## Chapter 5 Applications and extensions

### 1. Some philosophical remarks

There is a connection between problems of justification and discovery. Any hypothesis discovery scheme is, as previously remarked, only a part of a total system. It is necessary that hypotheses suggested by the scheme be justified (or criticized or whatever). In fact we will show that, under suitable conditions, best explanations, as described above, with almost arbitrary $\mathcal{L}$, are acceptable in the sense of Hintikka and Hilpinen (1966), whose work is in the spirit of Carnap. They wish to give an analysis of the concept of probable knowledge. Knowledge of h in the light of evidence e, is defined by:

$$K(h,e) \equiv Ac(h,e) \wedge h.$$

Ac(h,e) is to mean that e gives h enough support to make it acceptable. The naive analysis of Ac(h,e) is in terms of high probability:

$$Ac(h,e) \equiv P(h,e) > 1 - \varepsilon \text{ where } \varepsilon \leq 0.5.$$

Unfortunately, the naive definition leads to an inconsistency with some generally accepted closure principles. In particular, Hempel (1962) has formulated these conditions:

CA1: If $Ac(h_1,e)$ and .... and $Ac(h_n,e)$ and if $\vdash (h_1 \wedge \ldots \wedge h_n) \rightarrow h$ then $Ac(h,e)$.

CA2: The set $\{h \mid Ac(h,e)\}$ is logically consistent.

Condition CA1 fails because the multiplication theorem for probabilities makes it possible that $P(h_1,e) > 1 - \varepsilon$ and $P(h_2,e) > 1 - \varepsilon$ but $P(h_1 \wedge h_2,e) \leq 1 - \varepsilon$ .

Condition CA2 fails since one can, for example, find hypotheses $h_i(i=1,n)$ such that $P(h_i,e) > 1 - \varepsilon$ $(i=1,n)$ and $P(\bigvee_i \neg h_i,e) > 1 - \varepsilon$ . However $\{h_1,\ldots,h_n, \bigvee_i \neg h_i\}$ is inconsistent. These difficulties come under the general heading of the lottery paradox of Kyburg (1961).

Hintikka and Hilpinen solve these problems in the context of a simple language with exactly k monadic predicate symbols $P_i$, some fixed number of constant symbols but no other function symbols. The authors seem to make a background assumption that different constants denote different individuals in the universe. The lack of detail in the article makes this unclear. There is also the assumption that every universe contains infinitely many individuals. This is however a matter of convenience, and can, it appears, be dropped in a more detailed account.

In the context of an infinite universe, it seems that the assumption that different constants denote different individuals can be dropped. For if a and b are different constants then it seems that, on a priori grounds, probability(a=b) = 0.

Using the predicate symbols $P_i(i=1,k)$ one can define $K = 2^k$ different kinds of individuals using complex predicates $Ct_j(j=1,k)$ which have definitions of the form:

$$Ct_j(x) \equiv \bigwedge_{i=1}^{k} (\pm)\, P_i(x).$$

By saying of each $Ct_j$ whether or not it is instantiated, different world descriptions $C_l (l=1,2^k)$ can be given. To give a more exact definition we introduce the symbols +, -, $\mathcal{B}$ and the metalinguistic variables $\alpha$ and $\beta$, possibly with suffixes, to range over them. The symbols have an ordering $\sqsubseteq = \{< \mathcal{B}, +, >, < \mathcal{B}, - >\}$. Pseudo-formulae are defined by the conditions that if h is a formula then $\alpha$ h is a pseudo-formula, if $h_1$ and $h_2$ are pseudo-formulae so are $\neg h_1, h_1 \wedge h_2$ and $h_1 \vee h_2$. (The implication sign is regarded as an abbreviation.) Pseudo-formulae are abbreviations for certain formulae. It is sufficient, for our purpose, to give examples of the use of the symbols; rather than give a detailed definition:

$+P_1(x) \wedge \neg +P_2(x) \wedge \mathcal{B} P_3(x)$ abbreviates $P_1(x) \wedge \neg P_2(x)$.

$\neg -P_1(x) \vee \neg \mathcal{B} P_2(x)$ denotes $P_1(x)$.

Roughly - denotes $\neg$, + should be ignored and $\mathcal{B}$ means that the immediately following pseudo-formula should be removed. (Think of them as being analogous to +1, -1 and 0.) Nasty cases like $\mathcal{B} \exists x P_1(x) \vee \mathcal{B} P_2(x)$ are handled by the rule that the empty conjunction abbreviates an arbitrary tautology and the empty disjunction abbreviates the negation of an arbitrary tautology. Thus $\mathcal{B} \exists x P_1(x) \vee \mathcal{B} P_2(x)$ abbreviates $\neg (\forall x P_1(x) \vee \exists x \neg P_1(x))$, say. In these terms the $Ct_j$ have definitions of the form:

$$Ct_j(x) \equiv \bigwedge_{i=1}^{k} \alpha_i P_i(x) \quad (\alpha_i \neq \mathcal{B}).$$

They are called attributive constituents.

Constituents, $C_1$ have definitions of the form:

$$C_1 \equiv [\bigwedge_{j=1}^{K} \alpha_j \exists x Ct_j(x)] \wedge [\forall x \bigvee_{j=1}^{K} \alpha_j Ct_j(x)] \ (\alpha_j \neq -).$$

Each constituent is a description of a possible world, in so far as this is possible in the monadic calculus.

Only a certain type of evidence, e, is considered. It is assumed that there are n distinct constant symbols $a_i$, such that

$e = \bigwedge_{i'=1}^{n} \bigwedge_{i=1}^{k} \alpha_{ii'} P_i(a_{i'})$ (where $\alpha_{ii'} \neq \mathcal{B}$).

Using a probability function P due to Hintikka, the authors find by an effective means a constant $n_0 = n_0(\varepsilon, k)$ such that Ac may be defined by:

$Ac(h,e) \equiv P(h,e) > 1 - \varepsilon$ and $n \geq n_0$, where h is a general sentence. This definition satisfies conditions CA1 and CA2 applied to general sentences only. That is:

1) If $Ac(h_1,e)$ and ... and $Ac(h_n,e)$ and if $h_1,\ldots,h_n$ and h are general sentences and $\vdash (h_1 \wedge \ldots \wedge h_n) \rightarrow h$ then $Ac(h,e)$.

2) The set $\{h \mid Ac(h,e)$ and h is general$\}$ is consistent.

The function P has an important difference from the probability function of Carnap. It assigns non-zero values to generalisations. It is in fact one of a whole family of probability functions, Hintikka's $\alpha$-continuum (Hintikka (1965a)), for which the same results

can be established.

In particular, if we write e in the form $\bigwedge_{i'=1}^{n} Ct_{n_{i'}}(a_{i'})$ and consider the unique constituent, $C_{l(e)} \equiv [\bigwedge_{i'=1}^{n} \exists x Ct_{n_{i'}}(x)] \wedge [\forall x \bigvee_{i=1}^{n} Ct_{n_{i'}}(x)]$, then one may show that if $n \geq n_0$ then $Ac(C_{l(e)}, e)$ and further that, if $n \geq n_0$, Ac(h,e) if and only if $\vdash C_{l(e)} \rightarrow h$.

It is necessary to extend this analysis to other sentences than general ones.

Every general sentence h can be expressed as

$$h \equiv \bigvee_{l=1}^{2^K} \alpha_l C_l \text{ where } \alpha_l \neq -,$$

as is well-known (Hintikka, 1953).

One can show that every sentence, h, can be expressed as $h \equiv \bigvee_{l=1}^{2^K} \alpha_l (C_l \wedge h_l)$ where no $\alpha_l = -$, each $h_l$ is singular that is, contains no variable whether bound or not. (Singular just means ground.)

Suppose that $h \equiv \bigvee_{l=1}^{2^K} \alpha_l (C_l \wedge h_l)$ and $h' \equiv \bigvee_{l=1}^{2^K} \alpha'_l (C_l \wedge h'_l)$ where no $\alpha_l$ or $\alpha'_l$ is -. Then $\vdash h \rightarrow h'$ if and only if whenever $\alpha_l \neq \mathcal{O}$, $\alpha'_l \neq \mathcal{O}$ and $\vdash C_l \wedge h_l \rightarrow h'_l$.

Define a partial function of two variables, max, on $\{+, -, \mathcal{O}\}$ by: If $\alpha \sqsupseteq \alpha'$ then $\max(\alpha, \alpha') = \alpha$ and if $\alpha' \sqsupseteq \alpha$, $\max(\alpha, \alpha') = \alpha'$. Otherwise $\max(\alpha, \alpha')$ is undefined.

Define min similarly, with $\sqsupseteq$ replaced by $\sqsubseteq$.

Then $h \wedge h' \equiv \bigvee_l \min(\alpha_l, \alpha'_l)\ (C_l \wedge h_l \wedge h'_l)$,

$h \wedge h' \equiv \bigvee_l \max(\alpha_l, \alpha'_l)\ (C_l \wedge (h_l \vee h'_l))$.

The right hand sides of these equations are defined.

h is defined to be an e-sentence iff it is equivalent to a sentence of the form $\bigvee_l \alpha_l(C_l \wedge h_l)$ where $\vdash e \rightarrow h_l$ and no $\alpha_l$ is -. $\bigvee_l \alpha_l(C_l \wedge h_l)$ is a representation of h. From the above we see that if h and h' are e-sentences, so are $h \wedge h'$ and $h \vee h'$.

Any general sentence $h \equiv \bigvee \alpha_l C_l$ (no $\alpha_l$ = -) is an e-sentence, since $h \equiv \bigvee \alpha_l (C_l \wedge (e \vee \neg e))$. If e' is singular and $\vdash e \rightarrow e'$ then e' is an e-sentence since $e' \equiv \bigvee (C_l \wedge e')$.

We are now in a position to define acceptability of e-sentences on the grounds of evidence e:

Let $h \equiv \bigvee_l \alpha_l (C_l \wedge e_l)$ be a representation of h as an e-sentence. Then:

$$Ac(h,e) \equiv Ac(\bigvee_l \alpha_l C_l, e).$$

The definition is independent of the representation, since if $\bigvee_l \alpha'_l (C_l \wedge e'_l)$ is another representation then, as $\bigvee_l \alpha_l (C_l \wedge e_l) \equiv \bigvee_l \alpha'_l (C_l \wedge e'_l)$, $\alpha_l = \alpha'_l$ for all l. Further the new definition of Ac extends the old one.

If $\vdash e \rightarrow e'$ for some singular e', then $Ac(e',e) \equiv$

$Ac(\bigvee_l(C_l \wedge e'),e) \equiv Ac(\bigvee_l C_l,e) \equiv n \geq n_0$. If this result is counter-intuitive, it is because $Ac(\bigvee_l C_l,e) \equiv n \geq n_0$ is counterintuitive. This seems to us to be a slight, but easily corrected, fault of the definition of Hintikka and Hilpinen.

There is an equivalent definition in terms of high probability and large enough evidence. Suppose h is an e-sentence with representation $\bigvee_l \alpha_l(C_l \wedge e_l)$. We have:

$$P(h,e) = P(\bigvee_l \alpha_l(C_l \wedge e_l),e)$$
$$= \sum_{\alpha_l \neq 0} P(C_l \wedge e_l,e) \text{ (as } \vdash \neg(C_l \wedge C_{l'}) \text{ if } l \neq l').$$
$$= \sum_{\alpha_l \neq 0} P(C_l,e) \text{ (as } \vdash e \rightarrow e_l).$$
$$= P(\bigvee_l \alpha_l C_l,e) \text{ (as } \vdash \neg(C_l \wedge C_{l'}), \text{ if } l \neq l').$$

Therefore $Ac(h,e) \equiv Ac(\bigvee \alpha_l C_l,e)$

$\equiv P(\bigvee \alpha_l C_l,e) > 1 - \varepsilon$ and $n \geq n_0$.

$\equiv P(\bigvee \alpha_l(C_l \wedge e_l),e) > 1 - \varepsilon$ and $n \geq n_0$, (by the above).

$\equiv P(h,e) > 1 - \varepsilon$ and $n \geq n_0$.

$\equiv \alpha_{l(e)} = +$ and $n \geq n_0$ (by previous remarks).

Hempel's conditions may now be demonstrated for e-sentences.

Suppose that $h_1$ and $h_2$ are e-sentences with representations $\bigvee \alpha_l(C_l \wedge h_l)$ and $\bigvee \alpha'_l(C_l \wedge h'_l)$. Then $h_1 \wedge h_2$ has a representation $\bigvee \min(\alpha_l, \alpha'_l)(C_l \wedge (h_l \wedge h'_l))$. So if $Ac(h_1,e)$ and $Ac(h_2,e)$ then $\alpha_{l(e)} = \alpha'_{l(e)} = \min(\alpha_{l(e)}, \alpha'_{l(e)}) = +$ and $n \geq n_0$. Therefore

$Ac(h_1 \wedge h_2, e)$. Suppose that $\vdash h_1 \rightarrow h_2$. Then $P(h_2,e) \geq P(h_1,e)$. Therefore if $Ac(h_1,e)$, $Ac(h_2,e)$. This verifies condition CA1.

If $Ac(h,e)$ then $\alpha_{1(e)} = +$, in any representation of h, and therefore $\vdash C \wedge e \rightarrow h$. As $C \wedge e$ is consistent so too therefore is $\{h \mid Ac(h,e)\}$. This verifies condition CA2.

We are now in a position to link up these results with our hypothesis discovery methods. There is one small point. We will temporarily use e' rather than e to stand for the phenomena and keep e for the uses described above.

Only certain special e' and f are considered.

$e' = \{\alpha_{ki'}P_k(a_{i'}) \mid i' = 1,n\}$ where the $a_{i'}$ are all different constant terms and no $\alpha_{ki'}$ is $\mathcal{B}$.

$Ev(\alpha_{ki'}P_k(a_{i'})) = \bigwedge_{i=1}^{k=1} \alpha_{ii'}P_i(a_{i'})$ where no $\alpha_{ii'}$ is $\mathcal{B}$. Then e and $C_{1(e)}$ are defined as above, and the above results are available. Note that $\vdash e \equiv \bigwedge_{i'=1}^{n}(e_{i'} \wedge f_{i'})$. This e' and f satisfy the conditions for providing a discovery problem.

The next theorem shows that if a certain mild restriction on $\rightarrow$ holds, then any solution is acceptable. This condition is that if H is a solution and C is in H, then for some $C_0$ in $H_0$, $C \leq C_0$. Thus H must not contain any completely unnecessary clause.

<u>Theorem 1</u> Suppose that $H \leq H_0$, $\forall H \wedge e$ is consistent, $n \geq n_0$ and that if C is in H there is a $C_0$ in $H_0$ such that $C \leq C_0$. Then $Ac(\forall H, e)$.

Proof Let D be in H. We may write $D = \bigcup_{s=1}^{t} D_s \cup D_0$ where

1) If $D_0 \neq \emptyset$, $D_0$ is ground.

2) $\bigcup_s D_s$ has no occurrences of constants.

3) If $t \geq s > 0$, $D_s \neq \emptyset$.

4) If $s \neq s'$, there is no variable common to $D_s$ and $D_{s'}$.

5) Each $D_s$ has exactly one variable symbol, $x_s$, say $(s > 0)$.

Consequently, $\vdash \forall D \equiv (\bigvee_{s=1}^{t} \forall_{x_s} D_s) \wedge D_0$. As $\forall D$ is consistent with e, so is some $\forall D_s (t \geq s \geq 0)$.

Suppose $D_0$ is consistent with e. Then $D_0 \neq \emptyset$ and as D subsumes some member, $C_{i'}$, say, of $H_0$, $D_0 \subseteq C_{i'}$. From the consistency of $D_0$ with e, it follows that $\alpha_{ki'} P_k(a_{i'}) \in D_0$ and so $\vdash e \rightarrow D_0$. Therefore as $n \geq n_0$, $Ac(D_0, e)$. As $\forall D$ is also an e-sentence and $\vdash D_0 \rightarrow \forall D$, $Ac(\forall D, e)$.

Suppose that $\forall_{x_s} D_s$ is consistent with e$(s > 0)$. As $D \leq C_{i'}$, $D_s \leq C_{i'} \equiv \neg Ev(\alpha_{ki'} P_k(a_{i'})) \vee \alpha_{ki'} P_k(a_{i'})$. For consistency to hold, $\alpha_{ki'} P_k(x_s)$ must be in $D_s$. Consider some arbitrary phenomenon, $\alpha_{ki'} P_k(s_{i''})$. Suppose $\alpha_{ki'} = \alpha_{ki''}$, then:

$$\vdash \bigwedge_{i=1}^{k} \alpha_{ii''} P_i(x_s) \rightarrow D_s.$$

Suppose, on the other hand that $\alpha_{ki'} \neq \alpha_{ki''}$. Then, by the consistency of $\forall_{x_s} D_s$ with e, $D_s \not\leq \neg Ev(\alpha_{ki''} P_k(a_{i''}))$. Therefore $\alpha_{ii''} P_i(x_s)$ is in $D_s$ for some i. So:

$$\vdash \bigwedge_{i=1}^{k} \alpha_{ii''} P_i(x_s) \rightarrow D_s.$$

Consequently, no matter what $i''$ is,

$$\vdash \bigwedge_{i=1}^{k} \alpha_{ii''} P_i(x_s) \rightarrow D_s.$$

Therefore, $\vdash \forall x \bigvee_{i''=1}^{n} \bigwedge_{i=1}^{k} \alpha_{ii''} P_i(x) \rightarrow \forall x_s D_s.$

So by the definition of $C_{1(e)}$, $\vdash C_{1(e)} \rightarrow \forall x_s D_s.$ We then see that $Ac(\forall x_s, D_s, e)$ and so $Ac(\forall D, e)$.

As D was any member of H, it follows that $Ac(\forall H, e)$ which concludes the proof.

Corollary 2 Suppose that H satisfies the following conditions, where $\prec$ is a lexicographic extension of $\prec_c$:

1) $H \leq H_0$

2) $\forall H \wedge e$ is consistent.

3) H is minimal wrt. $\prec$ amongst those sets of clauses satisfying conditions one and two.

If $n \geq n_0$ then $Ac(\forall H, e)$.

Proof It is immediate that H satisfies all the conditions of theorem 1, except perhaps, that if C is in H there is a $C_0$ in $H_0$ such that $C \leq C_0$. Suppose, to the contrary, that C is in H and $C \not\leq C_0$ for every $C_0$ in H. Then $H' = H \cup \{C\}$ satisfies conditions one and two of the hypothesis, but $H' \prec H$ and $H \not\prec H'$, which contradicts condition three. Therefore

all conditions are satisfied and the conclusion follows at once.

It follows from the work of Hintikka and Hilpinen that $n_0$ is calculable from the number of predicate symbols, k and also $\varepsilon$. Therefore for sets of clauses satisfying the conditions of theorem 1, there is a method of deciding acceptability. We have therefore, for a restricted class of cases, answers to questions H1 (on when a hypothesis is justified) and H2 (on how to tell if a hypothesis is justified) of chapter 1 which satisfy the strong coherence condition that generated hypotheses be acceptable. This answer to H2 is evidently complete and consistent.

There are a number of insufficiencies in our analysis. First the notion of acceptance is based on just one of Hintikka's inductive systems. One would really want to have results not only for the $\alpha$-continuum but also the whole two-dimensional $\alpha$ - $\lambda$-continuum (Hintikka, 1966). Negative results would hold for that part corresponding to Carnap's $\lambda$-continuum (Carnap 1952), since no generalisation, containing variables, is acceptable there.

It is also natural to try to dispense with the assumption that all the individuals are completely observed, that is that every $\alpha_{ii'} \neq \beta$.

Both of these insufficiencies could probably be remedied with the aid of Hilpinen's (1968) monograph. Only then could we have a firm conclusion in the case of monadic logic. It would still remain

to extend the results to richer languages and this in turn requires a generalisation of the definitions of acceptance not yet attempted. Perhaps, however, some use could be made of the work on axioms for rules of acceptance (Kemeny 1953, Putnam 1963) but this is a mere speculation.

Important, here, would be acceptance relative to other than singular propositions. One would wish to know whether or not $Ac(\forall H, Th \wedge e)$ for example. If Th contained axioms for equality, e could contain distinctness information, of the form $a_i \neq a_{i'}$ (when $i \neq i'$, of course). This would remove a difficulty, alluded to above, in the presentation of Hintikka and Hilpinen.

We turn next to arguments designed to show that although it is necessary that explanation is justified, one cannot, without some difficulty, formulate conditions sufficient for the rational choice of an explanation in terms of justification. These arguments are elaborations of those advanced in chapter 2 to provide an opening for the use of simplicity.

A reasonable-seeming Carnapian move would be to set $H_1 \prec' H_2$ iff $H_1$ has a greater probability (in some sense) than $H_2$, given the knowledge $Th \wedge Irr$ and all the phenomena f and their circumstances e. However in this case $H_0$ will have a probability of one, and so is a best solution. If the notion of probability being used is reasonable then a hypothesis H will have unit probability relative to all knowledge etc. if and only if it follows from Th and Irr and e and f. Therefore

all the solutions will follow from what is given. Thus nothing new could ever be hypothesized which is absurd.

A more sympathetic formulation would require not only that an explanation generalise $H_0$ and be consistent with Th and Irr and $\bigwedge(e_i \wedge f_i)$, but also that it be general as opposed to ground. In view of previously discussed difficulties with the notion of a general law and since in this case, we do not wish to allow ground clauses as a degenerate case, let us specify that Th and Irr are empty, that no function symbols, other than constants, occur in e or f and that H is general if it contains no constants. Let $e' = \bigwedge_{i=1}^{n}(e_i \wedge f_i)$.

It seems reasonable to assume that if $H_1$ is general and $H_1 \leq H_2$ but $H_2 \not\leq H_1$, then probability $(H_1,e') <$ probability$(H_2,e')$.

In these circumstances the solutions are also maximal with respect to $\leq$.

Let $H_1$ be obtained from $H_0$ by replacing distinct constants by distinct variables. We may see that if $H \leq H_0$ and H is general, then $H \leq H_1$. As solutions are maximal with respect to $\leq$ every solution must in fact be equivalent to $H_1$. There are therefore two possibilities: either $H_1 \wedge \bigwedge_i (e_i \wedge f_i)$ is inconsistent and there is no solution or else $H_1$ is a solution and any other one is equivalent to it. This seems counter-intuitive. For example, suppose that $f = \{Q(a_i) \mid i=1,2n\}$, $Ev(Q(a_{2i})) = P_1(a_{2i}) \wedge P_2(a_{2i})$ $(i=1,n)$ and $Ev(Q(a_{2i-1})) = P_1(a_{2i-1}) \wedge P_3(a_{2i-1})$ $(i=1,n)$.

Here, $H_1 \sim \{\{\bar{P}_1(x)\ ,\ \bar{P}_2(x)\ ,\ Q(x)\}, \{\bar{P}_1(x)\ ,\ \bar{P}_3(x)\ ,\ Q(x)\}\}$.

For large enough n, one would expect a guess to be made that both $P_2$ and $P_3$ are irrelevant to the truth of Q.

A reasonable Popperian move would be to set $H_1 \prec' H_2$ iff $H_1$ is more falsifiable than $H_2$ given Th $\wedge$ Irr and f and e. In this case it is quite clear that there can never be a solution. If $H_2$ is proposed as a solution, we need merely find an $H_1$ such that $\forall(H_1 \cup H_2)$ is consistent with Th $\wedge$ Irr $\wedge \bigwedge_i (e_i \wedge f_i)$, and $\forall(H_1 \cup H_2)$ is more falsifiable than $\forall H_1$. This can always be done, as $H_1$ can contain arbitrary assertions as long as it has no vocabulary in common with any of $H_2$, Th, Irr or $\bigwedge(e_i \wedge f_i)$. We need a more sympathetic interpretation.

$H_1$ is an <u>irredundant</u> generalisation of $H_2$ relative to Th iff $H_1 \leq H_2$ (Th) and if $H_3 \subseteq H_1$ and $H_3 \leq H_2$ (Th) then $H_3 = H_1$. (This use of the word "irredundant" is distinct from that in chapter 4.) It is now required as an extra condition for a solution that H be an irredundant generalisation of $H_0$. This will prevent the above absurdities. It seems reasonable to assume that if $H_1 \leq H_2$ (Th) but $H_1$ does not follow logically from $H_2$, $\bigwedge_i (e_i \wedge f_i)$, Th and Irr then $H_1$ is more falsifiable than $H_2$ relative to e and f. Any solution will then be maximally general relative to Th. In other words, the most falsifiable hypotheses satisfy these three conditions (with $\prec$ equal to generalisation relative to Th):

1) H is an irredundant generalisation of $H_0$, relative to Th.

2) $\forall H \wedge Th \wedge Irr \wedge \bigwedge_{i=1}^{n}(e_i \wedge f_i)$ is consistent.

3) H is minimal, with respect to $\preceq$, amongst those sets of clauses satisfying conditions one and two.

Unfortunately, we do not know if there are any solutions, even in the case where Th and Irr are empty and there are no function symbols, other than constants, in e and f. On the other hand if we take $\preceq$ to be the lexicographic product of the simplicity ordering, $\preceq_s$, (the number of symbol occurrences) and generalisation relative to Th then there is always a finite number of solutions, to within equivalence relative to Th. This can be seen using a technique similar to that in chapter 6, section 5.

It seems worthwhile spending some effort on the problem where $\preceq$ is relative generalisation. If reasonable hypotheses could be produced with this niceness ordering, one would have a good argument against the necessity of an explicit syntactical simplicity ordering (Goodman, 1961). Alternatively, one might be able to produce arguments correlating simplicity with falsifiability in accordance with the views of Popper.

Finally, we give an example of a loose connection between confirmation theory and hypothesis discovery methods. Sometimes arguments are produced that certain evidence confirms certain hypotheses, against one's intuitions. These are paradoxes of confirmation. Two famous ones are Goodman's (1965) and Hempel's (1945). It may be

possible to show that the same hypotheses could be discovered from the same evidence and that this too is paradoxical. Such is the case with Goodman's paradox, of which a brief version can easily be stated. Suppose many emeralds have been examined and all of them are found to be green. This would seem to strongly confirm the hypothesis that all emeralds are green. All the emeralds must have been examined before some time, say the year 2000. Call a thing grue if and only if it has been examined before the year 2000, and found to be green, or if it has not been examined before then and is blue. Evidently all the emeralds examined are also grue. So just as strong confirmation is provided for the hypothesis that all emeralds are grue. These two hypotheses are contradictory. Although the grue hypothesis seems absurd, the absurdity has proved highly resistant to attempts at dissolution. Perhaps the best proposal was given by Goodman himself when he proposed the paradox.

Let us take $f = \{\text{Green}(em_i) \mid i=1,n\}$ and
$\text{Ev}(\text{Green}(em_i)) = \text{Emerald}(em_i) \wedge \text{Examinedby}(em_i, 2000)$, $(i=1,n)$.

Certainly, $\{\{\neg\, \text{Emerald}(x), \text{Green}(x)\}\} \leq H_0$.

Now suppose Th contains the definition:

$$\text{Grue}(x) \equiv (\text{Examinedby}(x,2000) \rightarrow \text{Green}(x)) \wedge (\neg\, \text{Examinedby}(x,2000) \rightarrow \text{Blue}(x)).$$

Then, $\{\{\neg\, \text{Emerald}(x), \text{Grue}(x)\}\} \leq H_0$ (Th).

Simple syntactical definitions of simplicity will not distinguish

the two hypotheses. One might hope to distinguish the Grue hypothesis as more complex, if complexity is measured after Grue is replaced by its definition. This will not do if Th is regarded as an unstructured set of sentences. Let Grue $\equiv \Delta$(Green,Blue) be the form of the definition of Grue. Suppose Th contains in addition the statement, Bleen $\equiv \Delta$(Blue,Green). Then both Green $\equiv \Delta$(Grue,Bleen) and Blue $\equiv \Delta$(Bleen,Grue) are logical consequences of Th and so there is complete symmetry between Grue and Green.

It seems therefore that Th must be given some structure and Grue be regarded as given by a definition. Even then, the proposal to count simplicity after the replacement of Grue by its definition seems to be an unwarranted bias in favour of particular predicates as, say, observational rather than theoretical. We are being lead, quite quickly, into wider issues. Note for example that focussing only on the stage of hypothesis formation causes distortion. There must also be a stage of forming definitions and theory of how definitions interact with discovery and justification. Goodman's proposed solution consisted of proposals involving these stages. All in all, the discussion of the paradox of discovery is much the same as that of confirmation. To put it briefly, the confirmation paradox concerns which predicates should be projected; the discovery paradox concerns how to select predicates for projection.

There is one practical point. The "wrong" predicates should somehow be avoided at either the stage of definition or else that of discovery. For if any definition is allowed and any of the simplest explanatory

hypotheses, there will generally be infinitely many hypotheses still to be eliminated at the stage of justification. For example: consider the definition

$$\text{Gruet}(x,t) \equiv (\text{Examinedby}(x,t) \rightarrow \text{Green}(x)) \wedge (\neg\, \text{Examinedby}(x,t) \rightarrow \text{Blue}(x)).$$

Now none of the infinitely many hypotheses $\text{Emerald}(x) \rightarrow \text{Gruet}(x,t_0)$, - where $t_0$ is a constant greater than 2000 - will be eliminated.

## 2. Two extensions to a more complex kind of theory, Th

### 2.1 Sorted languages

The first extension concerns languages with several disjoint sorts. It is a matter of folklore that the usual unification procedure automatically takes account of the sort restriction. Similarly the procedure for generalising two literals works when dealing with sorts, provided only that when matching distinct terms, the new variable substituted should be of the same sort as the old ones. With this one difference, all of the theory goes through without any trouble. Instead of putting the sort restrictions in the language we could let Th include the usual sort axioms using unary predicate symbols. One could then show that this leads to essentially the same results as the somewhat neater linguistic procedure.

### 2.2 Algorithms for a simple kind of theory: ground clauses

The second extension concerns algorithms for finding l.g.g's when Th is an arbitrary consistent, finite non-empty set of ground literals. We will give the essential theorems and algorithms for literals and clauses.

The first theorem gives the result for literals.

We define a function $\inf_{Th}$ by:

If $\vdash_{Th} \forall M$ then $\inf_{Th}\{M,N\} = N$. Otherwise, if $\vdash_{Th} \forall N$ then $\inf_{Th}\{M,N\} = M$. Otherwise, if M and N have the same predicate letter and sign then, $\inf_{Th}\{M,N\} = \inf\{M,N\}$. Otherwise, $\inf_{Th}\{M,N\} = \bar{L}_1$

($L_1$ is some fixed member of Th).

Theorem 1 1 $L \leq M$ (Th) iff $\vdash_{Th} \neg \forall L$ or $\vdash_{Th} \forall M$ or for some substitution $\sigma$, $L\sigma = M$.

2 $L \sim M$ (Th) iff either $\vdash_{Th} \forall M \wedge \forall L$ or else $\vdash_{Th} \neg \forall M \wedge \neg \forall L$ or else $L \sim M$.

3 Every pair of literals M and N has a least generalisation relative to Th, $\inf_{Th}\{M, N\}$.

Proof 1 $L \leq M$ (Th) iff for some $\sigma$ $\vdash_{Th} L\sigma \rightarrow M$

iff $\vdash_{Th} L\sigma\delta \rightarrow M\delta$ (where $\delta = \{a_1()/x_1, \ldots, a_n()/x_n\}$, the $x_i$ are the free variables of $L\sigma \rightarrow M$ and none of the $a_i$ occur in Th or $L\sigma \rightarrow M$)

iff $\emptyset \in \mathcal{R}(\{L_i \mid L_i \in Th\} \cup \{L\sigma\delta, \bar{M}\delta\})$.

There are now three possibilities, as Th is consistent. First, for some i, $\emptyset \in \mathcal{R}(\{L_i, L\sigma\delta\})$. This is equivalent to $\vdash_{Th} \neg \forall L$.

Second, for some i, $\emptyset \in \mathcal{R}(\{L_i, \bar{M}\delta\})$. This is equivalent to $\vdash_{Th} \forall M$.

Lastly, $\emptyset \in \mathcal{R}(\{L\sigma\delta, \bar{M}\delta\})$. This is true iff $L\sigma\delta = M\delta$ which is equivalent to $L\sigma = M$. This concludes the proof of the first part.

2 Suppose $L \sim M$ (Th). Then as $L \leq M$ (Th) either $\vdash_{Th} \neg \forall L$ or $\vdash_{Th} \forall M$ or, for some $\sigma$, $L\sigma = M$.

Suppose that $\vdash_{Th} \neg\forall L$. As $M \leq L$ (Th) either $\vdash_{Th} \neg\forall M$ or $\vdash_{Th} \forall L$, for some $\mu$, $M\mu = L$. The second case is impossible and if $M\mu = L$ then, as $\vdash_{Th} \neg\forall L$, $\vdash_{Th} \neg\forall M$. Hence in this case $\vdash_{Th} \neg\forall L \wedge \neg\forall M$.

Suppose that $\vdash_{Th} \forall M$. As $M \leq L$ (Th) the possibilities are that $\vdash_{Th} \forall L$ or $M\mu = L$ for some $\mu$. Now $M\mu = L$ and $\vdash_{Th} \forall M$ implies $\vdash_{Th} \forall L$. Hence in this case $\vdash_{Th} \forall L \wedge \forall M$.

Suppose that $L\sigma = M$. As $M \leq L$ (Th) either $\vdash_{Th} \neg\forall M$ or $\vdash_{Th} \forall L$ or for some $\mu$, $M\mu = L$. In the first case it follows that $\vdash_{Th} \neg\forall L$, the second that $\vdash_{Th} \forall M$ and in the third that $L \sim M$. Hence in this case $\vdash_{Th} \forall L \vee \forall M$ or $\vdash_{Th} \neg\forall L \wedge \neg\forall M$ or $L \sim M$. This concludes the proof of the second part.

3 If $\vdash_{Th} \forall M$ then $N \leq M$ (Th) and so $N = \inf_{Th}\{M,N\}$ is a l.g.g. of M and N relative to Th. If $\vdash_{Th} \forall N$, the proof is similar.

Suppose that M and N have the same predicate letter and sign and neither $\vdash_{Th} \forall M$ nor $\vdash_{Th} \forall N$ then $\inf\{M,N\}$ is defined and as $\inf\{M,N\} \leq M$, $\inf\{M,N\} \leq M$ (Th). Similarly, $\inf\{M,N\} \leq N$ (Th). Suppose that $L \leq M$ (Th) and $L \leq N$ (Th). Either $\vdash_{Th} \neg\forall L$ or else there are $\sigma$, $\mu$ such that $L\sigma = M$ and $L\mu = N$. In the first case $L \leq \inf\{M,N\}$ (Th). In the second, $L \leq \inf\{M,N\}$ and so $L \leq \inf\{M,N\}$ (Th). Thus $\inf_{Th}\{M,N\}$ is a l.g.g. of M and N relative to Th in this case.

Suppose that M and N differ in predicate letter or sign and that neither $\vdash_{Th} \forall M$ nor $\vdash_{Th} \forall N$. Then if $L \leq M$ (Th) and $L \leq N$ (Th),

the only possibility is that $\vdash \neg\forall L$. Then $L \leq L_1$ (Th). Since $L_1 \in Th$, $\vdash_{Th} \neg\forall \bar{L}_1$ and so $\bar{L}_1 \leq M$ (Th) and $\bar{L}_1 \leq N$ (Th). Therefore $\inf_{Th}\{M,N\}$ is a l.g.g. of M and N relative to Th in this, the last case.

This concludes the proof of the theorem.

Let $\overline{Th}$ be the clause $\{\bar{L} \mid L \in Th\}$. It follows from the characterisation of relative generalisation given by theorem 3.1.3.1 that $C \leq D$ (Th), iff D is a tautology, $C\sigma \subseteq D \cup \overline{Th}$ for some $\sigma$ or $Th \cap D \neq \emptyset$.

Let $L_1$ be some member of Th. Reduction will be defined relative to this choice. A clause C is reduced relative to Th iff $C = \{L_1\}$ or C is not a tautology, $C \cap Th = \emptyset$ and for any $C' \subseteq C$, $C \leq C'$ (Th) implies $C = C'$.

A clause, E is a <u>reduction</u> of a clause C, relative to Th, iff when $C \cap Th \neq \emptyset$ or C is a tautology, $E = \{L_1\}$ and, otherwise E is a reduced subset of C equivalent to C.

We define a function $\inf_{Th}\{C,D\}$. If D is a tautology or if $D \cap Th \neq \emptyset$ then $\inf_{Th}\{C,D\} = C$. Otherwise, if C is a tautology or if $C \cap Th \neq \emptyset$ then $\inf\{C,D\} = D$. Otherwise, $\inf_{Th}\{C,D\} = \inf\{C \cup \overline{Th}, D \cup \overline{Th}\}$.

There should be no confusion between the function defined here and the function $\inf_{Th}$ defined above on sets of literals with two elements. Notice that $\inf_{Th}\{L,M\} \sim \inf_{Th}\{\{L\},\{M\}\}$ (Th).

<u>Theorem 2 1</u> The following algorithm stops. It gives a clause $E_1$ which is a reduction of C, relative to Th:

1) Set $E_2$ to C.

2) If $E_2$ is a tautology or $E_2 \cap Th \neq \emptyset$ then set $E_1$ to $\{L_1\}$ and stop.

3) Set $E_1$ to $\emptyset$.

4) If $E_2$ is empty, stop.

5) Choose a literal, L, in $E_2$.

6) If there is a substitution, $\sigma$, such that $E_2\sigma \subseteq (E_1 \cup E_2 \cup \overline{Th}) \setminus \{L\}$ and $M\sigma = M$ for every literal M in $E_1$ then change $E_2$ to $E_2\sigma \setminus (E_1 \cup \overline{Th})$. Otherwise remove L from $E_2$ and add it to $E_1$.

7) Go to 4.

2 If $C \sim D$ (Th) and C and D are reduced, then they are alphabetic variants.

3 Every pair of clauses C and D has a l.g.g. relative to Th, $\inf_{Th}\{C,D\}$.

Proof 1 If C is a tautology or $C \cap Th \neq \emptyset$ then the algorithm outputs $\{L_1\}$ which is reduced and equivalent, relative to Th, to C. Otherwise, the proof that the algorithm works is a slight elaboration of the proof of theorem 3.3.1.1.

2 Suppose that C and D are reduced and equivalent, relative to Th. If $C = \{L_1\}$ then as $C \leq D$ (Th) either $C\sigma \subseteq D \cup \overline{Th}$ for some $\sigma$, D is a

tautology or $D \wedge Th \neq \emptyset$. In the first case, $L_1 = L_1\sigma \in D \cup \overline{Th}$. As $L_1 \in Th$ and Th is consistent, $L_1 \in D$. The other two cases are inconsistent with D's being reduced. Therefore the only possibility consistent with D's being reduced is that $D = \{L_1\}$. Similarly, if $D = \{L_1\}$, $C = \{L_1\}$. Therefore when either one of C and D is $\{L_1\}$ so is the other which proves part 2 in these cases.

Suppose, then, that there are $\sigma$, $\mu$ such that $C\sigma \subseteq D \cup \overline{Th}$ and $D\mu \subseteq C \cup \overline{Th}$, neither C nor D are tautologies and $C \wedge Th = \emptyset$ and $D \wedge Th = \emptyset$. Then $C\sigma\mu \subseteq \overline{Th}\mu \cup D\mu \subseteq \overline{Th} \cup C$. Let $C_1 = \{L \in C \mid L\sigma\mu \in \overline{Th}\}$ and $C_2 = C \setminus C_1$. Since C is reduced, $C_1 = \emptyset$ and $C_2 = C$. Therefore as $\{L \in C \mid L\sigma \in \overline{Th}\} \subseteq C_1$, $C\sigma \subseteq D$. Similarly $D\mu \subseteq C$. Under the conditions that neither C nor D are tautologies and $C \wedge Th = D \wedge Th = \emptyset$. C is reduced relative to Th implies C is reduced and similarly for D. Therefore, by theorem 3.3.1.1, C and D are alphabetic variants.

3 If D is a tautology or if $D \wedge Th \neq \emptyset$ then $C \leq D$ (Th) and so C is a l.g.g. of C and D relative to Th.

When C is a tautology or if $C \wedge Th \neq \emptyset$, the proof is similar.

In the last case, $\inf\{C \cup \overline{Th}, D \cup \overline{Th}\}$ is certainly a lower bound of C and D, relative to Th. Suppose that $E \leq C$ (Th) and $E \leq D$ (Th). If $E \leq C \cup \overline{Th}$ and $E \leq D \cup \overline{Th}$ then $E \leq \inf\{C \cup \overline{Th}, D \cup \overline{Th}\}$. As this is the only possible case, the proof is finished.

It would indeed be desirable to extend these results to a Th with general literals. However great difficulties arise. There is no trouble as regards l.g.g's of literals. Theorem 1 is true as it stands and the proof hardly needs any alteration. One can obtain analogues to the first two parts of theorem 2. The trouble comes with part 3. No l.g.g. relative to Th can exist. Here is a counterexample.

Let $Th = \{ \forall xx' \bar{P}(g(x),x'), \forall xx' \bar{P}(f(x),x') \}$. There is no l.g.g. of $Q(f(a()))$ and $Q(g(a()))$ relative to Th.

Let $g_1,\ldots,g_i,\ldots$ be an infinite sequence of distinct unary function symbols each of which is also distinct from f and g.

Let $C_n = \{Q(x)\} \cup \{P(x,g_i(x)) \mid i=1,n\}$.

Now $C_n \leq \{Q(f(a()))\}$ (Th). Figure 1 gives a $G_i$-derivation of $\{Q(f(x))\}$. (See chapter 3, section 1.3).

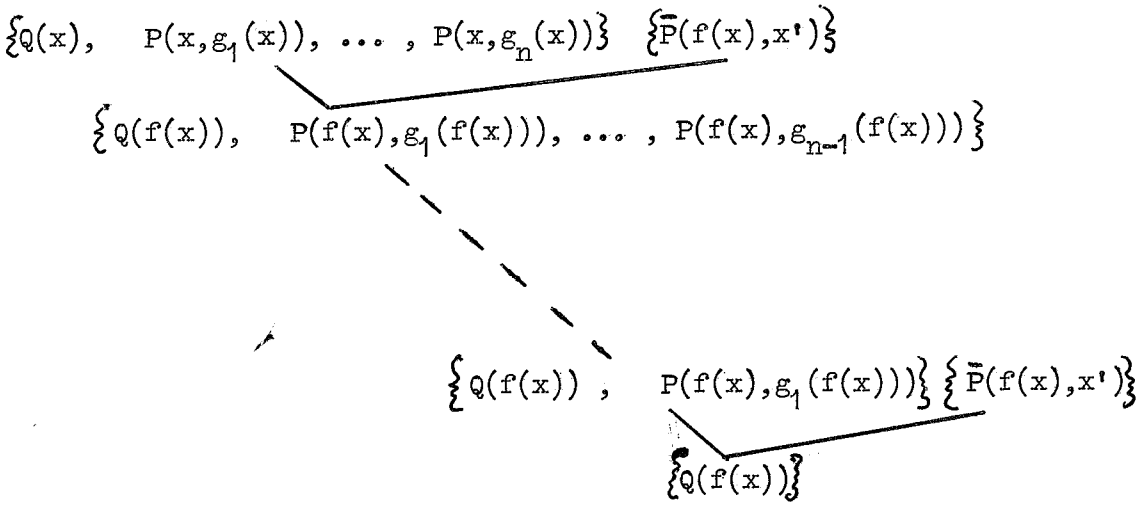


Figure 1

Similarly, $C_n \leq \{Q(g(a()))\}$ (Th). Suppose that D is a l.g.g. of $\{Q(f(a()))\}$ and $\{Q(g(a()))\}$. D must contain a variant of $\{Q(x)\}$ and can contain no literal of the form Q(t) where t has an occurrence of a function symbol. Since $C_n \leq D$ (Th) there must be a $C_n$-derivation of a clause subsuming D from $C_n$. This clause must contain a literal with predicate symbol Q. This literal may not, therefore, contain any function symbol. So the occurrence of x in $C_n$ may only be replaced by an occurrence of some other variable. Therefore no $\{P(x,g_i(x))\}$ may be resolved with a literal from Th.

Therefore D contains an occurrence of every $g_i$ in $C_n$. As this argument is independent of n, we have arrived at a contradiction. Therefore $\{Q(f(a()))\}$ and $\{Q(g(a()))\}$ can have no l.g.g. relative to Th.

We might hope that there is, say, an infinite set of non-equivalent, relative to Th, generalisations, relative to Th, of C and D with the property that if E is a generalisation, relative to Th, of C and D then it is a generalisation, relative to Th, of some member of the set. This would still allow some kind of computational procedure. However, one can show that if $E \leq C$ (Th) and $E \leq D$ (Th) there is an E', also a generalisation of C and D, relative to Th, such that $E \leq E'$ (Th), but $E' \not\leq E$ (Th). Thus there are not even any minimally general generalisations of C and D relative to Th. The technique is to choose a $C_n$ such that $g_n$ does not occur in E, and to let $E' = E\xi \cup C_n$ where $\xi$ standardises E apart from $C_n$. It is easy to show that $E' \leq C$ (Th) and $E' \leq D$ (Th). Evidently $E \leq E'$. If $E' \leq E$ (Th), one finds by

arguments, similar to those showing that C and D can have no l.g.g. relative to Th, that $g_n$ must occur in E which is a contradiction. One can obtain similar counterexamples by replacing $g_i(x)$ by $h(....h(x))$ where there are i occurrences of h, which avoids the use of infinitely many function symbols.

In general, therefore, there is no solution when Th contains general literals and $\mathcal{S}$ is $\mathcal{S}$ cpg, since relative least general generalisations do not always exist. It is in fact not difficult to produce an example, using the above techniques, where, although there is a consistent explanation, there is no best one.

One can do spectacularly better however, when $\mathcal{S}$ is $\mathcal{S}_s$. ($H_1$ $\mathcal{S}_s$ $H_2$ iff $H_1$ has no more symbol occurrences than $H_2$).

Suppose, for the moment, that there are only finitely many function and predicate symbols.

Let $x_1, ....$ be an infinite list of variables. One can find a list $H_1, H_2, ....$ of sets of clauses such that:

1) If $H_i$ has m variables, they are $x_1, ...., x_m$.

2) Given any H there is an $H_i$ which is an alphabetic variant of H.

3) If $i \leq j$ then $H_i$ $\mathcal{S}_s$ $H_j$.

Let Th and Irr be arbitrary, take $\mathcal{S}$ to be $\mathcal{S}_s$, and choose some e and f. Then the first $H_i$ in the list, which consistently explains

f given e and Th, is a solution to the resulting generalisation problem. There always is one, since $H_0$ is a consistent explanation of f given e and Th. Let this first explanation be $H_{i_1}$. As there are finitely many sets of clauses in the list with a given number of symbols, there is a finite set $\{H_i \mid H_i \mathrel{\mathcal{G}}_s H_{i_1}$ and $H_i$ is a consistent explanation of f given e and Th$\}$.

Every solution is a variant of some member of this set, and every member of the set is a solution. Whether or not the set is effectively obtainable depends, as usual, on the decidability of consistency.

It is possible in principle, therefore, to accept the non-existence of least generalisations, if one alters $\mathcal{G}$. One would wish however to use a $\mathcal{G}$ for which the solutions are easily obtainable. Certainly $\mathcal{G}_s$ is of no use if the algorithm which searches an infinite list has to be employed.

## 3. A general algorithm using a limited consistency check

In chapter 4 the unsolvability results were caused by the difficulty in checking for consistency. Meltzer (1970) has suggested that the requirements of consistency be replaced by the requirement that a determined effort has been made, but failed, to prove that $\forall H \wedge Th \wedge Irr \wedge \bigwedge_{i=1}^{n}(e_i \wedge f_i)$ is inconsistent.

We could still add on at the end a (perhaps computationally very expensive) test for consistency. If H passed this test we would have a solution. If it did not, we could recycle, making a more determined effort to check consistency. This combination might be quite practical.

For our problem we decided, following Meltzer, to formalize the 'determined' test as $\forall H \wedge Th \wedge \bigwedge_i (e_i \wedge f_i) \wedge Irr$ is 1-consistent, where 1-consistent means that a binary resolution theorem prover has not found a contradiction at level 1. The theory goes through much as before, and theorem 4.1 holds when the evidence changes have been made.

This theory does not parallel the procedure of Meltzer exactly. There are great differences in the way generalisations are found. He abstracts individual members of $H_0$ rather than combining them to form least generalisations. This is partly motivated by his Popperian niceness relation which prefers more to less general sentences.

We hand-simulated the method, with 1 set equal to ten, for the problem that Meltzer tried. The facts are represented using a binary predicate symbol **E**, for equality, and a binary function symbol f for

multiplication. Thus ab = cd is represented by two facts, $f_1 = \mathbf{E}(f(a,b),f(c,d))$ and $f_2 = \mathbf{E}(f(c,d),f(a,b))$ (similarly for ab ≠ cd). The corresponding $e_i$ are empty. Th was empty, but one took Irr to be the axioms for equality. It can be shown that $\mathbf{E}(t_1,t_2)$ is in the solution H iff $\mathbf{E}(t_2,t_1)$ is (similarly for $\overline{\mathbf{E}}(t_1,t_2)$) so we present the input and output in the ordinary notation.

The facts given were:

ee = e,

ae = a,

(aa)e = a(ae),

(ea)a = e,

ea ≠ e,

aa ≠ a,

bc = cb,

(bb)b = c,

(bb)c ≠ c,

(bc)c ≠ b.

The solution was:

1. $xe = x$,

2. $xa \neq x$,

3. $(aa)e = a(ae)$,

4. $(ea)a = e$,

5. $xy = yx$,

6. $(bb)c \neq c$,

7. $(bc)c \neq b$,

8. $(bb)b = c$.

Thus we found the right-identity and commutative laws. When we added.

$(ab)b = a(bb)$,

the solution was as above except that 3 was replaced by $(xy)z = x(yz)$, the associative law. This is as good a result as Meltzer obtained. Both methods obtained the right-identity and the commutative laws. Meltzer obtained a part of the associative law, viz.:

$(xx)y = w \supset x(xy)$.

If he altered his method so that occurrences of terms were abstracted, rather than abstracting on every occurrence at once, he would have obtained $(xy)z = w \supset x(yz) = w$ which is equivalent to the associative law in the presence of the axioms of equality.

Rather different laws, true only for the example groups, were found by the two methods.

What neither method does, however, is to use the equality axioms when forming generalisations. Preliminary investigation of generalisation relative to these axioms shows that this is not an easy problem.

## 4. Some pilot experiments

### 4.1 Description of the program

To test out our ideas on algorithms for forming generalisations, we have programmed a method which works on a case of the general problem, generated by the following assumptions:-

1) The language is sorted and there are no function symbols other than constants.

2) The niceness relation, $\succ$, is $\succ_{cpg}$.

3) The knowledge used for generalisation, Th, is empty.

4) $Irr \wedge \bigwedge_i (e_i \wedge f_i)$ is the conjunction of the literals in some Herbrand base of $Irr \wedge \bigwedge_i (e_i \wedge f_i)$.

5) Only one predicate symbol occurs in f.

From the discussion of the simple solvable case after corollary 2 in chapter 4 we see that consistency is easily checked. Indeed, if $\mathbf{E} = \{\bar{\mathbf{L}} \mid \mathbf{L}$ is a conjunct of $Irr \wedge \bigwedge_i (e_i \wedge f_i)\}$ then $\forall H \wedge Irr \wedge \bigwedge_{i=1}^{n} (e_i \wedge f_i)$ is consistent iff $H \not\subseteq \mathbf{E}$.

The program calculates a heuristic approximation to an irredundant explanation, H, rather than looking for a best one. This is to save time. The program is written in the POP-2 programming language (Burstall, Collins and Popplestone, 1971) and has been run on

the ICL 4130 machine. At the moment running times vary between three and fifteen minutes.

The program starts with H, the potentially irredundant explanation, set equal to $H_0$. It then continuously chooses a member, D, of H and a member $C_i$ of $H_0$ and replaces D by $\inf^*\{D,C_i\}$ a heuristic approximation to the reduced form of $\inf\{D,C_i\}$. This stops when any such replacement results in $\forall H \wedge \text{Irr} \wedge \bigwedge_i (e_i \wedge f_i)$ being inconsistent. Then H is output as the result.

The flowchart of the program is given in figure 1.

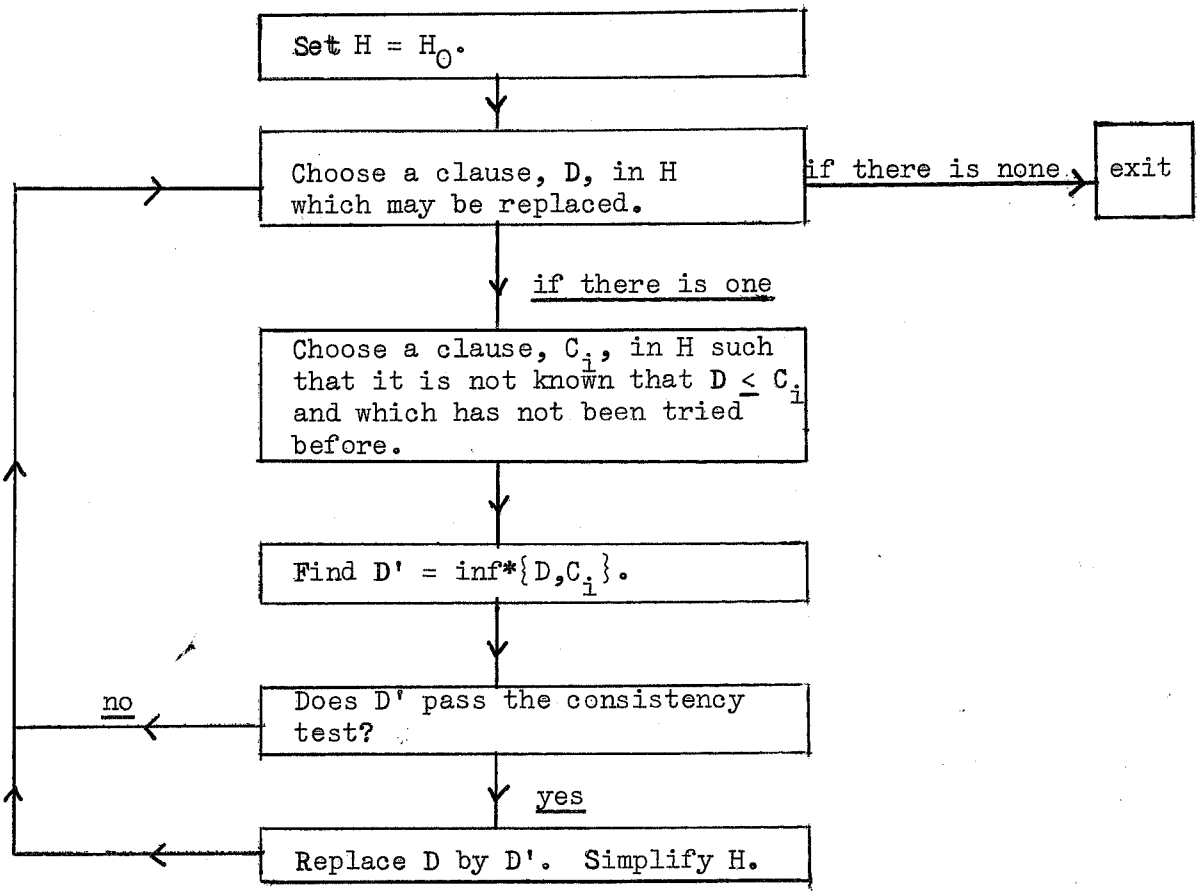


Figure 1

Various heuristic rules are used in making the choices mentioned in the flow chart.

Let Fail(D) = $\{C_i \mid C_i$ is in $H_0$ and in the course of building up D, an attempt to use $C_i$ failed$\}$.

Let Success(D) = $\{C_i \mid C_i$ is in $H_0$ and in the course of building up D, an attempt to use $C_i$ succeeded$\}$.

Initially, Fail(D) = $\emptyset$ and Success(D) = $\{D\}$, for any D in H. The clause, D, can only be selected if Fail(D) $\cup$ Success(D) $\neq H_0$. The program chooses a clause D in H with a largest Success(D) and of two such clauses prefers the one with the smaller failure set.

A clause $C_i$ in $H_0$ can only be chosen if it is not in Failure(D) $\cup$ Success(D). From these the program chooses a clause $C_i$ for which there are minimally many clauses D in H such that $C_i$ is in Success(D). This reflects our belief that the better-structured the problem the less likely it is that any clause in $H_0$ is generalised by more than one clause in a good explanation. Consequently we believe that the chance of failing, and so wasting a lot of computational effort, grows with the number of previous successes in explaining $f_i$ given $e_i$.

Next, $D^* = \inf\{D, C_i\}$ is calculated. An approximation to the reduced form of $D^*$ is found as follows. If $D \leq C_i$ then $D^*$ is D. Otherwise, $D^*$ is ordered into a list $L_1, \ldots, L_n, \ldots, L_m$ where any two distinct literals occurring in $L_1, \ldots, L_n$ have different predicate

symbol and sign pairs. Further if some predicate symbol and sign pair occurs in $D^{\prime\prime\prime}$ it occurs in $L_1,\ldots,L_n$. The literal $L_i$ has no more variables than any other literal with that predicate symbol and sign for i=1,n. For $n < i \leq m$, $L_i$ has less variables (not in $L_1,\ldots,L_{i-1}$) than in any other of the $L_i,\ldots,L_m$. The heuristic approximation to the reduced form of $D^{\prime\prime\prime}$ is the reduced version, calculated properly, of $\{L_i \mid 1 \leq i \leq \min(l,m)\}$, where l is a program parameter, which is set as large as possible without inconveniently long running times. We set l to be 11 throughout our experiments.

The consistency test checks whether $D^{\prime\prime\prime} \leq E$, with E as defined above. This is a necessary and sufficient condition for the consistency of H with $Irr \wedge \bigwedge_i (e_i \wedge f_i)$ since the other clauses in H will either already have been checked or else are in $H_0$.

Simplification is performed by continuous application of the following operation, until this is no longer possible:

Remove from H a clause, $D_1$ such that

$$Success(D_1) \subseteq \bigcup_{D_2 \in H \text{ and } D_2 \neq D_1} Success(D_2).$$

Notice that $D^{\prime\prime\prime}$ itself cannot be removed by this process. The program takes advantage of this fact.

When the program terminates, and it must do so, H is reduced. For suppose $D_1$ and $D_2$ are distinct clauses in H then and $D_1 \leq D_2$. Suppose that D is any clause in H then. Now since the program has terminated,

$H_0$ = Success(D) $\cup$ Fail(D). If C $\in$ Success(D) then D $\leq$ C. Otherwise, as $\forall H \wedge$ Irr $\wedge \bigwedge(e_i \wedge f_i)$ is consistent, D $\not\leq$ C. Therefore Success(D) = $\{C \in H_0 \mid D \leq C\}$. It follows that Success($D_2$) = $\{C \in H_0 \mid D_2 \leq C\} \subseteq \{C \in H_0 \mid D_1 \leq C\}$ = Success($D_1$). But then the simplification process would have removed $D_2$. This contradicts the fact that the program has terminated and establishes the conclusion.

On the other hand, H need not be irredundant at termination since inf* is not inf. If inf* were inf, it would be.

## 4.2 Evaluation of experimentation

It is now convenient to discuss why experimentation is helpful. First it serves to give more complex examples than can easily be produced by hand. Secondly, we can try to evaluate our hypothesis generation method. Some ways of evaluation will not be considered. We pay little attention to the efficiency of the program, since it has only been written to provide answers in a reasonable amount of time which varied, as stated above, between three and fifteen minutes for the examples described below. Neither will the behaviour of the hypothesis method through time be considered; we have only begun to investigate the possibilities theoretically (see chapter 6). Finally we do not investigate either theoretically or practically how useful the hypothesis generation method is to the organism employing it (see chapter 1).

The method will be judged by the hypotheses it produces. But

this is not too easy; by definition the method must produce the nicest explanatory hypothesis possible, in the sense of $\mathcal{H}_{cpg}$. Consequently experiment will merely confirm theorem 4.1. However we can test the correlation of $\mathcal{H}_{cpg}$ with other niceness relations. For illustration we will try $\mathcal{H}_1$, (see chapter 1 for a definition) and $\leq$ (this is a Popperian niceness relation, as discussed in section 1 of this chapter).

It would certainly be possible to try to calculate the predictive power of a generated hypothesis, when all cases and the correct hypothesis are known. However one should also take into account how good the information provided to the hypothesis generation machine is. Suppose, at one extreme, that no information is given (where $f = \emptyset$). Then one cannot expect any predictive power of a generated hypothesis. At the other extreme when all possible cases are given one would expect a hypothesis to possess total predictive power, as a simple consequence of its being an explanation.

More generally, one can only expect the hypothesis generated to do well in cases similar to those it is given information about. This would require what we do not possess: a method of giving a sense to the word "similar" induced by the correct hypothesis.

A partial way around this problem would be to compare with respect to predictive power one hypothesis generated by the generation method with another either generated by a different method or else by humans,

when both hypotheses are generated from the same information. This would require rather more than a pilot experiment.

What we shall do is award good marks according to how well the predictions of the generated hypothesis match those of the correct one. If the agreement is, intuitively, rather small and we can show, intuitively, that a "fair sample" of cases have been supplied, then we shall award a bad mark.

One should compare our difficulties with those which arise when evaluating Evans' Analogy program (Evans, 1968). As long as his program gives the "correct" answers, it is certainly doing well. If it gives no answer, it is certainly doing badly. But suppose it gives the "wrong" answer. Then whether it is doing well or not seems to depend on what its reasons for giving that answer were. Two comparisons are in order. First consider a program which simply always selects the first allowable answer figure. It will certainly never have any reason for making its choice. Next consider a clever person who knows the correct answer and deliberately tries to find good reasons for choosing some wrong answer. It is well known that most I.Q. tests can be so treated by an intelligent person! Perhaps therefore an analogy program should try to find as large a number of answers as possible and try to convince us that each was "correct".

## 4.3 An experiment using the win predicate of noughts and crosses

The aim of the experiment is to discover a sufficient set of

conditions for a position to be a win. The board is considered to be a three by three matrix. We use a language with two sorts: numbers and positions. There are three predicate symbols, XX, OO and Win. XX(i,j,p) is true iff position p has an X in square (i,j). The predicate OO is defined similarly. We can formulate a set $H_{correct} = \{\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3, \mathbf{E}_4\}$, of four clauses expressing sufficient conditions for a win:

$\mathbf{E}_1 = \{\overline{XX}(i,1,p), \overline{XX}(i,2,p), \overline{XX}(i,3,p), Win(p)\}$,

$\mathbf{E}_2 = \{\overline{XX}(1,i,p), \overline{XX}(2,i,p), \overline{XX}(3,i,p), Win(p)\}$,

$\mathbf{E}_3 = \{\overline{XX}(1,1,p), \overline{XX}(2,2,p), \overline{XX}(3,3,p), Win(p)\}$,

$\mathbf{E}_4 = \{\overline{XX}(1,3,p), \overline{XX}(2,2,p), \overline{XX}(3,1,p), Win(p)\}$.

These state, respectively, that a row, a column, a forward diagonal or a backward diagonal of X's is a sufficient condition for a win.

Consider the position, $p_1()$ say, displayed in figure 1.

| | | |
|---|---|---|
| | X | O |
| O | X | X |
| | X | O |

Figure 1

Certainly the fact, $f_1$, to be explained is

$f_1 = Win(p_1())$.

There are two possibilities for $\mathbf{Ev}(f_1)$. If we give the exact reasons why $f_1$ is a win then

$$\mathbf{Ev}(f_1) = XX(1,2,p_1()) \wedge XX(2,2,p_1()) \wedge XX(3,2,p_1()).$$

If we decide that the win depends on where the marks X and O are then

$$\begin{aligned}\mathbf{Ev}(f_1) = {} & XX(1,2,p_1()) \wedge XX(2,2,p_1()) \\ & \wedge XX(3,2,p_1()) \wedge OO(1,3,p_1()) \\ & \wedge OO(2,1,p_1()) \wedge OO(3,3,p_1()).\end{aligned}$$

We shall try both. There are certainly others in which, for instance we might include in $\mathbf{Ev}(f_1)$ the literal $\overline{OO}(1,1,p_1())$ or even $\overline{\overline{OO}}(1,2,p_1())$.

The choice of **Ev** is not prescribed by our theory and is but one of many omissions (see chapter 1).

Next, consider a loss, such as the position, $p_2()$, displayed in figure 2.

| | | |
|---|---|---|
| | X | X |
| O | O | O |
| X | | X |

Figure 2

In this case we simply include a complete description of the

figure in Irr. So Irr will contain the conjunction:

$$\bigwedge_{j=1}^{3} \overline{OO}(1,j,p_2()) \wedge \bigwedge_{j=1}^{3} \overline{OO}(3,j,p_2())$$

$$\wedge \bigwedge_{j=1}^{3} \overline{XX}(2,j,p_2()) \wedge \overline{XX}(3,2,p_2())$$

$$\wedge \bigwedge_{j=1}^{3} OO(2,j,p_2()) \wedge \bigwedge_{j=2}^{3} XX(1,j,p_2())$$

$$\wedge XX(3,1,p_2()) \wedge XX(3,3,p_2()) \wedge \overline{Win}(p_2())$$

So given a set of won or lost positions, we have shown how f, Ev and Irr are obtained. However the resulting problem will not satisfy assumption 3, given in the description of the program, since we have not included a complete description of every won position in Irr.

Let $\mathbf{E} = \{\bar{L} \mid L \text{ is a conjunct of Irr} \wedge \bigwedge_{i=1}^{n}(e_i \wedge f_i)\}$ as defined there; let Irr' be Irr together with a complete description of every win and let $\mathbf{E}' = \{\bar{L} \mid L \text{ is a conjunct of Irr}' \wedge \bigwedge_{i=1}^{n}(e_i \wedge f_i)\}$. Now if only a single position variable occurs in a clause C which contains only positive occurrences of the Win predicate, then the reader should verify that $C \leq \mathbf{E}$ iff $C \leq \mathbf{E}'$. Now every clause in $\mathcal{G}_{\mathcal{S}}(H_0)$ has this character and so we conclude that we may safely use the program with Irr as defined. This represents a useful computational saving.

The first result is in a case where Ev gave the exact reasons and the positive wins were as displayed in table 1. We took all possible

non-wins to form Irr. The calculation was performed by hand, since it is easy in this case to generate all solutions, rather than just one heuristic approximation to an irredundant set of clauses.

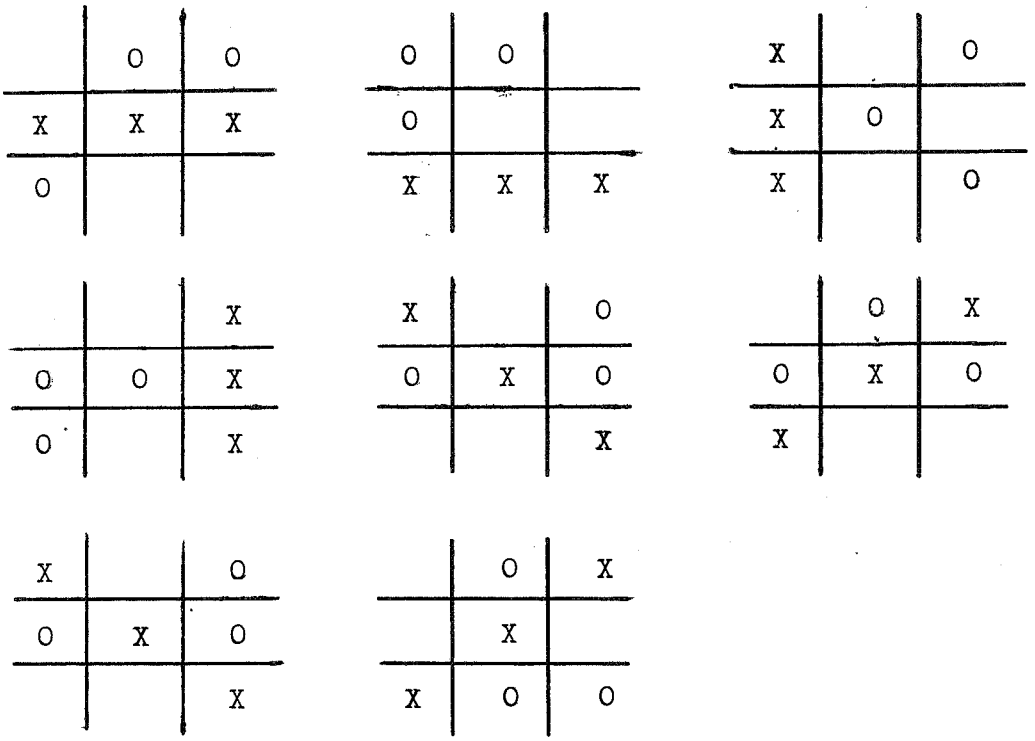

Table 1

There is exactly one solution, $H_{soln} = \{D_1, D_2, D_3, D_4\}$ where

$D_1 = \{\overline{XX}(i,1,p), \overline{XX}(i,2,p), \overline{XX}(i,3,p),$
$\overline{XX}(i,i,p), Win(p)\},$

$D_2 = \{\overline{XX}(1,i,p), \overline{XX}(2,i,p), \overline{XX}(3,i,p),$
$\overline{XX}(i,i,p), Win(p)\},$

$D_3 = \{\overline{XX}(1,1,p), \overline{XX}(2,2,p), \overline{XX}(3,3,p),$
$Win(p)\},$

$D_4 = \{\overline{XX}(1,3,p),\ \overline{XX}(2,2,p),\ \overline{XX}(3,1,p),$
$\quad Win(p)\}.$

Evidently a position is a win according to $H_{soln}$ iff it is according to $H_{correct}$. So $H_{soln}$ has good predictive power. It is not maximally nice with respect to either $\prec_1$, or $\leq$ since the literal $\overline{XX}(i,i,p)$ occurring in the clauses $D_1$ and $D_2$ is superfluous.

The next two examples use an Ev of the second sort, where we record all occurrences of O's and X's in each $Ev(f_i)$ $(i=1,n)$. For the first example, the win and non-win positions are displayed in tables 2 and 3 respectively. They are from an example given in Popplestone (1970).

| | | |
|---|---|---|
| | | X |
| | X | |
| X | | |

| | | |
|---|---|---|
| X | | |
| | X | |
| | | X |

| | | |
|---|---|---|
| O | X | O |
| | X | |
| | X | O |

| | | |
|---|---|---|
| X | | |
| X | | |
| X | | |

| | | |
|---|---|---|
| | | |
| X | X | X |
| | | O |

| | | |
|---|---|---|
| X | X | X |
| | | |
| | | |

Table 2

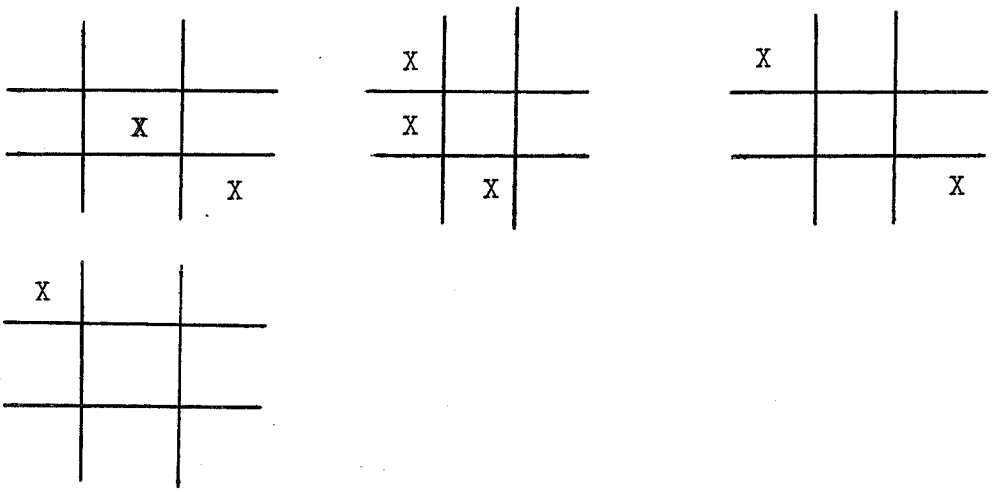


Table 3

The program found a set of clauses $H_{soln} = \{D_1, D_2\}$, where

$D_1 = \{\overline{XX}(i,3,p), \overline{XX}(m,i,p), \overline{XX}(m,1,p),$
$\overline{XX}(i,m,p), Win(p)\}$

and $D_2 = \{\overline{XX}(3,d,p), \overline{XX}(d,d,p), \overline{XX}(2,g,p),$
$\overline{XX}(g,g,p), \overline{XX}(i,i,p), \overline{XX}(1,i,p), Win(p)\}$.

The clause $D_1$ has as a consequence that any position containing a column of X's or a backward diagonal of X's is a win. The clause $D_2$ does the same for rows and the forward diagonal. In fact $H_{soln} \leq H_{correct}$. Therefore if $H_{correct}$ predicts that a position is a win, so will $H_{soln}$. However $H_{soln}$ makes some wrong predictions. For example, according to $D_1$ any position containing an X in (1,3) and in (3,1) is a win.

We cannot say if this is because we do not have examples of all the "ways" in which a position can fail to be a win since we do not have a good concept of such a "way". $H_{soln}$ is not maximally nice with respect to either $\prec_1$, or $\leq$ since one can remove $\overline{XX}(i,m,p)$ from $D_1$ without

affecting consistency with $\text{Irr} \wedge \bigwedge_i (e_i \wedge f_i)$.

The win and non-win positions for the second example are displayed in tables 4 and 5 respectively.

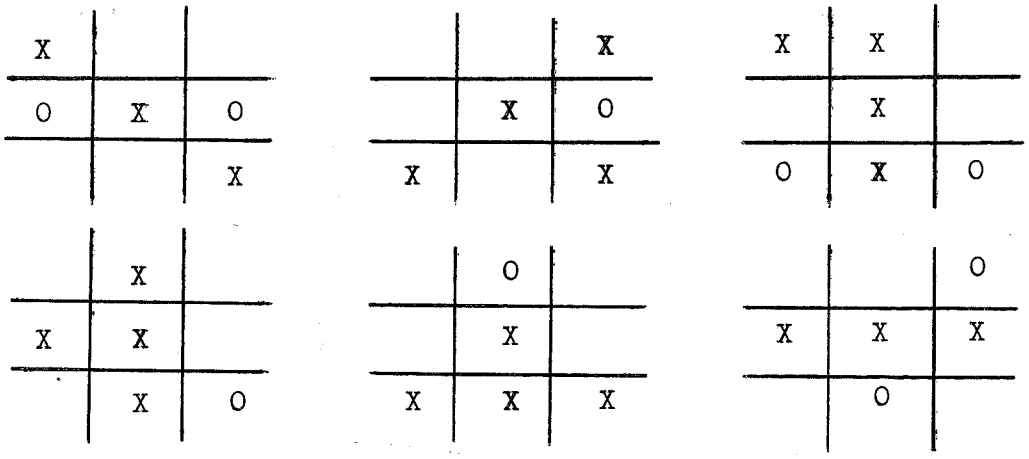

Table 4

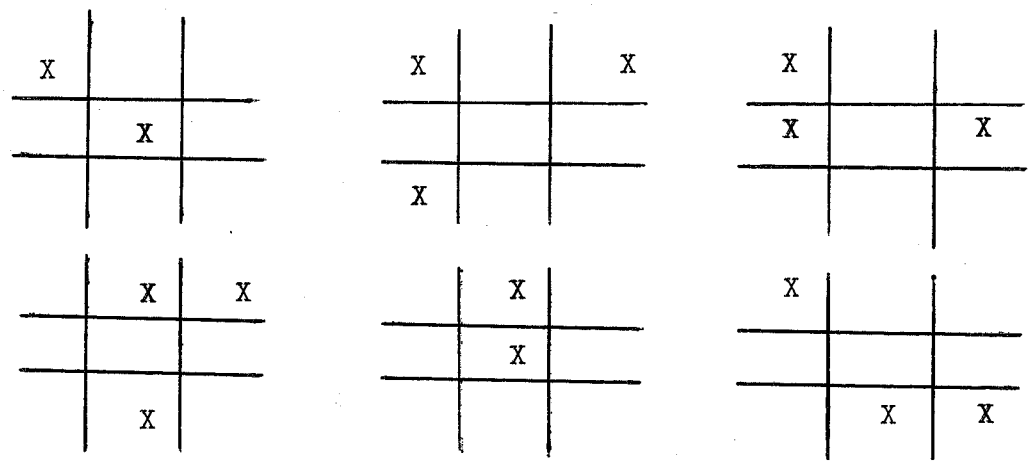

Table 5

The program found a set of clauses, $H_{soln} = \{D_1\}$ where

$D_1 = \{\overline{XX}(2,2,p), \overline{XX}(k,k,p), \overline{XX}(n,k,p), \overline{OO}(q,n,p), \text{Win}(p)\}$.

As we inadvertently did not put a 0 in any of the non-wins, it is possible to find an explanation with only one clause. This invalidates any comparison with $H_{soln}$. The solution is not maximally nice with respect to either $\prec_1$, or $\leq$ since the clause

$$\{\overline{OO}(q,n,p),\ Win(p)\}$$

would do just as well.

To sum up, generated hypotheses compare well with $H_{soln}$, but do not have maximal niceness with respect to either $\prec_1$, or $\leq$.

## 4.4 Learning the patrilineal ancestor relationship

The binary relation, patrilineal ancestor, is recursively defined by

$$Anc(x,y) \equiv Father(x,y) \vee \exists z(Father(x,z) \wedge Anc(z,y)).$$

We are using a language with one sort. There are three binary predicate symbols, Father, Daughter and Anc with evident meanings. The hypothesis generation machine is required to find sufficient conditions for one individual to be the patrilineal ancestor of another. The definition of Anc gives the set, $H_{cor} = \{E_1, E_2\}$, of sufficient conditions where

$$E_1 = \{\overline{Father}(x,y),\ Anc(x,y)\}$$

and $$E_2 = \{\overline{Father}(x,z),\ \overline{Anc}(z,y),\ Anc(x,y)\}.$$

A typical member of f has the form Anc(a,b). There are, again, at least two ways of choosing Ev(anc(a,b)). One is to give exact reasons. Another is to choose a reasonably small set of literals which establish a link between a and b. A link is a set of literals $\{L_k \mid k=1,l\}$ where each $L_k$ is of the form $P_k(a_k,b_k)$ or $P_k(b_k,a_k)$ where $a=a_1$, $b_k=a_{k+1}$ $(1\leq k\leq l-1)$ and $b_l=b$. In general we let Ev(Anc(a,b)) be a few of the smallest links between a and b.

Irr is simply a conjunction of all the true literals in the example under consideration.

In the examples, we were concerned with two families whose trees are displayed in figures 1 and 2.

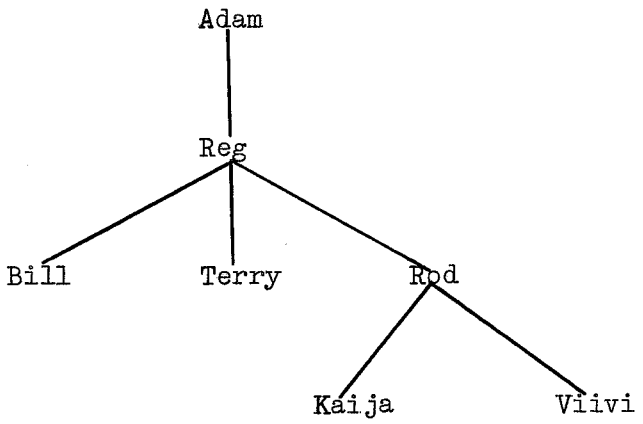


Figure 1


Figure 2

In order to define the Irr used in the various examples, without repetition, we define three sets of clauses $Irr_1$, $Irr_2$ and $Irr_3$, using some auxiliary sets.

Let $Irr_{1,A} = \{\{Anc(a,b)\} \mid a \in \{Adam,Reg\}, b \in \{Bill,Terry,Rod,Kaija,Viivi\}\}$
$\cup \{\{Anc(Adam,Reg)\}\}$
$\cup \{\{Anc(Rod,b)\} \mid b \in \{Kaija,Viivi\}\}$,

$Irr_{2,A} = \{\{Anc(Isa,Manuel)\}, \{Anc(Isa,Karen)\}, \{Anc(Manuel,Karen)\}\}$,

$Irr_{1,F} = \{\{Father(Reg,b)\} \mid b \in \{Bill,Terry,Rod\}\}$
$\cup \{\{Father(Rod,Kaija)\}, \{Father(Rod,Viivi)\}\}$,

$Irr_{2,F} = \{\{Father(Isa,Manuel)\}, \{Father(Manuel,Karen)\}\}$,

$Irr_{1,D} = \{\{Daughter(Kaija,Rod)\}, \{Daughter(Viivi,Rod)\}\}$,

$Irr_{2,D} = \{\{Daughter(Karen,Manuel)\}\}$,

$$Irr_1 = Irr_{1,A} \cup Irr_{1,F} \cup Irr_{1,D} \cup \{\{\bar{P}(a,b)\} \mid P \in \{Anc, Father, Daughter\}, a, b \in \{Adam, Reg, Bill, Terry, Rod, Kaija, Viivi\}, P(a,b) \notin Irr_{1,A} \cup Irr_{1,F} \cup Irr_{1,D}\},$$

$$Irr_2 = Irr_{2,A} \cup Irr_{2,F} \cup Irr_{2,D} \cup \{\{\bar{P}(a,b)\} \mid P \in \{Anc, Father, Daughter\}, a,b \in \{Isa, Manuel, Karen\}, P(a,b) \notin Irr_{2,A} \cup Irr_{2,F} \cup Irr_{2,D}\}$$

$$Irr_3 = \{\bar{P}(a,b) \mid P \in \{Anc, Father, Daughter\}, a \in \{Adam, Reg, Bill, Terry, Rod, Kaija, Viivi\}, b \in \{Isa, Manuel, Karen\}\}$$

$$\cup\{\bar{P}(b,a) \mid P \in \{Anc, Father, Daughter\}, a \in \{Isa, Manuel, Karen\}, b \in \{Adam, Reg, Bill, Terry, Rod, Kaija, Viivi\}\}.$$

In the first example, we used an Ev of the first kind. Ev and f are described in table 1; Irr was taken to be $Irr_1 \cup Irr_2 \cup Irr_3$.

The program output the set of clauses $H_{cor}$, described above. It is possible to prove that these are optimal with respect to both $\prec_1$ and $\leq$.

| f | e |
|---|---|
| $f_1$ = Anc(Rod,Kaija) | $e_1$ = Father(Rod,Kaija) |
| $f_2$ = Anc(Reg,Terry) | $e_2$ = Father(Reg,Terry) |
| $f_3$ = Anc(Reg,Kaija) | $e_3$ = Anc(Reg,Rod) ∧ Father(Rod,Kaija) |
| $f_4$ = Anc(Reg,Viivi) | $e_4$ = Anc(Reg,Rod) ∧ Father(Rod,Viivi) |
| $f_5$ = Anc(Isa,Karen) | $e_5$ = Anc(Isa,Manuel) ∧ Father(Manuel,Karen) |

Table 1

In the second example, we used an Ev of the second kind. Ev and f are described in table 2; Irr was taken to be $Irr_1$.

| f | e |
|---|---|
| $f_1$ = Anc(Rod,Kaija) | $e_1$ = Daughter(Kaija,Rod) ∧ Father(Rod,Kaija). |
| $f_2$ = Anc(Reg,Terry) | $e_2$ = Father(Reg,Terry). |
| $f_3$ = Anc(Reg,Kaija) | $e_3$ = Father(Reg,Rod) ∧ Daughter(Kaija,Rod)∧ Father(Rod,Kaija)∧ Anc(Rod,Kaija) ∧ Anc(Reg,Rod). |
| $f_4$ = Anc(Adam,Kaija) | $e_4$ = Anc(Adam,Reg) ∧ Anc(Reg,Kaija)∧ Anc(Rod,Kaija)∧ Father(Reg,Rod) ∧ Father(Rod,Kaija). |

Table 2

The program output a set of clauses, H = $\{D_1, D_2\}$ where

$D_1$ = $\{\overline{\text{Father}}(x,y),\ \text{Anc}(x,y)\}$,

$D_2$ = $\{\overline{\text{Anc}}(x,y),\ \overline{\text{Father}}(x,y),\ \text{Anc}(\text{Reg},y),$
$\overline{\text{Father}}(\text{Reg},\text{Rod}),\ \overline{\text{Father}}(\text{Rod},\text{Kaija}),\ \overline{\text{Anc}}(\text{Rod},\text{Kaija}),$
$\overline{\text{Anc}}(z,r),\ \overline{\text{Anc}}(r,\text{Kaija}),\ \text{Anc}(z,\text{Kaija})\}$.

Now H does not have the same explanatory power as $H_{cor}$. For example H $\not\vdash$ {Anc(Reg,Viivi} ↔ Anc(Reg,Rod ∧Anc(Rod,Viivi)).

Further H is obviously not optimal according to either $\prec_1$, or $\leq$ since, for example, one could remove $\overline{\text{Father}}(x,y)$ from $D_2$ retaining consistency. It is not optimal with respect to $\leq$ for another important reason: one could replace every occurrence of Kaija in $D_2$ by one of the variable w.

It is interesting to notice that $D_2$ is equivalent, relative to Irr, to the clause,

$$D_3 = \{\overline{\text{Anc}}(x,y),\ \overline{\text{Anc}}(y,\text{Kaija}),\ \text{Anc}(x,\text{Kaija})\}$$

This suggests that it would be interesting to extend the program so that it handles the simple kind of Th discussed in section 2.2 of this chapter.

The next example uses a little less biased evidence. Ev and f are described in table 3; Irr was taken to be $\text{Irr}_1 \cup \text{Irr}_2 \cup \text{Irr}_3$.

| f | e |
|---|---|
| $f_1$ = Anc(Rod,Kaija) | $e_1$ = Daughter(Kaija,Rod) ∧Father(Rod,Kaija). |
| $f_2$ = Anc(Reg,Terry) | $e_2$ = Father(Reg,Terry). |
| $f_3$ = Anc(Reg,Kaija) | $e_3$ = Father(Reg,Rod) ∧Daughter(Kaija,Rod) ∧Father(Rod,Kaija) ∧ Anc(Rod,Kaija) ∧ Anc(Reg,Rod). |
| $f_4$ = Anc(Reg,Viivi) | $e_4$ = Father(Reg,Rod) ∧ Daughter(Viivi,Rod) ∧ Father(Rod,Viivi) ∧ Anc(Rod,Viivi) ∧Anc(Reg,Rod). |
| $f_5$ = Anc(Isa,Karen) | $e_5$ = Anc(Isa,Manuel) ∧ Anc(Manuel,Karen)∧ Father(Isa,Manuel) ∧ Father(Manuel,Karen) ∧ Daughter(Karen,Manuel). |
| $f_6$ = Anc(Adam,Kaija) | $e_6$ = Anc(Adam,Reg) ∧ Anc(Reg,Kaija) ∧ Anc(Rod,Kaija) ∧ Father(Reg,Rod) ∧ Father(Rod,Kaija). |

Table 3

The program output a set of clauses, H = $\{D_1, D_2\}$ where

$D_1$ = $\{\overline{\text{Father}}(x,y),\ \text{Anc}(x,y)\}$,

$D_2$ = $\{\overline{\text{Father}}(w,u),\ \overline{\text{Father}}(u,z),\ \overline{\text{Anc}}(u,z),$

$\overline{\text{Anc}}(x,y),\ \overline{\text{Anc}}(y,z),\ \text{Anc}(x,z)\}$.

In fact H and $H_{cor}$ have the same predictive power: in any family tree they will both predict the patrilineal ancestors correctly given all the information about Father. More can be said. Suppose Th is a theory, expressing the tree-like structure of family trees, whose only predicate symbol is Father and look at the following definition obtained by changing the proposed sufficient condition into a necessary and sufficient one:

$$\begin{aligned}\mathrm{Anc}(x,z) \equiv\ & \mathrm{Father}(x,z) \vee \exists w,u,y(\mathrm{Father}(w,u) \\ & \wedge \mathrm{Father}(u,z) \wedge \mathrm{Anc}(u,z) \wedge \mathrm{Anc}(x,y) \\ & \wedge \mathrm{Anc}(y,z)).\end{aligned}$$

Then, assuming Th, this definition is equivalent to the original one.

However, one can easily see that H is not optimal with regard to either $\prec_1$, or $\leq$.

In conclusion, we see that although we obtain formulae with good predictive power, they are not, in general, optimal with regard to either $\prec_1$, or $\leq$. This accords with our experience in the case of O's and X's.