Abduction over unbounded domains via ASP

Piero A. Bonatti[1]

Abstract. It is known that abduction can be embedded into Answer Set programming (ASP). This enables sophisticated answer set solvers to be applied to abduction problems. However this approach does not scale to abduction over infinite domains, nor to unbounded abduction of individual existence, due to well-known undecidability results. The approaches to open abduction usually rely on 3-valued semantics to overcome technical difficulties, but this approach changes the underlying semantics and prevents the application of ASP solvers. In this paper we apply the theory of finitary programs to prove that for an expressive and very interesting class of domain theories, ASP-based abduction with unbounded domains can effectively be computed. We also prove that each observable has a finite set of finite explanations representative of all the observable's infinitely many explanations. Moreover, the set of representative explanations can be computed with standard ASP engines.

1 INTRODUCTION

Answer Set Programming (ASP) is a declarative problem solving framework capable of modeling commonsense reasoning, reasoning about action and change, planning, as well as a variety of combinatorial problems within the first two levels of the polynomial hierarchy. ASP has some well-engineered and optimized implementations [16, 8], and is approaching the technological maturity needed by applications.

Abduction is one of the applications of ASP. Standard abduction frameworks [9, 14, 18] can be embedded into ASP with the same embedding into TMS used in [19]. In general, this technique does not scale to abduction frameworks with infinite domains, due to well-known undecidability results. The same applies to *open abduction*, that is, a form of abduction where the existence of new individuals can be abduced. Open abduction mechanisms that tolerate infinite domains in the presence of nonmonotonic negation exist [7] but they have to relax semantics to three-valued logic to make top-down, resolution-like procedures complete.

In this paper we show that ASP techniques can be applied to compute (two-valued) open abduction over possibly infinite domains. Our results are based upon the theory of *finitary programs* [4, 2], that is, a very expressive class of normal programs with function symbols and recursion such that stable model reasoning is effectively computable.

We prove that when the domain theory of the abduction framework is finitary, the set of minimal explanations of any given ground observation Q is finite, as well as any such explanation, despite the fact that the generalized stable models of the theory can be infinite. We derive that the set of minimal explanations is decidable. Moreover, we show how to embed open abduction into ASP, so that state-of-the-art ASP solvers can be used to compute explanations over finitary domain theories with unbounded, open domains.

The paper is organized as follows. In Section 2 we recall the basic definitions concerning open abduction and finitary programs. In Section 3 we show how any open abduction framework can be embedded into a normal logic program under the stable model semantics. The embedding contains function symbols, and hence in general its stable models are highly undecidable. In Section 4 we introduce finitary open abduction frameworks and study the properties of their sets of minimal explanations. Then, in Section 5, we refine the results of Section 3 to show how to use ASP solvers to compute a representative set of explanations over finitary open abduction frameworks.

2 PRELIMINARIES

We assume the reader to be familiar with normal logic programs and the stable model semantics [10]. We say that a normal logic program is *consistent* if it has at least one stable model.

The following presentation of open abduction frameworks is a syntactic variant of the approach followed in [3].

An *open abduction framework* is a triple $\langle T, A, Sk \rangle$, where T is a normal logic program (the domain theory), A is a set of *abducible predicates*, and Sk is an infinite denumerable set of (Skolem) constants, representing individuals whose existence can be abduced. The members of Sk do not occur in T.

For each open abduction framework $\langle T, A, Sk \rangle$, let $\mathsf{Abducibles}(T, A, Sk)$ be the set of all ground atoms $p(t_1, \ldots, t_n)$ such that $p \in A$ and each t_i is a term freely generated by the function and constant symbols occurring in T and Sk. $\mathsf{Abducibles}(T, A, Sk)$ will be called the set of *open abducibles* of the open abduction framework $\langle T, A, Sk \rangle$.

An *open generalized stable model* of an open abduction framework $\langle T, A, Sk \rangle$ is a stable model of $T \cup E$, for some $E \subseteq \mathsf{Abducibles}(T, A, Sk)$.

An *open explanation* of a closed sentence Q (called *observation*) w.r.t. an open abduction framework $\langle T, A, Sk \rangle$ is a set $E \subseteq \mathsf{Abducibles}(T, A, Sk)$ such that there exists a stable model M of $T \cup E$ that satisfies Q.

Next we introduce the preliminary definitions needed to define finitary programs.

The *(atom) dependency graph* of a program P is a labelled directed graph denoted by $DG(P)$, whose vertices are the ground atoms of P's language. Moreover, (i) there exists an edge labelled '$+$' (called positive edge) from B to A iff for some rule $R \in \mathsf{Ground}(P)$, $A \in head(R)$ and $B \in body(R)$; (ii) there exists an edge labelled '$-$' (called negative edge) from B to A iff for some rule $R \in \mathsf{Ground}(P)$, $A \in head(R)$ and $\neg B \in body(R)$.

An atom A depends positively (resp. negatively) on B if there is a directed path from B to A in the dependency graph with an even (resp. odd) number of negative edges. Moreover, each atom depends positively on itself. If A depends positively (resp. negatively) on B

[1] Università di Napoli Federico II, Napoli, Italy. E-mail: bonatti@na.infn.it

we write $A \geq_+ B$ (resp. $A \geq_- B$). We write $A \geq B$ if either $A \geq_+ B$ or $A \geq_- B$. If both $A \geq_+ B$ and $A \geq_- B$ hold then we write $A \geq_\pm B$.

By *odd-cycle* we mean a cycle in the dependency graph with an odd number of negative edges. A ground atom is *odd-cyclic* if it occurs in an odd-cycle.

In the context of normal logic programs, a *splitting set* for a program P [15] is a set of atoms U containing all the atoms occurring in the body of any rule $r \in \mathsf{Ground}(P)$ whose head is in U. The set of rules $r \in \mathsf{Ground}(P)$ whose head is in U—called the "bottom" of P w.r.t. U—will be denoted by $b_U(P)$. By $e_U(P, I)$ we denote the following partial evaluation of P w.r.t. $I \cap U$: remove from $\mathsf{Ground}(P)$ each rule $A \leftarrow L_1, \ldots, L_n$ such that some L_i containing an atom of U is false in I, and remove from the remaining rules all the L_i containing a member of U. The following is a specialization to normal programs of a result in [15].

Theorem 1 (Splitting theorem) *Let U be a splitting set for a normal logic program P. An interpretation M is a stable model of P iff $M = J \cup I$, where*

1. *I is a stable model of $b_U(P)$, and*
2. *J is a stable model of $e_U(\mathsf{Ground}(P) \setminus b_U(P), I)$.*

The next definitions and results, taken from [4, 2], characterize finitary programs.

Definition 2 [Finitary programs] We say a program P is *finitary* if the following conditions hold:

1. Each ground atom A in $DG(P)$ depends on finitely many ground atoms B. In other words, the cardinality of $\{B \mid A \geq B\}$ must be finite for all ground atoms A.
2. There are finitely many odd-cyclic atoms in $DG(P)$.

∎

Two examples of finitary programs can be found in figures 1 and 2. Many further examples of interesting finitary programs, showing that the above definition is compatible with a rather free use of function symbols and recursion, can be found in [4, 2]. Most finitary programs satisfy the first condition because the size of some arguments does not increase across recursive calls (this is also the case for the programs in figures 1 and 2). Typically, condition 2 is satisfied either because $DG(P)$ has only cycles with an even number of negative edges (figures 1 and 2), or because there exists a single odd-cycle defining a symbol f which is always false, by means of rules like $p \leftarrow f, \neg p$, where p occurs only within this rule.

The consequences of finitary programs can be computed by using only a finite fragment of their (potentially infinite) domain.

Definition 3 [Kernel atoms, Relevant universe and subprogram] A *kernel atom* for a normal program P and a ground formula Q is either an odd-cyclic atom or an atom occurring in Q (note that kernel atoms are ground by definition). The set of kernel atoms for P and Q is denoted by $K(P, Q)$.

The *relevant universe* for P and Q, denoted by $U(P, Q)$, is the set of all ground atoms B such that some kernel atom for P and Q depends on B. In symbols:

$$U(P, Q) = \{B \mid \text{for some } A \in K(P, Q),\ A \geq B\}\,.$$

The *relevant subprogram* for a ground formula Q (w.r.t program P), denoted by $R(P, Q)$, is the set of all rules in $\mathsf{Ground}(P)$ whose head belongs to $U(P, Q)$:

$$R(P, Q) = \{R \mid R \in \mathsf{Ground}(P) \text{ and } head(R) \in U(P, Q)\}\,.$$

∎

If P is finitary, then both $U(P, Q)$ and $R(P, Q)$ are finite and computable. Moreover, $R(P, Q)$ is all that is needed for query answering:

Lemma 4 *For all finitary normal programs P and all ground formulae Q, $R(P, Q)$ has a stable model M_Q iff P has a stable model M such that $M \cap U(P, Q) = M_Q$.*

Theorem 5 *For all finitary normal programs P and all ground formulae Q,*

1. *P credulously entails Q iff $R(P, Q)$ does.*
2. *P skeptically entails Q iff $R(P, Q)$ does.*

Since $R(P, Q)$ is finite, it follows that all the above reasoning tasks are decidable. Moreover, if Q is a quantifier-free nonground formula, then credulous and skeptical entailment are semi-decidable and Turing-equivalent.

In the rest of the paper we shall use other technical properties of relevant universes and subprograms: $U(P, Q)$ is a splitting set for P and $R(P, Q) = b_{U(P,Q)}(P)$. Moreover, if P is finitary, then the partially evaluated top part $e_{U(P,Q)}(\mathsf{Ground}(P) \setminus b_{U(P,Q)}(P), I)$ is consistent.

3 EMBEDDING OPEN ABDUCTION INTO ASP

Open abduction can be embedded into ASP. A preliminary approach covering function-free and constant-free domain theories can be found in [1]; here we extend that approach to arbitrary domain theories. Moreover, [1] deals only with the complexity of checking whether an explanation exists, while in the next sections we show how to obtain explanations, and prove that the set of minimal explanations can be finitely presented.

Each open abductive framework $\langle\, T, A, Sk \,\rangle$ can be captured by a normal logic program $\Pi(T, A, Sk)$ defined as follows. For each predicate symbol $p \in A$, introduce a new distinct predicate symbol $\bar{p}$. Moreover, let V, $\bar{V}$, and U be new predicate symbols distinct from the symbols $\bar{p}$, and let c and s, respectively, be a constant and a unary function not occurring in T. $\Pi(T, A, Sk)$ consists of the following rules, for all rules $H \leftarrow Body$ in T, for all n-ary function symbols f occurring in T or Sk, and for all $p \in A$:

$$H \leftarrow Body, U(x_1), \ldots, U(x_n) \qquad \text{where } x_1 \ldots x_n \text{ are the variables of } H \leftarrow Body$$
$$U(f(x_1, \ldots, x_n)) \leftarrow U(x_1), \ldots, U(x_n)$$
$$U(x) \leftarrow V(x)$$

$$V(c) \leftarrow \neg \bar{V}(c)$$
$$\bar{V}(c) \leftarrow \neg V(c)$$
$$V(s(X)) \leftarrow V(X), \neg \bar{V}(s(X))$$
$$\bar{V}(s(X)) \leftarrow \neg V(s(X))$$

$$p(x_1, \ldots, x_n) \leftarrow \neg \bar{p}(x_1, \ldots, x_n), U(x_1), \ldots, U(x_n)$$
$$\bar{p}(x_1, \ldots, x_n) \leftarrow \neg p(x_1, \ldots, x_n), U(x_1), \ldots, U(x_n)$$

Note that the possible extensions of predicate V under the stable model semantics are the initial segments of the infinite sequence

$\Sigma = c, s(c), s(s(c)), \ldots$ (including the empty sequence and Σ). Intuitively, the extension of V represent the subset of Sk that has been abduced. Under this interpretation, the extension of U models the universe of the program $T \cup E$ underlying the definition of open generalized stable models. The last two rules nondeterministically choose E [19, 11], while T is taken into account by the first rule schema in the program.

Then it can be proved that the stable models of the above embedding projectively characterize the explanations of any given query under $\langle\, T, A, Sk \,\rangle$. More precisely, given a stable model M of $\Pi(T, A, Sk)$, the new atoms must be filtered away, and each new term $s^k(c)$ must be matched with a suitable element of F.

Let a *renaming function* for $\langle\, T, A, Sk \,\rangle$ be an injective substitution, preserving the symbols in T and A, and mapping the new terms occurring in Σ onto elements of Sk. Finally, let $M|_{T,A}$ be the set of atoms of M whose predicate symbol occurs in T or A (equivalently, $M|_{T,A}$ is obtained by removing from M all the atoms with one of the new predicate symbols introduced in $\Pi(T, A, Sk)$).

Theorem 6 *If M is a stable model of $\Pi(T, A, Sk)$ then for all renaming functions ρ there exists an open generalized stable model M' of $\langle\, T, A, Sk \,\rangle$ such that*

1. *$\rho(M(V))$ (the image of ρ restricted to the extension of V in M) equals the subset of Sk actually occurring in M'.*
2. *$\rho(M|_{T,A}) = M'$.*

Conversely, if M' is an open generalized stable model of $\langle\, T, A, Sk \,\rangle$, then there exist a stable model M of $\Pi(T, A, Sk)$ and a renaming function ρ such that 1 and 2 hold.

4 FINITARY ABDUCTION FRAMEWORKS

In the previous section we proved that open abduction can be embedded into ASP, but we said nothing about decidability. Note that the embedding's domain contains at least one function symbol (s). So it remains to be seen how to deal effectively with an infinite Herbrand domain. For this purpose we focus our attention on finitary domain theories.

Definition 7 An open abduction framework $\langle\, T, A, Sk \,\rangle$ is *finitary* iff T is finitary. ∎

Note that adding facts to a normal logic program does not change its dependency graph. Then the following proposition holds.

Proposition 8 *For all open abduction frameworks $\langle\, T, A, Sk \,\rangle$ and all $E \subseteq \mathsf{Abducibles}(T, A, Sk)$, $T \cup E$ is finitary iff T is finitary.*

In particular, if an open abduction framework $\langle\, T, A, Sk \,\rangle$ is finitary, then all possible programs $T \cup E$ are finitary, too.

Theorem 9 *For all open abduction frameworks $\langle\, T, A, Sk \,\rangle$ and all ground observations Q,*

1. *If E is an explanation of Q w.r.t. $\langle\, T, A, Sk \,\rangle$, then there exists an explanation $E' \subseteq E \cap U(T, Q)$ of Q w.r.t. $\langle\, R(T, Q), A, Sk \,\rangle$.*
2. *If $\langle\, T, A, Sk \,\rangle$ is finitary , then every explanation E of Q w.r.t. $\langle\, R(T, Q), A, Sk \,\rangle$ is also an explanation of Q w.r.t. $\langle\, T, A, Sk \,\rangle$.*

Proof.(Sketch)

1) Suppose E is an explanation of Q w.r.t. $\langle\, T, A, Sk \,\rangle$. By definition, there exists a stable model M of $T \cup E$ such that $M \models Q$. Recall that $U(T \cup E, Q)$ is a splitting set for $T \cup E$, and $R(T \cup E, Q) = b_{U(T \cup E, Q)}(T \cup E)$; then $M = I \cup J$, where I is a stable model of $R(T \cup E, Q)$ and $J \cap U(T \cup E, Q) = \emptyset$. By the latter equality, $I \models Q$. Next note that $U(T \cup E, Q) = U(T, Q) \cup E'$, for some $E' \subseteq E \cap U(T, Q)$, and hence I is a generalized stable model of $\langle\, R(T, Q), A, Sk \,\rangle$ satisfying Q. Part 1) follows immediately.

2) Suppose E is an explanation of Q w.r.t. $\langle\, R(T, Q), A, Sk \,\rangle$, that is, there exists a stable model I of $R(T, Q) \cup E$ such that $I \models Q$. Since $\langle\, T, A, Sk \,\rangle$ is finitary, the upper part $e_{U(T,Q)}(\mathsf{Ground}(T \cup E) \setminus b_{U(T \cup E, Q)}(T \cup E), I)$ is consistent, that is, it has a stable model J. Recall that $R(T \cup E, Q) = b_{U(T \cup E, Q)}(T \cup E)$, therefore $I \cup J$ is a stable model of $T \cup E$, by the splitting theorem. Moreover, since $I \models Q$, E is an explanation of Q w.r.t. $\langle\, T, A, Sk \,\rangle$. ∎

The first corollary tells us that minimal explanations can be computed by means of the relevant subprogram $R(T, Q)$ (that is, a fragment of the ground instantiation of the domain theory T).

Corollary 10 *For all ground observations Q and all finitary $\langle\, T, A, Sk \,\rangle$, E is a minimal explanation of Q w.r.t. $\langle\, T, A, Sk \,\rangle$ iff E is a minimal explanation of Q w.r.t. $\langle\, R(T, Q), A, Sk \,\rangle$.*

The second corollary follows from the property that $R(T, Q)$ is finite when T is finitary.

Corollary 11 *If $\langle\, T, A, Sk \,\rangle$ is finitary, then for all ground observations Q:*

1. *The minimal explanations of Q are finite;*
2. *Q has finitely many minimal explanations;*
3. *The set of minimal explanations of Q is decidable.*

5 COMPUTING EXPLANATIONS WITH ASP

The above results do not tell us how to search for explanations. We would like ASP solvers to perform such search. For this purpose, we refine the relationships between open abduction frameworks and the embedding into ASP.

We start by proving that if an open abductive framework is finitary, then also the corresponding normal program $\Pi(T, A, Sk)$ is finitary.

Proposition 12 *Let W be the set of ground atoms $U(t)$, $V(t)$ and $\bar{V}(t)$ occurring in $\mathsf{Ground}(\Pi(T, A, Sk))$.*

1. *W is a splitting set for $\Pi(T, A, Sk)$.*
2. *$b_W(\Pi(T, A, Sk))$ is finitary.*
3. *If $\langle\, T, A, Sk \,\rangle$ is finitary, then $\mathsf{Ground}(\Pi'(P, F, O)) \setminus b_W(\Pi(T, A, Sk))$ is finitary.*

From the above proposition and [1, Lemma 2] (stating that if $b_U(P)$ and $\mathsf{Ground}(P) \setminus b_U(P)$ are finitary then P is finitary) we obtain:

Theorem 13 *If $\langle\, P, F, O \,\rangle$ is a finitary open abduction framework then its embedding $\Pi(T, A, Sk)$ is a finitary normal program.*

It turns out that the relationships between open generalized stable models and the stable models of the ASP embedding formulated in Theorem 6 carry over to relevant subprograms.

Then, by Theorem 9, we obtain the following result.

Theorem 14 *Suppose $\langle\, T, A, Sk \,\rangle$ is finitary. Then for all minimal explanations E of Q w.r.t. $\langle\, T, A, Sk \,\rangle$ there exist an explanation $E' \supseteq E$ of Q w.r.t. $\langle\, T, A, Sk \,\rangle$, a stable model M of $R(\Pi(T, A, Sk), Q)$ and a renaming function ρ such that $E' = \rho(M) \cap \mathsf{Abducibles}(T, A, Sk)$.*

In other words, a set of explanations covering all minimal explanations of the given observables Q can be obtained by running an ASP solver on the finite, ground normal program $R(\Pi(T, A, Sk), Q)$. From the set of stable models generated by the solver, one can easily extract individual explanations by projecting each model onto $\mathsf{Abducibles}(T, A, Sk)$. In order to generate skeptical consequences it may be profitable to adopt skeptical methods [12, 6] (it is not clear if the approach of [12] can handle unbounded domains spontaneously; [6] can).

The extended version of [4] describes an algorithm for computing the relevant subprogram. A prototype implementation in XSB Prolog is available on the author's home page. A prototype finitary program recognizer is available, too. The recognizer is described in [5].

We conclude this section with two examples of abduction frameworks whose domain theory T is finitary. The first domain theory illustrated in Figure 1 is a finitary program for model-based circuit diagnosis. Each atom $\mathtt{out}(\mathtt{G})$ models the output of gate `G`. The values of circuit inputs are specifed through the same predicate. For example the fact $\mathtt{out}(\mathtt{y2})$ states that the value of input `y2` is 1. The rules model the behavior of each gate, both under the assumption that the gate is working properly ($\neg\mathtt{ab}(G)$), and under the assumption that a fault exists ($\mathtt{ab}(G)$). This formalization makes the common assumption that in the absence of observable faults gates are not faulty. This program is stratified, so there are no odd-cycles. The set of abducible predicates is $A = \{\mathtt{ab}\}$.

The second domain theory, illustrated in Figure 2, can be used to find sequences of events that explain sequences of observations by abducing the predicate `do`. Most atoms with time argument $\mathtt{T}+1$ depend on atoms with time argument `T`. So the only source of cycles with negative edges are the rules for nondeterministic actions (predicates `fails` and `succeeds`). However, these are not odd-cycles.

An open version of this example (where the existence of new individuals can be abduced) can produce explanations based on blocks and actions not explicitly mentioned in the domain theory.

Note the advantages of using function symbols and recursion. Infinite and structured domains (such as the set of all gates and time) can be modelled directly, in a natural way. An infinite number of problem instances can be encoded in one program, so there is no need for external components to build a specific encoding for each instance. More generally, all necessary pre- and post-processing can be carried out within the same language, and in principle such auxiliary computations can be interleaved naturally with proper problem solving activities. Once all instances are simultaneously encoded in one program, one can submit queries across (and relating) multiple instances. For further details on these topics and their impact on potential optimizations and expressiveness, see [4, 2].

6 SUMMARY AND CONCLUSIONS

Two-valued open abduction over infinite domains is effectively computable when the domain theory is finitary. We proved that for all finitary open abduction frameworks and all ground observables, a representative set of explanations (minimal explanations) is decidable and finitely presentable. Moreover, we showed how to use ASP solvers to compute a set of finite explanations covering all minimal ones. These results open the way to open abduction through ASP techniques.

One limitation of Theorem 14 is that the explanations obtained are not necessarily minimal. Theorem 14 guarantees only that each minimal explanation is contained in some of the explanations produced with $R(\Pi(T, A, Sk), Q)$. In this sense, the ASP-based approach proposed here produces a representative set of explanations. It would be interesting to see how to minimize explanations via ASP.

Finitary programs, in principle, allow a number of optimizations [4]. Identifying suitable such techniques for open abduction is an interesting subject for further research. Another interesting topic for future research is a precise estimate of the computational complexity of open abduction for finitary open abduction frameworks. Such results would precisely determine the expressiveness of this kind of abduction.

Acknowledgements

The work reported in this paper is partially supported by the European Community within the Fifth (EC) Framework Programme under contract IST-FET-2001-37004 – WASP working group.

REFERENCES

[1] P. A. Bonatti. Finitary Open Logic Programs. *Proc. of the ASP'03 Workshop*, 2003.

[2] P. A. Bonatti. Reasoning with infinite stable models II: Disjunctive programs. *Proc. of ICLP'02*, LNCS 2401, 333-346, Springer, 2002.

[3] P. A. Bonatti. Abduction, ASP and open logic programs. *Proc. of NMR'02*, Toulouse, 2002.

[4] P. A. Bonatti. Reasoning with infinite stable models. *Artificial Intelligence*, to appear. http://people.na.infn.it/~bonatti/pub/aij04.ps. Preliminary version in *Proc. of IJCAI*, Morgan Kaufmann, 2001.

[5] P.A. Bonatti. Prototypes for reasoning with infinite stable models and function symbols. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001*, volume 2173 of *LNCS*, pages 416–419. Springer, 2001.

[6] P.A. Bonatti. Resolution for Skeptical Stable Model Semantics. *Journal of Autom. Reasoning* 27(4): 391-421, 2001.

[7] D. Denecker, D. De Schreye. SLDNFA: An abductive procedure for abductive logic programs. *The Journal of Logic Programming*, 34(2):111-167, 1998.

[8] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A deductive system for non-monotonic reasoning. In *Logic Programming and Nonmonotonic Reasoning, 4th International Conference, LPNMR'97, Proceedings*, volume 1265 of *LNCS*, pages 364–375. Springer, 1997.

[9] K. Eshghi, R.A. Kowalski. Abduction compared with negation as failure. In *Proc. of the 6th Int.l Conf. on Logic Programming*, MIT Press, 1989.

[10] M. Gelfond, V. Lifschitz. The stable model semantics for logic programming. In *Proc. of the 5th ICLP*, pp.1070-1080, MIT Press, 1988.

[11] K. Inoue. Extended Logic Programs with Default Assumptions. In *Proc. of the Int.l Conf. on Logic Programming* (ICLP'91), 490-504, MIT Press, 1991.

[12] K. Inoue, C. Sakama. A fixpoint characterization of abductive logic programs. *Journal of Logic Programming* 27(2):107-136, 1996.

[13] A.C. Kakas, R.A. Kowalski, F. Toni. Abductive Logic Programming. *Journal of Logic and Computation*, 2(6):719-770, 1992.

[14] A.C. Kakas, P. Mancarella. Generalized Stable Models: a semantics for abduction. *Proc. of ECAI'90*, pp.385-391, 1990.

[15] V. Lifschitz, H. Turner. Splitting a logic program. In *Proc. ICLP'94*, pp.23-37, MIT Press, 1994.

[16] I. Niemelä, P. Simons. Smodels - an implementation of the stable model and well-founded semantics for normal LP. In J. Dix, U. Furbach, A. Nerode (eds.), *Logic Programming and Nonmonotonic Reasoning: 4th international conference, LPNMR'97*, LNAI 1265, Springer Verlag, Berlin, 1997.

[17] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36(1):27-47, 1988.

[18] K. Satoh, N. Iwayama. A query evaluation method for abductive logic programming. In *Proc. of the Joint Int.l Conf. and Symposium on Logic Programming* (JICSLP'92), 671-685, MIT Press, 1992.

[19] K. Satoh, N. Iwayama. Computing abduction by using the TMS. In *Proc. of the Int.l Conf. on Logic Programming* (ICLP'91), 505-518, MIT Press, 1991.

[20] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proc. of IJCAI'89*, 1055-ff, 1989.

```
out(and(X,Y)) ← out(X), out(Y), ¬ab(and(X,Y))
out(and(X,Y)) ← ab(and(X,Y)), ¬out(X)
out(and(X,Y)) ← ab(and(X,Y)), ¬out(Y)

out(or(X,Y)) ← out(X), ¬ab(or(X,Y))
out(or(X,Y)) ← out(Y), ¬ab(or(X,Y))
out(or(X,Y)) ← ab(or(X,Y)), ¬out(X), ¬out(Y)

out(not(X)) ← ¬out(X), ¬ab(not(X))
out(not(X)) ← ab(not(X)), out(X)

out(y2)    /* other inputs implicitly negated */
```

Figure 1. Model based diagnosis

```
/* Frame axiom */
holds(P,T+1) ← holds(P,T), ¬ab(P,T+1)

/* Sample deterministic action */
holds(on_top(A,B),T+1) ←
                    do(put_on(A,B),T),        /* action */
                    holds(is_clear(B),T),     /* preconds */
                    holds(in_hand(A),T)

/* Sample nondeterministic action */
holds(in_hand(B),T+1) ←
                    do(grasp(B),T),           /* action */
                    holds(is_clear(B),T),     /* preconds */
                    ¬fails(grasp(B),T)

holds(on_table(B),T+1) ←
                    do(grasp(B),T),           /* action */
                    holds(is_clear(B),T),     /* preconds */
                    fails(grasp(B),T)

ab(on_top(B,C),T+1) ←
                    do(grasp(B),T),           /* action */
                    holds(is_clear(B),T)      /* preconds */

fails(grasp(B),T) ← ¬succeeds(grasp(B),T)
succeeds(grasp(B),T) ← ¬fails(grasp(B),T)
```

Figure 2. Reasoning about actions