A Consequence Finding Approach for Full Clausal Abduction

Oliver Ray[1] and Katsumi Inoue[2]

[1] University of Bristol, United Kingdom
oray@cs.bris.ac.uk
[2] National Institute of Informatics, Japan
ki@nii.ac.jp

Abstract. Abductive inference has long been associated with the logic of scientific discovery and automated abduction is now being used in real scientific tasks. But few methods can exploit the full potential of clausal logic and abduce non-ground explanations with indefinite answers. This paper shows how the consequence finding method of Skip Ordered Linear (SOL) resolution can overcome the limitations of existing systems by proposing a method that is sound and complete for finding minimal abductive solutions under a variety of pruning mechanisms. Its utility is shown with an example based on metabolic network modelling.

1 Introduction

The importance of abductive inference in scientific discovery was recognised by C.S. Peirce over a century ago. He saw abduction as a form of reasoning, from known effects to possible causes, which underlies the process of hypotheses formation [6]. Recent advances in Artificial Intelligence have shown that (some aspects of) abductive logic can be automated and exploited in real domains like diagnosis [10, 4] and bioinformatics [21, 17]. The benefit of logic-based methods is their ability to use prior background knowledge and return meaningful testable hypotheses. For example, in [12], abduction was used to infer functional genomic hypotheses that were experimentally tested by a 'Robot Scientist'. The abductive reasoning used in these tasks all follow a simple but very useful logical pattern: given a theory T, a goal G, and a set of possible assumptions A, find a minimal consistent subset of A which can be added to T in order to ensure that (some instance of) G is satisfied.

In general, T, G and the elements of A are arbitrary first order formulae and an abductive solution consists of two parts: a set of formulae Δ, called an *explanation*, stating which assumptions in A should be added to T; and a set of substitutions Θ, called an *answer*, stating which instances of the free variables in G are satisfied. By utilising standard normalisation techniques, it suffices to consider the case when T is clausal theory, G is a conjunction of literals, and A is a set of literals called *abducibles*. But, existing approaches for abduction typically impose additional restrictions that rule out the possibility of unrestricted abductive reasoning in full clausal logic. In particular, most insist

V. Corruble, M. Takeda, and E. Suzuki (Eds.): DS 2007, LNAI 4755, pp. 173–184, 2007.

that either the goal or abducibles should be ground, and most only allow Horn clauses to appear in the theory.

The restrictions imposed by existing abductive systems reflect certain issues arising in non-Horn logic. Many use input resolution methods, which are not complete in general and cannot handle indefinite answers [20]. Moreover, non-ground abducibles complicate the relationship between variables in the query and explanation. Finally, the possibility of non-ground explanations and indefinite answers makes it harder to compare alternative solutions and define appropriate notions of minimality. A great deal of recent work in abduction has focussed on the framework of logic programming [11], which uses Negation as Failure (NAF) to avoid the need for non-Horn reasoning. But these approaches inherit a number of more serious logical concerns regarding the semantics of NAF, the difficulty of floundering, and the complexity of reasoning in a non-monotonic formalism where standard pruning strategies are inapplicable.

This paper gives a sound and complete method for finding minimal abductive answers in full clausal (classical) logic that overcomes the syntactic restrictions of existing systems. By treating abduction as a form of conditional query answering, we propose a semantics for abduction and minimality that correctly handles indefinite answers and non-ground explanations. We present a proof procedure, based on the clausal consequence finding approach of Skip Ordered Linear (SOL) resolution [7], which includes a rule for 'skipping' or assuming literals during a proof. To do this, we introduce a method for lifting a previous limitation on the language bias for specifying skipped literals that would otherwise prevent us from using arbitrary sets of abducibles. Previous results ensure our approach is sound and complete under a combination of efficient pruning strategies [8, 9].

The paper is structured as follows. Section 2 gives the relevant notation and background material on abduction, consequence finding and SOL. Section 3 formalises the semantics of minimal abduction in full clausal logic. Section 4 presents our abductive procedure. Section 5 compares our approach with related work. The paper concludes with a summary and directions for future work.

2 Background

2.1 Notation and Terminology

This paper assumes a first-order language $\mathcal{L}$ (without equality) containing the connectives $\wedge$, $\vee$, $\neg$, $\leftarrow$, $\rightarrow$, $\leftrightarrow$, logical constants $\top$, $\bot$, and quantifiers $\forall$, $\exists$. It also assumes standard first-order entailment $\models$ and equivalence $\equiv$ relations whose semantics is purely classical (and not restricted to Herbrand models). The term 'iff' abbreviates 'if and only if' and the term 'wrt' abbreviates 'with respect to'. A literal L is either an atom A or its negation $\neg A$. The complement of L, denoted $\overline{L}$, is defined as $\neg A$ (resp. A) if $L = A$ (resp. $\neg A$). A maximally general literal is one, e.g., $p(X, Y, Z)$, whose arguments are distinct variables. If S is a set of literals then $\bigwedge_{L \in S} L$ (resp. $\bigvee_{L \in S} L$) denotes the conjunction (resp. disjunction) of the literals in S and is defined as $\top$ (resp. $\bot$) when S is empty. A clause C is a disjunction of literals $L_1 \vee \ldots \vee L_m$ that, for convenience, will

often be identified with the set $\{L_i|1 \leq i \leq m\}$ of its disjuncts. As usual, any free variables are implicitly universally quantified at the front of the clause. A clause is Horn iff it has at most one positive literal and is full otherwise. A clause is a tautology iff it contains a literal L and its complement $\overline{L}$. The empty clause is denoted $\Box$. A variable binding is an expression of the form X/t where X is a variable and t is a term. In this case, we say that X is bound to t. A substitution is a set of variable bindings for distinct variables. The application of a substitution σ to an expression E is written $E\sigma$ and denotes the expression obtained from E by (simultaneously) replacing each free variable X by the corresponding term t for each binding X/t in σ. For any (set of) expressions E, let $Inst(E)$ denote the set of all instances of (members of) E. A clause D subsumes C, written $D \geq C$, iff D has no more literals than C and there is a substitution θ such that $D\theta \subseteq C$. Moreover D properly subsumes C, written $D > C$, iff $D \geq C$ and $C \not\geq D$. A theory T is a conjunction of clauses $C_1 \wedge \ldots \wedge C_n$ that, for convenience, will often be identified with the set $\{C_i|1 \leq i \leq n\}$ of its conjuncts. A theory is Horn iff all of its clauses are Horn and is full otherwise. For any theory T, let $\mu(T)$ denote the theory obtained from T by removing all clauses properly subsumed by another clause in T, and let $Th(T)$ denote the set of all clauses logically entailed by T. A goal G is a conjunction of literals. An underscore '_' is sometimes used to denote an anonymous variable.

2.2 Abduction

Abduction is an established AI technique for hypothetical reasoning [11]. In essence, abduction computes the conditions under which a goal G follows from a given theory T. Implicitly, G is understood as an existentially quantified query asking "is some instance of G satisfied in an extension of T"? If so, then the abductive computation should succeed, returning a set of assumptions Δ which must be added to T and a substitution σ stating which instances of the free variables in G are entailed. The assumptions in Δ are usually restricted to the instances of a set A of literals, called abducibles. Intuitively, these are literals whose truth is not specified in the intended domain: e.g., potential faults in a diagnosis task and possible actions in a planning problem. Each explanation Δ is implicitly understood as an existentially quantified conjunction that should be consistent with T and should be minimal in the sense of not containing atoms which could be removed to leave a smaller explanation Δ'. These notions are typically formalised in the literature as shown in Definition 1 below.[1]

Definition 1 (Naive Abduction). *Let T be a theory, G be a goal, and A be a set of literals. A (naive) abductive solution (for G wrt T and A) consists of a set of literals $\Delta \subseteq Inst(A)$ (called an explanation) and a substitution σ for G (called an answer) such that (i) $T \wedge \Delta \models G\sigma$ and (ii) $T \wedge \Delta \not\models \bot$. The explanation Δ is minimal iff there is no other explanation Δ' such that $\Delta' \subset \Delta$.*

[1] Analogous characterisations are obtained for logic program formalisms by replacing classical entailment with some appropriate completion or preferred model semantics. For convenience, we assume integrity constraints IC are included in the theory T.

2.3 Consequence Finding

Consequence finding is a general reasoning technique for computing the logical theorems entailed by a set of axioms [7]. Since the deductive closure of a logical theory may be infinite, it is generally infeasible or undesirable to compute all possible consequences. It is often more useful to consider a refinement of this task where it is required to compute the clausal consequences of a given theory that satisfy a given vocabulary and are minimal with respect to subsumption. If the vocabulary is specified by a form of language bias called a production field, the resulting consequences are known as characteristic clauses [7]. A production field P is a pair $\langle L, Cond \rangle$, where L is a set of literals and $Cond$ is a certain condition to be satisfied. When $Cond$ is not specified, P is just written as $\langle L \rangle$. A clause C is said to belong to $P = \langle L, Cond \rangle$ iff (i) every literal in C is an instance of a literal in L and (ii) C satisfies $Cond$. A production field P is stable iff, for any two clauses C and D such that $C \geq D$, the clause D belongs to P only if C belongs to P. If $\mathcal{L}_P$ denotes the set of clauses that belong to P, then the characteristic clauses of a theory T with respect to P are the set of clauses $Carc(T, P) = \mu(Th(T) \cap \mathcal{L}_P)$. The importance of these notions lies in the fact that many reasoning tasks, such as abduction, induction, and theorem proving, can be reduced to the computation of characteristic clauses [7].

2.4 SOL Resolution

Characteristic clauses can be computed a procedure called SOL resolution [7], which can be seen as extending the Model Elimination [14] calculus with a rule for 'skipping' literals. Intuitively, skipped literals represent assumptions that are needed for a proof to succeed. As explained in [7], this feature is needed to ensure the completeness of SOL for consequence finding. SOL deductions are defined using the notion of a structured clause, which is a pair $\langle A, B \rangle$ consisting of two clauses A and B, where the latter may contain so-called framed literals of the form $\boxed{L}$ denoting previously resolved upon literals. SOL deductions are formalised in Definition 2, which is recalled from [8].

Definition 2 (SOL Deduction). *Let T be a theory, S be a clause, and P be a production field. An SOL deduction of S from T and P (of length n) is a sequence of structured clauses $D_0, \ldots, D_n$ satisfying rules 1-6 below.*

1. *$D_0 = \langle \Box, C \rangle$ for some clause $C \in T$.*
2. *$D_n = \langle S, \Box \rangle$.*
3. *For each $D_i = \langle A_i, B_i \rangle$ clause $A_i \cup B_i$ is not a tautology.*
4. *For each $D_i = \langle A_i, B_i \rangle$ clause B_i is not subsumed by any B_j with the empty substitution, where $D_j = \langle A_j, B_j \rangle$ is a previous structured clause with $j < i$.*
5. *For each $D_i = \langle A_i, B_i \rangle$ clause A_i belongs to P.*
6. *$D_{i+1} = \langle A_{i+1}, B_{i+1} \rangle$ is obtained from $D_i = \langle A_i, B_i \rangle$ as follows:*
   (a) *let L be the left-most literal of B_i. Then A_{i+1} and R_{i+1} are obtained by applying one of the rules:*

i. Skip: if $A_i \cup \{L\}$ belongs to P, then $A_{i+1} = A_i \cup \{L\}$ and R_{i+1} is the clause obtained by removing L from B_i.

ii. Resolve: if there is a clause E_i from $T \cup \{C\}$ such that $\neg K \in E_i$ and L and K have a most general unifier θ, then $A_{i+1} = A_i\theta$ and R_{i+1} is the clause obtained by concatenating $E_i\theta$ and $B_i\theta$, framing $L\theta$, and removing $\neg K\theta$.

iii. Factoring: A_i or B_i contains an unframed literal K such that L and K have a most general unifier θ, then $A_{i+1} = A_i\theta$ and R_{i+1} is obtained from $B_i\theta$ by deleting $L\theta$.

iv. Reduction: B_i contains a framed literal $\boxed{\neg K}$, and L and K have a most general unifier θ, then $A_{i+1} = A_i\theta$ and R_{i+1} is obtained from $B_i\theta$ by deleting $L\theta$.

(b) B_{i+1} is obtained from R_{i+1} by deleting every framed literal not preceded by an unframed literal in the remainder (truncation).

It has been shown $Carc(T, P)$ is equal to the set of subsume-minimal clauses derivable by SOL from T and P [7]. Moreover, an efficient implementation of SOL has been developed [16] that uses several pruning mechanisms [9] to further constrain SOL deductions. These include mandatory rules native to SOL, such as merge and regularity for skipped literals, as well as some generic theorem proving methods, such as order preserving reduction, lemma matching, and local failure caching. It has been shown that these pruning strategies do not compromise the completeness of SOL if the production field P is stable [9].

3 Full Clausal Abduction

This section shows full clausal abduction can be efficiently realised by SOL resolution. Section 3.1 presents a semantics for abduction that correctly handles non-ground abducibles and disjunctive answers. It then shows how computing minimal abductive solutions can be reduced to the computation of characteristic clauses. Section 3.2 explains why unstable production fields arise in abductive problems and how this leads to the incompleteness of SOL. It then shows how to overcome this limitation by means of an efficient program transformation.

3.1 Semantics

The naive formulation of abduction in Definition 1 is only satisfactory when G and Δ are ground. Even though the implicit existential quantification on G and Δ correctly specifies if the goal should succeed, it does not adequately constrain the returned solutions. First, just like the answers returned by standard Prolog systems [13], every instance of the computed answer $G\sigma$ should be entailed by T (and Δ). Otherwise it suffices to simply take $\sigma = \emptyset$. Second, as in other non-Horn extensions of Prolog [20], it is necessary to return a *set* of answer substitutions to account for indefinite answers. For example, the goal $q(X)$ should succeed from the theory $q(a) \vee q(b)$ with answer $\{\{X/a\}, \{X/b\}\}$, indicating that X must be bound to either a or b in any model of T.

All of these issues are resolved in Definition 3 below, which gives a general clausal formalisation of abduction.[2] In effect, this formulation treats abduction as a type of conditional query answering [3, 8] with the explanation Δ denoting the assumptions under which the answer σ is valid. Intuitively, in order for $\langle \Delta, \Theta \rangle$ to solve G wrt T, condition (i) states that the conjunction of assumed literals L in Δ must imply the disjunction of answers obtained by applying each substitution σ in Θ to G. All variables are universally quantified at the front of the implication. Analogously, the consistency condition (ii) can be viewed as saying that Δ should not be an explanation for the contradictory goal $G = \bot$.

Definition 3 (General Abduction). *Let T be a theory, G be a goal, and A be a set of (abducible) literals. An abductive solution (for G wrt T and A) is a pair $\langle \Delta, \Theta \rangle$ consisting of a set of literals $\Delta \subseteq Inst(A)$ (called an explanation) and a set of substitutions Θ for G (called an answer) such that*

$$\textit{(i) } T \models \forall \left(\bigwedge_{L \in \Delta} L \rightarrow \bigvee_{\sigma \in \Theta} G\sigma \right) \textit{ and (ii) } T \not\models \forall \left(\bigwedge_{L \in \Delta} L \rightarrow \bot \right)$$

Example 1. Let T, G and A be as defined below. Theory T says there is a metabolic pathway from X to Z if there is a reaction from X to Y and a pathway from Y to Z; or if there is a reaction from X to Z. It also says there is a reaction from a to either b or c; and a reaction from either b or c to d; but no reaction from c to b. Goal G asks "from which metabolites U is there a pathway to d"? Abducibles A allow all instances of the predicate *reaction* to be assumed.

$$T = \left\{ \begin{array}{l} \neg reaction(X,Y) \vee \neg pathway(Y,Z) \vee pathway(X,Z) \\ \neg reaction(Y,Z) \vee pathway(Y,Z) \\ reaction(a,b) \vee reaction(a,c) \\ reaction(b,d) \vee reaction(c,d) \\ \neg reaction(c,b) \end{array} \right\}$$

$$G = \{\, pathway(U,d) \,\}$$

$$A = \{\, reaction(V,W) \,\}$$

From the 4th clause in T, there is a pathway from b or c to d: i.e., $\langle \Delta = \emptyset, \Theta = \{\{U/b\}, \{U/c\}\} \rangle$ is a solution. Moreover, there is a pathway from b to d if we assume a reaction from b to c: i.e., $\langle \Delta = \{reaction(b,c)\}, \Theta = \{\{U/b\}\} \rangle$ is also a solution. But, $\langle \Delta = \{reaction(c,b)\}, \Theta = \{\{U/c\}\} \rangle$ is not valid as it contradicts the 5th clause in T. However, there is a pathway from a to d given reactions from both b and c to d: i.e., $\langle \Delta = \{reaction(b,c), reaction(b,d)\}, \Theta = \{\{U/a\}\} \rangle$ is a solution. Similarly, we have $\langle \Delta = \{reaction(a,b), reaction(a,c)\}, \Theta = \{\{U/a\}\} \rangle$. In general, there is a pathway from any X to d if there is a reaction from X to d: i.e., we have $\langle \Delta = \{reaction(X,d)\}, \Theta = \{\{U/X\}\} \rangle$. Additionally, there is a pathway X or b to d if there is a reaction from X to c: i.e., we have $\langle \Delta = \{reaction(X,c)\}, \Theta = \{\{U/X\}, \{U/b\}\} \rangle$. In this case, Definition 3 is satisfied since (i) $T \models \forall X \left(reaction(X,c) \rightarrow pathway(X,d) \vee pathway(b,d) \right)$, and (ii) $T \not\models \forall X \left(reaction(X,c) \rightarrow \bot \right)$ as it is consistent to let $X = a$, for example.

[2] We assume that all instances of the abducible literals in A can be added to Δ and we assume Θ only binds variables in G. If G is ground, we can always set $\Theta = \{\emptyset\}$.

While the view of abduction presented above imposes the correct semantics on Δ and Θ, this pathway example shows that, in general, a large number of solutions are possible. It is therefore desirable to introduce a minimality criterion for eliminating redundant hypotheses. A simple subsumption test, formalised in Definition 5, can be used based on the notion of a solution clause, given in Definition 4.[3] This clause contains the complement of each abducible in Δ and an answer literal [5], with predicate ϕ, for each substitution in Θ. An abductive solution $\langle \Delta, \Theta \rangle$ can then be defined as minimal iff its solution clause is not strictly subsumed by the solution clause of another solution $\langle \Delta', \Theta' \rangle$.

Definition 4 (Solution Clause). *Given a theory T, goal G, and abducibles A, let $\Delta \subseteq Inst(A)$ be a set of literals, and let Θ be a set of substitutions for G. Let $X_1, \ldots, X_n$ be the variables in G, and let ϕ be a predicate not appearing in G, T or A. A solution clause for $\langle \Delta, \Theta \rangle$, denoted $Soln(\Delta, \Theta)$, is a clause of the form:*

$$Soln(\Delta, \Theta) = \bigvee_{L \in \Delta} \overline{L} \vee \bigvee_{\sigma \in \Theta} \phi(X_1, \ldots, X_n)\sigma.$$

Definition 5 (Minimal Solution). *Let T be a theory, G be goal, and A be a set of literals. A minimal solution for G wrt T and A is an abductive solution $\langle \Delta, \Theta \rangle$ (for G wrt T and A) for which there is no other abductive answer $\langle \Delta', \Theta' \rangle$ (for G wrt T and A) such that $Soln(\Delta', \Theta') > Soln(\Delta, \Theta)$.*

Example 2. All of the solutions mentioned in Example 1 are minimal except $\langle \Delta = \{reaction(a, b), reaction(a, c)\}, \Theta = \{\{U/a\}\} \rangle$ — as its solution clause $Soln(\Delta, \Theta) = \neg reaction(a, b) \vee \neg reaction(a, c) \vee \phi(a)$ is subsumed by the solution clause $Soln(\Delta', \Theta') = \neg reaction(X, b) \vee \neg reaction(X, c) \vee \phi(X)$ of the minimal solution $\langle \Delta' = \{reaction(X_1, b), reaction(X_1, c)\}, \Theta' = \{\{U/X_1\}\} \rangle$. There are infinitely many other minimal solutions of the form $\langle \Delta_n = \{reaction(X_1, X_2), \ldots, reaction(X_{n-1}, X_n), reaction(X_n, b), reaction(X_n, c)\}, \Theta_n = \{\{U/X_1\}\} \rangle$.

Proposition 1 shows that the solution clause $Soln(\Delta, \Theta)$ of any abductive solution $\langle \Delta, \Theta \rangle$ can be deduced from the theory T augmented with a so-called answer clause. As formalised in Definition 6, the answer clause $Ansr(G)$ of a goal G is composed of the complement of each literal in G and an answer literal, with predicate ϕ and arguments corresponding to the variables in G. Thus, in Example 1, $Ansr(G)$ is the clause $\neg pathway(U, d) \vee \phi(U)$.

Definition 6 (Answer Clause). *Let G be a goal with the variables $X_1, \ldots, X_n$. An answer clause for G, denoted $Ansr(G)$, is a clause of the form:*

$$Ansr(G) = \bigvee_{L \in G} \overline{L} \vee \phi(X_1, \ldots, X_n).$$

Proposition 1. *Let T be a theory, G be a goal, and A be a set of literals. Let $\Delta \subseteq Inst(A)$ be a set of literals, and Θ be a set of substitutions for G. Then $\langle \Delta, \Theta \rangle$ is an abductive solution for G wrt T and A iff*

$$\textit{(i) } T \wedge Ansr(G) \models Soln(\Delta, \Theta) \textit{ and (ii) } T \wedge Ansr(G) \not\models Soln(\Delta, \emptyset)$$

[3] We assume the variables $X_1, \ldots, X_n$ are always written in some standard order and we assume that the predicate ϕ does not appear in T, G or A.

3.2 Proof Procedure

Proposition 1 shows that the computation of abductive solutions can be reduced to a consequence finding problem by adding an answer clause to the theory and searching for any solution clauses that are entailed (i). This can be done using a production field containing the answer literal and the complement of every abducible. In this case, subsume minimal consequences, i.e., characteristic clauses, correspond to minimal solutions and thus can be computed by SOL resolution. Moreover, Proposition 1 also shows that inconsistent explanations can be avoided by simply rejecting solution clauses with no answer literals (ii).

Example 3. Let T, G and A be as defined in Example 1 and apply the method suggested above. First form the answer clause $Ansr(G) = \neg pathway(U, d) \vee \phi(U)$ and the production field $P = \langle \{\neg reaction(\_, \_), \phi(\_)\} \rangle$. Then search for all SOL deductions from $T \wedge Ansr(G)$ and P. One such deduction is shown below. For clarity, each step is annotated with its index and predicates are abbreviated. The deduction starts with $Ansr(G)$. Referring to Definition 2, clauses 2,3,4,5 & 10 are obtained by resolution; 7,9 & 12 are obtained by truncation; 8 is obtained by reduction; while 6 & 13 are obtained by skipping. The derived clause $\neg r(c, d) \vee \neg r(b, d) \vee \phi(a)$ gives the abductive solution $\Delta = \{r(c, d), r(b, d)\}$ and $\Theta = \{\{U/a\}\}$. Similar deductions yield the other minimal solutions. Note that the inconsistent explanation $\Delta = \{reaction(c, b)\}$ is easily recognised by the lack of answer literals in its solution clause.

$^{1}\langle \Box \,,\, \underline{\neg p(U, d)} \vee \phi(U) \rangle$

$^{2}\langle \Box \,,\, \underline{\neg r(U, Y)} \vee \neg p(Y, d) \vee \boxed{\neg p(U, d)} \vee \phi(U) \rangle$

$^{3}\langle \Box \,,\, \underline{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{4}\langle \Box \,,\, \underline{\neg p(c, Z)} \vee p(a, Z) \vee \boxed{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{5}\langle \Box, \underline{\neg r(c, Z)} \vee \boxed{\neg p(c, Z)} \vee p(a, Z) \vee \boxed{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{6}\langle \neg r(c, Z) \,,\, \boxed{\neg p(c, Z)} \vee p(a, Z) \vee \boxed{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{7}\langle \neg r(c, Z) \,,\, \underline{p(a, Z)} \vee \boxed{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{8}\langle \neg r(c, d) \,,\, \boxed{r(a, c)} \vee \boxed{\neg r(a, b)} \vee \neg p(b, d) \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{9}\langle \neg r(c, d) \,,\, \underline{\neg p(b, d)} \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{10}\langle \neg r(c, d) \,,\, \underline{\neg r(b, d)} \vee \boxed{\neg p(b, d)} \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{11}\langle \neg r(c, d) \vee \neg r(b, d) \,,\, \boxed{\neg p(b, d)} \vee \boxed{\neg p(a, d)} \vee \phi(a) \rangle$

$^{12}\langle \neg r(c, d) \vee \neg r(b, d) \,,\, \underline{\phi(a)} \rangle$

$^{13}\langle \neg r(c, d) \vee \neg r(b, d) \vee \phi(a) \,,\, \Box \rangle$

Example 3 shows SOL can be used for abductive query answering. But, a potential flaw is that the completeness of SOL is only true for stable production fields. This holds only if all abducibles are maximally general literals, which, in practice, will not be the case. For instance, suppose that we only wish to assume the existence of reactions that produce d. The most efficient way to do this is by using the abducibles $A' = \{ reaction(V, d) \}$. But the new production field $P' = \langle \{\neg reaction(\_, d), \phi(\_)\} \rangle$ is not stable as $\neg reaction(X, Y)$ does not belong to P' even though it subsumes a clause $\neg reaction(X, d)$ that does.

However, using an unstable production field results in a loss of completeness. In particular the explanation $\Delta = \{reaction(c, d), reaction(b, d)\}$ is no longer computed even though it falls within the given language bias. Under this new production field P', the SOL deduction shown in Example 3 is invalid, as it becomes impossible to apply the skip operation after step 5. While it is possible, in this particular case, to construct a deduction by postponing the skip until after the reduction binding Z to d between steps 7 and 8, such re-orderings are not considered as they would dramatically increase the search space.

Fortunately, completeness is restored by the procedure in Definition 9 which uses a transformation formalised in Definition 7. For each literal L in A, a bridge clause is added to T which contains L and the negation of an atom $p_L(X_1, \ldots, X_n)$, which becomes a new abducible in place of L. The predicate p_L is a new predicate symbol which, just like an answer literal, represents any bindings to the variables X_i in L.[4] This transformation ensures all abducibles are maximally general and that abductive solutions to the transformed and original problems are isomorphic by simply propagating the bindings from p_L back to L.

Definition 7 (Bridge Theory). *Let A be a set of literals. A bridge theory for A, denoted $Brdg(A)$ is a theory of the form*

$$Brdg(A) = \bigwedge_{L \in A} \neg p_L(X_1, \ldots, X_n) \vee L$$

where $X_1, \ldots, X_n$ are the variables in L, and where p_L is a new predicate symbol.

Definition 8 (SOL Procedure). *Let T be a theory, P be a production field, and k be a positive integer. $SOL(T, P, k)$ denotes the set of clauses S for which there exists an SOL deduction (from T and P) of length $n \leq k$.*

Definition 9 (Abduce Procedure). *Given a theory T, goal G, literals A, and integer k, let $Abduce(T, G, A, k)$ denote the set S computed as follows:*

1. *Let B be the theory $T \cup Ansr(G) \cup Brdg(A)$*
2. *Let N be the answer literal $\phi(X_1, \ldots, X_n)$ appearing in $Ansr(G)$*
3. *Let M be the set of bridge literals $\neg p_L(X_1, \ldots, X_m)$ appearing in $Brdg(A)$*
4. *Let P be the production field $\langle \{N\} \cup M \rangle$*
5. *Let Q be the set of clauses obtained from $SOL(B, P, k)$ by replacing each bridge literal of the form $p_L(t_1, \ldots, t_m)$ with the literal $L\{X_1/t_1, \ldots, X_m/t_m\}$*
6. *Let S be the set of solutions $\langle \Delta, \Theta \rangle$ such that $Ansr(\Delta, \Theta) \in \mu Q$ and $\Theta \neq \emptyset$*

[4] A new (predicate) proposition p_L is used for each (non-) ground abducible L.

The soundness and completeness result in Proposition 2 follows immediately from analogous results on the soundness and completeness of SOL for minimal conditional answers under stable production fields [8]. Note that minimality is ensured by applying the μ operator in step 6 *after* the computed bindings are transferred back to the original abducibles in step 5. Note also how consistency is ensured by removing in step 6 any explanations with an empty answer. Finally note that the transformation needed to ensure a stable production field is very efficient, resulting in the addition of only one theory clause for each abducible and incurring just one additional resolution step per abduced literal.

Proposition 2. *Let T be a theory, G be a goal, and A be a set of literals. $\langle \Delta, \Theta \rangle$ is a minimal solution of G wrt T and A iff there exists an integer k such that $\langle \Delta, \Theta \rangle \in Abduce(T, G, A, n)$ for all $n \geq k$.*

This formulation reflects the fact that, in practice, some bound k must be imposed on the depth of SOL derivations. A minimal solution is a solution that is computed at some depth k and not later subsumed by another solution of higher depth. While the existence of k is guaranteed, it follows from previous work [8] that its value may be undecidable. This is due to the undecidability of (consistency and) minimality checking. In practice, the procedure can only return those solutions that are minimal with respect to the depth bound k. But this is true of any abductive procedure and is often good enough in practice.

Returning to the previous example, the abducibles $A' = \{ reaction(V, d) \}$ result in the bridge theory $\{\neg p_r(V) \vee reaction(V, d)\}$ and production field $\langle \{\neg p_r(\_), \phi(\_)\} \rangle$. The literal $\neg reaction(c, Z)$, selected at step 5 of the earlier derivation now resolves with the bridge clause to leave the goal $\neg p_r(Z)$ which can be skipped. The SOL derivation, which proceeds just as before, results in the clause $\neg p_r(c) \vee \neg p_r(b) \vee \phi(a)$, which is subsequently replaced by the clause $\neg reaction(c, d) \vee \neg reaction(b, d) \vee \phi(a)$. Thus, our transformation overcomes the restriction of SOL to stable production fields and ensures the completeness of our abductive procedure for computing minimal solutions.

4 Related Work

Several other procedures have been proposed for abductive reasoning. Poole [18] describes a method for compiling full clausal abductive problems into Prolog. This approach handles indefinite answers, but only returns ground explanations. Kakas et al. [11] review several procedures for abductive logic programs. Some of these include constraint solvers, which allow the computation of non-ground explanations, but none of them compute indefinite answers. Mayer and Pirri [15] propose a general first-order abductive proof procedure, based on the tableaux and sequent calculi, which uses dynamic Skolemisation and anti-Skolemisation to avoid a conversion to clausal form. Their approach is complete for finding entailment-minimal explanations, but does not return answer substitutions and does not exploit the many resolution pruning strategies used by SOL.

Abductive reasoning is closely related to conditional query answering, which could, in principle, be used to implement abduction [2]. Baumgartner et al. [1] give a calculus for conditional answer computation, but this is incomplete for finding minimal solutions [8]. Demolombe [3] gives a calculus for computing minimal conditional answers, but this does not support function symbols or indefinite answers. Iwanuma and Inoue [8] show SOL resolution is complete for minimal answer computation, but only under the assumption of a stable production field and without addressing the issue of consistency.

In effect, the approach introduced in this paper overcomes the restriction of SOL resolution to stable production fields and can potentially be built directly into SOL so that other applications can benefit from this generalisation. We have also developed a slightly extended transformation that allows abducibles to be annotated with goals in order to impose typing and validity constraints on the terms in abduced literals, e.g., $A = \{reaction(U, V) : node(U), node(V)\}$. These goal literals are simply added to the bridge clause along with the abducible and thereby afford a finer degree of control over the abductive bias.

5 Conclusions

This paper presented a proof procedure and semantics for full clausal abduction that caters for indefinite answers and non-ground abducibles. Viewing abduction as a form of conditional query answering, it showed how the consequence finding approach of SOL resolution can be used to overcome the syntactic restrictions imposed by other systems. In so doing, it revealed the significance of a stability restriction underlying all previous work on SOL, and gave an efficient program transformation to overcome this assumption. The approach was illustrated on a small example motivated by metabolic pathway analysis which also showed the potential utility of logical abduction in Discovery Science. In this example, non-Horn clauses were used to represent incomplete knowledge and non-ground solutions were used to suggest possible refinements of the initial knowledge by inferring the presence of missing nodes and arcs. While our approach is not yet mature enough to tackle real scientific discovery tasks, we intend to validate our methodology more fully on a more realistic model of biochemical networks. We are also developing an extension of our method to allow the induction of general laws from examples and a background theory. This extended approach, called Hybrid Abductive Inductive Learning [19], uses abductive explanations returned by (a simplified version of) our procedure to seed the formation of a ground unit theory that is subsequently generalised by an inductive search procedure. In this way we eventually hope to utilise our abductive method in the process scientific knowledge discovery.

Acknowledgements

This work is supported by the Research Councils UK (RCUK) and the Japan Society for the Promotion of Science (JSPS).

References

[1] Baumgartner, P., Furbach, U., Stolzenburg, F.: Computing Answers with Model Elimination. Artif. Intel. 90(1-2), 135–176 (1997)

[2] Burhans, D., Shapiro, S.: Abduction and Question Answering. In: Proc. the IJCAI'01 Workshop on Abductive Reasoning, pp. 905–908 (2001)

[3] Demolombe, R.: A Strategy for the Computation of Conditional Answers. In: Proc. 10th Europ. Conf. on Artif. Intel. pp. 134–138 (1992)

[4] Gartner, J., Swift, T., Tien, A., Damásio, C., Pereira, L.: Psychiatric Diagnosis from the Viewpoint of Computational Logic. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS (LNAI), vol. 1861, pp. 1362–1376. Springer, Heidelberg (2000)

[5] Green, C.: Theorem-Proving by Resolution as a Basis for Question-Answering Systems. Mach. Intel. 4, 183–205 (1969)

[6] Hartshorne, C., Weiss, P., Burks, A.W. (eds.): Collected papers of Charles Sanders Peirce. Harvard University Press (1931–1958)

[7] Inoue, K.: Linear resolution for Consequence Finding. Artif. Intel. 56(2-3), 301–353 (1992)

[8] Iwanuma, K., Inoue, K.: Minimal Answer Computation and SOL. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 245–257. Springer, Heidelberg (2002)

[9] Iwanuma, K., Inoue, K., Satoh, K.: Completeness of Pruning Methods for Consequence Finding Procedure SOL. In: Proc. 3rd Int. Workshop on First-Order Theorem Proving, pp. 89–100 (2000)

[10] Josephson, J., Josephson, S. (eds.): Abductive Inference: Computation, Philosophy, Technology. Cambridge University Press, Cambridge (1994)

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

[12] King, R., Whelan, K., Jones, F., Reiser, P., Bryant, C., Muggleton, S., Kell, D., Oliver, S.: Functional Genomic Hypothesis Generation and Experimentation by a Robot Scientist. Nature 427, 247–252 (2004)

[13] Lloyd, J.: Foundations of Logic Programming. Springer, Heidelberg (1987)

[14] Loveland, D.: Automated Theorem Proving: A Logical Basis, North Holland (1978)

[15] Mayer, M., Pirri, F.: First-Order Abduction via Tableau and Sequent Calculi. Bulletin of the IPGL 1(1), 99–117 (1993)

[16] Nabeshima, H., Iwanuma, K., Inoue, K.: SOLAR: A Consequence Finding System for Advanced Reasoning. In: Mayer, M.C., Pirri, F. (eds.) TABLEAUX 2003. LNCS, vol. 2796, pp. 257–263. Springer, Heidelberg (2003)

[17] Papatheodorou, I., Kakas, A., Sergot, M.: Inference of Gene Relations from Microarray Data by Abduction. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 389–393. Springer, Heidelberg (2005)

[18] Poole, D.: Compiling a Default Reasoning System into Prolog. New Generation Computing 9(1), 3–38 (1991)

[19] Ray, O., Inoue, K.: Mode-Directed Inverse Entailment for Full Clausal Theories. In: Blockeel, H., Shavlik, J., Tadepalli, P. (eds.) ILP 2007. LNCS, Springer, Heidelberg (2007)

[20] Stickel, M.: A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog. Theoretical Computer Science 104(1), 109–128 (1992)

[21] Zupan, B., Demsar, J., Bratko, I., Juvan, P., Halter, J., Kuspa, A., Shaulsky, G.: GenePath: a System for Automated Construction of Genetic Networks from Mutant Data. Bioinformatics 19(3), 383–389 (2003)