Abduction in Model Generative Reasoning

Roger T. Hartley and Michael J. Coombs
Computer Science Department, and
Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

October 19, 1990

1 Introduction

Most of the abductive mechanisms in the literature are based on formalizations in logical inference. This is true of the model-based approaches of Poole and Shanahan, the resolution approaches of Stickel and Levesque and the work of Charniak and Reiter. This paper shows how a related formalization can be assembled from representations based on Sowa's conceptual graphs and the appropriate (binary) operations on these graphs. The equivalence of the method to logical abduction is noted. Abduction is only useful in a proper pragmatic setting of problem solving. Examples are given to show how the mechanism operates, and algorithms are presented, suitable for complexity analysis.

2 Principled abductive inferences

It has been noted by many people that abduction is a reasoning principle that does not fit comfortably in the framework of deductive logic. Whereas deduction proceeds from sound axioms to sound conclusions, abduction proceeds from sound axioms to potentially unsound conclusions. i.e.

$$\frac{B \qquad A \rightarrow B}{A}$$

That this form of reasoning is necessary for efficient problem solving has also been noted (Peirce, 1957; Reiter and deKleer, 1987) The question here is how to control abductive inference in order to solve problems, without experiencing the combinatorial explosion in a disastrous way. Bylander et al (Bylander et al, 1989) have shown that all abductive mechanisms will be NP-hard at least, unless certain simplifying constraints are met.

In order to examine these issues we need a way to formalize all the aspects of abduction. These are:

1. *hypothesis generation:* rules relating pieces of knowledge which can be 'run backwards'to build abductive inferences. These can relate types, properties, parts/wholes or causal events.

   There are four main uses of a rule:

   - as a selectional constraint on types, e.g. all U's are V's:

$$\forall x\, U(x) \rightarrow V(x) \tag{1}$$

   - as an Aristotelian definition, e.g. if something has properties A, B, C etc. then it is a V

$$\forall x\, A(x) \wedge B(x) \wedge C(x) \cdots \rightarrow V(x) \tag{2}$$

- as a contingent, or schematic definition, e.g. if something is a V, then it has properties A, B, C etc.

$$\forall x\, V(x) \rightarrow A(x) \wedge B(x) \wedge C(x) \cdots \quad (3)$$

- to express causality, e.g. if P, Q, R all happen, then X, Y, Z will happen as a direct consequence

$$P \wedge Q \wedge R.... \rightarrow X \wedge Y \wedge Z \cdots \quad (4)$$

2. *parsimony:* a principle to prevent generation of hypotheses that are either redundant or unwanted
3. *evaluation:* a principle to order hypotheses according to their worth. This might be complexity, faithfulness to data, coverage of data, predictive power, or others.

2.1 The best explanation

The task associated with abductive inference is explanation of data just as the task associated with deduction is prediction (Shanahan, 1989). Both these tasks produce alternatives. Given a set of (incomplete) observations about the present one can predict many futures, and hypothesize many pasts. Metaphysically speaking, with complete knowledge perhaps there can be only one past and one future, but since we always have incomplete knowledge, we must have the ability to put a preference order on both hypotheses and predictions. Non-monotonic logics prefer inferences with minimal models (Shoham, 1988) and other techniques involve minimal cost of assumption (Hobbs et al., 1988). General principles of systernaticity (Rescher, 1964)) point to coherence, elegance and simplicity as the driving criteria, and playing the 'numbers game' is one way to embody these principles.

Seen this way, abduction is a search in a space of hypotheses with an evaluation function whose parameter vector includes both intrinsic metrics (size, complexity, covering power etc.) and metrics formed from more semantic, or content-specific considerations.

Techniques for reducing the proportion of the space to be searched are well-known in AI. However, the pragmatics of abduction demand another consideration. There is indeed an obvious progression from abductive operators that produce multiple alternative hypotheses to heuristic search through that space, but that is only part of the story. In a problem solving framework, abduction (and deduction as well) is not done once, but is performed repeatedly as new data become available and as new predictions are made. Rather than repeat the abductive task from scratch, by, say, adding the new data to the existing data and building a new set of alternative explanations, it is better to review the current set of alternatives and revise them in the light of the new data. This technique, when applied to a single hypothesis has been called theory revision (O'Rorke et al., 1989) but here we are talking about modifying a population of hypotheses and using metrics on the whole population to help guide the next abduction. An associated technique is the ATMS method of belief revision (Reiter and deKleer, op *cit)* where the maintenance of a population of consistent partitions of a set of inconsistent statements is the aim.

The use of global metrics across a whole population, such as number of hypotheses, average size and complexity reminds us of the fitness function in a genetic algorithm where optimization relies on a scheme for measuring the average fitness across a set of trial points. We are exploiting this similarity by incorporating a global fitness function into the control portion of a problem solver (Coombs et al., 1990).

$$D \cup M \vdash F$$

Expressed in operator form this is

$$M = AB(F, D)$$

1 In our Model Generative Reasoning architecture, F is a set of facts, D a set of definitional schemata and M a set of models

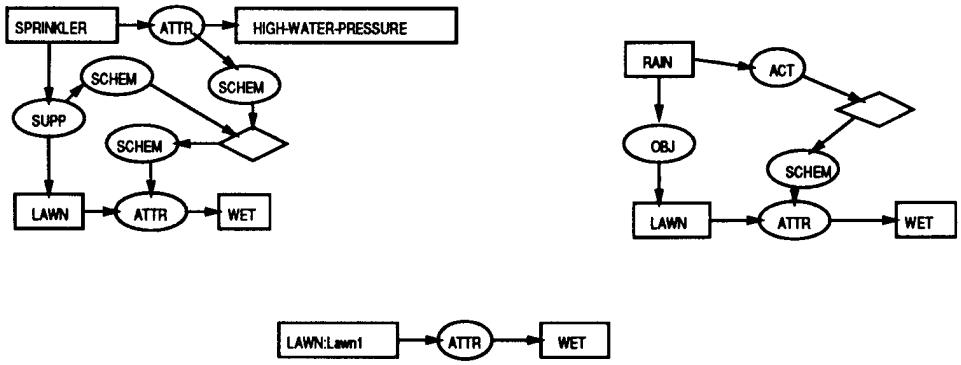


Figure 1: The 'wet' rules and the lawnl fact

where AB is the abductive operator. Various mechanisms for abduction are currently in use. Most are some adaptation of a resolution technique where the residuals of the proof (roughly, if these things were true, then the proof would succeed) form the M we are looking for. In our work we rely on the set covering ideas first used by Reggia (Reggia et al., 1984) and the maximal join operation between conceptual graphs to make abductions. Specifically, whereas many of the mechanisms in the literature effect *relational* abduction, our techniques effect *object* abduction. Although these are technically duals, object-centered abduction seems to offer better scope for a greater variety of types of abduction (Coombs et al., forthcoming). As an example, we take a simple causal event motivated by Pearl's lawn sprinkler.

A causal rule such as

$$\forall xy\ sprinkler(x) \wedge high - water - pressure(x) \wedge lawn(y) \wedge on(x, y) \rightarrow wet(y)$$

and

$$\forall x\ lawn(x) \wedge rained - on(x) \rightarrow wet(x)$$

together with a piece of data

$$wet(lawn1)$$

can be used to abduce two explanations for the wetness, i.e. that there is a high-water-pressure sprinkler on the lawn, or that the lawn was rained on. Rules such as this are *relational*, because their use is governed by the predicate names. It is the predicate *wet* in the data that picks up the two rules and allows the explanation to be built.

Contrast this with the conceptual graph versions of these ruleS2. In these (Figure 1) all relations are primitive and what were predicates in the logic version become concept types (wetness, high-water-pressure, the act of raining).

Abduction now proceeds first by covering the concepts contained in the data graph (LAWN and WET) with available rule graphs. Both provide complete cover, so both are relevant. By joining the data graph with each of the rule graphs maximally (i.e. merging the graphs at all possible common concepts) two hypothesis graphs may be generated each of which constitutes an explanation. Note that the nature of an explanation is also modified in this account. The equivalent to these graphs in the logic formulation would be

$$\forall x\ sprinkler(x) \wedge lawn(lawn1) \wedge high - water - pressure(x) \wedge on(x, lawn1) \rightarrow wet(lawn1)$$

and

$$lawn(lawn1) \wedge rained - on(lawn1) \rightarrow wet(lawn1)$$

that is, they are conditional structures that display the causality rather than assuming it.

Abduction is thus a two stage operation, cover, followed by maximal join, or

$$Abduction = cover + join$$

2Actually the diamond shaped actor nodes in these graphs are an extension to Sowa's original formulation (Sowa, 1984) to allow temporal and spatial reasoning (Hartley, forthcoming)

Although cover provides an equivalent mechanism to logical abduction, join can provide a further sort of abduction; that of type restriction, or 'label' abduction. If there is a rule like

$$\forall x\ P(x) \rightarrow Q(x)$$

or "all P's are Q's" then join can perform abduction from Q's to P's. This is so because the operation of maximal join is defined to merge graphs on nodes which have a non-trivial common subtype. Thus PET and MAMMAL can join to give CAT (or DOG or ...). If a data item talks about a PET, and the only available knowledge is about MAMMALS, then explanations can be built that talk about CATS etc. This of course assumes that the subtype relationship between these concepts is already known.

3.1 Sources of alternatives

Just as the logical form can produce alternative explanations (Prolog would, for instance return all alternatives if asked for resatisfaction) so can cover. Maximal join produces alternatives beacuse there may be more than one maximal common subtype for any two given types (the type structure is potentially a complete lattice). A third source of alternatives comes when graphs contain more than one occurrence of a concept type. In this case they can be merged with another graph containing only one occurrence of the type in a number of ways depending on the number of duplications. Where there are n duplications in one graph and m in another, then the graphs can be joined in

$$\frac{n!}{(n-m)!}$$

ways, where n > rn.

These three sources of alternatives are independent, giving a final number of

$$covers \times subtypes \times combined - duplicates$$

. Some of this total may be redundant, especially where the resulting graphs have symmetries in them. However, in general any one of the alternatives produced is valid. Maximal join assures us of coherence, and they all entail the original data graphs in some way or other. It is valuable to be able to pinpoint these sources of complexity so that they can be controlled if desired.

4 Algorithms for cover and join

The algorithms here are implementations of the first two of the general principles presented in section 2. Evaluation of graphs is omitted here for brevity. Hypothesis generation is partially implemented in the procedure MakeAlternatives, and maximal join. Parsimony is embodied in MinimizeAlternatives.

4.1 How graphs are represented

Let a *bor* be a *node-label* pair: (n, 1) where $l \in L$ and n is a natural number, used as a unique identifier. We will call these identifiers *nodes.* Boxes are elements of a mapping $NodeLabel : N \rightarrow L$ where every n has a unique 1, but every I may have multiple n's.

The links are then a mapping between nodes, together with a flag that indicates the direction of the connection between the nodes: $Conn : N \rightarrow N \times \{f, b\}$ (f means a forward link and *b* a back link).

A graph is then a triple

$$\langle N, B, Conn \rangle$$

4.2 Cover

Let $F' \subseteq F + M$ be the set of chosen facts[3] and D be the set of definitions. *F, M* and D are distinguished members of the set of graphs G, i.e. G = F + D + M. Let *Label(q)* be the set of concept labels in any graph $g \in G$. These terms come from a set *L* of concept labels. A cover of P is obtained if:

$$\cup Label(f' \in F') \subseteq \cup Label(d' \in D')$$

[3] for the purposes of cover, we can choose F or M as the source of graphs

where

$$D' \subseteq D$$

and a cover is minimal if *D'* is the smallest such subset of *D*.

4.2.1 The algorithm for cover

The representation system maintains a hash table of concepts and the definitions that contain them. Call this mapping *Contained : L -+ D.* It is easy then to form a set of alternative covers for each label in P in the following way:

MakeAlternatives

```
1 C = ∅
2 for each f_i ∈ F'
3     for each l_j ∈ Label(f_i)
4         C = C ∪ {Contained(l_j) }
```

The set C can be seen as a conjunct of disjuncts. Where each element of C contains at least one necessary definition. Viewed as a Boolean expression it is

$$\bigwedge_i \bigvee_j d_{ij}$$

Minimal cover is then a process of minimizing this expression so that there are the smallest number of alternatives and each alternative is as small as possible.

The minimization algorithm is:

MinimizeAlternatives

```
1 A^cover = {∅}
2 for each c_i ∈ C [the set of alternatives from above]
3     N = ∅
4     for each d_j ∈ c_i
5         T = ∅
6         for each a_k ∈ A^cover
7             T = T ∪ {a_k ∪ {d_j}}
8         N = N ∪ T, where ¬∃ n_p, n_q ∈ N : n_p ⊃ n_q
          [hence N does not contain a set which is a superset of any other]
9         A^cover = N
```

5 Maximal join and project

Given two conceptual graphs [4], produce a graphs representing their maximal join. The graphs contain labels taken from a type lattice on the relation subtype. The graphs may have duplicate labels (e.g. more than one occurrence of any given label), but the maximal common subtype of any two labels must be unique.

We now present an example of maximal join. It uses a type hierarchy where *B* and C < A, *E* < *B* and C[5]. The two graphs are

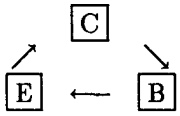
$\boxed{A_1} \rightarrow \boxed{\text{C}} \rightarrow \boxed{A_2}$ and $\boxed{\text{B}} \rightarrow \boxed{\text{E}}$ (subscripts indicate duplicate labels)

[4] in this paper we use the reduced form of conceptual graph where relation nodes are deleted

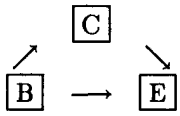
[5] < means 'subtype'

5.1 Maximal join

From the above graphs produce the following alternative maximal joins:

1. 

joining A_1 to E and A_2 to B.
2. [E] ↔ [E] → [A]

   joining A_1 to E and C to B.
3. 

joining A_2 to E and A_1 to B.
4. [A] → [E] → [E]

   joining A_2 to E and C to B.
5. [B] → [E] → [A]

   joining C to E and A_1 to B.
6. [A] → [E] ↔ [B]

   joining C to E and A_2 to B.

5.2 The algorithm for maximal join

The algorithm is in two parts:

1. Produce the set of alternative pairings of nodes from the two graphs.
2. Merge the graphs on paired nodes, producing one resultant graph.

5.2.1 Pair alternatives

Let the two graphs to be joined contain sets of nodes N, and N2, and the joined graph contain a set *N3.* Let the connection mappings be Conn,, Conn2 and Conn3 respectively.

Let labels be taken from the set of lowercase letters viz a,b,c etc. ’Let the labels in the first graph be a set U and the labels in the second be a set V. The possible joins of two graphs are then governed by the number of ways in which individual node pairings can be combined, subject to allowable pairs of labels.

A node can be paired with another if the maximal common subtype of their labels (the function M_b: $L \times L \rightarrow L$, where L is the domain of labels.) is not $\perp$, e.g.

$$M_b(a,b) = b,\ M_b(b,c) = e\ etc.$$

An alternative pairing ai taken from a set A^{pair} of all such pair combinations is the largest set of possible individual node pairings. ai is a set { ... pij ... } where

$$\begin{aligned} p_{ij} &= \langle n_i, n_j \rangle : n_i \in N_1, n_j \in N_2, \\ u_i &= NodeLabel(Conn_1(n_i) \uparrow 1) \in U, \\ v_j &= NodeLabel(Conn_2(n_j) \uparrow 1) \in V, \\ M_b(u_i, v_j) &\neq \perp \end{aligned}$$

6 these correspond to the uppercase ’nodes’ in the example, i.e. A, B, C etc.

Alternative joins of the two graphs are then obtained by forming the Cartesian product of the *ai,* but disallowing combinations where nodes are mentioned in more than one pair. Depending on these constraints, there may be alternatives of differing cardinality, i.e. when a particular node pairing stops one of its nodes being paired with nodes from the other graph.

Let this set be A^{join}. It elements are $a_i^{join} \in a_1 \times a_2 \times \ldots$ where

$$a_i^{join} = \{\langle n_i, n_j\rangle : \neg\exists kl\ (n_k = n_l \vee n_k = n_l, k \neq l)\}$$

The algorithm is:
(ALGORITHM FOLLOWS) In
our example we have:

$$\begin{aligned} B_1 &= \{\langle 1,a\rangle, \langle 2,a\rangle, \langle 3,c\rangle\} \\ B_2 &= \{\langle 1,b\rangle, \langle 2,e\rangle\} \\ N_1 &= \{1,2,3\} \\ N_2 &= \{1,2\} \\ Conn_1 &= \{\langle 1,3,f\rangle, \langle 3,2,f\rangle, \langle 3,1,b\rangle, \langle 2,3,b\rangle\} \\ Conn_2 &= \{\langle 1,2,f\rangle, \langle 2,1,b\rangle\} \\ M_b &= \{\langle a,b,b\rangle, \langle a,e,e\rangle, \langle c,b,e\rangle, \langle c,e,e\rangle, \ldots\} \end{aligned}$$

The set A^{pair} is then

$$\{\{\langle 1,1\rangle, \langle 1,2\rangle\}, \{\langle 2,1\rangle, \langle 2,2\rangle\}, \{\langle 3,1\rangle, \langle 3,2\rangle\}\}$$

since all pairings are possible.

A^{join} starts out containing the empty set (line 11). After the first pass *(a,),* A^{join} is:

$$\{\{\langle 1,1\rangle\}, \{\langle 1,2\rangle\}\}$$

After the second *(a2),* it is

$$\{\{\langle 1,2\rangle, \langle 2,1\rangle\}, \{\langle 1,1\rangle, \langle 2,2\rangle\}\}$$

Finally, after the third *(a3),* A^{join} is

$$\{\{\langle 3,1\rangle, \langle 1,2\rangle\}, \{\langle 1,2\rangle, \langle 2,1\rangle\}, \{\langle 3,1\rangle, \langle 2,2\rangle\}, \{\langle 2,2\rangle, \langle 1,1\rangle\}, \{\langle 3,2\rangle, \langle 2,1\rangle\}, \{\langle 3,2\rangle, \langle 1,1\rangle\}\}$$

All of these alternatives represent maximal joins.

5.2.2 Merge graphs

Having produced the set A^{join} of all possible label pairings, the remaining task is to make a graph for each pairing set from the two input graphs. This amounts to producing a new mapping in which the original node pairings are replaced by ones reflecting the merged nodes. Along the way, two sets of old-node to new-node maps are made, one for each input graph. They are called *NodeMap1* and *NodeMaP2.* They are both subsets of a function $NewNode : N \rightarrow N \times L + \epsilon$. NewNode returns c if a node has no mapping. The algorithm is:
(ALGORITHM FOLLOWS)
The first alternative is the graph given by:

$$\begin{aligned} N_3 &= \{1,2,3\} \\ B_3 &= \{\langle 1,d\rangle, \langle 2,e\rangle, \langle 3,a\rangle\} \\ Conn_3 &= \{\langle 1,2,f\rangle, \langle 2,1,f\rangle, \langle 2,1,b\rangle, \langle 1,2,b\rangle\ \langle 1,3,f\rangle, \langle 3,1,b\rangle\} \end{aligned}$$

The other alternatives are similarly obtained.

References

[Bylander et al., 1989] Bylander, T., Allemang, D., Tanner, M.C, and Josephson, J.R. Some results concernign the computational complexity of abduction. *Proc. 1st. Int. Conf. on Principles of knowledge representation and reasoning, pp.* 44-54.

[Coombs etal., forthcoming] Coombs, M.J., Pfeiffer, H.D. and Hartley, R.T., *e-MGR: an architecture for symbolic plasticity,* Int. J. Man-Machine Studies.

[Coombs and Hartley, 1988] Coombs, M. J. and R. T. Hartley. Explaining novel events in process control through model generative reasoning. *International Journal of Expert Systems* 1:89-109.

[Hartley, forthcoming] Hartley, R.T. A uniform representation for time and space and their mutual constraints. *Computers and Mathematics with Applications* Special Issue on Semantic Networks in Artificial Intelligence.

[Hobbs et al... 1988] Hobbs, J.R., Stickel, M., Martin, P., and Edwards, D. Interpretation as Abduction. In *Proc. 26th. Annual Meeting of the ACL, pp.* 95-103, 1988.

[O'Rorke et al., 1989] O'Rorke, P., Morris, S. and Schulenberg, D. Abduction and world model revision. *Proc. 11th. Annual Conference of the Cognitive Science Society pp.* 789-796.

[Peirce, 1957] Peirce, *C.S. Essays in the Philosophy of Science.* New York: Bobbs-Merrill.

[Reggia et al., 1984] Reggia, J.A., D.S. Nau and P.Y. Wang. Diagnostic expert systems based on the set covering model. In M.J. Coombs (ed.), *Development in Expert Systems*. London: Academic Press.

[Reiter and De Kleer, 1987] Reiter, R. and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. *Proc. AAAI-87.* Los Altos, CA: Kaufmann, pp. 183-188.

[Rescher, 1964] Rescher, N. *Cognitive Systematization. ????* 1964?.

[Shanahan, 1989] Shanahan, M. Prediction is deduction, but explanation is abduction. Proc. IJCAI89, pp. 1055-1060.

[Shoham, 19881 Shoham, Y., Chronological Ignorance: Experiments in Nonmontonic Reasoning. *Artificial Intelligence,* 36, pp. 279-331.

[Sowa, 1984] Sowa, J.F. *Conceptual Structures.* Reading, MA: Addison Wesley.