Abstract

Recently, considerable interest and research effort has been given to the problem of finding a suitable extension of the logic programming paradigm beyond the class of normal logic programs. In order to demonstrate that a class of programs can be justifiably called an extension of logic programs one should be able to argue that:

- the proposed syntax of such programs resembles the syntax of logic programs but it applies to a significantly broader class of programs;
- the proposed semantics of such programs constitutes an intuitively natural extension of the semantics of normal logic programs;
- there exists a reasonably simple procedural mechanism allowing, at least in principle, to compute the semantics;
- the proposed class of programs and their semantics is a special case of a more general non-monotonic formalism which clearly links it to other well-established non-monotonic formalisms.

In this paper we propose a specific class of extended logic programs which will be (modestly) called super logic programs or just super-programs. We will argue that the class of super-programs satisfies all of the above conditions, and, in addition, is sufficiently flexible to allow various application-dependent extensions and modifications. We also provide a brief description of a Prolog implementation of a query-answering interpreter for the class of super-programs which is available via FTP and WWW.

Keywords: Non-Monotonic Reasoning, Logics of Knowledge and Beliefs, Semantics of Logic Programs and Deductive Databases.

1

1An extended abstract of this paper appeared in the Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Boston, Massachusetts, 1996, pp. 529–541.

1Partially supported by the National Science Foundation grant #IRI-9313061.
1 Introduction

Recently, considerable interest and research effort\(^1\) has been given to the problem of finding a suitable extension of the logic programming paradigm beyond the class of normal logic programs. In particular, considerable work has been devoted to the problem of defining natural extensions of logic programming that ensure a proper treatment of disjunctive information. However, the problem of finding a suitable semantics for disjunctive programs and databases proved to be far more complex than it is in the case of normal, non-disjunctive programs\(^2\).

There are good reasons justifying this extensive research effort. In natural discourse as well as in various programming applications we often use disjunctive statements. One particular example of such a situation is reasoning by cases. Other obvious examples include:

- **Approximate information**: for instance, an age “around 30” can be 28, 29, 30, 31, or 32;
- **Legal rules**: the judge always has some freedom for his decision, otherwise he/she would not be needed; so laws cannot have unique models;
- **Diagnosis**: only at the end of a fault diagnosis we may know exactly which part of some machine was faulty but as long as we are searching, different possibilities exist;
- **Biological inheritance**: if the parents have blood groups A and 0, the child must also have one of these two blood groups (example from [Lip79]);
- **Natural language understanding**: here there are many possibilities for ambiguity and they are represented most naturally by multiple intended models;
- **Reasoning about concurrent processes**: since we do not know the exact sequence in which certain operations are performed, again multiple models come into play;
- **Conflicts in multiple inheritance**: if we want to keep as much information as possible, we should assume disjunction of the inherited values [BL93].

Formalisms promoting disjunctive reasoning are more expressive and natural to use since they permit direct translation of disjunctive statements from natural language and from informal specifications. The additional expressive power of disjunctive logic programs [EG93, EGM94, EG96] significantly simplifies the problem of translation of non-monotonic formalisms into logic programs, and, consequently, facilitates using logic programming as an inference engine for non-monotonic reasoning. Moreover, extensive recent work devoted to theoretic and algorithmic foundations of disjunctive programming suggests that there are good prospects for extending the logic programming paradigm to disjunctive programs.

What then should be viewed as an “extension of logic programming”? We believe that in order to demonstrate that a class of programs can be justifiably called an extension of logic programs one should be able to argue that:

- the proposed syntax of such programs resembles the syntax of logic programs but it applies to a significantly broader class of programs, which includes the class of disjunctive programs.

---

\(^1\)It suffices just to mention several recent workshops on Extensions of Logic Programming specifically devoted to this subject ([DPP95, DPP97, DHSH96]).

\(^2\)The book by Minker et. al. [LMR92] provides a detailed account of the extensive research effort in this area. See also [Dix95, Min93, Prz95a, Prz95b].
tive logic programs as well as the class of logic programs with strong (or “classical”) 
negation;

• the proposed semantics of such programs constitutes an intuitively natural extension
  of the semantics of normal logic programs, which, when restricted to normal logic pro-
grams, coincides with one of the well-established semantics of normal logic programs;

• there exists a reasonably simple procedural mechanism allowing, at least in principle,
to compute the semantics⁢³;

• the proposed class of programs and their semantics should be a special case of a more
  general non-monotonic formalism which would clearly link it to other well-established
  non-monotonic formalisms.

In this paper we propose a specific class of such extended logic programs which will be
(modestly) called super logic programs or just super-programs. We will argue that the class
of super-programs satisfies all of the above conditions, and, in addition, is sufficiently flexible
to allow various application-dependent extensions and modifications. We also provide a
brief description of a Prolog implementation of a query-answering interpreter for the class
of super-programs which is available via FTP and WWW⁴.

The paper is organized as follows. In the next Section 2 we recall the definition and
basic properties of non-monotonic knowledge bases, a rather simple and yet highly expressive
class of non-monotonic theories introduced earlier by Przymusinski [Prz97]. Subsequently,
in Section 3, we establish a new and powerful fixed-point characterization of the semantics of
such knowledge bases, which provides the foundation for the remaining results obtained in
the paper. In Section 4 we introduce the class of super logic programs as a special subclass
of the class of all non-monotonic knowledge bases and we establish basic properties of super
programs. In the following Sections 5 and 6 we describe two important characterizations,
one of which is syntactic and the other model-theoretic, of the semantics of super-programs
which, due to the restricted nature of super programs, are significantly simpler than those
applicable to arbitrary non-monotonic knowledge bases. Finally, in Section 7 we briefly
describe our implementation of a query-answering interpreter for super-programs which is
based on the previously established model-theoretic characterization of their semantics. We
conclude with some final remarks in Section 8. For the sake of clarity, most proofs are
contained in the Appendix.

2 Non-Monotonic Knowledge Bases

We first define the notion of a non-monotonic knowledge base. Super logic programs, defined
in the next section, are special knowledge bases.

Consider a fixed propositional language \( \mathcal{L} \) with standard connectives \( (\lor, \land, \neg, \rightarrow, \leftarrow, \leftrightarrow) \) and the propositional letters true and false. We denote the set of its propositions
by \( \text{At}_{\mathcal{L}} \). Extend the language \( \mathcal{L} \) to a propositional modal language \( \mathcal{L}_{\text{Not}} \) by augmenting

---

⁢³Observe that while such a mechanism cannot – even in principle – be efficient, due to the inherent NP-
completeness of the problem of computing answers just to positive disjunctive programs, it can be efficient
when restricted to specific subclasses of programs and queries and it can allow efficient approximation
methods for broader classes of programs.

⁴See ftp://ftp.informatik.uni-hannover.de/software/static/static.html.
it with a modal operator \( \text{Not} \), called the \text{default negation} operator. The formulae of the form \( \text{Not} \ F \), where \( F \) is an arbitrary formula of \( \mathcal{L}_{\text{Not}} \), are called \text{default negation atoms} or just \text{default atoms} and are considered to be atomic formulae in the extended propositional modal language \( \mathcal{L}_{\text{Not}} \). The formulae of the original language \( \mathcal{L} \) are called \text{objective}. Any propositional theory in the modal language \( \mathcal{L}_{\text{Not}} \) will be called a \text{non-monotonic knowledge base}.

**Definition 2.1 (Non-monotonic Knowledge Bases [Prz97])**

By a non-monotonic knowledge base we mean an arbitrary theory in the propositional language \( \mathcal{L}_{\text{Not}} \), i.e., a (possibly infinite) set of arbitrary clauses of the form:

\[
B_1 \land \ldots \land B_m \land \text{Not} \ G_1 \land \ldots \land \text{Not} \ G_k \rightarrow A_1 \lor \ldots \lor A_l \lor \text{Not} \ F_1 \lor \ldots \lor \text{Not} \ F_n
\]

where \( m, n, k, l \geq 0 \), \( A_i \)'s and \( B_i \)'s are objective atoms and \( F_i \)'s and \( G_i \)'s are arbitrary formulae of \( \mathcal{L}_{\text{Not}} \).

By an affirmative knowledge base we mean any such theory all of whose clauses satisfy the condition that \( l \neq 0 \).

By a rational knowledge base we mean any such theory all of whose clauses satisfy the condition that \( n = 0 \).

In other words, \text{affirmative} knowledge bases are precisely those theories that satisfy the condition that all of their clauses contain at least one positive \text{objective} atom. On the other hand, \text{rational} knowledge bases are precisely those theories none of whose clauses contain any positive default atoms \( \text{Not} \ F_i \). Observe that arbitrarily deep level of \text{nested default negations} is allowed in the language.

We assume the following two simple axiom schemata and one inference rule describing the arguably obvious properties of default atoms:

**(CA) Consistency Axiom:**

\[
\text{Not} \ (\text{false}) \ \text{and} \ \neg \text{Not} \ (\text{true}) \tag{1}
\]

**(DA) Distributive Axiom:**

For any formulae \( F \) and \( G \):

\[
\text{Not} \ (F \lor G) \leftrightarrow \text{Not} \ F \land \text{Not} \ G \tag{2}
\]

**(IR) Invariance Inference Rule:**

For any formulae \( F \) and \( G \):

\[
\frac{F \leftrightarrow G}{\text{Not} \ F \leftrightarrow \text{Not} \ G} \tag{3}
\]

The consistency axiom (CA) states that \text{false} is assumed to be false by default but \text{true} is not. The second axiom (DA) states that default negation \text{Not} is distributive with respect to disjunctions. The invariance inference rule (IR) states that if two formulae are known to be equivalent then so are their default negations. In other words, the meaning of \text{Not} \( F \) does not depend on the specific form of the formula \( F \), e.g., the formula \text{Not} \( (F \land \neg F) \) is equivalent to \text{Not} \( (\text{false}) \) and thus is true by (CA).
Remark 2.1 Observe that, in general, we do not assume the distributive axiom for conjunction:
\[ \text{Not} (F \land G) \leftrightarrow \text{Not} F \lor \text{Not} G. \]
This is because from the fact that \( F \land G \) is assumed false by default it may not necessarily follow that one of the formulae \( F \) or \( G \) should be assumed false by default. For example, from the fact that we can assume by default that we do not drink and drive, we do not necessarily want to conclude that we can either assume by default that we don’t drink or assume by default that we don’t drive. Expressed in the language of beliefs (see Section 2.1), this is equivalent to saying that a belief \( B(F \lor G) \) in a disjunction of formulae \( F \) and \( G \) does not necessarily imply belief \( B F \) in \( F \) or belief \( B G \) in \( G \). Note, however, that the opposite direction of the above equivalence, namely \( \text{Not} (F \land G) \leftarrow \text{Not} F \lor \text{Not} G \), easily follows from our axioms. In some specific applications, the inclusion of the distributive axiom for conjunction may be justified; in such cases it should simply be added to the above listed axioms.

A (propositional) interpretation of \( L_{\text{Not}} \) is a mapping \( I : \text{At}_L \cup \{ \text{Not} (F) : F \in L_{\text{Not}} \} \rightarrow \{ \text{true}, \text{false} \} \), i.e. we simply treat the formulas \( \text{Not} (F) \) as new propositions. Therefore, the notion of a model carries over from propositional logic. A formula \( F \in L_{\text{Not}} \) is a propositional consequence of \( T \subseteq L_{\text{Not}} \) iff for every \( I : I \models T \implies I \models F \). In the examples, we will represent models by sets of literals showing the truth values of only those objective and default negation atoms which are relevant to our considerations.

Definition 2.2 (Derivable Formulae [Prz97])
For any knowledge base \( T \) we denote by \( Cn_{\text{Not}}(T) \) the smallest set of formulae of the language \( L_{\text{Not}} \) which contains \( T \), all (substitution instances of) the axioms (CA) and (DA) and is closed under both propositional consequence and the invariance rule (IR).

We say that a formula \( F \) is derivable from the knowledge base \( T \) if \( F \) belongs to \( Cn_{\text{Not}}(T) \). We denote this fact by \( T \vdash_{\text{Not}} F \). A knowledge base \( T \) is consistent if \( Cn_{\text{Not}}(T) \) is consistent.

2.1 Equivalence to Belief Theories
The notion of a non-monotonic knowledge base defined above is completely equivalent to the notion of a belief theory in the Autoepistemic Logic of Beliefs, AEB, originally defined by Przymusinski in [Prz97].

Belief theories were defined using a propositional language \( L \) augmented with a belief operator \( B \) instead of the default negation operator \( \text{Not} \). In order to translate a non-monotonic knowledge base into a belief theory one all needs to do is to replace everywhere \( \text{Not} F \) by \( B \neg F \) thus giving the default negation \( \text{Not} F \) the intended meaning of “\( \neg F \) is believed”:
\[ \text{Not} F \equiv B \neg F. \]

Conversely, in order to translate a belief theory into a non-monotonic knowledge base all one needs to do is to replace everywhere \( BF \) by \( \text{Not} \neg F \) thus giving the belief atoms \( BF \) the intended meaning of “\( F \) can be assumed true by default”:
\[ BF \equiv \text{Not} \neg F. \]
Readers interested in more details should consult Section A of the Appendix where we recall the definition of belief theories in \textit{AEB} and precisely establish the equivalence between non-monotonic knowledge bases and belief theories.

While the two notions are equivalent, the notion of a non-monotonic knowledge base seems better suited for the introduction of the class of super logic programs and thus throughout most of the paper we use the language of non-monotonic knowledge bases. However, for the sake of completeness, major results of this paper are also restated in the language of belief theories.

From the equivalence of the two logics (see Proposition A.2) it follows immediately that the following lemma, which will be needed in the sequel, holds.

\textbf{Proposition 2.3} For any knowledge base \( T \) and for any two formulae \( F \) and \( G \) of \( \mathcal{L}_{\text{Not}} \):

\[
\begin{align*}
T & \vdash_{\text{Not}} Not F \rightarrow \neg \neg F \\
T & \vdash_{\text{Not}} \neg (F \rightarrow G) \rightarrow (\neg F \rightarrow \neg \neg G) \\
\text{If } T & \vdash_{\text{Not}} F \text{ then } T \vdash_{\text{Not}} \neg \neg F.
\end{align*}
\]

The first statement says that if a formula \( F \) is assumed to be false by default then its negation \( \neg F \) is not (i.e., \( F \) is not true by default). The second statement says that if we assume that the formulae \( F \rightarrow G \) and \( F \) hold by default then we can also assume that the formula \( G \) holds by default. We recall again that the formula \( \text{Not } \neg F \) has the intuitive reading of “\( F \) can be assumed true by default”. The last inference rule says that if a formula \( F \) is derivable then it is true by default.

\section{2.2 Intended Meaning of Default Negation}

As the name indicates, non-monotonic knowledge bases must be equipped with a \textit{non-monotonic semantics}. Intuitively this means that we need to provide a meaning to the default negation atoms \( \text{Not } F \). We want the intended meaning of default atoms \( \text{Not } F \) to be based on the principle of predicate minimization (see [Min82, GPP89] and [McC80]):

\[
\text{Not } F \text{ holds if } \neg F \text{ is minimally entailed or, equivalently:}
\]

\[
\text{Not } F \text{ holds if } F \text{ is false in all minimal models.}
\]

In order to make this intended meaning precise we first have to define what we mean by a minimal model of a knowledge base.

\textbf{Definition 2.4 (Minimal Models [Prz97])}

A model \( M \) is smaller than a model \( N \) if it contains the same default atoms but has fewer objective atoms, i.e.

\[
\{ p \in \text{At}_L : M \models p \} \subset \{ p \in \text{At}_L : N \models p \},
\]

\[
\{ F \in \mathcal{L}_{\text{Not}} : M \models \text{Not } (F) \} = \{ F \in \mathcal{L}_{\text{Not}} : N \models \text{Not } (F) \}.
\]

By a minimal model of a knowledge base \( T \) we mean a model \( M \) of \( T \) with the property that there is no smaller model \( N \) of \( T \). If a formula \( F \) is true in all minimal models of \( T \) then we write \( T \models_{\text{min}} F \) and say that \( F \) is minimally entailed by \( T \). \( \square \)
In other words, minimal models are obtained by first assigning arbitrary truth values to the default atoms and then minimizing objective atoms (see also [YY93]). For readers familiar with circumscription, this means that we are considering circumscription \( \text{CIRC}(T; \text{At}_L) \) of the knowledge base \( T \) in which objective atoms are minimized while the default atoms \( \text{Not } F \) are fixed, i.e., \( T \models_{\text{min}} F \equiv \text{CIRC}(T; \text{At}_L) \models F \).

**Example 2.5** Consider the following simple knowledge base \( T \):

\[
\begin{align*}
\text{Car} \\
\text{Car} \land \text{Not Broken} & \rightarrow \text{Runs}
\end{align*}
\]

Let us prove that \( T \) minimally entails \( \neg \text{Broken} \), i.e., \( T \models_{\text{min}} \neg \text{Broken} \). Indeed, in order to find minimal models of \( T \) we need to assign an arbitrary truth value to the only default atom \( \text{Not Broken} \), and then minimize the objective atoms \( \text{Broken}, \text{Car} \) and \( \text{Runs} \). We easily see that \( T \) has the following two minimal models (truth values of the remaining belief atoms are irrelevant and are therefore omitted):

\[
\begin{align*}
M_1 & = \{\text{Not Broken}, \text{Car}, \text{Runs}, \neg \text{Broken}\}; \\
M_2 & = \{\neg \text{Not Broken}, \text{Car}, \neg \text{Runs}, \neg \text{Broken}\}.
\end{align*}
\]

Since in both of them \( \text{Car} \) is true, and \( \text{Broken} \) is false, we deduce that \( T \models_{\text{min}} \text{Car} \) and \( T \models_{\text{min}} \neg \text{Broken} \).

2.3 Static Expansions

As in Moore’s Autoepistemic Logic, the intended meaning of default negation atoms in non-monotonic knowledge bases is enforced by defining suitable expansions of such databases.

**Definition 2.6 (Static Expansions of Knowledge Bases [Prz97])**

A non-monotonic knowledge base \( T^\circ \) is called a static expansion of a knowledge base \( T \) if it satisfies the following fixed-point equation:

\[
T^\circ = \text{Cn}_{\text{Not}}(T \cup \{\text{Not } F : T \models_{\text{min}} \neg F\}),
\]

where \( F \) ranges over all formulae of \( \mathcal{L}_{\text{Not}} \).

A static expansion \( T^\circ \) of \( T \) must therefore coincide with the database obtained by: (i) adding to \( T \) default negation \( \text{Not } F \) of every formula \( F \) that is false in all minimal models of \( T^\circ \), and, (ii) closing the resulting database under the consequence operator \( \text{Cn}_{\text{Not}} \).

**Remark 2.2** When stated in the language of belief theories, the above fixed-point definition takes the form:

\[
T^\circ = \text{Cn}_{\text{AEB}}(T \cup \{BF : T \models_{\text{min}} F\}).
\]

Clearly, for any formula \( F \) of \( \mathcal{L}_{\text{Not}} \) and for any static expansion \( T^\circ \) of \( T \) we have:

\[
T^\circ \models \text{Not } F \quad \text{if} \quad T^\circ \models_{\text{min}} \neg F,
\]

(4)
thus a formula $F$ is false by default in a static expansion $T^\circ$ if it is minimally entailed by $T^\circ$.

Consequently, the definition of static expansions formally enforces the intended meaning of default atoms described in the previous subsection. In general, the converse of (4) does not hold because default atoms may be forced in by the database itself. For example, the database $T = \{a \lor b, \text{Not } a\}$ has a consistent static expansion in which $\text{Not } a$ and $\text{Not } (\neg a \land \neg b)$ hold and yet $T \not\models_{\text{min}} \neg a$. However, in rational knowledge bases, i.e., those in which default atoms do not occur positively, the converse is also true. Intuitively, rational databases are characterized by the fact that defaults $\text{Not } F$ are not explicitly asserted but instead are only implicitly inferred by virtue of minimal entailment of the underlying formulae $F$.

**Theorem 2.7 (Semantics of Default Negation [Prz97])**

Let $T^\circ$ be a static expansion of a rational knowledge base $T$. For any formula $F$ of $\mathcal{L}_{\text{Not}}$ we have:

$$T^\circ \models \text{Not } F \iff T^\circ \models_{\text{min}} \neg F.$$  

It turns out that every knowledge base $T$ has the least (in the sense of set-theoretic inclusion) static expansion $\overline{T}$ which has an iterative definition as the least fixed point of the monotonic\(^5\) default closure operator:

$$\Psi_T(S) = \text{Cn}_{\text{Not}}(T \cup \{\text{Not } F : T^\alpha \models_{\text{min}} \neg F\}),$$

where $S$ is an arbitrary knowledge base and the $F$’s range over all formulae of $\mathcal{L}_{\text{Not}}$.

**Theorem 2.8 (Least Static Expansion [Prz97])**

Every knowledge base $T$ has the least static expansion, namely, the least fixed point $T^\lambda$ of the monotonic default closure operator $\Psi_T$.

More precisely, the least static expansion $\overline{T}$ of $T$ can be constructed as follows. Let $T^0 = T$ and suppose that $T^\alpha$ has already been defined for any ordinal number $\alpha < \beta$. If $\beta = \alpha + 1$ is a successor ordinal then define:

$$T^{\alpha+1} = \Psi_T(T^\alpha) := \text{Cn}_{\text{Not}}(T \cup \{\text{Not } F : T^\alpha \models_{\text{min}} \neg F\}),$$

where $F$ ranges over all formulae in $\mathcal{L}_{\text{Not}}$. Else, if $\beta$ is a limit ordinal then define $T^\beta = \bigcup_{\alpha < \beta} T^\alpha$. The sequence $\{T^\alpha\}$ is monotonically increasing and has a unique fixed point $\overline{T} = T^\lambda = \Psi_T(T^\lambda)$, for some ordinal $\lambda$.  

The above result allows us to establish the following useful characterization of the least static completion of a knowledge base.

**Theorem 2.9 (Characterization of Least Static Expansions)**

The least static expansion of a knowledge base $T$ coincides with the smallest theory $\hat{T}$ satisfying the conditions:

\(^5\)Strictly speaking the operator is only monotone [Prz97] on the lattice of all theories of the form $\text{Cn}_{\text{Not}}(T \cup \{\text{Not } F : F \in \mathcal{L}_{\text{Not}}\})$. See also Lemma B.2.
\begin{itemize}
\item 1. \( T \subseteq \hat{T} \);
\item 2. \( \hat{T} = Cn_{\text{Not}}(T) \);
\item 3. if \( \hat{T} \models_{\text{min}} \neg F \) then \( \text{Not } F \in \hat{T} \).
\end{itemize}

\textbf{Proof.} Clearly the least static expansion \( \overline{T} \) of \( T \) satisfies the above conditions 1–3 so \( \hat{T} \subseteq T \). Moreover, from the conditions 1–3 it follows that \( \Psi_T(\hat{T}) \subseteq \hat{T} \). Since the operator \( \Psi_T \) is monotonic on such theories (see [Prz97]), we conclude that \( T^1 = \Psi_T(T) \subseteq \Psi_T(\hat{T}) \subseteq \hat{T} \), and, more generally, \( T^\alpha \subseteq \hat{T} \), for any ordinal \( \alpha \). We conclude therefore that \( \hat{T} \supseteq T \). \( \square \)

Observe that the least static expansion \( \overline{T} \) of \( T \) contains those and only those formulae which are true in all static expansions of \( T \). It constitutes the so called \textit{static completion} of a knowledge base \( T \).

\textbf{Definition 2.10 (Static Completion and Static Semantics)}

The least static expansion \( \overline{T} \) of a knowledge database \( T \) is called the static completion of \( T \). It describes the static semantics of a knowledge base \( T \).

Consequently, like the predicate completion semantics of a logic program \( P \) is completely determined by its Clark’s completion \( \text{comp}(P) \), the static semantics of a knowledge base \( T \) is determined by its static completion \( \overline{T} \). It is easy to verify that a belief theory \( T \) either has a consistent static completion \( \overline{T} \) or it does not have any consistent static expansions at all. Moreover, it turns out that static completions of \textit{affirmative} belief theories, introduced in Definition 2.1, are always consistent.

\textbf{Theorem 2.11 (Consistency of Static Completions [Prz97])}

Static completions \( T \) of affirmative knowledge bases \( T \) are consistent. \( \square \)

It is time now to discuss some examples. For simplicity, unless explicitly needed, when describing static expansions we ignore nested defaults and list only those elements of the expansion that are “relevant” to our discussion, thus, for example, skipping such members of the expansion as \( \text{Not } (F \wedge \neg F) \), \( \text{Not } \neg \text{Not } (F \wedge \neg F) \), etc.

\textbf{Example 2.12} Consider the simple database discussed already in Example 2.5:

\begin{center}
\begin{tabular}{ll}
   Car & \\
   Car \wedge \text{Not Broken} & \rightarrow \text{Runs} \\
\end{tabular}
\end{center}

We already established that \( T \) minimally entails \( \neg \text{Broken} \), i.e., \( T \models_{\text{min}} \neg \text{Broken} \). As a result, the static completion \( \overline{T} \) of \( T \) contains \( \text{Not Broken} \). Consequently, as expected, the static completion \( \overline{T} \) of \( T \) derives \( \text{Not Broken} \) and \( \text{Runs} \) and thus coincides with the perfect model semantics of this stratified program. \( \square \)

\textbf{Example 2.13} Consider a slightly more complex knowledge base \( T \):

\begin{center}
\begin{tabular}{ll}
   \text{Not Broken} & \rightarrow \text{Runs} \\
   \text{Not Fixed} & \rightarrow \text{Broken}.
\end{tabular}
\end{center}
In order to iteratively compute its static completion $T$ we let $T^0 = T$. One easily checks that $T^0 \models_{\min} \neg Fixed$. Since:

$$T^1 = \Psi_T(T^0) = Cn_{Not}(T \cup \{Not F : T^0 \models_{\min} \neg F\}).$$

it follows that $Not Fixed \in T^1$ and therefore $Broken \in T^1$. Since:

$$T^2 = \Psi_T(T^1) = Cn_{Not}(T \cup \{Not F : T^1 \models_{\min} \neg F\}).$$

it follows that $Not \neg Broken \in T^2$. From Proposition 2.3 we conclude that $\neg Not Broken \in T^2$ and therefore $T^2 \models_{\min} \neg Runs$. Accordingly, since:

$$T^3 = \Psi_T(T^2) = Cn_{Not}(T \cup \{Not F : T^2 \models_{\min} \neg F\}),$$

we infer that $Not Runs \in T^3$. As expected, the static completion $\overline{T}$ of $T$, which contains $T^3$, asserts that the car is considered not to be fixed and therefore broken and thus is not in a running condition. Again, the resulting semantics coincides with the perfect model semantics of this stratified program.

3 Static Completions vs. Clark’s Completions

It is easy to see that if the knowledge base is finite then the construction of its static completion (or the least static expansion) will stop after countably many steps. However, a powerful and somewhat surprising result obtained in this section shows that static completions of finite knowledge bases $T$ are in fact obtained by means of a single iteration of the default closure operator $\Psi_T$. This result eliminates the need for multiple iterations in the computation of static completions and allows us to replace the fixed-point definition of static completions by the equivalent explicit definition given by $T = \Psi_T(T)$. By clarifying the notion of a static completion of a finite database it also provides the foundation for the remaining results obtained in the paper.

The second, closely related result establishes a very interesting and somewhat intriguing relationship between static completions $\overline{T}$ and Clark’s completions $comp(T)$ of finite knowledge bases. It shows that the static completion $\overline{T}$ of a knowledge base $T$ coincides with $Cn_{Not}(T \cup \{Not \neg F : F \in comp(T)\})$, i.e., with the set of formulae derivable from the knowledge base $T$ and the formulae $\{Not \neg F : F \in comp(T)\}$. Since Clark’s completion $comp(T)$ is easily computable, this result reduces reasoning under the static semantics to theorem-proving in the underlying modal logic (this can be done either by hand or using an automated theorem prover).

The proof of these two powerful results is based on the idea of adding to a knowledge base $T$ the set of formulae which ensure that models “seen” through the default negation operator $Not$ are in fact minimal. As we will soon see, this task can be accomplished by a suitable generalization of Clark’s completion but it only works for a restricted class of knowledge bases, namely those whose clauses do not have any objective premises. Such residual databases were previously introduced and investigated in the class of logic programs [Bry89, Bry90, DK89a, DK89b, BD97c, BD95]. Here we give a slightly more general definition.
Definition 3.1 (Residual Knowledge Bases [BD95])

By a residual knowledge base we mean an arbitrary non-monotonic knowledge base whose clauses do not contain any objective (positive) premises, i.e., a (possibly infinite) set of arbitrary clauses of the form:

\[
\text{Not } G_1 \land \ldots \land \text{Not } G_k \rightarrow A_1 \lor \ldots \lor A_1 \lor \text{Not } F_1 \lor \ldots \lor \text{Not } F_n
\]

where \( n, k, l \geq 0 \), \( A_i \)'s are objective atoms and \( F_i \)'s and \( G_i \)'s are arbitrary formulae of \( L_{\text{Not}} \).

Whenever convenient, clauses of the form:

\[
\text{Not } G_1 \land \ldots \land \text{Not } G_k \rightarrow \text{false} \lor \text{Not } F_1 \lor \ldots \lor \text{Not } F_n
\]

i.e., clauses without any objective atoms in their heads, will be considered as clauses with a single objective atom \text{false} in their head, i.e., clauses of the form:

\[
\text{Not } G_1 \land \ldots \land \text{Not } G_k \rightarrow \text{false} \lor \text{Not } F_1 \lor \ldots \lor \text{Not } F_n.
\]

Now we describe a natural extension of the notion of Clark’s completion \( \text{comp}(T) \). Clark’s completion was initially introduced in [Cla78] for the class of normal logic programs. Subsequently, its generalizations to disjunctive programs without positive premises were studied in [BD95]. Here we extend it to the class of all residual knowledge bases. As usual, the goal is to ensure that an objective atom \( A \) is true only if it really has to be true, i.e., if there is a rule with \( A \) in the head, in which the body is true and all the other head literals are false.

Definition 3.2 (Clark’s Completion of Residual Knowledge Bases [Cla78])

Given a finite residual knowledge base \( T \) and an objective atom \( A \) from \( L \) (including the falsity atom \text{false} ) we define Clark’s completion \( \text{comp}(A, T) \) of the atom \( A \) in \( T \) to be the formula:

\[
A \leftrightarrow \bigvee_{1 \leq m \leq s} (\text{Not } G_{1,m} \land \ldots \land \text{Not } G_{k,m} \land \neg A_{1,m} \land \ldots \land \neg A_{l,m} \land \neg \text{Not } F_{1,m} \land \ldots \land \neg \text{Not } F_{n,m}),
\]

where

\[
\text{Not } G_{1,m} \land \ldots \land \text{Not } G_{k,m} \rightarrow A \lor A_{1,m} \lor \ldots \lor A_{l,m} \lor \text{Not } F_{1,m} \lor \ldots \lor \text{Not } F_{n,m},
\]

for \( 1 \leq m \leq s \), are all clauses in \( T \) containing the atom \( A \) in their heads.

By Clark’s completion \( \text{comp}(T) \) of a knowledge base \( T \) we mean the union of all completions \( \text{comp}(A, T) \) of \( A \) in \( T \), for all objective atoms \( A \) in the language \( L \) (including the falsity atom \text{false} ).

The following interesting result shows that models of Clark’s completion \( \text{comp}(T) \) of a residual knowledge base \( T \) are precisely minimal models of \( T \) itself. In other words, Clark’s completion precisely describes the minimal model semantics of a residual knowledge base \( T \).
Proposition 3.3 Let $T$ be a finite residual knowledge base. An interpretation $M$ of the language $\mathcal{L}_{\text{Not}}$ is a minimal model of $T$ if and only if $M$ is a model of Clark’s completion $\text{comp}(T)$ of $T$.

Proof. Contained in the Appendix.

It is worth pointing out that one of the weaknesses of the original version of Clark’s completion proposed in [Cla78] was the fact that it was applied not just to residual but rather to arbitrary normal programs. As a result, Clark’s original completion did not enforce minimal model semantics, e.g., a tautology like $p \leftarrow p$ made the completion axiom for $p$ useless. Another problem with the original definition of Clark’s completion was that it did not distinguish between logical negation $\neg p$ and default negation $\text{Not} p$. As a result, Clark’s completion could be inconsistent when rules like $p \leftarrow \text{Not} p$ were present.

Remark 3.1 It is interesting to note that when $T$ represents a disjunctive program and when we replace $\text{Not} C$ by $\neg C$, we in fact obtain stable models of $T$. We used this fact in [BD95] to compute the stable semantics of disjunctive programs.

Although the above result, as well as the definition of Clark’s completion, applies only to the class of residual knowledge bases, it turns out that every finite knowledge base $T$ can be transformed into a finite residual knowledge base $T_{\text{res}}$ so that their sets of minimal models coincide.

Proposition 3.4 Every finite knowledge base $T$ can be transformed into a finite residual knowledge base $T_{\text{res}}$, called the residuum of $T$, so that an interpretation $M$ of the language $\mathcal{L}_{\text{Not}}$ is a minimal model of $T$ if and only if $M$ is a minimal model of $T_{\text{res}}$.

Proof. Contained in the Appendix.

In fact, just two elementary database transformations, namely, “unfolding” (or partial evaluation, GPPE [BD97c, SS94]) and “elimination of tautologies”, are sufficient to obtain such a residual program. Details of the transformation are given in the Appendix; here we only give a simple example. Whenever convenient, we use the traditional inverse notation for database clauses.

Consider the following knowledge base $T$:

$p \lor q \leftarrow \text{Not} r$
$q \leftarrow \text{Not} q$
$r \leftarrow q.$

The first two clauses already have the required form. In the last clause, we unfold the body literal $q$ and obtain the residual program $T_{\text{res}}$:

$p \lor q \leftarrow \text{Not} r$
$q \leftarrow \text{Not} q$
$p \lor r \leftarrow \text{Not} r$
$r \leftarrow \text{Not} q.$
One can easily see that this procedure represents a form of hyper-resolution.

As an immediate consequence of Propositions 3.3 and 3.4 we obtain the following important theorem which says that for any finite knowledge base $T$, Clark’s completion of its residuum $T_{res}$ precisely describes the minimal model semantics of $T$.

**Theorem 3.5** Let $T$ be any finite knowledge base. An interpretation $M$ of the language $\mathcal{L}_{Not}$ is a minimal model of $T$ if and only if $M$ is a model of Clark’s completion $\text{comp}(T_{res})$ of the residuum $T_{res}$ of $T$.

Finally, let us denote by $\text{Not} \neg \text{comp}(T_{res})$ the set $\{\text{Not} \neg F : F \in \text{comp}(T_{res})\}$. Intuitively, augmenting a given knowledge base $S$ with $\text{Not} \neg \text{comp}(T_{res})$ ensures that all formulae that belong to Clark’s completion $\text{comp}(T_{res})$ are assumed to be true by default (in $S$). We will need the following lemma.

**Lemma 3.6** Let $T$ be an arbitrary finite knowledge base and let

$$\hat{T} = \text{Cn}_{\text{Not}}(T \cup \text{Not} \neg \text{comp}(T_{res})).$$

If a formula $F$ is false in all minimal models of $\hat{T}$ then $\text{Not} F$ is already contained in $\hat{T}$. In other words, $\Psi_T(\hat{T}) \subseteq \hat{T}$.

**Proof.** Contained in the Appendix.

Now we are ready to state and prove the first of the two fundamental results obtained in this section.

**Theorem 3.7 (Static Completion vs. Clark’s Completion)**
The static completion $\hat{T}$ of a finite knowledge base $T$ can be equivalently defined as:

$$\hat{T} = \text{Cn}_{\text{Not}}(T \cup \text{Not} \neg \text{comp}(T_{res})).$$

**Proof.** Let $\hat{T} = \text{Cn}_{\text{Not}}(T \cup \text{Not} (\neg \text{comp}(T_{res})))$. By Theorem 3.5, $F \in \neg \text{comp}(T_{res})$ if and only if $F$ is false in all minimal models of $T$ and therefore:

$$\text{Cn}(\text{Not}(\neg \text{comp}(T_{res}))) = \{\text{Not} F : T \models_{\text{min}} \neg F\}.$$

It follows that:

$$\hat{T} = \text{Cn}_{\text{Not}}(T \cup \{\text{Not} F : T \models_{\text{min}} \neg F\}) = \Psi_T(T).$$

From Lemma 3.6 it follows that $\Psi_T(\hat{T}) \subseteq \hat{T}$. Since $\hat{T} = \Psi_T(T)$ we conclude that:

$$\Psi_T(\Psi_T(T)) \subseteq \Psi_T(T)$$

which means that $\Psi_T(\Psi_T(T)) = \Psi_T(T)$ and therefore $\Psi_T(\hat{T}) = \hat{T}$. This proves that $\hat{T}$ is a fixed point of the default closure operator $\Psi_T$ and thus is a static expansion of $T$. Since $\hat{T} = \Psi_T(T)$ we conclude that $\hat{T}$ is the least such fixed point and thus coincides with the static completion $\hat{T}$ of $T$. 

$\Box$
The above result states that the static completion $\mathcal{T}$ of $T$ is obtained by augmenting $T$ with the set $\not\rightarrow \text{comp}(T_{\text{res}})$ thus ensuring that all formulae that belong to Clark’s completion $\text{comp}(T_{\text{res}})$ of $T_{\text{res}}$ are assumed to be true by default. It establishes an interesting and somewhat intriguing relationship between static completions and Clark’s completions of finite knowledge bases. It also reduces reasoning under the static semantics (i.e., the computation of static completions $\mathcal{T}$) to the easily accomplished computation of Clark’s completion $\text{comp}(T_{\text{res}})$ together with theorem-proving in the underlying modal logic (which can be done either by hand or using an automated theorem prover).

**Remark 3.2** When stated in the language of beliefs, the above result takes the form:

$$\mathcal{T} = Cn_{\text{AEB}}(T \cup B(\text{comp}(T_{\text{res}}))).$$

It states that the static completion $\mathcal{T}$ of $T$ is obtained by augmenting $T$ with beliefs $BF$ in all formulae $F$ which belong to Clark’s completion of $T_{\text{res}}$. $\square$

As a consequence of the proof of the above result we immediately obtain the second of the two fundamental theorems obtained in this section, namely a powerful and somewhat surprising result stating that static completion $\mathcal{T}$ of an arbitrary finite knowledge base $T$ is always obtained by a single iteration of the default closure operator $\Psi_T(T)$.

**Theorem 3.8 (Explicit Characterization of Static Completions)**

The static completion $\mathcal{T}$ of finite knowledge base $T$ is always obtained by a single iteration of the default closure operator $\Psi_T(T)$. More precisely:

$$\mathcal{T} = \Psi_T(T).$$

In other words,

$$\mathcal{T} = Cn_{\text{min}}(T \cup \{\not\rightarrow F : T \models_{\text{min}} \neg F\}). \quad \square$$

The above theorem states that the static completion $\mathcal{T}$ of $T$ is obtained by augmenting $T$ with negation by default $\not\rightarrow F$ of all formulae $F$ which are false in all minimal models of the theory $T$ itself. It eliminates therefore the need for multiple iterations in the computation of static completions and allows us to replace the fixed-point definition of static completions by the equivalent explicit definition. Needless to say, the existence of an equivalent non-fixed point definition of static completions significantly simplifies this notion and the underlying theory. It also provides the foundation for the remaining results obtained in the paper.

**Remark 3.3** When stated in the language of beliefs, the above result takes the form:

$$\mathcal{T} = Cn_{\text{AEB}}(T \cup \{BF : T \models_{\text{min}} F\}). \quad \square$$

It states that the static completion $\mathcal{T}$ of $T$ is obtained by augmenting $T$ with beliefs $BF$ in all formulae $F$ which are minimally entailed by the theory $T$ itself. $\square$

**Example 3.9** Consider the knowledge base $T$ from Example 2.13:

$$\begin{align*}
\not\rightarrow \text{Broken} & \rightarrow \text{Runs} \\
\not\rightarrow \text{Fixed} & \rightarrow \text{Broken}.
\end{align*}$$
We already noted that $T |\models_{\text{min}} \neg Fixed$ and thus

$$\text{Not Fixed} \in T = Cn_{\text{Not}} \left( T \cup \{ \text{Not } F : T |\models_{\text{min}} \neg F \} \right).$$

This implies that $\overline{T}$ contains also $\text{Broken}$. By Proposition 2.3, $\text{Not } \neg \text{Broken}$ belongs to the static completion and so does $\neg \text{Not Broken}$. From the same Lemma it follows that $\text{Not Not Broken}$ belongs to the completion.

Moreover, it is easy to verify that $T |\models_{\text{min}} \text{Not Broken} \leftrightarrow \text{Runs}$ (in fact, the formula $\text{Not Broken} \leftrightarrow \text{Runs}$ belongs to Clark’s completion of $T$) and therefore $T |\models_{\text{min}} \neg \text{Not Broken} \leftrightarrow \neg \text{Runs}$. It follows that $T$ contains $\text{Not } (\neg \text{Not Broken} \leftrightarrow \neg \text{Runs})$. By Lemma 2.3, $\overline{T}$ also contains $\text{Not Not Broken} \leftrightarrow \text{Not Runs}$. As a result $\overline{T}$ contains also $\text{Not Runs}$.

We again conclude that the static semantics of $T$ derives $\text{Not Fixed, Broken}$ and $\text{Not Runs}$. Observe that while the above reasoning utilizes only one iteration of the default closure operator, it is slightly more complex than the one used in Example 2.13 where several iterations were performed.

Remark 3.4 Our work is closely related to the paper [YY93] where the authors define the well-founded circumscriptive semantics for disjunctive programs (without nested default negations). They introduced the concept of minimal model entailment with fixed default atoms and defined their semantics as a single step application of circumscription.

3.1 Adding Strong Negation to Knowledge Bases

As pointed out in [GL90, AP92, AP96, APP97] in addition to default negation, $\text{Not } F$, non-monotonic reasoning requires another type of negation $\neg F$ which is similar to classical negation $\neg F$ but does not satisfy the law of the excluded middle (see [APP97] for more information).

Strong negation $\neg F$ can be easily added to knowledge databases by:

- Extending the original objective language $L$ by adding to it new objective propositional symbols $\neg A$, called strong negation atoms, for all $A \in \text{At}_L$; as a result we obtain a new objective language $\hat{L}$ and the new extended modal language $\hat{L}_{\text{Not}}$;

- Adding to the database $T$ the following strong negation clause, for any strong negation atom $\neg A$ that appears in $T$:

$$\left( S_A \right) \quad A \land \neg A \rightarrow \text{false},$$

which says that $A$ and $\neg A$ cannot be both true and thus ensures the intended meaning of $\neg A$ as “the opposite of $A$”.

For example, a proposition $A$ may describe the property of being “qualified” while the proposition $\neg A$ describes the property of being “unqualified”. The strong negation axiom states that a person cannot be both qualified and unqualified. We do not assume, however, that everybody is already known to be either qualified or unqualified.

Since the addition of strong negation clauses simply results in a new knowledge base, all the results stated so far apply as well to knowledge databases with strong negation.
4 Super Logic Programs

We now introduce the class of super logic programs, briefly called super programs, as a sub-class of the class of all non-monotonic knowledge bases. This powerful class of databases includes all (monotonic) propositional theories, all disjunctive logic programs and all extended logic programs with strong negation (see Section 3.1).

Definition 4.1 (Super Logic Programs)
A Super Logic Program is a non-monotonic knowledge base consisting of (possibly infinitely many) super-clauses of the form:

\[ F \leftarrow G \land \text{Not } H \]

where \( F \) and \( G \) are arbitrary objective formulae and \( H \) is an arbitrary positive objective formula.

Observe that since any formula \( F \leftarrow G \) can be written as a finite conjunction of disjunctions of literals, we can always replace a super-clause by a finite set of clauses whose heads are disjunctions \( A_1 \lor \ldots \lor A_k \) of atoms and whose positive premises are conjunctions \( B_1 \land \ldots \land B_m \) of objective atoms. Finally, since the operator \( \text{Not} \) obeys the distributive law for disjunction (DA), the default atom \( \text{Not } H \) can be replaced by the conjunction \( \text{Not } C_1 \land \ldots \land \text{Not } C_n \) of default atoms \( \text{Not } C_i \), where each of the \( C_i \)'s is a conjunction of objective atoms. This allows us to establish the following useful fact.

Proposition 4.2 A super logic program \( P \) can be equivalently defined as a set of (possibly infinitely many) clauses of the form:

\[ A_1 \lor \ldots \lor A_k \leftarrow B_1 \land \ldots \land B_m \land \text{Not } C_1 \land \ldots \land \text{Not } C_n, \]

where \( A_i \)'s and \( B_i \)'s are objective atoms and \( C_i \)'s are conjunctions of objective atoms.
If \( k \neq 0 \), for all clauses of \( P \), then the program is called affirmative.

Clearly the class of super-programs contains all (monotonic) propositional theories and is syntactically significantly broader than the class of normal logic programs. In fact, it is somewhat broader than the class of programs usually referred to as disjunctive logic programs because:

- It allows constraints, i.e., headless rules. In particular it allows the addition of strong negation to such programs, as shown in Section 3.1, by assuming the strong negation axioms \( \leftarrow A \land \lnot A \), which themselves are program rules (rather than meta-level constraints);

- It allows premises of the form \( \text{Not } C \), where \( C \) is not just an atom but a conjunction of atoms. This proves useful when reasoning with default assumptions which themselves are rules, such as \( \text{Not} (\text{man} \land \lnot \text{human}) \), which can be interpreted as stating that we can assume by default that every man is a human (note that \( \lnot \text{human} \) represents strong negation of \( \text{human} \)).

\(^6\)I.e., a formula that can be represented as a disjunction of conjunctions of atoms.
Although the class of super programs is quite broad, it is significantly less general than the class of all non-monotonic knowledge bases because it does not allow any nesting of default negation. As a result, static completions of super programs admit much simpler and computationally more feasible characterizations. They allow the computation of their relevant parts without the need to involve arbitrarily deeply nested default negations, which are inherently infinite even for finite programs.

In the following two sections we provide two such powerful characterizations of static completions of super programs and one of them is later used to implement a query-answering interpreter for super programs. First, however, we discuss general properties of super programs and give a simple example.

The class of super programs has a number of useful properties. Since super programs are rational knowledge bases, from Theorem 2.7 it immediately follows that for any formula \( F \) the static semantics (or completion) \( \overline{P} \) of a super program \( P \) derives \( \neg F \) if and only if \( \neg F \) is minimally entailed by \( \overline{P} \).

**Theorem 4.3** Let \( P \) be any super logic program and let \( \overline{P} \) be its static completion. For any formula \( F \), \( \overline{P} \models \neg F \iff \overline{P} \models_{\text{min}} \neg F \). \( \square \)

The next result summarizes basic properties of super programs. It is an immediate consequence of Proposition A.2 and the results established in [Prz97].

**Theorem 4.4 (Basic Properties of Super Programs)**

- Every affirmative super program has a consistent static completion. In particular, this applies to all disjunctive logic programs.

- For normal logic programs, static semantics coincides with the well-founded semantics. More precisely, there is a one-to-one correspondence between consistent static expansions of a normal program and its partial stable models. Under this correspondence, total stable models correspond to those consistent static expansions which satisfy the axiom \( \neg A \lor \neg \neg A \), for every objective atom \( A \).

- For positive disjunctive logic programs, static semantics coincides with the minimal model semantics. \( \square \)

**Example 4.5** Consider a simple super-program \( P \):

\[
\begin{align*}
\text{VisitEurope} \lor \text{VisitAustralia} & \leftarrow \text{VisitEurope} \lor \text{VisitAustralia} \\
\text{Happy} & \leftarrow \text{VisitEurope} \lor \text{VisitAustralia} \\
\text{Bankrupt} & \leftarrow \text{VisitEurope} \land \text{VisitAustralia} \\
\text{Prudent} & \leftarrow \neg (\text{VisitEurope} \land \text{VisitAustralia}) \\
\text{Disappointed} & \leftarrow \neg (\text{VisitEurope} \lor \text{VisitAustralia}).
\end{align*}
\]

Given the usual exclusive interpretation of disjunctions we would most likely conclude that we will either visit Australia or Europe but not both and thus we expect the answer to the query \( \text{VisitEurope} \lor \text{VisitAustralia} \) to be positive while the answer to the query
VisitEurope \land VisitAustralia to be negative. As a result, we expect a positive answer
to the queries Happy and Prudent and a negative answer to the queries Bankrupt and Disappointed.

Observe that the query Not(VisitEurope \land VisitAustralia) intuitively means “can it be
assumed by default that we don’t visit both Europe and Australia?” and thus it is different
from the query Not VisitEurope \land Not VisitAustralia which says “can it be assumed by
default that we don’t visit Europe and can it be assumed by default that we don’t visit
Australia?”.

It turns out that the static semantics produces precisely the intended meaning discussed
above. Indeed, clearly Happy must belong to the static completion of P. It is easy to
check that Not VisitEurope \lor Not VisitAustralia or Not(VisitEurope \land VisitAustralia) holds in
all minimal models of the program P and therefore Not(VisitEurope \land VisitAustralia)
must be true in the completion. This proves that Prudent must hold in the comple-
tion. Moreover, since Bankrupt is false in all minimal models Not Bankrupt must
also belong to the completion. Finally, since VisitEurope \lor VisitAustralia is true,
we infer from Proposition 2.3 that Not (VisitEurope \lor VisitAustralia) and thus also
Not (VisitEurope \lor VisitAustralia) holds. This shows that Disappointed is minimally
entailed which implies that Disappointed belongs to the completion.

5 Fixed Point Characterization of Static Completions

Due to Proposition 4.2, we can consider the language of super logic programs to be restricted
to the subset \( L_{Not}^* \subseteq L_{Not} \), which is the propositional logic over the following set of atoms:
\[
At_L \cup \{ Not \ E : E \text{ is a conjunction of objective atoms from } At_L \}.
\]
In particular, the language \( L_{Not}^* \) does not allow any nesting of default negations. In fact,
default negation can be only applied to conjunctions of objective atoms. Note, however,
that we allow also the “empty conjunction”, so Not (true) is contained in \( L_{Not}^* \).

Clearly, in order to answer queries about a super program P we only need to know
which formulae of the restricted language \( L_{Not}^* \) belong to the static completion of P. In
other words, we only need to compute the restriction \( P|L_{Not}^* \) of the static completion \( P \)
to the language \( L_{Not}^* \). It would be nice and computationally a lot more feasible to be
able to compute this restriction without having to first compute the full completion, which
involves arbitrarily deeply nested default negations and thus is inherently infinite even
for finite programs. The following important result provides a positive solution to this
problem in the form of a much simplified syntactic fixed point characterization of static
completions. In the next section we provide yet another solution in the form of a model-
theoretic characterization of static completions.

**Theorem 5.1 (Fixed-Point Characterization of Static Completions)**

The restriction of the static completion \( \overline{P} \) of a finite super program P to the language \( L_{Not}^* \)
can be constructed as follows. Let \( P^0 = P \) and suppose that \( P^n \) has already been defined for
some natural number n. Define \( P^{n+1} \) as follows:
\[
Cn( P \cup \{ Not \ E_1 \land \ldots \land Not \ E_m \rightarrow Not \ E_0 : P^m \models_{\min} \neg \exists \ E_1 \land \ldots \land \neg \ E_m \rightarrow \neg \ E_0 \} ),
\]
where \( E_i \)'s range over all conjunctions of objective atoms and \( Cn \) denotes the standard propositional consequence operator. We allow the special case that \( E_0 \) is the empty conjunction (i.e. true) and identify Not true with false.

The sequence \( \{ P^n \} \) is monotonically increasing and has a unique fixed point \( P^{n_0} = P^{n_0+1} \), for some natural number \( n_0 \). Moreover, \( P^{n_0} = P|\mathcal{L}_{\mathit{Not}} \), i.e., \( P^{n_0} \) is the restriction of the static completion of \( P \) to the language \( \mathcal{L}_{\mathit{Not}}^* \).

**Proof.** Contained in the Appendix.

The above Theorem represents a considerable simplification over the original characterization of static completions given in Theorem 2.8. First of all, instead of using the modal consequence operator \( Cn_{\mathit{Not}} \) the definition uses the standard propositional consequence operator \( Cn \). Moreover, instead of ranging over the set of all (arbitrarily deeply nested) formulae of the language \( \mathcal{L}_{\mathit{Not}} \) it involves only conjunctions of objective atoms. These two simplifications greatly enhance the implementability of static semantics of finite programs. On the other hand, due to the restriction to the language \( \mathcal{L}_{\mathit{Not}}^* \), we can no longer expect the fixed point to be reached in just one step, as in Theorem 3.8.

### 6 Model-Theoretic Characterization of Static Completions

In this section, we complement Theorem 5.1 by providing a model-theoretic characterization of static completions of finite super programs. More precisely, we characterize models of static completions restricted to the narrower language \( \mathcal{L}_{\mathit{Not}}^* \). The resulting characterization was directly used in a prototype implementation of a query answering interpreter for static semantics described briefly in the next section.

**Definition 6.1 (Reduced Interpretations)**

1. Let \( \text{OBJ} \) be the set of all propositional valuations of the objective atoms \( \text{At}_L \).
2. Let \( \text{NOT} \) be the set of all propositional valuations of the default atoms \( \{ \text{Not} (p_1 \land \cdots \land p_n) : p_i \in \text{At}_L \} \), which interpret Not (true) (in case \( n = 0 \)) as false.
3. We call an interpretation \( I \) of the language \( \mathcal{L}_{\mathit{Not}}^* \), i.e. a valuation of \( \text{At}_L \cup \{ \text{Not} (p_1 \land \cdots \land p_n) : p_i \in \text{At}_L \} \), a reduced interpretation and write it as as \( I = I_{\text{obj}} \cup I_{\text{not}} \) with \( I_{\text{obj}} \in \text{OBJ} \) and \( I_{\text{not}} \in \text{NOT} \).
4. In order to emphasize the difference, we sometimes call an interpretation \( I \) of the complete language \( \mathcal{L}_{\mathit{Not}} \), i.e. a valuation for \( \text{At}_L \cup \{ \text{Not} (F) : F \in \mathcal{L}_{\mathit{Not}} \} \), a full interpretation.
5. Given a full interpretation \( I \), we call its restriction \( I = I_{\text{obj}} \cup I_{\text{not}} \) to the language \( \mathcal{L}_{\mathit{Not}}^* \) the reduct of \( I \).

Since an interpretation \( I = I_{\text{obj}} \cup I_{\text{not}} \) assigns truth values to all atoms occurring in a super program, the “is model of” relation between such interpretations and super programs...
is well defined. We call \( I = I_{\text{obj}} \cup I_{\text{not}} \) a minimal model of a super program \( P \) if there is no objective interpretation \( I'_{\text{obj}} \) satisfying
\[
\{ p \in \text{At}_L : I'_{\text{obj}} \models p \} \subset \{ p \in \text{At}_L : I_{\text{obj}} \models p \}
\]
such that \( I' = I'_{\text{obj}} \cup I_{\text{not}} \) is also a model of \( P \). This is completely compatible with the corresponding notion for full interpretations.

Our goal in this section is to characterize the reducts of the full models of the static completion \( \overline{P} \) of a finite super program \( P \). In fact, once we have the possible interpretations of the default atoms, finding the objective parts of the reducts is easy. The idea how to compute such possible interpretations is closely related to Kripke structures (which is not surprising for a modal logic, for more details see Appendix C). The worlds in such a Kripke structure are marked with objective interpretations, and the formula \( \text{Not} (p_1 \land \cdots \land p_n) \) is true in a world \( w \) iff \( \neg(p_1 \land \cdots \land p_n) \) is true in all worlds \( w' \) which can be “seen” from \( w \). Due to the consistency axiom, every world \( w \) must see at least one world \( w' \). What the static semantics does is to ensure that every world \( w \) sees only worlds marked with minimal objective models. So given a set \( O \) of minimal objective models, an interpretation \( I_{\text{not}} \) of the default atoms is possible iff there is some subset \( O' \subseteq O \) (namely the objective interpretations in the worlds \( w' \)) such that \( I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n) \) iff \( I_{\text{obj}} \models \neg(p_1 \land \cdots \land p_n) \) for all \( I_{\text{obj}} \in O' \). But the minimal objective models \( O \) also depend conversely on the possible interpretations of the default atoms (since a minimal model contains an objective part and a default part). So when we restrict the possible default interpretations, we also get less minimal objective models. This obviously results in a fixed-point computation, formalized by the following operators:

**Definition 6.2 (Possible Interpretation of Default Atoms)**

Let \( P \) be a super program.

1. The operator \( \Omega_P : 2^{\text{NOT}} \rightarrow 2^{\text{OBJ}} \) yields minimal objective models given a set of possible interpretations of the default atoms. For every \( N \subseteq \text{NOT} \), let \( \Omega_P(N) := \{ I_{\text{obj}} \in \text{OBJ} : \text{there is a } I_{\text{not}} \in N \text{ such that } I = I_{\text{obj}} \cup I_{\text{not}} \text{ is min. model of } P \} \).

2. The operator \( \Pi_P : 2^{\text{OBJ}} \rightarrow 2^{\text{NOT}} \) yields possible interpretations of the default atoms, given a set of minimal objective models. For every \( O \subseteq \text{OBJ} \), let \( \Pi_P(O) := \{ I_{\text{not}} \in \text{NOT} : \text{there is a non-empty } O' \subseteq O \text{ such that for every } p_1, \ldots, p_n \in \text{At}_L: I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n) \iff \text{for every } I_{\text{obj}} \in O': I_{\text{obj}} \models \neg(p_1 \land \cdots \land p_n) \} \).

3. The operator \( \Theta_P : 2^{\text{NOT}} \rightarrow 2^{\text{NOT}} \) is the composition \( \Theta_P := \Omega_P \circ \Pi_P \) of \( \Omega_P \) and \( \Pi_P \).

Now we are ready to state the main result of this section.

**Theorem 6.3 (Model-Theoretic Characterization of Static Completions)**

Let \( P \) be a finite super program:

1. The operator \( \Theta_P \) is monotone and thus its iteration beginning from the set \( \text{NOT} \) of all interpretations of the default atoms has a fixed point \( N^\infty \).
2. $\mathcal{N}^\circ$ consists exactly of all $I_{not} \in \mathcal{N}O\mathcal{T}$ that are default parts of a full model $\mathcal{I}$ of $\overline{\mathcal{P}}$.

3. A reduced interpretation $I_{obj} \cup I_{not}$ is a reduct of a full model $\mathcal{I}$ of $\overline{\mathcal{P}}$ iff $I_{not} \in \mathcal{N}^\circ$ and $I_{obj} \cup I_{not} \models \mathcal{P}$.

Proof. Contained in the Appendix.

Example 6.4 Let us consider the following logic program $P$:

\[
p \lor q \leftarrow \text{Not } r.
q \leftarrow \text{Not } q.
r \leftarrow q.
\]

Since only default atoms with a single atom occur in $P$, it suffices to consider only such default atoms. We start with the set $\mathcal{N}O\mathcal{T}$ containing all eight possible valuations of the three default atoms Not $p$, Not $q$, and Not $r$. They can be extended to minimal reduced models listed in the following table. Note that models numbered 6 to 10 are equal to models numbered 1 to 5, except that Not $p$ is true in them.

| No. | Not $p$ | Not $q$ | Not $r$ | $p$ | $q$ | $r$ |
|-----|---------|---------|---------|-----|-----|-----|
| 1   | 0       | 0       | 0       | 0   | 0   | 0   |
| 2   | 0       | 0       | 1       | 1   | 0   | 0   |
| 3   | 0       | 0       | 1       | 0   | 1   | 1   |
| 4   | 0       | 1       | 0       | 0   | 1   | 1   |
| 5   | 0       | 1       | 1       | 0   | 1   | 1   |
| 6   | 1       | 0       | 0       | 0   | 0   | 0   |
| 7   | 1       | 0       | 1       | 1   | 0   | 0   |
| 8   | 1       | 0       | 1       | 0   | 1   | 1   |
| 9   | 1       | 1       | 0       | 0   | 1   | 1   |
| 10  | 1       | 1       | 1       | 0   | 1   | 1   |

We conclude that there are only three possible valuations for the objective parts, i.e. $\Omega_P(\mathcal{N}O\mathcal{T})$ is:

\[
p \quad q \quad r
\begin{array}{ccc}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
\end{array}
\]

Now these minimal objective interpretations are the basis for the next round of default interpretations. Of course, from every minimal objective interpretation we immediately get a possible default interpretation if we translate the truth of $p$ to the falsity of Not $p$. But we can also combine minimal objective parts conjunctively and let Not $p$ be true only if $\neg p$ is true in all elements of some set of minimal models. Thus, we get the following four
valuations for the default atoms in $\Pi_P(\Omega_P(\mathcal{N}OT))$:

| $\text{Not } p$ | $\text{Not } q$ | $\text{Not } r$ |
|----------------|----------------|----------------|
| 1              | 1              | 1              |
| 0              | 1              | 1              |
| 1              | 0              | 0              |
| 0              | 0              | 0              |

This means that only the models numbered 1, 5, 6, 10 remain possible given the current knowledge about the defaults. Their objective parts are:

| $p$ | $q$ | $r$ |
|-----|-----|-----|
| 0   | 0   | 0   |
| 0   | 1   | 1   |

Finally, $p$ is false in all of these models, so $\text{Not } p$ must be assumed, and only the following two valuations of the default atoms are possible in the fixed point $\mathcal{N}^\omega$:

| $\text{Not } p$ | $\text{Not } q$ | $\text{Not } r$ |
|----------------|----------------|----------------|
| 1              | 1              | 1              |
| 1              | 0              | 0              |

The reduced models of the least static expansion $\mathcal{P}$ consist therefore of all reduced interpretations that include one of these two default parts and are models of $P$ itself.

Obviously, computing the reduct of a full interpretation is easy. However, it turns out that with the help of the following Kripke structure $\mathcal{K} = \langle W, R, V \rangle$, we can also perform the opposite operation, namely, extend the reduced interpretations to full interpretations, which are models of the static completion. We proceed as follows:

- The set of worlds $W$ is the set of those reduced interpretations $I = I_{\text{obj}} \cup I_{\text{not}}$ which are models of $P$ and satisfy $I_{\text{not}} \in \mathcal{N}^\omega$;
- $R(I, I')$, i.e. $I$ sees $I'$ iff $I'$ is a minimal model of $P$ and for all $p_1, \ldots, p_n \in \text{At}_{\mathcal{L}}$:
  $$I \models \text{Not } (p_1 \land \cdots \land p_n) \implies I' \models \neg p_1 \lor \cdots \lor \neg p_n.$$
- The valuation $V(I)$ of a reduced interpretation $I = I_{\text{obj}} \cup I_{\text{not}}$ is the objective part $I_{\text{obj}}$;
- In order to get a full interpretation from a reduced interpretation $I \in W$ we simply assign the formulas $\text{Not } (F)$ the truth values they have in the world $I$.

However, let us observe that not all full models of the least static expansion can be reconstructed in this way. Instead, we obtain only one representative from every equivalence class with the same reduct. For example, one can easily verify that the program $P := \{ p \leftarrow \text{Not } p \}$ has infinitely many different full models.
7 Implementation of the Model-Theoretic Characterization

We implemented a query-answering interpreter for the static semantics in the class of super programs based on the above model-theoretic characterization\(^7\). Let us explain in a little more detail how our prototype implementation works. First, because of the large number of possible default atoms, it seems impossible to construct explicitly the set \(\text{NOT} \) of all valuations — it would have \(2^2\) elements, where \(n\) is the number of objective atoms. However, it suffices to consider only those default atoms which occur explicitly in the program, other default atoms have no influence on the minimal models and we can always extend a valuation of these occurring default atoms consistently to a valuation of all default atoms.

Furthermore, it is possible to reduce the number of default atoms occurring in the program by evaluating the simple cases directly. For example, when it is possible to derive the disjunctive fact \(\text{VisitEurope} \lor \text{VisitAustralia}\) (in the example it is given explicitly), we can delete any rule like

\[
\text{Disappointed} \leftarrow \text{Not (VisitEurope)} \land \text{Not (VisitAustralia)},
\]

because we know that its body can never be true (this is a special case of the “negative reduction” transformation used in [BD97c]). In this example, deleting the rule also eliminates two default atoms.

Accordingly, the first step of our algorithm is to compute the residual program and to apply the reduction operators defined in [BD97c]. This transformation preserves the minimal models, and often significantly reduces the number of default atoms we have to consider (for instance, no default atoms remain if the input program is stratified and non-disjunctive).

Subsequently, we apply the definition of the operators \(\Omega_P\) and \(\Pi_P\) from Section 6 quite literally. However, two points are worth mentioning. In order to compute the minimal models for the \(\Omega_P\)-operator, we use the extension of Clark’s completion. Given an interpretation of the default atoms occurring in the residual program, we evaluate the rule bodies, so we get a set of positive disjunctions \(p_1 \lor \cdots \lor p_n\). We compute the completion by treating \(p_1 \lor \cdots \lor p_n\) like \(p_i \leftarrow \neg p_1 \land \cdots \land \neg p_{i-1} \land \neg p_{i+1} \land \cdots \land \neg p_n\). Then we use any model generation algorithm. Of course, other algorithms for generating minimal models can also be used, for instance the SATCHMO-like algorithm proposed in [BY96].

Next, it would be very inefficient to consider all subsets \(O' \subseteq O\), as required in the definition of the \(\Pi_P\)-operator. However, in order to check whether a given interpretation \(I_{not}\) of the default atoms is selected by \(\Theta_P\), only the maximal \(O'\) is of interest. We construct it as the set consisting of all \(I_{obj} \in O\) satisfying, for every default atom, the condition:

if \(I_{not} \models \text{Not (} p_1 \land \cdots \land p_n \text{)},\) then \(I_{obj} \models \neg (p_1 \land \cdots \land p_n)\).

Obviously, the inclusion of other \(I_{obj}\) into \(O'\) would immediately destroy the required property. Next, we check whether the so constructed \(O'\) is non-empty. Finally, we check that for every default atom \(\text{Not (} p_1 \land \cdots \land p_n \text{)}\) which is false in \(I_{not}\), there is an \(I_{obj} \in O'\) with \(I_{obj} \not\models \neg (p_1 \land \cdots \land p_n)\).

\(^7\)See ftp://ftp.informatik.uni-hannover.de/software/static/static.html from which the interpreter is available via FTP and WWW.
The program outputs the fixed point $N^\diamond$ of $\Theta_P$ and all minimal reduced models, whose default part is contained in $N^\diamond$. It then waits for queries. Queries to super programs are answered entirely on the basis of these minimal models.

8 Conclusion

In the first part of the paper we proved two powerful characterizations of the static semantics of finite non-monotonic knowledge bases (or belief theories). The main and fundamental result shows that static completions of such knowledge bases can in fact be obtained by means of a single iteration of the default closure operator. This result eliminates the need for multiple iterations in the computation of static completions and allows us to replace the fixed-point definition of static completions by the equivalent explicit definition. The second, closely related result, establishes a very interesting and somewhat intriguing relationship between static completions and Clark’s completions of non-monotonic knowledge bases (or belief theories). Since Clark’s completions are easily computable, it effectively reduces reasoning under the static semantics to theorem-proving in the underlying modal logic. By clarifying the notion of a static completion of finite knowledge bases, the two powerful characterizations provide the foundation for the remaining results obtained in the paper.

In the second part of the paper, we introduced the class of super-programs as a subclass of the class of all non-monotonic knowledge bases. We showed that this class of programs properly extends the classes of disjunctive logic programs, logic programs with strong (or “classical”) negation and arbitrary propositional theories. We demonstrated that the semantics of super programs constitutes an intuitively natural extension of the semantics of normal logic programs. When restricted to normal logic programs, it coincides with the well-founded semantics, and, more generally, it naturally corresponds to the class of all partial stable models of a normal program.

Subsequently, we established two additional characterizations of the static semantics of finite super-programs, one of which is syntactic and the other model-theoretic, which turned out to lead to procedural mechanisms allowing its computation. Due to the restricted nature of super programs, these important characterizations are significantly simpler than those applicable to arbitrary non-monotonic knowledge bases.

We used one of these characterizations as a basis for the implementation of a query-answering interpreter for super-programs which is available via FTP and WWW. We noted that while no such computational mechanism can be efficient, due to the inherent NP-completeness of the problem of computing answers to just positive disjunctive programs, they can become efficient when restricted to specific subclasses of programs and queries. Moreover, further research is likely to produce more efficient approximation methods.

The class of non-monotonic knowledge bases, and, in particular, the class of super-programs, constitutes a special case of a much more expressive non-monotonic formalism called the Autoepistemic Logic of Knowledge and Beliefs, $AELB$, introduced earlier in [Prz97]. $AELB$ isomorphically includes the well-known non-monotonic formalisms of Moore’s Autoepistemic Logic and McCarthy’s Circumscription. Via this embedding, the semantics of super programs is clearly linked to other well-established non-monotonic formalisms.

The proposed semantic framework for super programs is sufficiently flexible to allow various application-dependent extensions and modifications. We have already seen in Theorem
4.4 that by assuming an additional axiom we can produce the stable semantics instead of the well-founded semantics. By adding the distributive axiom for conjunction we can obtain a semantics that extends the disjunctive stationary semantics of logic programs introduced in [Prz95a]. Many other modifications and extensions are possible including variations of the notion of a minimal model resulting in inclusive, instead of exclusive, interpretation of disjunctions.

Even though most of the presented results, as well as the current implementation, are technically limited to finite propositional programs, they can be easily extended to range-restricted disjunctive Datalog programs, i.e., to programs in which every variable in the rule appears also in a positive body literal.

Acknowledgments
The authors would like to express their deep appreciation to Luis Moniz Pereira and Jose Alferes for their helpful comments.

References

[AP92] Jose Julio Alferes and Luiz Moniz Pereira. On Logic Program Semantics with Two Kinds of Negation. In K. R. Apt, editor, LOGIC PROGRAMMING: Proceedings of the 1992 Joint International Conference and Symposium, pages 574–589, Cambridge, Mass., November 1992. MIT Press.

[AP96] Jose Julio Alferes and Luiz Moniz Pereira, editors. Reasoning with Logic Programming, LNAI 1111, Berlin, 1996. Springer.

[APP97] J. Alferes, L. Pereira, and T. C. Przymusinski. ‘Classical’ negation in non-monotonic reasoning and logic programming. Journal of Automated Reasoning, 1997. In print. Extended abstract appeared in the Proceedings of the European Workshop on Logic in Artificial Intelligence (JELIA’96), Lecture Notes on Artificial Intelligence, Springer Verlag, 1996, vol. 1126, pp. 143-163.

[BD95] Stefan Brass and Jürgen Dix. A General Approach to Bottom-Up Computation of Disjunctive Semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, Nonmonotonic Extensions of Logic Programming, LNAI 927, pages 127–155. Springer, Berlin, 1995.

[BD97a] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. Journal of Logic Programming, 32(3):207–228, 1997. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation LPNMR, Proceedings of the Third International Conference, Kentucky, pages 85–98, 1995. LNCS 928, Springer.).

[BD97b] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. Journal of Automated Reasoning, to appear, 1997. (Extended abstract appeared in: Characterizing D-
WFS: Confluence and Iterated GCWA. *Logics in Artificial Intelligence, JELIA ’96*, pages 268–283, 1996. Springer, LNCS 1126.

[BD97c] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, accepted for publication, 1997. (Extended abstract appeared in: Disjunctive Semantics Based upon Partial and Bottom-Up Evaluation, *Proceedings of the 12-th International Logic Programming Conference, Tokyo*, pages 199–213, 1995. MIT Press.)

[BL93] Stefan Brass and Udo W. Lipeck. Bottom-up query evaluation with partially ordered defaults. In Stefano Ceri, Katsumi Tanaka, and Shalom Tsur, editors, *Deductive and Object-Oriented Databases, Third Int. Conf., (DOOD’93)*, number 760 in LNCS, pages 253–266, Berlin, 1993. Springer.

[Bry89] François Bry. Logic programming as constructivism: A formalization and its application to databases. In *Proc. of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’89)*, pages 34–50, 1989.

[Bry90] François Bry. Negation in logic programming: A formalization in constructive logic. In Dimitris Karagiannis, editor, *Information Systems and Artificial Intelligence: Integration Aspects*, pages 30–46, Berlin, 1990. Springer.

[BY96] F. Bry and A. Yahya. Minimal model generation with positive unit hyper-resolution tableaux. In *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, pages 143–159, Terrasini, Italy, May 1996. Springer-Verlag.

[Cla78] Keith L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data-Bases*, pages 293–322. Plenum, New York, 1978.

[DHS96] Roy Dyckhoff, Heinrich Herre, and Peter Schroeder-Heister, editors. *Extensions of Logic Programming*, LNAI 1050, Berlin, 1996. Springer.

[Dix95] Jürgen Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In Andre Fuhrmann and Hans Rott, editors, *Logic, Action and Information – Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327. DeGruyter, 1995.

[DK89a] P. M. Dung and K. Kanchansut. A fixpoint approach to declarative semantics of logic programs. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of North American Conference Cleveland,Ohio, USA*. MIT Press, October 1989.

[DK89b] P. M. Dung and K. Kanchansut. A natural semantics for logic programs with negation. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of the 9th Conference on Foundations of Software Technology and Theoretical Computer Science*, Berlin, 1989. Springer.

[DPP95] J. Dix, L. Pereira, and T. Przymusinski, editors. *Non-Monotonic Extensions of Logic Programming*, LNAI 927, Berlin, 1995. Springer.
[DPP97] J. Dix, L. Pereira, and T. Przymusinski, editors. Non-Monotonic Extensions of Logic Programming, LNAI 1216, Berlin, 1997. Springer.

[EG93] Thomas Eiter and Georg Gottlob. Complexity aspects of various semantics for disjunctive databases. In Proc. of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’93), pages 158–167, 1993.

[EG96] Thomas Eiter and Georg Gottlob. Mächtigkeit von Logikprogrammierung über Datenbanken. KI, 3:32–39, 1996.

[EGM94] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Adding disjunction to datalog. In Proc. of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’94), pages 267–278, 1994.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In Proceedings of the Seventh International Logic Programming Conference, Jerusalem, Israel, pages 579–597, Cambridge, Mass., 1990. Association for Logic Programming, MIT Press.

[GPP89] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. On the Relationship between Circumscription and Negation as Failure. Artificial Intelligence, 38:75–94, 1989.

[Lip79] W. Lipski, Jr. On semantic issues connected with incomplete information databases. ACM Transactions on Database Systems, 4:262–296, 1979.

[LMR92] Jorge Lobo, Jack Minker, and Arcot Rajasekar. Foundations of Disjunctive Logic Programming. MIT-Press, 1992.

[McC80] John McCarthy. Circumscription: A Form of Nonmonotonic Reasoning. Artificial Intelligence, 13:27–39, 1980.

[Min82] Jack Minker. On indefinite databases and the closed world assumption. In Proceedings of the 6th Conference on Automated Deduction, New York, pages 292–308, Berlin, 1982. Springer.

[Min93] Jack Minker. An Overview of Nonmonotonic Reasoning and Logic Programming. Journal of Logic Programming, Special Issue, 17, 1993.

[MT93] Wiktor Marek and Mirek Truszczyński. Nonmonotonic Logics; Context-Dependent Reasoning. Springer, Berlin, 1st edition, 1993.

[Prz95a] T. C. Przymusinski. Semantics of normal and disjunctive logic programs: A unifying framework. In J. Dix, L. Pereira, and T. Przymusinski, editors, Proceedings of the Workshop on Non-Monotonic Extensions of Logic Programming at the Eleventh International Logic Programming Conference, ICLP’95, Santa Margherita Ligure, Italy, June 1994, pages 43–67. Springer Verlag, 1995.

[Prz95b] T. C. Przymusinski. Static semantics for normal and disjunctive logic programs. Annals of Mathematics and Artificial Intelligence, Special Issue on Disjunctive Programs(14):323–357, 1995.
[Prz97] T. C. Przymusinski. Autoepistemic logic of knowledge and beliefs. *Journal of Artificial Intelligence*, 1997. In print. Extended abstract appeared in ‘A knowledge representation framework based on autoepistemic logic of minimal beliefs’ In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94, Seattle, Washington, August 1994*, pages 952-959, Los Altos, CA, 1994. American Association for Artificial Intelligence, Morgan Kaufmann.

[SS94] Chiaki Sakama and Hirohisa Seki. Partial Deduction of Disjunctive Logic Programs: A Declarative Approach. In *Logic Program Synthesis and Transformation – Meta Programming in Logic*, LNCS 883, pages 170–182, Berlin, 1994. Springer. Extended version to appear in *Journal of Logic Programming*.

[YY93] Jia-Huai You and Li-Yan Yuan. Autoepistemic Circumscription and Logic Programming. *Journal of Automated Reasoning*, 10:143–160, 1993.
A Equivalence of Knowledge Bases and Belief Theories

The notion of a non-monotonic knowledge base defined in this paper is equivalent to the notion of a belief theory in the Autoepistemic Logic of Beliefs, AEB, originally defined by Przymusinski in [Prz97]. Belief theories were defined using a propositional language \( \mathcal{L} \) augmented with a belief operator \( B \) instead of the default negation operator \( Not \). In order to translate a non-monotonic knowledge base into a belief theory all one needs to do is to replace everywhere \( Not F \) by \( B\neg F \). Conversely, in order to translate a belief theory into a non-monotonic knowledge base all one needs to do is to replace everywhere \( BF \) by \( Not \neg F \).

As a result, \( Not F \) can be simply viewed as a shorthand for \( B\neg F \) and, vice versa, \( BF \) can be viewed as a shorthand for \( Not \neg F \). However, in the definition of non-monotonic knowledge bases, in addition to using the above substitution, we also used different axioms and derivation rules from those used in the Logic AEB. Consequently, it is not entirely obvious that the two logics are equivalent. Below, we recall the definition of belief theories in AEB and formally establish the equivalence of the two logics.

Consider a fixed propositional language \( \mathcal{L} \) with standard connectives (\( \lor, \land, \rightarrow, \leftarrow, \neg \)) and the propositional letters \( true \) and \( false \). Extend the language \( \mathcal{L} \) to a propositional modal language \( \mathcal{L}_{AEB} \) by augmenting it with a modal operator \( B \), called the belief operator. The atomic formulae of the form \( BF \), where \( F \) is an arbitrary formula of \( \mathcal{L}_{AEB} \), are called belief atoms. The formulae of the original language \( \mathcal{L} \) are called objective. Any propositional theory in the modal language \( \mathcal{L}_{AEB} \) is called a belief theory.

Assume the following two simple axiom schemata and the necessitation inference rule describing the arguably obvious properties of belief atoms:

\[ (D) \text{ Consistency Axiom: } \neg Bfalse \]  
\[ (K) \text{ Normality Axiom: } \text{For any formulae } F \text{ and } G: \]  
\[ B(F \rightarrow G) \rightarrow (BF \rightarrow BG) \]  
\[ (N) \text{ Necessitation Inference Rule: } \text{For any formulae } F: \]  
\[ \frac{F}{BF}. \]

The first axiom states that falsity is *not* believed. The second axiom states that if we believe that a formula \( F \) implies a formula \( G \) and if we believe that \( F \) is true then we believe that \( G \) is true as well. The necessitation inference rule states that if a formula \( F \) has been proven to be true then \( F \) is also believed to be true.

**Definition A.1 (Formulae Derivable from a Belief Theory)**

*For any belief theory \( T \), we denote by \( Cn_{AEB}(T) \) the smallest set of formulae of the language \( \mathcal{L}_{AEB} \) which contains the theory \( T \), all (substitution instances of) the axioms (D) and (K) and is closed under both standard propositional consequence and the necessitation rule (N).*
We say that a formula $F$ is derivable from theory $T$ in the logic $AEB$ if $F$ belongs to $Cn_{AEB}(T)$. We denote this fact by $T \vdash_{AEB} F$. A belief theory $T$ is consistent if the theory $Cn_{AEB}(T)$ is consistent.

In the presence of the axiom (K), the axiom (D) is equivalent to the axiom $BF \rightarrow \mathcal{B}\neg F$, stating that if we believe in a formula $F$ then we do not believe in $\neg F$. For readers familiar with modal logics it should be clear by now that we are, in effect, considering here a normal modal logic with one modality $\mathcal{B}$ which satisfies the consistency ("no dead ends") axiom (D) [MT93].

In order to translate a non-monotonic knowledge base into a belief theory all we need to do is to replace everywhere $\text{Not } F$ by $\mathcal{B}\neg F$ thus giving the default negation $\text{Not } F$ the intended meaning of "$\neg F$ is believed":

$$\text{Not } F \equiv \mathcal{B}\neg F.$$

Clearly, upon such a translation, the axioms (CA) and (DA) and the rule (IR) take the form:

(CA’) Consistency Axiom:

$$B\text{true and } \neg B\text{false} \quad (8)$$

(DA’) Distributive Axiom:

For any formulae $F$ and $G$:

$$\mathcal{B}(F \land G) \leftrightarrow BF \land BG \quad (9)$$

(IR’) Invariance Inference Rule:

For any formulae $F$ and $G$:

$$F \leftrightarrow G \quad \frac{BF \leftrightarrow BG}. \quad (10)$$

The consistency axiom (CA’) states that true is believed while false is not believed. The second axiom (DA’) states that beliefs $\mathcal{B}$ are distributive with respect to conjunctions. The invariance inference rule (IR’) states that if two formulae are known to be equivalent then so are beliefs in these formulae. In other words, the meaning of $BF$ does not depend on the specific form of the formula $F$.

In order to show that the notion of a non-monotonic knowledge base is equivalent to the notion of a belief theory it suffices to show that the axioms (D) and (K) and the necessitation rule (IR) are equivalent to the axioms (CA’) and (DA’) and the inference rule (IR’).

Proposition A.2 (Equivalence of knowledge bases and belief theories)

The axioms (D) and (K) and the necessitation rule (N) are equivalent to the axioms (CA’) and (DA’) and the inference rule (IR’).

More precisely, a theory in the language $L_{AEB}$ contains all (substitution instances of) the axioms (D) and (K) and is closed under both standard propositional consequence and the necessitation rule (N) if and only if it contains all (substitution instances of) the axioms (CA’) and (DA’) and is closed under both standard propositional consequence and the inference rule (IR’).
Proof. “⇒” Suppose that a theory $T$ contains all (substitution instances of) the axioms (D) and (K) and is closed under both standard propositional consequence and the necessitation rule (N). Since $true$ belongs to $T$, from the necessitation rule (N) it follows that so does $Btrue$ and thus the axiom $(CA')$ holds.

To show that the axiom $(DA')$ holds we first show that for any formulae $F$ and $G$:

$$B(F \land G) \rightarrow BF \land BG$$

belongs to $T$. Clearly, $(F \land G) \rightarrow F \in T$ and therefore, by the Necessitation Rule (N), $B((F \land G) \rightarrow F) \in T$. From the Normality Axiom (K) we infer that $B(F \land G) \rightarrow BF \in T$. Similarly, $B(F \land G) \rightarrow BG \in T$. It follows that $B(F \land G) \rightarrow BF \land BG \in T$.

To show that the axiom $(DA')$ holds it now suffices to prove that for any formulae $F$ and $G$:

$$BF \land BG \rightarrow B(F \land G)$$

belongs to $T$. Clearly, $F \rightarrow (G \rightarrow F \land G) \in T$ and therefore, by the Necessitation Rule (N), $B(F \rightarrow (G \rightarrow F \land G)) \in T$. From the Normality Axiom (K) we infer that $BF \rightarrow B(G \rightarrow F \land G) \in T$. Applying the Normality Axiom (K) again we conclude that $BF \rightarrow (BG \rightarrow B(F \land G)) \in T$. This shows that $BF \land BG \rightarrow B(F \land G) \in T$.

To demonstrate that the inference rule $(IR')$ holds suppose that $F \leftrightarrow G$ belongs to $T$. By necessitation, also $B(F \leftrightarrow G)$ belongs to $T$. From the Normality axiom it follows immediately that so does $BF \leftrightarrow BG$.

“⇐” Suppose that a theory $T$ contains all (substitution instances of) the axioms $(CA')$ and $(DA')$ and is closed under both standard propositional consequence and the inference rule $(IR')$. Clearly, the axiom (C) also holds. Since $(F \rightarrow G) \land F \rightarrow F \land G$ belongs to $T$, from the rule $(IR')$ it follows that $B((F \rightarrow G) \land F) \leftrightarrow B(F \land G)$ belongs to $T$. From the axiom $(DA')$ we conclude that also $B(F \rightarrow G) \land B(F) \leftrightarrow BF \land BG$ belongs to $T$. This proves that $B((F \rightarrow G)) \land B(F) \rightarrow B(G)$ belongs to $T$ and thus the Normality axiom (N) holds.

It remains to show that $T$ is closed under necessitation. If $F$ belongs to $T$ then so does $F \leftrightarrow true$. By the rule $(IR')$, the formula $BF \leftrightarrow Btrue$ belongs to $T$ which implies that $BF$ also belongs to $T$.

Conversely, in order to translate a belief theory into a non-monotonic knowledge base all we need to do is to replace everywhere $BF$ by $Not \neg F$ thus giving the belief atoms $BF$ the meaning of “formula $F$ can be assumed true by default”.

### B Proofs of Theorems from Section 3

#### B.1 Proof of Proposition 3.3

“⇒” Suppose $I$ is not a model of $comp(T)$, so there is a completion axiom $comp(A,T)$ of an atom $A$ in $T$:

$$A \leftrightarrow \bigvee_{1 \leq m \leq s} (Not\,G_{1,m} \land \ldots \land Not\,G_{k,m} \land \neg A_{1,m} \land \ldots \land \neg A_{l,m} \land \neg Not\,F_{1,m} \land \ldots \land \neg Not\,F_{n,m})$$



31
which is not satisfied in \( \mathcal{I} \). If the direction \( \leftarrow \) is not satisfied then \( \mathcal{I} \) obviously is not even a model of \( T \). So assume that this direction is satisfied for every completion axiom, i.e. \( \mathcal{I} \) is a model of \( T \). If the direction \( \rightarrow \) is not satisfied then \( \mathcal{I} \models A \) (consequently, \( A \neq \text{false} \)) but for every rule

\[
\neg G_{1,m} \land \ldots \land \neg G_{k,m} \rightarrow A \lor A_{1,m} \lor \ldots \lor A_{l,m} \lor \neg F_{1,m} \lor \ldots \lor \neg F_{m,m},
\]

with \( A \) in the head, one of the \( A_{i,j} \) is true or one of the \( \neg F_{i,j} \) is true or one of the \( \neg G_{i,j} \) is false. But then the interpretation \( \mathcal{I}' \), which differs from \( \mathcal{I} \) only in \( \mathcal{I}' \not\models A \), is also a model of \( T \) and therefore \( \mathcal{I} \) is not minimal.

\( \leftarrow \leftarrow \) Suppose conversely that \( \mathcal{I} \) is not a minimal model of \( T \). If \( \mathcal{I} \) is not even a model, i.e. does not satisfy some rule

\[
\neg G_{1,m} \land \ldots \land \neg G_{k,m} \rightarrow A \lor A_{1,m} \lor \ldots \lor A_{l,m} \lor \neg F_{1,m} \lor \ldots \lor \neg F_{m,m},
\]

then obviously the direction \( \leftarrow \) of the completion axioms for every \( A_i \) is also violated. So let us assume that \( \mathcal{I} \) is a model, but not minimal, so there is a smaller model \( \mathcal{I}' \). We can choose \( \mathcal{I}' \) in such a way that it is minimal. Let \( A \) be an objective atom with \( \mathcal{I} \models A \) and \( \mathcal{I}' \not\models A \) (consequently, \( A \neq \text{false} \)). Let us consider the completion axiom \( \text{comp}(A,T) \) for \( A \):

\[
A \leftarrow \bigvee_{1 \leq m \leq s} (\neg G_{1,m} \land \ldots \land \neg G_{k,m} \land \neg A_{1,m} \land \ldots \land \neg A_{l,m} \land \neg F_{1,m} \land \ldots \land \neg F_{m,m}).
\]

By the contraposition of the already proven opposite direction, the minimal model \( \mathcal{I}' \) is a model of \( \text{comp}(T) \), so it satisfies the completion axiom. Since \( \mathcal{I}' \not\models A \), for every \( i \) there is a \( j \) such that \( \mathcal{I}' \models A_{i,j} \) or \( \mathcal{I}' \models \neg F_{i,j} \) or \( \mathcal{I}' \not\models G_{i,j} \). But since \( \mathcal{I}' \) is smaller than \( \mathcal{I} \), \( \mathcal{I}' \models A_{i,j} \iff \mathcal{I} \models A_{i,j} \) and \( \mathcal{I}' \models \neg F_{i,j} \iff \mathcal{I} \models \neg F_{i,j} \) and \( \mathcal{I}' \not\models G_{i,j} \iff \mathcal{I} \not\models G_{i,j} \). Thus, the right hand side is also false in \( \mathcal{I} \), but \( A \) is true in \( \mathcal{I} \), and therefore we conclude that \( \mathcal{I} \not\models \text{comp}(T) \).

\( \square \)

### B.2 Proof of Proposition 3.4

We are supposed to prove that every finite knowledge base \( T \) can be transformed into a finite residual knowledge base \( T_{\text{res}} \), called the \textit{residuum} of \( T \), so that an interpretation \( M \) is a minimal model of \( T \) if and only if \( M \) is a minimal model of \( T_{\text{res}} \).

We will show that every knowledge base \( T \) can be transformed into a residual knowledge base \( T_{\text{res}} \) by applying just two database transformations, namely, “elimination of tautologies” and “unfolding”. For a detailed definition of these transformations the reader is referred to [BD95, BD97a, BD97c, BD97b]. Here we describe them very briefly.

The “elimination of tautologies” allows us to remove an atom if it appears both in the head and in the body of a given clause. The transformation of “unfolding” - also called GPPE – is defined as follows. Suppose that \( B \) is an objective atom and let \( B \lor A_i \leftarrow R_i, \ i = 1, \ldots, n \), be all rules about \( B \), i.e., rules containing \( B \) in their head. The application of GPPE to the rule \( A \leftarrow B \land R \) that contains \( B \) in its body, results in a new knowledge base obtained by deleting the rule \( A \leftarrow B \land R \) and adding to the knowledge base the new rules \( A \lor A_i \leftarrow R \land R_i \).

32
It is easy to see that each application of GPPE, perhaps combined with tautology elimination, removes a given objective atom \( B \) from the body of one clause without introducing any new clauses containing \( B \) in their bodies. This means that a finite number of such transformations will lead to a knowledge base that does not contain \( B \)'s in bodies of its clauses. Applying this procedure to all objective atoms appearing in a (finite) knowledge base will therefore produce a residual knowledge base.

Accordingly, in order to prove Proposition 3.4, it suffices to establish the following lemma.

**Lemma B.1** Let \( T_1 \) and \( T_2 \) be two knowledge bases such that \( T_2 \) results from \( T_1 \) by an application of GPPE or deletion of tautologies. Then \( T_1 \) and \( T_2 \) have the same minimal models.

The deletion of tautologies is an equivalence transformation, so it does not change the set of models, and thus also not the set of minimal models. Now suppose that we apply GPPE to a rule \( A \leftarrow B \land R \), where \( B \) is an objective atom, and suppose that \( B \lor A_i \leftarrow R_i \), \( i = 1, \ldots, n \), are all the rules about \( B \), i.e., rules containing \( B \) in their heads.

- Since the new rules \( A \lor A_i \leftarrow R \land R_i \), resulting from the application of GPPE, are logical consequences of the rules in \( T_1 \), every model of \( T_1 \) is also a model of \( T_2 \).

- Let \( I \) be a minimal model of \( T_2 \). We first show that it is a model of \( T_1 \) (not necessarily minimal). Suppose this is not the case, so the rule \( A \leftarrow B \land R \) is violated in \( I \) (this is the only rule in \( T_1 \setminus T_2 \)). This means that \( I \models B \), \( I \models R \), and \( I \not\models A \). Since \( I \) is a minimal model of \( T_2 \), the interpretation \( I' \) with \( I' \not\models B \) (and otherwise equal to \( I \)) cannot be a model of \( T_2 \). Since objective atoms do not occur negated in rule bodies, this means that there is a rule with \( B \) in the head, i.e. one of the rules \( B \lor A_i \leftarrow R_i \), such that \( I' \models R_i \) and \( I' \not\models A_i \). Since \( I' \models R_i \), the atom \( B \) cannot be contained in \( R_i \), and since we exclude multiple occurrences in the head, it is also not contained in \( A_i \). Thus, \( I \models R_i \) and \( I \not\models A_i \). But now we infer that the new rule \( A \lor A_i \leftarrow R \land R_i \) contained in \( T_2 \) is violated by \( I \), which is a contradiction.

We now have to show that the sets of minimal models are equal:

- Suppose that \( I \) is a minimal model of \( T_1 \). By the first part of this proof, we know that it is a model of \( T_2 \). If it were not minimal, there would have to be a smaller minimal model \( I' \) of \( T_2 \). But by the second part of this proof it would also have to be a model of \( T_1 \), contradicting the assumed minimality of \( I \) as a model of \( T_1 \).

- Suppose that \( I \) is a minimal model of \( T_2 \). Then it is a model of \( T_1 \). If there were a smaller model \( I' \) of \( T_1 \), it would also have to be a model of \( T_2 \), again leading to a contradiction. \( \Box \)

\(^3\)It is not a new rule \( A \lor A_i \leftarrow R \land R_i \), because otherwise \( A \) would contain \( B \), and the unfolded rule \( A \leftarrow B \land R \) would be a tautology and could never be violated.
B.3 Proof of Lemma 3.6

First we prove the following lemma which characterizes minimal models of special knowledge bases.

**Lemma B.2** Let $T$ be any knowledge base and $T_{Not}$ be a set of formulae which contain only default atoms. Minimal models of $T \cup T_{Not}$ are precisely those minimal models of $T$ which satisfy $T_{Not}$.

**Proof.**

- Let $\mathcal{I}$ be a minimal model of $T \cup T_{Not}$. Of course, $\mathcal{I}$ is a model of $T$ and of $T_{Not}$. Now suppose that there is a smaller model $\mathcal{I}'$ of $T$. Since $\mathcal{I}$ and $\mathcal{I}'$ do not differ in the interpretation of default atoms, $\mathcal{I}'$ is also a model of $T_{Not}$, and thus a model of $T \cup T_{Not}$. But this contradicts the assumed minimality of $\mathcal{I}$.

- Let $I$ be a minimal model of $T$ which also satisfies $T_{Not}$. Clearly, $I$ is a model of $T \cup T_{Not}$. Since $\mathcal{I}'$ is also a model of $T$, the existence of a smaller model $\mathcal{I}'$ would contradict the minimality of $I$.

Now we continue the proof of Lemma 3.6.

**Lemma B.3** Let $T$ be an arbitrary finite knowledge base and let

$$\hat{T} = Cn_{Not}(T \cup Not \neg \text{comp}(T_{res})).$$

If a formula $F$ is false in all minimal models of $\hat{T}$ then $Not F$ is already contained in $\hat{T}$. In other words, $\Psi_T(\hat{T}) \subseteq \hat{T}$.

Let $T_{Not}$ be the subset of $\hat{T}$ which consists entirely of default negation atoms, i.e. $Not \left(\neg \text{comp}(T_{res})\right)$ together with the axioms (CA), all instances of the axiom (DA) and all formulae forced in by the derivation rule (IR). Then the models of $\hat{T}$ coincide with the models of $T \cup T_{Not}$ because the required closure under logical consequences does not change the set of models.

By Lemma B.2, minimal models of $T \cup T_{Not}$ are precisely those minimal models of $T$ which satisfy $T_{Not}$. Suppose that $\neg F$ holds in all minimal models of $\hat{T}$. By Proposition 3.3, the minimal models of $T$ coincide with the models of $\text{comp}(T_{res})$ and therefore $\neg F$ holds in all models of $\text{comp}(T_{res}) \cup T_{Not}$, i.e. it is a propositional consequence of $\text{comp}(T_{res}) \cup T_{Not}$. By the compactness of propositional logic, there is a finite subset $\{F_1, \ldots, F_n\} \subseteq \text{comp}(T_{res}) \cup T_{Not}$ such that $\{F_1, \ldots, F_n\} \models \neg F$. But then $F_1 \land \cdots \land F_n \rightarrow \neg F$ is a tautology and therefore, by Proposition 2.3, $Not (\neg F_1) \land \cdots \land Not (\neg F_n) \rightarrow Not (F)$ is contained in $\hat{T}$.

If $F_i$ is from $\text{comp}(T_{res})$, then $Not (\neg F_i)$ is obviously contained in $\hat{T}$ and otherwise we get $Not (\neg F_i)$ by necessitation (see Proposition 2.3). Thus, by the closure under classical consequences, we conclude $Not (F) \in \hat{T}$. \qed
C  Kripke Models of Static Expansions

Before we prove the model-theoretic characterization (Theorem 6.3), we prove here a theorem which easily allows us to construct models of the least static expansion from Kripke structures. It is used as a lemma in the proof to Theorem 6.3, but it is of its own interest. Although we later need only super programs and reduced models, we allow in this section arbitrary belief theories and consider full models.

In order to be precise and self-contained, let us briefly repeat the definition of Kripke structures [MT93]. Since our axioms entail the normality axiom, it suffices to consider normal Kripke structures:

Definition C.1 (Kripke Structure)
A (normal) Kripke structure is a triple $K = (W, R, V)$ consisting of

- a nonempty set $W$, the elements $w \in W$ are called worlds,
- a relation $R \subseteq W \times W$, the “visibility relation” (if $R(w, w')$ we say that world $w$ sees world $w'$), and
- a mapping $V : W \to OBJ$, which assigns to every world $w$ a valuation $I_{obj} = V(w)$ of the objective atoms $At_L$.

Definition C.2 (Truth of Formulas in Worlds)
The validity of a formula $F$ in a world $w$ given Kripke structure $K = (W, R, V)$ is defined by

- If $F$ is an objective atom (proposition) $p \in At_L$, then $(K, w) \models F :\iff V(w) \models p$.
- If $F$ is a negation $\neg G$, then $(K, w) \models F :\iff (K, w) \not\models G$.
- If $F$ is a disjunction $G_1 \lor G_2$, then $(K, w) \models F :\iff (K, w) \models G_1$ or $(K, w) \models G_2$ (and further propositional connectives as usual).
- If $F$ is a default negation atom $\text{Not}(G)$, then
  $$(K, w) \models F :\iff \text{for all } w' \in W \text{ with } R(w, w') : (K, w') \not\models G.$$  

Now given such a Kripke structure $K$, we get from every world $w$ a propositional interpretation $\mathcal{I} = \mathcal{K}(w)$ of $\mathcal{L}_{Not}$, i.e. a valuation of $At_L \cup \{\text{Not}(F) : F \in \mathcal{L}_{Not}\}$: We simply make an objective or belief atom $A$ true in $\mathcal{I}$ iff $(K, w) \models A$. Since the propositional connectives are defined in a Kripke structure like in standard propositional logic, we obviously have $\mathcal{I} \models F \iff (K, w) \models F$ for all $F \in \mathcal{L}_{Not}$.

Theorem C.3 (Kripke Structures Yield Static Expansions)
Let $T$ be an arbitrary belief theory and $K = (W, R, V)$ be a Kripke structure satisfying

- For every $w \in W$, there is a $w' \in W$ with $R(w, w')$ (consistency).
- For every $w \in W$, $(K, w) \models T$, i.e. the interpretation $\mathcal{I} = \mathcal{K}(w)$ is a model of $T$.  

• For every $w, w' \in W$ with $R(w, w')$, the interpretation $I = \mathcal{K}(w')$ is a minimal model of $T$ (“only minimal models are seen”).

Then $T^\circ := \{ F \in \mathcal{L}_{\text{Not}} : \text{for every } w \in W : w \models F \}$ is a static expansion of $T$.

Proof.

• Consistency Axioms: Let any $w \in W$ be given. The first part $(K, w) \models \text{Not (false)}$ is trivial, because for any $w' \in W$ we have $w' \not\models \text{false}$. Second, we have to show $(K, w) \models \neg \text{Not (true)}$. By the first requirement of the theorem, there is $w' \in W$ with $R(w, w')$. Now $(K, w') \models \text{true}$, therefore $(K, w) \not\models \text{Not (true)}$, i.e. $(K, w) \models \neg \text{Not (true)}$.

• Distributive Axiom: We have to show that for all formulas $F, G \in \mathcal{L}_{\text{Not}}$ and all $w \in W$ that the distributive axiom holds in $w$: $(K, w) \models \text{Not (} F \lor G \text{)} \iff \text{Not (} F \text{)} \land \text{Not (} G \text{)}$. This follows simply from applying the definitions: $(K, w) \models \text{Not (} F \lor G \text{)} \iff (K, w') \not\models F \lor G \iff$ for all $w' \in W$ with $R(w, w')$: $(K, w') \not\models F$ and $(K, w') \not\models G \iff (K, w) \models \text{Not (} F \text{)}$ and $(K, w) \models \text{Not (} G \text{)}$. The distributive axiom holds in $w$, therefore $(K, w) \models \text{Not (} F \lor G \text{)}$.

• Invariance Inference Rule: Let $F \iff G \in T^\circ$, i.e. for every $w \in W$ we have $(K, w) \models F \iff (K, w) \models G$. But then also $(K, w) \models \text{Not (} F \iff G \text{)}$ holds for every $w \in W$ since $(K, w') \not\models F \iff (K, w') \not\models G$ holds for every $w' \in W$ with $R(w, w')$.

• Closure under propositional consequences: Let $F \in \mathcal{L}_{\text{Not}}$ a formula which is a propositional consequence of $F_1, \ldots, F_n \in T^\circ$. Now $F_i \in T^\circ$ means that $(K, w) \models F_i$ for every $w \in W$. But the formulas valid in one world are closed under propositional consequences, since the meaning of the propositional connectives is defined as in the standard case. So we get $(K, w) \models F$, and thus $F \in T^\circ$.

• Now suppose that $T^\circ \models_{\text{min}} \neg F$, i.e. $F$ is false in all minimal models of $T^\circ$. We have to show that $\text{Not (} F \text{)} \in T^\circ$, i.e. $(K, w) \models \text{Not (} F \text{)}$ for every $w \in W$. So we have to show $(K, w') \not\models F$ for every $w' \in W$ with $R(w, w')$.

Let such a $w'$ be given, and let $I = \mathcal{K}(w')$. By the last condition of the theorem, we know that $I$ is a minimal model of $T$. By construction, it is also a model of $T^\circ$, and since $T \subseteq T^\circ$, there can be no smaller model. Thus, $I$ is a minimal model of $T^\circ$ and therefore satisfies $\neg F$, i.e. $(K, w') \not\models F$. □

This theorem gives us a simple way to construct models of the least static expansion $\overline{T}$: Obviously, for every $w \in W$, the interpretation $I = \mathcal{K}(w)$ is a model of $T^\circ$. But the least static expansion is a subset of every other static expansion, i.e. $\overline{T} \subseteq T^\circ$, and therefore we have $I \models \overline{T}$.

**Corollary C.4** If a Kripke structure $\mathcal{K} = (W, R, V)$ satisfies the conditions of Theorem C.3, then for every $w \in W$, the interpretation $I = \mathcal{K}(w)$ is a model of the least static expansion $\overline{T}$.  

36
Example C.5 Let us consider the knowledge base of Example 2.13:

\[
\text{Not Broken} \rightarrow \text{Runs} \\
\text{Not Fixed} \rightarrow \text{Broken}
\]

Here, we can construct a Kripke model \( K \) with only one world \( w \) with the valuation \( I_{\text{obj}} = \{ \neg \text{Fixed}, \text{Broken}, \neg \text{Runs} \} \). We let this world “see” itself, i.e. \( R := \{(w, w)\} \). Then \( (K, w) \models \text{Not Fixed}, (K, w) \not\models \text{Not Broken}, \) and \( (K, w) \models \text{Not Runs} \).

It is, however, also possible to add a world \( w' \) with a non-minimal valuation \( I'_{\text{obj}} = \{ \text{Fixed}, \text{Broken}, \text{Runs} \} \). Of course, both worlds can only see \( w \), i.e. \( R := \{(w, w), (w', w)\} \), because all seen worlds must be minimal models. This example shows that the static semantics does not imply \( \neg \text{Fixed} \), but it of course implies \( \neg \text{Fixed} \). So the nonmonotonic negation is cleanly separated from the classical negation.

Example C.6 Let us consider the theory \( P := \{ p \leftarrow \neg \text{Not} (p) \} \) corresponding to a well-known logic program. We claimed in Section 6 that the least static expansion \( \overline{P} \) of this program has infinitely many different models. But let us first look a Kripke structure which generates only two models:

\[
\begin{array}{c}
1 \\
\neg p \\
2 \\
p
\end{array}
\]

I.e. the set of worlds \( W \) is \{1, 2\}, the visibility relation \( R \) is \{ (1, 2), (2, 1) \} (each world can see the other one, but not itself), and \( p \) is false in world 1 and true in world 2. In world 1, \( \neg \text{Not} (p) \) is false, and in world 2, it is true.

But there are also quite different Kripke models. The construction in Section 6 will yield:

\[
\begin{array}{c}
1 \\
\neg p \\
2 \\
p \\
3 \\
p
\end{array}
\]

Note that in world 3, the default atom \( \neg \text{Not} (p) \) is false, so the interpretation of this world is non-minimal (there is no need to make \( p \) true). Thus, there can be no incoming edges.

Let us now finally present the Kripke structure which yields infinitely many models:

\[
\begin{array}{c}
1 \\
\neg p \\
2 \\
p \\
\neg p \\
3 \\
p \\
\neg p \\
4 \\
p \\
\neg p \\
5 \\
p \\
\cdots
\end{array}
\]
Now we of course have to explain that different worlds really yield different interpretations. The trick is that world 1 is the only world in which neither \( \text{Not} (p) \) nor \( \text{Not} (\neg p) \) are true. Now other worlds can be identified with the number of nested beliefs necessary to get to world 1. So the formula

\[
B^{n-1}(\neg \text{Not} (p) \land \neg \text{Not} (\neg p)) \land B^{n-2}(\text{Not} (p) \lor \text{Not} (\neg p))
\]

is true in world \( n \geq 2 \), but no other world. \( \square \)

D Proof of Modeltheoretic Characterization (Theorem 6.3)

We will first prove that a reduced model \( I = I_{\text{obj}} \cup I_{\text{not}} \) with \( I \models P \) and \( I_{\text{not}} \in \mathcal{N}^\circ \) can be extended to a full model \( \mathcal{I} \) of the least static expansion \( \mathcal{P} \). Of course, this proof is based on the Kripke structure already mentioned in Section 6:

**Definition D.1 (Standard Kripke Model)**

Let \( P \subseteq \mathcal{L}_{\mathcal{N}_{\text{not}}} \) be a super program. We call the Kripke structure \( \mathcal{K} = (W, R, V) \) defined as follows the “standard Kripke model” of \( P \). Let \( \mathcal{N}^\circ \) be the greatest fixpoint of \( \Theta_P \). Then:

- The set of worlds \( W \) are the reduced interpretations \( I = I_{\text{obj}} \cup I_{\text{not}} \) we are interested in, i.e. satisfying \( I \models P \) and \( I_{\text{not}} \in \mathcal{N}^\circ \).
- \( R(I, I') \), i.e. \( I \) sees \( I' \) iff \( I' \) is a minimal model of \( P \) and for all \( p_1, \ldots, p_n \in \text{At}_L \):
  
  \[
  I \models \text{Not} (p_1 \land \cdots \land p_n) \implies I' \models \neg p_1 \lor \cdots \lor \neg p_n.
  \]
- The valuation \( V(I) \) of a reduced interpretation \( I = I_{\text{obj}} \cup I_{\text{not}} \) is the objective part \( I_{\text{obj}} \).

As before, we denote by \( \mathcal{K}(I) \) the full interpretation satisfying the default atoms true in world \( I \), i.e.

\[
\mathcal{K}(I) \models \text{Not} (F) \iff \text{for all } I' \in W \text{ with } R(I, I'): (\mathcal{K}, I') \models \neg F.
\]

Now the reduced interpretation \( I = I_{\text{obj}} \cup I_{\text{not}} \) assigns a truth value to the atoms \( \text{Not} (p_1 \land \cdots \land p_n) \) and the Kripke structure also assigns a truth value to them. But the construction guarantees that the truth values always agree:

**Lemma D.2** For \( I = (I_{\text{obj}} \cup I_{\text{not}}) \in W \) and all \( p_1, \ldots, p_n \in \text{At}_L \):

\[
\mathcal{K}(I) \models \text{Not} (p_1 \land \cdots \land p_n) \iff I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n).
\]

**Proof.**

- Let \( I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n) \). By the construction, \( I' \models \neg p_1 \lor \cdots \lor \neg p_n \) holds for all worlds \( I' \) seen by \( I \), i.e. satisfying \( R(I, I') \). Thus, \( \mathcal{K}(I) \models \text{Not} (p_1 \land \cdots \land p_n) \).
Let \( I_{not} \not\models \text{Not} (p_1 \land \cdots \land p_n) \). Since \( N^\omega \) is a fixpoint of \( \Theta_P \), we have \( N^\omega = \Pi_P(\Omega_P(N^\omega)) \). By the definition of \( \Pi_P \), there is a non-empty \( O' \subseteq \Omega_P(N^\omega) \) such that the default atoms true in \( I_{not} \) are the intersection of the corresponding true negations in \( O' \). Thus, there is \( I'_{obj} \in O' \) with \( I'_{obj} \models p_1 \land \cdots \land p_n \). Furthermore, for all \( q_1, \ldots, q_m \) with \( I_{not} \models \text{Not} (q_1 \land \cdots \land q_m) \), we have \( I'_{obj} \models \neg q_1 \lor \cdots \lor \neg q_m \). Now by the definition of \( \Omega_P \), there is a \( I'_{not} \in N^\omega \) such that \( I' = I'_{obj} \cup I'_{not} \) is a minimal model of \( P \). It follows that \( R(I, I') \) holds, i.e. \( I \) sees a world with valuation \( I'_{obj} \), in which \( \neg p_1 \lor \cdots \lor \neg p_n \) is false. Thus \( \mathcal{K}(I) \not\models \text{Not} (p_1 \land \cdots \land p_n) \). 

\[ \Box \]

**Lemma D.3** For all reduced interpretations \( I \in W \), the full interpretation \( \mathcal{I} = \mathcal{K}(I) \) is a model of the least static expansion \( \mathcal{P} \).

**Proof.** We show that the conditions of Theorem C.3 are satisfied:

- We first have to show that every world \( I = I_{obj} \cup I_{not} \) sees at least one world \( I' \). This follows from \( I_{not} \in N^\omega \), i.e. \( I_{not} \in \Pi_P(\Omega_P(N^\omega)) \). By the definition of \( \Pi_P \), there is a non-empty subset \( O \subseteq \Omega_P(N^\omega) \) such that every \( I'_{obj} \in O \) satisfies for all \( p_1, \ldots, p_n \in \text{At}_L \):

\[
I_{not} \models \text{Not} (p_1 \land \cdots \land p_n) \implies I'_{obj} \models \neg p_1 \lor \cdots \lor \neg p_n.
\]

Now by the definition of \( \Omega_P \), there is a \( I'_{not} \in N^\omega \) such that \( I' = I'_{obj} \cup I'_{not} \) is a minimal model of \( P \). But then \( R(I, I') \) holds.

- The reduced interpretations \( I \in W \) satisfy the program \( P \), and by Lemma D.2, \( I \) and \( \mathcal{I} = \mathcal{K}(I) \) agree on the atoms occurring in \( P \). Thus, also \( \mathcal{I} \) is a model of \( P \).

- For every \( (I, I') \in R \), the reduced interpretation \( I' \) is a minimal model of \( P \). If the full interpretation \( \mathcal{I}' = \mathcal{K}(I') \) were not a minimal model of \( P \), i.e. there were a smaller model \( \mathcal{I}' \) of \( P \), then its reduct \( \mathcal{I}' \) would be a smaller model than \( I' \) (using again Lemma D.2 in order to conclude that \( I' \) and \( \mathcal{I}' \), and thus \( I' \) and \( \mathcal{I}' \) agree on the default atoms).

Now Theorem C.3 allows us to conclude that the formulas true in all worlds of \( W \) are a static expansion of \( P \). Of course, every \( \mathcal{I} = \mathcal{K}(I) \) is a model of this static expansion. But the least static expansion \( \mathcal{P} \) is a subset, so \( \mathcal{I} \) is also a model of \( \mathcal{P} \).

\[ \Box \]

**Lemma D.4** \( \text{Not} (\neg F_1 \land \cdots \land \neg F_m \land G) \vdash_{\text{Not}} \text{Not} (F_1) \land \cdots \land \text{Not} (F_m) \rightarrow \text{Not} (G) \).

**Proof.** This is a simple exercise in applying the axioms of \( L_{\text{Nat}} \): First, we get

\[
\text{Not} (F_1 \lor \cdots \lor F_m \lor (\neg F_1 \land \cdots \land \neg F_m \land G)) \leftrightarrow \text{Not} (F_1 \lor \cdots \lor F_m \lor G)
\]

by the invariance inference rule. Then we apply on both sides the distributive axiom:

\[
\text{Not} (F_1) \land \cdots \land \text{Not} (F_m) \land \text{Not} (\neg F_1 \land \cdots \land \neg F_m \land G)
\]

\[
\leftrightarrow \text{Not} (F_1) \land \cdots \land \text{Not} (F_m) \land \text{Not} (G).
\]

This implies propositionally:

\[
\text{Not} (F_1) \land \cdots \land \text{Not} (F_m) \land \text{Not} (\neg F_1 \land \cdots \land \neg F_m \land G) \rightarrow \text{Not} (G).
\]

Now we insert our precondition and get the required formula.

\[ \Box \]

39
Lemma D.5 Let \( P \) be finite, \( \mathcal{I} \) be a full model of the least static expansion \( \mathcal{P} \), and let \( I = I_{\text{obj}} \cup I_{\text{not}} \) be its reduct. Then \( I_{\text{not}} \in \mathcal{N}^\circ \).

Proof. We show by induction on \( k \) that the default part \( I_{\text{not}} \) of a model \( \mathcal{I} \) of the least static expansion \( \mathcal{P} \) is contained in \( \Theta_P^k(\mathcal{N}^\circ \mathcal{O}^T) \). For \( k = 0 \) this is trivial, since \( \mathcal{N}^\circ \mathcal{O}^T \) is the complete set of default interpretations.

Let us assume that \( I_{\text{not}} \in \Theta_P^k(\mathcal{N}^\circ \mathcal{O}^T) \). We have to show that \( I_{\text{not}} \) is not “filtered out” by one further application of \( \Theta_P \). Let \( \mathcal{N} := \Theta_P^k(\mathcal{N}^\circ \mathcal{O}^T) \), \( \mathcal{O} := \Omega_P(\mathcal{N}) \), and

\[
\mathcal{O}' := \{ I_{\text{obj}} \models \mathcal{O} : \text{for every default atom } \neg (p_1 \land \cdots \land p_n) : \\
\text{if } I_{\text{not}} \models \neg (p_1 \land \cdots \land p_n), \text{then } I_{\text{obj}} \models \neg p_1 \lor \cdots \lor \neg p_n \}. 
\]

We will now show that \( \mathcal{O}' \) has the properties required in the definition of \( \Pi_P \), namely we will show

\[
I_{\text{not}} \models \neg (p_1 \land \cdots \land p_n) \iff \text{for all } I'_{\text{obj}} \models \mathcal{O}' : I'_{\text{obj}} \models \neg p_1 \lor \cdots \lor \neg p_n.
\]

This implies that \( \mathcal{O}' \) is non-empty because for \( n = 0 \) the empty conjunction is logically true. Because of the consistency axiom, \( I_{\text{not}} \not\models \neg \text{true} \). But then there must be at least one \( I'_{\text{obj}} \models \mathcal{O}' \), because otherwise the “for all” on the right hand side would be trivially true. Also the direction \( \Rightarrow \) follows trivially from the construction.

Now we have to show that for every default atom \( \neg (p_1 \land \cdots \land p_n) \) which is false in \( I_{\text{not}} \), there is \( I_{\text{obj}} \models \mathcal{O}' \) with \( I'_{\text{obj}} \not\models \neg p_1 \lor \cdots \lor \neg p_n \). Let \( \neg (q_{i,1} \land \cdots \land q_{i,n_i}) \), \( i = 1, \ldots, m \), be all default atoms true in \( I_{\text{not}} \) (containing only the finitely many objective propositions occurring in \( P \)). Since \( \mathcal{I} \models \mathcal{P} \), the formula

\[
\left( \bigwedge_{i=1}^m \neg (q_{i,1} \land \cdots \land q_{i,n_i}) \right) \Rightarrow \neg (p_1 \land \cdots \land p_n)
\]

cannot be contained in \( \mathcal{P} \) (it is violated by \( I_{\text{not}} \) and thus by \( \mathcal{I} \)). However, Lemma D.4 shows that the above formula would follow from

\[
\neg \left( \bigwedge_{i=1}^m (\neg q_{i,1} \lor \cdots \lor \neg q_{i,n_i}) \right) \land (p_1 \land \cdots \land p_n).
\]

Thus, this formula is also not contained in \( \mathcal{P} \). But the static semantics requires that \( \neg (F) \in \mathcal{P} \) if \( \mathcal{P} \models_{\text{min}} \neg F \). So if \( \neg (F) \not\in \mathcal{P} \), there must be a minimal model \( \mathcal{I}' \) of \( \mathcal{P} \) violating \( \neg F \), i.e. satisfying \( F \). In our case this means that there is a minimal model \( \mathcal{I}' \) of \( \mathcal{P} \) with \( \mathcal{I}' \models \neg q_{i,1} \lor \cdots \lor \neg q_{i,n_i} \) for \( i = 1, \ldots, m \) and \( \mathcal{I}' \models (p_1 \land \cdots \land p_n) \), i.e. \( \mathcal{I}' \not\models \neg p_1 \lor \cdots \lor \neg p_n \).

But by our inductive hypothesis, the default part \( I'_{\text{not}} \) of \( \mathcal{I}' \) is contained in \( \mathcal{N} \), and thus the objective part \( I'_{\text{obj}} \) is in \( \mathcal{O} \). Since \( I'_{\text{obj}} \models \neg q_{i,1} \lor \cdots \lor \neg q_{i,n_i} \), it is contained in \( \mathcal{O}' \). Thus, \( I'_{\text{obj}} \) is the required element.

Now Theorem 6.3 follows directly from Lemma D.3 and Lemma D.5:

Theorem 6.3 (Model-Theoretic Characterization of Static Completions)
Let \( P \) be a finite super program:
1. The operator $\Theta_P$ is monotone and thus its iteration beginning from the set $\mathcal{N}^\mathcal{O}T$ of all interpretations of the default atoms has a fixed point $\mathcal{N}^\circ$.

2. $\mathcal{N}^\circ$ consists exactly of all $I_{not} \in \mathcal{N}^\mathcal{O}T$ that are default parts of a full model $I$ of $\mathcal{P}$.

3. A reduced interpretation $I_{obj} \cup I_{not}$ is a reduct of a full model $I$ of $\overline{\mathcal{P}}$ iff $I_{not} \in \mathcal{N}^\circ$ and $I_{obj} \cup I_{not} \models P$.

Proof.

1. The monotonicity of $\Theta_P$ is obvious: The more default interpretations we have, the more minimal objective models we get. But then there are more possibilities to choose the set $\mathcal{O}'$ in the definition of $\Pi_P$.

2. Part 2 means that for every $I_{not} \in \mathcal{N}^\mathcal{O}T$:

   There is a full model $I$ of $\overline{\mathcal{P}}$ with belief part $I_{not} \iff I_{not} \in \mathcal{N}^\circ$.

   The direction $\implies$ is Lemma D.5, and the direction $\impliedby$ follows from Lemma D.3 and the fact that any default interpretation $I_{not}$ can be extended to a reduced model $I = I_{obj} \cup I_{not}$ of $\mathcal{P}$ (by making all objective atoms $At_L$ true in $I_{obj}$). But then $I \in W$.

3. The third part requires that for every reduced interpretation $I = I_{obj} \cup I_{not}$:

   There is a full model $I$ of $\overline{\mathcal{P}}$ with reduct $I \iff I_{not} \in \mathcal{N}^\circ$ and $I_{obj} \cup I_{not} \models P$.

   Here the direction $\impliedby$ is Lemma D.3 and the direction $\implies$ follows trivially from Lemma D.5 and $\mathcal{P} \subseteq \overline{\mathcal{P}}$.

Remark D.1 Note that we have used the finiteness of $\mathcal{P}$ only in the proof to Lemma D.5. So even in the infinite case, our model-theoretic construction yields models of the least static expansion, but it does not necessarily yield all reduced models. One example, where a difference might occur, is $\mathcal{P} = \{q_i \lor r_i : i \in \mathbb{N}\} \cup \{r_i \lor p : i \in \mathbb{N}\}$. Our model-theoretic construction excludes an interpretation which makes all $\text{Not}(q_i)$ true and $\text{Not}(p)$ false. It seems plausible that $\overline{\mathcal{P}}$ allows such models, because it would need an “infinite implication” to exclude them. But this question needs further research.

E Proof of Fixpoint Characterization (Theorem 5.1)

We prove Theorem 5.1 by using the model-theoretic characterization. More specifically, we show that formulas $\text{Not}(E_1) \land \cdots \land \text{Not}(E_m) \rightarrow \text{Not}(E_0)$ contained in $\mathcal{P}^n$ characterize exactly the default interpretations remaining after $n$ applications of $\Theta_P$. But first we need the monotonicity of the sequence $\mathcal{P}^n$:

Lemma E.1 For every $n \in \mathbb{N}$: $\mathcal{P}^n \subseteq \mathcal{P}^{n+1}$.
Proof. The propositional consequence operator \( Cn \) is monotonic and has no influence on the minimal models, so it suffices to show that the sequence \( \hat{P}_0 := P \),
\[
\hat{P}^{n+1} := P \cup \{ \text{Not} E_1 \land \cdots \land \text{Not} E_m \rightarrow \text{Not} E_0 \} : \hat{P}^n \models_{\text{min}} \neg E_1 \land \cdots \land \neg E_m \rightarrow \neg E_0
\]
increases monotonically.

The proof is by induction on \( n \). The case \( n = 0 \) is trivial. For larger \( n \), the inductive hypothesis gives us \( \hat{P}^n \supseteq \hat{P}^{n-1} \) and all formulas in \( \hat{P}^n \setminus \hat{P}^{n-1} \) contain only default atoms. So Lemma B.2 yields that the minimal models of \( \hat{P}^n \) are a subset of the minimal models of \( \hat{P}^{n-1} \), and therefore \( \hat{P}^{n-1} \models_{\text{min}} \neg E_1 \land \cdots \land \neg E_m \rightarrow E_0 \) implies \( \hat{P}^n \models_{\text{min}} \neg E_1 \land \cdots \land \neg E_m \rightarrow E_0 \), i.e. \( \hat{P}^{n+1} \supseteq \hat{P}^n \).

Next, we have the small problem that we must in principle look at infinitely many default negation atoms, although we have required our program to be finite. For instance, \( \text{Not} (p \land \neg p) \), \( \text{Not} (p \land p) \), and so on are different default atoms, and in general, default interpretations could assign different truth values to them. However, already \( P^1 \) excludes this:

**Definition E.2 (Regular Model)**

Let a super program \( P \) be given. A default interpretation \( I_{\text{not}} \) is called regular wrt \( P \) iff
- If \( I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n) \) and \( \{p_1, \ldots, p_n\} \subseteq \{q_1, \ldots, q_m\} \), then \( I_{\text{not}} \models \text{Not} (q_1 \land \cdots \land q_m) \).
- \( I_{\text{not}} \models \text{Not} (p_1 \land \cdots \land p_n) \) if some \( p_i \) does not occur in \( P \).

**Lemma E.3** All \( I_{\text{not}} \in \mathcal{N}_i, i \geq 1 \) are regular.

**Proof.**
- Every interpretation \( I \) satisfies \( \neg (p_1 \land \cdots \land p_n) \rightarrow \neg (q_1 \land \cdots \land q_m) \), if \( \{p_1, \ldots, p_n\} \subseteq \{q_1, \ldots, q_m\} \). So \( \text{Not} (p_1 \land \cdots \land p_n) \rightarrow \text{Not} (q_1 \land \cdots \land q_m) \) is contained in all \( P^i, i \geq 1 \).
- Every minimal model \( I \) of \( P \) satisfies \( \neg (p_1 \land \cdots \land p_n) \) if some \( p_i \) does not occur in \( P \) (because \( I \models \neg p_i \)). But then \( \text{Not} (p_1 \land \cdots \land p_n) \in P^1 \), and by Lemma E.1 it is contained also in every \( P^n, n \geq 1 \).

**Lemma E.4** Let \( P_{\text{Not}}^n \) be the set of formulas of the form \( \text{Not} (E_1) \land \cdots \land \text{Not} (E_m) \rightarrow \text{Not} (E_0) \) contained in \( P^n \). Furthermore, let \( \mathcal{N}_0 := \mathcal{N}_{\text{OT}} \) and \( \mathcal{N}_{n+1} := \Theta_P (\mathcal{N}_n) \). Then for every \( I = I_{\text{obj}} \cup I_{\text{not}} \) with \( \models P^n \colon I_{\text{not}} \models P_{\text{Not}}^n \iff I_{\text{not}} \in \mathcal{N}_n \).

**Proof.** The proof is by induction on \( n \). The case \( n = 0 \) is trivial, since \( \mathcal{N}_0 = \mathcal{N}_{\text{OT}} \) and \( P^0 = P \) and we anyway consider only models of \( P \).

- “\( \Leftarrow \)”: Let \( I_{\text{not}} \in \mathcal{N}_{n+1} = \Theta_P (\mathcal{N}_n) \). We have to show that \( I_{\text{not}} \models P_{\text{Not}}^{n+1} \). Suppose that this were not the case, i.e. \( I_{\text{not}} \) violates a formula \( \text{Not} E_1 \land \cdots \land \text{Not} E_m \rightarrow \text{Not} E_0 \) contained in \( P_{\text{Not}}^{n+1} \). This means that \( I_{\text{not}} \models \text{Not} E_i \) for \( i = 1, \ldots, m \), but \( I_{\text{not}} \not\models \text{Not} E_0 \). By the definition of \( \Theta_P \), there must be an objective model \( I'_{\text{obj}} \in \Omega_P (\mathcal{N}_n) \) (contained in the non-empty \( O' \)) such that \( I'_{\text{obj}} \models \neg E_i \) for \( i = 1, \ldots, m \) and \( I'_{\text{obj}} \not\models \neg E_0 \). By the definition of \( \Omega_P \), there must be a default interpretation \( I'_{\text{not}} \in \mathcal{N}_n \) such that \( I' = I'_{\text{obj}} \cup I'_{\text{not}} \) is a minimal model of \( P \). Then the inductive hypothesis gives us \( I'_{\text{not}} \models P_{\text{Not}}^n \), and Lemma B.2 allows us to conclude that \( I' \) is a minimal model of \( P \cup P_{\text{Not}}^n \) and
thus of $P^n$. But this means that $P^n \not\models_{\min} \text{Not } E_1 \land \cdots \land \text{Not } E_m \rightarrow \text{Not } E_0$. Since $I'$ is a model of $P$, the critical formula cannot be contained in $P$ (if $E_0$ is the empty conjunction and $P$ is not affirmative, this would be syntactically possible). And finally, it cannot be introduced by the $\mathcal{C}n$-operator, since $I'$ is a model of its preconditions.

Thus, it is impossible that $\text{Not } E_1 \land \cdots \land \text{Not } E_m \rightarrow \text{Not } E_0$ is contained in $P^n_{\text{Not}}$.

- “$\Longrightarrow$”: Let $I_{\text{not}} \models P^n_{\text{Not}}$. By Lemma E.1 we have $P^n_{\text{Not}} \subseteq P^{n+1}_{\text{Not}}$, so $I_{\text{not}} \models P^n_{\text{Not}}$, and the inductive hypothesis gives us $I_{\text{not}} \in \mathcal{N}_n$. We have to show that $I_{\text{not}}$ is not “filtered out” by one further application of the $\Theta_P$-operator. Let

$$O' := \{ I'_{\text{obj}} \in \Omega_P(\mathcal{N}_n) : I'_{\text{obj}} \models \neg E \text{ for every } \text{Not } E \text{ with } I_{\text{not}} \models \text{Not } E \}.$$

We have to show that for every $E_0$ with $I_{\text{not}} \not\models \text{Not } E_0$ there is an $I'_{\text{obj}} \in O'$ with $I'_{\text{obj}} \not\models \neg E_0$. This especially implies that $O'$ is non-empty, since $I_{\text{not}} \not\models \text{Not (true)}$, which is obviously contained in $P^{n+1}_{\text{Not}}$.

Now suppose that this were not the case, i.e. there were an $E_0$ such that there is no $I'_{\text{obj}} \in O'$ with $I'_{\text{obj}} \not\models \neg E_0$.

Let $\text{Not } (E_i), i = 1, \ldots, m$, be all belief atoms which are true in $I_{\text{not}}$ and satisfy the following conditions (in order to make the set finite): First, the conjunctions $E_i$ contain only propositions $p \in \mathcal{A}_L$ occurring in $P$ (which was required to be finite), and second, each $E_i$ contains each proposition at most once.

We will now show that the formula $\text{Not } E_1 \land \cdots \land \text{Not } E_m \rightarrow \text{Not } E_0$ is contained in $P^{n+1}_{\text{Not}}$, which is a contradiction to $I_{\text{not}} \models P^n_{\text{Not}}$. Let $I' = I'_{\text{obj}} \cup I'_{\text{not}}$ be any minimal model of $P^n$. We have to show that it satisfies $\neg E_1 \land \cdots \land \neg E_m \rightarrow \neg E_0$. The induction hypothesis gives us $I'_{\text{not}} \in \mathcal{N}_n$, and thus $I'_{\text{obj}} \in \Omega_P(\mathcal{N}_n)$. Suppose that $I'_{\text{obj}} \not\models \neg E_i$ for $i = 1, \ldots, m$, since otherwise the formula is trivially satisfied. Since $I'$ is a minimal model and $I_{\text{not}}$ is regular, this means that $I'_{\text{obj}} \in O'$ (the validity of $\neg E_1, \ldots, \neg E_m$ implies the validity of all other $\neg E$ considered in the construction of $O'$). But we have assumed above that no element of $O'$ violates $\neg E_0$, so we get $I'_{\text{obj}} \models \neg E_0.$

\textbf{Lemma E.5} $P^{n_0} \subseteq P|L^*_{\text{Not}}$.

\textbf{Proof.} We show $P^n_{\text{Not}} \subseteq P$ by induction on $n$ (this implies the above statement since $P$ is closed under consequences and $P^n_{\text{Not}} \subseteq L^*_{\text{Not}}$).

For $n = 0$ this is trivial since $P \subseteq P$. Now suppose that $P^n \models_{\min} \neg E_1 \land \cdots \land \neg E_m \rightarrow \neg E_0$. By Theorem 3.8 and the definition of $\mathcal{C}n_{\text{Not}}, P$ differs from $P$ only by the addition of formulas containing only default atoms plus propositional consequences. By the inductive hypothesis, $P^n_{\text{Not}} \subseteq P$. Now Lemma B.2 gives us that all minimal models of $P$ are also minimal models of $P^n$, and therefore $P \models_{\min} \neg E_1 \land \cdots \land \neg E_m \rightarrow \neg E_0$, i.e. $P \models_{\min} \neg E_1 \land \cdots \land \neg E_m \land E_0$. This means that $\text{Not } (E_1 \land \cdots \land \neg E_m \land E_0) \in P$, and by Lemma D.4 we get that $\text{Not } E_1 \land \cdots \land \text{Not } E_m \rightarrow \text{Not } E_0$ is contained in $P$.

Now we can complete the proof of Theorem 5.1. First, a fixpoint is reached after a finite number of iterations, because by Lemma E.3 we know that there are only a finite number of “really different” default negation atoms, so after the first iteration (which ensures the regularity) it suffices to consider a finite number of implications $\text{Not } E_1 \land \cdots \land \text{Not } E_m \rightarrow \text{Not } E_0$, and the sets $P^n$ are monotonically increasing (Lemma E.1).
So we have $P_{\text{Not}}^{n_0} = P_{\text{Not}}^{n_0+1}$ and thus $\mathcal{N}_{n_0} = \mathcal{N}_{n_0+1} = \mathcal{N}^\circ$. Now for any reduced interpretation $I = I_{\text{obj}} \cup I_{\text{not}}$: if $I \models P_{\text{Not}}^{n_0}$, then $I \models P$ and $I \models P_{\text{Not}}^{n_0}$, and by Lemma E.4 we get $I_{\text{not}} \in \mathcal{N}^\circ$. Now the already proven Theorem 6.3 implies that $I$ is a reduct of a full model of the least static expansion $\overline{P}$. But this implies $I \models \overline{P} \mathcal{L}_{\text{Not}}^\ast$. The other direction $I \models \overline{P} \mathcal{L}_{\text{Not}}^\ast \implies I \models P_{\text{Not}}^{n_0}$ follows from Lemma E.5.

From the equivalence of $P_{\text{Not}}^{n_0}$ and $\overline{P} \mathcal{L}_{\text{Not}}^\ast$, we get $P_{\text{Not}}^{n_0} = \overline{P} \mathcal{L}_{\text{Not}}^\ast$, since both sets are closed under propositional consequences: For instance, let $F \in P_{\text{Not}}^{n_0}$ and suppose that $F \not\in \overline{P} \mathcal{L}_{\text{Not}}^\ast$. Since $\overline{P} \mathcal{L}_{\text{Not}}^\ast$ is closed under propositional consequences, there must be a reduced interpretation $I$ with $I \models \overline{P} \mathcal{L}_{\text{Not}}^\ast$, but $I \not\models F$. This is impossible since we already know that every model of $\overline{P} \mathcal{L}_{\text{Not}}^\ast$ is also a model of $P_{\text{Not}}^{n_0}$.

Remark E.1 We have used the finiteness of $P$ in the proof for $I_{\text{not}} \models P_{\text{Not}}^{n_0} \implies I_{\text{not}} \in \mathcal{N}^\circ$. This was to be expected, since we strongly conjecture that for infinite programs, the model-theoretic construction does not yield all models of the static completion. However, this does not give us any hint whether Theorem 5.1 might hold for infinite programs. This question is topic of our future research.