Notre Dame Journal of Formal Logic
Volume X, Number 1, January 1969

THEORY OF PROCEDURES
I. SIMPLE CONDITIONALS

M. K. RENNIE

Introduction. The aim of this paper is to set up the beginnings of a logical theory of procedures, one of the major uses of which will be as a theory of computation. Much of the initial portion of the paper consists of a generalization and re-setting of the theory developed in McCarthy [6]. (For the translation of McCarthy's theory into practice, see McCarthy et al. [7].) In [9], Thiele has developed a similar theory, oriented especially toward algorithmic languages such as are used in digital computer programming. It is not to be thought that a theory of *computability,* such as is provided by Turing machines or Markov algorithms[1] is in any way a theory of computation or a general theory of procedures. In fact the latter is used repeatedly in an informal fashion whenever one constructs Turing machines or Markov algorithms.

In section 1, we describe informally the nature and structure of procedures, giving examples of procedures and of the way in which they may be diagrammed and symbolized. The morphology of a formal theory for procedures is set out in section 2, and in section 3 and section 4 interpretations are given for the various parts of this formalism. In section 5, a theorem on the eliminability of propositional operators within the theory is proved. Then in section 6 we prove a normal form theorem, which yields a sufficient condition for complete axiomatizations of procedure theory at this first level, where only simple conditionals occur.

1. *The Structure of Procedures.* The treatment of procedures begins from a different point from the theory of Turing machines or Markov algorithms. In these theories, a *space* is assumed at the very start; for Turing machines the space is the tape which the machine uses for reading, shifting or printing, and for Markov algorithms the space is the alphabet over which the algorithm operates. A Turing machine is assessed in terms of its effect on the tape, in particular the 'number' which it writes on the tape as its final output, and Markov algorithms are assessed in terms of their effect on words drawn from their alphabet, in particular their terminal production. In procedure theory we can assess procedures purely

Received July 20, 1967

in terms of their internal structure, and not in terms of their effect on any space external to them.[2]

Without pretending to give a formal definition, we can say that a procedure consists of a coherent, interconnected set of steps, where each step is either an *operational* step or a *decision* step or a *structural* step. Examples of each sort of step may be found in the homely realm of office procedures. Some of the operational steps are: banking in cash, sending out invoices, sending out account overdue notices. Some of the decision steps are: checking whether a correspondent is already mentioned in our files, checking whether an account is more than 60 days overdue, determining whether an amount ordered is held in stock. Structural steps are not quite so easy to find; some examples are: taking a file to the credit manager, returning unfilled orders to the orders file. Structural steps are usually carried out on the basis of the result of decision steps, and their effect is always internal to the procedure.

Inside the quadruples with which Turing machines are defined (see e.g. Hermes [4] p. 31) we may also find examples of each sort of step. A typical quadruple has the form

$$q_i s_j o_k q_{i'}$$

where q_i and $q_{i'}$ denote internal states of the machine, s_j denotes one of the symbols which may appear on the tape, and o_k denotes a machine operation, which may be one of the symbols (for printing) or a shifting operation or a halting operation. In such a quadruple, $q_i s_j$ is a decision, since the quadruple is active only if the machine is in state q_i and scanning symbol s_j: o_k is an operation, which is carried out on the basis of the decision: and $q_{i'}$ causes a structural step, since it causes the machine to shift into state $q_{i'}$, which has no effect on anything external to the procedure.

As a sample of the way in which we may diagram a procedure, showing the interconnection of the three basic sorts of steps, consider Figure 1.

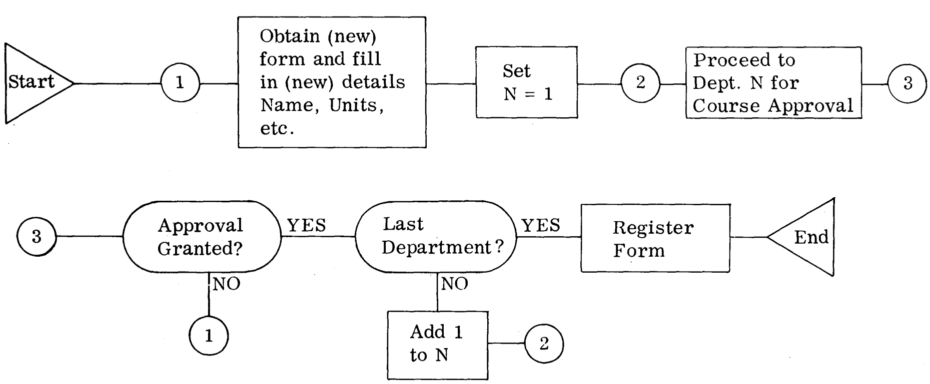

Figure 1. Enrollment Procedure

Each operational step is represented by a rectangular box, each decision step by an ovoid box and the two[3] possible paths leading away from it, and the structural steps are represented by the passage from one occurrence of a numbered connector[4] (in the small circles) to another.

Passage through the diagram is always from left to right, and from top to bottom; when we come to a numbered connector with nothing following, we pass to the other occurrence of a connector, with the same number, from which we may progress either towards the right or towards the bottom. Via the structural steps so defined, this procedure is iterative, e.g. if any department does not grant course approval then, via connector number 1, the procedure is iterated with a new form and new details.

In this first level of the theory of procedures, we will not treat iterative procedures of this sort. The present theory is constructed for the purposes of logical analysis of the properties of the simple conditional effect on operational steps given by decision steps, and in it there will be a minimum of structural steps. Indeed, there will be no explicit structural steps, and the only structural steps will be those implicitly involved in passing from one step in the procedure to another, or in ignoring a step if the condition for carrying out that step is not satisfied.

One semi-philosophical point is required before we proceed to the logic of procedures, and this concerns the expression of operations within operation steps. In general, we have to distinguish between an action itself, the doing of the action, the proposition that the action is done, a command or imperative that the action should be done, and the effect or result of the action's being done. Of these, we may rule out any consideration of effects or results, since we are operating in abstraction from any particular class of actions. We will use small Greek letters α, β, γ, δ, α' .... to refer to actions, and these will be interpreted as gerundive phrases such as "Your shutting the door" or "the adding of x to y". We will write '$D\alpha$' for the proposition that the action is done, or alternatively for the command "Do α!", often written as '$!\alpha$'. Although '$D\alpha$' and '$!\alpha$' must be clearly distinguished in general, they can be conflated for the purposes of procedure theory. This is because one must assume, in setting up a procedure, that just those actions which the procedure licenses or exhorts, and only those actions, will be done. Again, exhortation and licensing, or obligation and permission, coincide in procedure theory. Nothing is optional—there will be no steps saying to the office worker or the computer "You may do α here if you feel like it". As a further remark on the equivalence of '$D\alpha$' and '$!\alpha$' for the purposes of procedure theory, we note that a procedure will have precisely the same structure, as far as the relation of operational steps and decision steps is concerned, when presented in the indicative mood as when presented in the imperative mood. If we were describing an office procedure to an outsider, we would use the indicative mood; if we were setting it out to the office staff who have to operate under it we would use the imperative mood, yet it would be precisely the same procedure in each case. One often thinks in terms of the imperative mood when addressing a computer; however computers have a somewhat naturalistic ethic, and provided there are no technical difficulties, one has $!\alpha \equiv D\alpha$ for a computer—they do what they are told and only what they are told.

We will also quite often conflate α and $D\alpha$ in procedure theory, and the operator 'D' will often play little more part than a role as a punctuation sign. This conflation will only provide us with grammatical worries at an interpreted level; for purely manipulative reasons we will frequently conjoin and disjoin propositions and actions, where for ease of interpretation we should conjoin or disjoin propositions with propositions to the effect that an action is done.

2. *Formalism for the first level of procedure theory.*

Primitives

1. (a) The letters p, q, r, s, p', . . . designated as propositional variables.

   (b) 0 and 1 designated as propositional constants.

   (c) $-$, &, $\vee$, $\supset$, $\equiv$ designated as propositional operators.

2. The letters α, β, γ, δ, α', . . . designated as action variables.

3. (a) The letters D and E

   (b) The symbols C^i for $i \geqslant 2$. (C^2 will mostly be written as C).

4. (,) for use as parentheses.

Formation Rules

1. (a) A propositional variable or constant standing alone is a well-formed proposition (wfp).

   (b) If P and Q are wfps, then $(\overline{P})$, $(P \& Q)$, $(P \vee Q)$, $(P \supset Q)$, $(P \equiv Q)$ are wfps. (Propositional operators are used autonomously in the metalanguage).

2. If A is an action variable then (DA) is a well-formed procedure expression at the first level (wfpe1).

3. If $P_1 \ldots P_n$ are wfps, and $Y_1 \ldots Y_n$ are wfpe1's, then $(C^n P_1 Y_1 P_2 Y_2 \ldots P_n Y_n)$ is a wfpe1, for $n \geqslant 2$.

4. If X and Y are wfpe1's, then (YX) is a wfpe1 (where juxtaposition in the metalanguage denotes juxtaposition in the object language).

5. If X and Y are wfpe1's, then XEY is a well-formed formula of procedure theory at level 1 (wff1).

When writing wfpe1's and wff1's, we will conventionally drop outside parentheses and others not needed to prevent ambiguity. Some sample wfpe1's, with some parentheses dropped, are

(i) $((D\alpha)(D\beta))(D\gamma)$

(ii) $(C^3 p(D\alpha)q(D\beta)r(D\gamma))$

(iii) $(D\alpha)(Cp((D\beta)(D\gamma))q(D\delta))(D\delta')$

(iv) $CpD\beta 1D\gamma$

(v) $Cp(D\alpha)q(Cr(D\beta)s(D\gamma))$

3. *The interpretation of wfpel's*. Each wfpel can be interpreted as a description (or prescription) of a procedure. The (syntactical) order of occurrence of wfpel's within a larger wfpel determines the order in which each part is to be carried out. Thus (i) interprets as the simple unconditional procedure in which first α, then β and then γ is done, or is to be done. This can be represented by a simple flow-chart as in Figure 2.

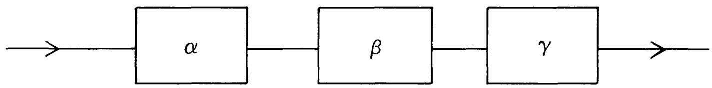

Figure 2.

Concatenation interprets, implicitly, as "and then", and this operator has intuitively the same formal properties as concatenation, namely associativity but not commutativity. If we take equivalence of procedures in an intuitive sense of 'having the same result', then $(D\alpha)(D\beta)$ will not in general be equivalent to $(D\beta)(D\alpha)$. To borrow an example of Russell's, let α be putting on one's socks and β be putting on one's boots. Then clearly doing α then doing β, *i.e.* $(D\alpha)(D\beta)$, does not have the same result as doing β then doing α, *i.e.* $(D\beta)(D\alpha)$. For actions within a Turing machine, we could let α be printing a '+', and β be shifting right one square. Then $(D\alpha)(D\beta)$ will cause a '+' to be printed on the square to the left of that on which $(D\beta)(D\alpha)$ will cause '+' to be printed.

However, a sequence of simple D-expressions does associate; in fact this is implicit in the lack of parentheses in Figure 2. Hence we are justified in using juxtaposition (or, for formal syntactical purposes, concatenation) to express the operation of direct passage from one step to another in a procedure.

The symbols C^i are our symbols for conditional expressions within the procedure. They do not correspond directly with the ovoid boxes for representing decision steps on flow-charts, nor are they to be confused with material implication, for which we are using the ordinary Peano-Russellian '$\supset$'. The interpretation of a C^i-expression is formally identical to the interpretation of the *if . . . then . . . else . . .* conditional statement-form in ALGOL[5]. The general form $C^nP_1Y_1P_2Y_2 \ldots P_nY_n$ is interpreted as the procedure of doing Y_1 if P_1 is true, or if P_1 is not true then doing Y_2 if P_2 is true, etc. In general, P_j .&. $\bigwedge_{i<j} P_i$ is the condition under which Y_j will be done according to the conditional procedure expressed by the wfpel: $C^nP_1Y_1 \ldots P_nY_n$.

Under this interpretation, the question arises as to effect of such a conditional procedure if none of the P_i, for $1 \leqslant i \leqslant n$ are true. In this case we say that the procedure is *not evaluable* and generally we will confine our attention to *evaluable* procedures. We will apply these properties ambiguously to procedures and the wfpel's which express them: a C^n-expression $C^n P_1 Y_1 \ldots P_n Y_n$ is defined to be evaluable if $\bigvee_{i=1}^{n} P_i$ is true. Of the evaluable C'-expressions, there will be a subclass of *logically evaluable* C^n-expressions, which are those for which $\bigvee_{i=1}^{n} P_i$ is logically true. We will call a whole wfpel evaluable if and only if all the C^n-expressions contained in it are evaluable, and logically evaluable, or simply *logical,* if and only if all the C^n-expressions contained in it are logically evaluable. Any procedure expressed by a wfpel containing only D-expressions, such as our sample wfpel (i), will be logical and *a fortiori* evaluable.

To facilitate the expression of logical procedures, we will introduce the symbol 'θ', whose interpretation is, roughly, 'otherwise'. Just as 'otherwise' expresses a definite condition only in context, so we cannot give 'θ' an explicit definition but instead we define it contextually. We require that 'θ' can only occur in the P_n-place of a C^n-expression, and the contextual definition is

$$C^n P_1 Y_1 \ldots P_{n-1} Y_{n-1} \theta Y_n =_{Df} C^n P_1 Y_1 \ldots P_{n-1} Y_{n-1} \overline{\left(\bigvee_{i=1}^{n-1} P_i \right)} Y_n$$

It immediately follows that any C^n-expression containing 'θ' is logically evaluable, since $\bigvee_{i=1}^{n-1} P_i \ .\mathrm{v}.\ \overline{\bigvee_{i=1}^{n-1} P_i}$ is logically true.

In fact, 'θ' could be replaced by '1', and an equivalent procedure would result. In either case, Y_n will be done if and only if none of $Y_1 \ldots Y_{n-1}$ are done. Of course, any C^n-expression containing a '1' for any P_i, in particular P_n, will be logically evaluable, since 1 is a logical truth, and if p is a logical truth then so is $p \vee q$. The point of using the 'θ' symbol is that it gives the condition under which Y_n is done, which we would not have if we used '1' in place of 'θ'.

With these remarks, we may now turn to a consideration of the interpretation of the wfpel's (ii)-(v). The wfpel (ii) interprets as the conditional procedure of doing α if p is true, doing β if q is true and p is not true, and doing γ if r is true and neither p nor q is true. As it stands, this procedure is not logical, since $p \vee q \vee r$ is not a logical truth. It would become logical if 'r' were replaced by 'θ', or if an extra action were added, resulting in $C^4 pD\alpha qD\beta rD\gamma\theta D\delta$. The flow-chart for this procedure is shown in Figure 3.

The portion above the dotted line is the original procedure, and the whole flow-chart is the procedure as made logical by the addition of '$\theta D\delta$'. Notice that we need add no decision step for 'θ': the effect is the same as if '1' were written in place of 'θ' and it is clearly otiose to ask if 1 is true. It is not otiose in general to ask if $\overline{(p \vee q \vee r)}$ is true, but given that we are only in a position to ask the question if neither p, nor q, nor r is true then the question becomes otiose.

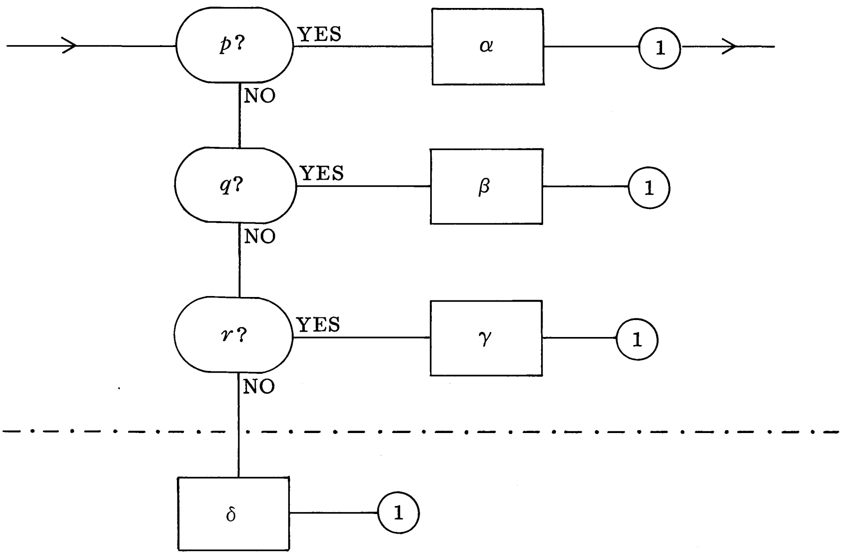

Figure 3.

The wfpel's (iii), (iv) and (v) may be interpreted in a similar fashion, and represented by a flow-chart. (iv) is logical, but (iii) and (v) are not logical as they stand. (v) illustrates the point that the Y_i's in a C^n-expression do not have to be simple D-expressions or concatenations of them, but may themselves be C^m-expressions. In terms of the procedure expressed by such a wfpel, this means that some decision steps may be followed by further decision steps rather than by an operational step.

4. *The interpretation of wffl's.* We now turn to the interpretation of wffl's, which are formed by infixing an 'E' between two wfpel's. The intuitive interpretation of 'E' is as a proposition-forming operator on procedures; thus 'XEY' is read as "Procedure X is equivalent to procedure Y". Now ". . . is equivalent to . . ." is an explicandum which requires explication: the first step towards an explicatum[6] is to spell the explicandum out into ". . . has the same results under the same conditions as. . .", and this in turn may be spelled out to ". . . causes (or prescribes) the same actions to be done, in the same order, for the same truth-values of the conditional propositions, as . . ." This latter form motivates the following precise definition of *evaluation* and equivalence.

Firstly, we give *evaluation rules* for wfpel's in primitive notation, *i.e.* with all occurrences of 'θ' removed by contextual definition. These are set up so that any C^n-expression $C^n P_1 Y_1 \ldots P_n Y_n$ which is not logically evaluable is treated as if it were $C^{n+1} P_1 Y_1 \ldots P_n Y_n \theta(D\alpha_0)$ where α_0 is the null action, *i.e.* the action of doing nothing. A wfpel is evaluated with respect to a standard truth-tabular matrix, formed from all the distinct propositional variables appearing in P-places within the wfpel. If the wfpel contains no C^n-expressions, it has a one-row evaluation. For each row of the matrix, we evaluate a wfpel according to the prescriptions:

1. The evaluation of (XY) is the evaluation of X followed by the evaluation of Y.
2. The evaluation of $C^nP_1Y_1 \ldots P_nY_n$ is the evaluation of the first Y_i, for $1 \leqslant i \leqslant n$, for which P_i is true for the particular row of the matrix, and is nil if there is no such P_i. If any P_i is a compound proposition, an ordinary truth-tabular calculation will be needed to determine its truth-value.
3. The evaluation of DA is carried out by writing A in the row of the matrix beneath its occurrence in the wfpel.

These rules are obviously effective, in that at any stage during the evaluation of a wfpel, the rule to be used is fully determined, and whenever the evaluation of a compound wfpel requires the evaluation of another wfpel, the second wfpel is shorter syntactically than the first and so the procedure must terminate. The evaluation procedure is itself a recursive procedure (or algorithm), of a type which cannot be expressed by a wfpel (but which can be expressed at a more advanced level of procedure theory). Some sample evaluations are:

$D\alpha\ D\beta\ D\gamma$		
α	β	γ

p	$(CpD\alpha\bar{p}D\beta)D\gamma$		
1	α		γ
0		β	γ

p	q	$CpD\alpha qD\beta(\bar{p}\ \&\ \bar{q})D\gamma$		
1	1	α		
1	0	α		
0	1		β	
0	0			γ

p	q	$CpD\alpha\bar{p}(CqD\beta\bar{q}D\gamma)$		
1	1	α		
1	0	α		
0	1		β	
0	0			γ

Now the evaluation of a wfpel sets out what actions are done (or prescribed) under all logically possible states of the conditional propositions within the wfpel. In evaluation rule 3, it is required that the action variable A should be written in the row of the evaluation beneath its occurrence in the wfpel: hence the order of occurrence of the actions on any row of the evaluation will be the same as the order in which the actions are done within the procedure. Hence we may use the evaluation of wfpel's to define their equivalence.

We will define XEY to be true if and only if each row of the evaluation of X contains the same action variables in the same order as the corresponding row of the evaluation of Y, where X and Y are evaluated with respect to a matrix formed for the distinct propositional variables occurring either in X or in Y. This definition is the explicatum of our original explicandum "X is an equivalent procedure to Y". An example of a true wffl is

$$(C^3pD\alpha qD\beta\theta D\gamma)E(CpD\alpha\theta CqD\beta\theta D\gamma),$$

as may be seen by checking the last two sample evaluations.

In the following sections we will assert that various wffl's are true without supplying any proof: in each case the proof is a routine matter using the evaluation rules of this section.

5. *The eliminability of propositional operators.* In this section we prove that for every logical wfpel there exists an equivalent wfpel which contains no propositional operators.

For the proof of this proposition, we note two preliminary results. Firstly, we note that every C^n-expression is equivalent to a wfpel containing only C^2-expressions. This follows by a generalization of

$$(C^3 pD\alpha qD\beta rD\gamma)E(C^2 pD\alpha\theta(C^2 qD\beta\theta D\gamma)),$$

which may be readily established by a truth-tabular evaluation. In general

$$(C^n P_1 Y_1 \ldots P_n Y_n)EC^2 P_1 Y_1 \theta C^2 P_2 Y_2 \theta \ldots \theta C^2 P_{n-1} Y_{n-1} P_n Y_n)$$

where the right-hand wfpel contains n-1 C^2-expressions.

Secondly, we note that every logical C^2-expression of the form $C^2 P_1 Y_1 P_2 Y_2$ is equivalent to $C^2 P_1 Y_1 \theta Y_2$. (It does not follow from this that $\theta = P_2$). Putting these two facts together gives the result that every logical C^n-expression is equivalent to an expression all of whose well-formed parts are of the form $C^2 P_1 Y_1 \theta Y_2$.

Hence, to establish the required result that every logical wfpe is equivalent to a wfpel containing no propositional operators, it suffices to show that all the propositional operators in a wfpe of the form $C^2 P_1 Y_1 \theta Y_2$ may be eliminated, and to show that it suffices to show that we may eliminate the '$-$' from $C^2 \bar{p} Y_1 \theta Y_2$ and the '&' from $C^2(p \& q) Y_1 \theta Y_2$. These latter eliminations are carried out using the equivalences

$$(C^2 \bar{p}\, Y_1 \theta\, Y_2)E(C^2 p\, Y_2 \theta\, Y_1)$$

$$(C^2(p \& q)\, Y_1 \theta\, Y_2)E(Cp(Cq\, Y_1 \theta\, Y_2)\theta\, Y_2),$$

and hence the required result is established.[7]

We do not retain this result if we insist that the wfpel should be *in primitive notation* and contain no propositional operators. We can however assert, on the basis of the proof just given, that for every wfpel there is a wfpel in primitive notation containing no *dyadic* propositional operators. This follows since in each case where we expand a 'θ' into primitive notation, we will introduce a negation operator, but no dyadic operators.

6. *Normal Forms and a sufficient condition for complete axiomatizations.* In this section we define explicit-conditional normal forms (ecnf's), and perfect explicit-conditional normal forms (pecnf's), and use them to derive a sufficient condition for complete axiomatizations of this level of procedure theory.

A wfpel is defined to be in *explicit-conditional normal form* iff either (a) it contains no C^n-expressions or (b) if it contains any C^n-expressions, then it contains just one, and this is the main operator of the wfpel, so that it is of the form $C^n Q_1 Y_1 \ldots Q_n Y_n$; and furthermore in this case each

Y_i for $1 \leqslant i \leqslant n$ (all of which are D-expressions or concatenations of them) is done if Q_i is true.

A wfpel is defined to be in *perfect explicit-conditional normal form* if it is in ecnf and moreover each Q_i for $1 \leqslant i \leqslant n$ has the form of a state-description (see [2] p. 9) in all the propositional variables occurring in the wfpel.

We require to prove that for every wfpel there is an equivalent wfpel which is in pecnf (and hence in ecnf). The first result we need is that for every C^n-expression of the form $C^n P_1 Y_1 \ldots P_n Y_n$ there is an equivalent C^n-expression of the form $C^n Q Y \ldots Q_n Y_n$, in the evaluation of which each Y_i is evaluated if the corresponding Q_i is true. The rule for construction is: for each i, $1 \leqslant i \leqslant n$, put $\underset{n-1}{\underline{Q_i = P_i}}$.&. $\bigwedge\limits_{j<i} \overline{P_j}$. In particular, $Q_1 = P_1$ and $Q_n = P_n$.&. $\bigwedge\limits_{j<n} \overline{P_j} = P_n$.&. $\overline{\bigvee\limits_{j=1}^{n-1} P_j}$ by DeMorgan's Law, so that if $P_n = \theta$ so does Q_n.

To show that $(C^n P_1 Y_1 \ldots P_n Y_n) E (C^n Q_1 Y_1 \ldots Q_n Y_n)$ we use an appropriate generalization of $(CpDaqD\beta)E(CpD\alpha(q \,\&\, \overline{p})D\beta)$, which may be established in truth-tabular fashion. To show that for $1 \leqslant i \leqslant n$, Y_i is evaluated if Q_i is true, we see that this will be so if

$$(\forall j)(1 \leqslant j < i \;.\!\supset\!.\; Q_i \supset \overline{Q}_j),$$

since if this holds then if Q_i is true, no Y_j for $1 \leqslant j < i$ will be evaluated, and so Y_i will be evaluated. Now by definition

$$Q_i \supset \bigwedge_{j<i} \overline{P}_j$$

and also

$$\overline{P}_j \supset \overline{Q}_j$$

and hence

$$Q_i \supset \bigwedge_{j<i} \overline{Q}_j$$

and hence the required condition holds. We call this the condition of explicitness, which we may express using the Scheffer stroke function thus

$$(\forall i)(\forall j)(i < n \;.\&.\; j < n \;.\&.\; i \neq j \;:\supset:\; Q_i | Q_j).$$

Whenever this holds we will write $C^n Q_1 Y_1 \ldots Q_n Y_n$ as $C_e^n Q_1 Y_1 \ldots Q_n Y_n$, the subscript '$e$' indicating that the conditional wfpel is explicit. Since $p|\overline{p}$, we always have $C_e^2 P_1 Y_1 \theta Y_2$ as a particular case of explicit conditional expressions.

Hence, since for any C^n-expression there is an equivalent C^n-expression, if we now show how any wfpel may be transformed into an equivalent C^n-expression if it contains any C^m-expression, then we will have given a procedure for placing any wff into ecnf.

Theorem: *For any wfpel there is an equivalent wfpel containing at most one C^n-expression where that C^n is the main operator.*

Proof: The proof is by induction and the basis established immediately since DA contains no C^n-expression if A is an action variable. Next, we make the inductive hypothesis that each Y_i contains at most one C^m-expression, and consider $C^n P_1 Y_1 \ldots P_n Y_n$. Let Y_j, $1 \leqslant j \leqslant n$, be the first Y_i in this C^n-expression to contain a C^m-expression (if there is no such Y_j, then clearly there is nothing to prove). Let $Y_j = C^m R_1 Z_1 \ldots R_m Z_m$, for from the inductive hypothesis, if Y_j contains a C^m-expression then that C^m is its main operator. Now consider the C^r-expression, where $r = n - 1 + m$:

$$C^r P_1 Y_1 \ldots (P_j \,\&\, R_1) Z_1 (P_j \,\&\, R_2) Z_2 \ldots (P_j \,\&\, R_m) Z_m \;\ldots P_n Y_n;$$

this C^r-expression results from the C^n-expression by substituting $(P_j \,\&\, R_1) Z_1 \ldots (P_j \,\&\, R_m) Z_m$ for $P_j Y_j$. We require to show that this expression is equivalent to $C^n P_1 Y_1 \ldots P_j (C^m R_1 Z_1 \ldots R_m Z_m) \ldots P_n Y_n$. Now so far as the pairs $P_i Y_i$, for $i \neq j$, are concerned, the C^r-expression is identical to the C^n-expression, so we need consider only the pairs occurring in the jth to $(j+m-1)$th positions of the C^r-expression, and show this portion equivalent to the jth pair of the C^n-expression. The required equivalence holds here because of an appropriate generalization of the equivalence

$$(C^3 pD\alpha q(CrD\beta sD\gamma)p'D\alpha')E(C^4 pD\alpha(q \,\&\, r)D\beta(q \,\&\, s)D\gamma p'D\alpha'),$$

and hence we have shown that the C^r-expression is equivalent to the C^n-expression and contains one less C^m-expression. By repeating the formation of such C^r-expressions, we may remove all C^m-expressions from the C^n-expression, except the final C^r-expression which constitutes the main operator. Hence this inductive step is established.

Next we prove two lemmas, to be used in establishing the inductive step for concatenation. The first lemma states that if a C_e-expression $C_e^n Q_1 Y_1 \ldots Q_n Y_n$ satisfies the *distinctness condition*

$$(\forall i)(\forall j)(1 \leqslant i, j \leqslant n \;.\&.\; i \neq j \;{:}\supset{:}\; Y_i \neq Y_j),$$

then $Q_i Y_i$ pairs may be permuted within that C_e-expression: *i.e.* given the distinctness condition then

$$(C_e^n Q_1 Y_1 \ldots Q_i Y_i \ldots Q_j Y_j \ldots Q_n Y_n)E(C_e^n Q_1 Y_1 \ldots Q_j Y_j \ldots Q_i Y_i \ldots Q_n Y_n)$$

This is immediate once we observe that in a C_e-expression satisfying the distinctness condition, Y_i will be evaluated if *and only if* Q_i is true, so that the position in which $Q_i Y_i$ appears is immaterial to the evaluation of the expression.

The second lemma states that for any C_e^n-expression, there is an equivalent C_e-expression satisfying the distinctness condition. To prove this, let $Y_i = Y_j$ for some $i < j$ in $C_e^n Q_1 Y_1 \ldots Q_n Y_n$. Then we have

$$(C_e^n Q_1 Y_1 \ldots Q_n Y_n)E(C_e^{n-1} Q_1 Y_1 \ldots (Q_i \vee Q_j) Y_i \ldots Q_{j-1} Y_{j-1} Q_{j+1} Y_{j+1} \ldots Q_n Y_n),$$

and on the right-hand side of this equivalence, Y_j no longer appears and so does not cause the expression to fail the distinctness condition. If $n = 2$, then we use the equivalence $(C_e^2 Q_1 Y_1 Q_2 Y_1)E(C_e^2 1 Y_1 0 D\alpha')$, where α' is an action-variable appearing nowhere else in the expression.

Now we may make the inductive hypothesis, that X and Y are in ecnf, and consider (XY). We may assume that both X and Y contain just one C-expression each, since we can use the equivalence $(D\alpha)E(C1D\alpha OD\alpha')$, where α' is an active-variable occurring nowhere else in X or Y, to introduce artificially a C-expression if one of them does not already contain one. (If neither X nor Y contains a C-expression, there is nothing to prove). Further, we may assume that each of these C-expressions is explicit, and then by the second lemma we may assume that each of them satisfies the distinctness condition as well. Under these assumptions, we set

$$X = C_e^n Q_1 Y_1 \ldots Q_1 Y_1, \quad Y = C_e^m Q_1' Y_1' \ldots Q_m' Y_m'$$

Now, for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, put

$$Q_{ij} = (Q_i \,\&\, Q_j'), \quad Y_{ij} = (Y_i Y_j')$$

Now consider the $C^{n\times m}$-expression

$$C^{n\times m} Q_{11}Y_{11}Q_{12}Y_{12} \ldots Q_{1m} Y_{1m} Q_{21} Y_{21} \ldots Q_{nm} Y_{nm} \ .$$

According to the inductive hypothesis, each Y_{ij} contains no C-expressions, and hence the $C^{n\times m}$-expression obeys the required conditions. We require to show that it is equivalent to (XY) to establish the inductive step.

Let us observe firstly that the $C^{n\times m}$-expression is explicit if both the C^n and the C^m-expressions are explicit. To show that the $C^{n\times m}$-expression is explicit, we have to show that the distinct Q_{ij}'s are incompatible, *i.e.*,

$$i \neq k \ .\mathrm{v}.\ j \neq l \ :\supset:\ Q_{ij} \,|Q_{kl}$$

By definition, this becomes $i \neq k \ .\mathrm{v}.\ j \neq 1 \ :\supset:\ Q_i \,\&\, Q_j' \ .|.\ Q_k \,\&\, Q_l'$

By hypothesis $\qquad i \neq k \ .\supset.\ Q_i \,|Q_k$

and $\qquad j \neq 1 \ .\supset.\ Q_j' \,|Q_l'$,

hence $\qquad i \neq k \ .\mathrm{v}.\ j \neq 1 \ .\supset.\ Q_i \,|Q_k \ .\mathrm{v}.\ Q_j' \,|Q_l'$.

Also, $\qquad p|q \ .\mathrm{v}.\ r|s \ :\supset:\ p \,\&\, r \ .|.\ q \,\&\, s$

is a tautology, and by substitution we have

$$Q_i \,|Q_k \ .\mathrm{v}.\ Q_j' \,|Q_l' \ :\supset:\ Q_i \ \&\, Q_j' \ .|.\ Q_k \ \&\, Q_l' \ ,$$

and hence by syllogism we have the required result.

Further, the $C^{n\times m}$-expression will also satisfy the distinctness condition if both X and Y do. Hence in the $C^{n\times m}$-expression, the order of writing the $Q_{ij}\, Y_{ij}$ pairs is immaterial, and Y_{ij} will be evaluated if and only if Q_{ij} is true, that is, iff Q_i and Q_j' are true. That is, Y_i and then Y_j' will be evaluated iff Q_i and Q_j' are true. But this is precisely the case for (XY), which is

$$((C_e^n Q_1 Y_1 \ldots Q_n Y_n)(C_e^m Q_1' Y_1' \ldots Q_m' Y_m')),$$

and hence the required equivalence holds.

Hence we have completed the proof that for every wfpel there is an equivalent wfpel in ecnf. In order to perfect an ecnf we simply note the result from propositional calculus that every truth-function,[8] $\phi(p_1, \ldots, p_n)$ say, may be expressed as a disjunction $\bigvee_i s_i$, where each s_i is a state-description in the n propositional variables $p_1, \ldots, p_n$. In any ecnf, we replace each propositional function by such a disjunction, and then we apply repeatedly the equivalence[9] $(C_e(p \vee q)D\alpha\theta D\beta)E(C_e pD\alpha qD\alpha\theta D\beta)$ in order to derive a pecnf. For any logical wfpel, its pecnf will be a C^{2^n}-expression, where n is the number of propositional variables occurring in the wfpel. (Appendix A contains a full-scale worked example of reduction of a wfpel to pecnf.)

Perfect explicit-conditional normal forms mirror the evaluation of a wfpel in much the same way as perfect disjunctive normal forms in propositional calculus mirror the ordinary truth-tables. Both pecnf and an evaluation set out the actions which are done under the conditions given by a state-description in the propositional variables concerned. From this it follows that if XEY, then the pecnf of X is *identical* (within the order of the terms, and this may be made unequivocal by a suitable lexicographic convention) to the pecnf of Y.

Hence any formal system whose wff's are our wffl's, and whose axioms and rules of derivation are sufficient to ensure that E is an RST relation and that the pecnf metatheorem holds with respect to the system, will be complete with respect to the set of true wffl's as determined by the evaluation procedure. For, if XEY, then the pecnf's are identical. We then express the (identical) pecnfs by a wff of the formal system, A say. Then AEA is a theorem of the system if E is reflexive. Then we apply the pecnf procedure to each side of this equivalence, but in reverse, so that we transform A on one side into a wff, X', expressing X, and on the other side into a wff, Y', expressing Y. If E is RST, then $X'EY'$ will be a theorem of the system, and hence the system is complete.

This then is our sufficient condition for completeness of axiom sets for procedure theory at level 1. Such an axiom set will also be decidable, for similar reasons. Any number of axiom sets will satisfy the condition: the simplest but least elegant way is to take as axiom schemata the general principles such as

$$(C_e^n Q_1 Y_1 \ldots Q_n Y_n)(C_e^m Q_1' Y_1' \ldots Q_m' Y_m')E(C^{n\times m} Q_{11} Y_{11} \ldots Q_{nm} Y_{nm})$$

which were needed in our informal pecnf proof. Or else we can take restricted versions of these, and prove the more general schemata using induction in the metalanguage. The choice of axiom sets depends upon our purpose, in particular whether the formal system is to be used for deriving theorems or for metatheoretic investigations. Since we are going to do neither of these in this paper, we leave open the choice of any particular set of axioms.

APPENDIX A

Take the wfpel:

$\Psi = C(pCqD\ \alpha\theta D\beta)(\theta C(rCqD\,\alpha\theta D\gamma)(\theta(CpD\,\alpha\theta D\,\delta)(CsD\,\alpha'\theta D\,\delta')))$.

Write Ψ as

$$\Psi = Cp(CqD\alpha\ \theta D\beta)\ \theta(CrA\ \theta B)$$

Remove θ's, and we get

$$\Psi = Cp\ (CqD\alpha\overline{q}D\beta)\overline{p}(CrA\overline{r}B)\ .$$

Now we have an instance of the case in the first inductive step. We "distribute" p and $\overline{p}$ according to the formula there, and derive

$$\Psi E\ C^4(p\ \&\ q)\ D\alpha(p\ \&\ \overline{q})\ D\beta(\overline{p}\ \&\ \overline{r})B$$

Now $B = (CpD\alpha\ \theta D\delta)\ (CsD\,\alpha'\theta D\,\delta') = (CpD\,\alpha\overline{p}D\delta)\ (CsD\,\alpha'\overline{s}D\delta')$

This is now an instance of the case in the second inductive step. We put

$$Y_{11} = (D\,\alpha D\alpha'),\ Y_{12} = (D\alpha D\delta'),\ Y_{21} = (D\,\delta D\alpha'),\ Y_{22} = (D\,\delta D\,\delta')$$

and

$$Q_{11} = p\ \&\ s\ ,\ Q_{12} = (p\ \&\ \overline{s}),\ Q_{21} = (\overline{p}\ \&\ s),\ Q_{22} = (\overline{p}\ \&\ \overline{s})$$

Then B is equivalent to

$$C^{2\times 2}Q_{11}\,Y_{11}\,Q_{12}\,Y_{12}\,Q_{21}\,Y_{21}\,Q_{22}\,Y_{22}$$

i.e.

$$C^4(p\ \&\ s)\,(D\,\alpha D\,\alpha')\,(p\ \&\ \overline{s})\,(D\ \alpha D\,\delta')\,(\overline{p}\ \&\ s)\,(D\,\delta D\,\alpha')\,(\overline{p}\ \&\ \overline{s})\,(D\,\delta D\,\delta')$$

Now we put this for B in Ψ, and $A = CqD\,\alpha\theta\,D\gamma$, and then we use the method of the first inductive step again. Expanding over A first we have

$$\Psi\ E\,C^5(p\ \&\ q)D\,\alpha(p\ \&\ \overline{q})\ D\beta(\overline{p}\ \&\ r\ \&\ q)\ D\alpha(\overline{p}\ \&\ r\ \&\ \overline{q})D\gamma(\overline{p}\ \&\ \overline{r})B$$

Then by expanding over B

$$\Psi E\,C^8(p\ \&\ q)\ D\alpha(p\ \&\ \overline{q})D\beta(\overline{p}\ \&\ r\ \&\ q)\ D\alpha(\overline{p}\ \&\ r\ \&\ \overline{q})D\gamma(\overline{p}\ \&\ \overline{r}\ \&\ p\ \&\ s)\,(D\,\alpha D\alpha')$$
$$(\overline{p}\ \&\ \overline{r}\ \&\ p\ \&\ \overline{s})\,(D\ \alpha D\delta')\,(\overline{p}\ \&\ \overline{r}\ \&\ \overline{p}\ \&\ s)\,(D\ \delta\ D\alpha')\,(\overline{p}\ \&\ \overline{r}\ \&\ \overline{p}\ \&\ \overline{s})\,(D\delta\ D\delta')$$

Now we remove a couple of terms with contradictory expressions in P-place, and get

$$\Psi\,E\,C^6(p\ \&\ q)\ D\alpha(p\ \&\ \overline{q})\ D\beta(\overline{p}\ \&\ r\ \ \&\ q)\ D\alpha(\overline{p}\ \&\ r\ \&\ \overline{q})D\gamma(\overline{p}\ \&\ \overline{r}\ \&\ s)\,(D\ \delta\ D\alpha')\,(\overline{p}\ \&\ \overline{r}\ \&\ \overline{s})$$
$$(D\delta\ D\delta')$$

We now have obtained an ecnf for Ψ, since it may be checked that the wfpel is an explicit conditional. The perfection process yields

$$\begin{array}{ll} C^{16} & (p\ \&\ q\ \&\ r\ \&\ s)\ \ (D\ \alpha) \\ & (p\ \&\ q\ \&\ r\ \&\ \overline{s})\ \ (D\ \alpha) \\ & (p\ \&\ q\ \&\ \overline{r}\ \&\ s)\ \ (D\ \alpha) \\ & (p\ \&\ q\ \&\ \overline{r}\ \&\ \overline{s})\ \ (D\ \alpha) \\ & (p\ \&\ \overline{q}\ \&\ r\ \&\ s)\ \ (D\ \beta) \end{array}$$

$$
\begin{array}{ll}
(p \ \& \ \overline{q} \ \& \ r \ \& \ \overline{s}) & (D\beta) \\
(p \ \& \ \overline{q} \ \& \ \overline{r} \ \& \ s) & (D\beta) \\
(p \ \& \ \overline{q} \ \& \ \overline{r} \ \& \ \overline{s}) & (D\beta) \\
(\overline{p} \ \& \ q \ \& \ r \ \& \ s) & (D\alpha) \\
(\overline{p} \ \& \ q \ \& \ r \ \& \ \overline{s}) & (D\alpha) \\
(\overline{p} \ \& \ q \ \& \ \overline{r} \ \& \ s) & (D\delta D\delta') \\
(\overline{p} \ \& \ q \ \& \ \overline{r} \ \& \ \overline{s}) & (D\delta D\delta') \\
(\overline{p} \ \& \ \overline{q} \ \& \ r \ \& \ s) & (D\gamma) \\
(\overline{p} \ \& \ \overline{q} \ \& \ r \ \& \ \overline{s}) & (D\gamma) \\
(\overline{p} \ \& \ \overline{q} \ \& \ \overline{r} \ \& \ s) & (D\delta D\alpha') \\
(\overline{p} \ \& \ \overline{q} \ \& \ \overline{r} \ \& \ \overline{s}) & (D\delta D\delta')
\end{array}
$$

after a little re-arrangement, and this is the pecnf in systematic form. The wfpel Ψ is logical, containing as it does a liberal sprinkling of 'θ' 's, and hence its pecnf is a $C^{2^n} = C^{2^4} = C^{16}$ -expression.

NOTES

1. For descriptions of Turing machines and Markov algorithms, see respectively Hermes [4] and Markov [5].

2. That is, we will do this until we explicitly assume a result space, when we will be able to assess procedures in terms of their external effects on this space. Result spaces will not be introduced in this paper.

3. In general, there is no reason why there should be only two possible branches from a decision. E.g. we could ask "For what month is the procedure being carried out?", and this would lead to a twelve-fold branch. However, all such decisions could be carried out by means of a series of two-branch decisions, in the sample case by a series of decisions beginning "Is this for January?" and ending "Is this for November?", and two-branch decisions are obviously more amenable to treatment by a logic of a propositional style.

4. For a general description of the use of this sort of diagram ('flowchart') in computer programming, and for some history of computing *lagniappe*, see Goldstine and von Neumann [3].

5. For which, see McCracken [8], or any commercial programming manual for ALGOL.

6. These technical terms are, of course, Carnap's. See, e.g. Carnap [1] pp. 1-8.

7. The normal form of a wfpel which results after all eliminations of propositional operators have been carried out corresponds to McCarthy's "canonical form for generalized Boolean functions", McCarthy [6] p. 56. McCarthy does not define a normal form corresponding to our explicit conditional normal form in section (4), and does not discuss the question of completeness of his axiom set.

8. Provided it is not a contradiction, in which case it may be removed in virtue of $(C^3 pD\alpha OD\beta\theta D\gamma)E(CpD\alpha\theta D\gamma)$, and similar equivalences and generalizations of them.

9. This equivalence is stated in terms of an explicit conditional, rather than any C-expression, since for example the following two wfpe1's are *not* equivalent

$$C^3(p \vee q) D\, \alpha q D \beta\, \theta D\, \gamma,\ C^4 p D\, \alpha q D\, \beta q D\, \alpha \theta D\, \gamma$$

For C_e-expressions, it is immaterial where the two terms resulting from a separation of a disjunction are placed, and this facilitates the systematic arrangement of the terms in a pecnf.

REFERENCES

[1] Carnap, R., *Logical Foundations of Probability*, Routledge and Kegan Paul, London, 1950.

[2] Carnap, R., *Meaning and Necessity*, University of Chicago Press, Chicago, 1947.

[3] Goldstine, H. H., and J. von Neumann, "Planning and Coding Problems for an Electronic Computing Instrument" in *John von Neumann Collected Works*, A. Taub (Ed.), Vol. V, pp. 80-235.

[4] Hermes, H., *Enumerability, Decidability, Computability*, Springer-Verlag, Berlin, 1965.

[5] Markov, A. A., *Theory of Algorithms*, Academy of Sciences of the USSR Translation, Volume 42, 1961.

[6] McCarthy, J., "A Basis for a Mathematical Theory of Computation", in Braffort & Hirschberg (Eds.), *Computer Programming and Formal Systems*, North-Holland, Amsterdam, 1963.

[7] McCarthy, J., (et al.), *LISP 1.5 Programmer's Manual*, M.I.T. Press, Massachusetts, 1962.

[8] McCracken, D. D., *A Guide to ALGOL Programming*, John Wiley & Sons, New York, 1962.

[9] Thiele, H., *Wissenschaftstheoretische Untersuchungen in algorithmischen Sprachen I*, VEB Deutscher Verlag der Wissenschaften, Berlin, 1966.

The University of Auckland
Auckland, New Zealand