanning mating Π for the matrix M of $W \cup W_D \cup \{\neg\varphi\}$ such that (M, Π) is admissible and compatible.*

As agreed upon above, we have that $W_D = \{\alpha_\delta \rightarrow \gamma_\delta \mid (\alpha_\delta : \beta_\delta)/\gamma_\delta \in D\}$.

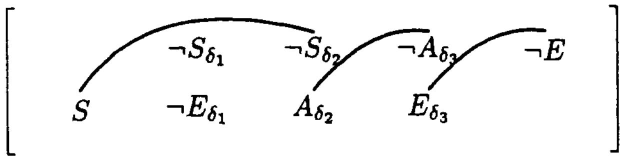
Finally, let us summarize our approach in the remainder of this section by means of a coherent example. Consider the statements 'students are typically not employed', 'students are typically adults', and 'adults are typically employed', along with the corresponding default theory dealing with a student.

$$\left(\left\{\frac{S:\neg E}{\neg E},\frac{S:A}{A},\frac{A:E}{E}\right\},\{S\}\right). \tag{4.10}$$

The encoding of the set of default rules yields the following set of implications.

$$W_D = \{S_\delta \rightarrow \neg E_{\delta_1}, S_{\delta_2} \rightarrow A_{\delta_2}, A_{\delta_2} \rightarrow E_{\delta_3}\}.$$

As before, the indexes denote the respective default rules in default theory (4.10) from left to right. Let us consider the query E, asking whether a student is employed. Transforming the fact S, the implications in W_D, and the negated query $\neg E$ into clausal form yields the matrix in (4.11).

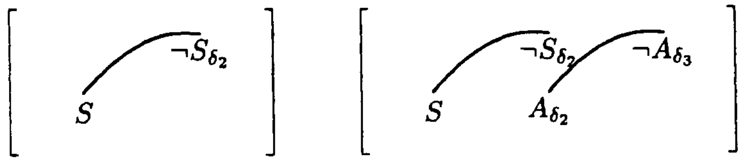


(4.11)

In fact, the matrix has a spanning mating, $\{\{S,\neg S_{\delta_2}\},\{A_{\delta_2},\neg A_{\delta_3}\},\{E_{\delta_3},\neg E\}\}$, whose connections are indicated as arcs linking the respective literals. This default proof yields the following enumeration

$$\langle\{\neg S_{\delta_2}, A_{\delta_2}\},\{\neg A_{\delta_3}, E_{\delta_2}\}\rangle$$

For admissibility, we have to consider the following submatrices of matrix (4.11)

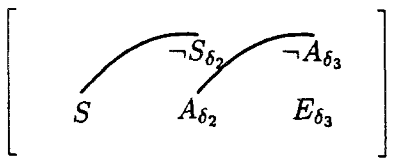


(4.12)

Observe that each of these submatrices has a spanning mating, so that the original matrix and its mating, given in (4.11), constitute an admissible proof.

For compatibility, we have to verify that the following matrix has no spanning mating.[17]

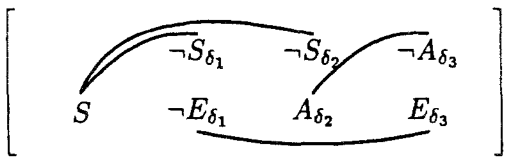
Obviously this is the case because there is a noncomplementary path, $\{S, A_{\delta_2}, E_{\delta_3}\}$. We thus obtain an admissible and compatible proof for the original query, E, asking whether a student is employed. Note that we used Theorem 4.2 for verifying compatibility.



Observe that there is yet another spanning mating for the matrix in (4.11), namely,

$$\{\{S, \neg S_{\delta_1}\}, \{S, \neg S_{\delta_2}\}, \{\neg E_{\delta_1}, E_{\delta_3}\}, \{A_{\delta_2}, \neg A_{\delta_3}\}\}. \tag{4.13}$$

This mating discards the negated query $\neg E$. The cause for this is that we deal with conflicting defaults. That is, from S we can derive $\neg E$ by the first default rule in default theory (4.11) as well as E by the second and third default rule. Although the resulting proof can be shown to be admissible, it is not compatible.



This matrix has the spanning mating given in (4.13), too. This shows that the corresponding proof is not compatible.

The last part of the example stresses the importance of the concept of compatibility. In particular, it seems advantageous to prune incompatible proofs as early as possible, since defaults might conflict with each other.

5. Implementing the Approach by Existing Automated Theorem Provers

In this section, we pursue our initial goal of providing a simple method for query answering in default logics that needs few modifications to existing implementations of automated theorem provers.

There are several ways of implementing our approach by using existing automated theorem provers. An extreme way would be to prove each query conventionally and to leave the verification of admissibility and compatibility to special-purpose algorithms. This is rather expensive, since one might have to generate numerous proofs before our qualifying conditions are confirmed or even denied. The opposite approach would be to modify an existing automated theorem prover in order to incorporate the verification of admissibility and compatibility. To this end, however, one has to put consistency checks into the 'inner loop' of a theorem prover, which is a difficult and (sometimes) expensive undertaking, too.

5.1. AN ALGORITHM

In all, both aforementioned approaches do not concur with our initial desire for a simple and feasible approach to default theorem proving that is easily adaptable by existing implementations of the connection method, like SETHEO [21] or PPP [26]. We address this problem in this section by separating the verification of compatibility (or consistency) from that of complementarity and admissibility. This is motivated by the incongruity between the 'global' notion of consistency employed in default logics (by referring to the final extension or constraints) and the stepwise execution of inference steps encountered in existing theorem provers. In order to avoid the resulting difficulties, we rather pursue an 'off-line' approach by compiling compatibility, thereby taking advantage of the compliant conception of consistency in constrained default logic. This approach is justified by the following corollary to Theorem 3.1.

COROLLARY 5.1. *Let (D, W) be a default theory, and let E and C be sets of formulas. Then, (E, C) is a constrained extension of (D, W) iff (E, C) is a constrained extension of (D', W) for a maximal $D' \subseteq D$ such that $W \cup$* $\mathrm{Justif}(D') \cup \mathrm{Conseq}(D')$ *is consistent.*

We say that a default theory (D, W) is *compatible* iff $W \cup \mathrm{Justif}(D) \cup \mathrm{Conseq}(D)$ is consistent. Accordingly, we compile a given default theory (D, W) into several compatible default theories (D', W). Compiling a default theory (D, W) amounts to computing the generating default rules[18] D' of each extension of the default theory

$$\left(\left\{\frac{: \beta \wedge \gamma}{\beta \wedge \gamma} \,\middle|\, \frac{\alpha : \beta}{\gamma} \in D\right\}, W\right).$$

Observe that any compatible default theory has a unique constrained extension.

For example, we can turn default theory (2.1) into a single compatible default theory by removing the last default rule, $(A : \neg S)/E$. This usually costly computation should be done 'off-line' by special-purpose algorithms, as described in [1] or even [43]. Once this has been done, we can verify whether a query is in the *unique* extension of a compatible default theory *without* any consistency checks. An effective way of querying multiple compatible default theories is described in [33]. The precomputation of compatible default theories has the advantage that we are able to prune computations with incompatible defaults in advance. Thus, for instance, the approach avoids the difficulties with incompatible default theories sketched at the end of Section 4.

On the other hand, the approach is problematic if there is a large number of compatible default theories. In fact, there may be an exponential number of such theories in the worst case.[19] In general, such a compilation is favorable whenever its computational cost can be amortized over the total set of subsequent queries.

The purpose of this compilation approach is to minimize modifications to existing automated theorem provers. In fact, it turns out that admissibility is more integrative than compatibility as regards such modifications. We discuss alternative approaches in brief at the end of this section and in more detail in Section 6.

Let us now turn to the verification of admissibility and complementarity. In fact, we confirm admissibility while systematically checking the complementarity of each path through a matrix. Following [14], we use compl(p, M) for defining a declarative algorithm for deciding whether a matrix is complementary and admissible. With it, Eder shows in [14] that a matrix M consisting of ω-clauses only is complementary iff compl$(\varnothing, M)$ is true wrt the first two conditions of the following definition.

DEFINITION 5.1. Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let p be a set of literals and let $M = C_{W'} \cup C_{D'}$ for $C_{W'} \subseteq C_W$ and $C_{D'} \subseteq C_D$. Then, we define compl(p, M) relative to C_W as follows.[20]

1. If $M = \varnothing$, then compl(p, M) is false.
2. If $M \neq \varnothing$ and $c \in M$ is an ω-clause, then compl(p, M) is true iff for all $L \in c$ at least one of the following two conditions holds.
   (a) L is complementary to some literal of p.
   (b) compl$(p \cup \{L\}, M \backslash \{c\})$ is true.
3. If $M \neq \varnothing$ and $c \in M$ is a δ-clause, then compl(p, M) is true iff the following two conditions hold, where $c = \{\neg\alpha_\delta, \gamma_\delta\}$.
   (a) γ_δ is complementary to some literal of p.
   (b) compl$(\{\neg\alpha_\delta\}, (M \backslash \{c\}) \cup C_W)$ is true.

As mentioned above, the first two conditions provide a sound and complete algorithmic characterization of the standard connection method (see [14] for details). In fact, the original characterization given in [14] differs only in two extremely minor points from the one obtained by deleting Condition (3) above. First, there is no case analysis in Condition (2) for distinguishing ω- from δ-clauses. Second, compl(p, M) is independent of C_W in [14]. The latter set represents in Definition 5.1 the original set of ω-clauses, whereas $C_{W'}$ and $C_{D'}$ function as parameters. This distinction is necessary because Condition (3b) makes reference to the original set of ω-clauses, given by C_W. We will come back to this below.

Now, let us discuss Definition 5.1 in some detail. Condition (1) accounts for the limiting case where the matrix is empty. Condition (2) deals with ω-clauses. Each literal L in the ω-clause at hand either has to be complementary to some literal on the active path p or all paths through p extended by L and the remaining clauses have to be complementary. The choice of the ω-clause from M is a *don't care*-choice. That is, the result is independent of what ω-clause is taken.

Condition (3) deals with δ-clauses. Condition (3a) corresponds to Condition (2a) and says that the consequent of a default rule γ_δ can be used for query answering as any other proposition – provided Condition (3b) is satisfied. In fact, (3b) 'implements' Statement (4.6) in Definition 4.2 and ensures that the prerequisite α_δ of a default rule is derivable in a noncircular way. Observe that we do not provide two alternatives for resolving γ_δ in (3b), as done in (2). In fact, we can restrict ourselves to one of the alternatives in (2a) and (2b) for solving a single literal.[21] The purpose of resolving γ_δ in analogy to (2a) rather than (2b) is to minimize the 'application of default rules' in the course of a proof search. This minimization is advantageous because the choice of δ-clauses in (3) is a *don't know* choice. That is, one has to find the right one, which means that – in the worst case – all possibilities have to be tested. The choice is *don't know* because a selected δ-clause may not lead to an admissible proof so that Condition (3b) will be falsified.

Another interesting point in Condition (3b) is the addition of the initial set of ω-clauses C_W to $(M \backslash \{c\})$. The need for this is obvious because admissibility has to be verified wrt the given set of facts represented by C_W. Some of these ω-clauses, however, might have been 'consumed' at an earlier stage. This is so because Eder's formulation deletes in Condition (2b) ω-clauses after their 'usage'.[22] So on the one hand, our approach avoids verifying admissibility by ever-increasing submatrices, as stipulated in Definition 4.2. In this way, it compromises the query-oriented and thus 'top-down' search for a proof with the 'bottom-up' verification of admissibility. On the other hand, an ω-clause may contribute to the derivation of several 'prerequisites'. Thus, in the worst case, this yields a proof length bounded by $O(|C_W| \times (|C_D| + 1))$, where $|M|$ stands for the number of clauses in a matrix M.

We observe how easily complementarity and admissibility can be verified simultaneously by adding a single condition to the original definition of compl(p, M) in [14] – provided that M represents a compatible default theory. Let us illustrate this along with our algorithm by investigating the 'compatible' matrix

$$\{\{S\}, \{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}, \{\neg C\}\}$$

obtained by removing clause $\{\neg A_{\delta_3}, E_{\delta_3}\}$ from matrix (4.4). To proceed in a query-oriented way, we 'push' the negated query $\neg\varphi$ on the initial path, and use the ω- and δ-clauses, $C_W \cup C_D$ representing the underlying default theory as the initial matrix. That is, we verify whether compl$(\{\neg\varphi\}, C_W \cup C_D)$ is true. Selecting the query clause $\{\neg C\}$ makes us confirm

$$\text{compl}(\{\neg C\}, \{\{S\}, \{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}\}). \tag{5.14}$$

This can be done by choosing clause $\{\neg A_{\delta_2}, C_{\delta_2}\}$ in Condition (3). This choice is not arbitrary; rather, it reflects the connection-driven search used in the connection method. That is, C_{δ_2} is complementary to $\neg C$ so that (3a) is satisfied. In addition, we have to establish

$$\text{compl}(\{\neg A_{\delta_2}\}, \{\{S\}, \{\neg S_{\delta_1}, A_{\delta_1}\}\}) \tag{5.15}$$

according to (3b). It is instructive to verify that the latter corresponds to invoking our algorithm on the second (sub)matrix in (4.7), after applying Condition (2b) to the clause $\{\neg A_{\delta_2}\}$.

Applying Condition (3) to the goal in (5.15) yields

$$\text{compl}(\{\neg S_{\delta_1}\}, \{\{S\}\}) \tag{5.16}$$

which is true by (2a). Accordingly, the query C is provable from the compatible equivalent of default theory (2.1) by means of default rules $(S : A)/A$ and $(A : C)/C$.

For a complement, let us consider the proof in (4.5) involving reasoning by contraposition. This proof is not admissible. We start with

$$\text{compl}(\{A\}, \{\{\neg E\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}).$$

We choose $\{\neg A_{\delta_3}, E_{\delta_3}\}$ in Condition (3). Since $\neg A_{\delta_3}$ is complementary to A, we have to confirm

$$\text{compl}(\{\neg A_{\delta_3}\}, \{\{\neg E\}\}).$$

Clearly, this is false because there are no complementary literals left. The same result is obtained by initially choosing $\{\neg E\}$. Hence, our initial goal is not confirmed, thus showing that the proof in (4.5) is not admissible.

The general relation between query answering in constrained default logic and the above algorithm is made precise in the following theorem.[23]

THEOREM 5.2. *Let (D, W) be a default theory in atomic format, and let φ be an atomic formula. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff* compl$(\{\neg\varphi\}, M)$ *is true for the matrix M of $W \cup W_{D'}$ for some $W_{D'} \subseteq W_D$ such that $W \cup \text{Justif}(D') \cup \text{Conseq}(D')$ is consistent.*

Observe that merely the choice of the compatible set of default rules D' is specific to constrained default logic. Hence this result and with it the underlying algorithm apply to 'compatible' default theories in any (semi-monotonic) default logic – provided that an appropriate notion of compatibility is provided (cf. Section 9).

The previous exposition is dominated by the view that the integration of consistency into existing implementations of automated theorem provers is difficult. In particular, we have argued in favor of special-purpose algorithms for compilation into compatible default theories. Without question, these algorithms show a better performance than a theorem prover whose failure indicates that the underlying formula is satisfiable. For coherence, actually, notice that we can also compile default theories into compatible ones by means of $\mathrm{compl}(p, M)$. That is, a default theory (D, W) is compatible iff $\mathrm{compl}(p, M)$ is false for the matrix of $W \cup \mathrm{Justif}(D) \cup \mathrm{Conseq}(D)$.

Also, recall that the compilation of default theories leads to difficulties whenever there are a large number of compatible default theories. Then, an 'on-line' approach is definitely preferable over an 'off-line' approach. In fact, this can also be accomplished by means of what we have developed so far. Let $(C_D)^\omega$ be the set of ω-clauses obtained by turning each δ-clause in C_D into an ω-clause. This leads us to the following corollary to Theorem 5.2, which allows us to reason from arbitrary and thus also noncompatible default theories.

COROLLARY 5.3. *Let (D, W) be a default theory in atomic format, and let φ be an atomic formula. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff* $\mathrm{compl}(\{\neg\varphi\}, M)$ *is true, and* $\mathrm{compl}(\mathrm{Justif}(D'), M^\omega)$ *is false for the matrix M of $W \cup W_{D'}$ for some $W_{D'} \subseteq W_D$.*

Observe that this corollary relies on Theorem 4.2, which shows that compatibility is verifiable on the same matrix M as used in $\mathrm{compl}(\{\neg\varphi\}, M)$. Also, we used the fact that for any (unit-)clause $\{L\}$ containing a single literal L, we have that $\mathrm{compl}(p, M \cup \{\{L\}\})$ is true iff $\mathrm{compl}(p \cup \{L\}, M)$ is true. Hence, we shifted all justifications on the initial path. In this way, the actual matrix M remained the same in $\mathrm{compl}(\{\neg\varphi\}, M)$ and $\mathrm{compl}(\mathrm{Justif}(D'), M^\omega)$.

5.2. A CASE STUDY

Aaron Rothschild has implemented the approach in [33] by slightly extending PPP, a PROLOG implementation of a (first-order) theorem prover carrying out the pool-based connection calculus [26]. The purpose of this implementation was to provide an initial case study in how far an existing theorem prover for the connection method has to be modified for incorporating admissibility.

We briefly describe the main idea underlying the implementation while assuming some basic familiarity with the connection method: We prove in a goal-oriented fashion, starting from the goal and attempting to find complementary paths through the matrix. As soon as a path cannot be complemented by using facts only, we call in δ-clauses to achieve complementarity. As in Definition 5.1, we do not attempt to use ever-increasing submatrices, as stipulated in Definition 4.2 for verifying admissibility. Rather, we enforce the admissible application of defaults by two extra conditions resembling the ones in Condition (3) in Definition 5.1: The first condition (corresponding to (3a)) says that only connections 'into' γ_δ-literals of δ-clauses $\{\neg\alpha_\delta, \gamma_\delta\}$ are permissible in the course of the backward chained search. In this way, a subgoal is never resolvable by the prerequisite α_δ of a default rule. Once an inference step with a δ-clause is performed, the second condition restricts the resolution of subgoals with literals of the current path to literals that entered the path *after* the aforementioned 'default step'. This amounts to discarding the literals of the path p in (3b) in order to avoid circular chains of inference.

Importantly, these two conditions are implemented by simply adding another PROLOG clause (along with some case-analysis distinguishing δ- and ω-clauses) to the PROLOG implementation of PPP. That is, apart from a *single* PROLOG clause the rest of our implementation corresponds to the original implementation given in [26]. Hence, it was possible to minimize modifications by utilizing as much of the original implementation as possible.

In practice, however, the length of the proofs had to be limited by a parameter in the size of $O(|C_W| \times (|C_D| + 1))$ in order to guarantee completeness in the propositional case. Otherwise the implementation ran into infinite branches since PPP is a first-order theorem prover that deals with clause instances. A simpler PROLOG implementation that relies on the characterization in Definition 5.1 is given in Section 8.

Finally, the question arises how our method can be transposed onto a high-performance theorem-prover like SETHEO [21]. In fact, this should not be that difficult provided that we keep separating the verification of compatibility. SETHEO is a PROLOG-technology theorem prover written in the programming language C. It is built on top of the SETHEO-abstract machine that works with PROLOG-like rules. To this end, each clause, like $\{A \vee B\}$, is transformed into its contrapositives, $\neg A \rightarrow B$ and $\neg B \rightarrow A$. Roughly speaking, this suggests the following two changes for dealing with δ-clauses – apart from some case analysis. First, δ-clauses like $\{\neg\alpha_\delta, \gamma_\delta\}$ are transformed in a single contrapositive $\alpha_\delta \rightarrow \gamma_\delta$. Second, the head of such a 'δ-contrapositive' has to be proven by deleting all literals on the active path. Observe that these two restrictions correspond to Conditions (3a) and (3b) in Definition 5.1. A detailed study of modifying the treatment of contrapositives is given in [44, 25].

6. An Alternative Approach

In the preceding section, we developed an algorithm for our method while aiming at implementing the approach by using existing automated theorem provers. For this purpose, we concentrated on minimizing the modifications to existing implementations. This has led to a pragmatic solution separating the verification of compatibility (or consistency) from that of complementarity and admissibility. In this section, we investigate an alternative approach that requires more modifications to an automated theorem prover but that allows for integrating the verification of compatibility.

6.1. An Alternative Characterization of Extensions

The fundamental basis for the approach developed in the preceding sections was provided by Theorem 3.1. In particular, we have stressed the fact that semi-monotonicity allows for focusing on the default rules needed for proving a query, while developing the corresponding characterization of query answering in Corollary 3.2.

In fact, semi-monotonicity offers yet another but conceptually different characterization of extensions. Observe that the specification given in Theorem 3.1 employs a rather 'global' notion of consistency. Now, semi-monotonicity implies that extensions are constructible in a truly iterative way by applying one applicable default rule after another. This involves an incremental and thus rather local notion of consistency. To this end, semi-monotonicity leads us to the following corollary to Theorem 3.1 that provides an alternative characterization of constrained extensions:

COROLLARY 6.1. *Let (D, W) be a default theory, and let E and C be sets of formulas. Then, (E, C) is a constrained extension of (D, W) iff*

$$E = \mathrm{Th}(W \cup \mathrm{Conseq}(D')) \text{ and}$$

$$C = \mathrm{Th}(W \cup \mathrm{Justif}(D') \cup \mathrm{Conseq}(D'))$$

for a maximal $D' \subseteq D$ such that there exists an enumeration $\langle\delta_i\rangle_{i\in I}$ of D', where for $i \in I$ we have that

(1) $W \cup \mathrm{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \mathrm{Prereq}(\delta_i)$, *and*

(2) $W \cup \mathrm{Conseq}(\{\delta_0, \ldots, \delta_{i-1}) \cup \mathrm{Justif}(\{\delta_0, \ldots, \delta_{i-1}\}) \nvdash \neg\mathrm{Justif}(\delta_i) \vee \neg\mathrm{Conseq}(\delta_i)$.

This specification explicates the formation of sequences of default rules that remained implicit in Theorem 3.1. In fact, Condition (1) spells out that D' has to be grounded in W. So the conceptional difference between the two alternative characterizations rests on the second condition. Condition (2) expresses the aforementioned notion of *incremental consistency*. Here, the 'consistent' application of a default rule is checked at each step, whereas this is done jointly for all default rules in D' in Theorem 3.1.

Corollary 6.1 provides the fundamental basis for the approach to query answering, which we develop in this section. The characterization of query answering is analogous to the one given in Corollary 3.2. Observe that in Theorem 3.1 groundedness and consistency constitute rather separate constraints on the 'generating default rules' in D'. We strongly relied on the possibility of separating these concepts in Section 5. In contrast to this, the concepts of groundedness and consistency are meshed together in Corollary 6.1. Hence, both concepts jointly direct the *formation of sequences of default rules*. This is the salient feature of the approach developed in the sequel. Moreover, we can see in [39] that the combination of both concepts is very useful for implementing priorities.

6.2. Incremental compatibility

Clearly, the 'global' notion of compatibility given in Definition 4.3 is inappropriate in order to account for the above characterization. Rather Condition (2) in Corollary 6.1 requires an incremental approach in which compatibility is gradually verified each time a δ-clause is considered. This motivates the following definition.

DEFINITION 6.1 (Incremental compatibility). Let C_W be a set of ω-clauses and C_D be a set of δ-clauses, and let Π be a mating for $C_W \cup C_D$. Let $\langle\{\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i \in I}$ be an enumeration of $\kappa(C_D, \Pi)$. Then, $(C_W \cup C_D, \Pi)$ is incrementally compatible wrt I iff for all $i \in I$, there is no spanning mating for

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\beta_{\delta_j}\}, \{\gamma_{\delta_j}\}\} \right) \cup \{\{\beta_{\delta_i}\}\} \cup \{\{\gamma_{\delta_i}\}\}, \tag{6.17}$$

where $\beta_{\delta_i} = \text{Justif}(\delta_i)$.

We say that $(C_W \cup C_D, \Pi)$ is compatible at i in an index set I if (6.17) holds for $i \in I$. Moreover, we say that $(C_W \cup C_D, \Pi)$ is incrementally compatible wrt an index set I if it is compatible at all $i \in I$.

The next theorem tells us that 'global' and incremental compatibility are in fact equivalent.

THEOREM 6.2. *Let C_W be a set of ω-clauses and C_D be a set of δ-clauses, and let Π be a mating for $C_W \cup C_D$. Let $\langle\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i \in I}$ be an enumeration of $\kappa(C_d, \Pi)$. Then, $(C_W \cup C_D, \Pi)$ is compatible iff $(C_W \cup C_D, \Pi)$ is incrementally compatible wrt I.*

Consider our initial student example. Instead of checking whether the matrix in (4.8) has no spanning mating, we can stepwisely verify whether this holds for the following matrices.

$$[S \quad A_{\delta_1}] \qquad [S \quad A_{\delta_1} \quad C_{\delta_2}] \tag{6.18}$$

In this way, we check first the compatibility of the facts $\{S\}$ and the consequent of δ_1. Then, the same test is performed on the matrix extended by the consequent of δ_2. Note that at each step it is sufficient to consider only the noncomplementary paths obtained in the previous step.

We obtain the following corollary to Theorem 4.3 and Theorem 6.2. This result shows that our incremental method is correct and complete for query answering in constrained default logic.

COROLLARY 6.3. *Let (D, W) be a default theory in atomic format and φ an atomic formula. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff there is a spanning mating Π for the matrix $M = C_W \cup C_D \cup \{\{\neg\varphi\}\}$ of $W \cup W_D \cup \{\neg\varphi\}$ and an enumeration $\langle c_i \rangle_{i \in I}$ of $\kappa(C_D, \Pi)$ such that (M, Π) is admissible wrt I and incrementally compatible wrt I.*

Now, recall that by Theorem 4.2 compatibility and hence also incremental compatibility can be verified by using δ-clauses, like $\{\neg\alpha_\delta, \gamma_\delta\}$, instead of clauses containing merely the consequent of a default, like $\{\gamma_\delta\}$, in the case of 'admissible matrices'. Consequently, incremental compatibility can be equivalently verified by replacing the matrices in (6.17) by matrices of the following form:

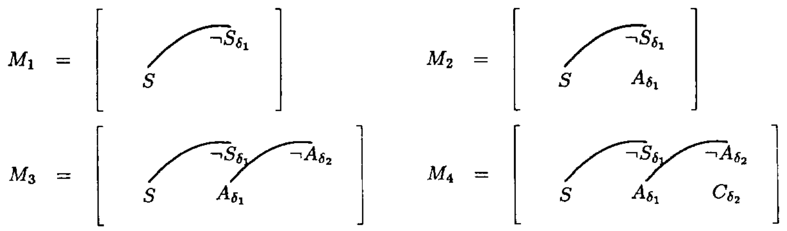
$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\} \right) \cup \{\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\} \cup \left(\bigcup_{j=0}^{i-1} \{\{\beta_{\delta_j}\}\} \right) \cup \{\{\beta_{\delta_i}\}\}.$$

This offers the computational advantage that we can verify incremental compatibility on (almost) the same matrices as used for verifying admissibility. In fact, admissibility is checked on matrices of the form (cf. Equation (4.6))

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\} \right) \cup \{\{\alpha_{\delta_i}\}\}.$$

For admissibility, all paths through the latter matrix have to be complementary. For compatibility, there has to emerge an open path if we replace the clause $\{\{\neg\alpha_{\delta_i}\}\}$ by the clause $\{\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\}$, simply by adding the consequent of the default δ_i. In addition, each such open path must not contain a literal complementary to any justification in $(\bigcup_{j=0}^{i-1}\{\{\beta_{\delta_j}\}\}) \cup \{\{\beta_{\delta_i}\}\}$. This additional requirement is obsolete in the case of normal default theories. On the whole, Theorem 4.2 provides a valuable refinement that allows for structure *and* information sharing while jointly verifying admissibility and compatibility.

To illustrate this, let us look at our initial default proof in (4.4). For verifying its admissibility, we used the submatrices in (4.12). These are repeated as M_1 and M_3 below. In fact, we can share the use of these matrices for testing compatibility. This amounts to considering in turn the following submatrices of (4.4):



We start by verifying whether all paths through matrix M_1 are complementary. Since this is the case, M_1 is admissible. For checking the compatibility of M_2, we merely have to look for a noncomplementary path through the facts, here $\{\{S\}\}$ and the consequent of δ_1, A_{δ_1}. All other paths are complementary by admissibility. In fact, the path $\{S, A_{\delta_1}\}$ is not complementary and so the matrix M_2 is compatible.

For verifying admissibility in the case of matrix M_3, we can make use of the information gathered on M_2. In this example, it is enough to check whether adding the negated prerequisite of $\delta_2, \neg A_{\delta_2}$ closes all open path in matrix M_2. In fact, this is the case since $\{S, A_{\delta_1}\}$ in the only open path in the matrix M_2 and $\{S, A_{\delta_1}\} \cup \{\neg A_{\delta_2}\}$ is complementary. The compatibility of matrix M_4 is established in analogy to that of M_2; again by reusing the information gathered while verifying admissibility for M_3. The final proof of the query C in (4.4) is obtained by adding the query clause $\{\neg C\}$ to matrix M_4. Clearly, the resulting matrix is complementary so that the proof is completed.

6.3. An Alternative Algorithm

The general idea of our algorithmic approach is to proceed in a query-oriented manner. We extend the definition of compl(p, M) as given in Definition 5.1 in the following way. We use a predicate compl(p, C_W, C_D) for defining a declarative algorithm for deciding whether a matrix is complementary, admissible, *and* compatible. The first argument is a set of literals describing a partial path, the second argument represents a set of ω-clauses, and the last argument accounts for δ-clauses.

DEFINITION 6.2. Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let p be a set of literals, and let $C_{W'} \subseteq C_W$ and $C_{D'} \subseteq C_D$. Then we define compl$(p, C_{W'}, C_{D'})$ relative to C_W as follows.[24]

1. If $C_{W'} \cup C_{D'} = C_{D'_L}, \varnothing$, then compl$(p, C_{W'}, C_{D'})$ is false.
2. If $C_{W'} \neq \varnothing$ and $c \in C_{W'}$, then compl$(p, C_{W'}, C_{D'})$ is true iff the following two conditions hold for $c = c_1 \cup c_2$.
   (a) for all $L \in c_1$, L is complementary to some literal of p.
   (b) for all $L \in c_2$, there is a set of δ-clauses $C_{D'_L} \subseteq C_{D'}$ such that following two conditions hold.
      (i) compl$(p \cup \{L\}, C_{W'} \setminus \{c\}, C_{D'_L})$ is true.
      (ii) compl(Justif$(\bigcup_{L \in c_2} D'_L), C_W \cup \bigcup_{L \in c_2} C_{D'_L}, \varnothing)$ is false.
3. If $C_{D'} \neq \varnothing$ and $c \in C_{D'}$, then compl$(p, C_{W'}, C_{D'})$ is true iff the following two conditions hold for $c = \{\neg\alpha_\delta, \gamma_\delta\}$.
   (a) γ_δ is complementary to some literal of p.
   (b) There is a set of δ-clauses $C_{D''} \subseteq C_{D'}$ such that $\{\neg\alpha_\delta, \gamma_\delta\} \in C_{D'} \setminus C_{D''}$ and the following two conditions hold.
      (i) compl$(\{\neg\alpha_\delta\}, C_W, C_{D''})$ is true.

(ii) compl(Justif$(D'' \cup \{\delta\}), C_W \cup C_{D''} \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}, \varnothing)$ is false.

As in Definition 5.1, C_W and C_D represent the original set of ω- and δ-clauses, whereas $C_{W'}$ and $C_{D'}$ function as parameters. Conditions (1) and (2) correspond to the ones in Definition 5.1. There are two differences. First, we have separated ω- and δ-clauses. This separation allows for an easier formulation of Condition (3). Second, we have added Condition (2bii) in order to guarantee the compatibility of multiple subproofs found in (2bi). We explain the treatment of compatibility in (2bii) in the context of Condition (3bii) below. Anyway, we have that a matrix M (representing a satisfiable formula) consisting of ω-clauses only is complementary iff compl$(\varnothing, M, \varnothing)$ is true wrt the first two conditions of Definition 6.2.

As in Definition 5.1, Condition (3a) corresponds to Condition (2a) and allows for solving subgoals on the actual path by the consequent of a default rule γ_δ – provided Condition (3b) is satisfied. Condition (3b) combines the verification of admissibility with that of incremental compatibility. For that, a set of δ-clauses $C_{D''}$ is selected from the available set of δ-clauses $C_{D'}$. $C_{D''}$ is meant to represent a compatible subset of δ-clauses that allows for deriving the 'prerequisite' α_δ. In this way, $C_{D''}$ can be seen as the 'default proof' of α_δ. Condition (3bi) corresponds to Condition (3b) in Definition 5.1; here it is restricted to the δ-clauses in $C_{D''}$. Condition (3bii) 'implements' incremental compatibility. For coherence, the compatibility of δ-clauses in $C_{D''}$ is verified by the part of compl(p, C_W, C_D) accounting for ω-clauses only. For this, we turn all δ-clauses in $C_{D''} \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}$ into ω-clauses and add the latter ones to the original set of ω-clauses in C_W. In this way, we make use of Theorem 4.2 and pass the matrix $C_W \cup C_{D''} \cup \{\{\neg\alpha_\delta\}\}$ – whose admissibility is verified in Condition (3bi) – to the compatibility check in (3bii). There, the clause $\{\neg\alpha_\delta\}$ is extended by γ_δ. Moreover, we push the justifications of the default rules in D'' along with the justification of the considered default rule δ on the path. This additional requirement is obsolete in the case of normal default theories. The failure of compl(Justif$(D'' \cup \{\delta\}), C_W \cup C_{D''} \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}, \varnothing)$ indicates that $W \cup$ Conseq$(D'' \cup \{\delta\}) \cup$ Justif$(D'' \cup \{\delta\})$ is consistent (by completeness of the standard connection method). A minimal condition that is equivalent to Condition (3bii) is the following one:

(ii$'$) compl(Conseq$(D'' \cup \{\delta\}) \cup$ Justif$(D'') \cup \{\delta\}), C_W, \varnothing)$ is false.

The incoherence of this condition to Condition (3bi), however, is not favorable for an efficient algorithm meshing conditions (3bi) and (3bii). However, in the above algorithm, neither condition benefits from the information gathered in the other one, since both are verified separately for the sake of simplicity.

To illustrate the definition of compl(p, C_W, C_D), let us reconsider the derivation given in (5.14) to (5.16). There, we have shown that C is a default conclusion of our initial default theory (2.1). For this, we have shown that compl$(\{\neg C\}, M)$

is true for the 'compatible' matrix $M = \{\{S\}, \{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}\}$. Now, the restriction to 'compatible' matrices is obsolete. Rather we show that

$$\text{compl}(\{\neg C\}, \{\{S\}\}, \{\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}) \text{ is true.}$$

Observe that together, the query clause $\{\neg C\}$, the ω-clause in $\{\{S\}\}$, and the δ-clauses in $\{\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_2}, C_{\delta_2}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}$ form the matrix given in (4.4).

As in Section 5, we select clauses in a connection-driven way. Thus, we select the clause $\{\neg A_{\delta_2}, C_{\delta_2}\}$ since C_{δ_2} is complementary to the literal $\neg C$ on the active path. This establishes Condition (3a). Next, we have to verify Condition (3b). For this, we have to find a subset $C_{D''}$ of the remaining δ-clauses in $\{\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}$ that satisfies Conditions (3bi) and (3bii). For illustration, we direct our subsequent choises along the line sketched by the derivation in (5.14) to (5.16). Accordingly, we choose $C_{D''} = \{\{\neg S_{\delta_1}, A_{\delta_1}\}\}$. This choice along with the previously chosen δ-clause $c = \{\neg A_{\delta_2}, C_{\delta_2}\}$ yields in turn the following evaluations.[25]

1. $\text{compl}(\{\neg A_{\delta_2}\}, \{\{S\}\}, \{\{\neg S_{\delta_1}, A_{\delta_1}\}\})$ is true, since by Condition (3), where $c = \{\neg S_{\delta_1}, A_{\delta_1}\}$ and $C_{D''} = \varnothing$,
   (a) $\text{compl}(\{\neg S_{\delta_1}\}, \{\{S\}\}, \varnothing)$ is true by (2a).
   (b) $\text{compl}(\varnothing, \{\{S\}, \{\neg S_{\delta_1}, A_{\delta_1}\}\}, \varnothing)$ is false by (2b) and (1).
2. $\text{compl}(\{C_{\delta_2}\}, \{\{S\}\}, \{\neg S, A\}, \{\neg A, C\}\}, \varnothing)$ is false, since by repeated applications of (2)
   $\text{compl}(\{C_{\delta_2}\}, S, A, C\}, \varnothing, \varnothing)$ is false.

Items (1) and (2) confirm Condition (3bi) and (3bii) so that our proof of C is completed.

Observe that choosing $C_{D''} = \{\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}$ yields the same result, while the choices $C_{D''} = \varnothing$ or $C_{D''} = \{\{\neg A_{\delta_3}, E_{\delta_3}\}\}$ lead to a failure. One possibility for choosing $C_{D''}$ is to consider ever-increasing subsets of the given set of δ-clauses $C_{D'}$. Another, more promising-possibility is to leave the choice of $C_{D''}$ to the admissibility check in (3bi). In concrete terms, this can be accomplished by passing all δ-clauses in $C_{D'}$ to Condition (3bi) and adding an additional argument to $\text{compl}(p, C_W, C_D)$ in order to account for the δ-rules that are actually used for establishing admissibility in (3bi). Then, the returned set of δ-clauses is checked for compatibility in (3bii). Such an approach is described in Section 8.

For a complement, consider the default rules $(S : A)/A$ and $(A : \neg S)/E$ along with the fact S. For answering the query E, we have to check whether

$$\text{compl}(\{\neg E\}, \{\{S\}\}, \{\{\neg S_{\delta_1}, A_{\delta_1}\}, \{\neg A_{\delta_3}, E_{\delta_3}\}\}) \text{ is true.}$$

For solving the negated query $\neg E$, we have to select clause $\{\neg A_{\delta_3}, E_{\delta_3}\}$ in Condition (3). This, however, requires by Condition (3bii), where the active path $\{\neg S\}$ is formed by the justification of the default rule $(A : \neg S)/E$, that

$$\text{compl}(\{\neg S\}, \{\{S\} \cup C_{D''} \cup \{\{\neg A_{\delta_3}, E_{\delta_3}\}\}\}, \varnothing) \text{ is false}$$

Algorithm	Method	Pruning by	
		admissibility	compatibility
A	M	incremental	compilation
A'	M	incremental	additional test
A^i	M^i	incremental	incremental

Fig. 1. A summary of the algorithmic approaches.

for some δ-clauses $C_{D''}$. This is impossible, however, since any path through the underlying matrix contains the connection $\{S, \neg S\}$. That is, the justification $\neg S$ is inconsistent with the set of facts $\{S\}$.

Finally, we obtain the following result showing that our incremental algorithm is correct and complete for query answering in constrained default logic:

THEOREM 6.4. *Let (D, W) be a default theory in atomic format, and let φ be an atomic formula. Then, $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff* compl$(\{\neg\varphi\}, C_W, C_D)$ *is true, where C_W is the matrix of W and C_D is the matrix of W_D.*

7. Discussion and Related Work

In this section, we summarize the different versions of our approach developed in the preceding sections. Afterwards, we compare our approach with other proposals found in the literature.

Our approach integrates the distinguishing features of default logics into a classical deduction method. This allows for a homogeneous characterization and treatment of default proofs at the level of the calculus. In this way, there are no limits for interactions between the three notions of complementary, admissibility, and compatibility – corresponding to classical deduction and the concepts of groundedness and consistency in default logic. Our basic method, say M, relies on Theorem 3.1 and provides with admissibility and compatibility two independent concepts restricting default proofs to classical proofs confirming the two previous properties. To a turn, we refined our approach by meshing together the concepts of admissibility and (incremental) compatibility. While the resulting method, say M^i, leaves admissibility unaffected, it offers an incremental approach to compatibility in which the consistent usage of δ-clauses is gradually verified. Hence, the conceptional difference between the two methods M and M^i rests on the treatment of consistency. While M relies on a rather global notion of consistency, M^i employs an incremental and thus rather local notion of consistency.

Apart from the encoding of default rules as implication, a distinguishing feature of our approach is the formation of sequences of δ-clauses. In M, this formation is mainly affected by the notion of gruundedness, while consistency plays more or less the role of a global constraint. In contrast to this, groundedness and consistency are meshed together in M^i and hence jointly direct the formation of sequences of δ-clauses. In both versions, groundedness and consistency are integrated into the underlying logical calculus. This is another feature distinguishing our methods from others found in the literature.

We have seen that both methods result in algorithms supporting an easy concurrent verification of complementarity and admissibility. In other words, groundedness is enforced while query answering. The concurrent verification of consistency is added to the algorithm derived from M^i. Moreover, we have shown in Section 6 that M^i allows for structure and information sharing while jointly verifying admissibility and (incremental) compatibility. All of the presented algorithms are query oriented. This reflects the idea that the theorem prover is in charge of finding a proof that is gradually confirmed by the concepts of admissibility and compatibility in the course of the proof search. In this way, the proof search is directed by the notions of groundedness and consistency. A summary of the derived algorithms along with their features is given in Figure 1.

We have proposed an 'off-line' integration of compatibility in order to minimize modifications while using existing automated theorem provers for implementing our method. This has resulted in the algorithm given in Definition 5.1, say A, which allows for simultaneously verifying complementary and admissibility. This approach is derived from our basic method M that allows for separating the concepts of admissibility and compatibility. Algorithm A expects matrices stemming from compatible default theories. In this way, it never runs into redundant computations with incompatible defaults. This approach is advantageous over all others whenever there are few compatible default theories. In such a case, the derivation of a query is not 'distracted' by any consistency checking, since all conflicts have been 'compiled away'. In fact, we can verify whether a query is in the unique extension of a compatible default theory without any consistency checks. In the worst case, however, there may be an exponential number of compatible default theories.

A simple alternative to compiling default theories is described in Corollary 5.3. This variant, say A', uses algorithm A but refrains from compiling consistency and rather verifies compatibility for each completed, admissible proof. This approach shares with A the advantage of minimizing changes to existing automated theorem provers. Also, it avoids representing a possibly exponential number of compatible default theories. However, it is a 'generate and test' approach in principle, in which we generate admissible proofs and verify their compatibility afterwards. In the worst case, however, this belated compatibility check may have to be performed exponentially many times. Consequently, A' may cause a lot of redundancy, since incompatible proofs are not avoided in the course of the proof.

Algorithm A^i is derived from M^i and hence integrates the verification of compatibility (cf. Definition 6.2). As A', this approach is not restricted to matrices stemming from compatible default theories. Rather each δ-clause has to confirm admissibility and compatibility when entering a proof. In this way, redundant computations with incompatible defaults are avoided. In general, this approach is advantageous over the other ones whenever a default theory contains many conflicting default rules. Then, the additional costs of repeated consistency checks can be amortized by pruning many incompatible subproofs. The disadvantage of this approach is, however, that it requires more modifications to existing automated theorem provers. This renders algorithm A^i orthogonal to A (and A'). Also, the successive consistency checks may slow down the performance of the prover. This applies in particular to domains where proofs are usually built from compatible default rules only. A promising way of avoiding this is to use model-checking techniques as described in [43] (see also Section 9).

In all, we argue that the concurrent verification of admissibility while query answering is indispensable. In particular, we have demonstrated that this can be done with few modifications to an automated theorem prover. The treatment of compatibility is more subtle. Here, a lot depends on the underlying theory. That is, if a default theory comprises a feasible number of compatible default theories, then a precompilation of compatibility is favorable. Otherwise, that is, if a default theory comprises too many compatible default theories, an integration of compatibility is preferable, as done in A^i. This has been confirmed by our experimental studies (see Section 8). On the whole, the common general idea of all versions of our approach is to employ a goal-directed search for a proof while minimizing redundancy. We will see below that other approaches exhibit a much stronger separation of the notions of derivability, groundedness, and consistency, which often causes much more redundancy.

A related approach is given in [30] for normal default theories. The first conceptual difference is given by the encoding of default rules. While we convert default rules into standard implications (along with some qualifying conditions), Reiter considers initially only their consequents. The latter requires a separate verification of the prerequisites, which finally leads to an iterative format of default proofs: Given a default theory (D, W) along with a query φ, the idea is to determine a subset D_0 of D such that $W \cup \mathrm{Conseq}(D_0) \vdash \varphi$. Next, the problem is to determine a set $D_1 \subseteq D$ such that $W \cup \mathrm{Conseq}(D_1) \vdash \mathrm{Prereq}(D_0)$. This process is iterated until $D_k = \varnothing$ for some k. In a final step, it is checked whether $W \cup \bigcup_{i=0}^{k} \mathrm{Conseq}(D_i)$ is consistent. In this way, a default proof consists of numerous successive proofs depending on the number of iterations required for verifying the respective prerequisite. This leads to meta-theoretic characterization of defaults proofs, since the approach steps outside the underlying calculus, namely, linear resolution. This is due to the iterative format of default proofs and the separation of consistency checking. In addition, this is a very rigid way of query answering. This is so because each derivation of the form

$W \cup \text{Conseq}(D_i) \vdash \text{Prereq}(D_{i-1})$ has to be completed before one can carry on with the next step. In contrast, our approach integrates groundedness and consistency into the underlying calculus by taking advantage of structure-oriented theorem proving. This allows for compacting the aforementioned iterations into a single default proof. In this way, our approach imposes no format on the way we proceed for answering a given query. In all, the notion of a default proof manifests a second conceptional difference to Reiter's approach.

From an algorithmic perspective, we observe that Reiter's procedure is also query oriented and in favor of a separate confirmation of consistency, as algorithm A′. That is, consistency is verified after a proof has been completed. As argued above, this causes a lot of redundancy in the presence of many proofs built upon incompatible defaults. In particular, one might have to generate numerous proofs before consistency is confirmed or even denied. Also, in the top-down resolution procedure of [30] groundedness is not explicitly taken into account, so that the proof procedure may exhibit a nonterminating behavior when faced with cyclic default rules, like $(A : C)/C$ and $(C : A)/A$ (and nothing else).[26] Such cyclic inferences are not possible within our algorithm, since groundedness is explicitly checked.

Camilla Schwind introduced in [41] a tableau-based method for computing extensions of normal default theories. This work has been extended in [42] to general default theories. Vincent Risch has adapted the approach in [32] to Łukaszewicz's variant of default logic [22].

In [42] a tableau-based theorem prover is used to construct a set of generating defaults $D' \subseteq D$ of an extension of a default theory (D, W) in Reiter's default logic. The idea is to start from a tableau, or simply a matrix in our jargon, representing the set of facts, W, and the consequences of all default rules in the original set of default rules, D. Then, consequents of default rules are successively removed until there is an open path through the resulting matrix of $W \cup \text{Conseq}(D')$ (where all defaults in $D \backslash D'$ have been removed). This indicates that $W \cup \text{Conseq}(D')$ is consistent. Observe that this procedure amounts to computing compatible default theories (cf. Section 5). Accordingly, we may obtain exponentially many tableaux, each representing a compatible default theory. Next, each default in D' and $D \backslash D'$ is inspected. For each $\delta \in D'$, it is checked that its prerequisite is derivable from the matrix of $W \cup \text{Conseq}(D')$ and that its justification is consistent wrt to the same matrix. For each $\delta \in D \backslash D'$, one of the two previous conditions must fail. The inspection of the default rules in $D \backslash D'$ is obsolete for normal default rules. In such a case, also the consistency of the justification of the default rules in D' is guaranteed since then we are reasoning from a 'compatible' set of default rules. In a final step, groundedness is verified for the default rules in D'. This is done in separation from the previous steps.

The logical basis of Schwind's approach is a (fixed-point) criterion resem bling the one given in Theorem 3.1. However, this characterization requires the inspection of all default rules in D due to the lack of semi-monotonicity in regular default logic. The conceptional differences to our approach are the following. First, default rules are represented by their consequents, which leads necessarily to a separate derivation of their prerequisites. Although our algorithms show a similar treatment of prerequisites, this is not stipulated by their underlying methods. Second, the proof procedure in [42] is merely consistency-driven, i.e., only those steps are performed that ensure the consistency of the formulas under consideration. No query is taken into account because entire extensions are computed. Third, groundedness of D' is checked separately at the end of the computation and thus leads to additional computational costs of the algorithm. As argued in the case of separating consistency, such a 'generate and test' approach may cause a lot of redundancy, since nongrounded proofs are not avoided in the course of the procedure. The same consideration apply to the approach described in [32]. This renders both approaches orthogonal to ours.

In [19] a default theory is transformed into a TMS network [13]. The nodes of the TMS reflect any possible dependency between the formulas of the default theory; thereby the derivability relation is 'coded' into the network. Finally, there exists a one-to-one correspondence between an admissible labeling of such a TMS network and the set of generating defaults of an extension of the default theory. However, this encoding requires the computation of a tremendous amount of derivational dependencies which might not even contribute to the formation of an extension.

In the two last approaches, entire extensions are computed. While the tableau approach is close to classical theorem proving, the TMS approach is completely abstracted from derivability in the base language. Once all extensions are computed using any of the two approaches, query answering may be performed using a small number of steps (within the respective framework). In the worst case, however, all – and there may be exponentially many – extensions have to be considered for query answering. Other approaches for computing entire extensions of default theories in Reiter's default logic are described in [15, 46]. Both approaches use approximation techniques for finding a set of generating default rules while abstracting from the underlying theorem prover. In [9], it is shown that this leads to an exponential amount of space in order to avoid non-termination.

With our method no such precomputation of all extensions in their entirety is needed (even though it is possible [33]). On the other hand, algorithm A may be seen as compromising the 'off-line' computation of compatible sets of defaults with 'on-line' query answering. So our general idea is to employ a goal-directed search for a proof while minimizing redundancy.

A number of other algorithms have been conceived for restricted classes of defaults. In particular, prerequisite-free normal default theories are attractive candidates for implementations, since they do not exhibit dependencies between the prerequisites of defaults. Thus, there is no need to enforce groundedness. Query answering in our approach then reduces to query answering in classical logic with an additional consistency check. Other implementations for this fragment of default logic exist [2, 29, 4].

Niemelä provides in [27, 28] different methods for nonmonotonic reasoning based on autoepistemic reasoning. This and other approaches to autoepistemic logic [24] are in principle adaptable to default logics via certain transformations between default and autoepistemic logic. The discussion of these approaches is, however, beyond the scope of this paper.

8. Experiments

In this section, we present prototypical implementations of the algorithms described in the previous sections. The purpose of this is twofold. First, we wish to show how these approaches can be implemented. Second, we wish to provide some experimental analysis.

We have seen how easily admissibility and compatibility are simultaneously verifiable. Hence, it will be interesting to investigate whether this is a feasible process. Moreover, we explore the issue of compatibility by comparing the results obtained in the diverse approaches. In all, we wish to know which modifications to an existing automated theorem provers are worthwhile under which circumstances.

We have refrained from using any advanced implementation techniques in order to keep the exposition transparent. Thus, the prototypes are intended to provide transparent case studies for certain default reasoning architectures rather than efficient implementations. For instance, none of the given programs makes use of structure sharing, as suggested in Section 6. Moreover, such enhancements would influence the different settings in different ways so that the corresponding results would be hardly comparable. Rather we use in each implementation the same classical inference mechanism and the same way of consistency checking. This allows for a simple common implementation platform. For simplicity, we restrict ourselves to normal default theories.

8.1. A STRAIGHTFORWARD IMPLEMENTATION

We start by giving an extremely simple and straightforward PROLOG implementation of Eder's algorithm for the standard connection method [14]. This implementation serves two purposes. First, it provides the basic theorem proving techniques that we will use for our prototypical implementations. Second, it supplies us with a simple way for consistency checking.

The implementation given in Figure 2 corresponds to the first two conditions given in Definition 5.1. Matrices are represented by lists of lists of literals. A literal is the form a or $-$a. The first program clause of `compl/2` implements Condition (1) in Definition 5.1, while the second one selects clauses out of the

```
compl(_Path, []      ) :- !,fail.
compl( Path, Matrix ) :-
        select(Clause,Matrix,MatrixRest),!,
        complC(Path,Clause,MatrixRest).

complC(_Path, []                     ,_Matrix ) .
complC( Path, [Literal|ClauseRest], Matrix ) :-
        complL(Path,Literal),!,
        complC(Path,ClauseRest,Matrix).
complC( Path, [Literal|ClauseRest], Matrix ) :-
        compl([Literal|Path],Matrix),
        complC(Path,ClauseRest,Matrix).

complL( Path, Literal ) :-
        neg(Literal,NegLiteral),!,
        member(NegLiteral,Path).
```

Fig. 2. An implementation of Eder's algorithm.

matrix M and initiates their treatment in complC(Path,Clause,MatrixRest) according to Condition (2) in Definition 5.1. complC/3 verifies that each literal in a considered clause satisfies either Condition (2a) or (2b). That is, while the first program clause of complC/3 captures the limiting case, the second one accounts for Condition (2a) and the third one accounts for Condition (2b). Finally, complL/2 checks whether the negation of the literal Literal is a member of the path Path. Auxiliary program clauses, like neg/2, are given in Figure 3. The predicates member/2, select/3, etc. have their obvious meaning and belong to the underlying PROLOG system.

As mentioned above, we refrain from using any advanced implementation techniques in order to provide a simple common implementation platform. For this purpose, we verify the compatibility of a set of δ-clauses relative to a set of ω-clauses by appeal to compl/2. This results in the predicate compatible/2 given in Figure 4. compatible/2 takes a list of ω-clauses CW and a list of δ-clauses CD, unions CW with the consequents of the δ-clauses in CD, and checks whether the result satisfies not(compl([],M)). This amounts to a classical satisfiability test.

8.2. Implementations separating compatibility

In this section, we give implementations that simultaneously verify admissibility and complementarity but separate the verification of compatibility. That is, we discuss implementations of the algorithm given in Definition 5.1.

For illustration, let us start by introducing the underlying representation. Consider the default theory

$$\left(\left\{\frac{S:A}{A}, \frac{S:\neg E}{\neg E}, \frac{A:C}{C}, \frac{A:E}{E}\right\}, \{S\}\right).$$

```
neg( -Literal,  Literal ) :- !.
neg(  Literal, -Literal ) .

split( [],                    [],          []          ) .
split( [Clause|MatrixRest], CW           , [Clause|CD] ) :-
        d_clause(Clause),!,
        split(MatrixRest,CW,CD).
split( [Clause|MatrixRest], [Clause|CW], CD           ) :-
        w_clause(Clause),!,
        split(MatrixRest,CW,CD).

d_clause( [ _AlphaLiteral > _GammaLiteral ] ).
w_clause( Clause                          ) :-
        not(d_clause(Clause)).

omega( []                   , []             ) .
omega( [[_Alpha>Gamma]|CDRest], [[Gamma]|CWRest] ) :-
        omega(CDRest,CWRest).
```

Fig. 3. Auxiliary program clauses.

```
compatible( CW, CD ) :-
        omega(CD,CDOmega),
        union(CW,CDOmega,M),
        not(compl([],M)).
```

Fig. 4. A simple way of checking compatibility.

The representation of this default theory is given in the left column of Figure 5. As mentioned above, ω-clauses are lists of literals. δ-clauses like $\{\neg S, A\}$ are represented as [s > a]. This allows for an easy distinction between ω- and δ-clauses and moreover between 'prerequisites' and 'consequents' of δ-clauses. The compatible counterpart of the default theory is given in the right column of Figure 5. For transparency, we have chosen this naive representation rather

```
sample(students,[                    sample(studentsC,[
        [s],                                 [s],
        [s>a],                               [s>a],
        [s> -e],                             [s> -e],
        [a>c],                               [a>c]
        [a>e]                                ]).
        ]).                          sample(studentsC,[
                                             [s],
                                             [s>a],
                                             [a>c],
                                             [a>e]
                                             ]).
```

Fig. 5. The representation of our example.

```
compl( Path, M, Proof ) :-
        split(M,CW,CD),
        compl(Path,CW,CD,CW,Proof).

compl(_Path, [], [], _,_Proof ) :- !,fail.
compl( Path, CW, CD, M, Proof ) :-
        select(OmegaClause,CW,CWRest),
        select(Literal,OmegaClause,OmegaClauseRest),
        complL(Path,Literal),
        complW(Path,OmegaClauseRest,CWRest,CD,M,Proof).
compl( Path, CW, CD, M, Proof ) :-
        select([Alpha>Gamma],CD,CDRest),
        complL(Path,Gamma),!,
        complD(Path,[Alpha>Gamma],CW,CDRest,M,Proof).

complW(_Path, []                    ,_CW,_CD,_M, []    ) .
complW( Path, [Literal|ClauseRest], CW, CD, M, Proof ) :-
        complL(Path,Literal),
        complW(Path,ClauseRest,CW,CD,M,Proof).
complW( Path, [Literal|ClauseRest], CW, CD, M, Proof ) :-
        compl([Literal|Path],CW,CD,M,Proof1),
        complW(Path,ClauseRest,CW,CD,M,Proof2),
        append(Proof1,Proof2,Proof).

complD(_Path, [Alpha>Gamma],_CW, CD, M, [[Alpha>Gamma]|Proof] ) :-
        neg(Alpha,NegAlpha),
        compl([NegAlpha],M,CD,M,Proof).
```

Fig. 6. I: An implementation of Algorithm A.

than a more efficient one where each δ-clause in represented only once. Such a representation is used in [33].

The implementation, I say, of the algorithm in Definition 5.1 is given in Figure 6. As in Definition 6.2, we have separated ω- and δ-clauses. Moreover, we have extended each predicate by two additional arguments: one containing the original set of ω-clauses, and another one accumulating the proof of the query. Observe that the counterpart of compl/2 in Figure 2 is given by compl/5. The principal difference is that compl/5 selects ω- as well as δ-clauses in a connection-driven way. That is, a clause is selected only if one of its literals is complementary to a literal on the active path. The counterpart of complC/3 in Figure 2 is given by complW/6 and complD/6. complW/6 treats ω-clauses and is identical to complC/3 in Figure 2. δ-clauses are processed by complD/6. Since the complementarity of the consequent Gamma to one of the literals on the active path is checked in compl/5, merely Condition (3b) in Definition 5.1 remains to be verified by complD/6. This is done by compl([NegAlpha], M, CD, M, Proof). Observe that this is the point where the original set of ω-clauses M 'reenters' the proof. This is necessary for checking admissibility. If the last subgoal in complD/6 succeeds, the δ-clause [Alpha > Gamma] is added to the proof in Proof.

A query like e is posed to the 'knowledge base' studentsC[27] in Figure 5 by evaluating the following PROLOG query:

? – sample(studentsC, M), compl([–e], M, P).

This yields the answer:

P = [[a > e], [s > a]].

We have tested the implementation on numerous examples. First of all, however, let us consider the results on the studentsC example given in Figure 5. All results on this example are summarized in Figure 7. The left column of tables given there shows the results on querying –e; the right one does the same for the query e. The tables have the following format[28]:

calleeName/calleeArity callerName/callerArity count

The number given in count expresses the number of calls of calleeName from callerName. Note that we keep track only of callers inside the considered implementation. In this way, we discard for instance all calls from predicates in the program given in Figure 2.

The first line of tables in Figure 7 gives the results of implementation I. It is worth noticing that compl/5 has been called two times from compl/3 on the query e. This expresses the problem with our naive representation of compatible default theories. In fact, the program tries to prove e first from the first compatible default theory in Figure 5, which is impossible. Afterwards, it finds a proof by looking at the second compatible default theory. Usually, connection method theorem provers employ so-called connection graphs that indicate the location of clauses containing complementary literals. Clearly, such a data structure would avoid this problem in our example.

For comparison, we have also implemented algorithm A′. Its implementation, say I′, is obtained by replacing the definition of compl/3 in Figure 6 by the one given in Figure 8. The results on the students example are given in the second line of tables in Figure 7. Observe that compatible/2 is called only once. This indicates that both proofs were initially found. That is, each of them was formed by a set of compatible default rules. We will take up this algorithmic variant of A in the next subsection.

The major question addressed in this subsection is whether the integration of admissibility slows the underlying theorem prover. To answer this question, let us consider the exemplary (compatible) matrices az and pyramid along with their classical counterparts azW and pyramidW obtained by replacing each δ-clause like [a > b] by a Hornclause like [–a, b]. The corresponding 'knowledge bases' are listed in Figure 9. In turn, we query both the 'default knowledge bases' and their classical counterparts with the same query. The results given in Figure 10 speak for themselves. In fact, we observe that the 'default proofs' (in the left column of Figure 10) need in all respects less counts than their classical counterparts (in the right column). This is a strong argument in favor of our approach to 'on-line' admissibility (or groundedness) checking.

8.3. Implementations Integrating Compatibility

This section gives implementations that integrate the verification of compatibility. Figure 11 contains an implementation of algorithm A^i, as described in Definition 6.2. This implementation is obtained from the one given in Figure 6 by modifying the definitions of `complW/6` and `complD/6`. As stipulated in Condition (2bii) in Definition 6.2, we extended `complW/6` in order to check the compatibility of all accomplished subproofs. This modification is given by the last two lines of `complW/6` in Figure 11. Analogously, we have added a single line of code to `complD/6` in order to check the compatibility of the considered δ-clause with the accomplished subproof. The results obtained in our simple students example are the same as for the previous implementations with the exception that `compatible/2` is called twice in the case of the query `e`. This reflects the fact that two δ-clauses are used for deriving `e`. See the tables in the third line of Figure 7.

It is interesting to observe that the implementation avoids the selection of the sets of δ-clauses $C_{D'_L}$ and $C_{D''}$ in conditions (2bii) and (3bii), respectively. As suggested in Section 6, we leave the selection of these sets to the theorem prover along with the admissibility check by taking the δ-clauses obtained from the subproof in `Proof` in `complW/6` and `complD/6`. In this way, compatibility acts as a local constraint on subproofs. In fact, we often observed that default proofs in non-artificial examples were quite rarely corrected by the compatibility check. Hence, the resulting default proofs contained only few occasions for distracting the theorem prover by choosing incompatible δ-clauses.

For comparison, we have also implemented a 'generate-and-test' version, called I^i_s, in which arbitrary subsets are generated and afterwards treated by admissibility and compatibility. This has been done by replacing the definition of `complD/6` in Figure 11 by the one given in Figure 12. The predicate `subseq0/2` is provided by the underlying PROLOG system and is used to generate subsets (or better subsequences) of the δ-clauses at hand. Even in our simple students example this yields a drastic increase of calls to the respective predicates. This can be verified by looking at the tables in the last line of Figure 7 and comparing them with the ones given for I^i.

In the remainder of this section, we analyze the influence of the compatibility check. For this purpose, we have slightly extended example `az`, given Figure 9. We have added the ω-clauses

$$[-\mathtt{c}, -\mathtt{b}], [\mathtt{z}, -\mathtt{c}], [\mathtt{z}, -\mathtt{g}], [\mathtt{z}, -\mathtt{k}], [\mathtt{z}, -\mathtt{o}], [\mathtt{z}, -\mathtt{s}], [\mathtt{z}, -\mathtt{w}]$$

to `az`. We call the resulting example `azno`. The purpose of the six ω-clauses containing the literal `z` is to provide several ways of proving `z`. However, all of them are denied because `c` and `b` are inconsistent by $[-\mathtt{c}, -\mathtt{b}]$. Hence, there is no proof for `z`. The left column of Figure 13 summarizes the results obtained in the various implementations. It is interesting to observe that our implementation

I	studentsC/-e	
compl/5	compl/3	1
compl/5	complD/6	1
complD/6	compl/5	1
complL/2	compl/5	4
complW/6	compl/5	1
neg/2	complD/6	1
neg/2	complL/2	4

I	studentsC/e	
compl/5	compl/3	2
compl/5	complD/6	2
complD/6	compl/5	2
complL/2	compl/5	11
complW/6	compl/5	1
neg/2	complD/6	2
neg/2	complL/2	11

I'	students/-e	
compatible/2	compl/3	1
compl/5	compl/3	1
compl/5	complD/6	1
complD/6	compl/5	1
complL/2	compl/5	4
complW/6	compl/5	1
neg/2	complD/6	1
neg/2	complL/2	6

I'	students/e	
compatible/2	compl/3	1
compl/5	compl/3	1
compl/5	complD/6	2
complD/6	compl/5	2
complL/2	compl/5	8
complW/6	compl/5	1
neg/2	complD/6	2
neg/2	complL/2	11

I^i	students/-e	
compatible/2	complD/6	1
compl/5	compl/3	1
compl/5	complD/6	1
complD/6	compl/5	1
complL/2	compl/5	4
complW/6	compl/5	1
neg/2	complD/6	1
neg/2	complL/2	6

I^i	students/e	
compatible/2	complD/6	2
compl/5	compl/3	1
compl/5	complD/6	2
complD/6	compl/5	2
complL/2	compl/5	8
complW/6	compl/5	1
neg/2	complD/6	2
neg/2	complL/2	13

I^i_s	students/-e	
compatible/2	complD/6	4
compl/5	compl/3	1
compl/5	complD/6	4
complD/6	compl/5	1
complL/2	compl/5	7
complW/6	compl/5	4
neg/2	complD/6	1
neg/2	complL/2	21

I^i_s	students/e	
compatible/2	complD/6	4
compl/5	compl/3	1
compl/5	complD/6	8
complD/6	compl/5	3
complL/2	compl/5	19
complW/6	compl/5	2
neg/2	complD/6	3
neg/2	complL/2	34

Fig. 7. Results in the student example.

```
compl( Path, M, Proof ) :-
        split(M,CW,CD),
        compl(Path,CW,CD,CW,Proof),
        compatible(CW,Proof).
```

Fig. 8. I': The change for algorithm A'.

I^i (integrating compatibility) invokes fewer times the predicate neg/2 than I' (which separates compatibility) – even though I^i performs almost twice as much compatibility checks, namely, 3914, as I', which performs 1957 compatibility checks. This is remarkable since neg/2 is one of the innermost predicates of the 'theorem-proving loop'. In this way, the incremental compatibility check pays off. The least number of calls to neg/2 is done by program I that deals with 'compatible matrices'.

```
sample(az,[
          [a],
          [a>b],
          [b>c],
          [c>d],
          [d>e],
          [e>f],
          [f>g],
          [g>h],
          [h>i],
          [i>j],
          [j>k],
          [k>l],
          [l>m],
          [m>n],
          [n>o],
          [o>p],
          [p>q],
          [q>r],
          [r>s],
          [s>t],
          [t>u],
          [u>v],
          [v>w],
          [w>x],
          [x>y],
          [y>z]
          ]).
```

```
sample(azW,[
          [a],
          [-a,b],
          [-b,c],
          [-c,d],
          [-d,e],
          [-e,f],
          [-f,g],
          [-g,h],
          [-h,i],
          [-i,j],
          [-j,k],
          [-k,l],
          [-l,m],
          [-m,n],
          [-n,o],
          [-o,p],
          [-p,q],
          [-q,r],
          [-r,s],
          [-s,t],
          [-t,u],
          [-u,v],
          [-v,w],
          [-w,x],
          [-x,y],
          [-y,z]
          ]).
```

```
sample(pyramid,[
          [a1],
          [a2],
          [a3],
          [a4],
          [a5],
          [a6],
          [a7],
          [a8],
          [a1>b1],
          [a2>b2],
          [-b1,-b2,c12],
          [c12>d12],
          [a3>b3],
          [a4>b4],
          [-b3,-b4,c34],
          [c34>d34],
          [-d12,-d34,e1234],
          [-e1234,f1234],
          [a5>b5],
          [a6>b6],
          [-b5,-b6,c56],
          [c56>d56],
          [a7>b7],
          [a8>b8],
          [-b7,-b8,c78],
          [c78>d78],
          [-d56,-d78,e5678],
          [-e5678,f5678],
          [-f1234,-f5678,g]
          ]).
```

```
sample(pyramidW,[
          [a1],
          [a2],
          [a3],
          [a4],
          [a5],
          [a6],
          [a7],
          [a8],
          [-a1,b1],
          [-a2,b2],
          [-b1,-b2,c12],
          [-c12,d12],
          [-a3,b3],
          [-a4,b4],
          [-b3,-b4,c34],
          [-c34,d34],
          [-d12,-d34,e1234],
          [-e1234,f1234],
          [-a5,b5],
          [-a6,b6],
          [-b5,-b6,c56],
          [-c56,d56],
          [-a7,b7],
          [-a8,b8],
          [-b7,-b8,c78],
          [-c78,d78],
          [-d56,-d78,e5678],
          [-e5678,f5678],
          [-f1234,-f5678,g]
          ]).
```

Fig. 9. The az and the pyramid example.

az/z		
compl/5	compl/3	1
compl/5	complD/6	25
complD/6	compl/5	25
complL/2	compl/5	351
complW/6	compl/5	1
neg/2	complD/6	25
neg/2	complL/2	351

azW/z		
compl/5	compl/3	1
compl/5	complW/6	25
complL/2	compl/5	676
complL/2	complW/6	25
complW/6	compl/5	26
complW/6	complW/6	25
neg/2	complL/2	701

pyramid/g		
compl/5	compl/3	1
compl/5	complD/6	12
compl/5	complW/6	16
complD/6	compl/5	12
complL/2	compl/5	653
complL/2	complW/6	16
complW/6	compl/5	17
complW/6	complW/6	16
neg/2	complD/6	12
neg/2	complL/2	669

pyramidW/g		
compl/5	compl/3	1
compl/5	complW/6	28
complL/2	compl/5	731
complL/2	complW/6	28
complW/6	compl/5	29
complW/6	complW/6	28
neg/2	complL/2	759

Fig. 10. Results in the az and pyramid example.

Observe that in this example the conflict given by the clause $[-\mathtt{c}, -\mathtt{b}]$ is located at the 'bottom' of the sequence of δ-clauses given in example az. For a complement, let us thus consider the example obtained by adding the clauses

$$[-\mathtt{i}, -\mathtt{j}], [\mathtt{z}, -\mathtt{k}], [\mathtt{z}, -\mathtt{o}], [\mathtt{z}, -\mathtt{s}], [\mathtt{z}, -\mathtt{w}]$$

```
compl( Path, M, Proof ) :-
        split(M,CW,CD),
        compl(Path,CW,CD,CW,Proof).

compl(_Path, [], [], _,_Proof ) :- !,fail.
compl( Path, CW, CD, M, Proof ) :-
        select(OmegaClause,CW,CWRest),
        select(Literal,OmegaClause,OmegaClauseRest),
        complL(Path,Literal),
        complW(Path,OmegaClauseRest,CWRest,CD,M,Proof).
compl( Path, CW, CD, M, Proof ) :-
        select([Alpha>Gamma],CD,CDRest),
        complL(Path,Gamma),!,
        complD(Path,[Alpha>Gamma],CW,CDRest,M,Proof).

complW(_Path, []                    ,_CW,_CD,_M, []     ) .
complW( Path, [Literal|ClauseRest], CW, CD, M, Proof ) :-
        complL(Path,Literal),
        complW(Path,ClauseRest,CW,CD,M,Proof).
complW( Path, [Literal|ClauseRest], CW, CD, M, Proof ) :-
        compl([Literal|Path],CW,CD,M,Proof1),
        complW(Path,ClauseRest,CW,CD,M,Proof2),
        append(Proof1,Proof2,Proof),
        compatible(M,Proof).

complD(_Path, [Alpha>Gamma],_CW, CD, M, [[Alpha>Gamma]|Proof] ) :-
        neg(Alpha,NegAlpha),
        compl([NegAlpha],M,CD,M,Proof),
        compatible(M,[[Alpha>Gamma]|Proof]).
```

Fig. 11. I^i: An implementation of algorithm A^i.

```
complD(_Path, [Alpha>Gamma],_CW, CD, M, [[Alpha>Gamma]|Proof] ) :-
        neg(Alpha,NegAlpha),
        subseq0(CD,CD1),
        compl([NegAlpha],M,CD1,M,Proof),
        compatible(M,[[Alpha>Gamma]|CD1]).
```

Fig. 12. The change for implementation I^i_s.

to az. In this way, we moved the conflict – now given by the clause $[-\mathtt{i}, -\mathtt{j}]$ – upwards in the sequence of δ-clauses (cf. Figure 9). We call the resulting example aznon. The right column of Figure 13 summarizes the obtained results. We observe that the number of compatibility checks in I^i is now nine times larger than in I'. This has to be contrasted with the number of calls to neg/2 that is in I^i only 1.2 times larger than in I'. As above, the least number of calls to neg/2 is obtained in program I.

We observe that program I performs best on the given examples because of the lack of consistency checking. However, we have already discussed that this approach has its difficulties in the presence of many conflicting default rules. In such a case, our experiments suggest that it is worthwhile to integrate compatibility checks in order to prune redundant incompatible subproofs, rather than to check compatibility separated from the actual proof procedure. The first of the two examples extending az gives an impression of situations where incremental compatibility checking pays off.

9. Extensions

This section provides some enhancements and extensions of our approach. A major extension, namely, the incorporation of priorities, is described in [39]. Another major extension is the treatment of skeptical reasoning. This is discussed in a forthcoming paper [40].

9.1. Other variants of default logic

As discussed in Section 2, constrained default logic coincides with other default logics, like Reiter's [30] and Łukaszewicz's [22], on the fragment of normal default theories. As a consequence, our approach can also be used for query answering from normal default theories in these variants.

A closely related approach is cumulative default logic [5] where assertions, that is, formulas labeled with the justifications and consequents of applied default rules (e.g., $\langle\alpha, \{\alpha_1, \dots, \alpha_n\}\rangle$), are used. An assertional default theory is a pair $(D, \mathcal{W})$, where D is a set of default rules and $\mathcal{W}$ is a set of assertions. Informally, an assertional extension of $(D, \mathcal{W})$ is the smallest set of assertions $\mathcal{E}$ being deductively closed under an extended[29] theory operator $\widehat{Th}$ and containing $\mathcal{W}$ such that for any $(\alpha : \beta)/\gamma \in D$, if $\langle\alpha, \mathrm{Supp}(\alpha)\rangle \in \mathcal{E}$ and $\mathrm{Form}(\mathcal{E}) \cup \mathrm{Supp}(\mathcal{E}) \cup \{\beta, \gamma\} \nvdash \bot$, then $\langle\gamma, \mathrm{Supp}(\alpha) \cup \{\beta, \gamma\}\rangle \in \mathcal{E}$.

The following theorem shows that constrained and cumulative default logic are in fact equivalent for assertional default theories $(D, \mathcal{W})$ having nonsupported facts, i.e., $\mathrm{Supp}(\mathcal{W}) = \varnothing$.

THEOREM 9.1 ([37]). *Let (D, W) be a default theory and $(D, \mathcal{W})$ be the assertional default theory, where $\mathcal{W} = \{\langle\alpha, \varnothing\rangle \mid \alpha \in W\}$. Then, if (E, C) is a constrained extension of (D, W), there is an assertional extension $\mathcal{E}$ of $(D, \mathcal{W})$ such that $E = \mathrm{Form}(\mathcal{E})$ and $C = Th(\mathrm{Form}(\mathcal{E}) \cup \mathrm{Supp}(\mathcal{E}))$; and, conversely if $\mathcal{E}$ is an assertional extension of $(D, \mathcal{W})$, then $(\mathrm{Form}(\mathcal{E}), Th(\mathrm{Form}(\mathcal{E}) \cup \mathrm{Supp}(\mathcal{E})))$ is a constrained extension of (D, W).*

This result has been extended in [38] to arbitrary assertional default theories and default theories supplied with an initial set of constraints in constrained default logic.

As a consequence, we can use our approach for query answering from assertional default theories with nonsupported facts without any modifications. For arbitrary assertional default theories, we merely have to add the formulas in

I	aznoC/z	
compl/5	compl/3	2
compl/5	complD/6	13098
compl/5	complW/6	3912
complD/6	compl/5	13098
complL/2	compl/5	348304
complL/2	complW/6	3912
complW/6	compl/5	3912
neg/2	complD/6	13098
neg/2	complL/2	352216

I	aznonC/z	
compl/5	compl/3	2
compl/5	complD/6	385
compl/5	complW/6	128
complD/6	compl/5	385
complL/2	compl/5	11848
complL/2	complW/6	128
complW/6	compl/5	128
neg/2	complD/6	385
neg/2	complL/2	11976

I'	azno/z	
compatible/2	compl/3	1957
compl/5	compl/3	1
compl/5	complD/6	5573
compl/5	complW/6	1956
complD/6	compl/5	5573
complL/2	compl/5	152045
complL/2	complW/6	1956
complW/6	compl/5	3913
complW/6	complW/6	9786
neg/2	complD/6	5573
neg/2	complL/2	1053720

I'	aznon/z	
compatible/2	compl/3	65
compl/5	compl/3	1
compl/5	complD/6	745
compl/5	complW/6	64
complD/6	compl/5	745
complL/2	compl/5	14339
complL/2	complW/6	64
complW/6	compl/5	129
complW/6	complW/6	196
neg/2	complD/6	745
neg/2	complL/2	23222

I^{2}	azno/z	
compatible/2	complD/6	3914
compl/5	compl/3	1
compl/5	complD/6	5573
compl/5	complW/6	1956
complD/6	compl/5	5573
complL/2	compl/5	152045
complL/2	complW/6	1956
complW/6	compl/5	3913
neg/2	complD/6	5573
neg/2	complL/2	983769

I^{2}	aznon/z	
compatible/2	complD/6	585
compl/5	compl/3	1
compl/5	complD/6	745
compl/5	complW/6	64
complD/6	compl/5	745
complL/2	compl/5	14339
complL/2	complW/6	64
complW/6	compl/5	129
neg/2	complD/6	745
neg/2	complL/2	28573

Fig. 13. Further results in the az example.

Supp($\mathcal{W}$) for verifying compatibility. The same applies to our algorithms, where clauses representing Supp($\mathcal{W}$) have to be added while checking consistency. The formal underpinnings for this approach are given in [38].

To capture query answering in Łukaszewicz's variant of default logic, we have to adjust the concept of compatibility. All other notions remain the same because this variant enjoys semi-monotonicity.

9.2. INTEGRATION OF LEMMA HANDLING

The integration of lemma handling is of great practical relevance in automated theorem proving. This is so because the use of lemmas is often needed for reducing computational efforts. Since computation in default logics involves not only deduction but also consistency checks, the need to incorporate lemmas is even greater in default theorem proving than in standard theorem proving.

In [37], we introduced an approach to lemma handling in Reiter's and constrained default logic.[30] Inspired by default logic's natural distinction between facts and defaults, default lemmas are regarded as abbreviations for the corresponding default proofs. This results in the concept of a *lemma default rule*:

DEFINITION 9.1 ([37]). Let (D, W) be a default theory, and let (E, C) be a constrained extension of (D, W). Let $\varphi \in E$ and D_φ be a default proof of φ in (E, C) from (D, W). We define a lemma default rule δ_φ for φ as

$$\delta_\varphi = \frac{: \bigwedge_{\delta \in D_\varphi} \mathrm{Justif}(\delta) \wedge \bigwedge_{\delta \in D_\varphi} \mathrm{Conseq}(\delta)}{\varphi}.$$

A default proof D_φ of φ in (E, C) from (D, W) is a subset of the set of generating default rules[31] of (E, C) such that $W \cup \mathrm{Conseq}(D_\varphi) \vdash \varphi$.

At the methodical level, we can generate lemma default rules or 'lemma δ-clauses' for any proven query φ. In such a case, we have spanning mating Π for the matrix M of $W \cup W_D \cup \{\neg\varphi\}$ such that (M, Π) is admissible and compatible. The default proof is then given by

$$D_\varphi = \{\delta \mid \{\neg\alpha_\delta, \gamma_\delta\} \in \kappa(C_D, \Pi)\}.$$

At the algorithmic level, we can generate lemma default rules for each obtained subproof as follows. Consider Condition (3b) in Definition 6.2. A 'lemma δ-clause' for γ_δ can be generated from the default proof

$$D_\varphi = \{\delta \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_{D''}\}.$$

For general formulas φ, it is sufficient to find a subset D_φ of D such that $\mathrm{compl}(\{\neg\varphi\}, C_W, C_{D_\varphi})$ is true. Similar considerations apply to the implementations in Section 8.

The usage of such lemma default rules is obvious at the methodical level, since they constitute ordinary albeit prerequisite-free default rules. The distinction between lemma default rules and standard default rules is more interesting at the algorithmic and implementation level. For using 'lemma δ-clauses', we can simplify Condition (3) in Definition 6.2 as follows:

(4) *If $C_{D'} \neq \varnothing$ and $c \in C_{D'}$ is a lemma δ-clause, then* $\mathrm{compl}(p, C_{W'}, C_{D'})$ *is true iff γ_δ is complementary to some literal of p for $c = \{\neg\alpha_\delta, \gamma_\delta\}$.*

This drastic simplification is possible because neither admissibility nor compatibility has to be verified for 'lemma δ-clauses'. The former is obsolete because lemma default rules are prerequisite-free default rules, while the latter is redundant because the compatibility with C_W has been checked while generating the lemma. The compatibility with the remaining δ-clauses in the proof is checked at a higher level of the recursion. Similar considerations apply to the implementations in Section 8.

9.3. Other Extensions

This section sketches some extensions and fruitful topics for further research.

Computing Extensions. Although we have presented the approach so far in a query-oriented setting, it can also be used for computing entire extensions by proceeding 'bottom up'. This can be accomplished by starting from the clausal representation of the initial set of facts and by successively extending this matrix by δ-clauses preserving admissibility and compatibility. In this way, we obtain a matrix representing the facts along with the set of generating default rules of an extension. Then, a formula belongs to this extension if the matrix is rendered complementary by adding the clausal representation of the negated formula.

Model Checking. A promising avenue for future research seems to replace consistency checking by model checking. The idea is to start with a model of the underlying facts. A default applies if it is compatible with the current model or if a new model can be constructed from the facts and all defaults contributing to the current proof. In this way, a model may be reused for several consistency checks in the course of the proof search. This is motivated by two observations. First, some experiments on 'meaningful' (i.e., nonartificial) examples have shown that a 'model' is changed quite rarely in the course of the proof search. That is, on some examples, we observed that the resulting default proofs contained only few occasions for distracting the theorem prover by choosing incompatible δ-clauses. This is an argument in favor of integrating a compatibility check that allows for using information gathered on compatibility checks in the subproofs. Second, the semantics of constrained default logic [34, 11] stipulates the existence of a single so-called 'focused model' which jointly satisfies the facts along with all generating default rules of an extension.

Other Execution Models. So far, we have pursued only a single execution model. However, our underlying methods leave room for a variety of procedures for query answering. One of them has already been sketched in Section 6, where we discussed an approach to structure sharing while verifying admissibility and compatibility. Also, there are parallel execution models of our algorithms. For instance, observe that all algorithms are nondeterministic and inherently parallel, although parallel processes sometimes have to be combined after their execution as a result of compatibility checks, as in Condition (2bii) in Definition 6.2. Another possibility is to design a 'daemon-driven' compatibility check by using a 'daemon process' to keep a watch on the compatibility of the respective subproofs. All these issues seem to be fruitful avenues for further research.

10. Conclusion

We have presented a new approach to query answering in default logics by treating default rules as classical implications along with some qualifying conditions restricting the use of such rules in the course of the proof search. This has resulted in a novel methodology taking advantage of the conception of structure-oriented theorem proving provided by the connection method. To this end, we have decomposed default theorem proving (in the connection method) into the verification of complementarity, admissibility, and compatibility – corresponding to classical deduction and the concepts of groundedness and consistency in default logic.

We introduced in Section 4 our basic method that provides with admissibility and compatibility two independent concepts restricting default proofs to classical proofs confirming the two previous properties. To a turn, we refined our approach in Section 6 by meshing together the concepts of admissibility and (incremental) compatibility. While the former approach relies on a rather global notion of consistency, the latter employs an incremental and thus rather local notion of consistency.

Apart from the enconding of default rules as implications, a distinguishing feature of our approach is the formation of sequences of default rules or δ-clauses, respectively. In our basic method, this formation is primarily directed by the notion of groundedness, while consistency plays more or less the role of a global constraint. In contrast to this, groundedness and consistency jointly direct the formation of sequences of δ-clauses in our refined method.

We have discussed in detail the different versions of our approach and their differences to other approaches found in the literature in Section 7. To summarize, the distinguishing methodical features of our approach are

- the treatment of default rules as classical implications,
- the formation of sequences of default rules or δ-clauses, respectively, and
- the integration of the concepts of groundedness *and* consistency into a classical deduction method.

These qualities allow for a homogeneous characterization and treatment of default proofs at the level of the logical calculus. This makes our approach especially qualified for implementations by means of existing automated theorem-proving techniques. We have substantiated this claim by implementing the resulting algorithms in diverse settings.

At first, we derived in Section 5 from our basic method an algorithm that supports the joint verification of complementarity and admissibility in a very natural way. Notably, the algorithm is obtained by slightly extending an existing algorithm for the standard connection method due to [14]. However, we have proposed an 'off-line' integration of compatibility by compiling default theories into compatible default theories. This separation is supported by the independence of admissibility and compatibility in our basic method. The advantages of this approach are the following. First, the algorithm never runs into redundant computations with incompatible defaults. Second, it is implementable with very few modifications to an existing automated theorem prover. This is underpinned by a case study due to Aaron Rothschild [33]. In fact, Rothschild was able to implement our algorithm by slightly extending an existing (simple yet full-fledged) automated theorem prover for the connection method. The disadvantage, however, is that one might obtain an exponential number of compatible default theories in the worst case. So in general, such an approach is favorable whenever the computational cost of the compilation can be amortized over the total set of subsequent queries.

Second, we have presented an alternative algorithm based on the refinement of our basic method. This algorithm fully integrates the concepts of complementarity, admissibility, and compatibility and accordingly supports their joint verification. Hence, it works with arbitrary default theories and thus avoids the pre-compilation of default theories into compatible fragments. At the same time, it also avoids redundant computations with incompatible defaults. On the other hand, this approach requires more changes to existing automated theorem provers. Also, successive consistency checking may slow the performance of the prover. A promising way of avoiding this seems to use model-checking techniques, as sketched in Section 9.

However, our experiments in Section 8 have shown that the latter approach is still favorable over a belated consistency check that verifies the compatibility of completed, admissible proofs. Moreover, our experiments have demonstrated that enforcing groundedness while query answering poses no additional burden on the theorem prover. Hence, we argue that the concurrent verification of admissibility and complementarity is indispensable for query answering in default logic. As discussed in Section 7, the treatment of compatibility is more subtle. Here, a lot depends on the underlying theory. That is, if a default theory comprises a feasible number of compatible default theories, then our former approach along with its precompilation of compatibility is favorable. Otherwise, that is, if a default theory comprises many conflicting defaults, an integration of compatibility as accomplished in our latter approach is preferable.

All of the presented algorithms along with their implementations are query oriented. This reflects the idea that the theorem prover is in charge of finding a proof while being directed by the concepts of admissibility and compatibility. On the other hand, our method leaves plenty of room for other algorithmic approaches, which have not yet been pursued. For instance, Section 6 provides another valuable refinement that allows for structure and information sharing while jointly verifying admissibility and compatibility.

Even though our approach has been presented from the perspective of constrained default logic, it has a general nature that principally allows for query answering in any (semi-monotonic) default logic. To this end, one merely has to adjust the concept of compatibility in order to account for the respective notion of consistency. In particular, we have shown in Section 9 that our approach carries over to cumulative default logic [5] without any substantial modifications. In the same section, we have described how lemma handling can be added to our approach.

For a complement, we have applied our approach in [39] to a prioritized version of default logic, recently introduced by Brewka in [7]. This has been accomplished by stepwisely refining the concepts developed for prioritized default logic and by mapping them in turn onto the techniques developed in the preceding sections. This extension of our method has served two purposes. First, it has shown that our method is flexible enough to be adapted to other conceptions of default logic. Second, it has shown how priorities can be integrated.

In all, our approach bridges the gap between default logics and classical theorem proving by providing a simple yet powerful method for default theorem proving that is easily adaptable by existing implementations of automated theorem provers. In particular, the approach should be easily extensible to a (decidable) first-order language because it relies on standard theorem-proving techniques.

Appendix. Proofs of Theorems

In the sequel, we will refer to some definitions and results on which we draw on in the following chapters. We give these results for the reader's convenience.

DEFINITION A.1. [36] Let (D, W) be a default theory and S and T sets of formulas. The set of generating default rules for (S, T) wrt D is defined as

$$GD_D^{(S,T)} = \left\{ \frac{\alpha : \beta}{\gamma} \in D \,\middle|\, \alpha \in S,\ T \cup \{\beta\} \cup \{\gamma\} \nvdash \bot \right\}.$$

THEOREM A.1. [36] *Let (E, C) be a constrained extension of a default theory (D, W). We have*

$$E = \mathrm{Th}\Big(W \cup \mathrm{Conseq}\Big(GD_D^{(E,C)}\Big)\Big),$$

$$C = \mathrm{Th}\Big(W \cup \mathrm{Conseq}\Big(GD_D^{(E,C)}\Big)\Big) \cup \mathrm{Justif}\Big(GD_D^{(E,C)}\Big).$$

THEOREM A.2. [36] *Let (E, C) be a constrained extension of (D, W). Then, there exists an enumeration $\langle\delta_i\rangle_{i \in I}$ of $GD_D^{(E,C)}$ such that for $i \in I$*

$$W \cup \mathrm{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \mathrm{Prereq}(\delta_i).$$

THEOREM A.3 [36] *Let (D, W) be a default theory, and let E and C be sets of formulas. Define $E_0 = W$ and $C_0 = W$ and for $i \geqslant 0$*

$$E_{i+1} = \mathrm{Th}(E_i) \cup \left\{ \gamma \,\middle|\, \frac{\alpha : \beta}{\gamma} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\gamma\} \nvdash \bot \right\},$$

$$C_{i+1} = \mathrm{Th}(C_i) \cup \left\{ \beta \wedge \gamma \,\middle|\, \frac{\alpha : \beta}{\gamma} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\gamma\} \nvdash \bot \right\}$$

(E, C) is a constrained extension of (D, W) iff $(E, C) = (\bigcup_{i=0}^{\infty} E_i, \bigcup_{i=0}^{\infty} C_i)$.

Proof 3.1. Recall that we have restricted ourselves early in the paper to consistent sets of facts. That is, for a default theory (D, W) we stipulate that W is consistent.

only-if part. Let (E, C) be a constrained extension of (D, W).
We define $D' = GD_D^{(E,C)}$ according to Definition A.1.
By Theorem A.1, we have

$$E = \mathrm{Th}(W \cup \mathrm{Conseq}(D')),$$

$$C = \mathrm{Th}(W \cup \mathrm{Justif}(D') \cup \mathrm{Conseq}(D')).$$

By Theorem A.2, we have that D' is grounded in W.

According to [36, Corollary 4.3.3], we have that W is consistent iff C is consistent. By definition, W is consistent; hence C is consistent. By Theorem A.1, we obtain that $W \cup \mathrm{Justif}(D') \cup \mathrm{Conseq}(D')$ is consistent.

Now, assume D', and so $GD_D^{(E,C)}$, is not a maximal subset of D satisfying the above requirements. Then, there is a set of default rules D'' such that $GD_D^{(E,C)} \subset D'' \subseteq D$, D'' is grounded in W and $W \cup \mathrm{Justif}(D'') \cup \mathrm{Conseq}(D'')$ is consistent. Consequently, there is a default rule $\delta'' \in (D'' \backslash GD_D^{(E,C)})$ such that $W \cup \mathrm{Conseq}(D') \vdash \mathrm{Prereq}(\delta'')$. This is so because, by assumption, D' and D'' are grounded in W. The last derivability relation implies that

$$\mathrm{Prereq}(\delta'') \in E,$$

since $E = Th(W \cup \mathrm{Conseq}(D'))$ by Theorem A.1.

By monotonocity, we have that $W \cup \mathrm{Justif}(D' \cup \{\delta''\}) \cup \mathrm{Conseq}(D' \cup \{\delta''\})$ is consistent, since $W \cup \mathrm{Justif}(D'') \cup \mathrm{Conseq}(D'')$ is consistent. Hence, we have that

$$C \cup \mathrm{Justif}(\delta'') \cup \mathrm{Conseq}(\delta'') \nvdash \bot,$$

since $C = Th(W \cup \mathrm{Justif}(D') \cup \mathrm{Conseq}(D'))$ by Theorem A.1.

By Definition A.1, (A.19) and (A.20) imply $\delta'' \in GD_D^{(E,C)}$, a contradiction.

if part. Let

$$E = \mathrm{Th}(W \cup \mathrm{Conseq}(D')),$$

$$C = \mathrm{Th}(W \cup \mathrm{Justif}(D') \cup \mathrm{Justif}(D'))$$

for a maximal $D' \subseteq D$ such that D' is grounded in W and $W \cup \mathrm{Justif}(D') \cup \mathrm{Conseq}(D')$ is consistent.

According to Theorem A.3, (E, C) is a constrained extension iff $(E, C) = (\bigcup_{i=0}^{\infty} E_i, \bigcup_{i=0}^{\infty} C_i)$ such that $E_0 = W$ and $C_0 = W$, and for $i \geqslant 0$

$$E_{i+1} = \mathrm{Th}(E_i) \cup \left\{\gamma \,\middle|\, \frac{\alpha : \beta}{\gamma} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\gamma\} \nvdash \bot\right\},$$

$$C_{i+1} = \mathrm{Th}(C_i) \cup \left\{\beta \wedge \gamma \,\middle|\, \frac{\alpha : \beta}{\gamma} \in D,\ \alpha \in E_i,\ C \cup \{\beta\} \cup \{\gamma\} \nvdash \bot\right\}.$$

We will show that $(E, C) = (\bigcup_{i=0}^{\infty} E_i, \bigcup_{i=0}^{\infty} C_i)$. Therefore, we consider the following two cases.

(1) $\bigcup_{i=0}^{\infty} E_i \subseteq E, \bigcup_{i=0}^{\infty} C_i \subseteq C$.
We show by induction that $E_i \subseteq E$ and $C_i \subseteq C$ for $i \geqslant 0$.

Base. By definition, $W \subseteq E$. Since $E_0 = W$, we have $E_0 \subseteq E$.
Analogously, we obtain $C_0 \subseteq C$.

Step. Let $E_i \subseteq E$ and $C_i \subseteq C$. Consider $\eta \in E_{i+1} \cup C_{i+1}$.

(a) If $\eta \in Th(E_i)$, then, by the induction hypothesis and the fact that E is deductively closed, we obtain $\eta \in E$.

(b) Similarly, if $\eta \in Th(C_i)$, we obtain $\eta \in C$.

(c) Otherwise, $\eta \in \{\beta, \gamma\}$ such that there is a default rule $(\alpha : \beta)/\gamma \in D$, where $\alpha \in E_i$ and $C \cup \{\beta\} \cup \{\gamma\} \nvdash \bot$.

By the induction hypothesis $\alpha \in E$. Hence, there is an $i \in I$ such that

$$W \cup \mathrm{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \alpha,$$

where $\{\delta_0, \ldots, \delta_{i-1}\} \subseteq D'$. Thus, $D' \cup \{(\alpha : \beta)/\gamma\}$ is grounded in W. Also, by definition of C, we have that $W \cup \mathrm{Justif}(D') \cup \{\beta\} \cup \mathrm{Conseq}(D') \cup \{\gamma\}$ is consistent. By maximality of D', this implies that $(\alpha : \beta)/\gamma \in D'$. Consequently, $\gamma \in E$ and $\beta \wedge \gamma \in C$ and both cases for η are covered.

From the three cases, we obtain $E_{i+1} \subseteq E$ and $C_{i+1} \subseteq C$.

(2) $E \subseteq \bigcup_{i=0}^{\infty} E_i, C \subseteq \bigcup_{i=0}^{\infty} C_i$.
Since D' is grounded in W, there is an enumeration $\langle \delta_i \rangle_{i \in I}$ of D' such that

$$W \cup \mathrm{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \mathrm{Prereq}(\delta_i)$$

for $i \in I$. With this, we define a sequence $\langle (E'_i, C'_i) \rangle_{i \in I}$ as follows:

$$(E'_0, C'_0) = (W, W)$$

$$(E'_{i+1}, c'_{i+1}) = (\mathrm{Th}(E'_i \cup \{\gamma_i\}), \mathrm{Th}(C'_i \cup \{\beta_i \wedge \gamma_i\})),$$

where $\delta_i = (\alpha_i : \beta_i)/\gamma_i$. Clearly, $E = \bigcup_{i=0}^{\infty} E'_i$ and $C = \bigcup_{i=0}^{\infty} C'_i$.

Hence, we show inductively that $E'_i \subseteq \bigcup_{i=0}^{\infty} E_i$ and $C'_i \subseteq \bigcup_{i=0}^{\infty} C_i$ for $i \geqslant 0$.

Base. Since $E'_0 = C'_0 = W$ and $E_0 = C_0 = W$, the result is obvious.

Step. According to the induction hypothesis, $E'_i \subseteq \bigcup_{i=0}^{\infty} E_i$ and $C'_i \subseteq \bigcup_{i=0}^{\infty} C_i$.

Because $W \cup \text{Conseq}(\{\delta_0, \ldots, \delta_{i-1}\}) \vdash \text{Prereq}(\delta_i)$ we have $\alpha_i \in E'_i$ and $(E'_{i+1}, C'_{i+1}) = (\text{Th}(E'_i \cup \{\gamma_i\}), \text{Th}(C'_i \cup \{\beta_i \wedge \gamma_i\}))$, where $\delta_i = (\alpha_i : \beta_i)/\gamma_i$.

By the induction hypothesis, we obtain $\alpha \in \bigcup_{i=0}^{\infty} E_i$. By compactness and monotonicity, there exists a k such that $\alpha_i \in E_k$. By definition of C and the fact that $W \cup \text{Justif}(D') \cup \text{Conseq}(D')$ is consistent, we obtain that $C \cup \{\beta_i\} \cup \{\gamma_i\} \nvdash \perp$. Then, $E_k \models \alpha_i$ and $C \cup \{\beta_i\} \cup \{\gamma_i\} \nvdash \perp$, implies $\gamma_i \in E_{k+1}$ and $\gamma_i \wedge \beta_i \in C_{k+1}$. Hence, $\gamma_i \in \bigcup_{i=0}^{\infty} E_i$ and $\gamma_i \wedge \beta_i \in \bigcup_{i=0}^{\infty} C_i$.

By the definition of E'_{i+1} and C'_{i+1} and the fact that $\bigcup_{i=0}^{\infty} E_i$ and $\bigcup_{i=0}^{\infty} C_i$ are deductively closed, we obtain $E'_{i+1} \subseteq \bigcup_{i=0}^{\infty} E_i$ and $C'_{i+1} \subseteq \bigcup_{i=0}^{\infty} C_i$.

Proof 4.1 Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let Π be a mating for $C_W \cup C_D$ such that $(C_W \cup C_D, \Pi)$ is admissible wrt I.

only-if part. Let $(C_W \cup C_D \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}, \Pi)$ be admissible for some δ-clause $\{\neg\alpha_\delta, \gamma_\delta\}$. Therefore, Π is a spanning mating for

$$C_W \cup \left(\bigcup_{i \in I} \{\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\} \right) \cup \{\{\neg\alpha_\delta\}\}.$$

Then, Π is also a spanning mating for the submatrix

$$C_W \cup \left(\bigcup_{i \in I} \{\{\neg\gamma_{\delta_i}\}\} \right) \cup \{\{\neg\alpha_\delta\}\}.$$

This is so because all paths through the latter matrix are also paths through the former matrix.

if part. Let Π be a spanning mating for $C_W \cup \bigcup_{i \in I} \{\{\gamma_{\delta_i}\}\} \cup \{\{\neg\alpha_\delta\}\}$, for some δ-clause $\{\neg\alpha_\delta, \gamma_\delta\}$. By assumption, $(C_W \cup C_D, \Pi)$ is admissible wrt I. That is, Π is a spanning mating for

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\} \right) \cup \{\{\neg\alpha_{\delta_i}\}\} \text{ for } i \in I.$$

Since Π is a spanning mating for $C_W \cup \bigcup_{i \in I} \{\{\gamma_{\delta_i}\}\} \cup \{\{\neg\alpha_\delta\}\}$, we obtain furthermore that Π is a spanning mating for

$$C_W \cup \left(\bigcup_{i \in I} \{\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\} \right) \cup \{\{\neg\alpha_\delta\}\}.$$

This implies that $(C_W \cup C_D \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}, \Pi)$ is admissible.

Proof 4.2 Let C_W be a set of ω-clauses and C_D be a set of δ-clauses. Let Π be a mating for $C_W \cup C_D$ such that $(C_W \cup C_D, \Pi)$ is admissible.

only-if part. Let Π be a spanning mating for

$$C_W \cup C_D \cup \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D,\ \beta_\delta = \text{Justif}(\delta)\}.$$

Then, Π is also a spanning mating for the submatrix

$$\begin{aligned} C_W\ &\cup\ \{\{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D\} \\ &\cup\ \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D,\ \beta_\delta = \text{Justif}(\delta)\}. \end{aligned}$$

This is so because all paths through the latter matrix are also paths through the former matrix.

if part. Let Π be a spanning mating for

$$\begin{aligned} C_W\ &\cup\ \{\{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D\} \\ &\cup\ \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D,\ \beta_\delta = \text{Justif}(\delta)\}. \end{aligned}$$

By assumption, $(C_W \cup C_D, \Pi)$ is admissible wrt some index set I. That is, Π is a spanning mating for

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\} \right) \cup \{\{\neg\alpha_{\delta_i}\}\} \text{ for } i \in I.$$

Consequently, Π is a spanning mating for

$$C_W \cup C_D \cup \{\{\neg\alpha_\delta\}\} \quad \text{for all} \quad \{\neg\alpha_\delta, \gamma_\delta\} \in C_D.$$

Consider the matrix

$$\begin{aligned} M = C_W\ &\cup\ \{\{\neg\alpha_\delta, \gamma_\delta\} \in C_D\} \\ &\cup\ \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D,\ \beta_\delta = \text{Justif}(\delta)\}. \end{aligned}$$

By (A.21), all paths through M containing a literal γ_δ are complementary. By (A.22), all paths through M containing a literal $\neg\alpha_\delta$ are complementary. Hence, all paths through M are complementary. Consequently, Π is a spanning mating for

$$M = C_W \cup C_D \cup \{\{\beta_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in C_D,\ \beta_\delta = \text{Justif}(\delta)\}.$$

Proof 4.3 Let (D, W) be a default theory in atomic format, and let $W_D = \{\alpha_\delta \rightarrow \gamma_\delta \mid (\alpha_\delta : \beta_\delta)/\gamma_\delta \in D\}$ and φ an atomic formula.

only-if part. Let (E, C) be a constrained extension of (D, W), and let $\varphi \in E$. Then, there is a set of default rules $D_\varphi \subseteq D$ such that
(1) $W \cup \text{Conseq}(D_\varphi) \vdash \varphi$,
(2) D_φ is grounded in W,
(3) $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$ is consistent.

By assumption, $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$ is consistent. By completeness of the connection method [3], this implies that the matrix of $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$ has no spanning mating. That is, there is no spanning mating for $C_W \cup C_J$ where C_W is the matrix of W and $C_J = \{\{\text{Justif}(\delta) \mid \delta \in D_\varphi\} \cup \text{Conseq}(\delta) \mid \delta \in D_\varphi\}\}$. Thus, (M, Π) is compatible for the matrix M of $W \cup W_D \cup \{\neg\varphi\}$ and any mating Π, if the core of M is given by the δ-clauses in $\{\{\neg\alpha_\delta, \gamma_\delta\} \mid \delta \in D_\varphi\}$.

We prove the rest of the theorem by induction on the cardinality of D_φ.

Base. $D_\varphi = \varnothing$. In this case, we have

$$W \vdash \varphi.$$

Then, by completeness of the connection method [3], there is a spanning mating Π for the matrix M of $W \cup \{\neg\varphi\}$. Clearly, (M, Π) is admissible.

Step. $D_\varphi \neq \varnothing$. By compactness and the fact that $W \cup \text{Conseq}(D_\varphi) \vdash \varphi$, there is a set $\{\delta_0, \dots, \delta_i\} \subseteq D_\varphi$ such that

$$W \cup \text{Conseq}(\{\delta_0, \dots, \delta_i\}) \vdash \varphi.$$

Consider δ_j. Clearly, $\text{Prereq}(\delta_j) \in E$. Then, by the induction hypothesis, there is a spanning mating Π_j for the matrix M_j of $W \cup W_{D_\varphi \setminus \{\delta_j\}} \cup \{\neg\text{Prereq}(\delta_j)\}$ such that (M_j, Π_j) is admissible.

Hence each path in M_j contains a connection from Π_j. Observe that each such path in M_j contains the literal $\neg\text{Prereq}(\delta_j)$.

Consider the matrix M of

$$W \cup \bigcup_{j=0}^{i} \left(W_{D_\varphi \setminus \{\delta_j\}} \cup \{\neg\text{Prereq}(\delta_j), \text{Conseq}(\delta_j)\}\right) \cup \{\neg\varphi\}.$$

Observe that each path in M containing a literal $\neg\text{Prereq}(\delta_j)$ contains a connection from Π_j for some $j \in \{0, \dots, i\}$. Hence all paths in M *not* containing a literal $\neg\text{Prereq}(\delta_j)$ contain the set of literals

$$\{\text{Conseq}(\delta_0), \dots, \text{Conseq}(\delta_i)\} \cup \{\neg\varphi\}.$$

By completeness of the connection method [3] and the fact that $W \cup \text{Conseq}(D_\varphi) \vdash \varphi$, there is a spanning mating Π' for the matrix M' of $W \cup \text{Conseq}(D_\varphi) \cup \{\neg\varphi\}$.

Consequently, we have that $\Pi = \Pi' \cup \bigcup_{j=0}^{i} \Pi_j$ is a spanning mating for the matrix M.

Now, it remains to be shown that (M, Π) is admissible.

By the induction hypothesis, we have that (M_j, Π_j) is admissible for all $j \in \{0, \dots, i\}$. Clearly, $\bigcup_{j=0}^{i} M_j \subset M$ and $\bigcup_{j=0}^{i} \Pi_j \subset \Pi$ implies the admissibility of (M, Π).

if part. Let Π be a spanning mating for the matrix M of $W \cup W_D \cup \{\neg\varphi\}$ such that (M, Π) is admissible and compatible. Let C_D be the set of δ-clauses in M, and let C_W be the set of ω-clauses in M.

We define

$$D_\varphi = \{\delta \mid \{\neg\alpha_\delta, \gamma_\delta\} \in \kappa(C_D, \Pi)\}.$$

Clearly, (M, Π) is complementary, admissible, and compatible if the matrix M_φ of $W \cup W_{D_\varphi} \cup \{\neg\varphi\}$ along with Π fulfills these conditions.

In what follows, we prove that

(1) $W \cup \text{Conseq}(D_\varphi) \vdash \varphi$,
(2) D_φ is grounded in W,
(3) $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$ is consistent.

By semi-monotonicity, the latter conditions imply that there is a constrained extension (E, C) of (D, W) such that $\varphi \in E$.

First, we prove Condition 3. Since (M_φ, Π) is compatible, there is no spanning mating for $C_W \cup \{\{\beta_\delta\}, \{\gamma_\delta\} \mid \{\neg\alpha_\delta, \gamma_\delta\} \in \kappa(C_D, \Pi),\ \beta_\delta = \text{Justif}(\delta)\}$. Since this is the matrix of $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$, we obtain, by completeness of the connection method [3], that $W \cup \text{Justif}(D_\varphi) \cup \text{Conseq}(D_\varphi)$ is consistent.

Since (M_φ, Π) is admissible, there is an enumeration $\langle\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i \in I}$ of $\kappa(C_D, \Pi)$ such that for $i \in I, \Pi$ is a spanning mating for

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}.$$

By compactness, I is finite. That is, $I = \{0, \ldots, n_{|I|}\}$, where $|I|$ stands for the cardinality of I.

We define $I' = \{0, \ldots, n_{|I|}, n_{|I|+1}\}$ and $\alpha_{\delta_{|I|+1}} := \varphi$. By assumption, Π is a spanning mating for M_φ, namely,

$$C_W \cup \left(\bigcup_{j=0}^{n_{|I|}} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_{|I|+1}}\}\}.$$

We prove that $W \cup \{\gamma_0, \ldots, \gamma_{i-1}\} \vdash \alpha_i$ for $i \in I'$. Clearly, this implies Conditions 1 and 2.

By assumption, Π is a spanning mating for the matrix M_i of

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}$$

for $i \in I'$. Consider the matrix M'_i,

$$C_W \cup \left(\bigcup_{j=0}^{i-1} \{\{\gamma_{\delta_j}\}\}\right) \cup \{\{\neg\alpha_{\delta_i}\}\}.$$

Assume that Π is not a spanning mating for M_i'. Then, there is a path p_W through C_W such that the path

$$p_W \cup \bigcup_{j=0}^{i-1} \{\gamma_{\delta_j}\} \cup \{\neg\alpha_{\delta_i}\}$$

is not complementary. However, this path is also a path through M_i, contradicting our initial assumption. Therefore, Π is also a spanning mating for M_i'. By correctness of the connection method [3], this implies that $W \cup \{\gamma_0, \dots, \gamma_{i-1}\} \vdash \alpha_i$.

Proof 5.2 Let (D, W) be a default theory in atomic format, and let $W_D = \{\alpha_\delta \to \gamma_\delta \mid \delta \in D\}$ and φ an atomic formula.

Without loss of generality, we assume that $W \cup \text{Justif}(D) \cup \text{Conseq}(D)$ is consistent. Then, $\text{compl}(\{\neg\varphi\}, M)$ is true[32] for the matrix M of $W \cup W_D$ iff $\varphi \in E$ for the unique constrained extension (E, C) of (D, W).

According to Theorem 4.3, this is equivalent to the following proposition: $\text{compl}(\{\neg\varphi\}, M)$ is true iff there is a spanning mating Π for $M \cup \{\neg\varphi\}$ such that $(M \cup \{\neg\varphi\}, \Pi)$ is admissible.

Observe that for any clause $\{L\}$ containing a single literal L, we have that $\text{compl}(p, M \cup \{\{L\}\})$ is true iff $\text{compl}(p \cup \{L\}, M)$ is true.

In fact, we prove below a slightly stronger statement: For a set of literals p and a matrix M, let M^p be the matrix $\bigcup_{L\in p}\{\{L\}\} \cup M$. Then, $\text{compl}(p, M)$ is true iff there is a spanning mating Π for M^p such that (M^p, Π) is admissible.

Given a set of literals p and a matrix $M = C_W \cup C_D$ consisting of ω- and δ-clauses, we define the rank of the matrix $M^p = \bigcup_{L\in p}\{\{L\}\} \cup C_W \cup C_D$ as

$$\rho(M^p) = (|C_D|, |C_W|, |p|),$$

where $|S|$ stands for the number of elements in the set S.

We prove the latter statement by induction on the lexicographic order $<$ on the rank of M^p.

only-if part. Let p be a set of literals and let $C_{W'} \subseteq C_W$ and $C_{D'} \subseteq C_D$. We prove for $M = C_{W'} \cup C_{D'}$ that if $\text{compl}(p, M)$ is true relative to C_W, then there is a spanning mating Π for $(M \cup C_W)^p$ such that $((M \cup C_W)^p, \Pi)$ is admissible. Clearly, this implies that if $\text{compl}(p, C_W \cup C_D)$ is true relative to C_W, then there is a spanning mating Π for $(C_W \cup C_D)^p$ auch $((C_W \cup C_D)^p, \Pi)$ is admissible.

Base. $M^p = \varnothing$ or $\rho(M^p) = (0, 0, 0)$. Trivial, since $\text{compl}(\varnothing, \varnothing)$ is false according to Condition 1 in Definition 5.1.[33]

Step. $M^p \neq \varnothing$ or $\rho(M^p) \neq (0, 0, 0)$. Recall that

$$M^p = (C_{W'} \cup C_{D'})^p = \bigcup_{L\in p}\{\{L\}\} \cup C_{W'} \cup C_{D'}.$$

Assume $\text{compl}(p, M)$ is true relative to C_W. Consider $c \in M^p$. We distinguish the following three cases.

(1) $c \in C_{W'}$. Consider $L \in c$. Since $c \in C_{W'}$, we have according to Condition 2 in Definition 5.1 that one of the following two cases holds.

(a) L is complementary to some literal of p.
Consequently, there is spanning mating Π_L for the matrix

$$M_L = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L\}\} \cup (M \backslash \{c\})$$

(b) $\text{compl}(p \cup \{L\}, M \backslash \{c\})$ is true.
First, observe, that

$$\rho\Big((M\backslash\{c\})^{p\cup\{L\}}\Big) < \rho(M^p).$$

Thus, we obtain by the induction hypothesis that there is spanning mating Π_L for the matrix

$$M_L = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L\}\} \cup (M \backslash \{c\})$$

such that (M_L, Π_L) is admissible.

By (a) and (b), we therefore obtain for all $L \in c$ a spanning mating Π_L for the matrix M_L. Clearly, this implies that

$$\bigcup_{L \in c} \Pi_L$$

is a spanning mating for the matrix

$$M^p = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L \mid L \in c\}\} \cup (M \backslash \{c\}).$$

Moreover, $(M^p, \bigcup_{L \in c} \Pi_L)$ is admissible because c is an ω-clause and (M_L, Π_L) is admissible for some $L \in c$. This is so because admissibility is verified wrt to the original set of ω-clauses C_W and all δ-clauses in $C_{D'}$.

The fact that $\bigcup_{L \in c} \Pi_L$ is a spanning mating for the matrix M^p such that $(M^p, \bigcup_{L \in c} \Pi_L)$ is admissible implies that the same holds for the matrix $(M \cup C_W)^p$.

(2) $c \in C_{D'}$. That is, $c = \{\neg\alpha_\delta, \gamma_\delta\}$.

According to Condition 3, in Definition 5.1, we consider the following two cases.

(a) γ_δ is complementary to some literal of p.
Consequently, there is spanning mating Π_γ for the matrix

$$M_\gamma = \bigcup_{K \in p} \{\{K\}\} \cup \{\{\gamma_\delta\}\} \cup (M \backslash \{c\}).$$

Clearly, Π_γ is also a spanning mating for the matrix $M_\gamma \cup C_W$.

(b) $\text{compl}(\{\neg\alpha_\delta\}, (M\backslash\{c\}) \cup C_W)$ is true.
First, observe, that
$$\rho\Big(((M\backslash\{c\}) \cup C_W)^{\{\neg\alpha_\delta\}}\Big) < \rho(M^p).$$
Thus, we obtain by the induction hypothesis that there is spanning mating Π_α for the matrix
$$M_\alpha = \{\{\neg\alpha_\delta\}\} \cup (M\backslash\{c\}) \cup C_W$$
such that (M_α, Π_α) is admissible.

Now, (a) and (b) imply that $\Pi_\gamma \cup \Pi_\alpha$ is a spanning mating for the matrix

$$(M \cup C_W)^p = \bigcup_{K \in p} \{\{K\}\} \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\} \cup (M\backslash\{c\}) \cup C_W.$$

It remains to be shown that $((M \cup C_W)^p, \Pi_\gamma \cup \Pi_\alpha)$ is admissible. From what we have shown, we obtain that $(\bigcup_{K\in p}\{\{K\}\} \cup C_W \cup (C_{D'}\backslash\{c\}), \Pi_\gamma \cup \Pi_\alpha)$ is admissible. Also, we have shown that $\Pi_\gamma \cup \Pi_\alpha$ is a spanning mating for $\bigcup_{K\in p}\{\{K\}\} \cup C_W \cup (C_{D'}\backslash\{c\}) \cup \{\{\neg\alpha_\delta\}\}$. This implies that $((M \cup C_W)^p, \Pi_\gamma \cup \Pi_\alpha)$ is admissible.

(3) $c \subseteq p$. This case is subsumed by the previous two cases.

if part. We prove for $M = C_{W'} \cup C_{D'}$ that if there is a spanning mating Π for M^p such that (M^p, Π) is admissible, then $\text{compl}(p, M)$ is true relative to C_W.

Base. $M^p = \varnothing$ or $\rho(M^p) = (0,0,0)$. There is only one path through M^p, namely, the empty path $\varnothing$. Clearly, this path is not complementary, since it contains no connection. Therefore, there is no spanning mating for M^p. This trivially verifies the claim.[34]

Step. $M^p \neq \varnothing$ or $\rho(M^p) \neq (0,0,0)$. Recall that

$$M^p = (C_{W'} \cup C_{D'})^p = \bigcup_{L \in p} \{\{L\}\} \cup C_{W'} \cup C_{D'}.$$

Let Π be a spanning mating for M^p such that (M^p, Π) is admissible. Consider $c \in M^p$. We distinguish the following three cases.

(1) $c \in C_{W'}$. Consider $L \in c$. Clearly, Π is also a spanning mating for the matrix

$$M_L = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L\}\} \cup (M\backslash\{c\})$$

such that (M_L, Π) is admissible. That is, each path through M_L contains a connection from Π. Then, one of the following two cases holds.
(a) L is complementary to some literal of p.
(b) By rewriting M_L, we have that Π is a spanning mating for matrix
$$(M\backslash\{c\})^{p\cup\{L\}}$$

such that $((M\backslash\{c\})^{p\cup\{L\}}, \Pi)$ is admissible. Observe that

$\rho((M\backslash\{c\})^{p\cup\{L\}}) < \rho(M^p)$.

Thus, we obtain by the induction hypothesis that compl$(p \cup \{L\}, M\backslash\{c\})$ is true.

Since both cases of Condition 2 in Definition 5.1 are covered, we have that compl(p, M) is true.

(2) $c \in C_{D'}$. Without loss of generality, we assume that c is necessary for the complementarity of M^p. That is, Π is no spanning mating for $M^p\backslash\{c\}$.

To show that compl(p, M) is true, we reduce this problem by applying Condition 2 in Definition 5.1 to all clauses $c \in C_{W'}$. As a consequence, compl(p, M) is true iff compl$(p \cup p_{W'}, C_{D'})$ is true for all noncomplementary paths $p \cup p_{W'}$ where $p_{W'}$ is a path through $C_{W'}$. That is, we have for all such paths $p \cup p_{W'}$ and all $\pi \in \Pi$ that $\pi \cap (p \cup p_{W'}) = \varnothing$.

Thus, in what follows, we show that compl$(p \cup p_{W'}, C_{D'})$ is true relative to C_W for all noncomplementary paths $p \cup p_{W'}$.

Consider $c = \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}$. We distinguish the following two cases.

(a) By assumption, Π is a spanning mating for the matrix

$$\{\{\gamma_{\delta_i}\}\} \cup (M\backslash\{c\})^p = \bigcup_{K\in p} \{\{K\}\} \cup C_{W'} \cup \{\{\gamma_{\delta_i}\}\} \cup (C_{D'}\backslash\{c\}).$$

Then, Π is also a spanning mating for all matrices

$$\bigcup_{K\in p\cup p_{W'}} \{\{K\}\} \cup \{\{\gamma_{\delta_i}\}\} \cup (C_{D'}\backslash\{c\}),$$

for all noncomplementary paths $p \cup p_{W'}$ (where $p_{W'}$ is a path through $C_{W'}$). Now, all paths through such a matrix are of the form

$$p \cup p_{W'} \cup \{\mu \mid \mu \in \{\neg\alpha, \gamma\} \in C_{D'}\backslash\{c\}\} \cup \{\gamma_\delta\}.$$

By assumption, some of these paths are not complementary without γ_{δ_i}. Since γ_{δ_i} cannot be complementary to any literal in $\{\mu \mid \mu \in \{\neg\alpha, \gamma\} \in C_{D'}\backslash\{c\}\}$, we have that γ_{δ_i} must be complementary to some negated literal in $p \cup p_{W'}$.

(b) By assumption, Π is a spanning mating for the matrix $\{\{\neg\alpha_{\delta_i}\}\} \cup (M\backslash\{c\})^p$. Also, $(\{\{\neg\alpha_{\delta_i}\}\} \cup (M\backslash\{c\})^p, \Pi)$ is admissible wrt $\{j \mid j < i\}$. By admissibility, Π is thus a spanning mating for the matrix

$$(C_W \cup \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\} \mid j < i\})^{\{\neg\alpha_{\delta_i}\}}.$$

In addition,

$$((C_W \cup \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\} \mid j < i\})^{\{\neg\alpha_{\delta_i}\}}, \Pi)$$

is admissible wrt $\{j \mid j < i\}$. Accordingly,

$$\text{compl}(\{\neg\alpha_{\delta_i}\}, \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\} \mid j < i\} \cup C_W)$$

is true by the induction hypothesis. This implies that compl$(\{\neg\alpha_{\delta_i}\}, M\backslash\{c\} \cup C_W)$ is true, since adding redundant δ-clauses does not change the satisfiability of the former condition.

We have shown in (a) that γ_{δ_i} is complementary to some literal in $p \cup p_{W'}$ (where $p_{W'}$ is a path through $C_{W'}$). In (b), we have shown that compl($\{\neg\alpha_{\delta_i}\}$, $M \backslash \{c\} \cup C_W$) is true. Therefore, compl($p \cup p_{W'}, C_{D'}$) is true for all noncomplementary paths $p \cup p_{W'}$. As shown above, this implies that compl(p, M) is true.

(3) $c \subseteq p$. This case is subsumed by the first cases.

Proof 6.2 Let Π be a mating for $C_W \cup C_D$, where C_W and C_D are sets of ω- and δ-clauses. Let $\langle\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i \in I}$ be an enumeration of $\kappa(C_D, \Pi)$.

only-if part. Assume that $(C_W \cup C_D, \Pi)$ is compatible. Then, there is no spanning mating for

$$C_W \cup \bigcup_{i \in I} \{\{\beta_{\delta_i}\}, \{\gamma_{\delta_i}\}\}.$$

Hence there is an open path

$$p = p_W \cup \bigcup_{i \in I} \{\{\beta_{\delta_i}\}, \{\gamma_{\delta_i}\}\}$$

through the latter matrix for some path p_W through C_W. This implies that for all $i \in I$,

$$p_i = p_W \cup \bigcup_{j \leqslant i} \{\{\beta_{\delta_i}\}, \{\gamma_{\delta_i}\}\}$$

is not complementary. Accordingly, there is no spanning mating for

$$C_W \cup \bigcup_{j \leqslant i} \{\{\beta_{\delta_i}\}, \{\gamma_{\delta_i}\}\}$$

for $i \in I$. This demonstrates that $(C_W \cup C_D, \Pi)$ is incrementally compatible wrt I.

if part. Trivial. This is so because checking incremental compatibility for that last δ-clause in the sequence $\langle\{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}\rangle_{i \in I}$ is equivalent to checking compatibility.

Proof 6.4 Let (D, W) be a default theory in atomic format, and let $W_D = \{\alpha_\delta \rightarrow \gamma_\delta \mid \delta \in D\}$ and φ an atomic formula.

We show that $\varphi \in E$ for some constrained extension (E, C) of (D, W) iff compl($\{\neg\varphi\}, C_W, C_D$) is true, where C_W is the matrix of W and C_D is the matrix of W_D.

According to Corollary 6.3, this is equivalent to the following proposition: compl($\{\neg\varphi\}, C_W, C_D$) is true iff there is a spanning mating Π for the matrix $M = C_W \cup C_D \cup \{\{\neg\varphi\}\}$ and an enumeration $\langle c_i\rangle_{i \in I}$ of $\kappa(C_D, \Pi)$ such that (M, Π) is admissible wrt I and incrementally compatible wrt I.

As in Proof 5.2, we prove below a slightly stronger statement: For a set of literals p and a matrix M, let M^p be the matrix $\bigcup_{L\in p}\{\{L\}\} \cup M$. Then, the theorem reduces to this: $\mathrm{compl}(p, C_W, C_D)$ is true iff there is a spanning mating Π for $(C_W \cup C_D)^p$ and an enumeration $\langle c_i\rangle_{i\in I}$ of $\kappa(C_D, \Pi)$ such that $((C_W \cup C_D)^p, \Pi)$ is admissible and incrementally compatible wrt I.

As in Proof 5.2, we define the rank of matrix $C_W \cup C_D$ along with a set of literals p, $(C_W \cup C_D)^p = \bigcup_{L\in p}\{\{L\}\} \cup C_W \cup C_D$ as

$$\rho((C_W \cup C_D)^p) = (|C_D|, |C_W|, |p|),$$

where $|S|$ stands for the number of elements in the set S.

In analogy to Proof 5.2, we prove the latter statement by induction on the lexicographic order $<$ on the rank of $(C_W \cup C_D)^p$.

only-if part. Let p be a set of literals and let $C_{W'} \subseteq C_W$ and $C_{D'} \subseteq C_D$. We prove that if $\mathrm{compl}(p, C_{W'}, C_{D'})$ is true relative to C_W, then there is a spanning mating Π for $((C_{W'} \cup C_{D'}) \cup C_W)^p$ and an enumeration $\langle c_i\rangle_{i\in I'}$ of $\kappa(C_{D'}, \Pi)$ such that $(((C_{W'} \cup C_{D'}) \cup C_W)^p, \Pi)$ is admissible and incrementally compatible wrt I'. Clearly, this implies that if $\mathrm{compl}(p, C_W, C_D)$ is true relative to C_W, then there is a spanning mating Π for $(C_W \cup C_D)^p$ and an enumeration $\langle c_i\rangle_{i\in I}$ of $\kappa(C_D, \Pi)$ such that $((C_W \cup C_D)^p, \Pi)$ is admissible and incrementally compatible wrt I.

Base. Analogous to Proof 5.2.

Step. $(C_{W'} \cup C_{D'})^p \neq \varnothing$ or $\rho((C_{W'} \cup C_{D'})^p) \neq (0,0,0)$. Recall that

$$(C_{W'} \cup C_{D'})^p = \bigcup_{L\in p}\{\{L\}\} \cup C_{W'} \cup C_{D'}.$$

Assume $\mathrm{compl}(p, C_{W'}, C_{D'})$ is true relative to C_W. Consider $C \in (C_{W'} \cup C_{D'})^p$. We distinguish the following three cases.

(1) $c \subseteq p$. Analogous to Proof 5.2.

(2) $c \in C_{W'}$. Consider $L \in c$. Since $c \in C_{W'}$, we have according to Condition 2 in Definition 5.1 that one of the following two cases holds, where $c = c_1 \cup c_2$.

(a) $L \in c_1$ and L is complementary to some literal of p.
Consequently, there is spanning mating Π_L for the matrix

$$M_L = \bigcup_{K\in p}\{\{K\}\} \cup \{\{L\}\}.$$

(b) $L \in c_2$ and there is a set of δ-clauses $C_{D'_L} \subseteq C_{D'}$ such that $\mathrm{compl}(p \cup \{L\}, C_{W'}\setminus\{c\}, C_{D'_L})$ is true.
First, observe, that

$$\rho\Big(((C_{W'}\setminus\{c\}) \cup C_{D'_L})^{p\cup\{L\}}\Big) < \rho((C_{W'} \cup C_{D'})^p).$$

Thus, we obtain by the induction hypothesis that there is spanning mating Π_L for the matrix

$$M_L = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L\}\} \cup ((C_{W'} \setminus \{c\}) \cup C_{D'_L})$$

and an enumeration $\langle c_i \rangle_{i \in I_L}$ of $\kappa(C_{D'_L}, \Pi_L)$ such that (M_L, Π_L) is admissible and incrementally compatible wrt I_L.

By (a) and (b), we therefore obtain for all $L \in c$ a spanning mating Π_L for the matrix M_L. Clearly, this implies that

$$\bigcup_{L \in c} \Pi_L$$

is a spanning mating for the matrix

$$(C_{W'} \cup C_{D'})^p = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L \mid L \in c\}\} \cup ((C_{W'} \setminus \{c\}) \cup C_{D'}).$$

Consider the enumeration $\langle c_i \rangle_{i \in I}$ obtained by meshing together the enumerations $\langle c_i \rangle_{i \in I_L}$ of $\kappa(C_{D'_L}, \Pi_L)$ for each $L \in c_2$ such that $\langle c_i \rangle_{i \in I}$ respects the order of δ-clauses in each $\langle c_i \rangle_{i \in I_L}$. Clearly, $\langle c_i \rangle_{i \in I}$ is an enumeration of $\kappa(C_{D'}, \bigcup_{L \in c} \Pi_L)$. Also it is easy to see that $((C_{W'} \cup C_{D'})^p, \bigcup_{L \in c} \Pi_L)$ is admissible wrt I, since (M_L, Π_L) is admissible for all $L \in c_2$.

Moreover, we have that compl(Justif($\bigcup_{L \in c_2} D'_L$), $C_W \cup \bigcup_{L \in c_2} C_{D'_L}, \varnothing$) is false. This implies that compl($\varnothing, M_J, \varnothing$) is false for the matrix

$$M_J = \bigcup_{K \in \text{Justif}\left(\bigcup_{L \in c_2} D'_L\right)} \{\{K\}\} \cup C_W \cup \bigcup_{L \in c_2} C_{D'_L}.$$

By [14] and the completeness of the connection method, this implies that there is no spanning mating for M_J. Since $((C_{W'} \cup \bigcup_{L \in c_2} C_{D'_L})^p, \bigcup_{L \in c_2} \Pi_L)$ is admissible wrt I, we obtain by Theorem 4.2 that there is no spanning mating for

$$M'_J = C_W \cup \bigcup_{K \in \text{Justif}\left(\bigcup_{L \in c_2} D'_L\right)} \{\{K\}\} \cup \bigcup_{K \in \text{Conseq}\left(\bigcup_{L \in c_2} D'_L\right)} \{\{K\}\}.$$

Accordingly, we have that $((C_{W'} \cup \bigcup_{L \in c_2} C_{D'_L})^p, \bigcup_{L \in c_2} \Pi_L)$ is compatible. Applying Theorem 6.2 to this along with $\langle c_i \rangle_{i \in I}$ yields that $((C_{W'} \cup C_{D'})^p, \bigcup_{L \in c_2} \Pi_L)$ is incrementally compatible wrt I.

The fact that $\bigcup_{L \in c} \Pi_L$ is a spanning mating for the matrix $(C_{W'} \cup C_{D'})^p$ and that $\langle c_i \rangle_{i \in I}$ is an enumeration of $\kappa(C_{D'}, \bigcup_{L \in c} \Pi_L)$ such that $((C_{W'} \cup C_{D'})^p, \bigcup_{L \in c} \Pi_L)$ is admissible and incrementally compatible wrt I implies that the same holds for the matrix $((C_{W'} \cup C_{D'}) \cup C_W)^p$.

(3) $c \in C_{D'}$. That is, $c = \{\neg\alpha_\delta, \gamma_\delta\}$.

According to Condition 3, in Definition 5.1, we consider the following two cases.

(a) γ_δ is complementary to some literal of p.
Consequently, there is spanning mating Π_γ for the matrix

$$M_\gamma = \bigcup_{K\in p} \{\{K\}\} \cup \{\{\gamma_\delta\}\}.$$

Clearly, Π_γ is also a spanning mating for the matrix

$$M_\gamma = \bigcup_{K\in p} \{\{K\}\} \cup \{\{\gamma_\delta\}\} \cup C_{W'} \cup (C_{D'}\setminus\{c\}) \cup C_W.$$

(b) There is a set of δ-clauses $C_{D''} \subsetneq C_{D'}$ such that $\{\neg\alpha_\delta,\gamma_\delta\} \in C_{D'}\setminus C_{D''}$ and the following two conditions hold.

(i) $\text{compl}(\{\neg\alpha_\delta\}, C_W, C_{D''})$ is true.
Observe that

$$\rho\Big((C_W \cup C_{D''})^{\{\neg\alpha_\delta\}}\Big) < \rho((C_{W'} \cup C_{D'})^p).$$

Thus, we obtain by the induction hypothesis that there is a spanning mating Π_α for the matrix

$$M_\alpha = \{\{\neg\alpha_\delta\}\} \cup C_W \cup C_{D''}$$

and an enumeration $\langle c_i\rangle_{i\in I''}$ of $\kappa(C_{D''},\Pi_\alpha)$ such that (M_α,Π_α) is admissible and incrementally compatible wrt I''.

(ii) $\text{compl}(\text{Justif}(D'' \cup \{\delta\}), C_W \cup C_{D''} \cup \{\neg\alpha_\delta,\gamma_\delta\}, \varnothing)$ is false.
This implies that $\text{compl}(\varnothing, M_J, \varnothing)$ is false for the matrix

$$M_J = \bigcup_{K\in \text{Justif}(D''\cup\{\delta\})} \{\{K\}\} \cup C_W \cup C_{D''} \cup \{\neg\alpha_\delta,\gamma_\delta\}.$$

By [14] and the completeness of the connection method, this implies that there is no spanning mating for M_J.
Since Π_α is a spanning mating for

$$M_\alpha = \{\{\neg\alpha_\delta\}\} \cup C_W \cup C_{D''}$$

and (M_α,Π_α) is admissible wrt I'', we have that all paths through $\neg\alpha_\delta$ or $\neg\alpha_{\delta_i}$ for $i \in I''$ in M_J are complementary.
Therefore, there is no spanning mating for

$$M'_J = C_W \cup \bigcup_{K\in \text{Justif}(D''\cup\{\delta\})} \{\{K\}\} \cup \bigcup_{K\in \text{Conseq}(D''\cup\{\delta\})} \{\{K\}\}.$$

Now, (a) and (b) imply that $\Pi_\gamma \cup \Pi_\alpha$ is a spanning mating for the matrix

$$\begin{aligned}&((C_{W'} \cup C_{D'}) \cup C_W)^p\\ &= \bigcup_{K\in p} \{\{K\}\} \cup \{\{\neg\alpha_\delta,\gamma_\delta\}\} \cup ((C_{W'} \cup C_{D'})\setminus\{c\}) \cup C_W.\end{aligned}$$

Consider the enumeration $\langle c_i\rangle_{i\in I'}$ obtained from appending the δ-clause $\{\neg\alpha_\delta,\gamma_\delta\}$ to the enumeration $\langle c_i\rangle_{i\in I''}$. Clearly, $\langle c_i\rangle_{i\in I'}$ is an enumeration of $\kappa(C_{D'},\Pi_\gamma,\Pi_\alpha)$ since Π_γ and Π_α refer exclusively to δ-clauses in $C_{D''} \cup \{\{\neg\alpha_\delta,\gamma_\delta\}\}$. Now, it remains to be shown that $(((C_{W'} \cup C_{D'}) \cup C_W)^p, \Pi_\gamma \cup \Pi_\alpha)$ is admissible and incrementally compatible wrt I'.

For admissibility, observe that $(\bigcup_{K\in p}\{\{K\}\} \cup C_W \cup C_{D''}, \Pi_\gamma \cup \Pi_\alpha)$ is admissible wrt I''. Also, we have shown that $\Pi_\gamma \cup \Pi_\alpha$ is a spanning mating for

$\bigcup_{K \in p}\{\{K\}\} \cup C_W \cup C_{D''} \cup \{\{\neg\alpha_\delta\}\}$. This implies that $(((C_{W'} \cup C_{D'}) \cup C_W)^p, \Pi_\gamma \cup \Pi_\alpha)$ is admissible wrt I'.

For compatibility, observe that there is no spanning mating for the matrix

$$C_W \cup \bigcup_{K \in \mathrm{Justif}(D'' \cup \{\delta\})} \{\{K\}\} \cup \bigcup_{K \in \mathrm{Conseq}(D'' \cup \{\delta\})} \{\{K\}\}.$$

Since $\kappa(C_{D'}, \Pi_\gamma \cup \Pi_\alpha) \subseteq C_{D''} \cup \{\{\neg\alpha_\delta, \gamma_\delta\}\}$, we obtain that $(C_W \cup C_{D'}, \Pi_\gamma \cup \Pi_\alpha)$ is compatible. Consequently, we have that $((C_{W'} \cup C_{D'}) \cup C_W, \Pi_\gamma \cup \Pi_\alpha)$ is compatible. Applying Theorem 6.2 to this along with $\langle c_i \rangle_{i \in I'}$ yields that $((C_{W'} \cup C_{D'}) \cup C_W, \Pi_\gamma \cup \Pi_\alpha)$ is incrementally compatible wrt I'.

if part. We prove that if there is a spanning mating Π for $(C_{W'} \cup C_{D'})^p$ such that $((C_{W'} \cup C_{D'})^p, \Pi)$ is admissible and incrementally compatible wrt to some index set I, then $\mathrm{compl}(p, C_{W'}, C_{D'})$ is true relative to C_W.

Base. Analogous to Proof 5.2.

Step. $(C_{W'} \cup C_{D'})^p \neq \varnothing$ or $\rho((C_{W'} \cup C_{D'})^p) \neq (0,0,0)$. Recall that

$$(C_{W'} \cup C_{D'})^p = \bigcup_{L \in p} \{\{L\}\} \cup C_{W'} \cup C_{D'}.$$

Let Π be a spanning mating for $(C_{W'} \cup C_{D'})^p$ such that $((C_{W'} \cup C_{D'})^p, \Pi)$ is admissible wrt I. Consider $c \in (C_{W'} \cup C_{D'})^p$. We distinguish the following three cases.

(1) $c \subseteq p$. Analogous to Proof 5.2.

(2) $c \in C_{W'}$. Consider $L \in c$. Clearly, Π is also a spanning mating for the matrix

$$M_L = \bigcup_{K \in p} \{\{K\}\} \cup \{\{L\}\} \cup ((C_{W'} \backslash \{c\}) \cup C_{D'})$$

such that (M_L, Π) is admissible and incrementally compatible wrt I. That is, each path through M_L contains a connection from Π. Then, one of the following two cases holds.

(a) L is complementary to some literal of p.

(b) By rewriting M_L, we have that Π is a spanning mating for the matrix
$$((C_{W'} \backslash \{c\}) \cup C_{D'})^{p \cup \{L\}}$$
such that $(((C_{W'} \backslash \{c\}) \cup C_{D'})^{p \cup \{L\}}, \Pi)$ is admissible and incrementally compatible wrt I. Observe that
$$\rho\Big(((C_{W'} \backslash \{c\}) \cup C_{D'})^{p \cup \{L\}}\Big) < \rho((C_{W'} \cup C_{D'})^p).$$
Thus, we obtain by the induction hypothesis that $\mathrm{compl}(p \cup \{L\}, C_{W'} \backslash \{c\}, C_{D'})$ is true.

Since $((C_{W'} \cup C_{D'})^p, \Pi)$ is incrementally compatible wrt I, there is no spanning mating for

$$C_W \cup \bigcup_{i \in I} \{\beta_{\delta_i}\} \cup \bigcup_{i \in I} \{\gamma_{\delta_i}\}.$$

Since $((C_{W'} \cup C_{D'})^p, \Pi)$ is admissible wrt I, we obtain by Theorem 4.2, that there is no spanning mating for the matrix

$$C_W \cup \bigcup_{i \in I} \{\beta_{\delta_i}\} \cup \bigcup_{i \in I} \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}.$$

Let $I' = \{i \in I \mid \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\} \in C_{D'}\}$ be set of indexes of all δ-clauses in $C_{D'}$. Clearly, there is also no spanning mating for the matrix

$$C_W \cup \bigcup_{i \in I'} \{\beta_{\delta_i}\} \cup \bigcup_{i \in I} \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}.$$

By [14] and the correctness of the connection method, this implies that

$$\text{compl}\Big(\varnothing, C_W \cup \bigcup_{i \in I'} \{\beta_{\delta_i}\} \cup \bigcup_{i \in I'} \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}, \varnothing\Big) \text{ is false.}$$

This implies that

$$\text{compl}\Big(\bigcup_{i \in I'} \{\beta_{\delta_i}\}, C_W \cup \bigcup_{i \in I'} \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}, \varnothing\Big) \text{ is false.}$$

That is, compl(Justif(D'), $C_W \cup C_{D'}, \varnothing)$ is false.

Since both cases of Condition 2, in Definition 5.1 are covered, we have that compl$(p, C_{W'}, C_{D'})$ is true.

(3) $c \in C_{D'}$. Without loss of generality, we assume that c is necessary for the complementarity of $(C_{W'} \cup C_{D'})^p$. That is, Π is no spanning mating for $(C_{W'} \cup C_{D'})^p \backslash \{c\}$.

To show that compl$(p, C_{W'}, C_{D'})$ is true, we reduce this problem by applying Condition 2, in Definition 6.2 to all clauses $c \in C_{W'}$. As a consequence, compl$(p, C_{W'}, C_{D'})$ is true iff compl$(p \cup p_{W'}, \varnothing, C_{D'})$ is true for all noncomplementary paths $p \cup p_{W'}$ where $p_{W'}$ is a path through $C_{W'}$. That is, we have all such paths $p \cup p_{W'}$ and all $\pi \in \Pi$ that $\pi \cap (p \cup p_{W'}) = \varnothing$.

Thus, in what follows, we show that compl$(p \cup p_{W'}, \varnothing, C_{D'})$ is true relative to C_W for all noncomplementary path $p \cup p_{W'}$.

Consider $c = \{\neg\alpha_{\delta_i}, \gamma_{\delta_i}\}$. We distinguish the following two cases.

(a) By assumption, Π is a spanning mating for the matrix

$$\begin{aligned}&\{\{\gamma_{\delta_i}\}\} \cup (C_{W'} \cup C_{D'} \backslash \{c\})^p \\ &\quad = \bigcup_{K \in p} \{\{K\}\} \cup C_{W'} \cup \{\{\gamma_{\delta_i}\}\} \cup (C_{D'} \backslash \{c\}).\end{aligned}$$

Then, Π is also a spanning mating for all matrices

$$\bigcup_{K \in p \cup p_{W'}} \{\{K\}\} \cup \{\{\gamma_{\delta_i}\}\} \cup (C_{D'} \backslash \{c\}),$$

for all noncomplementary paths $p \cup p_{W'}$ (where $p_{W'}$ is a path through $C_{W'}$). Now, all paths through such a matrix are of the form

$$p \cup p_{W'} \cup \{\mu \mid \mu \in \{\neg\alpha, \gamma\} \in C_{D'} \backslash \{c\}\} \cup \{\gamma_{\delta_i}\}.$$

By assumption, some of these paths are not complementary without γ_{δ_i}. Since γ_{δ_i} cannot be complementary to any literal in $\{\mu \mid \mu \in \{\neg\alpha, \gamma\} \in C_{D'} \backslash \{c\}\}$, we have that γ_{δ_i} must be complementary to some negated literal in $p \cup p_{W'}$.

(b) By assumption, Π is a spanning mating for the matrix $\{\{\neg\alpha_{\delta_i}\}\} \cup (C_{W'} \cup C_{D'} \backslash \{c\})^p$. Also, $(\{\{\neg\alpha_{\delta_i}\}\} \cup (C_{W'} \cup C_{D'} \backslash \{c\})^p, \Pi)$ is admissible and incrementally compatible wrt $\{j \mid j < i\}$. Let $C_{D''} = \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\} \mid j < i\}$. By admissibility, Π is thus a spanning mating for the matrix

$$(C_W \cup C_{D''})^{\{\neg\alpha_{\delta_i}\}}.$$

In addition,

$$((C_W \cup C_{D''})^{\{\neg\alpha_{\delta_i}\}}, \Pi)$$

is admissible and incrementally compatible wrt $\{j \mid j < i\}$. By the induction hypothesis, this implies that compl$(\{\neg\alpha_{\delta_i}\}, C_W, C_{D''})$ is true.

Since $((C_{W'} \cup C_{D'})^p, \Pi)$ is incrementally compatible wrt I and $\{j \mid j \leqslant i\} \subseteq I$, there is no spanning mating for the matrix

$$C_W \cup \bigcup_{j \leqslant i} \{\beta_{\delta_j}\} \cup \bigcup_{j \leqslant i} \{\gamma_{\delta_j}\}.$$

Since $((C_{W'} \cup C_{D'} \backslash \{c\})^p, \Pi)$ is admissible wrt I and $\{j \mid j \leqslant i\} \subseteq I$, we obtain by Theorem 4.2, that there is no spanning mating for the matrix

$$C_W \cup \bigcup_{j \leqslant i} \{\beta_{\delta_j}\} \cup \bigcup_{j \leqslant i} \{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}.$$

By [14] and the correctness of the connection method, this implies that

$$\text{compl}\left(\varnothing, C_W \cup \bigcup_{j \leqslant i} \{\beta_{\delta_j}\} \cup \bigcup_{j \leqslant i} \{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}, \varnothing\right) \text{ is false.}$$

This implies that

$$\text{compl}\left(\bigcup_{j \leqslant i} \{\beta_{\delta_j}\}, C_W \cup \bigcup_{j \leqslant i} \{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\}, \varnothing\right) \text{ is false.}$$

That is, compl(Justif$(D'' \cup \{\delta\}), C_W \cup C_{D''} \cup \{\neg\alpha_\delta, \gamma_\delta\}, \varnothing)$ is false.

We have shown in (a) that γ_{δ_i} is complementary to some literal in $p \cup p_{W'}$ (where $p_{W'}$ is a path through $C_{W'}$). For $C_{D''} = \{\{\neg\alpha_{\delta_j}, \gamma_{\delta_j}\} \mid j < i\}$, we have shown in (b) that compl$(\{\neg\alpha_{\delta_i}\}, C_W, C_{D''})$ is true and compl(Justif$(D'' \cup \{\delta\}), C_W \cup C_{D''} \cup \{\neg\alpha_\delta, \gamma_\delta\}, \varnothing)$. Therefore, compl$(p \cup p_{W'}, \varnothing, C_{D'})$ is true for all noncomplementary paths $p \cup p_{W'}$. As shown above, this implies that compl$(p, C_{W'}, C_{D'})$ is true.

Acknowledgements

I am indebted to Aaron Rothschild, who helped develop some of my initial ideas for query answering in default logics. Also, he programmed and tested the extension of PPP. I thank Gerhard Brewka, James Delgrande, Michael Thielscher, and in particular Stefan Brüning for many useful discussions and comments on earlier drafts of this paper. Moreover, the comments of the anonymous referees helped to improve the quality of the paper. This work was supported by the Commission of the European Communities under grant no. ERB4001GT922433.

Notes

[1] [11] essentially marries and extends the work found in [10] and [35]. Moreover, constrained default logic, as introduced in [11], subsumes the variants introduced in [10] and [35].

[2] The restriction to consistent set of facts is not really necessary, but it simplifies matters.

[3] We discuss semi-monotonicity in detail in Section 3.

[4] For a general and thorough account on the differences between Reiter's and constrained default logic, we refer the reader to [11].

[5] Observe that both conditions coincide for normal default rules, which (roughly) explains why both approaches coincide in the case of normal default theories.

[6] The notion of groundedness was first explicated for default logic in [41], although it was already present in [30]. Groundedness was also studied in [20] in the case of autoepistemic logic.

[7] These projections extend to sets of default rules in the obvious way.

[8] See Theorem A.1 for a formal formulation.

[9] For simplicity, we introduce only two new propositional letters, since δ_1 is a normal default rule.

[10] In fact, this has been shown in [33] for regular as well as constrained default logic.

[11] In the sequel, we simply say literal instead of literal occurrences; the latter allow for distinguishing between identical literals in different clauses.

[12] As pointed out by one the referees, an alternative approach is to view the connection method as a meta-theoretic approach. To this end, one could incorporate default rules by means of so-called theory connections, as used for instance in [3] for incorporating equality or induction. We have not pursued this line of inquiry since it makes the use of existing automated theorem-proving technology more difficult.

[13] Without loss of generality, we deal with atomic queries only, since any query can be transformed into 'atomic format' in the spirit of transformation τ (cf. Section 2).

[14] Recall that we deal with literal occurrences.

[15] For illustration, we often explicate the respective δ-clauses rather than expressing things in terms of indexes.

[16] Observe that we omit in (4.9) the clauses $\{A_{\delta_1}\}$ and $\{C_{\delta_2}\}$ corresponding to C_J, since δ_1 and δ_2 are normal default rules.

[17] Observe that we omit the clauses $\{A_{\delta_2}\}$ and $\{E_{\delta_3}\}$ due to the fact that δ_2 and δ_3 are normal default rules.

[18] See Definition A.1 for a formal definition.

[19] This is so because there may be an exponential number of extensions in the worst case.

[20] For simplicity, we assume in what follows that compl(p, M) is always relative to the original set of ω-clauses C_W.

[21] That is, for any non-empty path p and any (unit-)clause containing a single literal L we can restrict ourselves either to testing (2a) or (2b). This however renders the choice in (2) a '*don't know*'-choice (see below).

[22] Recall that we want to stick as close as possible to existing technologies, so that we refrain from making additional changes.

[23] Recall that $W_{D'} = \{\alpha_\delta \rightarrow \gamma_\delta \mid (\alpha_\delta : \beta_\delta)/\gamma_\delta \in D'\}$ for any subset D' of D.

[24] As above, we assume in what follows that $\mathrm{compl}(p, C_{W'}, C_{D'})$ is relative to C_W.

[25] Recall that we do not have to add the justification to the path in the case of normal default rules.

[26] Observe that this does not render Reiter's procedure incomplete, since neither A nor C should be derivable in this case.

[27] This is the compatible version of the 'knowledge base' `students`.

[28] Thanks to Richard O'Keefe, the author of the count program.

[29] Let $\mathrm{Form}(\xi)$ be the asserted formula and $\mathrm{Supp}(\xi)$ the support of an assertion ξ: if $\xi_1, \ldots, \xi_n \in \widehat{\mathrm{Th}}(S)$ and $\mathrm{Form}(\xi_1), \ldots, \mathrm{Form}(\xi_n) \vdash \alpha$ then $\langle \alpha, \bigcup_{i=1}^{n} \mathrm{Supp}(\xi_i) \rangle \in \widehat{\mathrm{Th}}(S)$.

[30] In what follows, we focus on the approach given for constrained default logic.

[31] See Definition A.1 for a formal definition.

[32] Recall that $\mathrm{compl}(p, M)$ is always relative to the original set of ω-clauses C_W.

[33] Reasoning by contraposition would be an alternative to prove the same result.

[34] Reasoning by contraposition would be an alternative to prove the same result.

References

1. Baader, F. and Hollunder, B.: Embedding defaults into terminological knowledge representation formalisms, in B. Nebel, C. Rich, and W. Swartout (eds), *Proc. 3rd Int. Conf. Principles of Knowledge Representation and Reasoning,* Cambridge, MA, pp. 306–317 October 1992.
2. Besnard, P., Quiniou, R. and Quinton, P.: A theorem-prover for a decidable subset of default logic, in *Proc. AAAI Nat. Conf. Artificial Intelligence,* 1983, pp. 27–30.
3. Bibel, W.: *Automated Theorem Proving,* 2nd edn, Vieweg, Braunschweig, 1987.
4. Brass, S.: Deduction with supernormal defaults, in P. Schmitt, G. Brewka and K. Jantke (eds), *Nonmonotonic and Inductive Logic,* Springer, Berlin, 1991, pp. 153–174.
5. Brewka, G.: Cumulative default logic: In defense of nonmonotonic inference rules, *Artificial Intelligence* **50**(2) (1991), 183–205.
6. Brewka, G.: *Nonmonotonic Reasoning: Logical Foundations of Commonsense,* Cambridge University Press, Cambridge, 1991.
7. Brewka, G.: Adding priorities and specificity to default logic, in L. Pereira and D. Pearce (eds), *European Workshop on Logics in Artificial Intelligence (JELIA'94),* Springer, Berlin, 1994.
8. Cadoli, M., Eiter, T. and Gottlob, G.: Default logic as a query language, in J. Doyle, P. Torasso and E. Sandewall (eds), *Proc. 4th Int. Conf. Principles of Knowledge Representation and Reasoning,* 1994.
9. Cadoli, M. and Schaerf, M.: A survey on complexity results for non-monotonic logics, *J. Logic Programming* **17** (1993).
10. Delgrande, J. and Jackson, W.: Default logic revisited, in J. Allen, R. Fikes and E. Sandewall (eds), *Proc. 2nd Int. Conf. Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991, pp. 118–127.
11. Delgrande, J., Schaub, T. and Jackson, W.: Alternative approaches to default logic, *Artificial Intelligence* **70** (1994), 167–237.
12. Dimopoulos, Y.: The computational value of joint consistency, in L. Pereira and D. Pearce (eds), *European Workshop on Logics in Artificial Intelligence*, Springer, Berlin, 1994, pp. 50–65.
13. Doyle, J.: A truth maintenance system, *Artificial Intelligence* **12** (1979), 231–272.
14. Eder, E.: *Relative Complexities of First Order Calculi,* Vieweg, Braunschweig, 1992.
15. Etherington, D.: *Reasoning with Incomplete Information*, Research Notes in Artificial Intelligence, Pitman/Morgan Kaufmann, London, 1988.
16. Etherington, D. and Reiter, R.: On inheritance hierarchies with exceptions, in *Proc. AAAI Nat. Conf. Artificial Intelligence*, 1983, pp. 104–108.

17. Gelfond, M. and Lifschitz, V.: Logic programs with classical negation, in *Proc. Int. Conf. Logic Programming*, 1990, pp. 579–597.
18. Gottlob, G.: Complexity results for nonmonotonic logics, *J. Logic and Computation* **2**(3) (1992), 397–425.
19. Junker, U. and Konolige, K.: Computing the extensions of autoepistemic and default logic with a TMS, in *Proc. AAAI Nat. Conf. Artificial Intelligence*, 1990.
20. Konolige, K.: On the relation between default and autoepistemic logic, *Artificial Intelligence* **35**(2) (1988), 343–382.
21. Letz, R., Bayerl, S., Schumann, J. and Bibel, W. SETHEO: A high-performance theorem prover, *J. Automated Reasoning* **8**(2) (1992), 183–212.
22. Łukaszewicz, W.: Considerations on default logic–an alternative approach, *Computational Intelligence* **4** (1988), 1–16.
23. Mercer, R.: Using default logic to derive natural language suppositions, in *Proc. Canadian Soc. Computational Studies of Intelligence Conference*, 1988, pp. 14–21.
24. Moore, R.: Semantical considerations on nonmonotonic logics, *Artificial Intelligence* **25** (1985), 75–94.
25. Neugebauer, G.: From horn clauses to first order logic: A graceful ascent, Technical Report AIDA-92-21, FG Intellektik, FB Informatik, TH Darmstadt, 1992.
26. Neugebauer, G. and Schaub, T.: A pool-based connection calculus, Technical Report AIDA-91-2, FG Intellektik, FB Informatik, TH Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, Germany, January 1991.
27. Niemelä, I.: Decision procedure for autoepistemic logic, in *Proc. Conf. Automated Deduction*, Argonne, 1988, pp. 675–684.
28. Niemelä, I.: A decision method for nonmonotonic reasoning based on autoepistemic reasoning, in J. Doyle, P. Torasso and E. Sandewall (eds), *Proc. 4th Int. Conf. Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1994, pp. 473–484.
29. Poole, D., Goebel, R. and Aleliunas, R.: Theorist: A logical reasoning system for defaults and diagnosis, in N. Cercone and G. McCalla (eds), *The Knowledge Frontier: Essays in the Representation of Knowledge*, Chapter 13, Springer, New York, 1987, pp. 331–352.
30. Reiter, R.: A logic for default reasoning, *Artificial Intelligence* **13**(1–2) (1980), 81–132.
31. Reiter, R.: A theory of diagnosis from first principles, *Artificial Intelligence* **32**(1) (1987), 57–96.
32. Risch, V.: Les Tableaux Analytiques au Service des Logiques de Defauts, PhD Thesis, Université Aix-Marseille II, G.I.A., Parc Scientifique et Technologique de Luminy, April 1993.
33. Rothschild, A.: Algorithmische Untersuchungen zu Defaultlogiken, Master Thesis, FG Intellektik, FB Informatik, TH Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, 1993.
34. Schaub, T.: Assertional default theories: A semantical view, in J. Allen, R. Fikes and E. Sandewall (eds), *Proc. 2nd Int. Conf. Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991, pp. 496–506.
35. Schaub, T.: On commitment and cumulativity in default logics, in R. Kruse and P. Siegel (eds), *Proc. European Conf. Symbolic and Quantitative Approaches to Uncertainty*, Springer, Berlin, 1991, pp. 304–309.
36. Schaub, T.: Considerations on Default Logics, PhD Thesis, Technische Hochshule Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, Germany, November 1992.
37. Schaub, T.: On constrained default theories, in B. Neumann (ed.), *Proc. European Conf. on Artificial Intelligence*, Wiley, New York, 1992, pp. 304–308.
38. Schaub, T.: Variations of constrained default logic, in M. Clarke, R. Kruse and S. Moral (eds), *Proc. European Conf. Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, Springer, Berlin, 1993, pp. 312–317.
39. Schaub, T.: Computing queries from prioritized default theories, in Z. Ras and M. Zemankova (eds), *8th Int. Symp. Methodologies for Intelligent Systems*, Springer, Berlin, 1994, pp. 584–593.
40. Schaub, T. and Thielscher, M.: A method for skeptical reasoning in constrained default logic, Technical Report, FG Intellektik, FB Informatik, TH Darmstadt, 1994.

41. Schwind, C.: A tableaux-based theorem prover for a decidable subset of default logic, in M. E. Stickel (ed.), *CADE-10*, Springer, Berlin, 1990.
42. Schwind, C. and Risch, V.: A tableaux-based characterization for default logic, in R. Kruse (ed.), *Proc. European Conf. Symbolic and Quantitative Approaches to Uncertainty*, Springer, Berlin, 1991, pp. 310–317.
43. Slaney, J. SCOTT: A model-guided theorem prover, in *Proc. Int. Joint Conf. on Artificial Intelligence*, 1993, pp. 109–114.
44. Stickel, M.: A Prolog technology theorem prover, *New Generation Computing* **2** (1984), 371–383.
45. Thielscher, M. and Schaub, T.: Default reasoning by deductive planning, *J. Automated Reasoning* **15**(1) (1995), 1–40.
46. Zhang, A. and Marek, W.: On the classification and existence of structures in default logic, *Fundamenta Informaticae* **8**(4) (1990), 485–499.