A COMPUTER PROGRAM FOR DETERMINING MATRIX MODELS OF PROPOSITIONAL CALCULI

Ross T. Brady

La Trobe University
Dpt of Philosophy
Bundoora Victoria AUSTRALIA 3083

I. Introduction

(a) *General*

Let us consider propositional calculi which are axiomatically characterised by a set of axioms and rules, as in Harrop [3], pp. 273-4. Sets of matrices are often used to provide models for these propositional calculi and these models are used to prove the consistency of the propositional calculus, to show that certain wffs are non-theses of the propositional calculus and to prove the independence of an axiom from the rest of the axioms and the rules of the propositional calculus. This paper provides some techniques for developing a computer program to determine such matrix models. Hence I will be concerned with the practical matter of finding sets of matrices and I will be considering the sorts of propositional calculus one might be likely to be dealing with when setting up systems of modal logic, systems of entailment, etc., rather than considering propositional calculi in general.

The propositional variables will be taken as p, q, r, s, t, ... The propositional calculus must have a rule of uniform substitution or use axiom schemes. The matrix method requires this because one must be able to substitute all the matrix values for each propositional variable.

The consistency shown by the matrix model is usually in all three senses, that is, absolute consistency, consistency in the sense of Post and consistency with respect to negation (c.f. [4], pp. 137-138). If one prescribes that not all matrix values are designated then, if the set of matrices form a model of a propo-

sitional calculus, then the propositional calculus will be consistent in the sense of Post and absolutely consistent. If the propositional calculus has a negation $\sim$ and, for each designated value d, the d — position of the $\sim$ — matrix contains an undesignated value, then the propositional calculus will also be consistent with respect to the negation $\sim$. However, if a designated value d is not taken by any valid wff, then that d — position of the $\sim$ — matrix need not contain an undesignated value. One should also note that the propositional calculus can be consistent with respect to negation without this condition on its negation matrix holding.

As in [4], pp. 132-136, independence of an axiom of a propositional calculus can be shown by using a matrix model. The independent axiom must be invalid in the model while the rest of the axioms are valid and the rules preserve validity. In a similar way, matrix models can also show that a given wff is not provable from the axioms and rules of a propositional calculus. This enables one to pin down a propositional calculus in that one can show that certain wffs are provable and others not provable. This may be the best one can do if there is no decision procedure available.

If C is a monadic connective, D is a dyadic connective, ([1]) and v_1 and v_2 are values in a matrix model then let us call the value in the v_1 — position of the C — matrix, C (v_1), and the value in the (v_1, v_2) — position of the D — matrix, D (v_1, v_2).

(b) *Assumptions about Matrix Models.*

Henceforth, let us assume that each matrix model has at least one designated value, to ensure that there are some valid wffs, and at least one undesignated value, to ensure consistency in the sense of Post. Also let us disregard matrix models with a designated value which is not taken by any valid wff because these models would have the same set of valid wffs as a

([1]) I consider only monadic and dyadic connectives in this paper because I am only concerned with practical examples of propositional calculi. However, most of what I have to say using monadic and dyadic connectives will generalise to n-adic connectives.

model with the appropriate designated value undesignated and because the disregarding of such matrix models serves to cut down the number of matrix models that need to be considered for the purposes of showing consistency and independence.

The number of matrix models that need to be considered can also be reduced by using two further methods of showing that two matrix models have the same set of valid wffs, assuming the wffs are the same for each model. These two methods are due to Kalicki, [5], pp. 176-7.

(a) If it is possible to identify some groups of the designated values of a matrix model M or some groups of its undesignated values without creating any inconsistency in the matrices, then M and M' have the same set of valid wffs, where M' is obtained from M by the above identification and by the subsequent omission of the unnecessarily repeated rows and columns of M. In this case, the matrix model M is disregarded in favour of the simpler model M'.

(b) If matrix models M and M' are such that

(i) M' has n values and M has $(n + k)$ alues, where $k > 0$ and M and M' have n values in common,

(ii) v is a designated value of M iff v is a designated value of M' or v is one of the extra k values of M,

(iii) if v_1 and v_2 are both values of M' and C and D are monadic and dyadic connectives respectively, then C (v_1) in model M is the same value as in model M' and D (v_1, v_2) in model M is the same value as in model M',

(iv) if v_1 or v_2 is a value not in M' and C and D are monadic and dyadic connectives respectively, then C (v_1) and D (v_1, v_2) are both values not in M',

then M and M' have the same set of valid wffs. In this case, the matrix model M is disregarded in favour of the simpler model M'. An example of two such matrix models occurs in [2]. Halldén introduces a 3 — valued logic with the values, truth, falsity and nonsense, where both truth and nonsense are designated. His logic has the properties of M with 3 values and 2 — valued propositional calculus has the properties of M'. Hence, for any wff A of the 2 — valued propositional calculus,

A is valid in Halldén's 3 — valued logic iff A is valid in the 2 — valued propositional calculus.

(c) *Weak and Strong Models.*

In order for a set of matrices to be a matrix model M of a propositional calculus P, as well as all the axioms of P being valid in M, the rules of P must preserve validity in M. Harrop, in [3], p. 275, calls such a model M, a «finite weak model of P», and such rules as «weakly satisfied in M». If, for each rule of P, if one associates a value with each wff-scheme in the rule and if the premises of the rule all take designated values then the conclusion of the rule also takes a designated value, then Harrop calls such a model, a «finite strong model of P», and such rules as «strongly satisfied in the model». It is clear that every finite strong model is a finite weak model. Harrop also indicates that every derived rule of P is weakly (strongly) satisfied in every weak (strong) model of P. For the purpose of finding matrix models, I will be mainly concerned with strong models because these are more intuitive models in that they mirror the intended interpretation more accurately than a model which is not a strong one. Also, the conditions imposed by a rule on the matrices of a strong model are more clear cut and facilitate the finding of such models by using a computer. It is only when one cannot find an appropriate strong model that one need look for models that are not strong.

To obtain an example of a condition imposed on a matrix of a strong model by a rule, consider a propositional calculus with a connective $\rightarrow$ such that $\vdash A$, $\vdash A \rightarrow B \Rightarrow \vdash B$ is a rule. Hence, for any designated value d and d_1, and for any value v, such that $\rightarrow (d, v) = d_1$, the value v will be designated. Therefore, if v is undesignated, then d_1 would be undesignated, given that d remains designated and $\rightarrow (d, v) = d_1$. That is, for all designated values d and all undesignated values u, $\rightarrow (d, u)$ is undesignated.

Another example can be obtained by considering a propositional calculus with a connective & and a rule $\vdash A$, $\vdash B \Rightarrow$

$\vdash$ A & B. Then, for all designated values d_1 and d_2, & (d_1, d_2) will be designated.

(d) *A Result about Rules Strongly Satisfied in a Model.*

If M is a strong matrix model of a propositional calculus P, then the condition imposed on M by a rule $\vdash \mathcal{A}_1, \ldots, \vdash \mathcal{A}_n \Rightarrow \vdash \mathcal{B}$ of P is the same as that imposed by an axiom A_1 & ... & $A_n \supset$ B, where

(i) each distinct wff-scheme in $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{B}$ is replaced by a distinct propositional variable to obtain $A_1, \ldots, A_n$, B, respectively,

(ii) for all designated values d and d_1, and undesignated values u and u_1, & (d, d_1) is designated, & (d, u) is undesignated, & (u, d) is undesignated, & (u, u_1) is undesignated, $\supset$ (d, d_1) is designated, $\supset$ (d, u) is undesignated, $\supset$ (u, d) is designated and $\supset$ (u, u_1) is designated,

and (iii) if connectives satisfying the property (ii) are not already present in P, the connectives & and $\supset$, satisfying (ii) would have to be added to P.

This is so, because A_1 & ... & $A_n \supset$ B is valid iff, for all assignments of values to the propositional variables in $A_1, \ldots, A_n$, B, if $A_1, \ldots, A_n$ all take designated values then B takes a designated value, i.e. for all assignments of values to the wff-schemes in $\mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{B}$, if $\mathcal{A}_1, \ldots, \mathcal{A}_n$ all take designated values then $\mathcal{B}$ takes a designated value, i.e. $\vdash \mathcal{A}_1, \ldots, \vdash \mathcal{A}_n \Rightarrow \vdash \mathcal{B}$ is strongly satisfied in the strong model M.

(e) *Equivalence of Values of a Strong Model.*

Let a propositional calculus P have a connective $\rightarrow$ and have the following theses and rule:

(i) $p \rightarrow p$

(ii) $p \rightarrow q \rightarrow . \; q \rightarrow r \rightarrow . \; p \rightarrow r$

(iii) $p \rightarrow q \rightarrow . \; r \rightarrow p \rightarrow . \; r \rightarrow q$

(iv) For all monadic connectives C,

either $p \to q \to . \; Cp \to Cq$

or $p \to q \to . \; Cq \to Cp$.

(v) For all dyadic connectives D, excepting $\to$,

either $p \to q \to . \; p \; D \; r \to q \; D \; r$,

or $p \to q \to . \; q \; D \; r \to p \; D \; r$,

and either $p \to q \to . \; r \; D \; p \to r \; D \; q$

or $p \to q \to . \; r \; D \; q \to r \; D \; p$.

(vi) $\vdash A, \; \vdash A \to B \Rightarrow \vdash B$.

[The wffs in (ii), (iii), (iv) and (v) can also be in rule form,]

Let the values v_1 and v_2 of a strong model M of P be called equivalent, symbolised as $v_1 \leftrightarrow v_2$, iff $\to (v_1, v_2)$ and $\to (v_2, v_1)$ are both designated. Since $p \to p$ is valid in M, for all values v, $\to (v, v)$ is designated and $\leftrightarrow$ is a reflexive relation. Clearly, by the definition, $\leftrightarrow$ is a symmetric relation. Since $p \to q \to . \; q \to r \to . \; p \to r$ is valid in M and M is a strong model, for all values v_1, v_2 and v_3, if $\to (v_1, v_2)$ is designated then, if $\to (v_2, v_3)$ is designated then $\to (v_1, v_3)$ is designated, and also, if $\to (v_3, v_2)$ is designated then, if $\to (v_2, v_1)$ is designated then $\to (v_3, v_1)$ is designated. Hence $\leftrightarrow$ is a transitive relation and an equivalence relation. The set of values of M can be divided up to form equivalence classes using the equivalence relation $\leftrightarrow$. Note that a designated value cannot be equivalent to an undesignated value.

Let A $(p_1, \ldots, p_n)$ be a wff and assign values $v_1, \ldots, v_n$ to each of its propositional variables $p_1, \ldots, p_n$, respectively. Let the wff A take a value v'. If any of the values $v_1, \ldots, v_n$ are replaced by an equivalent value then the value taken by A will be equivalent to v'. The proof of this is by induction the number of connectives in A. Using (ii), if $v_1 \leftrightarrow v_2$ then $\to (v_1, v_3) \leftrightarrow \to (v_2, v_3)$, and, using (iii), if $v_1 \leftrightarrow v_2$ then $\to (v_3, v_1) \leftrightarrow \to (v_3, v_2)$. Using (iv), if $v_1 \leftrightarrow v_2$ then $C(v_1) \leftrightarrow C(v_2)$, for each monadic connective C. Using (v), if $v_1 \leftrightarrow v_2$ then $D(v_1, v_3) \leftrightarrow D(v_2, v_3)$ and $D(v_3, v_1) \leftrightarrow D(v_3, v_2)$, for all dyadic connectives D.

Hence, one can form a strong model M', by replacing each equivalence class of values of M by a single value and de-

signating those values that correspond to equivalence classes of designated values, such that each matrix of M' can be consistently determined and such that M and M' have the same set of valid wffs. Hence, without loss of generality, one can disregard such matrix models M in favour of such matrix models M'. One can, then assume that no two values are equivalent and also that if $A \rightarrow B$ and $B \rightarrow A$ are both valid then, for each valuation of their variables, A and B both take the same value.

Let P be a propositional calculus with theses (i) to (v) and rule (vi), as above. Let M' be a strong matrix model of P with no two values equivalent. Let the values of M' be 1, 2, ..., n. Then, if $p \,\&\, p \rightarrow p^2$ and $p \rightarrow p \,\&\, p$ are theses of P, & (v, v) = v, for all values v. If $p \rightarrow \sim \sim p$ and $\sim \sim p \rightarrow p$ are both theses of P, then $\sim(\sim(v)) = v$, for all values v, and there is a one-one correspondence between values v and their negations $\sim(v)$. If $\vdash A$, $\vdash \sim A \Rightarrow \vdash B$ is a (derived) rule then, for all designated values d, $\sim(d)$ is undesignated. If $p \vee p \rightarrow p$ is a thesis of P, then, for all undesignated values u, v (u, u) is undesignated. If, as well as $p \vee p \rightarrow p$, $p \vee \sim p$ is also a thesis of P, then, for all undesignated values u, $\sim(u) \neq u$. (3)

Thus, combining these results, for any propositional calculus with theses (i) to (v) above, the rule (vi), the theses $p \rightarrow \sim \sim p$, $\sim \sim p \rightarrow p$, $p \vee p \rightarrow p$, $p \vee \sim p$, and the rule $\vdash A$, $\vdash \sim A \Rightarrow \vdash B$, the only matrices that need to be tested for strong models of this system have an even number of values, 2n, and a negation $\sim$ such that $\sim(v) = 2n + 1 - v$, where $v = 1, \ldots, 2n$. The designated values are 1, ..., d, where $1 \leqq d \leqq n$.

The number of possible matrix models can usually be reduced using permutations of the designated values and also of the undesignated values, such that there is no change in the set of valid wffs. If $\sim(v) = 2n + 1 - v$, where the values v range through 1, ..., 2n, then any permutation of the designated values completely determines the appropriate permutation of the undesigned values which are negations of the permuted designated values.

(2) Assume that the appropriate connectives are in P.

(3) This result is due to G. Fitzhardinge.

(f) *An Example.*

Let us now consider an example to illustrate the foregoing theory. Angell, in [1], axiomatises a propositional calculus P_{A1} and shows that it is consistent by using a 4-valued strong matrix model. Consistency needs to be shown for this system because it cannot be interpreted in the 2-valued propositional calculus, in any obvious way.

The axioms and rules for P_{A1} are given on pp. 328-9 of [1], and the theorems and derived rules that I will need are given on pp. 329-334. Angell shows that all the theses of 2-valued propositional calculus are derivable in P_{A1}. Let us now determine what conditions can be placed on strong matrix models of P_{A1} (assuming that there is at least one) by the axioms, rules, theses and derived rules of P_{A1}.

Since $\vdash A,\ \vdash B \Rightarrow \vdash A \,\&\, B$ [4] is a rule, for all designated values d_1 and d_2, $\&\,(d_1, d_2)$ is designated. Since $\vdash A,\ \vdash A \supset B \Rightarrow \vdash B$ is a derived rule, for all designated values d and all undesignated values u, $\supset$ (d, u) is undesignated. Since $p \,\&\, q \supset p$ is a theorem, for all undesignated values u and all values v, $\&\,(u, v)$ is undesignated. Since $q \,\&\, p \supset p \,\&\, q$ is a theorem, for all values v and all undesignated values u, $\&\,(v, u)$ is undesignated. Since $\vdash A,\ \vdash \sim A \Rightarrow \vdash B$ is a derived rule, for all designated values d, $\sim$ (d) is undesignated.

Theses (i) to (v) and rule (vi) of the previous section all hold for $\rightarrow$ and hence one can assume that no two values in the matrix model are equivalent. Also, since $\vdash A,\ \vdash A \rightarrow B \Rightarrow \vdash B$, for all designated values d and all undesignated values u, $\rightarrow$ (d, u) is undesignated. Since $\vdash p \rightarrow \sim\sim p$ and $\vdash \sim\sim p \rightarrow p$, $\sim(\sim(v)) = v$, for all values v. Since $\vdash (p \rightarrow q) \rightarrow (\sim q \rightarrow \sim p)$ and $\vdash (\sim q \rightarrow \sim p) \rightarrow (p \rightarrow q)$, $\rightarrow (v_1, v_2) = \rightarrow (\sim(v_2), \sim(v_1))$, for all values v_1 and v_2. Since $\vdash p \,\&\, q \rightarrow q \,\&\, p$ and $\vdash q \,\&\, p \rightarrow p \,\&\, q$, $\&\,(v_1, v_2) = \&\,(v_2, v_1)$, for all values v_1 and v_2.

Since $\vdash p \rightarrow p$, $\rightarrow$ (v, v) is designated, for all values v. Since $\vdash \sim(p \rightarrow \sim p)$, $\sim(\rightarrow(v, \sim(v)))$ is designated and hence $\rightarrow$ (v, $\sim$ (v)) is undesignated and takes a value u such that $\sim$ (u)

[4] I will use '$\sim$' for Angell's '—' and '&' for Angell's '.'.

is designated, for all values v.

There is no way of using the fact that all the theses of 2-valued propositional calculus are derivable in P_{A1} to show that, for all undesignated values u, $\sim$ (u) is designated. The reason is that there are matrix models of 2-valued propositional calculus where there are undesignated values u such that $\sim$ (u) is undesignated. Such a matrix model is the product matrix model of the two-valued model of propositional calculus with itself, as in [6], pp. 115-6. Here, only the value '1' is designated while the values '2', '3' and '4' are undesignated, $\sim$ (2) = 3 and $\sim$ (3) = 2.

Let us consider 4-valued matrices with 2 values designated. This seems the next best after 2-valued matrices with 1 value designated, which cannot be used to form a model of P_{A1}. Let the values be '1', '2', '3' and '4' and let '1' and '2' be designated. Then the $\sim$ -matrix must be:

$\sim$	
*1	4
*2	3
3	2
4	1

Hence, for all undesignated values u, $\sim$ (u) is designated and $\sim$ (v) = 5 — v, for all values v. Also $\rightarrow (v_1, v_2) = \rightarrow (5 - v_2, 5 - v_1)$ for all values v_1 and v_2, which means there is symmetry across the cross-diagonal of the $\rightarrow$ — matrix. Also $\rightarrow$ (v, 5 — v) is undesignated for all values v, which means that the values along the cross-diagonal of the $\rightarrow$ — matrix are all undesignated.

Such matrix models of P_{A1} will have the following form, using 'd' or 'u' to show that a matrix place is occupied by a

$\sim$		&	1	2	**3**	4	$\rightarrow$	1	2	**3**	4
*1	4	*1	d	d	u	u	*1	d		u	u
*2	3	*2	d	d	u	u	*2		d	u	u
3	2	3	u	u	u	u	3		u	d	
4	1	4	u	u	u	u	4	u			d

designated value or by an undesignated value, respectively. There is symmetry of values across the main diagonal of the &-matrix and symmetry of values across the cross-diagonal of the $\rightarrow$ — matrix.

Next I will consider various ways of filling in 'd' 's and 'u' 's into the vacant positions of the $\rightarrow$ — matrix.

a) Let $\rightarrow$ (4, 2) be undesignated. Then, by symmetry, $\rightarrow$ (3, 1) is undesignated. Hence, for all undesignated values u and designated values d, $\rightarrow$ (u, d) is undesignated. Consider Angell's axiom 7, i.e. $(p \rightarrow q) \rightarrow \sim (p \,\&\, \sim q)$, with p taking an undesignated value u and with q taking a designated value d. Then $\sim (p \,\&\, \sim q)$ takes a designated value while $p \rightarrow q$ takes an undesignated value. Hence $(p \rightarrow q) \rightarrow \sim (p \,\&\, \sim q)$ takes an undesignated value and such a set of matrices cannot form a model of P_{A1}. Hence $\sim$ (4, 2) must be a designated value.

(b) Let $\rightarrow$ (4, 2) be designated. Then, by symmetry, $\rightarrow$ (3, 1) is designated.

(i) Let $\rightarrow$ (2, 1) be designated. Consider axiom 1, i.e. $(q \rightarrow r) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$, with p taking the value 4, q taking the value 2 and r taking the value 1. Since $\rightarrow$ (2, 1) is designated, $\rightarrow$ ($\rightarrow$ (4, 2), $\rightarrow$ (4, 1)) must be designated. Since $\rightarrow$ (4, 2) is designated, $\rightarrow$ (4, 1) is designated. But $\rightarrow$ (4, 1) is undesignated and hence $\rightarrow$ (2,1) must be undesignated. By symmetry, $\rightarrow$ (4,3) is undesignated.

(ii) Let $\rightarrow$ (1, 2) be designated. Consider axiom 1, i.e. $(q \rightarrow r) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$, with p taking the value 3, q taking the value 1 and r taking the value 2. Since $\rightarrow$ (1, 2) is designated, $\rightarrow$ ($\rightarrow$ (3, 1), $\rightarrow$ (3, 2)) is designated. Since $\rightarrow$ (3, 1) is designated $\rightarrow$ (3, 2) is designated. But $\rightarrow$ (3, 2) is undesignated and hence $\rightarrow$ (1, 2) is undesignated. By symmetry, $\rightarrow$ (3, 4) is undesignated.

The $\rightarrow$ — matrix can now be filled in as follows:

$\rightarrow$	1	2	3	4
*1	d	u	u	u
*2	u	d	u	u
3	d	u	d	u
4	u	d	u	d

Note that no two values are equivalent.

The total number of sets of matrices that the computer needs to consider for models of P_{A1} is now 2^{20}, i.e. approximately 10^6 or one million. This number of possibilities can easily be handled by a computer and saves one working out a possible matrix model by hit and miss methods which involve a lot of tedious checking of axioms.

II. The Working of the Computer Program

Consider a propositional calculus P, with certain axioms, rules and theorems that are to be checked for validity, invalidity or preservation of validity in various sets of matrices. One only checks for invalidity of a wff when determining whether it cannot be derived from the axioms and rules of P. The computer can only check strong matrix models of P and the rules to be checked are always put into the form of a wff, as in section (d) of the Introduction, and the appropriate connectives & and $\supset$ are added if not available in P. The axioms, rules and theorems of P that are checked are carefully determined so as to make the computer time as short as need be. However, the computer can be used to find matrix models that are not strong, but independent methods must be used to show that the sets of matrices found by the computer are indeed weak matrix models.

The range of values for each matrix position is read into the computer and it increments these to obtain all the sets of matrices to be considered. These ranges of values have to be determined by doing similar preliminary work to that done in section (f) of the Introduction.

(a) *Conditions Imposed on the Matrices*

There are two types of conditions that can be imposed on a particular set of matrices established by incrementation as above. These conditions are checked before any of the axioms, rules or theorems are checked.

The first type of condition is called an *equivalence condition,* and this ensures that a certain position of a certain matrix contains the same value as another position of the same or another matrix. In the example in section (f) of the Introduction, $\&(v_1, v_2) = \&(v_2, v_1)$, for all values v_1 and v_2, and this can be ensured by six equivalence conditions, viz. the ones ensuring $\&(1, 2) = \&(2, 1)$, $\&(1, 3) = \&(3, 1)$, $\&(1, 4) = \&(4, 1)$, $\&(2, 3) = \&(3, 2)$, $\&(2, 4) = \&(4, 2)$ and $\&(3, 4) = \&(4, 3)$. Thus equivalence conditions can take account of symmetry in a matrix, which is not accounted for in the ranges of values read in for each matrix position.

The second type of condition is called an *if-then condition,* and this ensures that, if a certain position of a certain matrix contains a certain range of values, then certain other positions of the same matrix or other matrices contain certain ranges of values. This condition is much more flexible than the equivalence condition. In the above example, one can show, using $\vdash (p \rightarrow q) \rightarrow (p \,\&\, r \rightarrow q \,\&\, r)$, that, if $\&(3, 1) = 3$ then $\&(1, 1) = 1$ and if $\&(3, 1) = 4$ then $\&(1, 1) = 2$. This can be ensured by two if-then conditions where the ranges of values consist of one value only and where there is only one «then» per «if». If-then conditions can also be used to ensure the consequences of trying out a value or a range of values in a certain matrix position.

After determining the ranges of values for each matrix position, the equivalence conditions and if-then conditions, one should weed out any wffs to be checked that are made redundant by any of these conditions.

(b) *Grouping of Wffs and the Use of Cases*

Two further refinements can be made to cut down the computer time. Consider the set W of wffs of P that are to be checked. W can be divided up into mutually exclusive *groups* W_i $(i = 1, 2, \ldots)$ such that in each group only members of a certain subset C_i $(i = 1, 2, \ldots)$ of the set C of all connectives in P appear and each member of C_i appears at least once in W_i. Also, when checking the wffs of a particular group,

several cases may be considered so that several sets of ranges of values for the matrix positions can be read into the computer.

Each group W_i of wffs is checked for validity (or invalidity) in the sets of matrices, independently of the wffs in other groups, so that the only matrices the computer needs to consider are those for the connectives in C_i. For ease of calculation, consider the case of two groups W_1 and W_2 and two corresponding sets of connectives C_1 and C_2, such that $C_1 \supset C_2$ and $C_2 = C$. Also, let there be no conditions imposed on the matrices that affect both a matrix for a connective in C_1 and a matrix for a connective in $C - C_1$. Let the number of possible sets of matrices for the connectives in C_1 be N_1 and the number of possible sets of matrices for the connectives in C be N. If the number of possible sets of matrices for the connectives in $C - C_1$ is D_1 then $N = N_1D_1$. Let the number of sets of matrices that all the wffs in W_1 are valid (or invalid) in be M_1. Then, for group W_2, the number of sets of matrices for the computer to sift through $= M_1D_1$. The total number of sets of matrices for the computer to sift through $= N_1 + M_1D_1 = N/D_1 + \frac{M_1}{N_1} \cdot N = N(\frac{M_1}{N_1} + \frac{1}{D_1})$. M_1 is likely to be much smaller than N_1 because many of the N_1 possible sets of matrices will be rejected by the wffs in W_1. Depending on the conditions on the matrices for the connectives in $C - C_1$, D_1 is likely to be large. In most cases $N(\frac{M_1}{N_1} + \frac{1}{D_1})$ will be much smaller than N, and in almost all cases $N(\frac{M_1}{N_1} + \frac{1}{D_1})$ will be less than N. Hence computer time will be saved by such grouping of the wffs.

There is a choice, for the sets of matrices that all the wffs in a group are valid (or invalid) in, of either storing them on disk ([5]) so that they can be incremented from there for another

([5]) Magnetic tape can be used instead of disks, if available.

group, or branching to another group as soon as each set of matrices is found so that it can be tested further in conjunction with the new connectives of that group and branching back to the original group afterwards. This choice gives rise to various methods of proceeding and the user of the program can indicate in the data fed into the computer which method he prefers. Also, whether all possible groupings of wffs are used or not is up to the user to decide by means of the data.

One method would involve the use of no disk storage at all. If $W_1, W_2, \ldots, W_n$ are the groups in order and $C_1, C_2, \ldots, C_n$ are the corresponding sets of connectives then $C_1 \subset C_2 \subset C_3 \subset \ldots \subset C_n = C$, because as each set of matrices satisfying the wffs in W_i is found, the computer branches to the next group so that the set of matrices can be tested further in conjunction with other connectives, i.e. with the connectives in $C_{i+1} - C_i$. This is a practical method with few compications and has the advantage that it may lead to a quick solution if one of the first sets of matrices obtained satisfying the wffs of W_1 turns out to be part of a set of matrices satisfying all the wffs.

Another method would involve the use of disk storage but with no two groups of wffs with properly intersecting sets of connectives. If W_i, W_{i+1}, W_{i+2} are three groups in order and C_i, C_{i+1}, C_{i+2} are the corresponding sets of connectives such that $C_i \cap C_{i+1} = \varnothing$ and $C_i \cup C_{i+1} \subseteq C_{i+2}$, then the sets of matrices satisfying the wffs of W_i and the sets of matrices satisfying the wffs of W_{i+1} can both be stored on disk and can be incremented from there for group W_{i+2} along with the matrices for any other connectives that might belong to C_{i+2}. The sets of matrices satisfying the wffs of W_i must be stored on disk, but when a set of matrices satisfying the wfs of W_{i+1} is found the computer can branch to the group W_{i+2} for further testing, instead of storing these sets of matrices on disk.

If W_j and W_{j+1} are two groups in order and C_j and C_{j+1} are the corresponding sets of connectives such that $C_j \subset C_{j+1}$, then the sets of matrices satisfying the wffs of W_j can be stored on disk and incremented for group W_{j+1} but it is not necessary to store the matrices on disk in this case.

A third method would involve the use of disk storage with two groups of wffs with properly intersecting sets of connectives. If W_i, W_{i+1}, W_{i+2} are three groups in order and C_i, C_{i+1}, C_{i+2} are the corresponding sets of connectives such that $C_i \cap C_{i+1} \neq \emptyset$, $C_i - C_{i+1} \neq \emptyset$, $C_{i+1} - C_i \neq \emptyset$ and $C_i \cup C_{i+1} = C_{i+2}$, then the sets of matrices satisfying the wffs of W_i and the sets of matrices satisfying the wffs of W_{i+1} must both be stored on disk (6) and then the computer matches up the common connectives to form sets of matrices for all the connectives in C_{i+2} and increments them for the group W_{i+2}. Note that no extra connectives may be present in C_{i+2} that are not present in C_i or C_{i+1}.

If a group of wffs contains some connectives whose matrices are determined from the ranges of values for each matrix position and optional equivalence or if-then conditions then, if these ranges of values and conditions can be reduced to a number of sets of ranges of values and conditions so that the number of sets of matrices the computer has to consider is reduced, then each of these sets of ranges of values and conditions can be treated separately in a loop for *cases*, which is inside the loop for groups of wffs. To simplify the programming, one can specify that there be only one case for groups of wffs where the sets of matrices satisfying them are brought forward into the next group for further testing .

III. The Computer Program in more Detail

(a) *General*

The range of problems one can tackle on the computer depends largely on the speed of the computer and the extent to which one uses conditions on the matrices and uses grouping of the wffs to be checked. However, I have written a program in the SPS language for the IBM 1620 (Model II) computer,

(6) A program could be constructed where the sets of matrices satisfying the wffs of W_{i+1} are not stored on disk or where $C_i \cup C_{i+1} \subset C_{i+2}$, but the programming is simplified by the above stipulations.

one of the earlier and slower computers. A wider range of problems could be tackled using a faster computer than this one.

SPS is a machine language but with each statement on a separate card, each machine operation or definition alphabetically symbolised, and each machine core address that is required alphabetically symbolised or referred to numerically. SPS also has some error messages to facilitate the debugging of the program and allows indirect addressing, which enables one to perform calculations on the core addresses and gives one more control in determining where instructions and data are to be put in the machine core.

My program actually consists of two programs, called AXCORE and MATMOD, respectively. AXCORE reads in the wffs to be checked and places in the top end of the machine core a piece of program, which is the most efficient routine for checking the wffs for validity (or invalidity) in a set of matrices. The reason why this routine must be so efficient is that it is inside all the loops of the program which increment sets of matrices. MATMOD consists of everything else needed in the program and branches to and from the piece of program written by AXCORE, whenever a set of matrices is ready for testing against the wffs and whenever the testing is completed, respectively.

Any language used for this program should preferably be a machine language with mnemonics and with indirect addressing. If a language such as Fortran is used, the program would be less efficient and hence the range of problems it could tackle would be smaller.

(b) *Flow Chart of the Program*

The following is a flow chart of the program, which leaves out most of the detail but contains the main loops.

START

Read in the wffs
for AXCORE

Execution of AXCORE

Read in data to determine whether wffs are to be valid or invalid

Read in data concerning the numbers of values

Initial setting for the number of values

Read in data concerning the numbers of designated values

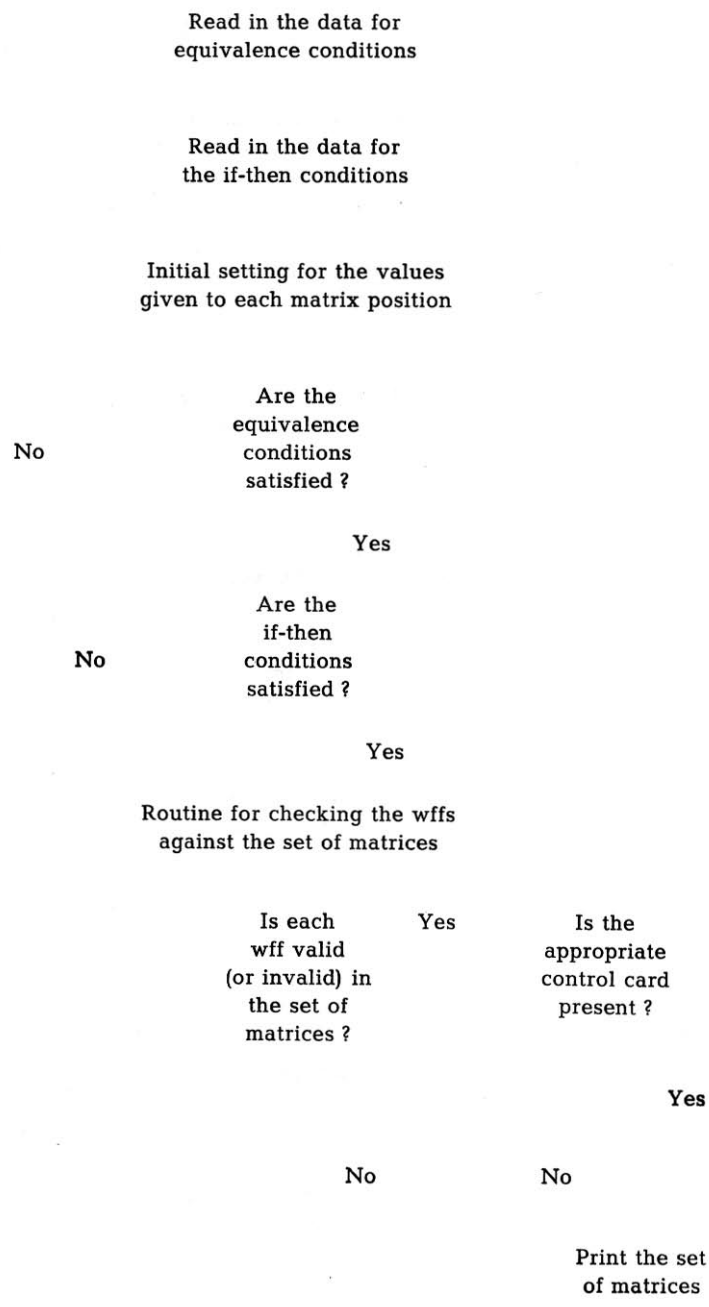
Initial setting for the number of designated values

Read in all data relevant to grouping

Initial setting for the group number

Initial setting for the case number

Read in the ranges of values for each matrix position


Read in the data for equivalence conditions
Read in the data for the if-then conditions
Initial setting for the values given to each matrix position
Are the equivalence conditions satisfied ?
No
Yes
Are the if-then conditions satisfied ?
No
Yes
Routine for checking the wffs against the set of matrices
Is each wff valid (or invalid) in the set of matrices ?
Yes
Is the appropriate control card present ?
Yes
No
No
Print the set of matrices

Increment the values given to each matrix position

No

Is this the last setting for the values given to each matrix position ?

Yes

Increment the case number

No

Is this the last case number ?

Yes

Increment the last group

No

Is this the group number number ?

Yes

Increment the number of designated values

No

Is this the last number of designated values ?

Yes

Increment the number of values

No

Is this the last number of values ?

Yes

END

Note that there is no account taken of branching from one group to another due to sets of matrices satisfying the wffs of one group being tested further in the next group. Also, when asking whether each wff is valid (or invalid) in the set of matrices, this is asked of each wff in turn and as soon as one wff is found not to be valid (or invalid) then the computer branches to ask whether this is the last setting for the values given to each matrix position.

(c) *Control Cards, Program Switches and Error Messages*

Use of control cards, program switches and error messages is optional and they are added for the sake of convenience, and for ease of correcting any errors in the data.

The following are suggestions for control cards. One control card could cause the printing of why each set of matrices satisfying all the other conditions did not satisfy one of the wffs. Another could determine whether to print the sets of matrices satisfied by all the wffs of a group, or not. Another could determine whether to stop after printing the first matrix which satisfies all the wffs, or not. Each of these control cards could alternatively be replaced by appropriate data, but it depends on convenience which is preferable.

The following are suggestions for program switches. Switch 3 could be used to print out why the set of matrices the computer is now dealing with does not satisfy one of the wffs provided it satisfies all the other conditions. This switch would only be effective when the control card which does *not* cause the above printing is present. Switch 2, if on, could cause the printing of the set of matrices being considered at the time, followed by the computer stopping. If switches 1 and 2 are on simultaneously, the computer could then branch to monitor control instead of stopping after the printing.

The following are suggestions for error messages, which would be printed out if there is one of a number of possible mistakes in the data read in to the computer. For the AXCORE data, one could have the message, WRONG SYMBOL IN

AXIOM NO. xx, followed by the computer branching to monitor control, if such a wrong symbol appears. Also, one could have the message, AXIOM NO. xx IS ILL-FORMED, followed by the computer branching to monitor control, if such an axiom is ill-formed.

One can put into the MATMOD data certain data checks in the various loops to ensure that no data has been missed or that no excess data has been added. Appropriate error messages would be needed if such an error arises. The computer could then go on to the next number of designated values. There are also various other possible error messages which can be used to show that there has been a wrong sumbol used in a piece of data or that none of the allowable alternatives in a piece of data are present. Usually, the computer would then go on to the next number of designated values.

La Trobe University Ross T. Brady

REFERENCES

1. R.B. Angell, «A Propositional Logic with Subjunctive Conditionals», Journal of Symbolic Logic, Vol. 27 (1962), pp. 327-343.
2. S. Halldén, «The Logic of Nonsense», Uppsala, 1949.
3. R. Harrop, «Some Structure Results for Propositional Calculi», Journal of Symbolic Logic, Vol. 30 (1965), pp. 271-292.
4. G.E. Hughes and D.G. Londey, «The Elements of Formal Logic», Methuen, 1965.
5. J. Kalicki, «Note on Truth Tables», Journal of Symbolic Logic, Vol. 15 (1950), pp. 174-181.
6. N. Rescher, «Many-Valued Logic», McGraw-Hill, 1969.