1

Relational Programming

R. J. Popplestone
Department of Artificial Intelligence
University of Edinburgh, UK

1. INTRODUCTION

A programming language needs simple and well defined semantics. The two favoured theoretical bases for languages have been lambda calculus as advocated by Landin and others, and predicate calculus as advocated by Kowalski (see Landin (1966) and Kowalski (1973)). In this paper I adopt an approach based on predicate calculus, but in a manner that differs from the existing PROLOG language (Warren 1975 and Battani & Meloni 1973) in that I adopt a "forward inference" approach – inferring conclusions from premises, rather than the "backward inference" approach of PROLOG, which starts with a desired conclusion and tries to find ways of inferring it. This difference is reflected in the internal structure of the associated implementations, that of PROLOG being a "backtrack search" kind of implementation, while the most obvious implementation of the system proposed here involves a kind of mass operation on tables of data, reminiscent of APL (Iverson 1962) but in fact identical in many respects with the work of Codd (Codd 1970) on relational data bases. Indeed, from one perspective this paper can be seen as an extension of Codd's work into the realm of general purpose computing.

As in the case of PROLOG it is necessary for the user of the relational programming system to make statements which are not associated with the logical structure of the problem, but reflect the need to control the computation. In PROLOG these are effected by the use of extra-logical control primitives, but in our system control is exercised by the introduction of predicates for that purpose, which have exactly the same semantics as the predicates relevant to the logical structure of the problem.

In later sections I deal with the problem of introducing equality into the system, in a way that reflects the normal mathematical usage of equality. In this I am attempting something that programming systems normally do not try, although the ABSYS system (Elcock et. al. 1971) was built with equality as a

central concept, and the unification algorithms of PROLOG provide a limited treatment.

When we come to consider the question of implementation, the relational system seems to open a number of avenues of possibility, and it certainly raises a number of problems. The most interesting developments compared with other Artificial Intelligence languages are the possibility of making efficient use of the address space of a modest size computer, or of handling much bigger problems on a larger machine, and of exploiting parallelism. The major problem raised by the system is that the most natural implementation involves repeating computations unnecessarily.

2. AN EXAMPLE

Figure 1A shows a simple "blocks world" in which we have five blocks denoted by the numbers 1, 2, 3, 4, 5. In this example we use numbers to denote the entities in the world as a matter of convenience, since we are going to express the relations holding between entities in the form of tables, and numbers are the most convenient symbols to use. In later sections of the paper we shall use the numbers "as numbers", that is as entities that can be operated on by the normal functions of arithmetic, and will want to distinguish them from non-numeric entities like blocks.

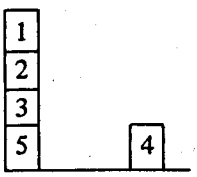

Fig. 1A – A simple "blocks world".

ON	
1	2
2	3
3	5

Fig. 1B – The table for the ON relation.

Suppose that the state of this world is specified by stating which blocks are on others. This relationship, which we shall call ON, can be expressed in tabular form as in Fig. 1B. This can be regarded as an association of the name ON, which we shall call a predicate name, or just a predicate, with a binary relation, that is the set of pairs {(1,2) (2,3) (3,5)}. Now suppose we want to

define a new relation, called ABOVE, in terms of ON. This can be done by using the following two clauses

$$\text{ON}(X,Y) \Rightarrow \text{ABOVE}(X,Y) \qquad \text{(C1)}$$
$$\text{ON}(X,Y) \;\&\; \text{ABOVE}(Y,Z) \Rightarrow \text{ABOVE}(X,Z) \qquad \text{(C2)}$$

That is if X is ON Y, then X is ABOVE Y, and if X is ON Y, and Y is ABOVE Z, then X is ABOVE Z.

It is possible to find a value for ABOVE, while keeping the same value for ON, by applying a process illustrated in Fig. 2. In this process we cycle round the clauses C1 and C2 and at each stage we form a table whose columns are labelled with the names of the variables of the clause currently being examined, and whose rows tabulate the possible values of the variables of the left side of each clause. We then use this table to produce new rows to be added into the table which is the current value of ABOVE. In this way we create a sequence $r1 \ldots . r4$ of tables which specify possible approximations to the ABOVE relationship. If we carry on with the process beyond $r4$ no new rows are added to in producing $r5 \ldots .$, and we say we have reached a fixed point. At this point ON = {(1,2) (2,3) (3,5)} and ABOVE = {(1,2) (1,3) (1,5) (2,3) (2,5) (3,5)} and with these values (1) and (2) are satisfied according to the usual laws of logic.

In a more general case, where the values of a number of predicates are being built up, each step in the approximation will be represented by a sequence of tables, each table specifying a possible value for one of the predicates, and in

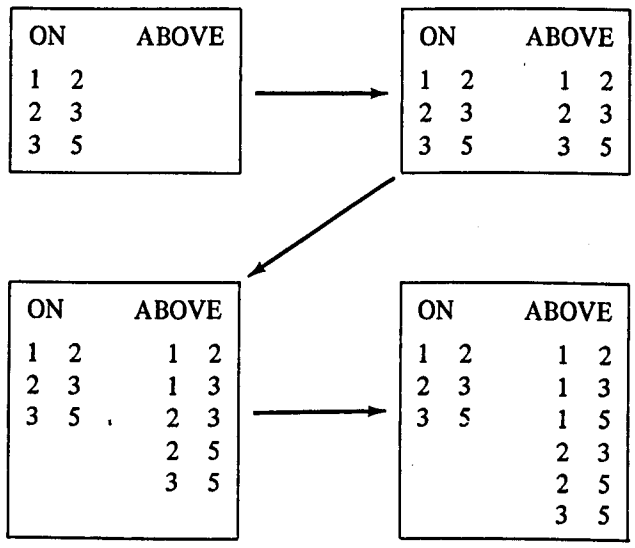

Fig. 2 – Computing the ABOVE relation.

the next two sections we shall give a formal definition of the process illustrated by the example above, and prove that it gives rise to a sequence of relations which form an interpretation of a given set of logic clauses.

3. A FORMAL SPECIFICATION OF THE SYSTEM

Let us now develop a precise mathematical specification of systems of the type outlined above.

3.1 The basic sets

We need a set E of entities, which form the universe of discourse. Thus in the above example $E = \{1, 2, 3, 4, 5\}$.

We also need a set P of "predicate symbols" and a set V of "variable symbols". Thus in our example $P = \{$"ON", "ABOVE"$\}$, and $V = \{$"X", "Y", "Z"$\}$, where the quotes denote that the symbol itself is being referred to. We insist that E and V are disjoint sets.

3.2 Finite sequences

For many purposes we will need to make use of finite sequences of elements, usually of entities and variables. We shall use bold lower case letters to denote sequences. If a is a sequence, then a_k denotes the kth element of a. (a_k) is occasionally used to denote the sequence a. By |a| we mean the set $\{a_k\}$ of elements of the sequence a. The length of a sequence a, denoted by length (a) is the number of elements in it (counting repetitions). It is convenient to use a sequence without repetitions as a way of referring to the members of another sequence of the same length. Let v be a sequence without repeated members, and let a be a sequence. Let v_k be a member of |v|. Then we use the notation

$$v_k \text{ of a wrt v} = a_k \quad .$$

For example, let v = ("X", "Y") then

$$\text{"}Y\text{" of (5,6) wrt ("}X\text{", "}Y\text{")} = 6 \quad ,$$

and

$$\text{"}Y\text{" of (5,7) wrt ("}Y\text{", "}X\text{")} = 5 \quad .$$

The empty sequence, which we shall denote by (), is a sequence of no elements. If x is a sequence, and x is an element, then conseq(x,x) = t, where $t_1 = x$, and $t_k = x_{k-1}$. We shall use a form of structural induction (Burstall 1969) on sequences, whereby to prove a result for all sequences, it is sufficient to prove it for the empty sequence, and to prove that if a result holds for a sequence x, then it holds for conseq(x,x).

We will often need to extend a mapping defined on elements to apply to a sequence. If σ is a mapping and x is a sequence then $\sigma(\mathbf{x}) = (\sigma(x_j))$.

3.3 Horn clauses

The logical sentences which we shall use will consist of conjunctions of Horn clauses. A Horn clause has the form L_1 & L_2 & ..... L_n => L where the L_j and L are literals, that is they simply take the form of a predicate applied to arguments. More formally, a clause is a pair (L,L') consisting of a sequence L of literals which we shall call the negative literals of the clause, and a literal L' which we shall call the positive literal of the clause. Each literal itself is a pair (p,a) where p is a predicate in P, and a is a sequence of "arguments" drawn from $V \cup E$.

3.3.1 *A restriction on clauses*

An additional constraint which we place on the clauses allowed in our system is that if (L ,L') is a clause then all variables occurring in L' must occur in some L in |L|.

Thus from our example, ("ON"("X", "X")), and ("ON",("X",1)) are literals, as well as ("ON",("X", "Y")). In the sequel, we shall use the normal notation (for example ON(X,Y)) when referring to particular literals of the object language. We are not of course restricted to binary predicates, for example one might have BETWEEN, which is ternary, as in BETWEEN(X,Y,Z). However, we suppose that with each predicate symbol p there is associated a positive integer arity(p) which specifies the length of any argument sequence which is associated with it in forming a literal.

3.4 Relations

Our clauses will be interpreted in terms of relations on E. By a k-ary relation on E we mean a subset r of the cartesian product E^k. We denote the set of all k-ary relations on E by Rel(E,k) which is of course the power set of E^k. Thus in our example the tables, without column headings, represent relations, with the rows being the "tuples" taken from E^k.

We will interpret a sequence of clauses C by associating a relation with each predicate occurring in C. In fact, let p be a sequence, without repetitions of the predicates of C. Then a sequence of relations r, with length (r) = length (p) can be considered as a possible interpretation if arity (r_k) = arity (p_k) $1 \leqslant k \leqslant$ length (p). We denote the set of all such relation sequences by $\mathcal{R}$ and call it the domain of interpretation of C. $\mathcal{R}$ is then the cartesian product

$$\mathcal{R} = \Pi_k \text{ Rel}(E, \text{arity } (p_k)) \qquad (3.4.1)$$

In our example, C can be the sequence (C1, C2) and p the sequence ("ON","ABOVE").

$\mathcal{R}$ is the product Rel(E,2) X Rel(E,2), so that if (r1,r2) in $\mathcal{R}$ then r1 is a possible value for ON, and r2 is a possible value for ABOVE.

Now, for any k, Rel(E,k) is a complete lattice under the normal set theory operations of union and intersection. It follows that $\mathcal{R}$ is also a complete lattice,

since it is the direct product of such lattices. We need to make use of the following "fix point" theorem about completing lattices.

3.4.2 *Theorem*

If $\mathcal{L}$ is a complete lattice, and $\theta : \mathcal{L} \to \mathcal{L}$ has the property that $\theta(1) \geqslant 1$, for each 1 in $\mathcal{L}$, then there is an element u in $\mathcal{L}$ with the property that $\theta(u) = u$.

Proof

Let $v = \cup\{1 | \text{theta}(1) > 1\}$. Then either $\theta(v) = v$, in which case the required result holds, or $\theta(v) > v$, (where $\cup$ denotes the lattice operation). Let us suppose that $\theta(v) > v$. Then either $\theta(\theta\ (v)) = \theta(v)$, in which case $\theta(v)$ is the required "fixed point" or $\theta(\ \theta(v)) > (v)$. But $v = \cup\{1 | \theta(1) > 1\}$, and $\theta(v) > v$, a contradiction.

That $\theta(v)$ is in fact sometimes the fixed point can be shown by the following example. Let $\mathcal{L} = [0,1]$ union $\{2\}$, with the standard ordering on the reals. Let $\theta(x) = x + (1-x)2$, $x < 1$, $\theta(x) = 2$ otherwise. Then $v = 1$ in the above proof, but $\theta(v) = 2$ is the fixed point.

We shall also have need of a more constructive form of the fixed point theorem.

3.4.3 *Theorem*

Let $\mathcal{L}$ be a complete lattice. Let $\theta : \mathcal{L} \to \mathcal{L}$ have the properties (i) $\theta(1) \geqslant 1$, and (ii) If $\{1_\alpha\} \subset \mathcal{L}$ is an ascending chain of members of $\mathcal{L}$, then $\theta(\cup\{1_\alpha\}) = \cup\{\theta(1_\alpha)\}$. Let 1_α be any element of $\mathcal{L}$. Then if $1_\alpha = \bigcup_i \theta^i(1_\alpha)$, $\theta(1_\alpha) = 1_\alpha$.

Proof

$\theta(1_\alpha) = \theta(\bigcup_{i>0} \theta^i(1_\alpha)) = \bigcup_{i>0} \theta(1_\alpha) = \bigcup_{i>0} \theta^i(1_\alpha)$ since $\theta^i(1_\alpha) \geqslant \theta^0(1_\alpha) = 1_\alpha$.

In the theory of computation, it is customary to refer to a function satisfying the preconditions of (3.4.3) as being continuous.

3.5 Labelled relations

To simplify some of the definitions we shall make later, and to clarify the processes involved, we need to introduce the notion of a labelled relation, which is a pair, $(\mathbf{v},r)$ where $\mathbf{v}$ is a sequence of variables without repetition, and r is a relation having the property that arity(r) = length $(\mathbf{v})$. The main benefit of this device is to be found in the definitions of attach, detach, and join, to be found below. In Fig. 2, the labelled relations are represented by tables with column labels, the row of column labels corresponding to the sequence of variables. In the first development of the theory we used unlabelled relations, which necessitated permuting columns in a way that was difficult to follow. It should be noted that the relational data base work of Codd and others makes use of labelled relations, but the necessity of attaching and detaching labels seems not to be generally recognised.

We shall denote labelled relations by q, q', q'', etc. In particular, $q_0 = (0, \{0\})$ will be called the null labelled relation.

If $\{q_\alpha\} = \{(\mathbf{v}, r_\alpha)\}$ is a family of labelled relations with the same label sequence, then we write $\cup_\alpha q_\alpha$ for $(\mathbf{v}, \cup_\alpha r_\alpha)$. We will not require a union operation over labelled relations having different v's.

3.6 The rho function

In this section we define a function ρ, which has the property that if **C** is a sequence of clauses: $\rho(\mathbf{C}) : \mathcal{R} \to \mathcal{R}$, and $\rho(\mathbf{C})$ satisfies the conditions specified for θ in (3.4.2) and (3.4.3). In Sec. 4 we shall show that any fixed point of $\rho(\mathbf{C})$ is an interpretation of **C**.

ρ is defined for clause sequences by building up a definition via predicates, literals, and literal-sequences and clauses. The definition involves a number of auxiliary functions which we shall define, namely inject, project, attach, detach, and join (this last is written $*$). Let **C** be a clause sequence, $\mathbf{C} = (\mathbf{L}, (p', \mathbf{a}'))$ be a clause, where **L** is a literal sequence. Let $L = (p, \mathbf{a})$ be a literal. Then

$$\rho((\mathbf{C})) = \rho(\mathbf{C}) \qquad \rho 1$$
$$\rho(\text{consseq}(\mathbf{C}, \mathbf{C})) = \rho(\mathbf{C}) \circ \rho(\mathbf{C}) \qquad \rho 2$$
$$\rho((\mathbf{L}, (p', \mathbf{a}')))(\mathbf{r}) = \mathbf{r} \cup \text{inject}(p', \mathbf{p})(\text{detach}(\mathbf{a}, \rho'(\mathbf{L})(\mathbf{r}))) \qquad \rho 3$$
$$\rho'((\,))(\mathbf{r}) = q_0 \qquad \rho 4$$
$$\rho'(\text{consseq}((L, \mathbf{L}))(\mathbf{r}) = \rho(L)(\mathbf{r}) * \rho'(\mathbf{L})(\mathbf{r}) \qquad \rho 5$$
$$\rho((p, \mathbf{a}))(\mathbf{r}) = \text{attach}(\mathbf{a}, \rho(p)(\mathbf{r})) \qquad \rho 6$$
$$\rho(p)(\mathbf{r}) = \text{project}(p, \mathbf{p})(\mathbf{r}) \qquad \rho 7$$

where $\circ$ is the functional composition operation defined by $(f \circ g)(x) = g(f(x))$.

Before we go on to complete the formal details of the definition of ρ, let us consider its meaning in our example. $\rho(\mathbf{C}1)$ is a function which takes a pair, $(r1, r2)$, where $r1$ is a possible value for the ON relation, and $r2$ is a possible value for the ABOVE relation, and produces $(r1', r2')$, which are again possible values for ON, and ABOVE respectively, according to the procedure sketched out in Sect. 2. Note that the value of ON will not in fact change, since it does not occur positively in any clause, nevertheless it is convenient to include it in the considerations.

3.7 The auxiliary functions

Returning to the definition of $\rho 1$-$\rho 7$, the inject and project mappings simply serve to access components of members of the cartesian product, $\mathcal{R}$. In fact

$$\text{inject}(p, \mathbf{p})(r) = (r_k) \text{ where } r_k = r \text{ if } p_k = p, \text{ and } r_k = \emptyset \text{ otherwise.} \qquad \rho 8$$
$$\text{project}(p, \mathbf{p})(\mathbf{r}) = p \text{ of } \mathbf{r} \text{ wrt } \mathbf{p} \qquad \rho 9$$

Thus in our example,

$$\text{project (ON, (ON,ABOVE))}(\{(1,2),(2,3),(3,5)\}, \{\ \}) = \{(1,2),(2,3),(3,5)\}$$

and selects the value of the ON relation. Likewise,

$$\text{inject(ABOVE, (ON,ABOVE)}\ (\{(1,3),(2,5)\}) = (\{\ \}, \{(1,3),(2,5)\})$$

and is used in the creation of a new value for the ABOVE relation.

While unlabelled relations are associated with predicates, labelled relations are associated with literals, and sequences of literals. If L is a literal, with associated labelled relation $(\mathbf{v}, r)$ then v is the sequence of variables of L, without repetitions. The same is true for literal sequences. The attach function is used to go from unlabelled relations associated with predicates to labelled relations associated with literals, and the detach function makes the opposite transition. detach (a,(v,r)) takes an argument sequence a, $|\mathbf{a}| \subset E \cup V$ and a labelled relation (v,r), and produces an unlabelled relation r' for which arity(r') = length(a). Our clauses are restricted by (3.3.1) so that the condition $|\mathbf{a}| \cap V \subset |\mathbf{v}|$ is always satisfied.

$$\begin{aligned}&\text{detach } (\mathbf{a}(\mathbf{v},r)) = \\ &\{\mathbf{t}' \mid \exists \mathbf{t} \in r \\ &\quad a_k \in |\mathbf{v}| = \mathbf{t}'_k = a_k \text{ of t wrt v} \\ &\quad a_k \in E \Rightarrow t'_k = a_k\}.\end{aligned} \qquad \rho 10$$

In our example, for C2, when we are forming a new value of ABOVE, we use detach with arguments ("X","Z") and (("X","Y","Z"), (1,2,3),(2,3,5)) to obtain the relation {(1,3),(2,5)} to be added into ABOVE.

Suppose now that a is a sequence for which $|\mathbf{a}| \subset \mathrm{E} \cup \mathrm{V}$ and r is a relation, such that arity(r) = length(a). Then attach (a,r) is a labelled relation (v,r'), where $|\mathbf{v}| = |\mathbf{a}| \cap V$ and

$$\begin{aligned}&r' = \{\mathbf{t}' \mid \exists \mathbf{t} \in r \\ &\forall k,k' v_{k'} = a_k \Rightarrow v_{k'} \text{ of } \mathbf{t}' \text{ wrt } \mathbf{v} = \mathrm{t_k} \\ &\forall k\ a_k \in |E| \Rightarrow t_k = a_k\}.\end{aligned} \qquad \rho 11$$

In our example, if L = ABOVE(Y,Z), then we have to perform

$$\text{attach}((\text{"}Y\text{"},\text{"}Z\text{"}), \{(1,2),(2,3),(3,5)\}) = ((\text{"}Y\text{"},\text{"}Z\text{"}), \{(1,2),(2,3),(3,5)\})$$

a trivial example in fact. If, however, we attach ("X","X") to the above relation,

then we get the labelled relation (("X"), {[]}), since nothing is above itself. If we attach ("X",5), we get (("X"), {(3)}).

The next operation we have to consider is the join operation. This is involved in combining the labelled relations derived from two different literals. Let (v,r) and (v',r') be two labelled relations, then

$$
\begin{aligned}
&(v'',r'') = (v,r) * (v',r') <=> \\
&|v''| = |v| \cup |v'|. \\
&r'' = \{t'' \mid \exists t,t' \\
&\quad v \in |v| => v \text{ of } t'' \text{ wrt } v'' = v \text{ of } t \text{ wrt } v \\
&\quad v' \in |v'| => v' \text{ of } t'' \text{ wrt } v'' = v' \text{ of } t' \text{ wrt } v'\}.
\end{aligned}
\qquad \rho 12
$$

For example, in processing C2 we have to compute the join

$$
\begin{aligned}
&((\text{"}X\text{"},\text{"}Y\text{"}),\{(1,2),(2,3),(3,5)\}) * ((\text{"}Y\text{"},\text{"}Z\text{"}),\{(1,2),(2,3),(3,5)\}) \\
&= ((\text{"}X\text{"},\text{"}Y\text{"},\text{"}Z\text{"}),\{(1,2,3),(2,3,5)\}).
\end{aligned}
$$

Note that $q0 = ((),\{()\})$ is an identity for $*$, and that $*$ is commutative and associative.

3.8 Summary of section 3

We have now completed the definition of ρ. Note that $\rho 3$ implies that $\rho(C)(\mathbf{r}) \geqslant \mathbf{r}$ and so, from $\rho 1$ and $\rho 2$, $\rho(\mathbf{C})(\mathbf{r}) \geqslant \mathbf{r}$. Thus $\rho(\mathbf{C})$ has at least one fixed point by 3.4.2, $\mathbf{r}_1$, say, and moreover, $\mathbf{r}_1$ must also be a fixed point for $\rho(\mathbf{C})$ or all C in $|\mathbf{C}|$, by the definition of $\rho(\mathbf{C})$. It follows that $\mathbf{r}_1$ is a fixed point for any clause sequence $\mathbf{C}'$ st $|\mathbf{C}'| = |\mathbf{C}|$ so that the fixed point depends only on the set of clauses, not on their order. Before we investigate the properties of ρ further in Sec. 4, let us work through an example of the application of ((C1,C2)).

Let us consider the initial state of our example, represented by $\mathbf{r}_0 = (\{(1,2),(2,3),(3,5)\},\{\})$, $\mathbf{p}$ = ("ON", "ABOVE") so that "ON" is associated with the first relation in the pair $\mathbf{r}_0$, and ABOVE with the second (null) relation. Then by $\rho 1$ & $\rho 2$

$$
\begin{aligned}
&\rho((\mathbf{C1},\mathbf{C2}))(\mathbf{r}_0) = \rho(\mathbf{C2})(\rho(\mathbf{C1})(\mathbf{r}_0)) && (3.8.1) \\
&\rho(\mathbf{C1})(\mathbf{r}_0) = \mathbf{r}_0 \cup \text{inject}(\text{"ABOVE"},\mathbf{p})(\text{detach}((\text{"}X\text{"},\text{"}Y\text{"}),\rho'(\mathbf{L})(\mathbf{r}_0) \\
&\quad \text{where } \mathbf{L} = (\text{"ON"},(\text{"}X\text{"},\text{"}Y\text{"}))) && (3.8.2) \\
&\rho'(\mathbf{L})(\mathbf{r}_0) = q_0 * \rho((\text{"ON"},(\text{"}X\text{"},\text{"}Y\text{"})))(\mathbf{r}_0) && (3.8.3) \\
&\rho((\text{"ON"},(\text{"}X\text{"},\text{"}Y\text{"})))(\mathbf{r}_0) = \text{attach}((\text{"}X\text{"},\text{"}Y\text{"}),\rho(\text{"ON"})(\mathbf{r}_0)) \\
&= \text{attach}((\text{"}X\text{"},\text{"}Y\text{"}),\text{project}((\text{"ON"}(\text{"ON"},\text{"ABOVE"})))(\mathbf{r}_0)) \\
&= \text{attach}((\text{"}X\text{"},\text{"}Y\text{"}),\{(1,2),(2,3),(3,5)\}) \\
&= ((\text{"}X\text{"},\text{"}Y\text{"}),\{(1,2),(2,3),(3,5)\}) = \rho'(\mathbf{L})(\mathbf{r}_0) \\
&\text{since } q_0 \text{ is an identity for } *.
\end{aligned}
$$

Thus, applying (3.8.2) we get that

$$\rho(\mathrm{C1})(\mathrm{r}_0) = \mathrm{r}_0 \cup \mathrm{inject}(\text{“ABOVE”},\mathrm{p})\,(\mathrm{detach}((\text{“}X\text{”},\text{“}Y\text{”}),((\text{“}X\text{”},\text{“}Y\text{”}),\{(1,2),(2,3),(3,5)\})))$$
$$= \mathrm{r}_0 \cup \mathrm{inject}(\text{“ABOVE”},\mathrm{p})(\{(1,2),(2,3),(3,5)\})$$
$$= \mathrm{r}_0 \cup (\{\},\{(1,2),(2,3),(3,5)\}$$
$$= (\{(1,2),(2,3),(3,5)\},\{(1,2),(2,3),(3,5)\}).$$

We shall not compute $\rho(\mathrm{C2})(\mathrm{r}_0)$ in detail, but note that the join operation involved is the one given as an example after the definition of join.

4. FIXED POINTS ARE INTERPRETATIONS

In this section we prove two theorems. The first states that the fixed point of $\rho(\mathbf{C})$ gives rise to an interpretation of **C**, that is a correspondence between predicates of **C** and relations in which the clauses of **C** are satisfied.

The second theorem shows that $\rho(\mathbf{C})$ is continuous, and thus provides a basis for finding fixed points by repeated applications of $\rho(\mathbf{C})$ to an initial relation sequence.

4.1 *Theorem*

Let **C** be a sequence of clauses. Let **r** be a fixed point of $\rho(\mathbf{C})$. Let $C \in |\mathbf{C}|$. Let $\sigma : V \cup E \to E$ be a function for which $\sigma(\ell) = \ell$ for $\ell \in E$. Suppose that for each literal (p,a) occurring negatively in C, $\sigma(\mathrm{a}) \in$ project $(p,\mathrm{p})(\mathrm{r})$. Let (p',a') be the positive literal of C. Then $\sigma(\mathrm{a}') \in$ project $(p',\mathrm{p})(\mathrm{r})$.

Comment on the meaning of this theorem.

With each predicate p occurring in **C** we associate a relation, project $(p,\mathrm{p})(\mathrm{r}_0)$. We can regard this association relation as an interpretation of the predicate. The theorem states that however we substitute constants for variables in C, if we regard $p(e_1, e_2 \text{ -- } e_k)$ as being satisfied when $(e_1, e_2 \ldots\ldots)$ in r, where r is the associated relation with p, then if all the literals on the left of a clause are satisfied, then that on the right must be satisfied.

Thus in our example, let us consider C2, and let

$$\sigma(\text{“}X\text{”}) = 2, \sigma(\text{“}Y\text{”}) = 3 \text{ and } \sigma(\text{“}Z\text{”}) = 5$$

and let

$$\mathrm{r} = (\{(1,2),(2,3),(3,5)\}, \{(1,2),(2,3),(3,5),(1,3),(2,5),(1,5)\}).$$

Then

$$\sigma((\text{“}X\text{”},\text{“}Y\text{”})) = (2,3) \in \text{project (“ON”,(“ON”, “ABOVE”))}$$
$$= \{(1,2),(2,3),(3,5)\}$$

and likewise sigma((“Y”,“Z”)) $\in$ project(“ABOVE”,(“ON”,“ABOVE”))(r_0) thus the left-hand side of C2 is satisfied, and we find that the right-hand side is satisfied, since

$$\sigma(\text{“}X\text{”},\text{“}Z\text{”}) = (2,5) \in \text{project}(\text{“ABOVE”},(\text{“ON”},\text{“ABOVE”}))(\mathrm{r}_0)\,.$$

Before we can prove (4.1) we need the following two lemmas.

4.2 *Lemma*

Let a be a sequence with $|\mathrm{a}| \subset E \cup V$. Let r be a relation. Let $(\mathrm{v}, r') = \mathrm{attach}(\mathrm{a}, r)$.

Then for any $\sigma : V \cup E \rightarrow E$ for which $e \epsilon E \Rightarrow \sigma(e) = e$

$$\sigma(\mathrm{a}) \epsilon r \Rightarrow \sigma(\mathrm{v}) \epsilon r'.$$

Proof

Suppose $\sigma(\mathrm{a}) \epsilon r$

Let $v_{k'} = a_k$ for some k, k'

$$v_{k'} \text{ of } \sigma(\mathrm{v}) \text{wrt } \mathrm{v} = \sigma(\mathrm{v})_{k'} = \sigma(v_{k'}) = \sigma(\mathrm{a}_k) = \sigma(\mathrm{a})_k .$$

Moreover if $a_k \epsilon E$ then $\sigma(\mathrm{a})_k = a_k$.

Hence $\sigma(\mathrm{a}) \epsilon r'$ from the definition of attach ($\rho 11$).

4.3 *Lemma*

Let $\mathrm{v}, \mathrm{v}', \mathrm{v}''$ be sequences of variables, and let r, r', r'' be relations for which

$$(\mathrm{v}'', r'') = (\mathrm{v}, r) * (\mathrm{v}', r')$$

then if $\sigma : V \cup E \rightarrow E$

$$\sigma(\mathrm{v}) \epsilon r \,\&\, \sigma(\mathrm{v}') \epsilon r' \Rightarrow \sigma(\mathrm{v}'') \epsilon r''.$$

Proof

Let $\nu \epsilon |\mathrm{v}|$ and let $\nu' \epsilon |\mathrm{v}'|$.

$$\nu \text{ of } \sigma(\mathrm{v}'') \text{ wrt } \mathrm{v}'' = \sigma(\mathrm{v}) = \nu \text{ of } \sigma(\mathrm{v}) \text{wrt } \mathrm{v}$$
$$\nu' \text{ of } \sigma(\mathrm{v}'') \text{ wrt } \mathrm{v}'' = \sigma(\mathrm{v}') = \nu' \text{ of } \sigma(\mathrm{v}') \text{ wrt } \mathrm{v}'.$$

Thus $\sigma(\mathrm{v})$ & $\sigma(\mathrm{v}')$ satisfy the requirements to be the t & t' in the definition of *, and so we conclude $\sigma(\mathrm{v}'') \epsilon r''$.

4.4 The proof of theorem 4.1

It follows from Sec. 3.8 that r must be a fixed point for $\rho(C)$ for each $C \epsilon |\mathrm{C}|$.

Now let (p, a) be a literal occurring negatively in C. Then from Lemma 4.2 and $\rho 7$

$$\sigma(\mathrm{a}) \epsilon \text{ project}(p, \mathrm{p})(\mathrm{r}_0) \Rightarrow \sigma(\mathrm{a}) \epsilon \rho(p)(\mathrm{r}_0)$$
$$\Rightarrow \sigma(\mathrm{v}) \epsilon r' \text{ where } (\mathrm{v}, r') = \rho((p, \mathrm{a}))(\mathrm{r}_0)$$

Thus if σ satisfies the preconditions of our theorem (4.1), then for each literal (p,a) occurring negatively in C

$$(v,r') = \rho((p\ a))(r_0) \Rightarrow \sigma(v) \in r' \qquad (4.4.1)$$

Let L'' be a sequence of these literals. We shall show, by structural induction (Burstall, 1969), that if $(v'',r'') = \rho'(L'')$ then $\sigma(v) \in r''$.

Suppose $L'' = ()$. Then $(v'',r'') = q_0 = ((),\{()\})$, by $\rho 4$. Thus $\sigma(v'') = \sigma(()) = \sigma(()) = () \in \{()\}$, so founding our induction. Suppose $L'' = \text{consseq}(L,L')$, and suppose that $\sigma(v') \in r'$, where $\rho(L')(r_0) = (v',r')$.

Now by $\rho 5$,

$$\sigma'(L'')(r_0) = \rho(L)(r_0) * \rho'(L')(r_0) = (v'',r''), \text{ say}$$

Let $\rho(L)(r_0) = (v,r)$, so that $\sigma(v) \in r$ by (4.4.1).
We can conclude from Lemma 4.3 that $\sigma(v'') \in r''$.
Thus if $C = (L,L')$ and $\rho'(L)(r_0) = (v,r)$ then $\sigma(v) \in r$.
Now r_0 is a fixed point of $\rho(C)$, and from $\rho 3$ we see that this implies that

$$\text{inject}\ (p',p)\ (\text{detach}(a',(v,r))) \subset r_0 .$$

Thus

$$\text{project}(p',p)\text{inject}(p',p)\ \text{detach}(a',(v,r))) = \text{detach}(a',(v,r)) \subset \text{project}(p',p)(r)) .$$

Consider $\sigma(a')$

If $a'_k \in E$ then $\sigma(a')_k = a'_k$

If $a'_k \in V$ then a'_k of $\sigma(v)$ wrt $v = \sigma(a'_k) = \sigma(a')_k$.

Hence, by the definition of detach with $\sigma(v)$ playing the part of t, and $\sigma(a')$ playing the part of t', we conclude that $\sigma(a') \in \text{detach}(a',(v,r))$, and so $\sigma(a') \in \text{project}(p',p)(r_0)$.

This concludes the proof.

Let us now return to the proof that $\rho(C)$ is chain continuous. We begin with lemmas showing that attach, detach and join are continuous, the latter in both its arguments, and then prove the chain-continuity of $\rho(C)$ by building up the definitions $\rho 1$-7.

4.5 *Lemma*

If $\{r_\alpha\}$ is a set of relations, and a sequence for which $|a| \subset E \cup V$, then

$$\text{attach}\ (a, \bigcup_\alpha r_\alpha) = \bigcup_\alpha (\text{attach}(a,r_\alpha)).$$

Proof

Let $(v,r') = \text{attach}(a, \bigcup_\alpha r_\alpha)$

$$(v,r'') = \bigcup_\alpha \text{attach}(a\, r_\alpha)$$

Then $r'' = \bigcup_\alpha r''_\alpha$ where $(v\, r''_\alpha) = \text{attach}(a\, r_\alpha)$

Let $t' \in r'$. Then

$$\exists t, t \in \bigcup_\alpha r_\alpha \;\&\; a_k = v_{k'} \Rightarrow a_k \text{ of } t' \text{wrt } a = v_k \text{ of } t \text{ wrt } v$$
$$\&\; a_k \in E \Rightarrow t_k = a_k \,.$$

Suppose $t \in r_\beta$, for some β. Then $t' \in r''_\beta$ and hence $t' \in \bigcup_\alpha r_\alpha$. The converse proof is similar.

4.6 *Lemma*

If $\{(v,r_\alpha)\}$ is a set of labelled relations, with identical labels, and a is an argument sequence, then

$$\text{detach}\,(a, \bigcup_\alpha (v,r_\alpha)) = \bigcup_\alpha \text{detach}\,(a,(v,r_\alpha))\,.$$

The proof is straightforward, and is not included in this paper.

4.7 *Lemma*

If (v,r) is a labelled relation, and $\{(v', r'_\alpha)\}$ is a set of labelled relations, then

$$(v,r) * \bigcup_\alpha (v', r_\alpha) = \bigcup_\alpha ((v,r) * (v', r'_\alpha))\,.$$

Proof

Let $(v'', r'') = (v', r) * (\bigcup_\alpha (v', r'_\alpha))$

Then $|v''| = |v| \cup |v'|$

Let $t'' \in r''$. Then

$$\exists t, t', t \in r \;\&\; t' \in \bigcup_\alpha r'_\alpha \;\&\; \forall v \in v, \forall v' \in v'$$
$$v \text{ of } t'' \text{ wrt } v'' = v \text{ of } t \text{ wrt } v$$
$$v' \text{ of } t'' \text{ wrt } v'' = v \text{ of } t' \text{ wrt } v'\,.$$

Suppose $t' \in r'_\beta$ for some β. Then

$$t \in (v,r) * (v', r'_\beta)$$
and so $t \in \bigcup_\alpha ((v,r) * (v', r'_\alpha))$

and similarly conversely.

We also need the following two results, for which the proof is sufficiently straightforward to be omitted.

4.8 *Lemma*

If $\{r_\alpha\}$ is a set of relation sequences, and p is a predicate, then

$$\text{project}(p,\mathrm{p})\,(\bigcup_\alpha r_\alpha) = \bigcup_\alpha (\text{project}(p,\mathrm{p})(r_\alpha)).$$

4.9 *Lemma*

If $\{r_\alpha\}$ is a set of relations, and p is a predicate, then

$$\text{inject}\,(p,\mathrm{p})\,(\bigcup_\alpha r_\alpha) = \bigcup_\alpha (\text{inject}(p,\mathrm{p})(r_\alpha)).$$

We are now able to prove the following.

4.10 *Theorem*

$\rho(x)$ is a chain-continuous, from $\mathcal{R}$ to $\mathcal{R}$, whether x be a predicate, literal, clause or clause sequence, and if $\mathbf{L}$ is a literal sequence, then $\rho'(\mathbf{L})$ is chain-continuous.

Proof

Let r_α be an ascending chain of members of $\mathcal{R}$, indexed by α in some totally ordered set A.

(i) Let p be a predicate. Then

$$\begin{aligned}
&\rho(p)\,(\bigcup_\alpha r_\alpha) = \text{project}\,(p,\mathrm{p})\,(\bigcup_\alpha \mathrm{r}_\alpha) \\
&= \bigcup_\alpha \text{project}(p,\mathrm{p})(r_\alpha) && \text{(by 4.8)} \\
&= \bigcup_\alpha \rho(p)\mathrm{r}_\alpha. && \text{(by } \rho 7)
\end{aligned}$$

(ii) Let (p,a) be a literal. Then

$$\begin{aligned}
&\rho(p,\mathrm{a})(\bigcup_\alpha \mathrm{r}_\alpha) = \text{attach}(\mathrm{a}\,\rho(\mathrm{p})(\bigcup_\alpha \mathrm{r}_\alpha)) \\
&= \text{attach}\,(\mathrm{a},\bigcup_\alpha \rho(p)(\mathrm{r}_\alpha)) && \text{(by (i))} \\
&= \bigcup_\alpha \text{attach}\,(\mathrm{a},\rho(p)(\mathrm{r}_\alpha)) && \text{(by 4.5)} \\
&= \bigcup_\alpha \rho(p,\mathrm{a})(\mathrm{r}_\alpha). && \text{(by } \rho 6)
\end{aligned}$$

(iii) This section of the proof, which lifts continuity over the join operation, is the root of the restriction to chain-continuity, which arises essentially from the cross-terms generated by join. So, we shall prove, by structural induction, that

$$\rho'(\mathbf{L})(\bigcup_\alpha \mathrm{r}_\alpha) = \bigcup_\alpha (\rho'(\mathbf{L})(\mathrm{r}_\alpha))\,.$$

To found the induction we observe that

$$\rho'(())(\bigcup_\alpha r_\alpha) = q_0 = \bigcup_\alpha \rho'(())(r_\alpha).$$

Now suppose that for some **L**

$$\rho'(\mathbf{L})(\cup r_\alpha) = \cup\, \rho'(\mathbf{L})(r_\alpha)\ .$$

Then, by $\rho 5$

$$\begin{aligned}
&\rho'(\text{consseq}(\mathbf{L},\mathbf{L})(\bigcup_\alpha r_\alpha) = \rho(\mathbf{L})(\bigcup_\alpha r_\alpha) * \rho'(\mathbf{L})(\bigcup_\alpha r_\alpha)\\
&= \bigcup_\alpha \rho(L)(r_\alpha) * \bigcup_\beta \rho'(\mathbf{L})(r_\beta) && \text{(by (ii) and inductive hypothesis, where } \rho \in A)\\
&= \bigcup_\beta (\bigcup_\alpha \rho\,(L)(r_\alpha) * \rho'(\mathbf{L})(r_\beta)) && \text{(by 4.7)}\\
&= \bigcup_\beta \bigcup_\alpha (\rho(L)(r_\alpha) * \rho'(\mathbf{L})(r_\beta)) && \text{(by 4.7 and commutativity of } *)\\
&= \bigcup_\alpha (\rho'(\text{consseq}(L,\mathbf{L}))(r_\alpha)\,) && \text{(by } \rho 5\text{, and the properties of } \cup)
\end{aligned}$$

Since for any α and β in A, either $\alpha \leqslant \beta$ or $\beta \leqslant \alpha$ and so

$$\rho(L)(r_\alpha) * \rho'(\mathbf{L})(r_\beta) \leqslant \rho(L)(r_\alpha) * \rho'(\mathbf{L})(r_\alpha)$$

or

$$\rho(L)(r_\alpha) * \rho'(\mathbf{L})(r_\beta) \leqslant \rho(L)(r_\beta) * \rho'(\mathbf{L})(r_\beta)\ .$$

(iv) Let $(\mathbf{L}, (p',a'))$ be a clause

$$\begin{aligned}
&\rho(\mathbf{L}(p', a'))(\bigcup_\alpha r_\alpha) =\\
&\quad \bigcup_\alpha r_\alpha \bigcup_\alpha \text{inject}(p',p)(\text{detach}(a',\rho'(\mathbf{L})(\bigcup_\alpha r_\alpha)))\\
&= \bigcup_\alpha r_\alpha \cup \bigcup_\alpha (\text{inject}(p',p)(\text{detach}(a',\rho'(\mathbf{L})(r_\alpha))) && \text{(by (iii), 4.6, 4.9)}\\
&= \bigcup_\alpha (r_\alpha \cup \text{inject}(p',p)(\text{detach}(a',\rho'(\mathbf{L})(r_\alpha)))).
\end{aligned}$$

(v) Finally we prove, by structural induction, that $\rho(\mathbf{C})$ is chain-continuous for the clause-sequence **C**.

$$\begin{aligned}
&\rho((C))(\cup r_\alpha) = \rho(C)(\cup r_\alpha) && \text{(by } \rho 1)\\
&= \bigcup_\alpha \rho(C)(r_\alpha) && \text{(by (iv))}\\
&= \bigcup_\alpha \rho((C))(r_\alpha) && \text{(by } \rho 1)
\end{aligned}$$

Suppose now for some **C** that $\rho(\mathbf{C})(\cup r'_\alpha) = \cup\, \rho(\mathbf{C})(r'_\alpha)$ for any ascending chain r'_α.

Then

$$\rho(\text{consseq}(C,\mathbf{C}))(\bigcup_\alpha \mathbf{r}_\alpha)$$
$$= \rho(C)(\rho(\mathbf{C})(\bigcup_\alpha \mathbf{r}_\alpha) \quad \text{(by } \rho 2\text{)}$$
$$= \rho(C)(\bigcup_\alpha \rho(\mathbf{C})(\mathbf{r}_\alpha)) \quad \text{(by (iv)}$$
$$= \bigcup_\alpha \rho(C)(\rho(\mathbf{C})(\mathbf{r}_\alpha))$$

by inductive hypothesis, since $\rho(\mathbf{C})(\mathbf{r}_\alpha)$ is an ascending sequence

$$= \bigcup_\alpha \rho(\text{consseq}(C,\mathbf{C}))(\mathbf{r}_\alpha) \quad \text{(by } \rho 2\text{)}$$

which concludes the proof.

5. APPLICATION OF THE THEORY

The conclusion that we can draw from Secs. 3 and 4, is that given a sequence of clauses, **C** and an initial sequence of relations $\mathbf{r}_0$, that there exists a relation sequence **r** which is defined by

$$r_\infty = \bigcup_n^n (\mathbf{C})(\mathbf{r}_0)$$

and which provides an interpretation of **C** in the sense that for any substitution of constants for variables in a clause C of **C** if the left-hand side of C is satisfied then the right-hand side must be.

Now if we are dealing only with finite relations, the sequence $(\rho^n(\mathbf{C})(\mathbf{r}_0))$ must reach its least upper bound after a finite number of steps, so that, for some $n, r_\infty = \rho^n(C)(r_0)$.

The interesting problems arise when some of the relations concerned are infinite. We are not proposing a treatment of the general case, but shall restrict ourselves to consideration of the case where infinite relations only occur negatively in clauses, and where join only produces finite results.

In order to be able to perform arithmetic computations within our relational system, we need some representation specifying that the set E contains the reals, and that the arithmetic operations, $+ - * /$, are represented as relations on E (there need be no confusion between the use of $*$ at the meta-level for join, and at the object-level for multiplication). It is also convenient to add the distinguished set {T,F} for "true", "false", to E.

We shall use R to denote the set of real numbers.

The technical device used to incorporate the real number operations into the relational system is based on the following definition.

5.1 ***Definition***

Let $E' \subset E$. Let $f : E'^n \to E'$. Then $\mu(f)$ is the relation

$$\{t \mid t_{n+1} = f(t, \ldots t_n), t, \ldots t_n \in E'\}.$$

Thus it is possible to incorporate arithmetic into the system by insisting that P contain the set {"+", "−", "*", "/"} and by associating with each an operator applied to the corresponding function. It is also clear that the arithmetic relations, $<, >, \leqslant, \geqslant$, can be extended to apply to E.

5.2 Implementation of infinite relations

The introduction of infinite relations into the system can only be bought at a cost. The representation of finite relations in a computer raises no problems of principle, although there may be practical difficulties in handling large relations economically. On the other hand there are difficulties involved in representing infinite relations in a way that is effective computationally, and of course there is no guarantee that any clause sequence will give rise to a fixed point in a finite number of iterations. This second difficulty is unavoidable if the system is to have full computational power – it is equivalent to the halting problem.

There are conventionally two methods of representing infinite objects in a computer – as a program or symbolically. (These two are not necessarily distinct; LISP program can be treated as symbolic, although this is seldom done in practice). We propose to restrict the system and to deal only with finite relations, apart from the basic arithmetic ones defined above. This is possible on account of the following.

5.2.1 *Theorem*

Let (v, r) be a finite labelled relation. Let $f : E'^n \rightarrow E'$, $E' \subset E$. Let a be an argument sequence of length $n + 1$, for which

$$i < n+1 \Rightarrow a_i \subset |\mathrm{v}| \cup E$$

then

$$(\mathrm{v},r) * \mathrm{attach}(\mathrm{a}, \mu(f))$$

is finite.

Proof

Let $(\mathrm{v}',r') = \mathrm{attach}(\mathrm{a}, \mu(f))$

Let $(\mathrm{v}'',r'') = (\mathrm{v},r) * (\mathrm{v}',r')$

Let $\theta : r'' \rightarrow r$ be defined by

If $\mathrm{t}'' \in r''$, choose $\theta(\mathrm{t}'') \in r$ s.t.

$v \in |\mathrm{v}| \Rightarrow v$ of t'' wrt $\mathrm{v}'' = v$ of $\theta(\mathrm{t}'')$ wrt v

We shall show that θ is $1-1$.

Suppose $\theta(t'') = \theta(s'') = t$, say, for some $s'', t'' \in r''$
There are two cases to consider.

Case 1

$a_{n+1} \in E \cup \{a, \ldots . a_n\}$. Then $|v'| \subset |v|$
therefore $v'' \in |v''| \Rightarrow v'' \in |v|$ since $|v''| = |v| \cup |v'|$
therefore $\forall v'' \in |v''| \Rightarrow$
v'' of t'' wrt $v'' = v''$ of $\theta(t'')$ wrt $v = v''$ of $\theta(s'')$ wrt v
$= v''$ of s'' wrt v''
therefore $s'' = t''$.

Case 2

$a_{n+2} \in V, a_{n+1} \neq a_i \; i \leqslant n.$
Let $v'_k = a_{n+1}$
Let $t^0 \; s^0 \in \mu(f)$ for which
$v_{k'} = a_k \Rightarrow v_{k'}$ of t' wrt $v' = t^0_k$
$v_{k'}$ of s' wrt $v' = s^0_k$
$a_k \in E \Rightarrow t^0_k = a_k = s^0_k$.

Then for $k \leqslant n$, let $a_k = v_{k'}$. Then $v_{k'} \in |v|$.
$t''_k = v_{k'}$ of t' wrt $v' = v_{k'}$ of t'' wrt v''
$= v_{k'}$ of t, wrt $v = v_{k'}$ of s wrt v
$= v_{k'}$ of s'' wrt $v'' = v_{k'}$ of s' wrt $v' = s^0_k$.
Thus $t^0_k = s^0_k \; k \leqslant n$.
But $t^0_{k+1} = f(t^0_1 \ldots t^0_n) = f(s^0_1, -s^0_n) = s^0_{n+1}$
therefore v'_k of t'' wrt $v'' = v'_k$ of t' wrt $v' = t^0_{n+1}$
$= s^0_{n+1} = v'_k$ of s' wrt $v' = v'_k$ of s'' wrt v''
and, for $v'' \neq v_k, v'' \in |v|$
so v'' of t'' wrt $v'' = v''$ of $\theta(t'')$ wrt $v = v''$ of $\theta(s'')$ wrt v
$= v''$ of s'' wrt v''
therefore $t'' = s''$
Thus θ is $1-1$, hence r'' is finite, since r is finite.

5.2.2 *Theorem*
Let (v, r) be a finite labelled relation. Let r' be relation. Let a be an argument sequence, length(a) = arity(r'). Then

$$(v,r) * \text{attach}(a,r')$$

is finite.

The proof is similar to the preceding theorem, and is omitted.

The import of the above two theorems is that the relations involved in the computation of $\rho(C)$, for some clause C, will be finite if for every literal (p,a) where p is associated with an infinite relation $\mu(f), a, \ldots a_n$ are either constants, or are variables which have already occurred in earlier literals in the clause (arity$(p) = n+1$). Moreover, every literal (p,a) for which p is associated with a non-functional, infinite relation of arity n, then $a, \ldots . a_n$ must either be constant, or have occurred earlier in the clause.

5.3 Free functions

In order to provide some equivalent facility to the data structures of conventional programming languages, let us suppose that there is a set $\{f_{mn}\}$ of functions, $f_{mn} : E^m \rightarrow E$, for which

$$f_{mn}(x, \ldots x_m) = f_{m'n'}(x', \ldots x'_{m'}) \Rightarrow m = m', n = n', x_i = x'_i .$$

The f_{mn} are called free functions on E.

We can associate free functions with predicate symbols, as in the last section. It should be noted that in addition to the uses of these permitted by Theorems 5.2.1 and 5.2.2, it is possible to have a literal (p,a), for which p is associated with f_{mn}, where a_{m+1} is a variable which has occurred earlier in the clause, while a_i, $i \leqslant n$ have not necessarily occurred earlier. This observation does not carry through to the quotient interpretations discussed in the next section.

6. THE TREATMENT OF EQUALITY

Most mechanised logic systems have problems in their treatment of equality. The basic intuitive notion that if entities are equal then they should behave identically when acted on by functions can be expressed by axioms of the sort

$$x = y \Rightarrow f(x) = f(y)$$

which have to be written out for every function named in the system. It is possible to ensure that this substitutivity property of equality is automatically provided by adding the predicate "=" to the set P of predicates, and by modifying the definition of $p(\mathbf{C})$, for some sequence of clauses C to

$$\rho_{eq}(\mathbf{C}) = \rho(\mathbf{C}) \circ \eta(\mathbf{r})$$

where $\eta(\mathbf{r})$ is defined as follows:

Let r_{eq} = "=" of r wrt p
Let r'_{eq} be the reflexive symmetric and transitive closure of r_{eq}.

Then $\eta(\mathbf{r})$ is $\mathbf{r}'$ where

(i) "=" of $\mathbf{r}'$ wrt $\mathbf{p} = r'_{eq}$
(ii) If $p \in |\mathbf{p}|, p \neq$ "=" and let
$r = p$ of $\mathbf{r}$ wrt $\mathbf{p}$. Then
$r' = p$ of $\mathbf{r}'$ wrt $\mathbf{p}$ is defined to be
$r' = \{t' \mid \exists t \in r, \forall i (t'_i, t_i) \in r'_{eq}\}$.

The effect of the above definition can be stated simply by saying that in each cycle of the interactive process of interpreting clauses, we take the equalities that have been deduced, apply the rules of reflexivity, symmetry and transitivity to produce an extended equality relation, and then use this to infer that if two entities are equal, and one is related to some further, then the other must also be related to these entities.

4.6.1 Introducing η is equivalent to introducing equality axioms

In this section we show that if $\mathbf{C}$ is a clause sequence, then any fixed point of $\rho_{eq}(\mathbf{C})$ is a fixed point of $\mathbf{C}'$ where $\mathbf{C}'$ is formed from $\mathbf{C}$ by adjoining equality axioms.

Let $\mathbf{r}_1$ be a fixed point of $\rho_{eq}(\mathbf{C})$, and let us note that for all $\mathbf{r}$, $\eta(\mathbf{r}) \geqslant \mathbf{r}$, from the reflexivity of r'_{eq} in the definition of η.

We can easily see that $\mathbf{r}_1$ is a fixed point of $\rho(\mathbf{C})$ for

$$\rho(\mathbf{C})(\mathbf{r}_1) \leqslant \eta(\rho(\mathbf{C})(\mathbf{r}_1)) = \rho_{eq}(\mathbf{C})(\mathbf{r}_1) = \mathbf{r}_1.$$

But from the definition of ρ

$$\rho(\mathbf{C})(\mathbf{r}_1) \geqslant r_1.$$

Similarly, we can show that $\eta(\mathbf{r}_1) = \mathbf{r}_1$.

6.1.1 *Theorem*

If $\mathbf{C}$ is a clause sequence, and $\mathbf{r}$ is a fixed point of $\rho_{eq}(\mathbf{C})$ then

(i) $\mathbf{r}_1$ is a fixed point of
$\rho(x = y \Rightarrow y = x)$

(ii) $\mathbf{r}_1$ is a fixed point of
$\rho(x = y \,\&\, y = z \Rightarrow x = z)$

(iii) for any $p \in |\mathbf{p}|$, r_1 is a fixed point of
$\rho(x = y \,\&\, p(x_1 \ldots x \ldots x_n) \Rightarrow p(x_1 \ldots y \ldots x_n))$.

We shall omit the proof of (i) and (ii) and only give the proof of (iii). Since q_0 is an identity of $*$, we need to consider

$$t \in \text{detach}((x_1 \ldots y \ldots x_n), \text{attach}((x,y), r_{eq}) * \text{attach}((x_1 \ldots x \ldots x_n), r_p)$$
where r_{eq} = "=" of r_1 wrt p and $r_p = p$ of r_1 wrt p .

We need to show that $t \in r_p$
Now from the definition of detach ($\rho 10$),

$$\exists t' \in \text{attach}((x,y), r_{eq}) * \text{attach}((x, \ldots x \ldots x_n) r_p)$$

for which

$$v \in |v'| \Rightarrow v \text{ of t wrt } (x_1 \ldots y \ldots x_n) = v \text{ of } t' \text{ wrt } v' \qquad \text{(I)}$$

where v' is a sequence without repetitions and $|v'| = \{x_1 \ldots x_n, x, y\}$.
Now, form the definition of $*$ and attach, $\rho 11$ & $\rho 12$,

$$\exists t'', t''', t'' \in r_{eq} \;\&\; t''' \in r_p \quad st.$$
$$v'' \in \{x,y\} \Rightarrow$$
$$v'' \text{ of } t' \text{ wrt } v' = v'' \text{ of } t'' \text{ wrt } (x,y) \qquad \text{(II)}$$
$$v''' \{x, \ldots x \ldots x_n\} \Rightarrow$$
$$v''' \text{ of } t' \text{ wrt } v' = v''' \text{ of } t''' \text{ wrt } (x_1 \ldots x \ldots x_n). \qquad \text{(III)}$$

Let $v \in \{x_1 \ldots \ldots x_n\}$. Then

$$v \text{ of t wrt } (x \ldots y \ldots x_n) = v \text{ of } t' \text{ wrt } v' \qquad \text{from (I).}$$
$$= v \text{ of } t''' \text{ wrt } (x_1 \ldots x \ldots x_n) \qquad \text{from (III).}$$

and

$$y \text{ of t wrt } (x_1 \ldots y \ldots x_n) = y \text{ of } t' \text{ wrt } v' \qquad \text{from (I)}$$
$$= y \text{ of } t'' \text{ wrt } (x,y). \qquad \text{from (II)}$$

Now $\eta(r_1) = r_1$ so that r_{eq} is its own reflexive symmetric & transitive closure, so that it is itself reflexive symmetric and transitive. Thus

$$v \in \{x_1 \ldots x_n\} \Rightarrow (v \text{ of t wrt}(x_1 \ldots y \ldots x_n),$$
$$v \text{ of } t''' \text{ wrt}(x_1 \ldots x \ldots x_n)) \in r_{eq}$$

by the reflexivity of r_{eq} and

$$(y \text{ of t wrt } (x_1 \ldots y \ldots x_n), x \text{ of } t''' \text{ wrt } (x_1 \ldots x, \ldots x_n)) \in r_{eq}$$

since

$$(y \text{ of t wrt}(x_1 \ldots y \ldots x_n), x \text{ of } t''' \text{ wrt } (x_1 \ldots x \ldots x_n))$$
$$= (y \text{ of } t'' \text{ wrt}(x,y), x \text{ of } t' \text{ wrt } v) = (y \text{ of } t'' \text{ wrt}(x,y), x \text{ of } t'' \text{ wrt } (x)$$
$$= (t_\alpha t_\beta) \text{ say, from the definition of } *. (\rho 12)$$

Now since r_{eq} is symmetric and

$$t'' = (x \text{ of } t'' \text{ wrt}(x,y), y \text{ of } t'' \text{wrt}(x\ y))$$

it follows that $(t_\alpha t_\beta) \in r_{eq}$
But $t''' \in r_p$, and we have shown that for each i, $(t_i\ t_i''') \in r_{eq}$.
Therefore, from the definition of η, and the fact that $\eta(\mathfrak{r}_1) = \mathfrak{r}_1$ it follows that $t \in r_p$.

6.2 A discussion of equality

Theorem 6.1.1 shows that it is possible to make the interpretation system behave as though the equality axioms were explicitly present.

From a practical point of view this method of treating equality suffers from a major disadvantage in that the application of the equality rules results in a large expansion of the relations. It seems likely that the obvious ploy of using r_{eq} to define equivalence classes of entities, and only recording relationships between canonical members of these classes, would be theoretically sound, but I have not completed a proof that this is so.

7. DISCUSSION

In this paper we have shown how it is possible to use certain combinators on relations to produce an interpretation of a class of clauses (Horn Clauses) in predicate logic. The work was inspired by a particular view of the task of writing certain kinds of program, but has not yet given rise to a system implemented on a digital computer, although some initial studies have been made. The mathematical apparatus used is hardly novel – perhaps my most direct debt is to D. Park (1969).

REFERENCES

Batani, G. and Meloni, H. (1973). *Interpreteur du Language de Programmation* PROLOG. Marseille: Université d'Aix-Marseille.

Burstall, R. M. (1969). Proving properties of programs by structural induction. *Computer Journal*, 12, 41–48.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Comm. Ass. Comp. Mach.*, 13, 377–387.

Elcock, E. W., Foster, J. M., Gray, P. M. D., McGregor, J. J. and Murray, A. M. (1971). ABSET: A programming language based on sets; motivation and examples. *Machine Intelligence 6*, pp. 467–492 (eds. Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press.

Iverson, K. E. (1962). *A Programming Language.* New York: Wiley.

Kowalski, R. (1973). Predicate logic as a programming language. *DCL Memo No. 70.* Edinburgh: Dept. of Artificial Intelligence, University of Edinburgh.

Landin, P. J. (1966). The next 700 programming languages. *Comm. Ass. Comp. Mach.*, 9, 157–166.

Park, D. (1969). Fixpoint induction and proofs of program properties. In *Machine Intelligence 5*, pp. 59–77 (eds. Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press.

Warren, D. (1975). Users guide to DEC System 10 PROLOG. *DAI Internal Memo.* Edinburgh: Dept. of Artificial Intelligence, University of Edinburgh.