Chapter 3 The mathematics of generalisation

1. Preliminaries

1.1 Notation

Our notation is that of Robinson (1965), with some additions. A word is a literal or a term. The symbols V, V_1, W,.... are used as meta-linguistic variables ranging over words. The symbol ϕ is used as a meta-linguistic variable ranging over predicate symbols and function symbols. Var(V) is the set of variables occurring in V. Suppose that some property can be shown to hold for variables and constants and that wherever it holds for terms $t_1,\ldots,t_n$ it holds for $\phi(t_1,\ldots,t_n)$. The property can then be seen to hold for all words. This method of proof is called induction on words. A translation is a substitution of the form, $\tau = \{y_1/x_1,\ldots,y_n/x_n\}$ where the y_i are all distinct. The inverse of τ is $\tau^{-1} = \{x_1/y_1,\ldots,x_n/y_n\}$. Note that $(\tau^{-1})^{-1} = \tau$ and if $Var(V) \subseteq \{x_i \mid i=1,n\}$, then $V\tau\tau^{-1} = V$. The Greek letters τ, ξ and η are reserved as meta-linguistic variables ranging over translations. The translation, τ, standardises W and V apart iff $W\tau$ and V have no common variables. Similarly τ standardises the clauses C and D apart iff $C\tau$ and D have no common variables.

The words W and V are alphabetic variants iff there is a translation τ such that $W\tau = V$. Alphabetic variance is an equivalence relation. Similarly, C and D are alphabetic variants iff there is a translation τ such that $C\tau = D$. Again, this defines an equivalence

relation.

We denote sequences of integers, perhaps empty, by the symbols, $I, J, \ldots$.

The term t is in the Ith place in the word W iff:

when $I = \langle\rangle$, $t = W$ or else

when $I = \langle i_1, \ldots, i_n \rangle$ $(n>0)$ then W has the form $\phi(t_1, \ldots, t_m)$ $(m>0)$ and $i_1 \leq m$ and t is in the $\langle i_2, \ldots, i_n \rangle$th place in t_{i_1}.

For example, x is in the $\langle\rangle$th place in x, the $\langle 2 \rangle$th place in $g(y,x)$ and the $\langle 3,2 \rangle$th place in $P(a,b,g(y,x))$.

Note that a term t is never in the $\langle\rangle$th place in a literal, L.

1.2 Generalisation

We say that $W \leq V$ (read W generalises V) iff there is a σ such that $W\sigma = V$. In the literature on automatic theorem proving, $W \leq V$ is usually read as "V is an instance of W".

Example $P(x,x,f(g(y))) \leq P(k(a()), k(a()), f(g(x)))$

We can take $\sigma = \{k(a())/x, x/y\}$

If $W \leq V$, there is a unique minimal σ_0 such that $W\sigma_0 = V$. For let σ be such that $W\sigma = V$ and let $\sigma_0 = \{t/x \mid t/x \in \sigma \text{ and } x \text{ appears in } W\}$. Then $W\sigma_0 = V$.

To see that σ_0 is minimal suppose $W\mu = V$. Let $t/x \in \sigma_0$.

It follows that x appears in W. A simple induction on words shows that for any W' if W' σ_0 = W'μ and x appears in W' then x σ_0 = xμ. Applying this to W, we see that t/x $\in \mu$. Hence $\sigma_0 \subseteq \mu$. Uniqueness is evident.

We say that C$\leq$D (read C generalises D) iff there is a σ so that C$\sigma \subseteq$ D. In the literature on automatic theorem proving, C$\leq$D is usually read as "C subsumes D". If we identify literals, for the moment, with their unit sets, $\leq$ on literals is just the restriction of $\leq$ on clauses to literals. Robinson (1965) shows that subsumption is decidable. It follows that generalisation on literals and on words are as well.

Generalisation on clauses is a quasi-ordering. It is reflexive as C$\varepsilon \subseteq$ C. It is transitive, for suppose that C$\sigma \subseteq$ D and D$\mu \subseteq$ E. Then C$\sigma\mu \subseteq$ D$\mu \subseteq$ E, as required. It follows that generalisation on literals is also a quasi-ordering. It is easy to see, then, that generalisation on words is a quasi-ordering.

One can show, much as above, that if C$\leq$D, there is a unique finite set $\{ \sigma_0, \sigma_1, \ldots, \sigma_n \}$ of distinct substitutions such that C $\sigma_i \subseteq$ D$(0 \leq i \leq n)$, and for all different i and j $\sigma_i \not\leq \sigma_j$, and if C$\mu \subseteq$ D then $\mu \geq \sigma_i$ for some i.

The algorithm for deciding subsumption is easily extended to one which will generate this set.

Example $\{P(x), P(f())\} \leq \{P(f())\}$. We can take $\sigma = \{f()/x\}$.

We say that $H_1 \leq H_2$ (read H_1 generalises H_2) iff for every D in H_2, there is a C in H_1 such that $C \leq D$.

Example $\{\{Q(h())\}, \{P(x), P(f())\}, \{P(x), Q(x)\}\} \leq \{\{P(f())\}, \{P(g()), Q(g())\}\}$

Evidently, this is a decidable relation.

1.3 Relative generalisation

We say that the literal **L** generalises **M** relative to Th iff there is a σ so that $\vdash_{Th} \mathbf{L}\sigma \equiv \mathbf{M}$. This is written as $\mathbf{L} \leq \mathbf{M}$ (Th). For the terms t and u, relative generalisation is defined by: $t \leq u$ (Th) iff there is a σ such that $\vdash_{Th} t\sigma = u$. When Th is empty, this reduces to the previous definition of generalisation between words. For $\vdash \mathbf{L}\sigma \equiv \mathbf{M}$ holds iff $\mathbf{L}\sigma = \mathbf{M}$ and $\vdash t\sigma = u$ holds iff $t\sigma = u$.

For clauses relative generalisation may be defined by: $C \leq D$ (Th) iff there is an **E** such that $\vdash_{Th} \mathbf{E} \equiv D$ and $C \leq \mathbf{E}$. An equivalent definition is: $C \leq D$ (Th) iff there is a σ so that $\vdash_{Th} C\sigma \rightarrow D$. Suppose $C \leq \mathbf{E}$ and $\vdash_{Th} \mathbf{E} \equiv D$. There is a σ such that $C\sigma \subseteq \mathbf{E}$. Then $\vdash C\sigma \rightarrow \mathbf{E}$ and so $\vdash_{Th} C\sigma \rightarrow D$. Conversely, suppose $\vdash_{Th} C\sigma \rightarrow D$. Then $\vdash_{Th} C\sigma \vee D \equiv D$, and $C \leq C\sigma \cup D$. We can, therefore, take $\mathbf{E} = C\sigma \cup D$. So the definitions are equivalent. When Th is empty and D is not a tautology, the definition reduces to the previous definition of generalisation between clauses, as was seen in chapter 2. The relation of "equivalence, provable from Th" is a congruence for relative generalisation. Suppose that $\vdash_{Th} C \equiv C'$ and $C \leq D$ (Th). There is a

σ so that $\vdash_{Th} C\sigma \rightarrow D$ and further, $\vdash_{Th} C\sigma \equiv C'\sigma$. Consequently $\vdash_{Th} C'\sigma \rightarrow D$ and $C' \leq D$ (Th). Suppose, conversely, $\vdash_{Th} D \equiv D'$ and $C \leq D$ (Th). There is an **E** so that $C \leq E$ and $\vdash_{Th} E \equiv D$. Then, $\vdash_{Th} E \equiv D'$ and $C \leq D'$.

Relative generalisation is a quasi-ordering. Since $\vdash_{Th} C\epsilon \rightarrow C$, $C \leq C$ (Th). Suppose $C \leq D \leq E$ (Th). There are σ, μ so that $\vdash_{Th} C\sigma \rightarrow D$ and $\vdash_{Th} D\mu \rightarrow E$. Then, as $(C\sigma \rightarrow D)\mu$ is $C\sigma\mu \rightarrow D\mu$, $\vdash_{Th} C\sigma\mu \rightarrow D\mu$ and $\vdash_{Th} C\sigma\mu \rightarrow E$. So $C \leq E$ (Th). It follows that relative generalisation on words is a quasi-ordering.

Define equivalence, relative to Th, by: $C \sim D$ (Th) iff $C \leq D$ (Th) and $D \leq C$ (Th). Since relative generalisation is a quasi-ordering, relative equivalence is an equivalence relation. If $\vdash_{Th} D \equiv D'$, then $D \sim D'$ (Th). The converse does not hold, as will be seen. We also define equivalence by $C \sim D$ iff $C \leq D$ and $D \leq C$. As generalisation is a quasi-ordering, equivalence is an equivalence relation. If $C \sim D$ then $C \sim D$ ($\emptyset$). On the other hand if $C \sim D$ ($\emptyset$) then either C and D are both tautologies or else $C \sim D$. The following lemma is well-known (Robinson, 1965), and so no proof is given.

Lemma 1 $U \sim W$ iff they are alphabetic variants.

We see that $P(x) \sim P(y)$, although it is not the case that $\vdash P(x) \equiv P(y)$. A characterisation of equivalence of clauses will be given later.

There is an interesting reformulation of the definition of relative

generalisation. It is shown that $C \leq D$ (Th) iff D is a tautology or else can be obtained from $\{C\} \cup$ Th by means of a special sort of derivation.

A derivation in which binary resolution is the sole deduction rule, is a C-derivation iff no two descendants of an occurrence of C at a tip are resolved together. (We assume a knowledge of some standard formulation of derivation trees, such as that of Andrews (1968)).

Let Th' be the set of Skolemisations of members of Th. We assume that none of the Skolem function symbols in Th' occur in C or D. Notice that Th' is a conservative extension of Th. That is, in this case, if a formula A does not contain any of the Skolem constants in Th' then $\vdash_{Th} A$ iff $\vdash_{Th'} A$. It follows that for any C and D, $C \leq D$ (Th) iff $C \leq D$ (Th').

There is one other useful fact. If $\vdash_{Th} A$ then $C \leq D$ (Th) iff $C \leq D$ (Th $\cup \{A\}$), for any C, D, Th and A. We define $\mathcal{R}(E,D) = \{C \mid C$ is a resolvent of E and D$\}$.

Lemma 2 If $D_1 \leq D$ (Th) and $D_1 \in \mathcal{R}(E,D_2)$ then $D_2 \leq D$ (Th $\cup \{\forall E\}$).

Proof As $D_1 \in \mathcal{R}(E,D_2)$ we can write E as $E' \cup E''$ and D_2 as $D_2' \cup D_2''$ and find a σ and a μ such that $E''\sigma$ and $D_2''\mu$ are unit sets containing complementary literals and $D_1 = E'\sigma \cup D_2'\mu$.

As $D_1 \leq D$ (Th) there is a λ such that $\vdash_{Th} D_1\lambda \rightarrow D$.

Now $\vdash E\sigma \wedge D_2\mu \rightarrow D_1$. Therefore, $\vdash E\sigma\lambda \wedge D_2\mu\lambda \rightarrow D_1\lambda$.

It follows that $\vdash_{Th} E\sigma\lambda \rightarrow (D_2/\mu\lambda \rightarrow D)$ and so $\vdash_{Th \cup \{\forall E\}} D_2/\mu\lambda \rightarrow D$. As this means that $D_2 \leq D\ (Th \cup \{\forall E\})$, the proof is finished.

Theorem 1 $C \leq D\ (Th)$ iff D is a tautology or there is a C-derivation from $Th' \cup \{C\}$ of a clause D' which subsumes D.

Proof Sufficiency If D is a tautology then $\vdash_{Th} C\varepsilon \rightarrow D$ and so $C \leq D\ (Th)$.

In any C-derivation there are either no occurrences of C on a tip or else there is exactly one occurrence. If there are none in the derivation given in the hypothesis then $\vdash_{Th'} D$ and so $\vdash_{Th} D$ as Th' is a conservative extension of Th and no Skolem function symbol of Th' occurs in D. Therefore $\vdash_{Th} C\varepsilon \rightarrow D$ which implies that $C \leq D\ (Th)$.

Suppose there is exactly one occurrence of C at a tip of the derivation given by the hypothesis. There must then be clauses $E_i\ (i=1,n;\ n \geq 0)$ derivable from Th' such that

$$D' \in \mathcal{R}(E_n, (\mathcal{R}(E_{n-1}, \ldots, \mathcal{R}(E_1, C)\ldots))).$$

As $D' \leq D$, n successive applications of lemma 2 show that $C \leq D\ (Th' \cup \{\forall E_i \mid i=1,n\})$ As $\vdash_{Th'} E_i$ for all i, we see that $C \leq D\ (Th')$ and conclude that $C \leq D\ (Th)$.

Necessity Suppose that $C \leq D\ (Th)$. For some σ, $\vdash_{Th} C\sigma \rightarrow D$. Let $C = \{L_i \mid i=1,n\}$. Then $\vdash_{Th} \bar{L}_i\sigma \vee D$ for each L_i. Let Th' be the set of Skolemisations of members of Th. None of the Skolem

function symbols should occur in $C\sigma$ -> D. By the subsumption theorem (Lee 1967, Kowalski 1970) for each L_i either $\bar{L}_i\sigma \vee D$ is a tautology or else there is a derivation of a clause D_i from Th', which subsumes $\{\bar{L}_i\sigma\} \cup D$. We may assume, without loss of generality, that the first alternative holds for $L_1,\ldots,L_m$ and the second for $L_{m+1},\ldots,L_n$.

If D is a tautology, we are finished. If m=n then $C\sigma = \{L_i\sigma \mid i=1,m\} \subseteq D$ and we are also finished. Finally, suppose that m<n.

Then we may express D as $\{L_i\sigma \mid i=1,m\} \cup D$.

Evidently there is a clause in

$$\mathcal{R}(\{\bar{L}_n\sigma\}, \mathcal{R}(\ldots \mathcal{R}(\{\bar{L}_{m+1}\sigma\}, C\sigma)\ldots))$$

which is a subset of $\{L_i\sigma \mid i=1,m\}$.

Consequently there is a clause, D', in

$$\mathcal{R}(\{\bar{L}_n\sigma\} \cup D, \mathcal{R}(\ldots \mathcal{R}(\{\bar{L}_{m+1}\sigma\} \cup D), C\sigma)\ldots))$$

which is a subset of $D \cup \{L_i\sigma \mid i=1,m\} = D$.

Since C subsumes $C\sigma$ and D_i subsumes $\{\bar{L}_i\sigma\} \cup D$ for $m<i\leq n$, it follows from the contraction theorem (Kowalski, 1970) that there is a C-derivation of a clause D'' from $\{D_i \mid m<i\leq n\} \cup \{C\}$ which subsumes D', and therefore D. As the D_i are derived, in their turn, from Th' we see that there is a C-derivation from Th' $\cup$ {C} of a clause D'' which subsumes D.

This concludes the proof.

One can obtain other versions of theorem 1 by using any other resolution principle for which the subsumption theorem holds. For example, one can use M-clash resolution (Kowalski, 1970).

2. Generalisation theory of literals

In this section we establish the theory of the relation, $\leq$, on literals. For technical convenience, the results are often derived for words. The corresponding result for literals is then an immediate specialization.

2.1 Least general generalisations of literals

A least general generalisation of some words is a generalisation which is less general than any other such generalisation. For example a least general generalisation of the literals Black(crow1), Black(crow2) is Black(x). One may, roughly, view this as inducing "Everything is black" from "This crow is black" and "That crow is black". The evident absurdity of such an induction was one of the reasons for considering clauses rather than literals.

Less trivially, we shall show later that a least general generalisation of P(f(a(),g(y)),x,g(y)) and P(h(a(),g(x)),x,g(x)) is P(y,x,g(z)).

Let K be a set of words. We say that W is a least general generalisation of K, abbreviated by W is a l.g.g. of K, iff:

1 For every V in K, $W \leq V$

2 If for every V in K, $W_1 \leq V$, then $W_1 \leq W$.

It follows from requirement 2 and lemma 1, that any two least general generalisations of K are alphabetic variants.

Similarly, one can give a definition relativised to Th. We say that W is a least general generalisation of a set of literals, K relative to Th (abbreviated by W is a l.g.g. of K relative to Th), iff:

1 For every M in K, $W \leq M$ (Th)

2 If, for every M in K, $W_1 \leq M$ (Th), then $W_1 \leq L$ (Th).

Two words are compatible iff they are both terms or have the same predicate letter and sign.

We can define also the least generalisation as a product in the following category. The objects are the words and σ is a morphism from V to W iff $V\sigma = W$ and σ acts as the identity, ϵ, on variables not in V. There is at most one morphism from V to W. If σ is a morphism from V to W, and μ is one from W to U, then their categorical composition is the unique morphism from W to U. It should be noted that the categorical composition is not the same as the standard composition of substitutions defined in Robinson (1965). For example, if W is x, V is f(y), and U is f(g(z)), then σ is {f(y)/x} and μ is {g(z)/y}. The categorical composition of σ and μ is {f(g(z))/x}, but their standard composition is {f(g(z))/x, g(z)/y}.

V is the least generalisation of $\{W_1, W_2\}$ iff it is a product of W_1 and W_2 in the above category. The category will be used mainly for expository purposes.

Theorem 1

Every non-empty, finite set of words has a least general

generalisation iff any two words in the set are compatible.

Let W_1, W_2 be any two compatible words. The following algorithm terminates at stage 3, and the assertion made there is then correct.

1. Set V_i to $W_i (i=1,2)$. Set θ_i to $\varepsilon (i=1,2)$. ε is the empty substitution.

2. Try to find terms t_1, t_2 which have the same place in V_1, V_2 respectively and such that $t_1 \neq t_2$ and either t_1 and t_2 begin with different function letters or else at least one of them is a variable.

3. If there are no such t_1, t_2 then halt. V_1 is a least general generalisation of $\{W_1, W_2\}$ and $V_1 = V_2$, $V_i \theta_i = W_i (i=1,2)$.

4. Choose a variable x distinct from any in V_1 or V_2 and wherever t_1 and t_2 occur in the same place in V_1 and V_2, replace each by x.

5. Change θ_i to $\{t_i/x\}\ \theta_i (i=1,2)$.

6. Go to 2.

Example. We will use the algorithm to find a least general generalisation of $\{P(f(a(),g(y)),x,g(y)), P(h(a(),g(x)),x,g(x))\}$.

Initially (at stage 1),

$V_1 = P(f(a(),g(y)),x,g(y))$

$V_2 = P(h(a(),g(x)),x,g(x))$,

and $\theta_1 = \theta_2 = \varepsilon$.

We take $t_1=y$, $t_2=x$ and z as the new variable at stage 2. Then after stage 4,

$$V_1=P(f(a(),g(z)),x,g(z))$$
$$V_2=P(h(a(),g(z)),x,g(z))$$

and after stage 5,

$$\theta_1=\{y/z\},\quad \theta_2=\{x/z\}.$$

Next, we take $t_1=f(a(),g(z)),t_2=h(a(),g(z))$ and y as the new variable at stage 2. After stages 4 and 5,

$$V_1=P(y,x,g(z))=V_2$$
$$\theta_1=\{f(a(),g(z))/y\}\{y/z\}$$
$$=\{f(a(),g(y))/y,y/z\}$$
$$\theta_2=\{h(a(),g(z))/y\}\{x/z\}$$
$$=\{h(a(),g(x))/y,x/z\}.$$

The algorithm then halts at stage 3 with $P(y,x,g(z))$ as the least general generalisation.

Proof Evidently the compatibility condition is necessary. Let $\{W_1,\ldots,W_n\}$ be a finite compatible set of words. If n=1, then the theorem is trivial. Suppose that the algorithm works and that $\inf\{V,W\}$ is the result of applying it to V and W. Then it is easy to see that

$$\inf\{W_1,\inf\{W_2,\ldots,\inf\{W_{n-1},W_n\}\ldots\}\}$$

is a least general generalisation of the set. Hence we need only show that the algorithm works.

The rest of the proof proceeds as follows. In order to avoid a constant repetition of the conditions on t_1, t_2 given in stage 2, we say that t_1 and t_2 are replaceable in V_1 and V_2 iff they fulfil the conditions of stage 2. We also denote by V_1', V_2' the result of replacing t_1 and t_2 in V_1, V_2 by x in the way described in stage 4. To show that the algorithm halts and that when it does $V_1=V_2$, we define a difference function by difference(V_1,V_2) = number of members of the set $\{I \mid$ if t_1, t_2 are both in the Ith place in V_1, V_2 respectively then they are replaceable in V_1 and $V_2\}$. Lemma 1.2 below then shows that every time a pair of replaceable terms is replaced the difference drops. Consequently by lemma 1.1, below it will eventually become zero and when it does, lemma 1.1, below shows that we must have $V_1=V_2$ and the algorithm will then halt. We still have to show that the replacements take us in the correct direction. First of all, $V_i' \leq V_i$ since by lemma 1.3, below, $V_i'\{t_i/x\} = V_i$. It is also immediate from this that when the algorithm halts, $V_i' \theta_i = W_i$. Now suppose that W is any lower bound of $\{W_1,W_2\}$. Then a lower bound V is a product of W_1, W_2 if the diagram of figure 1 can always be filled in along the dotted line, so that it becomes commutative in a unique way.

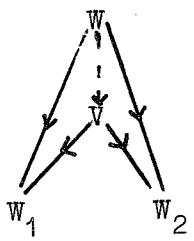


Figure 1

The category is the one defined above. In it there is either one or no morphisms between any two objects and hence it is not necessary in figure 1 to name the morphisms. Indeed, if a diagram can be filled in at all, it can be filled in commutatively and uniquely.

We show in lemma 1.4 below that the diagram on figure 2 can be filled in commutatively.

Thus every time a replacement is made, the V_i' are greater than any lower bound of W_1, W_2. Consequently when the algorithm halts, we have a product. We now give the statements and proofs of the lemmas.

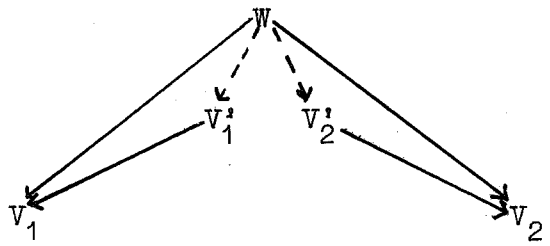


Figure 2

Lemma 1.1 If V_1 and V_2 are distinct compatible words, then there are t_1, t_2 which are replaceable in them.

Proof By induction on words on V_1. If one of V_1, V_2 is a constant or a variable, or if they begin with different function symbols, then V_1, V_2 will do for t_1, t_2 respectively.

If V_1 is $\phi(t_1^1,\ldots,t_n^1)$ and V_2 is $\phi(t_1^2,\ldots,t_n^2)$, then for some i, $t_i^1 \neq t_i^2$ and by the induction hypothesis, applied to t_i^1, there are u_1, u_2 which are replaceable in t_i^1, t_i^2 which have the same place in V_1, V_2

respectively, and so u_1 and u_2 are also replaceable in V_1, V_2.

Lemma 1.2 If V_1, V_2 are distinct compatible words, then $\text{Difference}(V_1',V_2') < \text{Difference}(V_1,V_2)$.

Proof By induction on words on V_1. If one of V_1 or V_2 is a variable or a constant then $t_1=V_1$, $t_2=V_2$ and $V_1'=V_2'=x$, so $0=\text{Difference}(V_1',V_2') < 1 = \text{Difference}(V_1,V_2)$.

If V_1 is $f(v_1,\ldots,v_n)$ and V_2 is $g(u_1,\ldots,u_m)$ where $f\neq g$, then if $t_i=V_i(i=1,2)$, $0=\text{Difference}(V_1',V_2') < \text{Difference}(V_1,V_2)$, by lemma 1.1; otherwise

$$\begin{aligned}\text{Difference}(V_1',V_2') &= 1 + \sum_{i=1,\min(m,n)} \text{Difference}(v_i',u_i') \\ &< 1 + \sum_{i=1,\min(m,n)} \text{Difference}(v_i,u_i) \\ &\qquad \text{(by induction hypothesis, since } m,\ n\neq 0) \\ &= \text{Difference}(V_1,V_2).\end{aligned}$$

In the remaining case where V_1 and V_2 both have the form $\phi(t_1,\ldots,t_n)$, a similar but less complicated argument applies.

Lemma 1.3 $V_i'\{t_i/x\} = V_i(i=1,2)$.

Proof Since V_i' is obtained from V_i by replacing some occurrences of t_i in V_i by x, and since x does not occur in V_i, substituting t_i for x in V_i' will produce $V_i(i=1,2)$.

Lemma 1.4 If V_1, V_2 are distinct compatible words and $V\sigma_i=V_i(i=1,2)$, then there are σ_1', σ_2' so that $V\sigma_i'=V_i'(i=1,2)$.

Proof It is convenient to denote by $f_i(u_1,u_2,t_1,t_2)$ the result of applying the operation of 4 to u_i (with u_1 and u_2 standing for V_1 and V_2) for i=1,2.

Let the variables which occur in V be $y_1,\ldots,y_m$ and suppose that $y_j\ \sigma_i=u_i^j$ (i=1,2; j=1,m) and choose σ_i' so that $y_j\ \sigma_i'=f_i(u_1^j,u_2^j,t_1,t_2)$ (i=1,2; j=1,m).

By lemma 1.3, $y_j\ \sigma_i=y_j\ \sigma_i'\{t_i/x\}$ (i=1,2; j=1,m).

We need only show, by induction on words with variables $y_1,\ldots,y_m$ applied to W that if W, W_1 and W_2 are such that $W\ \sigma_i = W_i$ (i=1,2) then $W\ \sigma_i' = f_i(W_1,W_2,t_1,t_2) = W_i'$, say (i=1,2).

Suppose W is a constant; then $W=W_1=W_2$ and the result is trivial.

Suppose W is the variable y_j. Then $u_i^j=W_i$ (i=1,2) and so:

$$\begin{aligned} W\ \sigma_i' &= y_j\ \sigma_i' = f_i(u_1^j,u_2^j,t_1,t_2) \\ &= f_i(W_1,W_2,t_1,t_2) \\ &= W_i' \text{ (i=1,2).} \end{aligned}$$

Suppose W is $\phi(u_1,\ldots,u_n)$. Then W_i has the form $\phi(w_1^i,\ldots,w_n^i)$ and so W_i' is $\phi(w_1^{i'},\ldots,w_n^{i'})$, say, where $w_l^{i'}=f_i(w_l^1,w_l^2,t_1,t_2)$, (i=1,2; l=1,n). (i=1,2).

Therefore, $W\ \sigma_i' = \phi(u_1\ \sigma_i',\ldots,u_n\ \sigma_i')$

$= \phi(w_1^{i'},\ldots,w_n^{i'})$ (by induction hypothesis)

$= W_i'$.

This concludes the induction and the proof.

There is a more efficient version of the algorithm given in the statement of Theorem 1. This involves fewer passes through W_1 and W_2. It is a slight generalisation of the algorithm given by Reynolds (1970).

1 Set V_i to W_i $(i=1,2)$. Set θ_i to ε, the empty substitution $(i=1,2)$.

2 Try to find terms t_1, t_2 which have the same place in V_1 and V_2 respectively and such that $t_1 \neq t_2$ and either t_1 and t_2 begin with different function letters or else at least one of them is a variable.

3 If there are no such t_1, t_2 then halt. V_1 is a l.g.g. of $\{W_1, W_2\}$, $V_1 = V_2$ and $V_i \theta_i = W_i$ $(i=1,2)$.

4 Find a variable x such that $x\theta_i = t_i$ $(i=1,2)$. If there is no such variable, let x be a variable distinct from any in V_1 or V_2.

5 Find an occurrence of t_1 in V_1 and an occurrence, in the same place, of t_2 in V_2 and replace the occurrence of t_1, and the occurrence of t_2 by one of x.

6 Change θ_i to $\{t_i/x\}\theta_i$ $(i=1,2)$.

7 Go to 2.

We shall adopt the convention that $\inf\{W_1, W_2\}$ is the l.g.g. of W_1 and W_2 produced by applying some definite version of one of these algorithms to $\{W_1, W_2\}$.

2.2 A technical lemma

The following lemma is useful in establishing the existence of l.g.g.'s of clauses.

Lemma 1 Let $K = \{W_i | i=1,n\}$ be a set of words with a l.g.g. W and let $\mu_i (i=1,n)$ be substitutions such that $W \mu_i = W_i (i=1,n)$.

1 If t occurs in W, then t is a l.g.g. of $\{t \mu_i | i=1,n\}$.

2 If x, y are variables occurring in W and $x \mu_i = y \mu_i (i=1,n)$, then x=y.

Proof We can assume, without loss of generality, that the μ_i are the minimal substitutions such that $W \mu_i = W_i$.

1 Evidently, t is a generalisation of $\{t \mu_i | i=1,n\}$. Let u be any other and suppose that $u \lambda_i = t \mu_i (i=1,n)$. Let $\tau = \{y_1/x_1, \ldots, y_m/x_m\}$ be a translation such that $x_1, \ldots, x_m$ are the variable symbols of u, and $u\tau$ and W have no common variables. Let W' be W, but with every occurrence of t replaced by one of $u\tau$. Then $\sigma_i = \{x_1 \lambda_i/y_1, \ldots, x_m \lambda_i/y_m\} \cup \mu_i$ is defined, using the minimality of $\mu_i (i=1,n)$. From the construction of σ_i, $W' \sigma_i = W_i (i=1,n)$. Hence there is a ν so that $W' \nu = W$, as W is a l.g.g. of $\{W_i | i=1,n\}$. Hence $u (\tau \nu) = (u \tau) \nu = t$. So t is a l.g.g. of $\{t \mu_i | i=1,n\}$.

2 Suppose that $y \neq x$. Let $W' = W\{y/x\}$.
Then W', W are not alphabetic variants, but $W \leq W'$. Let $W = W_{x,y,y_3,\ldots,y_m}[x,y,y_3,\ldots,y_m]$, where $x,y,y_3,\ldots,y_m$ are the variables occurring in W. We have:

$$W_i = W\,\mu_i = W_{x,y,y_3,\ldots,y_m}[x\,\mu_i, y\,\mu_i, y_3\,\mu_i, \ldots, y_m\,\mu_i]$$

$$= W_{x,y,y_3,\ldots,y_m}[y\,\mu_i, y\,\mu_i, y_3\,\mu_i, \ldots, y_m\,\mu_i]$$

(by hypothesis)

$$= W_{x,y,y_3,\ldots,y_m}[y,y,y_3,\ldots,y_m]\,\mu_i$$

$$= W'\,\mu_i \text{ (by construction) } (i=1,n).$$

This contradicts the fact that W is a least generalisation of $\{W_i \mid i=1,n\}$. Hence y=x.

This completes the proof.

2.3 Lattice properties of literals

It is possible to define the dual of the l.g.g. of two words.

The word, U, is a most general instance (m.g.i.) of V and W iff:

1 $V \leq U$ and $W \leq U$

2 If $V \leq U'$ and $W \leq U'$ then $U \leq U'$.

Note that, by lemma 1.1, any two m.g.i.'s of V and W are alphabetic variants. The relevant theorem here, is the Unification Theorem of Robinson (1965). A version suitable for our present purposes is given in the following lemma.

Lemma 1 (Unification Theorem) If $V\theta = W\theta$ for some substitution θ, there is a most general unifier, (m.g.u.) μ, with the properties:

1 $V\mu = W\mu$

2 For any σ, if $V\sigma = W\sigma$, then there is a λ such that $\sigma = \mu\lambda$.

Robinson gives an algorithm for calculating the m.g.u.

In terms of the category described in section 2.1, there is a close correlation between the co-product and the m.g.u.

Lemma 2 Let $W\tau$ and V have no variables in common. Then $W\tau$ and V are unifiable iff W and V have a co-product. If μ is an m.g.u. of W and V, and U is a co-product, then $U \sim W\tau\mu$.

We leave the rather easy proof to the reader.

In terms of m.g.i.'s, the appropriate remark, again easily proved, is:

Lemma 3 Let $W\tau$ and V have no variables in common. Then $W\tau$ and V are unifiable iff W and V have an m.g.i. If μ is an m.g.u. and U an m.g.i. of W and V, then $W\tau\mu \sim U$.

We can now define a lattice following Reynolds (1970). First the ordering $\leq$ on words is extended by adding special top and bottom elements, $\mathcal{A}$ and $\mathcal{B}$ respectively. So we consider the set $\text{Words}^+ = \{W \mid W \text{ is a word}\} \cup \{\mathcal{A}, \mathcal{B}\}$.

More formally, $\leq$ is defined on Words^+ to be:

$\{\langle W,V\rangle \mid W \leq V\} \cup \{\langle \mathcal{A}, W\rangle \mid W \text{ is a word}\} \cup \{\langle W, \mathcal{B}\rangle \mid W \text{ is a word}\} \cup \{\langle \mathcal{A}, \mathcal{B}\rangle\}$.

For the rest of us the section, we will use, W, W', W_0 etc. to stand for an arbitrary member of Words$^+$.

The following facts are then obvious.

The relation $\leq$ on Words$^+$ is a quasi-ordering. The extension of to Words$^+$, defined by:

$W \sim W'$ iff $W \leq W'$ and $W' \leq W$, is an equivalence relation. If we consider the set of equivalence [W] classes of members of Words$^+$:

$$[W] = \{W' \mid W' \sim W\},$$

then the induced ordering, $\leq$, on equivalence classes given by $[W] \leq [W']$ iff $W \leq W'$ is a well-defined partial ordering.

In fact it is a lattice.

Let us define $\sqcap$ and $\sqcup$ as operations on Words$^+$ by:

1 a $[\mathfrak{A}] \sqcap [W] = [W] \sqcap [\mathfrak{A}] = [\mathfrak{A}]$

b If V and V' have an l.g.g. U then $[V] \sqcap [V'] = [U]$, otherwise $[V] \sqcap [V'] = [\mathfrak{A}]$.

c $[\mathfrak{B}] \sqcap [W] = [W] \sqcap [\mathfrak{B}] = [W]$.

2 a $[\mathfrak{B}] \sqcup [W] = [W] \sqcup [\mathfrak{B}] = [\mathfrak{B}]$.

b If V and V' have an m.g.i. U then $[V] \sqcup [V'] = [U]$, otherwise $[V] \sqcup [V'] = [\mathfrak{B}]$.

c $[\mathfrak{A}] \sqcup [W] = [W] \sqcup [\mathfrak{A}] = [W]$.

Reynolds shows that Words$^+$ is indeed a lattice under the (well-defined) operations $\sqcap$ and $\sqcup$. We will not consider the relativised lattice obtained from the relativised generalisation relation. The lattice is non-modular since:

$$[P(f(x),y)] \leq [P(f(x),f(y))]$$

but:

$$\begin{aligned}
&([P(x,x)] \sqcap [P(f(x),f(y))]) \sqcup [P(f(x),y)] \\
&\quad = [P(x,y)] \sqcup [P(f(x),y)] \\
&\quad = [P(f(x),y)] \\
&\quad \neq [P(f(x),f(y))] \\
&\quad = [P(f(x),f(x))] \sqcap [P(f(x),f(y))] \\
&\quad = ([P(x,x)] \sqcup [P(f(x),y)]) \sqcap [P(f(x),f(y)],
\end{aligned}$$

contradicting the modular equality.

Let $[W]<[W']$ mean $[W]\leq[W']$, but $[W] \neq [W']$. Then $\{[W_i] \mid i\geq 0\}$ is an <u>infinite ascending chain</u> iff $W_i<W_{i+1}$ $(i\geq 0)$. <u>Infinite descending chains</u> are defined similarly. One sees immediately that $[P(f(x))]$, $[P(f(f(x)))]$,$[P(f(f(f(x))))]$..is an infinite strictly ascending chain. It follows easily from Reynolds' work that there are no infinite strictly descending chains and no infinite strictly ascending chains $\{[W_i] \mid i\geq 0\}$ such that every W_i contains no function symbols. On the other hand, the lattice is complete, that is any subset of Words$^+$ has a greatest common instance and a least common generalisation. Later,

we will contrast this situation with that of a lattice of equivalence classes of clauses which is even more irregular than Words^{+}.

3. Generalisation theory of clauses

The theory of the generalisation relation, $\leq$, defined on clauses is developed in this section.

3.1 Elementary properties

The relation $\leq$ is a quasi-ordering and $\sim$ is an equivalence relation. However, two equivalent clauses need not be alphabetic variants. For example, let $C = \{P(x), P(f())\}$ and $D = \{P(f())\}$.

As neither C nor D are tautologies, this gives an example, with $Th = \emptyset$, where $C \sim D$ (Th) but $C \neq D$.

We develop a slightly more complicated characterisation of equivalence. The clause C is reduced iff $D \subseteq C$, $D \sim C$ implies that $C = D$. In other words, C is reduced iff it is equivalent to no proper subset of itself.

A clause C is a reduction of a clause D iff it is a reduced subset of D which is equivalent to D.

Lemma 1 Suppose that $C\mu = C$. Then

1) For some integer, $n_0 \geq 1$, $L\mu^{n_0} = L$ for every literal L in C and $x\mu^{n_0} = x$ for every variable x in C.

2) The substitution μ maps distinct variables of C to distinct variables of C.

Proof 1) Define a mapping π : C -> C by $\pi(L) = L\mu$. Since $C\mu = C$, π is onto and therefore, as C is finite, π is a permutation. There is therefore an integer, $n_0 \geq 1$ such that $\pi^{n_0} = \iota$, the identity permutation. So if L is in C, $L\mu^{n_0} = \pi^{n_0}(L) = \iota(L) = L$. Let x be a variable in C. It must occur in some literal, L say, in C. As $L\mu^{n_0} = L$, it follows that $x\mu^{n_0} = x$. (Strictly speaking, this last step requires an easy proof by induction on words.)

2) This is an obvious consequence of 1).

In order to calculate reductions of clauses, a slight variant of the test for subsumption is useful (Robinson, 1965).

Lemma 2 Let C, D and E be clauses and let $\delta = \{a_1()/x_1, \ldots, a_n()/x_n\}$ where $x_1, \ldots, x_n$ are all the variables in E and the a_i are distinct constants. Then there is a σ such that $E\sigma \subseteq C$ and, for all L in D, $L\sigma = L$ iff $E\delta \leq C\delta$.

Proof Suppose there is a σ satisfying the conditions. Then $x_i\sigma = x_i$, for all i and so $\delta\sigma\delta = \sigma\delta$, for $x_i\,\delta\sigma\delta = a_i() = x_i\delta = x_i\sigma\delta$ and if $y \neq x_i$, for any i, then $y\delta\sigma\delta = y\sigma\delta$. Therefore, $E\sigma\delta = E\delta\sigma\delta \subseteq C\delta$ which shows that $E\delta \leq C\delta$.

Suppose that $E\delta \leq C\delta$. Then there is a substitution μ such that $E\delta\mu \subseteq C\delta$. As $E\delta$ contains no occurrence of an x_i, we may

assume, w.l.o.g., that $x_i \mu = x_i$ for all i. Therefore, if L is in E, $L\mu = L$. Further, as above, $\delta\mu\delta = \mu\delta$. Therefore, $E\mu\delta = E\,\delta\mu\delta \subseteq C\delta\delta = C\delta$. As δ maps distinct variables to distinct constants, we see that $E\mu \subseteq C$ which completes the proof.

The question of the existence of a substitution satisfying the conditions of the lemma can, therefore, be answered by using the standard test for subsumption.

Theorem 1 If $C \sim D$ and both C and D are reduced, then they are alphabetic variants. Every clause has a reduction. The following algorithm gives a reduction, E_1, of a given clause C.

1) Set E_1 to $\emptyset$ and E_2 to C.

2) If E_2 is empty, stop.

3) Choose a literal, L, in E_2.

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

5) Go to 2).

(To implement step 4, one finds a δ as described in lemma 2 and then tests whether $E_2\delta \leq (E_2 \setminus \{L\})\delta$. In doing this one can increase efficiency by noticing that if $E\,\delta\sigma \subseteq (E_2 \setminus \{L\})\delta$ then $L\,\delta\sigma$ is in $E_2\delta$. Therefore one need only calculate all the

minimal substitutions $\sigma_j (j=1,s)$ such that $L\,\delta\sigma_j$ is in $E_2\,\delta$, testing for each, in turn, whether $E_2\,\delta\,\sigma_j \leq (E_2 \setminus \{L\})\,\delta$.

Proof Suppose that $C \sim D$ and C and D are reduced. There are substitutions σ and μ such that $C\sigma \subseteq D$ and $D\mu \subseteq C$. Therefore $C\sigma\mu \subseteq D\mu \subseteq C$. As C is reduced, $C\sigma\mu = C$. Therefore, by lemma 1, $\sigma\mu$ maps distinct variables of C to distinct variables of C. Hence σ maps distinct variables of C to distinct variables of D. Now $D \leq C\sigma$ since $D\mu\sigma \subseteq C\sigma$ and therefore, as $C\sigma \subseteq D$ and D is reduced, $C\sigma = D$. Combining this with the fact that σ maps distinct variables of C to distinct variables of D, we see that C and D are alphabetic variants.

Next we show that any clause, C, has a reduction. Let $H = \{D \subseteq C \mid C \leq D\}$. Let C' be a minimal, with respect to $\subseteq$, member of H. We claim that C' is a reduction of C. Certainly $C' \subseteq C$ and $C' \sim C$. If C' is not reduced, it has a proper subset C'' which is reduced and is equivalent to C'. But C'' must be in H, and this contradicts the minimality of C.

The algorithm always terminates since the number of literals in E_2 always decreases by at least one every time round the loop.

Let E_1^f and E_2^f be the final values of E_1 and E_2 respectively. We show that $E_1^f \subseteq C$ and $C \leq E_1^f$. The first assertion is obvious.

To prove the second, suppose that E_1 and E_2 have the values E_1'

and E_2' at some stage when the execution reaches step 4, and the values E_1'' and E_2'' immediately after step 4. We show that $E_1' \cup E_2' \leq E_1'' \cup E_2''$.

Suppose a suitable σ exists. Then $E_2'' = E_2'\sigma \setminus E_1'$ and $E_1'' = E_1' = E_1'\sigma$. Therefore,

$$(E_1' \cup E_2')\sigma = E_1'\sigma \cup E_2'\sigma = E_1' \cup (E_2'\sigma \setminus E_1')$$
$$= E_1'' \cup E_2''.$$

If no suitable σ exists, $E_1' \cup E_2' = E_1'' \cup E_2''$ since $E_1'' = E_1' \cup \{L\}$ and $E_2'' = E_2' \setminus \{L\}$ for some L.

It follows easily that at every stage of the execution, $E_1 \cup E_2 \leq E_1^f \cup E_2^f = E_1^f$. Applying this to the initial values, $\emptyset$ and C of E_1 and E_2 we see that $C \leq E_1^f$ as required.

We show next that E_1^f is reduced, which will complete the proof. The proof is by induction with the hypothesis that there is at any stage a clause E which is a reduction of $E_1 \cup E_2$ and contains E_1.

This is true initially since $E_1 = \emptyset$ then and we have already shown that every clause has a reduction. Suppose E_1', E_2', E_1'' and E_2'' are as described above and that $E \supseteq E_1'$ is a reduction of $E_1' \cup E_2'$.

Suppose there is a suitable σ as described in step 4. Then $E\sigma \supseteq E_1'\sigma = E_1' = E_1''$. Further $E\sigma \subseteq (E_1' \cup E_2')\sigma = E_1'' \cup E_2'' \subseteq E_1' \cup E_2' \leq E \leq E\sigma$. Therefore $E\sigma \sim (E_1'' \cup E_2'') \sim E$. There is therefore a substitution μ such that $E\sigma\mu \subseteq E$. Therefore $E\sigma\mu \sim E$

and as $\mathbf{E}$ is reduced, $\mathbf{E}\sigma\mu = \mathbf{E}$. By lemma 1, $\sigma\mu$ maps distinct variables of $\mathbf{E}$ to distinct variables. So therefore does σ and so $\mathbf{E}$ and $\mathbf{E}\sigma$ are alphabetic variants. Therefore, $\mathbf{E}\sigma$ is reduced, equivalent to $\mathbf{E}_1' \cup \mathbf{E}_2'$ and contains $\mathbf{E}_1'$.

Suppose there is no such suitable σ. Suppose also that the literal $\mathbf{L}$ chosen in step 3 is not in $\mathbf{E}$. Then there is a μ such that $(\mathbf{E}_1' \cup \mathbf{E}_2')\mu \subseteq \mathbf{E} \subseteq (\mathbf{E}_1' \cup \mathbf{E}_2') \setminus \{\mathbf{L}\}$. As $\mathbf{E} \subseteq (\mathbf{E}_1' \cup \mathbf{E}_2')$, $\mathbf{E}\mu \subseteq (\mathbf{E}_1' \cup \mathbf{E}_2')\mu \subseteq \mathbf{E}$. Therefore, as $\mathbf{E}$ is reduced $\mathbf{E}\mu = \mathbf{E}$. By lemma 1, there is an integer $n_0 \geq 1$ such that $\mathbf{M}\mu^{n_0} = \mathbf{M}$ for every literal $\mathbf{M}$ in $\mathbf{E}$. Then μ^{n_0} is a suitable substitution in stage 4 as, if $\mathbf{M}$ is in $\mathbf{E}_1$, $\mathbf{M}\mu^{n_0} = \mathbf{M}$ as $\mathbf{E}_1 \subseteq \mathbf{E}$ and $(\mathbf{E}_1' \cup \mathbf{E}_2')\mu^{n_0} = (\mathbf{E}_1' \cup \mathbf{E}_2')\mu \cdot \mu^{n_0-1} \subseteq \mathbf{E}\mu^{n_0-1} = \mathbf{E} \subseteq (\mathbf{E}_1' \cup \mathbf{E}_2') \setminus \{\mathbf{L}\}$. This contradicts the assumption that $\mathbf{L}$ is not in $\mathbf{E}$. Therefore $\mathbf{L}$ is in $\mathbf{E}$ and so $\mathbf{E}_1'' = \mathbf{E}_1' \cup \{\mathbf{L}\} \subseteq \mathbf{E}$ and furthermore, as $\mathbf{E}_1'' \cup \mathbf{E}_2'' = \mathbf{E}_1' \cup \mathbf{E}_2'$, $\mathbf{E}$ is a reduction of $\mathbf{E}_1'' \cup \mathbf{E}_2''$ which concludes the inductive proof.

Applying the result to the final stage we see that there is a clause $\mathbf{E} \supseteq \mathbf{E}_1^f$ which is a reduction of $\mathbf{E}_1^f \cup \mathbf{E}_2^f = \mathbf{E}_1^f$. Therefore $\mathbf{E}_1^f \subseteq \mathbf{E} \subseteq \mathbf{E}_1^f$ and so $\mathbf{E}_1^f$ is reduced completing the proof.

Corollary 1 $\mathbf{C} \sim \mathbf{D}$ iff any two reductions of $\mathbf{C}$ and $\mathbf{D}$ respectively are alphabetic variants.

Proof Obvious from theorem 1.

It is convenient to develop a characterisation of equivalence

between sets of clauses at this point.

This relation is defined by:

$H \sim H'$ iff $H \leq H'$ and $H' \leq H$.

A set, H, of clauses is <u>reduced</u> iff $H' \subseteq H$ and $H' \sim H$ implies that $H' = H$.

A set, H', of clauses is a <u>reduction</u> of a set, H, of clauses iff H' is reduced, $H' \sim H$ and $H' \subseteq H$.

<u>Theorem 2</u> If $H' \sim H$ and H' and H are reduced then there is a unique bijection θ: H' -> H such that $\theta(C) \sim C$ for every clause, C, in H.

The following algorithm gives a reduction, H', of a given set of clauses H.

1) Set H' to H.

2) Stop if every clause in H' is marked.

3) Choose an unmarked clause, C, in H.

4) If $H' \setminus \{C\} \leq \{C\}$ then change H' to $H' \setminus \{C\}$. Otherwise mark C.

5) Go to 2).

<u>Proof</u> Choose θ: H' -> H so that $\theta(C) \leq C$ for every clause, C, in H'. This is possible as $H' \leq H$.

Similarly one can choose a mapping θ': H -> H' so that $\theta'(C) \leq C$ for every clause, C, in H.

Then, if C is in H', $\theta'(\theta(C)) \leq \theta(C) \leq C$. But H' is reduced and therefore $\theta'(\theta(C)) = C$. Similarly, if C is in H, $\theta(\theta'(C)) = C$. Hence θ is a bijection with inverse θ'. We see too that $C = \theta'(\theta(C)) \leq \theta(C) \leq C$. Therefore $\theta(C) \sim C$.

If θ'' is another bijection from H' to H such that $\theta''(C) \sim C$ for any clause C in H' then it must also have θ' as an inverse by a similar argument to that above. Therefore $\theta = \theta''$.

Since the number of unmarked literals in H' decreases by one every time the execution of the algorithm goes round the loop, the algorithm terminates.

Let H'_f be the final value of H'. It is evident that $H'_f \subseteq H$ and $H'_f \leq H$ and indeed that $H'_f \subseteq H'$ and $H'_f \leq H'$ for any value of H'. To complete the proof we need only show that H'_f is reduced.

If it is not we can find a marked clause C in H'_f such that $H'_f \setminus \{C\} \leq H'_f$. Let H'_1 be the value of H' just before C was marked. Then $H'_1 \setminus \{C\} \supseteq H'_f \setminus \{C\}$, as $H'_1 \supseteq H'_f$, and $H'_f \setminus \{C\} \leq H'_f \leq H'_1$. Therefore $H'_1 \setminus \{C\} \leq H'_1 \leq \{C\}$ which contradicts the fact that C is marked in H'_1. This concludes the proof.

<u>Corollary 2</u> $H_1 \sim H_2$ iff given two reductions H'_1 and H'_2 of H_1 and H_2 respectively there is a bijection, θ, from H'_1 to H'_2 such that

that $\theta(C) \sim C$ for any clause C in H_1.

Proof Obvious from theorem 2.

3.2 Least general generalisations of clauses

A least general generalisation of some clauses is a generalisation which is less general than any other such generalisation. For example, a least general generalisation of the clauses $\{\overline{\text{Crow}}(\text{crow1}), \text{Black}(\text{crow1})\}$ and $\{\overline{\text{Crow}}(\text{crow2}), \text{Black}(\text{crow2})\}$ is $\{\overline{\text{Crow}}(x), \text{Black}(x)\}$. This may be viewed as an induction from "This crow is black" and "That crow is black" to "All crows are black". This is a much more satisfying phenomenon than our induction of "Everything is black" when we were restricted to literals. We shall give a rather less trivial example later.

The operation of taking a least general generalisation followed by a reduction of the result will play a major role in algorithms for finding nicest explanatory hypotheses. Reductions are taken in order to make the output more perspicuous and in order to reduce demands on the internal storage space of the computer being used to implement such algorithms. Both these points will be illustrated later in this section.

Let H be a set of clauses. The clause, C, is a least general generalisation (which is abbreviated by l.g.g.) of H iff:

1 For every D in H, $C \leq D$.

2 If for every D in H, $C' \leq D$, then $C' \leq C$.

Any two l.g.g's of H are evidently equivalent.

In an analogous way, we say that C is a least general generalisation of H, relative to Th iff:

1 For every D in H, $C \leq D$ (Th).

2 If for every D in H, $C' \leq D$ (Th) then $C' \leq C$ (Th).

One sees that if C and C' are l.g.g.'s of H (relative to Th), then they are equivalent (relative to Th).

By temporarily introducing some auxiliary syntactic concepts, it is possible to give a short demonstration of the existence of least general generalisations of a finite set of clauses. We consider sequences of literals including the null sequence. A literal is identified with that one element sequence whose only member is the literal. The behaviour of sequences of literals is very similar to that of literals.

$\prod_{i=1}^{n} L_i$ is defined to be the sequence with n members whose ith member is L_i. Notice that if n=0, this is the null sequence. We extend the meaning of the $\prod$ operator, so that sequences themselves may be "multiplied", by:

$$\prod_{i=1}^{n} \left(\prod_{j=1}^{m(i)} L_{ij} \right) = L_{11} \ldots L_{1m(1)} \ldots L_{n1} \ldots L_{nm(n)}$$

The reader is left to provide proper formal definitions.

Powers of literals are defined by:

$$L^n = \prod_{i=1}^{n} L_i, \text{ where, for every } i,\ L_i = L.$$

Application of a substitution to a sequence is defined by:

$$(\prod_{i=1}^{n} L_i)\sigma = \prod_{i=1}^{n} (L_i\sigma).$$

The sequence $\prod_{i=1}^{n} L_i$ is a generalisation of $\prod_{j=1}^{m} M_j$ iff for some σ $(\prod_{i=1}^{n} L_i)\sigma = \prod_{j=1}^{m} M_j$. This is written symbolically as $\prod_{i=1}^{n} L_i \leq \prod_{j=1}^{m} M_j$. Notice that if this relationship does hold then n=m.

We leave the reader to provide a definition of a least general generalisation of a set of sequences of literals and to enlarge the definition of word, place and occurrence given in 3.1.1 to include and apply to sequences of literals. Notice that no word can appear in any place other than the <>th in the null sequence.

Two sequences $\prod_{i=1}^{n} L_i$ and $\prod_{j=1}^{m} M_j$ are compatible iff m=n and for every i between 1 and min(n,m) L_i and M_i are compatible (that is, have the same predicate letter and sign). The next lemma shows how to calculate l.g.g's of sequences and when this is possible.

Lemma 1 Every non-empty finite set of sequences of literals has a least general generalisation iff any two sequences in the set are compatible.

Let $\prod_{i=1}^{n} L_i$ and $\prod_{j=1}^{m} M_j$ be any two compatible sequences of literals.

The algorithm given in theorem 3.2.1.1 may be applied to them. It terminates at stage 3, and the assertion made there is then correct.

Proof We do not give the proof which is directly analogous to that of theorem 3.2.1.1.

Of course one could also use Reynolds' (1970) algorithm, given in section 3.2.1.

The next lemma relates l.g.g's of certain related sets of sequences.

Lemma 2 Let $\prod_{i=1}^{n} L_i$ be a l.g.g. of $\prod_{i=1}^{n} M_i$ and $\prod_{i=1}^{n} N_i$.

1) If π is a permutation of the numbers $1,\ldots,n$ then $\prod_{i=1}^{n} L_{\pi(i)}$ is a l.g.g. of $\prod_{i=1}^{n} M_{\pi(i)}$ and $\prod_{i=1}^{n} N_{\pi(i)}$.

2) Let n_i be positive integers $(1 \leq i \leq n)$. $\prod_{i=1}^{n} L_i^{n_i}$ is a l.g.g. of $\prod_{i=1}^{n} M_i^{n_i}$ and $\prod_{i=1}^{n} N_i^{n_i}$.

3) If $1 \leq n' < n$ then $\prod_{i=1}^{n'} L_i$ is a l.g.g. of $\prod_{i=1}^{n'} M_i$ and $\prod_{i=1}^{n'} N_i$.

4) Let π be a permutation of the numbers $1,\ldots,n$ and let n_i be integers $(1 \leq i \leq n)$. $\prod_{i=1}^{n} L_{\pi(i)}^{n_i}$ is a l.g.g. of $\prod_{i=1}^{n} M_{\pi(i)}^{n_i}$ and $\prod_{i=1}^{n} N_{\pi(i)}^{n_i}$.

Proof 1) and 2) are obvious. The proof of 3) is analogous to that of lemma 3.2.2.1. Part 4) follows immediately from the other three parts.

This concludes the necessary supply of lemmas on sequences of literals.

A set of literals $K = \{L_i \mid i=1,n\}$ is a <u>selection</u> from $H = \{C_i \mid i=1,n\}$ iff L_i is in C_i $(i=1,n)$ and any two literals in K are compatible. To see the motivation for this definition, suppose that $E \leq C_i$ for every i. Then there are σ_i such that $E\sigma_i \subseteq C_i$ for every i. Choose a literal L in E, if it is not empty. $\{L\sigma_i \mid i=1,n\}$ is a selection from H.

<u>Theorem 1</u> Every finite set, H, of clauses has a least general generalisation which is not $\emptyset$ iff H has a selection. Let C and D be two clauses with selections $\{M_i,N_i\}$, where $1\leq i\leq n$, so that C and D have at least one selection. Let $\prod_{i=1}^{n} L_i$ be a l.g.g. of $\prod_{i=1}^{n} M_i$ and $\prod_{i=1}^{n} N_i$. Then $\{L_i \mid i=1,n\}$ is a l.g.g. of C and D.

<u>Proof</u> We demonstrate the last part first. Evidently $\{L_i \mid i=1,n\}$ is a generalisation of C and D. Suppose $E = \{L_j \mid j=1,m\}$ is a generalisation of C and D. Then there are substitutions σ and μ such that $E\sigma \subseteq C$ and $E\mu \subseteq D$. Therefore $\{L'_j\sigma, L'_j\mu\}$ is a selection from C and D for any j. Consequently we can find a permutation π of the numbers $1,\ldots,n$ and integers $n_i \geq 0$ $(i=1,n)$, such that $\prod_{j=1}^{m} L'_j$ is a generalisation of $\prod_{i=1}^{n} M_{\pi(i)}^{n_i}$ and $\prod_{i=1}^{n} N_{\pi(i)}^{n_i}$. By lemma 2.3, $\prod_{i=1}^{n} L_{\pi(i)}^{n_i}$ is a l.g.g. of the latter two sequences. Therefore $\prod_{j=1}^{m} L'_j \leq \prod_{i=1}^{n} L_{\pi(i)}^{n_i}$ and so $E \leq \{L_i \mid i=1,n\}$ which proves that $\{L_i \mid i=1,n\}$ is a l.g.g. of C and D.

Notice that if some combination of sign and predicate letter occurs in both C and D then it also occurs in $\{L_i \mid i=1,n\}$. Therefore, if a finite set, H, of clauses has a selection it has a nonempty least general

generalisation obtained by repeatedly taking l.g.g's of pairs of clauses in a similar way to that in the proof of theorem 3.2.1.1. Suppose H has no selection. Then it cannot have a nonempty generalisation. (See the remark after the definition of selection.) Therefore its only, and hence its least, generalisation is $\emptyset$. This concludes the proof.

It follows from lemma 1, theorem 1 and its proof that one can effectively obtain the l.g.g. of a finite nonempty set of clauses. Consequently we can find an effective function inf from sets of clauses to clauses such that if $H \neq \emptyset$ then inf H is a l.g.g. of H and (for convenience) if $H = \emptyset$ then inf H is some fixed tautology. This definition seems to conflict with another definition of inf given in 3.2.1. However, we can define inf so that $\inf\{\{L_i\} \mid i=1,n\} = \inf\{L_i \mid i=1,n\}$, where the new definition is being used on the left and the old on the right of the equation. This follows from the following:

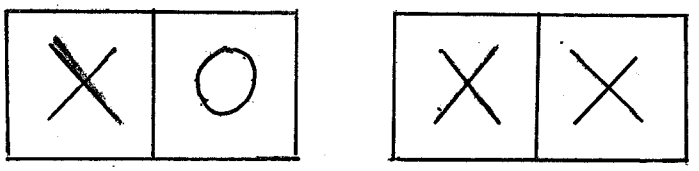
Corollary 1 A literal **L** is a l.g.g. of two literals M and N iff $\{\mathbf{L}\}$ is a l.g.g. of the clauses $\{M\}$ and $\{N\}$.

Proof If $\{\mathbf{L}\}$ is a l.g.g. of $\{M\}$ and $\{N\}$ then it is immediate that **L** is a l.g.g. of M and N. If **L** is a l.g.g. of M and N then $\{M\}$ and $\{N\}$ have a selection and the result follows from the second part of theorem 1.

Thus identifying literals with the corresponding one element clauses would not cause any conflict between the two definitions of l.g.g. This is not a trivial result.

Here is a less trivial example than the one presented at the beginning of this section.

Suppose that some two-person game is being played on a board with two squares, 1() and 2() and that the positions in figure 1 are won positions for the first player:



Position $p_1()$ Position $p_2()$

Figure 1

1() is the name of the left hand side square, and 2() of the right hand square; $p_1()$ and $p_2()$ are the names of the positions and O(), X() are the names of the marks O, X. The use of p, n_1, n_2 as variables is temporary. We describe the fact that these positions are wins by means of the following two clauses:

1. $\{\overline{Occ}(1(),X(),p_1()), \overline{Occ}(2(),O(),p_1()), Win(p_1())\}$.
2. $\{\overline{Occ}(1(),X(),p_2()), \overline{Occ}(2(),X(),p_2()), Win(p_2())\}$.

The course of the calculation is indicated in table 1.

$\overline{Occ}(1(),X(),p_1())$ $\overline{Occ}(1(),X(),p)$

$\overline{Occ}(1(),X(),p_2())$ $\overline{Occ}(1(),X(),p)$

$\overline{Occ}(1(),X(),p_1())$ $\overline{Occ}(1(),X(),p)$ $\overline{Occ}(n_1,X(),p)$

$\overline{Occ}(2(),X(),p_2())$ $\overline{Occ}(2(),X(),p)$ $\overline{Occ}(n_1,X(),p)$

$\overline{Occ}(2(),O(),p_1())$ $\overline{Occ}(2(),O(),p)$ $\overline{Occ}(2(),O(),p)$ $\overline{Occ}(n_2,O(),p)$ $\overline{Occ}(n_2,x,p)$

$\overline{Occ}(1(),X(),p_2())$ $\overline{Occ}(1(),X(),p)$ $\overline{Occ}(1(),X(),p)$ $\overline{Occ}(n_2,X(),p)$ $\overline{Occ}(n_2,x,p)$

$\overline{Occ}(2(),O(),p_1())$ $\overline{Occ}(2(),O(),p)$ $\overline{Occ}(2(),O(),p)$ $\overline{Occ}(2(),O(),p)$ $\overline{Occ}(2(),x,p)$

$\overline{Occ}(2(),X(),p_2())$ $\overline{Occ}(2(),X(),p)$ $\overline{Occ}(2(),X(),p)$ $\overline{Occ}(2(),X(),p)$ $\overline{Occ}(2(),x,p)$

$Win(p_1())$ $Win(p)$

$Win(p_2())$ $Win(p)$

Table 1

Each vertical column displays on alternate rows the sequences M_1 and M_2 at an instance of stage 2 of the algorithm of theorem 3.2.1.1. We find t_1 and t_2 by searching through M_1 and M_2 from left to right which is top to bottom in the table. As soon as two literals have become the same in a column, we do not mention them in subsequent columns.

Thus the least generalisation is:

$\{\overline{Occ}(1(),X(),p),\overline{Occ}(n_1,X(),p)\overline{Occ}(n_2,x,p),\overline{Occ}(2(),x,p),Win(p)\}$.

We use the algorithm of theorem 3.3.1.1. We can take $L=\overline{Occ}(n_1,X(),p)$ and $\sigma = \{1()/n_1\}$. This gives

$E_2 = C\sigma = \{\overline{Occ}(1(),X(),p), \overline{Occ}(n_2,x,p), \overline{Occ}(2(),x,p), Win(p)\}$ and $E_1 = \emptyset$.

Next, we can take $L=\overline{Occ}(n_2,x,p)$ and $\sigma = \{2()/n_2\}$ and obtain $E_2 = C\sigma = \{\overline{Occ}(1(),X(),p), \overline{Occ}(2(),x,p), Win(p)\}$ and $E_1 = \emptyset$.

After this every literal in E_2 goes into E_1 and eventually, $E_1 = \{\overline{Occ}(1(),X(),p), \overline{Occ}(2(),x,p), Win(p)\}$ and $E_2 = \emptyset$.

The algorithm stops at this point. The final clause says that if a position has an X in hole 1 and hole 2 has something in it, then the position is a win, which, given the evidence, is fairly reasonable.

The main computational weakness in the method for finding a reduced least generalisation lies in that part of the reducing algorithm which requires a test for subsumption. For suppose that we are looking for the l.g.g. of two clauses each with nine literals in a single predicate letter (this can arise in descriptions of tic-tac-toe, say); there will be at least eighty-one literals in the raw l.g.g., and we will have to try to tell whether or not a clause of eighty-one literals subsumes one of eighty.

It may very well be possible to find an algorithm which alternates the processes of finding an l.g.g. and reduction. In this way, in the tic-tac-toe example, for instance, sequences of eighty-one literals simply might not arise. Unfortunately we have not investigated this possibility.

3.4 Lattice properties of clauses

It is possible to define a lattice of equivalence classes of clauses in a way analogous to the construction of the lattice of equivalence classes of literals, given in 3.2.3. Once again, we will not consider the relativised lattice one could obtain using generalisation relative to Th.

The lattice, although not in general modular, does have a limited form of distributivity.

We also give a representation theorem which shows that the general clauses can be reached by a single lattice operation from the ground clauses. However it is not possible in general to take sups or infs of infinite sets and we give some counter-examples. We also give examples of strictly ascending and descending infinite chains of clauses with one binary predicate symbol and no function symbols. This is in strong constrast with the lattice of equivalence classes of literals where such chains (and all strictly descending infinite chains, even with function symbols) are impossible.

The representation theorem in the form of theorem 3.3.1.4. will prove a useful tool in later chapters. Section 3.3.2 on the infinitary properties of the lattice will be a source of counterexamples for various conjectures. The rest of the material is not essential.

3.3.1 Finitary properties

Before we are able to define the lattice, we need the dual of

the notion of an l.g.g. of two clauses.

The clause C is a most general instance (m.g.i.) of D and E iff:

1 $D \leq C$ and $E \leq C$.

2 For all C', if $D \leq C'$ and $E \leq C'$, then $C \leq C'$.

Evidently, any two m.g.i's of D and E are equivalent.

Lemma 1 Let ξ be a translation such that $D\xi$ and E have no common variables. Then $D\xi \cup E$ is an m.g.i. of D and E.

Proof 1 $D \leq D\xi \leq D\xi \cup E$.

$E \leq D\xi \cup E$.

2 Suppose $D \leq C'$ and $E \leq C'$. Then $D\xi \leq D \leq C'$ and so, for some minimal σ and μ, $D\xi\sigma \subseteq C'$ and $E\mu \subseteq C'$. Since σ and μ are minimal and $D\xi$ and E have no common variables, $\sigma \cup \mu$ exists and $(D\xi \cup E)(\sigma \cup \mu) = (D\xi\sigma \cup E\mu) \subseteq C' \cup C' = C'$.

We may now define the lattice. Let

$[C] = \{C' \mid C' \sim C\}$.

This set can be made into a lattice by the following definitions.

1 $[C] \sqcap [D] = [\inf\{C,D\}]$

2 $[C] \sqcup [D] = [C\xi \cup D]$ where ξ is a translation such that $C\xi$ and D have no common variables.

3 $[C] \leq [D]$ iff $C \leq D$.

One may see from the definition of inf and lemma 1 and the properties of $\sim$ that this does provide a good definition of a lattice. Notice that since m.g.i's and l.g.g's always exist, it is unnecessary to add any special elements. There is a natural bottom element, $[\emptyset]$, for $\emptyset \leq C$ for all C.

There is, in general, no top element. For example suppose P_i $(i \geq 1)$ is an infinite set of distinct predicate symbols. Then there can be no clause C such that $\{P_i(x)\} \leq C$ for all i. Alternatively let P be a unary predicate symbol and f a unary function symbol. There can be no clause C such that $\{P(f^i(x))\} \leq C$ for all i. However, suppose we consider only the equivalence classes of those clauses whose predicate symbols come from some fixed finite set whose function symbols are all from some fixed finite set and whose terms have depth less than or equal to some fixed integer. This set of equivalence classes forms a sublattice of the lattice of equivalence classes of clauses as may be seen from lemma 1 and the remarks preceding the definition of inf. The sublattice has a top element, namely $[\{L \mid [\{L\}]$ is in the sublattice and the only variable in L is $x\}]$.

The lattice is not modular, in general, for if one takes

$[C] = [\{Q(x), P(f())\}]$

$[D] = [\{Q(x), P(f()), P(x)\}]$

$[E] = [\{Q(f())\}]$

then $C \leq D$ but

$$
\begin{aligned}
&([E] \sqcap [D]) \sqcup [C] \\
&= ([\{Q(f())\}] \sqcap [\{Q(x),P(f()),P(x)\}]) \sqcup [C] \\
&= [\{Q(x)\}] \sqcup [\{Q(x),P(f())\}] \\
&= [\{Q(y),Q(x),P(f())\}] \\
&= [\{Q(x),P(f())\}] \\
&\neq [\{P(f()),P(y),Q(y)\}] \\
&= [\{P(f()),Q(f())\}] \sqcap [\{Q(x),P(f()),P(x)\}] \\
&= [\{Q(x),P(f()),Q(f())\}] \sqcap D \\
&= ([\{Q(f())\}] \sqcup [\{Q(x),P(f())\}]) \sqcap D \\
&= ([E] \sqcup [C]) \sqcap [D]
\end{aligned}
$$

This contradicts the modular equality. The corresponding non-modular sublattice is given in figure 1.

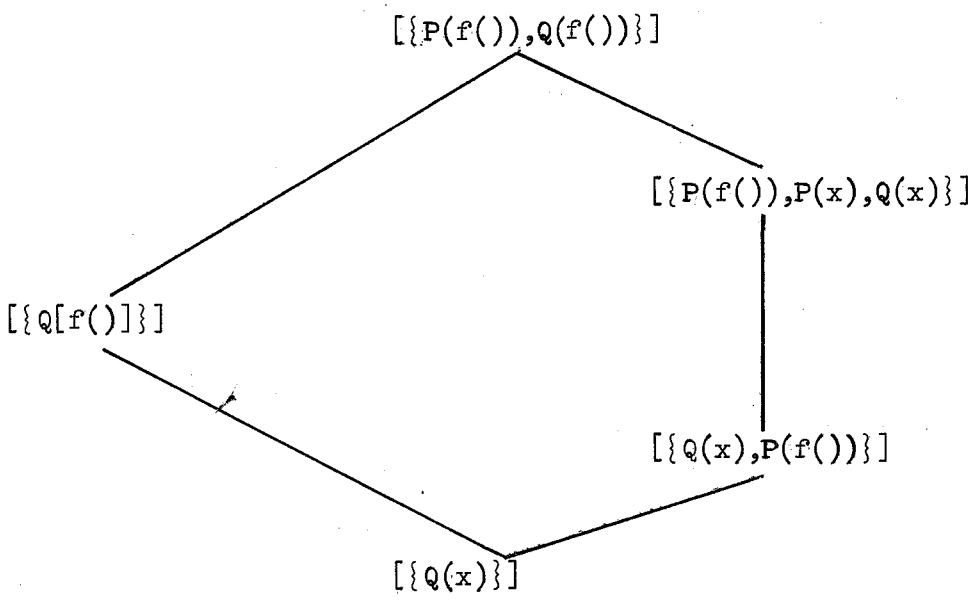


Figure 1

However we have:

Theorem 1 1) If B and C have no common ground terms, then

$$[A] \sqcap ([B] \sqcup [C]) = ([A] \sqcap [B]) \sqcup ([A] \sqcap [C]).$$

2) If A and B have no common ground terms and A and C have no common ground terms then

$$[A] \sqcup ([B] \sqcap [C]) = ([A] \sqcup [B]) \sqcap ([A] \sqcup [C]).$$

Proof 1) First of all, it follows from lattice theory that

$$[E] = ([A] \sqcap [B]) \sqcup ([A] \sqcap [C]) \leq [A] \sqcap ([B] \sqcup [C]).$$

That is, [E] is a lower bound of [A] and [B] $\sqcup$ [C]. We show that it is the greatest and hence that equality holds.

Suppose $[D] \leq [A]$, $[D] \leq [B] \sqcup [C]$. We need only show that $[D] \leq [E]$ to prove the theorem.

As $[D] \leq [B] \sqcup [C]$, there is a σ so that $D\sigma \subseteq B\xi \cup C$ where ξ standardises B and C apart, and so we can set $D = D_1 \cup D_2$ where $D_1\sigma \subseteq B\xi$; $D_2\sigma \subseteq C$. Now if D_1 and D_2 have a common variable x, then $B\xi$ and C have a common term $x\sigma$ which is impossible - for it cannot be ground as B and C have no common ground terms and it cannot be general as ξ standardises B and C apart. Hence D_1 and D_2 have no common variable and so: $[D] = [D_1] \sqcup [D_2]$

Hence as $[D_1] \leq [D] \leq [A]$ and $[D_1] \leq [B]$,

$$[D_1] \leq ([A] \sqcap [B])$$

Similarly $[D_2] \leq ([A] \sqcap [C])$

Hence $[D] = [D_1] \sqcup [D_2] \leq ([A] \sqcap [B]) \sqcup ([A] \sqcap [C]) = [E]$

which completes the proof.

2) Applying part 1 three times, we get

$$([A] \sqcup [B]) \sqcap ([A] \sqcup [C]) = ([A] \sqcap [A]) \sqcup ([A] \sqcap [C]) \sqcup ([B] \sqcap [A]) \sqcup ([B] \sqcap [C])$$

$= [A] \sqcup ([B] \sqcap [C])$, since

$[A] \sqcap [A] = [A]$; $[A] \sqcup ([A] \sqcap [C]) = [A]$

and $[A] \sqcup ([B] \sqcap [A]) = [A]$

by the absortive laws.

Corollary 1 The sublattice containing only the equivalence classes of those clauses which do not have any constant symbols is distributive.

Proof Note that this is a sublattice, since if $A \sim B$ then A has no constant symbols iff B has none and $\sqcap$ and $\sqcup$ do not introduce constant symbols. Distributivity is then immediate from Theorem 1 as B and C have no constant symbols implies that B and C have no ground terms (and hence no common ones).

We conjecture the following converse of theorem 1:

If $[A] \sqcap ([B] \sqcup [C]) = ([A] \sqcap [B]) \sqcup ([A] \sqcap [C])$

for all A, then B and C have no ground terms in common.

However to completely "understand" distributivity (or similarly for any other lattice properties) it would be best to have a necessary and sufficient simple syntactic criterion for A,B,C together for satisfaction of the distributive law.

Representation theorem

We need some definitions:

Let $x_1, x_2, \ldots$ be all the variable symbols written out as an infinite list.

Let $\delta_n = \{h() / x_1, \ldots, h() / x_n\}$

$\gamma_n = \{f(h()) / x_1, \ldots, f^n(h()) / x_n\}$

Theorem 2. If C does not contain the unary function symbol f then for all n,

$$[C] = [C \ \delta_n] \sqcap [C \ \gamma_n].$$

Before proving this, we remark that by choosing n large enough, we can express the equivalence class of any clause as the meet of the equivalence classes of two ground clauses. This theorem entails Reynold's Theorem 8 for literals, as can be seen using corollary 3.3.2.1.

However in this special case, one can strengthen Theorem 2 to:

Theorem 3. For every literal L and every n,

$$[\{L\}] = [\{L\} \ \delta_n] \sqcap [\{L\} \ \gamma_n]$$

We do not give the proof of theorem 3 which is a modification of Reynolds' proof of his theorem 8.

In general, one cannot remove the hypothesis of theorem 2. For if

$$C = \{P(x_1) \ , \ P(f(x_1))\},$$

$$\inf\{C\ \delta_1, C\ \gamma_1\} = \{P(x) \ , \ P(y) \ , \ P(f(h())) \ , \ P(f(x))\}$$

$$\sim \{P(x) \ , \ P(f(h())) \ , \ P(f(x))\}$$

$$\not\sim C.$$

We need some more definitions and a lemma for the proof of theorem 2.

Suppose W is a word. Then W' is obtained from W by continued application of (*) until this is no longer possible.

(*) Let n be the largest positive integer such that $f^n(h())$ occurs in W. Replace every occurrence of $f^n(h())$ in W by x_n.

Suppose $\sigma = \{t_1 / y_1, \ldots, t_m / y_m\}$ where the y_i occur amongst the x_i. We define $\sigma' = \{t_1' / y_1, \ldots, t_m' / y_m\}$.

Lemma 2.1 Suppose W does not contain f. Then

1 $(W \gamma_n)' = W$

2 $(W\sigma)' = W\sigma'$

Proof 1 This is obvious.

2 By induction on words on W.

Proof of theorem 2

Evidently C is a lower bound of C δ_n and C γ_n. We show that if $D \leq C\ \delta_n$ and $D \leq C\ \gamma_n$ then $D \leq C$.

Since $D \leq C\ \delta_n$ and C does not contain f and δ_n does not introduce f, D itself does not contain f.

For some σ, $D\sigma \subseteq C\ \gamma_n$.

Let L be in D. Then for some M in C

$$L\sigma = M\ \gamma_n$$

Hence $L\sigma' = (L\sigma)'$ (lemma 2.1.2)

$= (M\ \gamma_n)'$

$= M$ (lemma 2.1.1).

Hence $D\ \sigma' \subseteq C$, which concludes the proof.

For practical applications another version of the representation theorem is convenient. Let a_{ij} $(i,j \geq 1)$ be a doubly infinite sequence of distinct constant terms chosen so as to leave at least denumerably many constants unused.

Let $\gamma^i_j = \{a_{i1}/x_1, \ldots, a_{ij}/x_j\}$ $(j \geq 1)$.

Theorem 4 Suppose C does not contain any of the a_{ij}. If $i \neq i'$ then for all j, $[C] = [C\ \gamma^i_j] \sqcap [C\ \gamma^{i'}_j]$.

Proof Evidently $C \leq C\ \gamma_j^i$ for all i and j. Let W be a word. W'' is obtained from W by continued application of (**) until this is no longer possible.

(**) Replace an occurrence of a_{ij} by one of x_j

If $\sigma = \{t_1/y_1, \ldots, t_n/y_m\}$ then $\sigma'' = \{t_1''/y_1, \ldots, t_m''/y_m\}$. The rest of the proof is exactly analogous to the proof of theorem 2; the role of the operation, ', being played by the operation, ''.

3.3.2 Infinitary properties

It is shown here that the lattice has no pleasant infinite properties. It has strictly ascending and descending chains and is incomplete.

Infinite chains

We write $C < D$ when $C \leq D$ but not $D \leq C$.

$\{C_i \mid i \geq 1\}$ is a strictly ascending chain of clauses.

iff $C_i < C_{i+1}$ $(i \geq 1)$. Strictly descending chains are defined similarly.

We could define $[C] < [D]$ when $[C] \leq [D]$ but not $[D] \leq [C]$ and say that $\{[C_i] \mid i \geq 1\}$ is a strictly increasing chain of equivalence classes of clauses iff $[C_i] < [C_{i+1}]$ $(i \geq 1)$. But since $\{[C_i] \mid i \geq 1\}$ is strictly increasing iff $\{C_i \mid i \geq 1\}$ is, we prefer to deal with clauses rather than equivalence classes of clauses. A similar remark holds for

descending chains.

We will give examples of both ascending and descending chains using positive clauses with a single binary predicate symbol P and no function symbols.

Such a clause can be pictured as a graph whose nodes are the variable symbols appearing in the clause. The graph has an arc from x to y if and only if $P(x,y)$ is a literal in the clause.

Thus to $\{P(x,y), P(y,z), P(z,x)\}$ corresponds the graph:

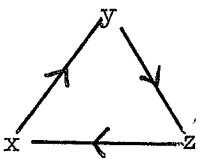


Figure 1

If $C_1\sigma \subseteq C_2$ then σ is a homomorphism of the corresponding graphs and in fact there is a 1-1 correspondence between such substitutions and graph homomorphisms.

We do not use this correspondence in any formal way, but with its help the reader should be able to see the truth of the various theorems.

Here is an example of a strictly ascending chain. Define $\{C_i \mid i \geq 1\}$ by $C_1 = \{P(x_0,x_1)\}$ and $C_i = C_{i-1} \cup \{P(x_{i-1},x_i)\}$ where the x_i are all different. In graph terms, C_i is the chain:

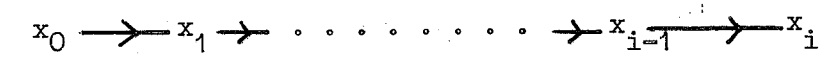


Figure 2

Evidently $C_i \leq C_{i+1}$ as $C_i \subseteq C_{i+1}$. Suppose $C_i\sigma \subsetneq C_{i-1}$.

If $x_k\sigma = x_l$, then as $P(x_k,x_{k+1})$ is in C_i, $P(x_l,x_{k+1}\sigma)$ is in C_{i-1}.

Hence $x_{k+1}\sigma = x_{l+1}$ as $P(x_l,x_{l+1})$ is the only literal in C_{i+1} that has x_l as its first argument. Thus it follows easily that σ is injective - but this contradicts the fact that C_i has i variables and C_{i-1} only i-1. Hence no such σ exists and $C_i \not\leq C_{i-1}$.

This chain is bounded above by $\{P(x,x)\}$ for $C_i\sigma = \{P(x,x)\}$ where $\sigma = \{x / x_0, \ldots, x / x_i\}$. The ascending chain could have been produced in a trivial way by using a unary function, one takes the clauses $\{Q(f^i(x))\}$, or we could use an infinite supply of unary predicate symbols Q_i and take the chain:

$$\{C_i' \mid i \geq 1\} \text{ where } C_1' = \{Q_1(x)\};\quad C_{i+1}' = C_i' \cup \{Q_{i+1}(x)\}$$

With a little more effort we can produce a strictly descending chain. Let $[i,n]$ $(i \geq 0,\ n \geq 1)$ be the equivalence class of i under congruence modulo n. Thus $[0,2]$ is the set of even integers. Let $x_{[i,n]}$ be an infinite supply of variables labelled by the set $\{[i,n] \mid i \geq 0,\ n \geq 1\}$.

We define the positive clause D_n in the binary predicate letter P, with no function symbols and whose variables are a subset of

$\{x_{[i,n]} \mid i \geq 0,\ n \geq 1\}$ by

$P(x_{[i,m]}, x_{[j,l]}) \in D_n$ iff
$m = l = n$ and $j \equiv i+1 \pmod{n}$. D_n contains n literals.

In graph terms, D_n is the cycle:

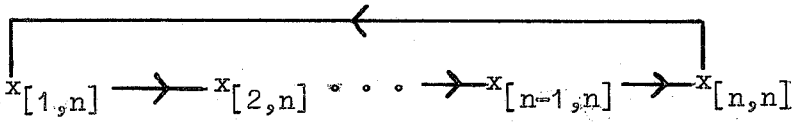


Figure 3

Then we will show that:

$\{D_2n \mid n \geq 1\}$ is a strictly descending chain.

Let γ_n be a substitution such that

$x_{[1,2^{n+1}]}\ \gamma_n = x_{[i,2^n]}$.

Such a γ_n exists as if $i \equiv j \pmod{2^{n+1}}$ then $i \equiv j \pmod{2^n}$.

Then $D_2n+1\ \gamma_n = \{P(x_{[i,2^{n+1}]}, x_{[i+1,2^{n+1}]})\ \gamma_n \mid 0 \leq i < n\}$
$= \{P(x_{[i,2^n]}, x_{[i+1,2^n]}) \mid 0 \leq i < n\}$
$= D_2n$.

On the other hand, suppose that σ_n is a substitution such that

$D_2n\ \sigma_n \subseteq D_2n+1$. Let $x_{[1,2^n]}\ \sigma_n = x_{[1,2^{n+1}]}$

We will show by finite induction that if

$1 \leq i \leq 2^n$ then $x_{[i,2^n]}\ \sigma_n = x_{[i+1-1,\ 2^{n+1}]}$.

This is true for i=1. Suppose that it holds for i. Then as

$L = P(x_{[i,2^n]}, x_{[i+1,2^n]}) \in D_2n$, it follows that

$L\ \sigma_n = P(x_{[i+l-1,2^{n+1}]}, x_{[j,2^{n+1}]})$ say. Hence by the definition of D_2n+1, $j \equiv i+1 \pmod{2^{n+1}}$ so $x_{[i+1,2^n]}\ \sigma_n = x_{[i+1,2^{n+1}]}$. This completes the induction.

Now, $L = P(x_{[2^n,2^n]}, x_{[1,2^n]}) \in D_2n$. Hence

$L\ \sigma_n = P(x_{[2^n+l-1,2^{n+1}]}, x_{[1,2^{n+1}]}) \in D_2n+1$.

Hence $2^n+1-1 \equiv 1-1 \pmod{2^{n+1}}$. This is false and it follows that no such σ_n exists. Hence, D_2n $(n \geq 1)$ is indeed a strictly descending chain of clauses with one binary predicate symbol and no function symbols.

One can show easily that there are only a finite number $(\leq 2^{(2^{n-1})}-1)$ of inequivalent, under $\sim$, positive clauses with n unary predicate symbols and no function symbols. It follows that there can be no strictly descending chain of clauses with unary predicate symbols but no function symbols. Hence our result is the best possible as regards arities of the predicates involved.

Incompleteness

We show next that $\{[C_i] \mid i \geq 1\}$ has no supremum and $\{[D_2j] \mid j \geq 1\}$ has no infimum. Thus our lattice is not complete wrt. sups or infs. The existence of infinite strictly ascending and

descending chains would follow from this by general lattice theory but it is better to display actual examples.

Lemma 1 1) $C_i \leq D_2 j$ $(i,j \geq 1)$

2) There is no E such that

$C_i \leq E \leq D_2 j$ for all $i, j \geq 1$.

Proof 1) Let $\gamma_{ij} = \{x_{[0,2^j]} / x_0, \ldots, x_{[i,2^j]} / x_i\}$.

Then $C_i \gamma_{ij} = \{P(x_l, x_{l+1}) \mid 0 \leq l \leq i\}\ \gamma_{ij}$

$= \{P(x_{[l,2^j]}, x_{[l+1,2^j]}) \mid 0 \leq l \leq i\}$.

$\subseteq D_2 j$.

2) Suppose there is such an E. Evidently E must be a positive clause, with no function symbols, in the single binary predicate symbol P. Suppose E has n variable symbols. There is a σ_n such that $C_n \sigma_n \subseteq E$. Now C_n has n+1 variable symbols. Hence there are i_1, i_2 $(i_1 < i_2)$, so that $x_{i_1} \sigma_n = x_{i_2} \sigma_n$. We choose such an i_1, i_2 so as to minimise $i_2 - i_1$.

Then $E' = \{P(x_l, x_{l+1}) \sigma_n \mid i_1 \leq l < i_2\} \subseteq E$. We show that $E' \sim D_{(i_2 - i_1)}$.

Let $\mu = \{x_{[i_1, i_2 - i_1]} / x_{i_1} \sigma_n, \ldots, x_{[i_2, i_2 - i_1]} / x_{i_2} \sigma_n\}$

Then μ is well-defined and 1-1 for $x_{l_1} \sigma_n = x_{l_2} \sigma_n$ $(l_1 \leq l_2)$ <=> $l_1 = i_1$ and $l_2 = i_2$ or $l_1 = l_2$ (by minimality of $i_2 - i_1$)

<=> $l_1 \equiv l_2 \pmod{i_2 - i_1}$

$\Leftrightarrow x_{[l_1, i_2-i_1]} = x_{[l_2, i_2-i_1]}$

$\Leftrightarrow x_{l_1} \sigma_n \mu = x_{l_2} \sigma_n \mu$.

Next $E'\mu = D_{(i_2-i_1)}$ for $E'\mu = \{P(x_l \sigma_n \mu, x_{l+1} \sigma_n \mu) \mid i_1 \leq l < i_2\}$

$= \{P(x_{[l,i_2-i_1]}, x_{[l+1,i_2-i_1]}) \mid i_1 \leq l < i_2\}$

$= D_{(i_2-i_1)}$.

Hence $E' \sim D_{(i_2-i_1)}$. Now since $D_{(i_2-i_1)} \sim E' \subseteq E \leq D_2 j (j \geq 1)$, there is a ν so that

$D_{(i_2-i_1)} \nu \subseteq D_2 (i_2-i_1)$ We can show that this is a contradiction in exactly the same way we showed that there is no σ_n such that

$D_2 n \ \sigma_n \subseteq D_2 n+1$, thus completing the proof.

Note that since $\{C_i \mid i \geq 1\}$ is a strict chain, all the inequalities of the first part of lemma 1 are actually strict.

Theorem 1 $\{[C_i] \mid i \geq 1\}$ has no supremum and $\{[D_2 j] \mid j \geq 1\}$ has no infimum.

Proof Suppose $[E] = \bigsqcup [C_i]$ exists. Then as $C_i \leq D_2 j$, by the first part of lemma 1, and since $[E]$ is a supremum, $E \leq D_2 j$. This contradicts the second part of lemma 1. One proves that $\bigsqcap [D_2 j]$ cannot exist in the same way.

Note that the two trivial examples of strictly increasing infinite chains have no upper bound at all. This contrasts with the strictly decreasing case where $\emptyset$ is always a lower bound. $\{C_i \mid i \geq 1\}$ is an

example of a strictly increasing infinite chain, which has no supremum but is bounded above by $\{P(x,x)\}$.

It is possible by a further extension of the above methods to produce a set of equivalence classes of clauses $\{[C_i] \mid -\infty < i < +\infty\}$ which has no supremum or infimum and is such that $C_i < C_{i+1}$. However we have done enough to show that the lattice does not possess any of the usual properties of infinite sets of elements, at least in the general case.