Well-founded Abduction via Tabled Dual Programs

José Júlio Alferes[1]
Luís Moniz Pereira[2]
Terrance Swift[3]

Abstract

Abductive Logic Programming offers a formalism to declaratively express and solve problems in areas such as diagnosis, planning, belief revision and hypothetical reasoning. Tabled Logic Programming offers a computational mechanism that provides a level of declarativity above that of Prolog, and which has supported successful applications in fields such as parsing, program analysis, and model checking. In this paper we show how to use tabled logic programming to evaluate queries to abductive frameworks with integrity constraints when these frameworks contain both default and explicit negation. Our approach consists of a transformation and an evaluation method. The transformation adjoins to each rule R of a finite ground program a new rule that is true if and only if R is false. We call the resulting program a *dual* program. The evaluation method, ABDUAL, then operates on the dual program. ABDUAL is sound and complete for evaluating queries to abductive frameworks whose entailment method is based on either the well-founded semantics with explicit negation, or answer sets. Further, ABDUAL is asymptotically as efficient as any known method for either class of problems. In addition, when abduction is not desired, ABDUAL operating on a dual program provides a novel tabling method for evaluating queries to ground extended programs whose complexity and termination properties are similar to those of the best tabling methods for the well-founded semantics. A publically available meta-interpreter has been developed for ABDUAL using the XSB system.

1 Introduction

Abductive Logic Programming [11] is a general non-monotonic formalism whose potential for applications is striking. As is well known, problems in domains such as diagnosis, planning, and temporal reasoning can be naturally modeled through abduction. In this paper we lay the basis for efficiently computing queries over ground three-valued abductive frameworks based on extended logic programs and whose notion of entailment rests on the well-founded semantics. Our query processing technique, termed ABDUAL, relies on a mixture of program transformation and tabled evaluation. In our abductive frameworks, a transformation removes negative literals from both the program over which abduction is to be performed and from the integrity rules. Specifically a *dual transformation* is used, that defines for each rule R a dual rule, that is true if and only if R is false. Tabled evaluation of the resulting program turns out to be much simpler than for untransformed normal programs, when abduction is needed, while at the same time maintaining the termination and complexity properties of tabled evaluation of extended programs when abduction is not needed.

The contributions of this paper are

- We describe ABDUAL fully and first consider its use over abductive frameworks whose entailment method is based on the well-founded semantics with explicit negation. ABDUAL is sound, complete, and terminating for queries to such frameworks over finite ground programs and integrity rules.
- We show that over abductive frameworks whose entailment method is based on the well-founded semantics

[1]Dep. Matematica, Univ. Évora and A.I. Centre, Univ. Nova de Lisboa, 2825-114 Caparica, Portugal. jja@dmat.uevora.pt
[2]A.I. Centre, Faculdade de Ciências e Tecnologia, Univ. Nova de Lisboa, 2825-114 Caparica, Portugal. lmp@di.fct.unl.pt
[3]Department of Computer Science, University of Maryland, College Park, MD, USA. tswift@cs.umd.edu

with explicit negation, the complexity of ABDUAL is in line with the best known methods. In addition, for normal and extended programs — viewed as abductive frameworks containing no abducibles or integrity constraints — query evaluation has polynomial data complexity.

- We provide a transformation whereby generalized stable models [12] can be computed using ABDUAL and show that ABDUAL provides a sound and complete evaluation method for computing such models. Furthermore, the efficiency of ABDUAL in computing generalized stable models is in line with the best known methods.

- Finally, we provide access to an ABDUAL meta-interpreter, written using the XSB system, illustrating how to evaluate ABDUAL in practice and describe how ABDUAL can be applied to medical diagnosis.

2 Preliminaries

2.1 Terminology and assumptions

Throughout this paper, we use the terminology of Logic Programming as defined in [13] with the following modifications. An *objective literal* is either an atom A, or the *explicit negation* of A, denoted $-A$. If an objective literal O is an atom A, the explicit conjugate of O ($conj_E(O)$) is the atom $-A$; otherwise if O has the form $-A$, the explicit conjugate of O is A. A *literal* either has the form O, where O is an objective literal, or $not(O)$ the *default negation* of O. Default conjugates are defined similiarly to explicit conjugates: the default conjugate ($conj_D(O)$) of an objective literal O is $not(O)$, and the default conjugate of $not(O)$ is O. Thus, every atom is an objective literal and every objective literal is a literal. An extended program P, formed over some countable language of function and predicate symbols $\mathcal{L}_P$, is the set of rules of the form H :- *Body* in which H is an objective literal, and *Body* is a possibly empty sequence of literals. If no objective literals in a program P contain the explicit negation symbol, P is called *normal*. The set of literals occurring in P is termed $literals(P)$.

The extended Herbrand base of a program P, denoted $\mathcal{H}_P$, is the set of all ground objective literals in $\mathcal{L}_P$. By a *3-valued interpretation* I of a program P we mean a set of literals over $\mathcal{H}_P$. We denote as I_T the set of objective literals in I, and I_F the set of literals of the form $not(O)$ in I. For a ground objective literal, O, if neither O nor $not(O)$ is in I, the truth value of O is undefined. An interpretation I is *consistent* if there is no objective literal O such that $O \in I_T$ and $not(O) \in I_F$; I is *coherent* if $O \in I_T$ implies $not(conj_E(O)) \in I_F$[4]. The *information ordering* of interpretations is defined as follows. Given two interpretations, I^1 and I^2, $I^1 \subseteq_{Info} I^2$ if I^1_F is a subset of I^2_F, and I^1_T is a subset of I^2_T. Given a program P, we denote by $WFS(P)$ the three-valued well-founded model of P [17], and by $WFSX(P)$ the three-valued well-founded model with explicit negation [1] of P, which generalizes WFS. Any consistent 3-valued interpretation can be viewed as a function from $\mathcal{H}_P$ to the set $\{\mathbf{f}, \mathbf{u}, \mathbf{t}\}$. Accordingly, for convenience we assume that the symbols $\mathbf{t}$ and $not(\mathbf{f})$ belong to every model, while neither $\mathbf{u}$ nor $not(\mathbf{u})$ belong to any model. For simplicity of presentation, we assume a left-to-right literal selection strategy throughout this paper, although any of the results presented here will hold for any fixed literal selection strategy. Finally, because dual programs (introduced below) allow any literal as the head of a rule, the terms *goal*, *query* and *literal* are used interchangibly.

2.2 The Well-Founded Semantics with Explicit Negation

We first recall definitions of the well-founded and stable models for extended programs. Both of these definitions make use of the operator θ^X_I [4].

Definition 2.1 [The θ^X_J Operator] Let P be an extended program and J an interpretation. The operator θ^X_J maps interpretations of P to interpretations of P and is defined as follows. If I and J are 3-valued

[4]In a coherent interpretation if some atom is explicitly false (resp. true) then it must be false (resp. true) by default.

interpretations of P and A an objective literal, then $\theta_J^X(I)$ is defined as

1. $A \in \theta_J^X(I)$ iff there is a rule A :- $L_1, ..., L_n$ in P such that for $1 \leq i \leq n$, L_i is an objective literal and $L_i \in I$, or L_i is of the form $not(O)$, and $not(O) \in J$.
2. $not(A) \in \theta_J^X(I)$ iff
   (a) for every rule A :- $L_1, ..., L_n$ in P, there is a literal L_i for $1 \leq i \leq n$, such that either L_i is of the form $not(G)$ and $G \in J$, or L_i is an objective literal and $not(L_i) \in I$; or
   (b) $conj_E(A) \in J$

□

The θ_J^X operator is an extension of a similar operator for normal programs (e.g. [15]). Indeed, the only addition required for explicit negation is clause 2b, which ensures coherency. It can be shown [4] that θ_J^X is monotonic on the truth ordering for 3-valued interpretations, so that it has a unique least fixed point for a given J, denoted by $\omega^X(J) = lfp(\theta_J^X(Neg\_Olits))$, where $Neg\_Olits = \{not(O) | O \in \mathcal{H}_P\}$. Furthermore, it can be shown [4] that the operator ω^X is also monotonic on the information ordering of interpretations leading to the following definition of the well-founded semantics with explicit negation.

Definition 2.2 [Paraconsistent Well-founded Semantics with Explicit Negation] Let P be an extended program. Then $WFSX_P(P)$ is defined as the least fixed point of $\omega^X(\emptyset)$. □

Example 2.1 Let P be the program containing the rules $\{c$:- $not(b)$; b :- a; $-b$; a :- $not(a)\}$. Then $WFSX_P(P) = \{-b, c, not(-a), not(b), not(-c)\}$. Note that to compute c, coherency (condition 2b of Definition 2.1) must be used to infer $not(b)$ from $-b$.

Using the operator ω^X it is possible to define a stability operator for extended programs that allows partial, and possibly paraconsistent models.

Definition 2.3 [Partial Stable Interpretation of an Extended Program] Let P be an extended program. We call an interpretation J a *partial stable interpretation of* P if $J = \omega^X(J)$ □

If an interpretation I contains both O and $-O$, then through coherency, $\omega^X(I)$ will contain both O and $not(O)$ and so will be inconsistent. Thus, by definition an interpretion I can be a partial stable interpretation even if it is inconsistent. However as we will see, within abductive frameworks consistency can be ensured by means of integrity constraints — for instance, prohibiting O and $-O$ to be true for any objective literals O.

2.3 Three-Valued Abductive Frameworks

The definitions of three-valued abductive frameworks modify those of [5].

Definition 2.4 [Integrity Rule] An *integrity rule* for a program P has the form

$$\perp \text{ :- } L_1, \ldots, L_n$$

where each L_i, $1 \leq i \leq n$ is a literal formed over an element of $\mathcal{L}_P$. □

Definition 2.5 [Abductive Framework] An abductive framework is a triple $\langle P, \mathcal{A}, I \rangle$ where $\mathcal{A}$ is a set of ground objective literals of $\mathcal{L}_P$ called *abducibles*, such that for any objective literal O, $O \in \mathcal{A}$ iff $conj_E(O) \in A$, I is a set of ground integrity rules, and P is a ground program such that (1) there is no rule in P whose head is in $\mathcal{A}$; and (2) $\perp /0$ is a predicate symbol not occurring in $\mathcal{L}_P$.

An *abductive subgoal* $S = \langle A, Set \rangle$ is a literal A together with a set of abducibles, Set, called the *context* of A. If the context contains both an objective literal and its (explicit or default) conjugate, it is termed *inconsistent* and is *consistent* otherwise. □

The requirement that there can be no rule in P whose head is an abducible leads to no loss of generality, since any program with abducibles can be rewritten to obey it [5].

[5]For instance, if it is desired to abduce A, one may introduce a new abducible predicate A', along with a rule A :- A' [11].

Definition 2.6 [Abductive Scenario] A scenario of an abductive framework $\langle P, \mathcal{A}, I\rangle$ is a tuple $\langle P, \mathcal{A}, \mathcal{B}, I\rangle$, where $\mathcal{B}$ is a consistent 3-valued interpretation of each $A \in \mathcal{A}$. $P_{\mathcal{B}}$ contains, for each $A \in \mathcal{A}$, the rule A :- $\mathbf{t}$ iff $A \in \mathcal{B}$; A :- $\mathbf{f}$ iff $not(A) \in \mathcal{B}$; and A :- $\mathbf{u}$ if neither A nor $not(A)$ is in $\mathcal{B}$. □

Definition 2.7 [Abductive Solution] An abductive solution is a scenario $\sigma = \langle P, \mathcal{A}, \mathcal{B}, I\rangle$ of an abductive framework, such that $\perp$ is false in $M(\sigma) = WFSX_P(P \cup P_{\mathcal{B}} \cup I)$. □

We say that $\sigma = \langle P, \mathcal{A}, \mathcal{B}, I\rangle$ is an abductive solution for a query Q if $M(\sigma) \models Q$. σ is minimal, if there is no other abductive solution $\sigma = \langle P, \mathcal{A}, \mathcal{B}', I\rangle$ for Q such that $WFM(\mathcal{B}') \subseteq_{info} WFM(\mathcal{B})$.

3 Evaluation of the Abductive Solutions for a Query over the Well-Founded Semantics With Explicit Negation

We informally introduce ABDUAL through a series of examples.

Example 3.1 We first illustrate how ABDUAL can be used to compute queries to ground programs according to the well-founded semantics when neither abduction nor integrity constraints are needed. Accordingly, consider the abductive framework $\langle P_1, \emptyset, \emptyset\rangle$, in which the set of abducibles and the set of integrity rules are both empty, and $P_1 =$

```
s  :- not(p), not(q), not(r).
p  :- not(s), not(r), q.
q  :- not(p), r.
r  :- not(q), p.
```

Note that, taken as a normal program, WFM(P_1) = {`s, not(p), not(q), not(r)`}. In order to evaluate the query `?-s` through ABDUAL, we first create the dual form of P_1 taken together with a *query rule*

```
query  :- s, not(⊥).
```

where `query` is assumed not to be in $\mathcal{L}_{P_1}$. This rule ensures that integrity constraints are checked for any abductive solutions that are derived. This dual program, $dual(P_1 \cup query$:- $s, not(\perp))$ is shown in Figure 1.

```
s  :- not(p), not(q), not(r).              not(s)   :- p.
                                           not(s)   :- q.
                                           not(s)   :- r.
p  :- not(s), not(r), q.                   not(p)   :- s.
                                           not(p)   :- r.
                                           not(p)   :- not(q).
q  :- not(p), r.                           not(q)   :- p.
                                           not(q)   :- not(r).
r  :- not(q), p.                           not(r)   :- q.
                                           not(r)   :- not(p).
query  :- s,not(⊥).                        not(query) :- not(s).
                                           not(query) :- ⊥.
                                           not(⊥).

             not(p)  :- -p.    not(-p)  :- p.
             not(q)  :- -q.    not(-q)  :- q.
             not(r)  :- -r.    not(-r)  :- r.
             not(s)  :- -s.    not(-s)  :- s.
```

Figure 1: *Dual Program for* $(P_1 \cup query$:- $s, not(\perp))$

Note that in the dual form of a program, rules can have default literals of the form $not(A)$ in their heads, and a rule for $not(A)$ is designed to be true if and only if A is false. The last four lines of Figure 1 are *coherency axioms* so-named because they ensure coherency of the model computed by ABDUAL. As is usual with tabled evaluations (e.g. [2]), the ABDUAL evaluation of a query to $dual(P_1)$ is represented as a sequence $\mathcal{F}_0, ..., \mathcal{F}_i$, of forests of ABDUAL trees. $\mathcal{F}_0$ is the forest consisting of the single tree $< query, \emptyset >$:- $|query$, which calls the query rule. Given a successor ordinal $i+1$, a forest $\mathcal{F}_{i+1}$ is created when an ABDUAL operation either adds a new tree to $\mathcal{F}_i$ or expands a node in an existing tree in $\mathcal{F}_i$. A forest of trees at the end of one possible ABDUAL evaluation of the above query is shown in Figure 2 [6]. Nodes in Figure 2 are all *regular* having the form *Abductive_subgoal* :- *GoalList*|*DelayList*, where *Abductive_subgoal* is an abductuve subgoal (Definition 2.5), and *GoalList* and *DelayList* are sequences of literals. In ABDUAL terminology, when an evaluation encounters

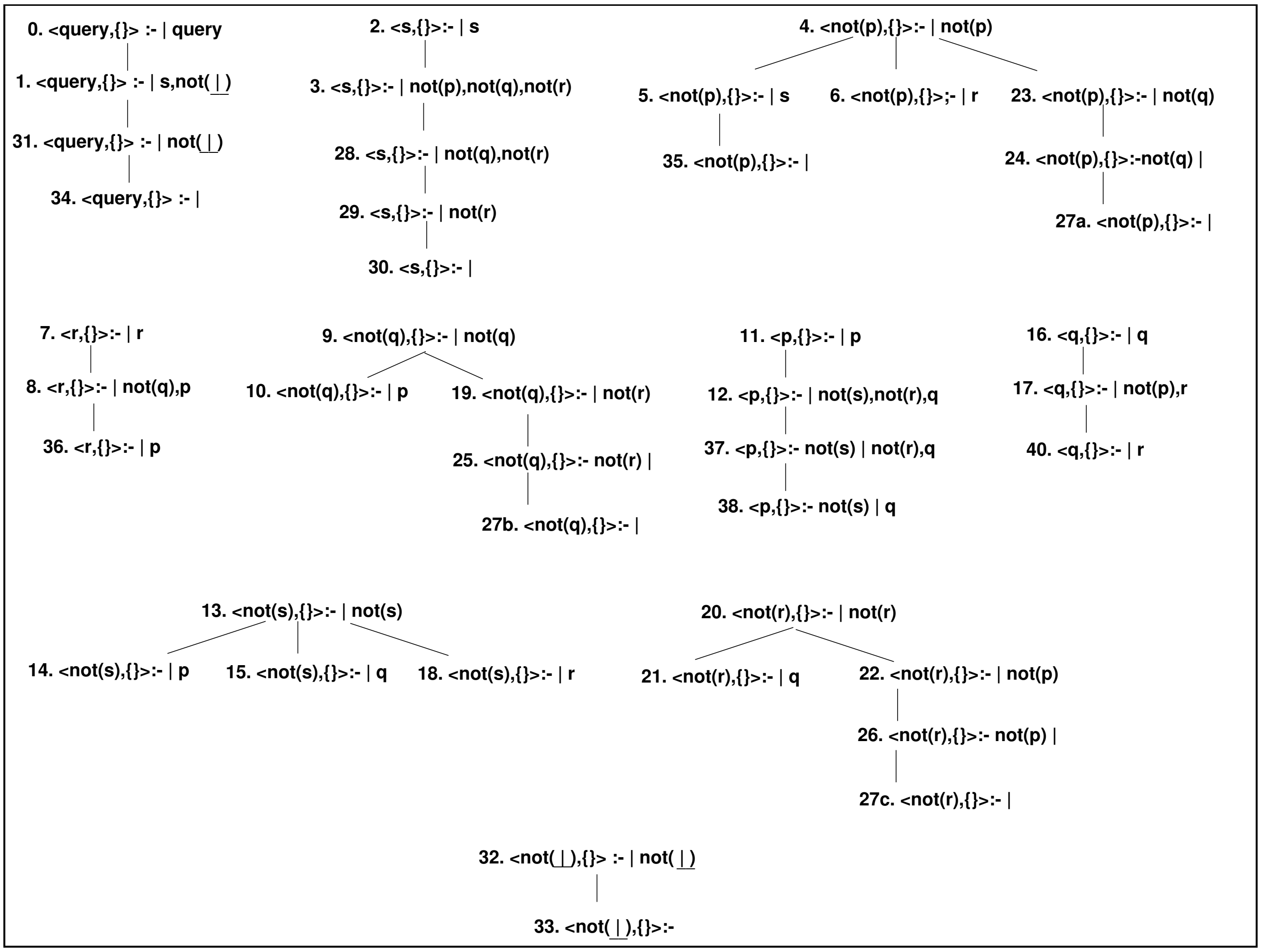

Figure 2: Simplified ABDUAL Evaluation of a Normal Program

a new literal, S, a tree with root S :- $|S$ is added to the forest via the NEW SUBGOAL operation. Thus, in Figure 2, when the literal s is selected in node 1, a NEW SUBGOAL operation creates node 2 as a single tree. Children of the roots of trees are created via PROGRAM CLAUSE RESOLUTION operations. The left-to-right literal selection strategy can be broken by a DELAYING operation of a non-root node, which moves a selected literal from the *GoalList* to the *DelayList* of the node — nodes 24, 25, and 26 are created in this way. An

[6]For simplicity of presentation, Figure 2 does not include computation paths that include the coherency axioms.

answer is a leaf node with an empty *GoalList*. In the subforest of Figure 2 consisting of nodes whose index is 26 or less, nodes 24, 25, and 26 are all answers. Because each of their delay lists is non-empty they are termed *conditional answers*. However, in the well-founded model of P_1 each of these answers should be unconditionally true, and in order to derive this a CO-UNFOUNDED SET REMOVAL operation is needed. Note that nodes 24, 25, and 26 together form an analogue in the dual program to an unfounded set [17] consisting of p, q, and r in P_1. Such an analogue is called a *co-unfounded set*. Whereas positive literals of an unfounded set are all false, negative literals for a co-unfounded set are all true. When a set of conditional answers is determined to form a co-unfounded set, the answers are all made unconditionally true. Answers are returned to other nodes via the ANSWER CLAUSE RESOLUTION operation which also combines the abductive contexts of the answer and the node. For instance, the answer `<not(p),{}> :-` | is returned to the node 3 through ANSWER CLAUSE RESOLUTION.

We now formalize the definitions of some concepts introduced in Example 3.1.

Definition 3.1 [Dual Program] Let P be an extended program. The *dual transformation* creates a *dual program* $dual(P)$, defined as the union of P with smallest program containing a set of rules R_1 and R_2 as follows:

1. If O is an objective literal for which there are no facts in P, and all of its rules are:

$$\begin{array}{ll} O \text{ :- } & L_{1,1}, ..., L_{1,n_1} \\ \vdots & \\ O \text{ :- } & L_{m,1}, ..., L_{m,n_m} \end{array}$$

   (a) then R_1 contains the rule

$$not(O) \text{ :- } \quad fold\_O_1, ..., fold\_O_m.$$

   (b) and for $1 \leq i \leq m$, R_1 contains the rules $fold\_O_i, 1 \leq i \leq m$

$$\begin{array}{ll} fold\_O_i \text{ :- } & conj_D(L_{i,1}). \\ \vdots & \\ fold\_O_i \text{ :- } & conj_D(L_{i,n_i}). \end{array}$$

   where $fold\_O_i$ is assumed not to occur in $\mathcal{L}_P$ for $1 \leq i \leq m$ (such rules are termed *folding rules*, and literals formed from atoms whose predicate symbol is $fold\_O_i$ are called folding literals).

   (c) Otherwise, if $not(O)$ is in $literals(P)$, but there is no rule with head O in P, then R contains the rule $not(O)$:-

   (d) R_2 consists of *axioms of coherence* that relate explicit and default negation, defined as:

$$not(O) \text{ :- } conj_E(O)$$

   For each objective literal $not(O)$ in either $literals(P \cup R_1)$ or A.

□

In the $dual(P)$ form, $not(O)$ is true iff O is false in P. For instance, if there is a fact in P for some objective literal O then the dual has no rule for $not(O)$.

Definition 3.2 [ABDUAL Trees and Forest] An ABDUAL forest consists of a forest of ABDUAL trees. Nodes of ABDUAL trees are either *failure nodes* of the form *fail*, or *regular nodes* of the form

$$Abductive\_Subgoal \text{ :- } DelayList|GoalList$$

where *Abductive_subgoal* is an abductive subgoal. Both *DelayList* and *GoalList* are sequences of literals (also called delay literals and goal literals, respectively).

We call a (non-failure) leaf node N an *answer* when *GoalList* is empty. If *DelayList* is also empty, N is *unconditional*; otherwise it is *conditional*. □

Definition 3.7 ensures that the root node of a given ABDUAL tree, T, has the form $\langle S, \emptyset\rangle$:- $|S$, where S is a literal. In this case, we say that S is the *root goal* for T or that T is the *tree for* S. Similarly by Definitiondef:ops, a forest contains a root goal S if the forest contains a tree for S. Literal selection rules apply to the *GoalList* of a node; as mentioned in Section 2, we use a fixed left-to-right order for simplicity of presentation.

The next example illustrates how ABDUAL can evaluate queries to general abductive frameworks.

Example 3.2 Consider the abductive framework $\langle P_2, A, I\rangle$, in which P_2 is the program

```
p :- not(q*).
q :- not(p*).
```

$A = \{p*, q*, -p*, -q*\}$, and I is the program

```
⊥ :- p_constr
⊥ :- q_constr
p_constr :- p, -p*.
q_constr :- q, -q*.
```

So that the (ground) integrity constraints represent an abductive interpretation of default negation. Let the query rule be

```
query  :- q,not(⊥).
```

The dual program is shown in Figure 3.

```
p :- not(q*).                      not(p) :- q*.
q :- not(p*).                      not(q) :- p*.
⊥ :- p_constr                      not(⊥) :- not(p_constr),not(q_constr)
⊥ :- q_constr
p_constr :- p, -p*.                not(p_constr) :- not(p)
                                   not(p_constr) :- not(-p*).
q_constr :- q, -q*.                not(q_constr) :- not(q)
                                   not(q_constr) :- not(-q*).
query :- q,not(⊥).                 not(query) :- not(q).
                                   not(query) :- not(⊥).

not(-p)  :- p.                     not(p)  :- -p.
not(-q)  :- q                      not(q)  :- -q
not(-p*)  :- p*                    not(p*)  :- -p*
not(-q*)  :- q*                    not(q*)  :- -q*
not(-p_constr)  :- p_constr        not(p_constr)  :- -p_constr
not(-q_constr)  :- q_constr        not(q_constr)  :- -q_constr
```

Figure 3: *Dual program for ($P_2\cup$ query :- q,not($\perp$)).*

Figure 4 illustrates a forest of trees created by an ABDUAL evaluation of this initial query. As with Figure 2, derivations stemming from coherency axioms are not presented. When abducibles are encountered in an evaluation, provision must be made for when the selected literal of a given node is an abducible, as well as for propagation of abducibles among abductive goals. In the first case, if the selected atom of a node N is an abducible, and the addition of the selected atom to the context of the abductive subgoal of N does not make the context inconsistent (Definition 2.5), an ABDUCTION operation is applicable to N. As an example, ABDUCTION is used to create a child for node 3, `<q,{}> :- not(p*)`. Abducibles are propagated through the ANSWER CLAUSE RESOLUTION operation, which has the restriction that the context of the answer must be consistent with the context of the node to which the answer is returned. For instance, of the two unique abductive solutions to `not(⊥)` only one can be returned to the node `<query,{p*}> :- not(⊥)`, namely

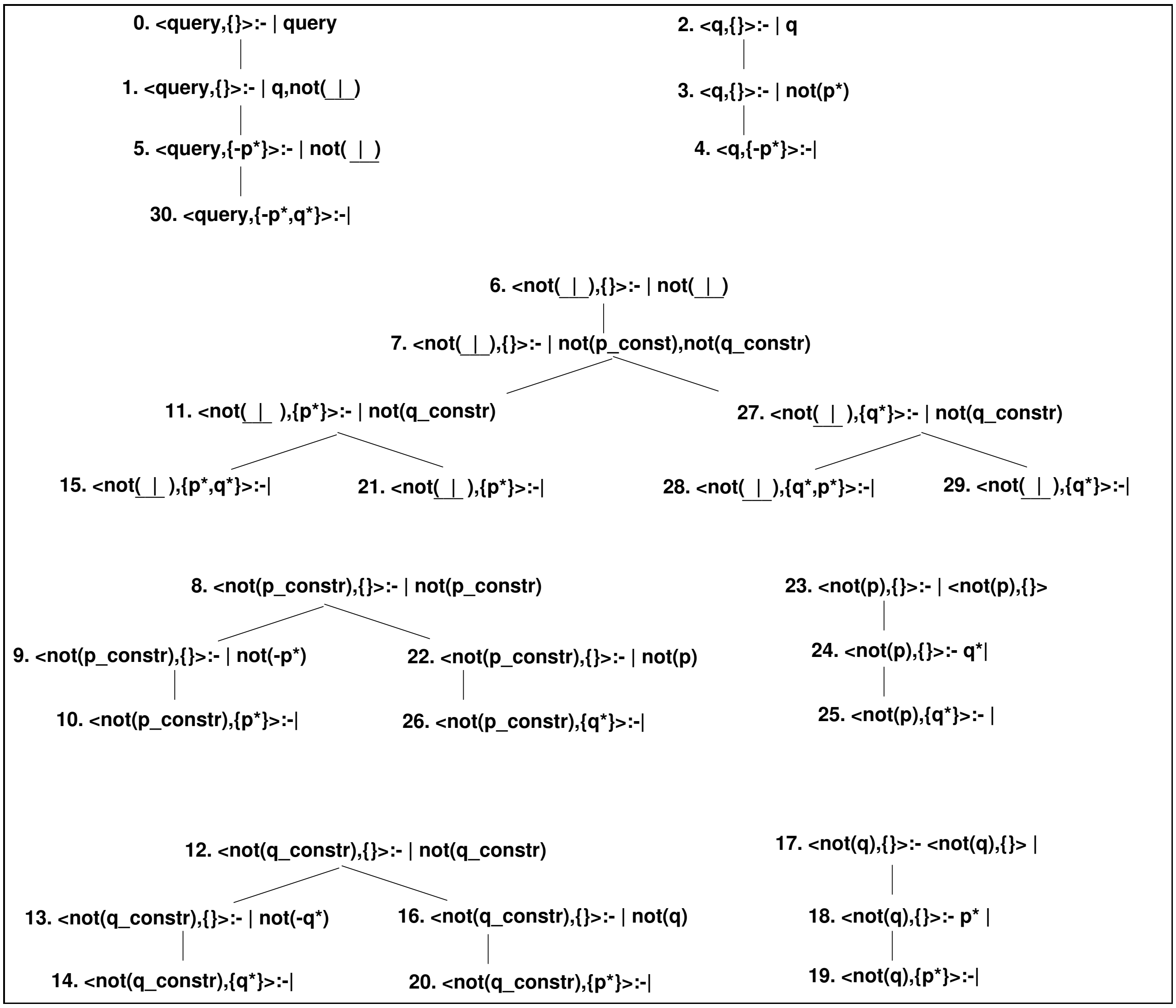

Figure 4: *A finite* ABDUAL *forest the query* `?- q` *to* P_2.

`<not(⊥),{q*}>`.

The final definitions for ABDUAL are now provided. First, a notion is needed to determine when a set of trees has produced all abductive answers that it will ever produce. This is captured by the notion of a set of trees being completely evaluated.

Definition 3.3 [Completely Evaluated] Given a forest $\mathcal{F}$, a set $\mathcal{T}$ of ABDUAL trees is *completely evaluated* iff at least one of the following conditions is satisfied for each tree $T \in \mathcal{T}$:

1. T contains an unconditional answer whose context is empty; or
2. For each node N in $\mathcal{T}$ with selected literal SL
   - The tree for SL belongs to a set S' of completely evaluated trees; and
   - No New Subgoal, Program Clause Resolution, or Answer Clause Resolution operations (Definition 3.7) are applicable to N.

A literal L is completely evaluated in $\mathcal{F}$ if the tree for L belongs to a completely evaluated set in $\mathcal{F}$. □

Evaluation of the program in Example 3.1 required detection of a co-unfounded set, whose formal definition is as follows.

Definition 3.4 [Co-Unfounded Set of Answers] Let $\mathcal{F}$ be an ABDUAL forest, and S a set of answers in $\mathcal{F}$. Then S is a co-unfounded set in $\mathcal{F}$ iff

1. Each literal S_i, such that $< S_i, C_i >$ is the abductive subgoal of an ansewr in S, is completely evaluated.
2. The set
$$Context = \{C_i | < S_i, C_i > \text{is the abductive subgoal of an answer in } S\}$$
is consistent; and
3. For each answer $< S_i, C_i >$:- $DL_i| \in S$
   (a) DL_i is non-empty; and
   (b) for each $S_j \in DL_i$, there exists an answer $< S_j, Context_j >$:- $DL_j| \in S$.

□

An ABDUAL evaluation consists of a (possibly transfinite) sequence of ABDUAL forests. Our definition here follows that of [16] for generalized SLG trees. In order to define the behavior of an ABDUAL evaluation at a limit ordinal, we define a notion of a least upper bound for a set of ABDUAL trees. If a global ordering on literals is assumed, then the elements in the *Context* of a node can be uniformly ordered, and using this ordering an equivalence relation can be defined for nodes of ABDUAL trees. Furthermore, any rooted tree can be viewed as a partially ordered set in which each node N is represented as $\{N, P\}$ in which P is a tuple representing the path from N to the root of the tree. When represented in this manner, it is easily seen that when T_1 and T_2 are rooted trees, $T_1 \subseteq T_2$ iff T_1 is a subtree of T_2, and furthermore, that if T_1 and T_2 have the same root, their union can be defined as their set union, for T_1 and T_2 taken as sets.

Definition 3.5 [ABDUAL Evaluation] Let $\mathcal{A} = \langle P, A, I \rangle$ be an abductive framework and Q a query. An ABDUAL evaluation $\mathcal{E}$ of Q to $\mathcal{A}$ is a sequence of ABDUAL forests $\mathcal{F}_0, \mathcal{F}_1, ..., \mathcal{F}_n$ operating on the ground instantiation of $dual(P \cup I \cup \{query \text{ :- } not(\perp), Q\})$ such that:

- $\mathcal{F}_0$ is the forest containing the single tree, $< query, \{\} >$:- $|query$,
- For each successor ordinal $n + 1 \leq \beta$, $\mathcal{F}_{n+1}$ is obtained from $\mathcal{F}_n$ by applying an ABDUAL operation from Definition 3.7.
- For each limit ordinal $\alpha \leq \beta$, $\mathcal{F}_\alpha$ is defined such that $T \in \mathcal{F}_\alpha$ iff

– The root node of T, $< S, \{\} >$:- $|S$ is the root node of some a tree in a forest F_i, $i < \alpha$;
– $T = \cup(\{T_i | T_i \in \mathcal{F}_i, i < \alpha$ and T_i has root S :- $|S)$

If no operation is applicable in $\mathcal{F}_n$, then it is called a *final forest* of $\mathcal{E}$. □

Similarly to SLG, ABDUAL does not propagate delay lists of delayed answers — that is if an answer p :- $q, r|$ is returned to a node a :- $|p, s$ the resulting node will be a :- $p|s$ rather than a :- $q, r|s$. This action is necessary for ABDUAL to have polynomial complexity for normal programs in the absence of abduction (c.f. Theorem 3.3). However, in certain special cases it is possible to propagate conditional answers that cannot be simplified away. In this case the answers become unsupported (see [2] for an example of this).

Definition 3.6 [Supported Answers] Let $\mathcal{F}$ be a forest, and S the root goal for a tree T in $\mathcal{F}$. Then S is supported in $\mathcal{F}$ iff

1. T is not completely evaluated; or
2. there exists an answer $< S, Context >$:- $DL|$ of S such that, for every (positive) delayed literal L_1 in DL, L_1 is supported in $\mathcal{F}$.

□

Definition 3.7 [ABDUAL Operations] Let $\mathcal{F}_n$ be an ABDUAL forest for an evaluation of a query Q to an abductive framework $\mathcal{A} = \langle P, A, I \rangle$, and suppose n is a successor ordinal. Then $\mathcal{F}_{n+1}$ may be produced by one of the following operations

1. NEW SUBGOAL: Let $\mathcal{F}_n$ contain a non-root node
$$N = \mathit{AbductiveSubgoal} \text{ :- } DL|S, \mathit{GoalList}.$$
Assume $\mathcal{F}_n$ contains no tree with root goal S. Then add the tree: $< S, \{\} >$:- $|S$.
2. PROGRAM CLAUSE RESOLUTION: Let $\mathcal{F}_n$ contain a root node
$$N = < S, \{\} > \text{ :- } |S$$
and C be a program clause S :- Body. Assume that in $\mathcal{F}_n$, N does not have a child:
$$N_{child} = < S, \{\} > \text{ :- } |\mathit{Body}$$
Then add N_{child} as a child of N.
3. ANSWER CLAUSE RESOLUTION: Let $\mathcal{F}_n$ contain a non-root node $N = < S, C_1 >$:- $DL_0|G, \mathit{Body}$ and suppose that $\mathcal{F}_n$ contains an answer node $< G, C_2 >$:- $DL_1|$ for G, such that $C_1 \cup C_2$ is consistent. Let $DL_2 = DL_0, G$ if DL_1 is not empty, and $DL_2 = DL_0$ otherwise. Finally, assume that in $\mathcal{F}_n$, N does not have a child
$$N_{child} = < S, C_1 \cup C_2 > \text{ :- } DL_2|\mathit{Body}$$
Then add N_{child} as a child of N.
4. DELAYING: Let $\mathcal{F}_n$ contain a non-root leaf node
$$N = < S, \mathit{Context} > \text{ :- } \mathit{DelayList}|\mathit{not}(G), \mathit{Body}$$
where G is not an abducible, and where $\mathcal{F}_n$ contains a tree for $\mathit{not}(G)$, but no answer of the form $< \mathit{not}(G), \{\} >$:- $|$. Then add: $< S, \mathit{Context} >$:- $\mathit{DelayList}, \mathit{not}(G)|\mathit{Body}$ as a child of N.
5. SIMPLIFICATION: Let $N = < S, C_1 >$:- $DL|$ be a node for a tree with root goal S, and let D be a delayed literal in DL. Then

- if $\mathcal{F}_n$ contains an unconditional answer node $< D, C_2 >$:- $|$, and if $C_1 \cup C_2$ is consistent, let $DL_1 = DL - D$. If
$$N_{child} =< S, C_1 \cup C_2 > \text{ :- } DL_1|$$
is not a descendent of N in $\mathcal{F}$, add N_{child} as a child of N.
- if the tree for D is completely evaluated and contains no answers; or if D is non-supported, then create a child $fail$ of N.

6. CO-UNFOUNDED SET REMOVAL: Let $\mathcal{S}$ be a co-unfounded set and let C_{union} be the consistent union of all contexts of answers in $\mathcal{S}$. Then for each
$$N =< S, C_1 > \text{ :- } DL| \in S$$
create a child of N: $N_{child} =< S, C_{union} >$:- $|$
7. ABDUCTION: Let
$$N =< S, Context > \text{ :- } DL|G, Body$$
where G is an abducible, and suppose that $G \cup Context$ is consistent. Finally, assume that in $\mathcal{F}_n$, N does not have a child
$$N_{child} =< S, Context \cup G > \text{ :- } DL|Body$$
Then add N_{child} as a child of N.

□

The definitions of ABDUAL bear some similarity to those of SLG (see Section 4).

Theorem 3.1 [7] *Let $\mathcal{A} = \langle P, A, I \rangle$ be an abductive framework such that P is an extended program, and I a set of extended integrity rules. Let $\mathcal{E}$ be an* ABDUAL *evaluation of Q against $\mathcal{A}$. Then*

- *$\mathcal{E}$ will have a final forest $\mathcal{E}_\beta$;*
- *if $< query, Set >$:- $|$ is an answer in $\mathcal{E}_\beta$ $\sigma = \langle P, A, Set, I \rangle$ is an abductive solution for $\mathcal{A}$;*
- *if $\langle P, A, Set, I \rangle$ is a minimal abductive solution for Q, then $< query, Set >$:- $|$ is an answer in $\mathcal{E}_\beta$.*

3.1 Finite Termination and Complexity of ABDUAL for Extended Programs

Termination of ABDUAL evaluations is characterized by the following theorem.

Theorem 3.2 *Let $\mathcal{A} = \langle P, A, I \rangle$ be an abductive framework such that P and I are finite ground extended programs, and A is a finite set of abducibles. Let $\mathcal{E}$ be an* ABDUAL *evaluation of a query Q against $\mathcal{A}$. Then $\mathcal{E}$ will reach a final forest after a finite number of* ABDUAL *operations.*

It is known that the problem of query evaluation to abductive frameworks is NP-complete, even for those frameworks in which entailment is based on the well-founded semantics. However, in the special case in which an abductive framework reduces to an extended (or normal) program ABDUAL can evaluate queries with polynomial data complexity, the definition of which we now recall.

Definition 3.8 [17] The *intension* of a program P, P_I, consists of all rules in P with non-empty bodies; the *extension* of P, P_E, consists of all rules in P whose body is empty. The data complexity of P is the computational complexity of deciding an answer to a ground atomic query as a function of the size of P_E. □

Theorem 3.3 *Let P be an extended program, and $\mathcal{A} = \langle P, \emptyset, \emptyset \rangle$ be an abductive framework of P, such that the ground instantiation of P is finite. Let $\mathcal{E}$ be an* ABDUAL *evaluation of a non-abductive query Q against $\mathcal{A}$, whose final forest is $\mathcal{E}_\beta$. Then $\mathcal{E}_\beta$ can be constructed in time polynomial in $|P_E|$.*

[7] Proofs of theorems are provided in the full version of this paper, available at `http://www.cs.sunysb.edu/~ tswift`.

3.2 Construction of Generalized Stable Models through Abdual

The three-valued abductive frameworks of Section 2 are not the only semantics used for abduction: Generalized Stable Models [12] provide an important alternative. In [5] it was shown that the abductive framework of Section 2 has the same expressive power as generalized stable models [11]. In this section, we reformulate these results to show that Abdual can be used to evaluate abductive queries over generalized stable models. By allowing all positive literals to be inferred through abduction, Abdual can be used to construct partial stable interpretations (Definition 2.3). By choosing appropriate integrity constraints, these interpretations can be constrained to be consistent and total. We begin by adapting the concept of a generalized stable model to the terminology of Section 2.

Definition 3.9 [Generalized Partial Stable Interpretation and Model] Let σ be scenario $\sigma = \langle P, A, \mathcal{B}, I\rangle$ of an abductive framework. Then $M(\mathcal{B})$ is a generalized partial stable interpretation of $\langle P, A, I\rangle$ if

- $M(\mathcal{B})$ is a partial stable interpretation of $(P \cup P_{\mathcal{B}} \cup I)$; and
- $\perp$ is false in $M(\mathcal{B})$.

If in addition $M(\mathcal{B})$ is an answer set of $(P \cup P_{\mathcal{B}} \cup I)$, σ is a generalized stable model of $\langle P, A, I\rangle$. □

Generalized stable models can be computed by adding additional program rules, abducibles, and integrity constraints to abductive frameworks and computing the solution to these frameworks as per Definition 2.7.

Definition 3.10 Let $\mathcal{A} = \langle P, A, I\rangle$ be an abductive framework. Then for each objective literal O, let abd_O be a new objective literal, not in $\mathcal{H}_P$. Let

$$R = O \text{ :- } Body$$

be a rule in P. Then a *shadow rule* for R is a rule

$$R_{shadow} = O \text{ :- } Body_{abd}$$

in which each literal of the form $not(O')$ in $Body$ $(O' \in \mathcal{H}_P)$ is replaced by $not(abd\_O')$. Corresponding to the shadow rules are *shadow constraints* (I_{shadow}) of the form

$$\begin{array}{rl} \perp \text{ :- } & O, not(abd\_O). \\ \perp \text{ :- } & not(O), abd\_O. \end{array}$$

for each abd_O such that $not(abd\_O) \in S(P)$.

The *consistency constraints* $(I_{consist})$ for $\mathcal{A}$ consist of integrity rules of the form

$$\begin{array}{rl} \perp \text{ :- } & O, not(O). \\ \perp \text{ :- } & O, not(abd\_O). \\ \perp \text{ :- } & not(O), abd\_O. \end{array}$$

The *totality rules* (I_{total}) for $\mathcal{A}$ have the form

$$\begin{array}{rl} \perp \text{ :- } & not(defined_O) \\ defined_O \text{ :- } & O \\ defined_O \text{ :- } & not(O) \end{array}$$

for each $O \in \mathcal{H}_P$. □

Applying the transformations in Definition 3.10 allows Abdual to compute GSMs.

Theorem 3.4 *Let $\langle P, A, \mathcal{B}, I\rangle$ be an abductive scenario, let $S(P)$ be the set of shadow rules for P, and $A_{shadow} = \{abd\_O | not(abd\_O) \in S(P)\}$. Then*

1. *$M(\mathcal{B})$ is a generalized partial stable interpretation of $\langle P, A, I\rangle$ iff there exists an abductive solution*

$$\langle (P \cup S(P)), (A \cup A_{shadow}), \mathcal{B}, (I \cup I_{shadow})\rangle$$

such that $M(\sigma) = M(\mathcal{B})$.

2. *$M(\mathcal{B})$ is a generalized stable model of $\langle P, A, I\rangle$ iff there exists an abductive solution*

$$\langle (P \cup S(P)), (A \cup A_{shadow}), \mathcal{B}, (I \cup I_{shadow} \cup I_{consist} \cup I_{total}\rangle$$

such that $M(\sigma) = M(\mathcal{B})$.

Theorem 3.4 has several implications. First, since the paraconsistent well-founded model of a program is a partial stable interpretation, use of the shadow program and constraints includes computation of the paraconsistent well-founded model as a special case. In addition, because Theorem 3.1 states that ABDUAL can be used for query evaluation to abductive frameworks based on $WFSX_P$, ABDUAL can be used to compute generalized partial stable interpretations and generalized stable models. The cost of these computation, of course, includes the cost of potentially evaluating shadow rules and the various additional integrity constraints. It is known that the problem of deciding the answer to a ground query to an abductive framework is NP-complete when the entailment method is based on the well-founded semantics [8], as is the problem of deciding whether an abductive framework has a generalized stable model. The lack of polynomial data complexity of ABDUAL for arbitrary abductive frameworks is therefore understandable, given the power of these frameworks.

4 Discussion

A Meta-interpreter for ABDUAL and its Application to Diagnosis A preliminary meta-interpreter for ABDUAL, written using the XSB system, is available from `http://www.cs.sunysb.edu/~ tswift`. This meta-interpreter has the termination property of Theorem 3.2, but does not have the complexity property of Theorem 3.3. Work is underway on the XSB system so that the CO-UNFOUNDED SET REMOVAL operation can be implemented at the engine level and the complexity results of Theorem 3.3 attained.

ABDUAL was originally motivated by a desire to implement psychiatric diagnosis. Knowledge about psychiatric illnesses is codified by *DSM-IV* [7] sponsored by the American Psychiatric Association. Knowledge in DSM-IV can be represented as a directed graph with positive links to represent relations from diagnoses to sub-diagnoses or to symptoms. These graphs also have negative links, called *exclusion* links that represent symptoms or diagnoses that must shown false in order to derive the diagnosis. The DSM-IV graph requires both abduction and non-stratified negation, as can be seen by considering the diagnosis of Adjustment Disorder ([7], pg. 626). One criterion for this diagnosis is

Once the stressor (or its consequences) has terminated, the symptoms do not persist for more than an additional 6 months.

Thus, to diagnose a patient as presently undergoing adjustment disorder, a physician must hypothesize about events in the future — a step naturally modeled with abduction. Adjustment disorder requires an exclusion criterion

The stress-related disturbance does not meet the criteria for another specific Axis I disorder and is not merely an exacerbation of a preexisting Axis I or Axis II disorder.

that admits the possibility of a loop through negation between adjustment disorder and another diagnosis. This can in fact occur, for instance with Alzheimer's Dementia ([7], pg. 142-143). If, as far as a physician can tell, a patient fulfills all criteria for adjustment disorder besides the above criterion, as well as all criteria for Alzheimer's (besides the criterion that the disturbance is not better accounted for by another disorder), the physician will essentially be faced with the situation:

The patient has an Adjustment Disorder if he does not have Alzheimer's Dementia, and has Alzheimer's Dementia of the patient does not have an Adjustment Disorder.

It is unclear whether such a situation was envisioned by the DSM-IV committees, or how to model what a physician should do in such a situation. A physician to make a provisional diagnosis of the Adjustment Disorder based on a revision of his beliefs about Alzheimer's Dementia; alternatively a physician might choose Adjustment Disorder using a mechanism based on argumentation semantics [11]. Work has begun to use ABDUAL to implement psychiatric diagnosis for a commercial system, Diagnostica 2.0, which is being developed using the ABDUAL interpreter mentioned above.

Comparisons with Other Methods The use of dual programs to compute the well-founded semantics of normal programs was introduced in [14], but this method has several limitations compared with ABDUAL: it does not handle abduction or explicit negation; and it can have exponential complexity for some queries. Many of the definitions of ABDUAL are derived from SLG [2] (as reformulated in [16]) which computes queries to normal programs according to the well-founded semantics. For normal programs, ABDUAL shares the same finite termination and complexity properties as SLG. ABDUAL adds the capability to handle abduction, the use of the dual transformation for extended programs and the CO-UNFOUNDED SET REMOVAL operation, but ABDUAL does not allow evaluation of a non-ground program as does SLG. Unfortunately, performance tradeoffs of ABDUAL and SLG are not yet available, due to the lack of an engine-level implementation of the CO-UNFOUNDED SET REMOVAL operation of ABDUAL.

The main contribution of ABDUAL is its incorporation of abduction. We are not aware of any other efforts that have added abduction to a tabling method. Indeed, it is the use of tabling that is responsible for the termination and complexity results of Sections 3.1 and 3.2. Furthermore, ABDUAL evaluations are confluent in the sense that Theorem 3.1 holds for *any* ordering of applicable ABDUAL operations. The complexity and termination for WFS distinguishes ABDUAL from approaches such as the IFF proof procedure [9] and SLDNFA [6]. These approaches do allow variables in rules which ABDUAL does not. The methods of [3] and [10] compute abductive explanation based on some form of two-valued rule completion for non-abducible predicates (the former based on Clark's completion, and the latter based on the so-called transaction programs). This is similar to our use of the dual program[8]. In both methods, abductive explanations are computed by using the only-if part of the completion in a bottom-up fashion. However, both methods have a severe restriction on the class of programs: they apply generally only to acyclic programs. This restriction is due to their being based on completion. In contrast, ABDUAL is based on the well-founded semantics, and does not impose any restriction on cycles in programs.

Generalizing ABDUAL to Programs with Variables Generalizing ABDUAL for non-ground covered programs[9] with ground queries is not a difficult task: as in Clark's completion, consider rule heads with free variables, and explicitly represent unifications in the body; the dual is then obtained from these rules as usual, where the negation of `=` is `\=`. Allowing non-ground queries in covered programs can be obtained by considering as abducibles all terms of the form `X \= T`, and by adding an appropriate method for verifying consistency of sets of such inequalities. The most difficult step in order to fully generalize ABDUAL to deal with non-ground programs is to abandon the restriction of covered programs. This is so because free variables in the body of program rules introduce universally quantified variables in the body of rules in the dual program — a problem similar to that of floundering in normal programs. Work is underway to generalize ABDUAL to deal with non-ground noncovered programs using constructive negation methods.

A practical advantage of ABDUAL is that it allows the easy propagation of abducibles through both positive and negative literals. As an abductive answer is returned to an abductive subgoal, contexts can be immediately checked for consistency, regardless of whether the subgoal is positive or negative, and regardless of how many levels of negation were needed to produce the answer.

[8] Note that the dual for non-abducible predicates in acyclic programs is the same as the completion.

[9] A program is covered iff all variables appearing in the body of rules also appear in the corresponding head.

Acknowledgements This work was partially supported by NSF grants CCR-9702581, EIA-97-5998, and INT-96-00598. The authors also thank PRAXIS XXI projects MENTAL and ACROPOLE and FLAD-NSF project REAP for their support.

References

[1] J. Alferes, C. Damásio, and L. M. Pereira. A logic programming system for non-monotonic reasoning. *Journal of Automated Reasoning*, 14(1):93–147, 1995.

[2] W. Chen and D. S. Warren. Tabled Evaluation with Delaying for General Logic Programs. *JACM*, 43(1):20–74, January 1996.

[3] L. Console, D. Dupré, and P. Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.

[4] C. Damásio. *Paraconsistent Extended Logic Programming with Constraints*. PhD thesis, Univ. Nova de Lisboa, 1996.

[5] C. Damásio and L. M. Pereira. Abduction over 3-valued extended logic programs. In *LPNMR*, pages 29–42. Springer-Verlag, 1995.

[6] M. Denecker and D. De Schreye. SLDNFA: An abductive procedure for normal abductive programs. In *JICSLP*, pages 868–700. MIT Press, 1992.

[7] *Diagnostic and Statistical Manual of Mental Disorders*. American Psychiatric Association, Washington,DC, 4th edition, 1994. Prepared by the Task Force on DSM-IV and other committees and work groups of the American Psychiatric Association.

[8] Thomas Eiter, Georg Gottlob, and Nicola Leone. Abduction From Logic Programs: Semantics and Complexity. *Theoretical Computer Science*, 189(1-2):129–177, December 1997.

[9] T. Fung and R. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, 1997.

[10] K. Inoue and C. Sakama. Computing extended abduction through transaction programs. *Annals of Mathematics and Artificial Intelligence*, 1999. To appear.

[11] A. Kakas, R. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993.

[12] A. Kakas and P. Mancarella. Generalized stable models: A semantics for abduction. In *ECAI*, pages 385–391. Morgan-Kaufmann, 1990.

[13] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin Germany, 1984.

[14] L. M. Pereira, J. Aparício, and J. Alferes. Derivation procedures for extended stable models. In *International Joint Conference on Artificial Intelligence*, 1991.

[15] T.C. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *ACM PODS*, pages 11–21. ACM Press, 1989.

[16] T. Swift. A new formulation of tabled resolution with delay. In *EPIA 99*, 1999. Available at http://www.cs.sunysb.edu/~tswift.

[17] A. van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. *JACM*, 38(3):620–650, 1991.