An Abductive Framework for General Logic Programs and other Nonmonotonic Systems

Gerhard Brewka
GMD, Postfach 12 40
5205 Sankt Augustin, Germany

Kurt Konolige
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025

Abstract

We present an abductive semantics for general propositional logic programs which defines the meaning of a logic program in terms of its extensions. This approach extends the stable model semantics for normal logic programs in a natural way. The new semantics is equivalent to stable semantics for a logic program *P* whenever *P* is normal and has a stable model. The abductive semantics can also be applied to generalize default logic and autoepistemic logic in a like manner. Our approach is based on an idea recently proposed by Konolige for causal reasoning. Instead of maximizing the set of hypotheses alone we maximize the union of the hypotheses, along with possible hypotheses that are excused or refuted by the theory.

1 Background and Motivation

In this paper we investigate the relationship between abduction, logic programming, and other nonmonotonic formalisms.[1] This investigation is interesting for several reasons. Firstly, abduction as a form of nonmonotonic reasoning has gained a lot of interest in recent years, and exploring the relationship between different forms of nonmonotonic reasoning is of interest in itself. Secondly, as we will show in this paper, it is possible to define a simple and elegant extension of Gelfond and Lifschitz's stable model semantics [9] based on abduction. This new abductive semantics has the following properties:

- The semantics is equivalent to stable model semantics for programs which possess at least one stable model.
- A program *P* has a defined meaning unless *P* considered as a set of inference rules is inconsistent. In particular, normal logic programs without stable models are not meaningless.
- The abductive semantics imitates the well-founded model in not assigning a truthvalue to propositions whose assertion is self-contradictory.
- The semantics is, without further modification, applicable to logic programs that contain classical negation and so-called epistemic disjunction [8].
- The semantics can be applied to other consistency-based nonmonotonic formalisms, including default logic and autoepistemic logic.

We consider all of these properties as highly desirable. Stable model semantics is currently the most widely accepted semantics for logic programs which have a stable model. We therefore believe that an extension of stable model semantics should preserve the meaning of those programs. On the other hand, many authors consider it a severe weakness of stable model semantics that not all normal logic programs have stable models; by contrast, the well-founded semantics always exists. Our semantics overcomes this weakness, in the same manner as the well-founded semantics, by allowing a truthvalue gap for self-contradictory propositions. At the same time, it does not suffer from the weakness of well-founded semantics, the "floating conclusions" problem.[2] Finally, there has been a great amount of recent work trying to extend the expressiveness of normal logic programs by, among other things, adding classical negation and "epistemic" disjunction. It turns out to be a non-trivial task to adapt existing semantics to more general logic programs. It is therefore clearly an advantage if a simple semantics for normal programs can directly be applied to theses generalizations.

Abduction, informally, is the generation of explanations for a given fact *p*. Given a background theory *T* and a set of possible hypotheses or abducibles *H,* an explanation for *p* is a subset *H'* of *H* such that $H' \cup T$ is consistent and *p* is provable from $H' \cup T$. Usually, there is a further acceptability criterion that distinguishes preferred explanations. In our approach we will consider negated atoms as hypotheses, a logic program (viewed as a set of inference rules) as the background theory. Moreover, we introduce a simple criterion defining the acceptable explanations or, in our terminology, extension bases. We consider a proposition *q* derivable 'om

*For simplicity we consider only finite propositional logic programs, that is programs with finite Herbrand base, in this preliminary report. All definitions also apply to the genera] case.

[2] Floating conclusions are conclusions that are intuitively justified by case analysis yet underivable in well-founded semantics. The standard example is $a \leftarrow \sim b$; $b \leftarrow \sim a$; $c \leftarrow a$; $c \leftarrow b$. Well-founded semantics does not conclude c.

a logic program if it is derivable from all of its extension bases.

Using abductive frameworks to define a semantics for logic programs is not a new idea. Eshghi and Kowalski [4] were the first to investigate logic programs, in particular negation as failure, in terms of abduction. More recently, Kakas and Mancarella [13] and Dung [3] have continued this line of research. We will in the rest of this section review these earlier approaches and discuss why we do not consider them entirely satisfactory.

Eshghi and Kowalski show that negation as failure in normal logic programs can be viewed as a special case of abduction. Their analysis is based on abductive frameworks of the form $\langle T, I, A\rangle$ where T is a set of definite clauses, I a set of integrity constraints, and A a set of abducible predicates. Atomic ground formulas built from the symbols in A are called abducibles. A set of abducibles Δ is an abductive solution for q iff $T \cup \Delta \vdash q$ and $T \cup \Delta$ satisfies I.

Since the authors restrict T to definite clauses and the set of abducibles to atomic formulas they have to eliminate negation signs from logic programs to handle negation as failure in their framework. Given a normal logic program P they introduce for each predicate symbol p in P a new predicate symbol p^*. P is then transformed to a program P^* by replacing each negative occurrence of a predicate symbol p in the body of a rule by a positive occurrence of p^*. Additionally, integrity constraints of the form

$$\leftarrow p^*(x) \wedge p(x)$$

are used to make sure that not both $p^*(t)$ and $p(t)$ can be true at the same time. Additionally, there are metalevel constraints of the form

$$Demo(T \cup \Delta, p^*(t)) \vee Demo(T \cup \Delta, p(t))$$

Such a disjunctive integrity constraint is satisfied iff $p(t)$ or $p^*(t)$ is provable from $T \cup \Delta$. Eshghi and Kowalski show that for every stable model of a program P there is a corresponding abductive solution for the transformed program P^* and vice versa. This one-to-one correspondence to stable models shows that the new abductive semantics does not give meaning to programs without stable models and thus inherits the weakness of stable model semantics: the existence for abductive solutions for normal programs is not guaranteed. Consider the program P_2

$$p \leftarrow \sim p$$

and its transform

$$p \leftarrow p^*$$

The introduction of the metalevel constraint leads to the non-existence of an abductive solution.

Kakas and Mancarella use a similar abductive framework as Eshghi and Kowalski to define a generalization of stable models. In their paper an abductive framework is a triple $\langle P, A, I\rangle$ where P is a normal logic program, A a set of abducible predicates, and I a set of integrity constraints. Contrary to Eshghi/Kowalski they directly use the notion of a stable model in their definitions. A pre-general stable model of $\langle P, A, I\rangle$ is a stable model of $P \cup \{p \leftarrow \mid p \in \Delta\}$, where Δ is an arbitrary set of abducibles. A general stable model is a pre-general stable model that implies all integrity constraints in I.

The authors then show how negation as failure can be treated through abduction. The approach is similar to Eshghi/Kowalski's but somewhat simpler: the metalevel constraints involving the Demo predicate are replaced by integrity constraints of the form

$$p(x) \vee p^*(x)$$

Obviously, the stable models of a normal program P are exactly the general stable models of the abductive framework $\langle P, \emptyset, \emptyset\rangle$. From this it is immediate that the existence of general stable models is not guaranteed. General stable models thus do not solve the problem of nonexistence of stable models for normal programs.

Dung's abductive frameworks [3] are equivalent to Esghi/Kowalski's, that is he requires the programs in frameworks to consist of definite clauses. To be able to handle normal programs he replaces predicate symbols p in negated literals by new symbols p^*, as in Eshghi/Kowalski's approach. He also uses integrity constraints of the form

$$\leftarrow p^*(x) \wedge p(x)$$

but no constraints corresponding to the metalevel constraints involving the predicate *Demo*. Atoms built from the new symbols become abducibles.

Dung calls $S = P \cup H$ a scenario of the abductive framework $\langle P, A, I\rangle$ if H is a subset of the abducible atoms such that $P \cup H \cup I$ is consistent. Let $inout(S)$ denote the set of ground atoms provable from a scenario S. A set of abducible atoms E is a P-evidence for an atom p iff $P \cup E \vdash p$. An abducible $p^*(t)$ is S-acceptable iff for every P-evidence E of $p(t)$, $E \cup inout(S) \cup I$ is inconsistent. A scenario S is admissible if every abducible atom in S is S-acceptable. An admissible scenario is complete iff every abducible atom that is also S-acceptable is contained in S.

The complete scenarios define the semantics of a normal logic program. Dung was able to show that the set of complete scenarios forms a semi-lattice with respect to set inclusion. The least complete scenario is equivalent to the well-founded model. Stable models correspond to maximal complete scenarios. However, not every maximal complete scenario represents a stable model. Consider the program P_3:

$$a \leftarrow \sim b$$
$$b \leftarrow \sim a$$
$$c \leftarrow \sim c, b$$

The transform of the program, P^*_3, is:

$$a \leftarrow b^*$$
$$b \leftarrow a^*$$
$$c \leftarrow c^*, b$$

Additionally we have, for $x \in \{a, b, c\}$ the integrity constraints

$$\leftarrow x, x^*$$

We get two maximal complete scenarios, namely

$S_1 = P^*_3 \cup \{b^*, c^*\}$, and
$S_2 = P^*_3 \cup \{a^*\}$

S_1 corresponds to the single stable model $\{a\}$ of P_3. 52, although maximal, does not correspond to a stable model. This raises the question what the "interesting" ones among the complete scenarios are. Dung argues that two views are reasonable: a skeptical view which considers the least complete scenario only, and a credulous view which considers all maximal complete scenarios. In examples like the one just discussed there seems to be no reason to dispense with stable semantics, unless one adheres to the skeptical view. In a sense, Dung's credulous view seems to move too far away from stable semantics, whereas the two abductive approaches mentioned earlier stick with it too closely.

The abductive framework we present in this paper is distinct from this earlier work in the following respects:

1. We do not restrict the abducibles to atoms. This has the advantage that we can operate on the original programs directly and do not have to use any kind of transformation of the programs. Moreover, this makes the use of integrity constraints unnecessary.
2. The above approaches treat program rules as clauses and need some implicit device to obtain the directedness of rules. We consider the rules of a program as a set of inference rules, not as clauses.
3. We apply a new simple maximality criterion that guarantees that the right sets of abducibles are chosen. This criterion models the intuition that undefinedness should be minimized.
4. Our framework is simpler than the above approaches and can, unlike them, be applied to logic programs with classical negation and epistemic disjunction, as well as other consistency-based nonmonotonic formalisms.

The rest of the paper is organized as follows: in Section 2 we introduce our abductive framework and show how it can be used to formalize normal logic programs. In Section 3 we treat general logic programs with classical negation and epistemic disjunction in the heads. Section 4 applies our abductive method to default logic, Section 5 applies it to autoepistemic logic.

2 The abductive framework

In this section we introduce our abductive semantics for normal logic programs. We define the notion of an extension for a logic program. This terminology reflects the similarity to other work in nonmonotonic reasoning, in particular default logic.

By $Lit^{\sim}$ we mean the set of literals, that is, any sentence of the form a or $\sim a$, where a is an atom.

Definition 1 *A normal logic program P is a set of rules of the form*

$$a \leftarrow b_1, \ldots, b_n$$

where a is an atom and the b_i are in $Lit^{\sim}$.

We use $NEG(P)$ to denote the set of negative literals relevant to a logic program P, i.e. $NEG(P) = \{\sim a \in Lit^{\sim} \mid a \text{ is a subformula of } P\}$.

The notion of consistency plays a predominant role in abduction. We therefore have to define its meaning in the context of a logic program:

Definition 2 *Let $L \subseteq NEG(P)$ be a set of negative literals, P a normal logic program. The closure of L under P, $C_P(L)$, is the smallest set such that*

1. $L \subseteq C_P(L)$,
2. *if $a \leftarrow b_1, \ldots, b_n \in P$ and $b_1, \ldots, b_n \in C_P(L)$ then $a \in C_P(L)$.*

Definition 3 *Let $L \subseteq NEG(P)$ be a set of negative literals, P a normal logic program. L is P-consistent iff $C_P(L)$ is consistent.*

Definition 4 *Let P be a normal logic program. P is consistent iff $\emptyset$ is P-consistent.*

Similar to the earlier abductive treatments of logic programs we model negation as failure abductively. However, since we do not require abducibles to be atoms, we do not need to transform programs but can directly use $NEG(P)$ as the set of abducibles. The main difficulty is that we cannot consider all maximally consistent subsets of $NEG(P)$ as representing the intended meaning of a program. This simple approach fails to capture the intuitions underlying logic programming as can be demonstrated by our program P_1.

$$\begin{array}{l} a \leftarrow \sim b \\ b \leftarrow \sim c \end{array} \qquad (1)$$

The standard reading of this program is that b is derivable and a underivable. However, there exists a maximal P-consistent subset of $NEG(P)$, namely $H_1 = \{\sim b\}$ that fails to capture this intuition. The closure of H_1 under P_1 contains a but not b. This clearly violates all of the standard semantics for logic programs, and H_1 should not be considered an acceptable set of hypotheses.

What then are the acceptable sets of hypotheses, or — in our terminology — the extension bases, that can be used to define the meaning of a logic program? It turns out that an idea used in [15] for reasoning about simple causal systems can be applied to solve this problem. The reader may observe that among the two maximally consistent subsets of $NEG(P)$ in the above example, the wanted subset, $H_2 = \{\sim c, \sim a\}$ allows us to derive b, that is refutes the remaining hypothesis in $NEG(P)$. The unintended subset, on the other hand, does not refute the hypothesis $\sim c$. It turns out that, to capture the intuition behind logic programs, we have to maximize not just the set of hypotheses, but also the set of refuted hypotheses. This leads to the following definitions:

Definition 5 *Let P be a logic program, $H \subseteq NEG(P)$, and C a P-closure of H. The P-cover of C, $COV_P(C)$, is the set*

$$H \cup \{\sim a \in NEG(P) \mid a \in C\}$$

Definition 6 *Let P be a logic program, $H \subseteq NEG(P)$, and C a P-closure of H. C is an extension of P iff*

1. *C is consistent,*
2. *there is no $H' \subseteq NEG(P)$ with consistent closure C' such that $COV_P(C) \subset COV_P(C')$.*

If C is an extension, H is called an extension base of P.

For normal logic programs there is only one closure of any set $H \subseteq NEG(P)$, so extension bases and extensions are isomorphic. For disjunctive logic programs there may be several closures, so we give the more general definition here.

It is obvious that our example gives only rise to one extension, as intended, since $COV_P(C_P(H_1)) = \{\sim a, \sim b\} \subset \{\sim a, \sim b, \sim c\} = COV_P(C_P(H_2))$. This extension coincides with the unique stable model of the program. This is not incidental. We can show that our semantics and stable model semantics coincide in cases where stable models exist.

Proposition 1 *Let P be a normal logic program for which a stable model exists. M is a stable model of P iff it is an extension of P.*

Obviously, the existence of extensions for normal logic programs is guaranteed since every such program must be consistent. This is achieved because, contrary to stable models, extensions do not have to contain either a or $\sim a$ for every atom a. Consider the following example:

$$\begin{array}{l} a \leftarrow \sim a \\ b \leftarrow \sim c \end{array} \qquad (2)$$

This program has no stable model, yet it has an extension generated by the extension base {~c}

Extensions thus have the well-known property of the well-founded semantics [20] in allowing truthvalue gaps, that is, neither a nor ~a is in the extension of the above program. But like stable models, extensions do not suffer from the problem of "floating conclusions." In the following example, the well-founded semantics does not conclude p, while p is part of every extension.

$$\begin{array}{l} a \leftarrow \sim b \\ b \leftarrow \sim a \\ p \leftarrow a \\ p \leftarrow b \end{array} \qquad (3)\ (3)'$$

We discussed several related abductive approaches and their weaknesses in the introduction already. Another approach which can, in a sense, be considered dual to ours is that of Inoue [12]. He distinguishes between a set of necessary rules T and a set of hypothetical rules H. A model in his system is a stable model of T U H', where H' is a maximal subset of// such that a stable model exists. Inoue thus dispenses with some of the hypothetical rules if necessary and sticks with stable semantics otherwise, whereas we leave the truth values of atomic propositions undecided if necessary.

A further - non-abductive - approach of interest here is that of Sacca and Zaniolo [19]. It is based on the notion of P-stable models. Such models are partial, i.e., not all atomic propositions need to get a truth value. We cannot give the exact definition of this notion here, but want to stress that it is intended to capture "the three key properties considered highly desirable by researchers in this area". According to the authors these properties are

- consistency: no proposition should be true and false at the same time in a model,
- justifiability: every positive conclusion should be demonstratable using the directed rules of the program,
- minimal undefinedness: the number of undefined facts should be reduced as much as possible.

As Sacca and Zaniolo show P-stable models do not guarantee minimal undefinedness: P-stable models can be proper subsets of other P-stable models. The authors therefore propose "that the minimal undefinedness principle should be enforced by restricting our attention to the class of P-stable models that are maximal." Unfortunately, maximality is insufficient to guarantee minimal-undefinedness as can be demonstrated by the program P_3 used earlier in the introduction:

$$\begin{array}{l} a \leftarrow \sim b \\ b \leftarrow \sim a \\ c \leftarrow \sim c, b \end{array}$$

Besides the uncontroversial maximal P-stable model S_1 = {a,~b,~c} we also obtain the P-stable model S_2 = {~a,b}. S_2 clearly is maximal, as the addition of -c leads to inconsistency whereas the addition of c cannot be justified by any of the available rules. Nevertheless, S2 does not minimize undefinedness: c has no truthvalue in S_2 but is false in S_1. In our framework the single extension corresponds to S_1. There is no extension corresponding to S_2

3 General Logic Programs

In this section we will consider general logic programs, that is, programs where negations and disjunctions may appear in the head of a rule, and in addition there is a classical negation operator. It turns out that all we have to do is slightly generalize the notion of P-closure.

As in [7], we use two negation operators, - and ~. -i is classical negation, while ~ is negation as failure to prove. By Lit we mean the set of classical literals, and by Lit^+, the set of literals of the form / or ~/, where / is in Lit.

Definition 7 A general logic program P is a set of rules of the form

$$c_1 | \cdots | c_m \leftarrow b_1, \ldots, b_n$$

where all the literals Ci and bj are in Lit^+.

Note that this definition generalizes [7] slightly by allowing negation-as-failure literals in the head of a clause. As usual, by NEG(P) we mean the set of negative literals {-/ I either / or ~/ is a literal of P}.

We interpret disjunction in the head in the same way as Gelfond and Lifschitz, that is, it is epistemtc disjunction. The condition on the P-closure is that at least one of the literals in the head must appear in the closure.

Definition 8 Let L be a set of literals from NEG(P). C is a P-closure of L if it is a smallest set such that

1. $L \subseteq C$,
2. *if l and $\sim l$ or a and $\neg a$ are in C, then $C = Lit^+$,*
3. *if $c_1 | \cdots | c_m \leftarrow b_1, \ldots, b_n \in P$, and $b_1, \ldots, b_n \in C$ then for at least one of the c_i, $c_i \in C$.*

Note that there is not necessarily a unique P-closure of a program; however, if it has a consistent P-closure, then all its P-closures are consistent. The unique inconsistent P-closure is the set of all literals. We say L is P-consistent if it has a consistent P-closure. All other definitions from the last section can now be applied without further changes to general logic programs. Here is an example involving disjunctions in the head of a rule:

$$a|b \leftarrow \sim c \tag{4}$$

We obtain two extensions, $E_1 = \{a, \sim c, \sim b\}$ and $E_2 = \{b, \sim c, \sim a\}$. Both have the cover NEG(P). Note that $\{\sim a, \sim b\}$ is not an extension since its cover does not contain ~c.

The following slight modification of the last example involves a negation in the head:

$$a|\sim b \leftarrow \sim c \tag{5}$$

Now there is only one extension, namely {~a,~6,~c}.

General programs can be used to implement many of the standard default reasoning examples. Here is a bird example:

$$\begin{array}{l} fly \leftarrow bird, \sim ab_1 \\ \neg fly \leftarrow penguin, \sim ab_2 \\ ab_1 \leftarrow \sim ab_2 \\ penguin \\ bird \end{array} \tag{6}$$

We obtain one extension from the extension base $\{\sim ab_2\}$ (we omit the irrelevant literals $\sim fly$, $\sim\neg fly$, etc.). The P-cover of this extension is NEG(P). Note that the set of abducibles $\{\sim ab_1\}$ is not an extension base as its P-cover does not contain $\sim ab_2$. As intended the more specific rule gets priority.

As with normal logic programs, extensions of general logic programs correspond to the answer sets of Gelfond and Lifschitz, when the latter exist.

Proposition 2 Let P be a general logic program with no negation-as-failure literals in the head and suppose it has at least one consistent answer set. Let Cfacts(L) denote the set of classical literals (without weak negation) tn L. Cfacts is a bijective mapping from the set of extensions of P to the set of answer sets of P.

If there are multiple extensions, then they are always disjoint.

Proposition 3 If F and F[f] are two extensions, then F U F' is inconsistent.

It should be noted that the introduction of negation in the heads of rules, of either type, leads to a situation where the existence of extensions can no longer be guaranteed for all programs. The reason is that the programs themselves may become inconsistent. Recall that a program P is inconsistent if the (single) P-closure of the empty set is inconsistent. It is not difficult to prove the following lemma:[3]

[3]The proof is easy since, as mentioned in the beginning, we only consider propositional programs with finite Herbrand base in this report. However, there are cases in which, for infinite NEG(P), there is an infinite ascending chain $COV_P(C_P(H_1)) \subset COV_P(C_P(H_2)) \subset \ldots$, and hence no extensions.

Lemma 1 Let P be a logic program. P has an extension iff P is consistent.

Obviously, all programs where the negation sign (and $\bot$) does not appear in the head of a rule are consistent and therefore have at least one extension. This includes normal logic programs.

Sometimes it is convenient and useful to restrict the set of abducibles to a proper subset of NEG(P). A logic program then consists of a set of rules P together with a set HYP of negated literals representing the atoms assumed to be false by default. The P-cover of a closure of a set of abducibles $H \subseteq HYP$ is, as before, the set of abducibles which are either assumed or refuted.

For instance, in the bird example we get the desired results if we restrict the abducibles to HYP = $\{\sim ab_1, \sim ab_2\}$. Again we get one extension containing -fly. The extension base is $\{\sim ab_2\}$. Note that the cover of this extension is HYP, whereas the cover of the closure of $H_2 = \{\sim ab_1\}$ does not contain $\sim ab_2$ therefore is not an extension base.

There is also no reason why we should not sometimes let positive information, that is unnegated atoms, or even arbitrary formulas be contained in the set of abducibles. This gives us the possibility to generalize "negation as failure to derive" to "assertion as failure to refute". Assume we represent the bird example in the following, equivalent way

$$\begin{array}{l} fly \leftarrow bird, normal_1 \\ \neg fly \leftarrow penguin, normal_2 \\ \sim normal_1 \leftarrow normal_2 \\ penguin \\ bird \end{array} \tag{7}$$

Letting HYP = $\{normal_1, normal_2\}$ obviously yields results which are equivalent to those of our original representation using ab-predicates.

Another interesting extension of this approach are prioritized logic programs. We may introduce explicit priorities among the hypotheses, e.g. in the style of preferred subtheories [1]. The set of assumables HYP can be divided into preference levels $H_1, H_2, \ldots$ An extension base E\ is preferred to an extension base E_2 iff there is an i such that

1. $E_1 \cap (H_1 \cup \ldots \cup H_{i-1}) = E_2 \cap (H_1 \cup \ldots \cup H_{i-1})$, and
2. $E_2 \cap H_i \subset E_1 \cap H_i$.

Here is an example

$$\begin{array}{l} Pac \leftarrow Quaker, \sim ab_1 \\ \sim Pac \leftarrow Rep, \sim ab_2 \\ Quaker \\ Rep \end{array} \tag{8}$$

Let HYP = $\{\sim ab_1, \sim ab_2\}$ and assume we want to give the Quaker rule priority. This can be done by splitting HYP to Hi = $\{\sim ab_1\}$ and $H_7 = \{\sim ab_2\}$. There are two extension bases, H\ and H_2 It is easy to see that, according to our definition, the first one is preferred over the second one.

Remark: In this example this is the same as adding $ab_2 \leftarrow \sim ab_1$ to the program, that is we have a choice

whether we want to represent priorities explicitly using an ordering on HYP, or via additional rules using the available implicit prioritization. We suspect that explicit orderings make programs often more readable. Note that in case of a conflict between explicit and implicit priorities the implicit ones win since only extension bases are compared in our definition of preferred extension bases, and these respect the implicit priorities.

4 Default logic

The same abductive method for general logic programs is applicable to default logic as well. In fact, default logic rules can be viewed as a generalization of logic program rules to the full sentences, rather than just literals.

We use the more general disjunctive default theories of Gelfond et al. [8], since there is a direct correspondence to disjunctive logic programs. A disjunctive default has the form

$$\alpha : \beta_1 \cdots \beta_n \ / \ \gamma_1 | \cdots | \gamma_m \ ,$$

where all the arguments are arbitrary sentences

We begin by recalling the definition of an extension of a default theory (W, D).

Definition 9 Let T(S) be any least set such that:

1. $W \subseteq \Gamma(S)$.
2. $\Gamma(S)$ *is deductively closed.*
3. *If* $\alpha \in \Gamma(S)$ *and all* β_i *are consistent with* S *then one of* γ_j *is in* $\Gamma(S)$.

An extension of (W, D) *is any set* S *that is equal to some* $\Gamma(S)$.

As was the case with general logic programs, the closure operator is not necessarily unique.

To make the connection with the abductive method, note that the hypothesized set S are the sentences that are supposed to be in the extension, rather than the complement, as for logic programs. The operator V(S) corresponds to the closure of a program. With this in mind, we can rewrite the above definition in the abductive spirit.

Definition 10 *Let* C *be some* $\Gamma(S)$. *The cover* $COV_\Delta(C)$ *with respect to* $\Delta = \langle W, D \rangle$ *is the set:*

$$\overline{S} \cup C \ ,$$

where $\overline{S}$ are all sentences not tn S.

Given the identification of S with the complement of the abductive hypotheses, this definition corresponds to that for the cover of logic programs. The set S are the sentences assumed to be unproven in the default theory, and the added set are those hypotheses refuted by the theory. Once again, abductive extensions are formed by maximizing this union.

Definition 11 *Let* C *be some* $\Gamma(S)$. C *is an abductive extension of* Δ *iff:*

1. $C \subseteq S$.
2. $COV_\Delta(C)$ *is maximal.*

The first condition is easily seen to be a consistency condition, as it is equivalent to saying the whenever a sentence ϕ appears in $\overline{S}$, it does not appear in T(S). The second condition is the familiar maximality filter. It is obvious that $COV_\Delta(\Gamma(S))$ is maximal and consistent if $S = \Gamma(S)$, precisely the requirement for standard extensions. Hence the following proposition.

Proposition 4 If the default logic theory $\Delta = (W, D)$ has an extension and if W is consistent, then E is an abductive extension of Δ if and only if it is an extension of Δ.

Even in the cases where there is no default extension, there is an abductive extension.

Proposition 5 If a default theory $\Delta = \langle W, D \rangle$ has a finite set of justifications (over all defaults) and if W is consistent, then it has an abductive extension.

Finally, the relation between logic programs with classical negation and default theories pointed out by Gelfond et al. holds for abductive extensions as well.

Proposition 6 Let D be a set of defaults formed from the logic program P by the translation:

$$c_1 | \cdots | c_m \leftarrow a_1 \cdots a_k, \sim b_1 \cdots \sim b_n \quad \Rightarrow$$
$$a_1 \wedge \cdots \wedge a_k : \neg b_1, \cdots \neg b_n \ / \ c_1 | \cdots | c_m \ ,$$

where the a_i, b_j, and c_k are all classical literals. L is an answer set for P if and only if it is the set of literals contained in an extension of the default theory $\langle \emptyset, D \rangle$.

5 Autoepistemic logic

The same abductive approach is also applicable to autoepistemic (AE) logic and its variants, including the minimal-knowledge logic MBNF [16]. We will just show the results for standard AE logic here.

We give a brief review of AE logic concepts; for a more complete reference see [14]. In AE logic, the language $\mathcal{L}$ is formed by augmenting a propositional language $\mathcal{L}_0$ with a modal operator L for self-belief. The construction $\neg LS$ is the set $\{\neg L\phi \mid \phi \in S\}$. Finally, deduction under the modal system $K45$ is denoted by $\vdash_{K45}$.

Definition 12 *Let* A *be a set of sentences of* $\mathcal{L}$. *Let* U *be a subset of* $\mathcal{L}_0$ *and* $\overline{U}$ *its complement* $\mathcal{L}_0 - U$. U *is a moderately-grounded extension of* A *iff it satisfies the equation:*

$$U = \{\phi \in \mathcal{L}_0 \mid A \cup \neg L\overline{U} \vdash_{K45} \phi\} \ .$$

We have used the definition of moderately-grounded extensions because it is easier to see the correspondence to the logic programming and default logic cases. The hypotheses will be a set of negative modal literals $\neg LS$, and the closure of an AE theory A under the hypotheses is given by $C_A(S) = \{\phi \in \mathcal{L}_0 \mid A \cup \neg LS \vdash_{K45} \phi\}$. The cover of the theory is defined as the hypotheses S, together with all those hypotheses refuted by the closure.

Definition 13 *The cover of an AE theory* A *under hypotheses* S, $COV_A(C_A(S))$, *is given by:*

$$S \cup \{\neg L\phi \mid \phi \in C_A(S)\} \ .$$

As before, an abductive extension is a consistent maximal hypothesis set.

Definition 14 The closure of an AE theory A, $C_A(S)$, is an abductive extension of A iff

1. $C_A(S)$ *is consistent.*
2. $COV_A(C_A(S))$ *is maximal.*

The main theorems about abductive extensions is that they always exist (if A has a finite set of modal literals) and that they are equal to AE moderately-grounded extensions, when the latter exist.

Proposition 7 If A ts K45-consistent and has a finite number of modal atoms that appear positively, then A has an abductive extension.

Proposition 8 If A is K45-consistent and has an AE moderately-grounded extension, then U is such an extension if and only if it is an abductive extension of A.

The key feature of abductive extensions is that a proposition may be neither believed (Lp) nor disbelieved (-Lp) in the extension. This corresponds to the truthvalue gaps of logic programming extensions. Morris [17] defines an alternative method for giving extensions to extensionless AE theories. However, he does so by adding propositions necessary to make the fixpoint consistent, rather than leaving the belief in the propositions undecided.

6 Conclusion

We have demonstrated an abductive approach to the major consistency-based nonmonotonic formalisms. Our approach improves over previous attempts in having a straightforward relationship to the standard semantics for these formalisms, and generalizing to more complex languages. We have shown how the same idea, maximizing assumed and refuted hypotheses, can be used to establish extensions in logic programs, default logic, and autoepistemic logic. The common characteristic of these abductive extensions is that they always exist (for finite theories), they reduce to the standard extensions when the latter exist, and they allow a truthvalue gap in the cases where standard extensions do not exist.

Given the abductive nature of the definition, it might be possible to use the abundant literature in abductive methods to find approximate proof-theory methods for these formalisms (see, for example, [10, 2, 5, 18, 11, 6]).

References

[1] G. Brewka. Preferred subtheories: An extended framework for default reasoning. IJCAI, Detroit, 1989.

[2] P. T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. CADE, Lecture Notes in Computer Science 230, pages 608-621. Springer-Verlag, 1986.

[3] P. M. Dung. Negations as hypotheses: An abductive foundation for logic programming. ICLP, Paris, 1991.

[4] K. Eshghi and R. A. Kowalski. Abductive compared with negation by failure. ICLP, 1989.

[5] J.J. Finger. Exploiting Constraints in Design Synthesis. PhD thesis, Stanford University, Stanford, Ca., 1987.

[6] H. Geffner and J. Pearl. Conditional entailment: bridging two approaches to default reasoning. Artificial Intelligence, 53:209-244, 1992.

[7] M. Gelfond and V. Lifschitz. Logic programs with classical negation. ICLP, pages 579-597. Cambridge, MA, 1990. MIT Press.

[8] M. Gelfond, V. Lifschitz, H. Przymusinska, and M. Truszczynski. Disjunctive defaults. Knowledge Representation and Reasoning Conference, Cambridge, MA, 1991.

[9] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. ICLP, 1988.

[10] Jr. H. E. Pople. On the mechanization of abductive logic. IJCAI, pages 147-152, Stanford, CA, 1973.

[11] Katsumi Inoue. Consequence-finding based on ordered linear resolution. IJCAI, Sydney, Australia, 1991.

[12] Katsumi Inoue. Extended Logic Programs with Default Assumptions. ICLP, Paris, 1991.

[13] A. C. Kakas and P. Mancarella. Generalized stable models: A semantics for abduction. ECAI, Stockholm, 1990.

[14] Kurt Konolige. On the relation between default and autoepistemic logic. Artificial Intelligence, 35(3):343-382, 1988.

[15] Kurt Konolige. Using default and causal reasoning in diagnosis. Principles of Knowledge Representation and Reasoning, San Mateo, CA, 1992. Morgan Kaufmann.

[16] V. Lifschitz and Y. C. Woo. Answer sets in general nonmontonic reasoning (preliminary report). In Principles of Knowledge Representation and Reasoning, Boston, MA, 1992.

[17] Paul Morris. Stable closures, defeasible logic and contradiction tolerant reasoning. AAAI, pages 506-511, Minneapolis, MN, 1988.

[18] David Poole. Explanation and prediction: an architecture for default and abductive reasoning. Computational Intelligence, 5(2), 1989.

'19] D. Sacca and C. Zaniolo. Partial Models and Three-Valued Models in Logic Programs with Negation. Workshop on Logic Programming and Nonmonotonic Reasoning, Washington, 1991.

20] Allen van Gelder, Kenneth Ross, and J.S. Schlipf. Unfounded Sets and well-founded Semantics for general logic Programs. In Proceedings 7th Symposion on Principles of Database Systems, pages 221-230, 1988.