KNOWLEDGE FACTORING USING NORMALIZATION THEORY

J. VANTHIENEN
M. SNOECK

Katholieke Universiteit Leuven
Department of Applied Economic Sciences
Dekenstraat 2, 3000 Leuven (Belgium)

tel. (+32) 16 28 58 09
fax. (+32) 16 28 57 99
E-mail: fdban06@blekul11

0. ABSTRACT

The number of designed knowledge based systems largely exceeds the number of operational expert systems. One of the main reasons for this is that knowledge bases still suffer from major difficulties to represent, organize, validate and manage knowledge in an appropriate manner. Research has shown that decision tables can offer a solution in many of these problem areas. However, as the number of actions and conditions contained in the knowledge increase, decision tables quickly become very large and complex. Such large and complex decision tables nullify the advantage of a good overview and suffer from the same flaws as large and unstructured sets of data. More especially the maintenance of a large decision table can lead to redundancy and conflicting information due to update anomalies In this paper a structuring process is proposed equivalent to normalization rules in relational database design, based on the correspondence between functional dependencies in database design and logical implication. Factoring decision tables offers an excellent guide-line to factor knowledge in order to improve the intelligibility and facilitate maintenance.

Index Terms: *Decision Modeling, Decision Tables, Decision Making, Knowledge Acquisition, Knowledge Representation, Normalization, Software Engineering, Systems Analysis and Design*

1. Introduction

In any kind of organization, many people deal with substantial amounts of knowledge. For companies it is important to have the knowledge of their experts explicitly stated in e.g. regulatory texts, procedures, norms, etc. In an increasing number of cases, the volume and complexity of the corporate knowledge raises the need for intelligent computer support in order to manage the knowledge and guarantee a higher quality of final decisions in terms of consistency and correctness. If we would be able to incorporate the corporate knowledge into a knowledge base, applying this knowledge to specific cases is a matter of inferencing and asking the right questions. And precisely these inferencing and consultation abilities are widely available in common expert system shells and tools.

The design and construction of expert systems is however not an easy task as is demonstrated by the poor number of operational knowledge based systems. Still a lot of research has to be done in areas such as knowledge validation, verification, representation, management, ... etc. In the specific area of knowledge representation, research has shown that in a lot of cases, the use of decision tables to represent procedural decision situations is preferable to other representation mechanisms (such as texts, decision trees, flowcharts, IF-THEN rules, Horn-clauses, nested IF-THEN-ELSE structures, ...). See e.g. SANTOS-GOMEZ L. & DARNELL M. [7] for an empirical evaluation of decision tables for constructing and comprehending expert system rules.

Large and complex decision tables however severely diminish one the most important advantages of intelligible knowledge representation. In addition their maintenance easily leads to redundant and conflicting information due to update anomalies. It is however possible to transform (factor) large decision tables into a structure of dependent tables and subtables. Although it can be left to the user to evaluate when and how a decision table should be transformed into a structure of dependent decision tables, it is preferable to use a well-founded technique to investigate wether and how a decision table can be factored.

A structure of dependent tables usually gains a lot in overview compared to the expanded decision table. But, even if in some cases it is possible to factor a decision table, this might not be desirable as the specific advantages of the decision table representation may be lost in the factoring process. In this paper a structuring process is proposed, based on the equivalence between functional dependencies in database design and (a subset of) propositional logic [4, 6]. Although the analogy between decision table knowledge and database dependencies is striking, there are some major differences. In the case of a decision table the functional dependencies themselves are stored in the database and are the possible subject of updates. In contrast, the functional dependencies for database relations form an implicit set of rules which must be enforced when the database is updated, but they are not subject to change themselves.

In spite of these important differences, the normalization rules of database design provide an excellent guide-line for the factoring of decision tables. Because of the correspondence between logical implication and functional dependency, the rules for normalization of database relations can be translated into rules for factoring decision tables into subtables. The following paragraph gives a short introduction to decision table theory. Next, the analogy with database relations is established. Before the normalization rules for decision tables are formulated in the fifth paragraph, a short review of database relation normalization is given.

2. Knowledge representation using decision tables

A decision table is a tabular representation used to describe and analyze procedural decision situations, where the state of a number of conditions determines the execution of a set of actions. Not just any representation, however, but one in which all distinct situations are shown as columns in a table, such that every possible case is included in one and only one column (completeness and exclusiveness).

"A decision table is a table, representing the exhaustive set of mutual exclusive conditional expressions, within a predefined problem area." (VERHELST [10])

In order to make a meaningful use of decision tables possible, the decision table has to be defined clearly and must meet the important requirements of consistency and completeness. For these purposes, the decision table is defined as follows.

A decision table consists of four parts:

- The condition subjects are the criteria which are relevant to the decision making process. They represent the items about which information is needed to take the right decision.
- The condition states are logical expressions determining the relevant sets of values for a given condition. Every condition has its set of condition states.
- The action subjects describe the results of the decision making process.
- The action values are the possible values a given action can take.

Condition subjects are found in the upper left part of the table, action subjects in the lower left part. Condition states and action values are found at the right hand side.

These four parts are defined more formally:

- CS = $\{CS_i\}$ (i=1..cnum) is the set of condition subjects;
- CD = $\{CD_i\}$ (i=1..cnum) is the set of condition domains, with CD_i the domain of condition i, i.e. the set of all possible values of condition subject CSi;
- CT = $\{CT_i\}$ (i=1..cnum) is the set of condition state sets, with $CT_i = \{S_{i,k}\}$ (k=1..ni) an ordered set of n_i condition states $S_{i,k}$. Each condition state $S_{i,k}$ is a logical expression concerning the elements of CD_i, that determines a subset of CD_i, such that the set of all these subsets constitutes a partition of CD_i (completeness and exclusiveness of the condition states);

- AS = $\{AS_j\}$ (j=1..anum) is the set of action subjects;
- AV = $\{AV_j\}$ (j=1..anum) is the set of action value sets, with AV_j = {true (x), false (-), nil (.)} the set of action values, which is, in first instance, equal for every action subject, for reasons of consistency checking.

Every column in the decision table contains a state for each condition subject or a contraction of states that yield the same result (possibly irrelevant (-) if this is the case for all states of the condition), followed by the resulting value for each action subject.

A table column represents a decision rule of the form:

IF CS_1 is S_{1k} AND CS_2 is S_{2m} AND ...
 	
 THEN action AS_j AND ...

If each column only contains simple states (no contractions or irrelevant conditions), the table is called an *expanded* decision table (*canonical form*), in the other case the table is called a *contracted* decision table (*consolidated form*). The translation from one form to the other is defined as *expansion (rule expansion*) and *contraction (consolidation*) respectively (CODASYL [1]).

The condition subjects and action subjects can refer to other tables (*subtables*). The replacement of these references by the tables themselves, the *junction* of tables, is called (*table) expansion*. The reverse process, the *division* into subtables, is defined as *factoring*. Two types of subtables are possible: the action subtable, i.e. a further specification of a certain action, and the condition subtable, determining the value of a condition. All subtables are of the closed type, this means that after ending a subtable, the calling table regains control.

3. NORMALIZATION RULES FOR DECISION TABLES

The formal correspondence between the decision table and the relational table is given in figure 1. Since the decision table is equivalent to the relational table, the relational technique (in the form of a relational DBMS) can be used to construct the decision table. This means that both the physical storage and the construction and manipulation of the decision table can be partly executed through the relational structure and operators (relational algebra) [9].

Decision table	Relational table
condition (row)	key attribute (column)
condition states	key domain
condition reference	foreign key
action	non-key attribute
action value	non-key domain
stub	heading
number of rows	degree
entry	attribute value
column	n-tuple
number of columns	cardinality

figure 1: terminology of the decision table and the relational table

In the case of actions and conditions the functional dependency can be translated into a logical implication in a straightforward way : if action A_i is functional dependent on condition subject CS_j, then CS_j logically implies A_i. In both cases the notation is the same : CS_j **Error! Reference source not found.**alization rules can be applied to both database relations and decision tables.

Both normalization of relations and of decision tables has as primary goal to avoid redundancy and update anomalies. In addition the normalization of decision tables simplifies decision tables and increases their readability. As the normalization procedure for decision tables is derived form the normalization process for database relations, we may assume that also this decomposition process is reversible. In fact it can be proved by means of propositional logic that the junction of the decision tables resulting from the decomposition process gives rise to the same set of decision rules. More formally, if a set of original decision rules D (D is represented as a decision table) is decomposed into a set of new decision tables $\{D_1, ..., D_n\}$, then D and $D_1 \cup D_2 \cup ... \cup D_n$ are equivalent.

3.1. First Normal Form

Definition : An expanded decision table is in **first normal form (1NF)** if every condition state and every action value is an atomic value and not a set.

Every decision table that conforms the definition of the second paragraph automatically is in first normal form. Indeed, the condition states are logical expressions[1], and only limited entry actions[2] are considered whose values exclude each other. Not accepting sets of values for condition states significantly facilitates the checking for consistency and completeness of the decision table.

3.2. Second Normal Form

3.2.1. Original Second Normal Form

Definition :

An expanded decision table is in **second normal form (2NF)** if and only if it is in first normal form and every action is fully dependent on all the conditions. More formally :

$\forall AV_j$:[CT $\forall$

The second normal form is a strong demand. Indeed, decision tables are a powerful way to clearly represent relations between conditions and actions, even if some actions are determined by only part of the conditions. The breakdown of decision tables into a structure of decision table in

[1]. Remark that although the logical expression can denote a *set* of values, it is the logical expression itself which is the condition value and which is indeed an atomic value.

[2]. Limited entry action values ('x' or '-') are always atomic, but of course also other action values are allowed by the first normal form, as long as these values are atomic (e.g. '1' is atomic but '1,2' is not).

second normal form is theoretically appealing but not always of practical use, because the overview can be lost by the breakdown.
Except in a number of specific cases, the conversion to second normal form leads to a number of sequential decision tables in which certain conditions (or actions) are repeated. This repetition of conditions is seen as inconvenient and a significant loss of readability while at the same time it introduces inefficiency due to repeated testing of the same conditions. The more that it is exactly a major goal of the decision table technique to visualize the combined effect of a set of conditions. The consequences of not meeting the second normal form are not always a sufficient reason to split a decision table.

However, in many cases the transformation to second normal form significantly simplifies the decision table because the resulting decision tables are smaller and easier to read. There are indeed some specific decision table constructs in which the transformed decision table is always a better representation of the decision logic. In this way a number of variants of the second normal form are derived which are weaker than the second normal form but generally applicable : every decision table is supposed to meet these weaker variants of the second normal form.

3.2.2. Weaker variants of the Second Normal Form

3.2.2.1 Elementary second normal form

Definition : A decision table is in **Elementary 2NF (E2NF)** if and only if it is in first normal form and the complete action set is fully dependent of the whole condition set. In other words : there does not exist a subset of the condition set of which the whole action set is dependent. More formally :
$\neg (\exists CT' \subset CT, CT' \neq CT : CT'$
In a decision table that is not in elementary second normal form, superfluous conditions can be found which can be deleted from the decision table without any loss of knowledge. E2NF does not imply that every single action is fully dependent of the whole set of conditions. The possible dependencies that can occur in a decision table in elementary second normal form are listed below :

Given the set of condition subjects CS = $\{CS_1, \dots CS_{cnum}\}$, then
A_j (1 ∣ j ∣ anum) is implied by CS (by definition);
$\{A_1, \dots , A_{anum}\}$ is implied by CS (union);
A_j (1 ∣ j ∣ anum) is strictly implied by CS (2NF);
$\{A_1, \dots , A_{anum}\}$ is strictly implied by CS (elementary 2NF);

3.2.2.2 Disjunctive second normal form

Definition : A decision table is in **disjunctive 2NF** (D2NF) if and only if it is in 1NF and it is not composed of unrelated sets of condition and action subjects. This means that there does not exist a subset of the action subjects that is fully dependent of a subset of the condition subjects while the remainder of the action subjects are dependent of the remainder of the condition subjects. More formally :
$\neg(\exists CT' \subseteq CT, A' \subset AV : (CT'$
This normal form can be used when the global decision table is a composition of two completely independent decision tables. The split decision table obviously is a better representation than the original table (under the assumption that there are no sequence restrictions between the condition subjects). In this case the transformation always leads to a smaller number of columns, except in the case of limited entry conditions with two values where the number of columns remains equal. An example is given in paragraph 4.

In one special situation there might be action subjects that must always be executed and are dependent on none of the condition subjects. Such actions can be put in a separate table, but are often kept in the original table to keep the overview. The same is valid for action subjects that must never be executed and which are in fact superfluous.

3.2.2.3 Partially related second normal form

Definition : A decision table is in **partially related 2NF (P2NF)**[3] if and only if it is in 1NF and there is no subset of action subjects

(i) that is fully dependent of a subset of the condition subjects while the remainder of the action subjects is dependent of a subset of the same condition subjects together with the remainder of the condition subjects and

(ii) for which holds that the different configurations[4] of the actions with relation to the common condition subjects do not have other common actions or condition subjects.

More Formally :

$\neg$ ($\exists$ CC $\subseteq$ CT, CT' $\subseteq$ CT, CT" $\subseteq$ CT, CT' $\cap$ CC = _, CT" $\cap$ CC = _, CT" $\cap$ CT' = _, A' $\subset$ AV :
(CC $\cup$ CT'

This is illustrated in figure 2. In this case subtables T1 and T2 are in parallel with each other : only one of both will be executed, depending on the result of condition 1.

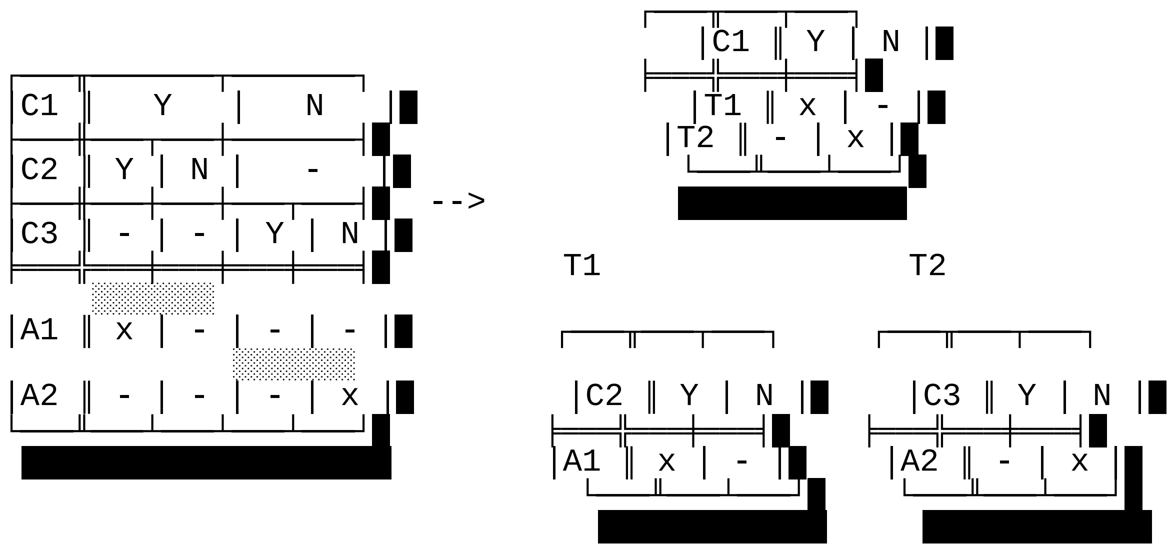

Figure 2 : Normalization of partially related subsets

One could argue that in this case the global decision table is shorter and clearer than the normalized structure and as a consequence there is no advantage to split the global table. This, however, is not always the case.

The partially related 2NF splits condition and actions subjects from the common action and condition subjects, but only if the different action configurations do not have conditions or actions in common with the subset, to avoid repetition. Figure 3 illustrates this with a few examples. The splitting of the table must in all respects be considered at construction and manipulation time. For validation purposes the global decision table might sometimes be preferred, which can then be considered as a view.

[3]. P2NF implies D2NF, which in turn implies E2NF.
[4]. A configuration is the configuration of values a certain set of actions takes for a given condition.

C1	a		b		c
C2	Y	N	Y	N	-
A1	-	-	x	x	x
A2	x	-	x	-	-
A3	-	x	-	x	-

-->

C1	a	b	c
A1	-	x	x
T1	x	x	-

└─>T1

C2	Y	N
A2	x	-
A3	-	x

(Two configurations of A2 and A3 with respect to C1 are equal, the other configurations have no actions or conditions in common)

C1	a		b		c
C2	Y	N	Y	N	-
A1	-	-	x	x	x
A2	x	-	-	-	-
A3	-	-	x	-	-

(no configurations are equal, different configurations have C2 in common)

C1	a		b		c
C2	Y	N	Y	N	-
A1	-	-	x	x	x
A2	x	-	x	-	x
A3	-	-	-	-	x

(two configurations are equal; different configurations have A2 in common)

Figure 3 : Partially related 2NF.

Beside the maintenance and isolation advantages, there are two major reasons to perform this normalization :

1) The global decision table will not always be as clear as the normalized structure, as the ordering of conditions plays a central role in the readability of the decision table. While the ordering of the parallel conditions is irrelevant for the resulting width of the decision table, the ordering of the common conditions is. Placing the common conditions at the end of the list, makes the decision table unsurveyable and obscures the existence of unrelated conditions (see figure 4).

C3	Y				N		
C2	Y		N	Y		N	
C1	Y	N	-	Y	N	Y	N
A1	x	-	-	x	-	-	-
A2	-	-	-	-	x	-	x

-->

C1	Y		N	
C2	Y	N	-	
C3	-	-	Y	N
A1	x	-	-	-
A2	-	-	-	x

Figure 4 : hidden mutual unrelated conditions

2) The don't care symbol as entry for an unrelated condition does not always indicate an irrelevancy. Testing this condition is not always only irrelevant, it might be undesirable, because of possible side-effects. This is the case when the condition is in fact a condition subtable or when the condition has a hidden bound action. Reordering of conditions is of no help; in fact the don't care symbol should be replaced by a "do not test" symbol. These kind of problems can be avoided by using a different notation, but in turn this is no solution for the condition ordering problem. Splitting the table solves both problems.

3.3. Third Normal Form

Definition :

A decision table is in **third normal form** if and only if it is in second normal form (possibly only elementary 2NF) and every action is non transitively dependent of the conditions. More formally :

$\neg(\exists$ CT' $\subseteq$ CT, $C_i \in$ CT, $C_j \notin$ CT' : CT'

Third normal form is always a matter of related condition or action subjects. When action subjects are mutually related it is most of the time sufficient to combine them into one action subject or an action subtable without conditions. Dependencies between conditions, (in fact impossibilities) indicate that a certain condition is dependent of other condition combinations and plays the role of an action. This can be obtained by putting the condition in a condition subtable where the actions determine the value of the condition subject (see figure **Error! Bookmark not defined.** where C3 | $((C1 \wedge C2) \vee (\neg C1 \wedge \neg C2))$).

If splitting the condition table does not imply repetition of conditions and actions, the conversion to third normal form is always recommendable. If repetitions are necessary the surveyability might be lost so that the global decision table is preferable.

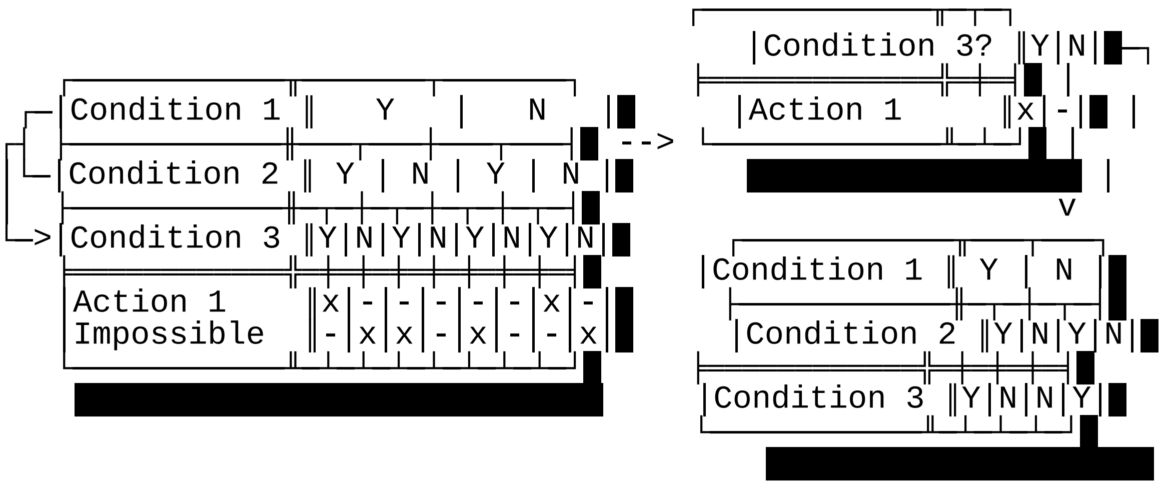

Figure 5 : conversion to third normal form

4. EXAMPLE

The following example of simple order entry rules illustrates how the normalization rules can successfully be used to factor decision knowledge. Figure 6 gives the expanded decision table as it was build starting from the specifications given by the expert. In fact this table contains two independent sets of actions and conditions and is therefore not in disjunctive 2NF. Indeed the action values of Ask written confirm depend only on the value of the condition Phone Order. More especially, the action values for this action are equal in columns 1,3,5,7,9,11,13,15,17 which match the 'Yes'-value for the Phone Order condition. In the same manner Ask written confirm has always the same value in the columns matching the 'no'-value for Phone Order. Hence the decision table can be factored into the two tables of figures 7 and 8 which must be executed in sequence.

The decision table of figure 8 still is not in partially related 2NF. Indeed, in figure 8 the action values for Execute, Refuse Order and Put on Waiting List are independent of the condition Quantity Ordered. More precisely, the action values of column 1 equal those of column 2 and 3 and the action values of column 5 equal those of column 6 and 7 for these actions. The common conditions (CC in the definition) Credit Limit, Customer and Stock Sufficient can be put in a separate table (see figure 9). The remainder of the conditions and actions is put in a second table (see figure 10) which is called by the action 'Discount' in the first table[5]. So finally, the table of figure 6 can be factored into the three independent tables of figures 7, 9 and 10. The decision table in figure 9 could be factored even further by moving the Stock Sufficient condition to the subtable. However, as argued before over-factoring must be avoided when the specific advantages of the decision table representation are lost by that factoring process..

5. PRACTICAL IMPLICATIONS

As has been illustrated in the various examples, normalization rules for decision tables are an excellent technique to investigate how and when a decision table can be factored. It was however also clear from the examples that a possible factoring is not always recommendable from a readability or surveyability point of view. As with normalization, ultimate decomposition may be abandoned for well-defined reasons, as long as one is aware of the potential risk of redundancy or dependency.

6. CONCLUSION

In this paper it was demonstrated how normalization rules for database design can successfully be transposed to the decision table formalism by making use of the strict correspondence between functional dependency and (a subset of) propositional logic. The resulting rules can be used as a guideline for decision table factoring.

7. REFERENCES

[1] Codasyl, *"A Modern Appraisal of Decision Tables*", Report of the Decision Table Task Group, ACM, New York, 1982, pp. 230-232.

[2] Codd E., "*A Relational Model of Data for Large Shared Data Banks",* Communications of the ACM, 13(6), pp. 377-387, 1970.

[3] Date C. J., *"An introduction to Database Systems, Volume 1, Fifth Edition*", Addison-Wesley Publishing Company, 1990, 854pp.

[4] Fagin, R., *Functional Dependencies in a Relational Database and Propositional Logic,* IBM Journal of Research & Development, 21(6), Nov. 1977, pp. 534-544.

[5] Kent, W. [83], A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM, 26(2), Febr. 1983, pp. 120-125.

[6] Sagiv J., Delobel C., Parker D. S., Fagin R., "*An equivalence Between Relational Database Dependencies and a Fragment of Propositional Logic*", JACM, Vol. 28, No. 3, July 1981, 435-453.

[7] Santos-Gomez L. and Darnell M., Empirical evaluation of decision tables for constructing and comprehending expert system rules, Knowledge Acquisition 4 (1992), 427-444.

[8] Ullman, J., *Principles of Database Systems,* Computer Science Press, Inc., 1980, 379 pp.

[5]. In this case CT' is empty so that the actions A' which depend from CC $\cup$ CT' can be put in the table with the common conditions CC.

[9] Vanthienen J., Wets G., *An Expert System Application Generator Based on Decision Table Modeling*, 9 pp., to be presented at The Second World Congress on Expert Systems, January 10-14, 1994, Lisbon, Portugal.

[10] Verhelst M., "*De Praktijk van Beslissingstabellen",* Kluwer, Deventer/Antwerpen, 175 pp, 1980.