Abduction Compared with Negation by Failure

K. Eshghi *
Hewlett Packard Laboratories,
Filton Road, Stoke Gifford, Bristol BS12 6QZ

R. A. Kowalski
Department of Computing,
Imperial College of Science and Technology
180 Queens Gate, London SW7 2BZ

Abstract

Horn clause logic programming can be extended to include abduction with integrity constraints. In the resulting extension of logic programming, negation by failure can be simulated by making negative conditions abducible and by imposing appropriate denials and disjunctions as integrity constraints. This gives an alternative semantics for negation by failure, which generalises the stable model semantics of negation by failure.

The abductive extension of logic programming extends negation by failure in three ways: (1) computation can be performed in alternative minimal models, (2) positive as well as negative conditions can be made abducible, and (3) other integrity constraints can also be accommodated.

* This paper was written while the first author was at Imperial College.

Introduction

The term "abduction" was introduced by the philosopher Charles Peirce [1931] to refer to a particular kind of hypothetical reasoning. In the simplest case, it has the form:

From A and A ← B
infer B as a possible "explanation" of A.

Abduction has been given prominence in Charniak and McDermot's [1985] "Introduction to Artificial Intelligence", where it has been applied to expert systems and story comprehension.

Independently, several authors have developed deductive techniques to drive the generation of abductive hypotheses. Cox and Pietrzykowski [1986] construct hypotheses from the "dead ends" of linear resolution proofs. Finger and Genesereth [1985] generate "deductive solutions to design problems" using the "residue" left behind in resolution proofs. Poole, Goebel and Aleliunas [1987] also use linear resolution to generate hypotheses. All impose the restriction that hypotheses should be consistent with the "knowledge base".

Abduction is a form of non-monotonic reasoning, because hypotheses which are consistent with one state of a knowledge base may become inconsistent when new knowledge is added. Poole [1988] argues that abduction is preferable to non-monotonic logics for default reasoning. In this view, defaults are *hypotheses* formulated within classical logic rather than *conclusions* derived within some form of non-monotonic logic. The similarity between abduction and default reasoning was also pointed out in [Kowalski, 1979].

In this paper we show how abduction can be integrated with logic programming, and we concentrate on the use of abduction to generalise negation by failure.

Conditional Answers Compared with Abduction

In the simplest case, a logic program consists of a set of Horn Clauses, which are used *backward* to reduce goals to subgoals. The initial goal is solved when there are no subgoals left.

In Query-the-User [Sergot, 83], a subgoal can be solved by asking the user whether the subgoal holds and receiving an answer "yes". In more sophisticated versions of query-the-user, if the user doesn't know the answer, then the solution can be regarded as being conditional on the answer being "yes".

Consider the following simple example:

Locomotion(x fly) ← Bird(x) & Normal-bird(x)
Locomotion(x walk) ← Ostrich(x)
Bird(x) ← Ostrich(x)
Ostrich(John)

Given the goal ←Locomotion(John y) query-the-user would generate an unconditional answer Locomotion(John walk) and a conditional answer Locomotion(John fly) if Normal-bird(John). Abduction would, in place of the conditional answer, generate the hypothesis Normal-bird(John) justifying the conclusion Locomotion(John fly).

In the propositional case, given a query ←Q and theory T, a *conditional answer* is a clause of the form $Q \leftarrow \Delta$ such that $T \models Q \leftarrow \Delta$.

Under the same conditions, abduction generates Δ such that $T \cup \Delta \models Q$

Conditional answers and abductive hypotheses can be implemented by means of the same backward reasoning mechanism. We believe that abduction is more appropriate in a knowledge assimilation framework [Kowalski, 1979] when the theory undergoes change for other reasons.

We restrict the hypotheses that can be generated by abduction by insisting that their predicate symbols should belong to a set A of predicate symbols, called *abducible* predicates. An atom is *abducible* if its predicate symbol belongs to A. Some authors consider more liberal syntactic forms for hypotheses. However, these can be reduced to the simpler case of abducible atoms, by using Poole's naming device [Poole 88].

In this paper we restrict our attention further to the generation of *variable-free hypotheses*. As we shall see later, this is the analogue of restricting the selection of negative subgoals to variable-free literals in negation by failure. It is possible to liberalise this restriction. But in this case it is necessary to introduce skolem constants into hypotheses [Cox and Pietrzykowski, 1986].

Integrity Checking

The generation of abductive hypotheses can be restricted by means of integrity constraints. Thus when we have the theory T, the integrity constraints I and the abductive hypotheses Δ, we insist that $T \cup \Delta$ must satisfy I. The simplest form of integrity constraint is a denial; and the simplest notion of constraint satisfaction is logical consistency.

Suppose we add to our original example the denial

← Ostrich(x) & Normal-bird(x).

The denial functions as an integrity constraint which causes the rejection of the abductive hypothesis Normal-bird(John) and of the conclusion Locomotion(John fly) that it justifies.

Integrity constraints which are denials can be checked (inefficiently) by reasoning backward from the denials. The constraints are satisfied if there is no refutation. Thus, in theory at least, the addition of denials as integrity constraints does not necessitate any extension of the theorem-proving techniques used to execute Horn clause programs.

In practice, however, it is generally more efficient to check consistency incrementally by reasoning forward from abductive hypotheses regarded as updates to the theory. Such forward reasoning from updates is the basis for the Consistency Method [Sadri & Kowalski 1988] for checking integrity in deductive databases.

In this paper we shall use a restricted version of the consistency method, to test the consistency of abductive hypotheses. We shall see that this version of the method is similar in behaviour to the behaviour of ordinary negation by failure.

The Abduction Framework

We can now state the general specification (and declarative *semantics*) for abduction:

> <T,I,A> is an abduction framework iff
> T is a Horn clause theory (without denials),
> I is a set of integrity constraints,
> A is a set of predicate symbols, called *abducible* predicates.

Given the abduction framework <T,I,A>, the hypothesis set Δ is *an abductive solution* for the existentially quantified conjunction of atoms Q iff

> Δ is a set of variable free *abducible* atoms,
> $T \cup \Delta \models Q$
> $T \cup \Delta$ satisfies I.

For integrity constraints which are denials, $T \cup \Delta$ satisfies I if and only if $T \cup \Delta \cup I$ is consistent. Later we will define satisfaction for more general kinds of integrity constraints.

Alternative Hypotheses

In general there may be several alternative collections of hypotheses that satisfy the integrity constraints but are mutually inconsistent. This can happen, for example, in situations where Reiter's (1980) Default Logic would derive multiple conclusions holding in alternative extensions. Consider the following formulation of one of his examples:

Support(x Pacifism) ← Quaker(x) & Normal-quaker(x)
Support(x Defence) ← Republican(x) & Normal-republican(x)
← Support(x Pacifism) & Support(x Defence)
Quaker(Nixon)
Republican(Nixon)

Using the following default rules:
Conclude Normal-quaker(x) if Normal-quaker(x) is consistent
Conclude Normal-republican(x) if Normal-republican(x) is consistent,

in Default Logic, it is possible both to derive the conclusion Support(Nixon Pacifism) and to derive the conclusion Support(Nixon Defence). But it is not possible to derive the conjunction of the two conclusions. This anomally arises because the extensions used to derive the two conclusions separately are mutually inconsistent. With an appropriate reformulation of the example, circumscription (McCarthy 1986) avoids the anomaly at the expense of deriving the weaker conclusion

> Support(Nixon Pacifism) xor Support(Nixon Defence)

where "xor" is exclusive "or".

Using abduction, making Normal-republican and Normal-quaker abducible, in response to the query ←Support(Nixon x) we obtain two alternative conclusions:

> Support(Nixon Pacifism), justified by the hypothesis
> Normal-quaker(Nixon) and
> Support(Nixon Defence), justified by the hypothesis
> Normal-republican(Nixon).

The two conclusions are incompatible with one another, but each is consistent on its own. Moreover, the alternative hypotheses under which the conclusions hold have been made explicit.

By making hypotheses explicit, abduction provides more information than either Default Logic or Circumscription. Thus, in this example, we might try to resolve the conflict between the alternative theories by gathering more information, perhaps by performing a "discriminating experiment" in an attempt to refute one of the hypotheses. This contrasts with approaches such as prioritized circumscription, [Lifschitz, 1986] which require that a priority between competing hypotheses be assigned in advance.

Both Poole and Finger-Genesereth show that there is a close connection between alternative hypotheses generated by abduction and alternative extensions in Reiter's Default Logic. The main difference is that abductive hypotheses are explicit and only determine partial extensions of the knowledge base, whereas default logic generates maximal extensions which are implicit. Reiter in the conclusion of his recent survey of non-monotonic reasoning [Reiter 1987] suggests that it might be profitable to view default reasoning as a kind of hypothesis formulation.

The purpose of this paper is to show that abduction is a generalisation of negation by failure. In particular, we will show that the situation where abduction generates alternative, mutually inconsistent sets of hypotheses corresponds to the case where a logic program which is not locally stratified (Przymuszynski 1988) has several stable models (Gelfond and Lifschitz 1988).

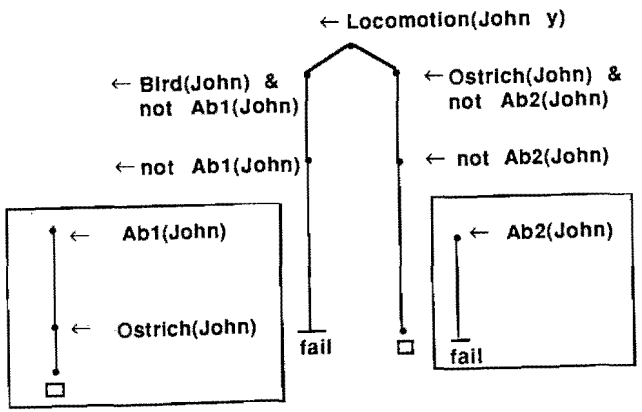
The simulation of negation by failure

The remainder of this paper concerns the use of abduction to generalise negation by failure. As touched upon in the previous sections of this paper, abduction has many other applications. A further discussion of these is beyond the scope of this work.

Consider the following variant of our earlier example. Notice that Ab1 is the complement of the predicate Normal-bird used earlier and that (v) below was expressed earlier as a denial.

(i) Locomotion(x fly) ← Bird(x) & not Ab1(x)
(ii) Locomotion(x walk) ← Ostrich(x) & not Ab2(x)
(iii) Bird(x) ← Ostrich(x)
(iv) Ostrich(John)
(v) Ab1(x) ← Ostrich(x)

The following search space is obtained using SLDNF [Clark 1978, Lloyd 1987] to find John's mode of locomotion:



In the above figure (following a notation suggested by Chris Hogger), the box shaped enclosures depict the subsidiary search spaces for negation by failure.

Using a simple transformation (to be elaborated later), we can convert a logic program L which uses negation by failure to a corresponding abduction framework <L*,I,A>. First, for every predicate symbol P in L, we introduce an additional, new predicate symbol P*.

a) L* is the set of all clauses obtained from L by replacing every occurrence of a negative condition not P(x), where x can be a vector of variables, by a positive condition P*(x). (Clauses in L which contain no negative conditions appear in L* unchanged).
b) I is the set of all denials of the form
   ← P*(x) & P(x)
   for all P* introduced by (a).
c) A is the set of all P* introduced by (a).

Notice that the integrity constraints in I express only half of a definition of P* as the complement of P. We will introduce integrity constraints corresponding to the other half later.

Applied to the program (i)-(v) above, the transformation yields the new program:

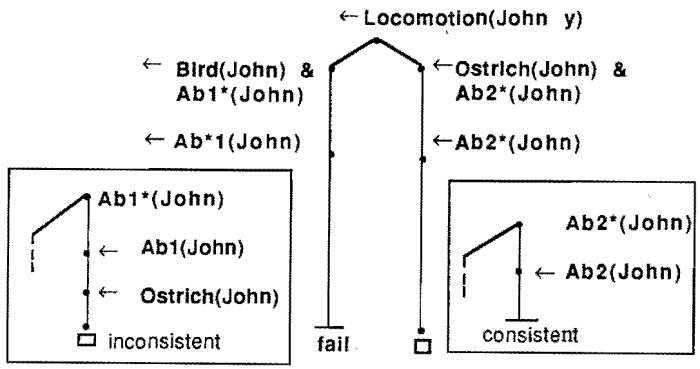
Locomotion(x fly) ← Bird(x) & Ab1*(x)
Locomotion(x walk) ← Ostrich(x) & Ab2*(x)
Bird(x) ← Ostrich(x)
Ostrich(John)
Ab1(x) ← Ostrich(x)

the integrity constraints

← Ab1*(x) & Ab1(x)
← Ab2*(x) & Ab2(x)

and the abducible predicates Ab1* and Ab2*. Notice that Ab1* is just the predicate Normal-bird used earlier.

With the same goals as before, we now obtain the following search space:



Thus we obtain the same answer as negation by failure, but with an explicit record of the hypothesis Ab2*. Notice too that the search space is almost identical to the one generated by negation by failure. There are two differences: First, there is an extra step involved in reasoning forward from the abductive hypothesis, resolving with an integrity constraint, before deriving the denial which is the top clause in the corresponding subsidiary search space for negation by failure. Second, there is an additional branch also generated by reasoning forward from the abductive hypothesis. This branch retraces in a forward direction the path originally generated backward from the initial goal to the abductive subgoal.

In the restricted version of the consistency method which we use in this paper, these branches will not be explored. However, at the end of the paper we will present examples where exploring such branches is necessary to ensure consistency.

Nested Negation

The following example illustrates the need for further integrity constraints relating abducible predicates to their complements. It also illustrates, more simply than the Hanks-McDermott (1987) example, further semantic anomalies of Default Logic and Circumscription. Notice that our example has the form of a stratified logic program:

$p \leftarrow \text{not } q$

$q \leftarrow \text{not } r$

The program has two minimal models {q} and {p, r}.

As in Reiter's Quaker-Republican example, Default Logic derives alternative, mutually inconsistent conclusions, and circumscription derives a weak, disjunctive conclusion. In considering their more complex example, Hanks and McDermott argue, in effect, that only the first of the two models is intuitively correct. They do not consider negation by failure which computes the one, "intended" model, but not the other. (In order to apply negation as failure to the Yale Shooting Problem, it is necessary to transform a sentence of the form $a \vee b \leftarrow c$ to a clause of the form $a \leftarrow c \ \&\ \text{not } b$. It is this transformation from a clause with disjunction to one with negation by failure which eliminates the unintended model).

Transforming the example into an abduction framework, we obtain the clauses and integrity constraints:

$p \leftarrow q^*$

$q \leftarrow r^*$

$\leftarrow q^* \ \&\ q$

$\leftarrow r^* \ \&\ r$

where q^* are r^* are abducible.

Like Default Logic and unlike negation by failure, abduction derives two mutually incompatible conclusions:

q supported by the hypothesis r^* and

p supported by the hypothesis q^*.

We can eliminate the second, "unintended" conclusion by including extra integrity constraints , as discussed in the following section.

More general forms of integrity constraints

The preceding example illustrates the need for integrity constraints other than denials. Such constraints are common in the field of deductive databases. Eshghi [1987] also discusses the use of such constraints in a abductive formulation of the plan-formation problem. He uses metalevel constraints of the form "p must be provable if q is provable". Reiter [1987] proposes a similar metalevel interpretation of constraints within a modal logic. Sadri and Kowalski [1988] rewrite integrity constraints as denials using negation by failure. Interpreting negation by failure as non-provability gives their integrity constraints a similar metalevel character. Noel [1988] and Small [1988] have also proposed metalevel interpretations of integrity constraints.

In this paper, we shall consider, in addition to integrity constraints which are denials, only metalevel constraints which are disjunctions of the form

Demo(T∪Δ P*(t)) ∨ Demo(T∪Δ P(t))

where Demo(A B) means the conclusion named B is provable from the theory named A and t is a variable-free term. A theory T∪Δ *satisfies* such a disjunctive integrity constraint if and only if at least one of P*(t) or P(t) is provable from T∪Δ.

In practice, because P* is abducible and does not occur in the conclusion of any clause, the disjunctive integrity constraint in effect forces P*(t) to be added to Δ if P(t) cannot be proved from T∪Δ. As we shall see below, the constraint is triggered during the consistency checking stage when a clause G of the form

← P*(t) & C

is derived and the variable-free abducible atom P*(t) is selected. Activation of the constraint causes a subsidiary search space with top clause ←P(t) to be constructed. If the search space contains a refutation then P*(t) is not provable from T ∪ Δ and the clause G has no successor. If the search space contains no refutation then P*(t) is added to Δ and G has a successor which is C.

Notice that, in practice, some form of finite failure will be needed to detect the failure of the subsidiary search space to contain a refutation. Notice too that further abductions may be made during the course of generating a successful refutation.

The Nested Negation Example Reconsidered

Returning now to our propositional example

p ← q*

q ← r*

← q & q*

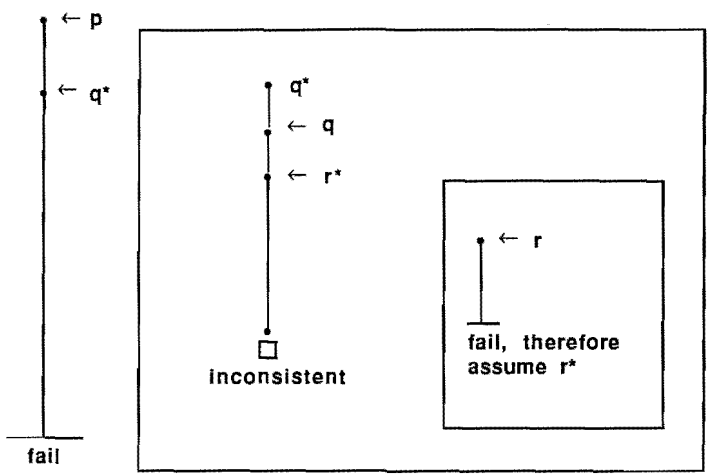
← r & r*

augmented with additional integrity constraints

Demo(T∪Δ q) ∨ Demo(T∪Δ q*)

Demo(T∪Δ r) ∨ Demo(T∪Δ r*)

we obtain the following search space:



Here the innermost subsidiary search space corresponds to the application of the disjunctive integrity constraint for r and r*. To satisfy this constraint, we must show that either r or r* is provable from T ∪ Δ. Since the attempt to prove r fails, we must ensure that r* is provable. But r* is an abducible atom. The only way to make it provable is to add it to Δ. But this means that our original assumption q* is inconsistent. Notice that except for the first step of the attempt to show q* is consistent, the search space is identical to the search space for negation by failure. This shows, therefore, that like negation by failure, abduction with appropriate integrity constraints avoids the Hanks-McDermott problem.

In the sequel we shall assume that the abduction framework which results from transforming a logic program includes a potentially infinite set of disjunctive integrity constraints having the form

Demo(T∪Δ P*(t)) ∨ Demo(T∪Δ P(t))

for every abducible predicate P* and for every variable-free term t. Where the context makes the intended meaning clear, we will avoid writing the integrity constraints explicitly.

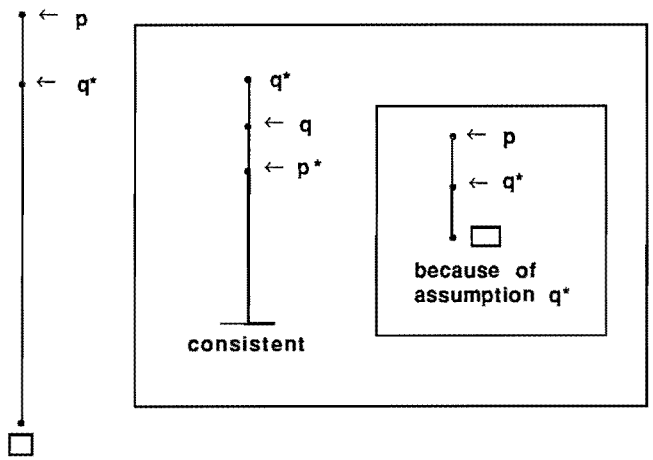
Non-stratified Negation

Abduction can deal with cases where ordinary negation by failure is semantically and operationally inadequate. Consider, for example, the non-stratified program

$p \leftarrow \text{not } q$

$q \leftarrow \text{not } p.$

Using the corresponding abduction framework, we obtain the following search space:



Here the refutation in the innermost subsidiary search space makes use of the hypothesis q^* which is assumed to hold *unless* it is inconsistent. The search space computes $\Delta = \{q^*\}$ which corresponds to the stable model {p}. Because p and q are symmetric in this example, it is similarly possible to compute an alternative $\Delta = \{p^*\}$ corresponding to the alternative stable model {q}. Thus abduction is semantically well-defined and operationally well behaved in this case, where negation by failure is not.

Before defining the abduction procedure more precisely for the negation by failure case, consider this elaboration of the preceding example:

$r \leftarrow p \text{ and } q$

$p \leftarrow \text{not } q$

$q \leftarrow \text{not } p$

Here p and q are abductive consequences supported by incompatible hypotheses. As a consequence r is not an abductive consequence. This example shows that it is necessary to keep an explicit record of the hypotheses generated when solving subgoals, so that they can be checked for mutual consistency when solving conjoint subgoals.

A Restricted Version of the Abduction Procedure

We now define a restricted version of the abduction procedure, which is general enough to deal correctly with all of the preceding examples. The proof procedure is a generalisation of SLDNF.

Let T be a Horn clause theory, A a set of abducible predicates of the form P^* (i.e. distinguished by the superscript "*"), and I a set of integrity constraints of the form

$\leftarrow P^*(x) \,\&\, P(x)$

$\text{Demo}(T \cup \Delta \;\; P^*(t)) \vee \text{Demo}(T \cup \Delta \;\; P(t)),$

for all abducible predicates P^* and for all terms t from the Herbrand universe of T. (We call these the integrity constraints for the atom $P^*(t)$). Let the abducible predicates P^* not occur in the conclusions of clauses of T. Analogously with SLDNF, let R be a safe computation rule (one that selects an abducible atom only if it contains no variables). Finally let Δ_1 be a set of abducible atoms satisfying the integrity constraints for these atoms. (Initially Δ_1 is empty).

An abductive derivation from $(G_1\ \Delta_1)$ to $(G_n\ \Delta_n)$ is a sequence

$(G_1\ \Delta_1), (G_2\ \Delta_2), \ldots, (G_n\ \Delta_n)$

such that, for each i, $1 \leq i < n$, G_i has the form $\leftarrow l \,\&\, l'$, where (without loss of generality) R selects l, and l' is a (possibly empty) collection of atoms, and

abd1) if l is not abducible, then
$G_{i+1} = C$ and $\Delta_{i+1} = \Delta_i$
where C is the resolvent of some
clause in T with the clause G_i on the selected literal l;

abd2) if l is abducible and $l \in \Delta_i$, then
$G_{i+1} = \leftarrow l'$ and $\Delta_{i+1} = \Delta_i$; and

abd3) if l is abducible, $l \notin \Delta_i$,
l has the form k^*, and
there is a consistency derivation
from $(\{\leftarrow k\}\ \Delta_i \cup \{k^*\})$ to $(\{\}\ \Delta')$,
then $G_{i+1} = \leftarrow l'$ and $\Delta_{i+1} = \Delta'$,
(where $\{\}$ is the empty set).

A *refutation* is an abductive derivation to a pair $(\square\ \Delta')$.

Note:

(1) Case (abd3) makes implicit the use of the step from k^* to $\leftarrow k$ obtained by resolving with the denial $\leftarrow k\ \&\ k^*$, which was made explicit before in our earlier examples.

(2) We shall define a consistency derivation to be a finite sequence of pairs $(F_i\ \Delta_i)$ where F_i is a set of clauses representing the tips of a search tree. Every step in the derivation involves selecting a branch of the search tree from which to continue the search, selecting a literal from the tip of the branch and attempting to extend the branch by resolution or abduction. If the branch cannot be extended (i.e. the branch "fails"), it is removed from the search tree. The derivation successfully terminates when there are no branches left in the search space.

Thus a consistency derivation is effectively a search space of abductive derivations all of whose branches finitely fail. None-the-less, we need separate definitions for consistency derivations and abductive derivations because of the different ways abductive hypotheses are treated in the two cases. In particular, in the case of abductive derivations we want abductive subgoals to succeed, in the case of consistency derivations we want them to fail.

Let T, A, I, R and Δ_1 be given as in the definition of abductive derivation. A *consistency derivation* from $(F_1\ \Delta_1)$ to $(F_n\ \Delta_n)$ is a sequence

$(F_1\ \Delta_1), (F_2\ \Delta_2), \ldots, (F_n\ \Delta_n)$

such that, for each i, $1 \leq i < n$, F_i has the form $\{\leftarrow k\ \&\ k'\} \cup F'_i$, where (without loss of generality) the clause $\leftarrow k\&k'$ has been selected (to continue the search), R selects k, and

con1) if k is not abducible, then
$F_{i+1} = C' \cup F_i'$ and $\Delta_{i+1} = \Delta_i$
where C' is the set of all resolvents
of clauses in T with the selected clause on the selected literal, and $\square \notin C'$;

con2) if k is abducible, $k \in \Delta_i$,
and k' is not empty, then
$F_{i+1} = \{\leftarrow k'\} \cup F_i'$ and $\Delta_{i+1} = \Delta_i$; and

con3) if k is abducible, $k \notin \Delta_i$, and
k has the form l^*, then
if there is an abductive derivation
from $(\leftarrow l\ \Delta_i)$ to $(\square\ \Delta')$
then $F_{i+1} = F'_i$ and $\Delta_{i+1} = \Delta'$;
otherwise, if k' is not empty, and there is no
such derivation then
$F_{i+1} = \{\leftarrow k'\} \cup F_i'$, and $\Delta_{i+1} = \Delta_i$

It is clear from the definition that the notion of abductive derivation is a generalisation of SLDNF.

Correctness

In general, because of the presence of disjunctive, metalevel integrity constraints, a theory T will not ordinarily, by itself, satisfy all of its integrity constraints. These constraints can only be satisfied by generating additional abductive hypotheses. However, the abduction procedure can only generate finitely many abductive hypotheses, and this, in general, is not adequate to satisfy the potentially infinite number of integrity constraints.

The abductive framework corresponding to the unstratified program

$r \leftarrow \text{not } r$
$p \leftarrow \text{not } q$

shows another limitation of the abduction procedure. Although there is an abductive derivation from $(\leftarrow p\ \{\})$ to $(\square\ \Delta')$, $\Delta' = \{q^*\}$, q^* is inconsistent with the framework, because the clause

$r \leftarrow \text{not } r$

alone is inconsistent with the integrity constraints. Because the consistency derivation is a restricted form of forward reasoning from an abductive hypothesis, it can only hope to detect inconsistencies which involve the hypothesis.

As a consequence we define the *correctness* of an abduction procedure to mean that for every abductive framework $<T, I, A>$, such that $T \cup I$ is consistent, whenever there exists an abductive derivation from $(\leftarrow Q\ \{\})$ to $(\square\ \Delta')$, then there exists a (possibly infinite) Δ such that $\Delta' \subseteq \Delta$ and $T \cup \Delta$ satisfies all the integrity constraints. In the locally stratified case, we can prove the following

Theorem: If the abduction framework corresponds to the transformation of a locally stratified logic program, then the restricted abduction procedure is correct.

The restricted abduction procedure is also correct for a wider class of programs, which includes for example the framework corresponding to the program

p if not q
q if not p,

as shown earlier. However, the proof procedure needs to be extended to deal correctly with other cases, as the following example shows.

The abduction framework corresponding to the program

$r \leftarrow \text{not } r$
$r \leftarrow q$
$p \leftarrow \text{not } q$
$q \leftarrow \text{not } p$

has a derivation from $(\leftarrow p\ \{\})$ to $(\square\ \Delta')$, $\Delta' = \{q^*\}$, but the only Δ such that $T \cup \Delta$ satisfies the integrity constraints $\Delta = \{p^*\}$, which is not a superset of Δ'.

At the moment we do not have a general criteria to decide for what types of program the restricted abduction procedure is correct. We conjecture that an appropriate adaptation of the Consistency Method for proving integrity would be sufficient to guarantee correctness in general.

The Relationship with Stable Model Semantics

At the time of writing this paper, the stable model semantics (Gelfond and Lifschitz, 1988) seems to be the most general semantics for logic programs. It can be shown that there is a one-to-one correspondence between stable models and abductive hypotheses Δ satisfying the integrity constraints for negation by failure:

Theorem: Let L be a logic program, and let $< L^*, I, A >$ be the corresponding abductive framework.

a) For any stable model M of L, there is a Δ such that $L^* \cup \Delta$ satisfies I, where Δ is defined by:
$d^* \in \Delta$ iff $d \notin M$.

b) For any Δ such that $L^* \cup \Delta$ satisfies I, there is a stable model M of L, where M is defined by:
$m \in M$ iff m belongs to the Herbrand base of L and
$L^* \cup \Delta \models m$.

Gelfond and Lifschitz regard a logic program as having a well-defined semantics if and only if it has a unique stable model. The abductive approach, on the other hand, assigns a semantics to programs having

multiple stable models. This is possible because, whereas stable models M are only implicit, the corresponding hypotheses Δ are explicit.

Conclusion

In this paper we have concentrated on a special case of abduction corresponding to negation by failure. We have shown that, in this special case, the semantics of abduction corresponds to, and generalises, the stable model semantics. We have also defined a restricted version of the abduction procedure, which closely resembles, and generalises, SLDNF. We have shown that the restricted procedure is correct for a class that includes locally stratified programs, and we have conjectured that the procedure can be made correct in general by adapting the Consistency Method for proving database integrity to the problem of proving the consistency of abductive hypotheses. In a related paper [Kowalski and Sadri, 1989] we have investigated similar problems from the opposite point of view, showing that in certain cases abduction with integrity checking can be replaced by negation by failure.

Acknowledgements

We are grateful to Fariba Sadri and Marek Sergot for many useful discussions. This research was supported by the Science and Engineering Research Council.

References

Bowen, K.A., and Kowalski, R.A., [1982]: "Amalgamating Language and Metalanguage in Logic Programming" Logic Programming (Clark, K.L., and Tarnlund, S.-A., eds). Academic Press, pp 153-172.

Charniak, E., and McDermott, D., [1985]: "Introduction to Artificial Intelligence" Addison-Wesley Publishing Company.

Clark, K.L., [1978]: "Negation as failure". Logic and Data Bases (Gallaire, H., and Minker, J., eds.) Plenium Press, pp. 293-322.

Cox, P.T., and Pietrzykowski, T., [1986]: "Causes for events: their computation and applications". Proc. CADE-86, (J.H. Siekmann, ed.) Spinger-Verlag, lecture notes in computer science, pp 608-621.

Eshghi, K., [1988]: "Abductive Planning with event calculus", Proc. Fifth International Logic Programming Conference, MIT Press.

Finger, J.J., and Genesereth, M.R., [1985]: "RESIDUE: A deductive approach to design synthesis. STAN-CS-85-1035, Stanford University.

Gelfond, M., and Lifschitz, V. [1988]:- The Stable Model Semantics for Logic Programming. In Proceedings of the Fifth International Logic Programming Conference and Symposium. (R. Kowalski and K. Bowen, eds.) MIT Press, Cambridge, Mass. pp. 1070-1080.

Hanks, S. and McDermott, D., [1987]: "Default reasoning, nonmonotonic logics, and the frame problem", AI Journal.

Kowalski, R.A., [1979]: "Logic for Problem Solving", Elsevier North Holland, New York.

Kowalski, R.A. and Sadri, F. [1989]: "Logic programming without integrity constraints", Department of Computing, Imperial College.

Lifschitz, V., [1986]: "Pointwise Circumscription", AAAI 1986.

Lloyd, J.W., [1987] "Foundations of Logic Programming", Second Edition, Springer-Verlag.

McCarthy, J., [1986]: "Applications of circumscription to formalising common sense knowledge". Artificial Intelligence, Vol. 28, No. 1, pp. 89-116.

Noel, P.,[1988] :"Semantic constraints in first order theories: a definition and its application", Phd. Thesis, Unveristy of Manchester

Peirce, C.S., [1931]: "Collected papers of Charles Sanders Peirce". Vol. 2, 1931-1958 (C. Hartshorn et al, eds.) Harvard University Press

Poole, D. L., Goebel, R.G., and Aleliunas [1987]: "Theorist: A logical reasoning system for defaults and diagnosis. In N. Cercone and G. McCalla, eds. The Knowledge Fronteer: Essays in the Representation of Knowledge, Spinger-Verlag, pp. 331-352.

Poole, D.L. [1988]: "A logical framework for default reasoning". AI Journal, August 88.

Przymusinski, T.C., [1988]:- "On the Declarative Semantics of Deductive Databases and Logic Programs", In Foundations of Deductive Databases and Logic Programming (J. Minker, ed.) Morgan Kaufman, Los Altos, Ca. pp. 193-216.

Reiter, R., [1980]: "A logic for default reasoning" Artificial Intelligence, Vol. 13, pp. 81-132.

Reiter, R.,[1987]: "Nonmonotonic reasoning", Annual Review of Computer Science , 1987

Reiter, R., [1988]: "On integrity constraints" To appear in Theoretical aspects of reasoning about knowledge II, Asilomar, Ca. March 6-9, 1988.

Sadri, F., and Kowalski, R.A., [1988]: "A theorem-proving approach to database integrity". In Foundations of Deductive Databases and Logic Programming (J. Minker, ed.) Morgan Kaufmann, Los Altos, Ca., pp. 313-362.

Sergot, M.J. [1983]: "A query-the-user facility for logic programming" Integrated Interactive Computer Systems (P. Degano and Sandewell, E., eds.) North Holland Press, pp. 27-41.

Small, C., [1988] "Guarded default databases: An approach to the control of incomplete information". Birkbeck College, University of London.