The relationship between stable, supported, default and autoepistemic semantics for general logic programs

W. Marek

Computer Science Department, University of Kentucky, Lexington, KY 40506-0027, USA

V.S. Subrahmanian

Department of Computer Science, A.V. Williams Building, University of Maryland, College Park, MD 20742, USA

Communicated by G. Jäger
Received January 1989
Revised September 1990

Abstract


We investigate the relationship between various alternative semantics for logic programming, viz. the stable model semantics of Gelfond and Lifschitz (1988), the supported model semantics as developed by Apt, Blair and Walker (1988), autoepistemic translations (cf. Moore (1985)) of general logic programs and default translations of general logic programs, Reiter (1980).

1. Introduction

Several techniques have been proposed to handle negative information in deductive databases and logic programs. These include, in the AI community, the methods of circumscription [22, 23, 13, 15], default logics [29, 32], and autoepistemic logics [24, 19].

In the logic programming community, the general idea has been to identify one or more models of the completion (cf. Clark [4]) of a program as being the intended meaning(s) of the program. These techniques led to the notion of stratification [2, 31] in which a so-called standard (or canonical) model of the program completion was constructed and it was claimed that this model was the intended model of the
program. This approach was extended by Przymusiński [27] who defined a class of programs called locally stratified programs. Recently, Gelfond and Lifschitz [7] proposed a stable model semantics for logic programs and showed that the stable model semantics extends the stratified semantics.

However, it was soon realized by various researchers that a close investigation of the relationship between these varying formalisms was needed. This is because the number of such schemes for handling negative information is rapidly increasing—before allowing such an increase, one needs to examine the relationships between different schemes to understand exactly what the differences are, and to determine where the strengths and/or weaknesses of a particular scheme lie. At this point we are aware of investigations of the following relationships:

1. between circumscription and Clark’s completion [10, 29],
2. between circumscription and the closed world assumption [5, 9, 13, 26],
3. between autoepistemic logic and circumscription [8],
4. between default logic and circumscription [11],
5. between the Clark completion and the Closed World Assumption [16, 30],
6. between autoepistemic logic and default reasoning [12, 20].

In this paper, we study the connection between the supported model semantics for logic programming as developed by Apt, Blair and Walker [2], and nonmonotonic logic based semantics for logic programming. The latter consists of translating general logic programs into either autoepistemic theories or default logic theories. We show that there is a close correspondence between the semantics of logic programs under these differing semantical characterizations.

2. Supported models and stable models

We assume that the reader is familiar with the usual notions of term, atom, etc. Unless explicitly stated otherwise, the languages we consider contain function symbols. Any unexplained notation may be assumed to be the same as that in Lloyd [16].

**Definition 1.** If \( A \) is an atom and \( L_1, \ldots, L_n \) are literals (that is atoms or negated atoms), then

\[
A \leftarrow L_1 \land \cdots \land L_n
\]

is a clause. \( A \) is called the head of the above clause, and \( L_1 \land \cdots \land L_n \) is called the body of the above clause.

**Definition 2.** A general logic program is a finite set of clauses.

For the sake of notational simplicity, we will assume that the body of any clause is written as

\[
B_1 \land \cdots \land B_k \land \neg D_1 \land \cdots \land \neg D_m
\]
where the $B_i$’s and $D_i$’s are all atomic. Thus, in the body of any clause, the negative atoms occur after the positive atoms. As conjunction is commutative in nature, there is no loss of generality, as long as we deal with semantics with a commutative interpretation of conjunction only, in making this assumption.

We may, in fact, assume that a general logic program $P$ is a possibly infinite set of ground clauses. This is the same simplifying assumption made by Gelfond and Lifschitz [7]. Unless explicitly mentioned otherwise, throughout this paper, we assume that $P$ is a possibly infinite set of ground clauses. The completion of $P$ is defined in the same way as in Lloyd [16] except that one may now have infinitary disjunctions occurring in the completion. As usual, we restrict our interest to Herbrand models only and consider an interpretation to be a subset of the Herbrand Base $B_P$ of the program $P$. The following definition is due to Apt, Blair and Walker [2].

**Definition 3.** An interpretation $M$ of $P$ is **supported** if and only if for all $A \in M$, there is a clause in $P$ of the form

$$
A \leftarrow B_1 \& \cdots \& B_k \& \neg D_1 \& \cdots \& \neg D_m
$$

such that $M \models B_1 \& \cdots \& B_k \& \neg D_1 \& \cdots \& \neg D_m$. (Here, $\models$ is the satisfaction relation.)

**Proposition 1** (Apt, Blair, Walker [2]). $M$ is a supported model of $P$ if and only if $T_p(M) = M$.

**Proposition 2.** $I$ is an Herbrand model of $\text{comp}(P)$ if and only if $T_p(I) = I$.

We now give the standard definition of stable models due to Gelfond and Lifschitz [7].

**Definition 4.** Suppose $P$ is a logic program, and $I$ an interpretation. The Gelfond-Lifschitz transformation, $G(I, P)$ of $P$ w.r.t. interpretation $I$, is the program defined below.

1. If $C$ is a clause in $P$ of the form

$$
A \leftarrow B_1 \& \cdots \& B_n \& \neg D_1 \& \cdots \& \neg D_m
$$

where $m \geq 0$, and if for all $1 \leq i \leq m$, $D_i \not\in I$, then the clause $A \leftarrow B_1 \& \cdots \& B_n$ is in $G(I, P)$.

2. Nothing else is in $G(I, P)$.

An interpretation $I$ is said to be **GL-stable** if and only if $I = T_{G(I,P)} \omega$.

The following alternative idea of stability is due to Truszczynski. We describe it below, and then prove that both concepts of stability are identical.
Definition 5. Suppose $P$ is a general logic program and $M \subseteq B_P$ is an interpretation. The Gelfond–Lifschitz–Truszczynski transform $G_T$ takes an Herbrand interpretation $M$ and a (general) logic program $P$ as input and produces a new pure (that is negation-free) logic program, denoted $G_T(M, P)$ as output. $G_T(M, P)$ is obtained as follows:

1. If $C$ is a negation-free clause in $P$ of the form $A \leftarrow B_1 \land \cdots \land B_n$ and for all $1 \leq i \leq n$, $B_i \in M$, then $C \in G_T(M, P)$.

2. If $A \leftarrow B_1 \land \cdots \land B_k \land \neg D_1 \land \cdots \land \neg D_m$ is a clause in $P$ such that for all $1 \leq j \leq k$, all $B_j$ belong to $M$ and for all $1 \leq j \leq k$, $D_j \notin M$, then $A \leftarrow B_1 \land \cdots \land B_k$ is in $G_T(M, P)$.

3. Nothing else is in $G_T(M, P)$.

$M$ is said to be $GLT$-stable if and only if $M = T_{G_T(M, P)}[\omega]$.

Even though, in general, $G(I, P)$ and $G_T(I, P)$ are not identical, it is nevertheless true that the concepts of $CL$-stable models, and $GLT$-stable models are the same.

Theorem 1. Suppose $P$ is a program, and $I$ is an interpretation. Then $I$ is $CL$-stable w.r.t $P$ if and only if $I$ is $GLT$-stable w.r.t $P$.

Proof. ($\Rightarrow$) Suppose $I$ is $GL$-stable. We need to show that $I = T_{G_T(I, P)}[\omega]$.

($I \subseteq T_{G_T(I, P)}[\omega]$) Suppose $A \in I$. Then, as $I$ is $GL$-stable, $A \in T_{G_T(I, P)}[\omega]$, and hence $A \in T_{G_T(I, P)}[k]$ for some $k$. We proceed by induction on $k$.

Base Case: ($k = 0$) If $k = 0$, then

\[ A \]

is in $G(I, P)$. Hence, there is a clause $A \leftarrow \neg B_1 \land \cdots \land \neg B_n$ in $P$ such that $I \models \neg B_1 \land \cdots \land \neg B_n$. Thus, the clause $A \leftarrow$ is also in $G_T(I, P)$. Hence, $A \in T_{G_T(I, P)}[1]$.

Inductive Case: Suppose $A \in T_{G_T(I, P)}[k]$ for $k > 1$. Then there is a clause

\[ A \leftarrow D_1 \land \cdots \land D_m \]

in $G(I, P)$ such that $I \models D_1 \land \cdots \land D_m$. Thus, there is a clause

\[ A \leftarrow D_1 \land \cdots \land D_m \land \neg B_1 \land \cdots \land \neg B_n \]

in $P$ such that $I \models \neg B_1 \land \cdots \land \neg B_n$. But then, the clause $A \leftarrow D_1 \land \cdots \land D_m$ is in $G_T(I, P)$. By the induction hypothesis, we may assume that $\{D_1, \ldots, D_m\} \subseteq T_{G_T(I, P)}[\omega]$, and hence is in $T_{G_T(I, P)}[k_0]$ for some integer $k_0$. Thus, $A \in T_{G_T(I, P)}[(k_0 + 1)] \subseteq T_{G_T(I, P)}[\omega]$.

As $G_T(I, P) \subseteq G(I, P)$, it follows immediately that $T_{G_T(I, P)}[\omega] \subseteq T_{G_T(I, P)}[\omega]$.

($\Leftarrow$) The proof proceeds in the same way as the proof of ($\Rightarrow$) above. \(\square\)

In view of Theorem 1, throughout the rest of this paper, we use the expression “stable model” to mean $GL$-stable model.
Example 1. Suppose $P$ is the general logic program

$$p \leftarrow s \& \neg q$$

$$q \leftarrow \neg r$$

$$r \leftarrow$$

$$s \leftarrow$$

Let $M = \{p, r, s\}$. Then $G(M, P) = G_{T}(M, P)$ is

$$p \leftarrow s$$

$$r \leftarrow$$

$$s \leftarrow$$

and it is easy to see that $M$ is stable.

On the other hand, consider $M' = \{p, r\}$. In this case, $G(M', P)$ and $G_{T}(M', P)$ are different. The clause

$$p \leftarrow s$$

is in $G(M', P)$; however, this clause is not in $G_{T}(M', P)$ because $s \notin M'$.

Theorem 2. Suppose $P$ is a logic program, and $I$ is an interpretation. Then $T_{p}(I) = T_{G_{i}(I, P)}(I)$.

Proof.

$$A \in T_{p}(I)$$

iff $\exists C \in P$ such that $C = A \leftarrow B_{i} \& \&_{j} \neg D_{j}$ and $I \models B_{i} \& \&_{j} \neg D_{j}$

iff $\exists C' \in G(I, P)$ such that $C' = A \leftarrow B_{i}$ and $I \models B_{i}$

iff $A \in T_{G_{i}(I, P)}(I)$. □

Given a logic program $P$ and an interpretation $I$, it may be felt that $G(I, P)$ could be given an equivalent declarative definition by declaring $G_{i}(I, P)$ to be the smallest (w.r.t. the ordering of subset inclusion) definite logic program which is a subset of $G(\emptyset, P)$, if any, such that $T_{G_{i}(I, P)} = T_{p}(I)$. Note that in general, a unique "smallest" set satisfying such conditions may not always exist. Even if such a unique smallest set $G_{i}(I, P)$ exists, it may not coincide with $G(I, P)$ as the following example shows.

Example 2. Let $P$ be the program

$$p \leftarrow p$$

$$p \leftarrow \neg q$$

and let $I = \{q\}$. Then $G(I, P) = \{p \leftarrow p\}$, while $G_{i}(I, P) = \emptyset$. Note here that $T_{G_{i}(I, P)}(I) = \emptyset = T_{p}(I)$. In this case, note that $G_{T}(I, P) = \emptyset$. 
In view of Theorem 2, we may ask ourselves whether it is possible that \( G(I, P) = G_I(I, P) \). This property does not hold either, as the following example shows.

**Example 3.** Let \( P \) be the pure logic program

\[
A \leftarrow \\
B \leftarrow \\
A \dashv B.
\]

Let \( I = \{B\} \). In this case, \( G_I(I, P) = G(I, P) = P \). However, \( T_P(I) = \{A, B\} \). Thus, the smallest logic program \( G_I(I, P) \) such that \( T_P(I) = T_{G_I(I, P)} = \{A \leftarrow, B \leftarrow\} \).

The following result is a corollary of Theorem 2. It tells us that all stable models are also supported.

**Theorem 3.** Every stable interpretation of \( P \) is a supported model of \( P \).

**Proof.**

\[
T_P(M) = I_{G(M, P)}(M) \quad \text{(by Theorem 2)}
\]

\[
= T_{G(M, P)}(T_{G(M, P)}^\omega) \quad \text{(since \( M \) is stable)}
\]

\[
= T_{G(M, P)}^\omega \quad \text{(since \( T_{G(M, P)}^\omega \) is a fixed point of \( T_{G(M, P)} \))}
\]

\[
= M \quad \text{(since \( M \) is stable).} \quad \square
\]

**Corollary 1.** (1) If \( M \) is a stable model of \( P \), then \( M \) is a model of \( \text{comp}(P) \).

(2) \( M \models \text{comp}(P) \) iff \( T_P(M) = M \) iff \( M \) is a supported model of \( P \).

In addition, a stronger version of the corollary holds.

**Theorem 4.** If \( M \) is a stable model of \( P \), then \( M \) is a minimal Herbrand model of \( \text{comp}(P) \), that is \( M \) is a minimal fixed-point of \( T_P \).

**Proof.** Suppose \( M \) is stable and \( M' \subseteq M \) such that \( T_P(M') = M \). \( G(M, P) \subseteq G(M', P) \).

Therefore,

\[
M = T_{G(M, P)}^\omega \subseteq T_{G(M', P)}^\omega.
\]

By Theorem 2, \( T_{G(M', P)}(M') = T_P(M') = M' \). Therefore \( M \subseteq T_{G(M', P)}^\omega \subseteq M' \). \( \square \)

Theorem 4 reflects an improvement upon a theorem of Gelfond and Lifschitz [7] in which they show that if \( M \) and \( N \) are different stable models of a program \( P \), then \( M \not\subseteq N \) and \( N \not\subseteq M \).

Note that the converse to Theorem 4 does not hold, that is \( \text{comp}(P) \) may have a minimal and supported model which is not stable (Example 7).
Stable models of logic programs are closely related to the Default Logic of [29]. Specifically, given a program $P$, we assign to it a default theory $(D_p, W_p)$ constructed as follows: If

$$A \leftarrow B_1 \land \cdots \land B_k$$

is a negation-free clause in $P$, then the formula

$$B_1 \land \cdots \land B_k \Rightarrow A$$

belongs to $W_p$. Nothing else is in $W_p$. Clauses containing negations in the body are interpreted as default rules

$$A \leftarrow B_1 \land \cdots \land B_k \land \neg D_1 \land \cdots \land \neg D_m$$

is transformed to

$$\frac{B_1 \land \cdots \land B_k : \neg D_1, \cdots, \neg D_m}{A}$$

which belong to $D_p$ (Notice, that the default resulting from a clause possesses, possibly, a justification consisting of multiple formulas).

**Example 4.** If $P$ is the general logic program

\[ p \leftarrow q \]
\[ q \leftarrow r \land \neg w \]
\[ w \leftarrow r \land \neg p. \]

Then the default theory $(D_p, W_p)$ associated with $P$ is given by

$$W_p = \{(q \Rightarrow p), w\}$$

$D_p$ consists of the following two default rules:

$$\frac{r \neg w}{q}$$

$$\frac{w \neg p}{r}$$

Formally, a default is a triple $d = \langle p(d), j(d), c(d) \rangle$ where $p(d)$ is called prerequisite of $d$, $j(d)$ is a finite list $\beta_1, \ldots, \beta_m$ of formulas of our underlying language $L$, called the justification of $d$, and $c(d)$ is again a formula of $L$ called the conclusion of $d$. Traditionally, we write

$$d = \frac{p(d): \beta_1, \ldots, \beta_k}{c(d)}.$$
Given a fixed theory $S$ included in $L$ and a fixed default theory $(D, W)$, define

Reiter's operator $R_S$ as follows (in the sequel, $T$ is a set of formulas):

$$R_S(T) = Cn(T \cup \{e(d): d \in D \& p(d) \in T \& \forall_{\beta \in \{d\}} \neg \beta \notin S\}).$$

Here $Cn$ is the consequence operator of classical logic, i.e. $Cn(X)$ is the set of all first order logical consequences of $X$. Further, we assume that double negations are eliminated, i.e. the formula $\neg \neg \psi$ is treated as $\psi$.

The following example illustrates the behavior of the $R_S$ operator.

**Example 5.** Let $(D_p, W_p)$ be as in Example 4. Let $S = \{w\}$ and $T = W_p$. Then

$$R_S(T) = Cn(W_p \cup \{r\}) = Cn(\{w, r, (q \Rightarrow p)\}).$$

Operator $R_S$ is monotone and finitary and so it possesses a fixed point above any set $W$ of formulas. Let $F^S_W$ be the least fixed point of $R_S$ above $W$. $S$ is called an extension of $(D, W)$, if this least fixed point is precisely $S$, that is $F^S_W = S$.

We quote the following result due to Marek and Truszczynski [20].

**Theorem 5.** $M$ is a stable model of a logic program $P$ if and only if $M$ is a maximal set of atoms such that $Cn(W_p \cup M)$ is an extension of $(W_p, D_p)$. \(\square\)

In this fashion we get a close connection between the class of stable models of a program and a classical mode of nonmonotonic reasoning. In Section 3 we shall establish yet another interpretation of logic programs in autoepistemic logic and with the help of it we also establish another connection with default logic. This ties up supported models with different structures for default logic. The following result is a corollary of Theorem 4.

**Lemma 1.** Let $M$ be any model of $P$. Then $T_{G(M,P)}^\omega \subseteq M$.

**Proof.** As $M$ is a model of $P$, $T_p(M) \subseteq M$. But by Theorem 2, $T_{G(M,P)}(M) = T_p(M)$. As $T_{G(M,P)}$ is a monotonic operator, for all ordinals $\alpha$ it is true that $T_{G(M,P)}^\alpha \subseteq M$. \(\square\)

In general, there may exist supported models that are not stable. The following example shows this.

**Example 6.** Let $P$ be the program $\{p \leftarrow p\}$. Then $M = \{p\}$ is a supported model of $P$, but $M$ is not a stable model of $P$. This is because $G(M, P)$ is $P$ itself, and the least model of $P$ is $\emptyset$. Hence, $M$ is not stable.
We know, by the theorems above, that if \( P \) has a stable model, then \( \text{comp}(P) \) is consistent. Unfortunately, there are simple programs having consistent completions but possessing no stable models. (Gelfond and Lifschitz present an example of a program having no stable models, but their program has an inconsistent completion.)

**Example 7.** Let \( P \) be the program

\[
\begin{align*}
C1: & \quad p \leftarrow \neg p \\
C2: & \quad p \leftarrow q \\
C3: & \quad q \leftarrow \neg q.
\end{align*}
\]

From Table 1, we can verify that this program has no stable model; however, \( \text{comp}(P) \) is consistent (the interpretation \( \{p, q\} \) is a model of \( \text{com}(P) \)) and indeed, this is the only Herbrand model of \( \text{comp}(P) \) and hence the only supported model of \( P \) which is necessarily minimal by uniqueness. It is however, not stable.

<table>
<thead>
<tr>
<th>( I )</th>
<th>( G(I, P) )</th>
<th>( T_{G(I, P)\uparrow} )</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( C2, C3, p \leftarrow {p} )</td>
<td>( \emptyset )</td>
<td>not stable</td>
</tr>
<tr>
<td>( {p} )</td>
<td>( C2, C3 )</td>
<td>( \emptyset )</td>
<td>not stable</td>
</tr>
<tr>
<td>( {q} )</td>
<td>( C2, C3, p \leftarrow {p} )</td>
<td>( \emptyset )</td>
<td>not stable</td>
</tr>
<tr>
<td>( {p, q} )</td>
<td>( C2, C3 )</td>
<td>( \emptyset )</td>
<td>not stable</td>
</tr>
</tbody>
</table>

Theorem 4 tells us that all stable models are minimal supported models. Example 7 demonstrates that there are minimal supported models of logic programs that are not stable, that is the converse of Theorem 4 does not hold.

**Example 8.** Let \( P \) be the program

\[
\begin{align*}
p & \leftarrow \neg q \\
q & \leftarrow \neg p.
\end{align*}
\]

Then \( M_1 = \{p\} \) and \( M_2 = \{q\} \) are stable models of \( P \), but \( M_1 \cup M_2 = \emptyset \) is not a stable model of \( P \). Similarly, \( M_1 \cup M_2 = \{p, q\} \) is not stable either. Indeed, it is easy to verify that there is no interpretation \( M \) such that \( M \supseteq M_1 \) and \( M \supseteq M_2 \) such that \( M \) is a stable model of \( P \). Similarly, there is no interpretation \( M' \) such that \( M' \subseteq M_1 \) and \( M' \subseteq M_2 \) such that \( M' \) is a stable model of \( P \).

By [7, Theorem 2], we know that if \( P \) is stratified, then \( P \) has a unique stable model, and this stable model coincides with the model constructed by the iterated fixed-point approach of Apt, Blair and Walker. We show below that this result can be extended to locally stratified programs. First, recall the definition of a locally stratified program [27].
Definition 6. A program $P$ is *locally stratified* if and only if its Herbrand Base $B_P$ can be decomposed into a family of disjoint sets $(H_\alpha)_{\alpha < \gamma}$, where $\gamma$ is an ordinal, and such that whenever

$A \leftarrow B_1 & \cdots & B_n & \neg D_1 & \cdots & \neg D_m$

is a ground instance of a clause in $P$, and $A$ belongs to the stratum $H_\alpha$, then

1. each $B_i$ belongs to the union $\bigcup_{\beta < \alpha} H_\beta$, and
2. each $D_k$ belongs to the union $\bigcup_{\beta < \alpha} H_\beta$.

The least such ordinal $\gamma$ is called the *length* of stratification $(H_\alpha)_{\alpha < \gamma}$.

Definition 7. If $(H_\alpha)_{\alpha < \gamma}$ is a stratification of $P$, then $P_\alpha$ is the set of clauses $C$ in $P$ such that the head of $C$ belongs to $H_\alpha$. Furthermore, $\bar{P}_\alpha = \bigcup_{\beta < \alpha} P_\beta$. Likewise, $\bar{H}_\alpha = \bigcup_{\beta < \alpha} H_\beta$.

In this fashion, $\bar{P}_\gamma = P$. Often, given a local stratification $(H_\alpha)_{\alpha < \gamma}$ of $P$, we will abuse notation and call $(P_\alpha)_{\alpha < \gamma}$ the local stratification of $P$.

Definition 8. Suppose $P$ is a program and $I$ an interpretation. The *cumulative upward iteration of $T_P$ from $I$* is defined as

$T_P \uparrow 0(I) = I$

$T_P \uparrow \beta(I) = \bigcup_{\alpha < \beta} T_P(\uparrow \beta(I))$.

Suppose now that $(H_\alpha)_{\alpha < \gamma}$ is a local stratification of $P$. Consider $(P_\alpha)_{\alpha < \gamma}$. Then the standard model $\mathcal{M}_P$ of $P$ is constructed by the following transfinite induction process;

$S_0 = T_P \uparrow \omega$

and for $0 < \alpha \leq \gamma$

$S_\alpha = \bigcup_{\beta < \alpha} T_P \uparrow \omega(S_\beta)$.

Then $\mathcal{M}_P = S_\gamma$ is the standard model of $P$. The following theorem has been announced by Gelfond and Lifschitz [7] and independently by Przymusiński and Przymusińska (cf. [25]). We include a proof below since none has been published thus far.

Theorem 6. *Suppose $P$ is a locally stratified program. Then*

1. $P$ has a unique stable model, denoted $\mathcal{F}_P$ and
2. $\mathcal{F}_P$ is exactly the model $\mathcal{M}_P$ constructed by the transfinite iteration procedure of Przymusiński.
Proof. Let \( \text{grd}(P) = \bigcup_{\eta \leq \gamma} P_\eta \) be a local stratification of \( P \). By induction of \( \eta < \gamma \), we show that every \( \bar{P}_\eta \) has \( S_\eta \) as its unique stable model.

**Base Case:** \( (\eta = 0) \ P_0 = \emptyset \) and \( S_0 = \emptyset \) and so we are done.

**Inductive Case:** \( (\eta > 0) \). Assume that for all \( \zeta < \eta \), \( \bar{P}_\zeta \) has \( S_\zeta \) as its unique stable model. In addition, we can assume that \( \zeta < \eta \rightarrow S_\zeta \subseteq S_\eta \).

**Case 1:** \( (\eta \) is a limit). In this case \( \bar{P}_\eta = \bigcup_{\zeta \leq \eta} \bar{P}_\zeta \). We need to show that \( S_\eta = \bigcup_{\zeta \leq \eta} S_\zeta \) is a stable model of \( \bar{P}_\eta \). Consider \( G(S_\eta, \bar{P}_\eta) \). By the local stratification of \( P \), \( \bar{P}_\eta \) is also locally stratified. Consequently, \( G(S_\eta, \bar{P}_\eta) = \bigcup_{\zeta \leq \eta} G(S_\zeta, \bar{P}_\zeta) \). Hence, the least model of \( G(S_\eta, \bar{P}_\eta) \) is then equal to \( \bigcup_{\zeta \leq \eta} S_\zeta \). This last set is equal to \( S_\eta \). Hence, \( S_\eta \) is a stable model of \( \bar{P}_\eta \). If \( M \) is a stable model of \( \bar{P}_\eta \), then \( M \cap H_\eta \) is a stable model of \( \bar{P}_\zeta \) and thus equal to \( S_\zeta \) by the induction hypothesis. Therefore,

\[
M = M \cap \bar{H}_\eta
\]

\[
= M \cap \left( \bigcup_{\zeta \leq \eta} \bar{H}_\zeta \right)
\]

\[
= \bigcup_{\zeta \leq \eta} (M \cap \bar{H}_\zeta)
\]

\[
= \bigcup_{\zeta \leq \eta} S_\zeta
\]

\[
= S_\eta.
\]

Hence, \( S_\eta \) is the unique stable model of \( \bar{P}_\eta \).

**Case 2:** \( (\eta = (\zeta + 1) \) for some ordinal \( \zeta \). We know that \( S_\zeta \) is the unique stable model of \( \bar{P}_\zeta \) and \( \bar{P}_\eta = \bar{P}_\zeta \cup P_\zeta \). Now

\[
S_\eta = \bigcup_{\mu < \zeta} T_{P_\mu} \uparrow \omega(S_\mu)
\]

\[
= \bigcup_{\mu < \zeta} T_{P_\mu} \uparrow \omega(S_\mu) \cup T_{P_\zeta} \uparrow \omega(S_\zeta)
\]

\[
= S_\zeta \cup T_{P_\zeta} \uparrow \omega(S_\zeta).
\]

To see that \( S_\eta \) is stable is routine. We shall now prove that this is the unique stable model of \( \bar{P}_\eta = \bar{P}_{\zeta + 1} \). Suppose \( M \) is a stable model of \( \bar{P}_\eta \). Thus, \( M = T_{G_{M, P_\zeta}} \uparrow \omega \). We show that \( T_{G_{M, P_\zeta}} \uparrow \omega = S_\eta \).

\[
(T_{G_{M, P_\zeta}} \uparrow \omega \subseteq S_\eta).
\]

Suppose \( A \in T_{G_{M, P_\zeta}} \uparrow \omega \). Then \( A \in T_{G_{M, P_\zeta}} \uparrow n \) for some integer \( n \). The proof is by a straightforward induction on \( n \).

\[
(S_\eta \subseteq T_{G_{M, P_\zeta}} \uparrow \omega).
\]

We know that

\[
S_\eta = S_\zeta \cup T_{P_\zeta} \uparrow \omega(S_\zeta).
\]

By the induction hypothesis \( S_\zeta \) is the sole stable model of \( \bar{P}_\zeta \), and hence it follows that \( S_\eta \subseteq M \). A straightforward induction demonstrates that \( T_{P_\zeta} \uparrow \omega(S_\zeta) \subseteq M \). Hence, \( S_\eta = S_\zeta \cup T_{P_\zeta} \uparrow \omega(S_\zeta) \subseteq M \). \( \square \)
Proposition 3. Let $P$ be a general logic program and $M$ a recursively enumerable subset of the Herbrand Base $B_P$ of $P$. Then

1. the problem “Is $M$ a stable model for $P$?” is $\Pi_2^0$-hard. Moreover, it is in $\Pi_3^0$.
2. the problem “Is $P' = G(M, P)$?” is $\Pi_2^0$-hard. Moreover, it is in $\Pi_3^0$.

Proof. We outline the proof of (1), the proof of (2) is similar.

First we show that problem (1) is $\Pi_2^0$-hard. This is shown by demonstrating a reducibility to the well-known $\Pi_2^0$-complete problem “Given r.e. sets $S_1$, $S_2$, is $S_1 = S_2$?” Let $Q$ be a pure logic program having success set $S_1$. Then $G(S_1, Q) = G(S_2, Q) = Q$. Clearly,

\[ S_1 = S_2 \iff T_G^1 \omega = S_2 \]

\[ \text{iff } T_{G(S_1, Q)}^1 \omega = S_2 \]

\[ \text{iff } T_{G(S_2, Q)}^1 \omega = S_2 \]

\[ \text{iff } S_2 \text{ is a stable model of } Q. \]

It follows that the problem $S_1 = S_2$ can now be solved by an oracle query asking if $S_2$ is a stable model of $Q$. Hence (1) is $\Pi_2^0$-hard.

We now show that (1) is in $\Pi_1^3$. As $M$ is r.e. and a $P$ is recursive, it is easy to see that the problem of whether a ground clause is in $G(M, P)$ is $\Sigma_0^2$. Thus, $G(M, P)$ is a $\Sigma_0^2$ set of clauses. Let $\Gamma[x]$ be a $\Sigma_0^2$ formula containing free variable $x$ that defines $G(M, P)$. It is easy to show that $T_{G(M, P)}^1 \omega$ is r.e. relative to $G(M, P)$. Thus, $T_{G(M, P)}^1 \omega$ is in $\Sigma_2^0$. Let $G[x]$ be a $\Sigma_0^2$ formula containing free variable $x$ that defines $T_{G(M, P)}^1 \omega$. Then $M$ is a stable model of $P$ if and only if $M = T_{G(M, P)}^1 \omega$ if and only if $(\forall x)(F[x] \Leftrightarrow G[x])$ which is a $\Pi_1^3$ formula.

to see (2) we notice that comparing recursive set ($P'$) and a $\Delta_2^0$ set is expressible as a $\Pi_1^3$ sentence.  \( \square \)

Remark 1. Suppose $P_1$, $P_2$ are general logic programs such that no predicate symbol occurring in $P_1$ occurs in $P_2$ and vice versa. Then, if $M_1$, $M_2$ are stable models of $P_1$, $P_2$ respectively, $M_1 \cup M_2$ is a stable model of $P_1 \cup P_2$.

Theorem 7. There exists a logic program $P$ having continuum-many stable models.

Proof. Take $P$ to be the program

\[ r(0) \leftarrow \]
\[ r(s(X)) \leftarrow r(X) \]
\[ p(X) \leftarrow \neg q(X) \]
\[ q(X) \leftarrow \neg p(x). \]

Let $R = \{ r(0), r(s(0)), r(s(s(0))), \ldots \}$. Any superset $X$ of $R$ such that for all ground terms $t$ exactly one of $p(t), q(t)$ is in $X$ is a stable model of $P$.  \( \square \)
Definition 9. Suppose $P$ is a general logic program, and $A \in B_P$ is a ground atom. We say $A$ is stably valid w.r.t. $P$ (denoted $P \rightarrow_{st} A$) if and only if $A$ is true in all stable models of $P$.

The following results are immediate corollaries of Theorem 4.

Theorem 8. Suppose $P$ is a general logic program. If $A \in T_{G_{I}, P, \gamma} \uparrow \omega$, then $P \rightarrow_{st} A$.

Proof. We need to show that if $M$ is a stable model of $P$, then $A \in M$. As $M \subseteq B_P$, $G(B_P, P) \subseteq G(M, P)$. Hence, $T_{G_{I}, B_P, \gamma} \uparrow \omega \subseteq T_{G_{I}, M, P} \uparrow \omega$. As $A \in T_{G_{I}, B_P, \gamma} \uparrow \omega$, $A \in T_{G_{I}, M, P} \uparrow \omega$. As $M$ is stable, $M = T_{G_{I}, M, P} \uparrow \omega$. Hence, $A \in M$. □

Intuitively, Theorem 8 gives us a sufficient condition for checking whether an atom $A$ is stably entailed by $P$. Likewise, Theorem 9 below gives us a sufficient condition for checking whether a negated atom is stably entailed by $P$.

Theorem 9. Suppose $P$ is a general logic program and $A \in B_P$. If $A \notin T_{G_{I}, P, \gamma} \uparrow \omega$, then $P \rightarrow_{st} \neg A$.

Proof. Let $M$ be any stable model of $P$. As $\emptyset \not\subseteq M$, $G(M, P) \subseteq G(\emptyset, P)$. Hence, $T_{G_{I}, M, P} \uparrow \omega \subseteq T_{G_{I}, \emptyset, P} \uparrow \omega$. As $M = T_{G_{I}, M, P} \uparrow \omega$ (M is stable), $M \subseteq T_{G_{I}, \emptyset, P} \uparrow \omega$. Thus, if $A \notin T_{G_{I}, \emptyset, P} \uparrow \omega$, $A \not\in M$. □

Unfortunately, the problem of stable validity of both ground atoms, and negated ground atoms may be highly undecidable as Corollary 2 below demonstrates.

Proposition 4 (Apt and Blair [1, Theorem 22]). For each $n > 0$, there is a program $P$ such that the standard model of $P$ constructed by the transfinite iteration procedure of Apt, Blair and Walker [2] is a $\Sigma^0_n$-complete subset of the Herbrand Base of $P$.

Corollary 2. For each $n > 0$, there is a logic program $P$ such that

1. the set $\mathcal{F}(P) = \{A \in B_P \mid P \rightarrow_{st} A\}$ is a $\Sigma^0_n$-complete subset of $B_P$;
2. the set $\mathcal{F}(P) = \{A \in B_P \mid P \rightarrow_{st} \neg A\}$ is a $\Pi^0_n$-complete subset of $B_P$.

The above corollary demonstrates that the stable model semantics can be highly intractable.
3. Autoepistemic translation of a logic program

In this section we investigate the relationship between supported models of a general logic program and the logic of an ideally introspective agent. We show that under the epistemic translation $ET$ (to be introduced below) of logic programs there is a one to one correspondence between the supported models and the so-called autoepistemic expansions of translations (cf. [24]).

Autoepistemic logic was proposed by Moore [24] as a formalism for a reasoning agent to be able to reflect upon her/his own knowledge. We observe that Gelfond [6] has also defined a translation of logic programs into autoepistemic theories, but as will shortly become apparent, our transformation is somewhat different and leads to some interesting connections between supported models of logic programs and expansions of autoepistemic theories.

We briefly review the basic results pertaining to autoepistemic logic, as proved in [18, 12, 19].

Let $L$ denote a propositional language whose logical symbols are the usual symbols of propositional logic$^1$. $Cn$ is the well-known Tarski consequence operation of the propositional logic, sometimes called tautological consequence. $L$ can be extended to a modal language $L_K$ by introducing a unary connective $K$. Every formula $\phi$ of $L$ is in $L_K$, and if $\psi$ is a formula of $L_K$, then $K\psi$ is a formula of $L_K$. Intuitively, if $\psi$ is a formula, then $K\psi$ may be read as "$\psi$ is known to be true". When discussing $L_K$, the consequence operation $Cn$ acts on the theories in the modal language as well, except that here every expression $K\phi$ is treated as an atom.

Given the language of modal logic $L_K$, a theory $T \subseteq L_K$ is called stable if it satisfies the following conditions:

- (St1) $T$ is closed under propositional consequence,
- (St2) $\phi \in T \Rightarrow K\phi \in T$,
- (St3) $\phi \not\in T \Rightarrow \neg K\phi \in T$.

Given a theory $I \subseteq L_K$ (think about $I$ as the initial assumptions of an agent), a theory $T \subseteq L_K$ is called an expansion of $I$ [24] if it satisfies the following condition:

$$T = Cn(I \cup \{K\phi: \phi \in T\} \cup \{\neg K\phi: \phi \not\in T\}).$$

(\*)

Hence, as it happens often both in logic programming and related topics, expansion is defined via a fixed point of an operator. Not every theory $I$ possesses an expansion and if one exists, it need not be unique. The operator whose fixed points are expansions is by no means monotone. One notices that every expansion of $I$ is a stable theory in $L_K$. In [18, 12] it was proved that every $I \subseteq L$ (that is without occurrence of $K$) possesses a unique expansion. This unique expansion, called below Exp$(I)$ possesses the property that Exp$(I) \cap I = Cn(I)$. Before showing how Exp$(T)$ is constructed when $T \subseteq L$, we explain notation. $L_K$ denotes the language of $L$ extended with the modal operator $K$ as explained earlier. Formulas of $L_K$ have an

\footnote{Throughout the rest of this section, we assume that logic programs are all propositional in nature.}
associated K-depth defined as follows: if \( \phi \) is in \( L \), then \( K \text{-depth}(\phi) = 0 \). K-depth(\( K\phi \)) = 1 + K-depth(\( \phi \)). K-depth(\( \neg \phi \)) = K-depth(\( \phi \)). K-depth(\( \phi \& \psi \)) = max\{K-depth(\( \phi \)), K-depth(\( \psi \))\}. \( L_{K,n} \) denotes the set of all formulas of \( L_K \) of K-depth at most \( n \). For \( T \subseteq L \), \( Exp(T) \) is constructed as follows:

\[
\begin{align*}
Exp_0(T) &= Cn(T) \\
Exp_{n+1}(T) &= L_{K,n+1} \cap Cn(Exp_n(T)) \cap \{K\phi \mid \phi \in Exp_n(T)\} \\
&\quad \cup \{\neg K\psi \mid \psi \in (L_{K,n} - Exp_n(T))\}
\end{align*}
\]

\[
Exp(T) = \bigcup_{i=0}^{\infty} Exp_n(T).
\]

Note that \( Exp(T) \) is defined even when \( T \not\subseteq L \). However, in such cases, \( Exp(T) \) may not be an expansion of \( T \). Marek and Truszczynski and independently Konolige [18, 12] show that for all \( T \subseteq L \), \( Exp(T) \) is an expansion of \( T \). In order to investigate expansions we need a criterion from [19] which establishes both a normal form for expansions and necessary and sufficient conditions for existence of expansions. To this end, notice first that theories with identical propositional consequences have precisely the same expansions (this follows easily from the definition of expansion). Consequently, every theory \( I \) can be represented by a collection of implications of form.

\[
K\phi_1 \& \cdots \& K\phi_n \& \neg K\psi_1 \& \cdots \& \neg K\psi_n \Rightarrow \omega
\]  (**

where \( \omega \in L \), that is \( \omega \) contains no occurrences of the modal operator \( K \).

Hence assume that \( I \) consists of implications of form (**). To simplify notation we write such formulae as \( A \Rightarrow \omega \) where

\[
A = K\phi_1 \& \cdots \& K\phi_n \& \neg K\psi_1 \& \cdots \& \neg K\psi_n
\]

is called the epistemic justification of \( \omega \) and \( \omega \) is called the objective part of the implication.

Hence, let \( I = \{\varphi_i : 1 \leq i \leq k\} \), \( \varphi_i = A_i \Rightarrow \omega_i \). We have (see [19]), the following theorem.

**Theorem 10.** (1) Every expansion of \( I \) is of the form \( Exp(\{\omega_i : i \in J\}) \) for a suitably chosen \( J \subseteq \{1, \ldots, k\} \).

(2) If \( T \) is an expansion of \( I \), then for some set \( J \subseteq \{1, \ldots, k\} \), \( T = Exp(\{\omega_i : i \in J\}) \) and \( I \subseteq T \) for all \( i \in J \), \( A_i \subseteq T \).

(3) If \( I \subseteq T = Exp(\{\omega_i : i \in J\}) \) and for all \( i \in J \), \( A_i \subseteq T \), then \( T \) is an expansion of \( I \).

A word of caution is appropriate here. In general, a given theory \( T \) does not uniquely determine a set \( J \) such that \( T = Exp(\{\omega_i : i \in J\}) \). \( T \subseteq I \) is an expansion of \( I \) if and only if there exists at least one set \( J \) such that \( T = Exp(\{\omega_i : i \in J\}) \) and for all \( i \in J \), \( E_i \subseteq T \).
Before we introduce the translation and prove the results connecting supported models and explanations of translation, we need to introduce one more property of stable sets (and hence expansions as well).

**Theorem 11.** (cf. [19]). Let the formula \( \phi \) have the property that for every atom \( a \), every occurrence of \( a \) appears within the scope of modal operator \( K \). Then, for every stable theory \( T \), \( \phi \in T \) or \( \neg \phi \in T \).

We introduce now the notion of epistemic translation of a logic program. This translation is different from that of Gelfond (cf. [6]) and in fact relates to the notion of supported model and not to that of stable model.

**Definition 10.** Suppose \( P \) is a logic program, and \( A \in B_p \). Denote by \( \mathcal{H}(P, A) \) the set of clauses in \( P \) having \( A \) as the head. Thus,

\[
P = \bigcup_{A \in B_p} \mathcal{H}(P, A).
\]

Let \( \mathcal{H}(P, A) = \{ C_1, C_2, \ldots \} \), where each \( C_i \) is of the form

\[
A \leftarrow B_i^1 & \cdots & B_i^m & \neg D_i^1 & \cdots & \neg D_i^n.
\]

Then the epistemic translation \( \mathcal{E}(P, A) \) of \( P \) with respect to \( A \) is the set of clauses

\[
\{ KB_i^1 & \cdots & KB_i^m & \neg KD_i^1 & \cdots & \neg KD_i^n \rightarrow A : i = 1, 2, \ldots \}
\]

together with the (possibly infinitary) sentence

\[
\bigwedge_{i \geq 1} (\neg(KB_i^1 & \cdots & KB_i^m & \neg KD_i^1 & \cdots & \neg KD_i^n)) \rightarrow \neg A.
\]

The **epistemic translation** \( \mathcal{E}(P) \) of the logic program \( P \) is

\[
\mathcal{E}(P) = \bigcup_{A \in B_p} \mathcal{E}(P, A).
\]

As usual, in the above translation, we replace all occurrences of formulas of the form \( \neg \neg F \) by \( F \). The potentially infinitary clause of the form \( A \rightarrow \neg p \) in \( \mathcal{E}(P) \) above allows us to derive a negated atom when no epistemic justification is available. Although this formula may be infinitary, it poses no problem as we may assume that \( \text{Exp}(S) \) is closed under infinitary conjunctions.

**Example 9.** Suppose \( P \) is the logic program below,

\[
p \leftarrow q
\]
\[
p \leftarrow r, \neg q
\]
\[
q \leftarrow q
\]
\[
r \leftarrow
\]

\[
\]
then $ET(P)$ is

\[ Kq \rightarrow p \]
\[ Kr \& \neg Kq \rightarrow p \]
\[ \neg (Kq \lor (Kr \& \neg Kq)) \rightarrow \neg p \]
\[ Kq \rightarrow q \]
\[ \neg Kq \rightarrow \neg q \]
\[ r. \]

**Example 10.** If $P$ is the program

\[ p \leftarrow \neg p \]

$ET(P)$ is

\[ \neg Kp \rightarrow p \]
\[ Kp \rightarrow \neg p. \]

**Theorem 12.** Suppose $P$ is a general logic program and $M$ is a supported model of $P$. Then $\text{Exp}(Th(M))$ is an expansion of $ET(P)$.

**Proof.** Notice first that the formulas of the translation have the form

\[ A \Rightarrow p \]

or

\[ A \Rightarrow \neg p \]

for epistemic formulas $A$. Moreover, in the latter case, the formula $E$ unique (up to reordering of the individual conjuncts).

Since $Th(M) = Cn(M \cup \{\neg p : p \in B_p - M\})$, we will have a proof of the theorem if the following three items are proved:

(a) $ET(P) \subseteq \text{Exp}(Th(M))$

(b) Whenever $p \in M$ there exists a formula $A \Rightarrow p$ in $\text{EY}(P)$ such that $A$ belongs to $\text{Exp}(Th(M))$.

(c) Whenever $p \in B_p - M$, the unique $A$ such that $A \Rightarrow \neg p \in ET(P)$ belongs to $\text{Exp}(Th(M))$.

(a) **Case 1:** $\varphi = A \Rightarrow p$.

(1) If $p \in M$ then $p \in \text{Exp}(Th(M))$, hence $\varphi \in \text{Exp}(Th(M))$.

(II) If $p \not\in M$ then, since $M$ is a supported model all the bodies of clauses of $P$ having $p$ as the head must be false in $M$. It is easy to check that the epistemic translation of such a body does not belong to $\text{Exp}(Th(M))$ and consequently, using
Theorem 11 we find that its negation does belong to \( \text{Exp}(\text{Th}(M)) \). Since the latter is closed under consequence, \( A \Rightarrow p \in \text{Exp}(\text{Th}(M)) \).

Case 2: \( \phi = A \Rightarrow \neg p \).

(1) If \( p \notin M \) then \( \neg p \in \text{Th}(M) \), hence \( A \Rightarrow \neg p \in \text{Exp}(\text{Th}(M)) \).

(II) If \( p \in M \) we reason similarly to case 1(II). Since \( M \) is a supported model, at least one of the epistemic justifications for \( p \) belongs to \( \text{Exp}(\text{Th}(M)) \). Since the unique epistemic justification of \( \neg p \) is the conjugation of negations of epistemic justifications for \( p \), it is easy to see that the negation of the epistemic justification of \( \neg p \) belongs to \( \text{Exp}(\text{Th}(M)) \). Hence \( A \Rightarrow \neg p \in \text{Exp}(\text{Th}(M)) \).

(b) Let \( p \in M \). We need to prove that \( p \) possesses at least one epistemic justification in \( \text{Exp}(\text{Th}(M)) \). This, however, follows, directly from the fact that \( M \) is a supported model of \( P \).

(c) Let \( p \notin M \). Since \( M \) is a supported model, all the bodies of clauses of \( P \) having \( p \) as the head must be false in \( M \). Hence their epistemic translations do not belong to \( \text{Exp}(\text{Th}(M)) \). Consequently negations for all the epistemic justifications of \( p \) do belong to \( \text{Exp}(\text{Th}(M)) \). Hence their conjunction (which is the epistemic justification of \( \neg p \) belongs to \( \text{Exp}(\text{Th}(M)) \)).

This completes the proof of the theorem. \( \square \)

Theorem 13. Suppose \( P \) is a general logic program. If \( T \) is an expansion of \( \text{ET}(P) \), then \( T \) is of the form \( \text{Exp}(\text{Th}(M)) \) for a unique supported Herbrand model \( M \) of \( P \).

Proof. It is quite clear what \( M \) should be, namely let

\[
M = \{ p \in B_p : p \in T \}.
\]

(This is the same as saying that \( M = B_p \cap T \), that is \( M \) is the set of all ground atoms occurring in \( T \). We need to prove three items:

(a) \( M \) is a model of \( P \),
(b) \( M \) is supported,
(c) \( T = \text{Exp}(\text{Th}(M)) \).

(a) If \( C = A \Rightarrow B_1 \& \cdots \& B_m \& \neg D_1 \& \cdots \& \neg D_m \) is a clause of \( P \), then, if \( A \in M \), then \( M \) models \( C \). If \( A \notin M \), then since \( \text{ET}(P) \subseteq T \), \( A \) cannot possess an epistemic justification in \( T \). Consider a justification

\[
E = KB_1 \& \cdots \& KB_m \& \neg KD_1 \& \cdots \& \neg KD_n.
\]

Since \( E \) does not belong to \( T \),

\[

\neg KB_1 \lor \cdots \lor \neg KB_m \lor KD_1 \lor \cdots \lor KD_n
\]

does belong to \( T \). Since \( T \) is stable and consistent

\[

B_1 \notin T \lor \cdots \lor B_m \notin T \lor D_1 \in T \lor \cdots \lor D_n \in T.
\]

This means, according to the definition of \( M \), that \( M \) does not satisfy \( B_1 \& \cdots \& B_m \& \neg D_1 \& \cdots \& \neg D_m \). Hence \( M \models C \).
(b) We need to show that, whenever \( A \in M \), there is a clause \( A \leftarrow B_1 \land \cdots \land B_m \land \neg D_1 \land \cdots \land \neg D_m \) of \( P \), such that \( M \models \exists_1 \land \cdots \land B_m \land \neg D_1 \land \cdots \land \neg D_m \) of \( P \).

Otherwise, for every epistemic justification \( E \) of \( A \), \( E \not\in T \), but then the conjunction of the negations of justifications does belong to \( T \). Since \( ET(P) \subseteq T \), and the justification of \( \neg A \) is in \( T \), \( \neg A \in T \). This, however, is a contradiction since for all \( A \in B_p \), \( A \in M \) if and only if \( A \in T \).

(c) Finally, we show that \( T = \text{Exp}(\text{Th}(M)) \). Let us notice that the reasoning of point (b) shows that for every atom \( A \in B_p \), \( A \in T \) or \( \neg A \in T \). This, in particular, implies that \( T \cap L \) is a complete theory. Now, \( \text{Th}(M) \) is also a complete theory and \( T \cap L \) and \( \text{Th}(M) \) contain precisely the same atoms. Hence \( \text{Th}(M) = T \cap L \) and since \( T = \text{Exp}(T \cap L) \), the proof is complete. \( \square \)

**Corollary 3.** \( M \) is a supported model of \( P \) iff \( \text{Exp}(\text{Th}(M)) \) is an expansion of \( ET(P) \).

In Section 2 we noticed (cf. Theorem 5) a connection between stable models and extensions of default translation of logic programs. Our results of this section provide us with yet another connection with default logic, this time, however, with different structures.

In [21] the connection between autoepistemic logic and default logic was studied in detail and the class of objects in default logic corresponding to autoepistemic expansions was fully identified. These objects, called in [21] weak extensions are defined as follows.

**Definition 11.** Let \((D, W)\) be a default theory. A theory \( T \subseteq L \) is called a weak extension of \((D, W)\) if and only if \( T \) satisfies the following fixed point equations:

\[
T = Cn(W \cup \{ \forall \beta \in p(d) \neg (\beta \in T) \}).
\]

Extensions of a default theory are weak extensions but the converse implication does not need to hold.

By Konolige’s translation of a default theory \((D, W)\) we mean a theory

\[
T_{D,W} = W \cup \{ \forall \beta \in p(d) \neg (\beta \in T) \}. \]

Let us quote now two results of [21] which are immediately seen to be relevant to our considerations.

**Theorem 14.** A theory \( S \subseteq L \), closed under propositional consequence, is a weak extension of a default theory \((D, W)\) if and only if \( \text{Exp}(S) \) is an expansion of the autoepistemic theory \( T_{D,W} \).

**Theorem 15.** For every theory \( I \subseteq L_K \) there exists a default theory \((D, W)\) such that the expansions of \( I \) are precisely the same as those of \( T_{D,W} \). Hence the expressive power of autoepistemic logic is precisely the same as that of default logic but with weak extensions.
Let us indicate how Theorem 14 is proved (for full details, see [21]). If $I \subseteq L_k$, then there exists a theory $I'$ such that every $\phi \in I'$ is of $K$-rank at most one and $I$ and $I'$ have precisely the same expansions. Once this fact (proved by reduction of the $K$-rank of formulas) is established it is easy to see that a formula of rank at most one is equivalent to the conjunction of a finite number of Konolige's translations of defaults. Consequently, Theorem 14 is applicable.

Combining these results and Theorem 13 together, we get the following.

**Corollary 4.** For every general logic program $P$ there exists a default theory $(D, W)$ such that for every subset $M$ of $B_P$, $M$ is a supported model for $P$ if and only if $Th(M)$ is a weak extension of $(D, W)$.

To summarize, the results of this section show that whereas stable models are intimately related to extensions of default theories, supported models are associated (by a different translation) with weak extensions of default theories.

4. Conclusions

It is always surprising when constructions from seemingly unrelated domains turn out to be closely connected. Our results as well as other results mentioned in Section 1 point to the existence of the same basic principles behind various modes of reasoning considered by the artificial intelligence community and by various interpretations of negation considered by the logic programming community. We can only hope that a single, unifying, approach eventually emerges.

The primary aim of this paper is to clarify the various relationships between stable models of logic programs, supported models of logic programs, the default semantics for logic programs and the autoepistemic semantics for logic programs. The first two of these essentially claim that the meaning of a program is just the set of models of the program possessing certain properties. On the other hand, the last two essentially claim that the program's meaning is exactly that of an (appropriately defined) translation of the program into a different logic (viz. default logic and autoepistemic logic). In this paper, we have studied

1. the connection between stable and supported models (Section 2),
2. the relationship between supported models and default logics (Corollary 4),
3. the relationship between supported models of a program and expansions of the program's autoepistemic translation (Theorems 12 and 13).

Thus, these results demonstrate the intricate, yet intimate, relationship between differing formalisms for negation in logic programming. We strongly believe that a through study of the interrelationships between varying formalisms for treating negation in logic programming and in AI are necessary, as there are far too many such formalisms today.
Acknowledgment

We are grateful to an anonymous referee for his/her insightful comments which led us to strengthen many of the results of the paper. We thank Howard Blair for comments which helped simplify and improve many of the proofs as well as the presentation. First author’s work was partly supported by National Science Foundation Grant RII 8610671 and the Commonwealth of Kentucky EPSCoR program.

References


