Available online at www.sciencedirect.com
ScienceDirect
Journal of Mathematical Psychology 50 (2006) 501–510
Journal of Mathematical Psychology
www.elsevier.com/locate/jmp

A note on the complexity of Boolean concepts

Ronaldo Vigo

Cognitive Science Department, 819 Eigenman Hall, Indiana University, Bloomington, IN 47406, USA

Received 16 August 2005; received in revised form 30 May 2006
Available online 10 August 2006

Abstract

What is the relationship between the degree of learning difficulty of a Boolean concept (i.e., a category defined by logical rules expressed in terms of Boolean operators) and the complexity of its logical description? Feldman [(2000). Minimization of Boolean complexity in human concept learning. *Nature*, *407*(October), 630–633] investigated this question experimentally by defining the complexity of a Boolean formula (that logically describes a concept) as the length of the shortest formula logically equivalent to it. Using this measure as the independent variable in his experiment, he concludes that in general, the subjective difficulty of learning a Boolean concept is well predicted by Boolean complexity. Moreover, he claims that one of the landmark results and benchmarks in the human concept learning literature, the Shepard, Hovland, and Jenkins learning difficulty ordering, is precisely predicted by this hypothesis. However, in what follows, we introduce a heuristic procedure for reducing Boolean formulae, based in part on the well-established minimization technique from Boolean algebra known as the Quine–McCluskey (QM) method, which when applied to the SHJ Boolean concept types reveals that some of their complexity values are notably different from the approximate values obtained by Feldman. Furthermore, using the complexity values for these simpler expressions fails to predict the correct empirical difficulty ordering of the SHJ concept types. Motivated by these findings, this note includes a brief tutorial on the QM method and concludes with a brief discussion on some of the challenges facing the complexity hypothesis.

Keywords: Boolean concept learning; Complexity; Categorization; Rule-based classification

1. Introduction

An important aspect of theories of concept learning is the ability to predict the degree of learning difficulty for different types of concepts. In this respect, one particular class of concepts known as Boolean concepts have been of fundamental interest in the literature. Boolean concepts are concepts that are definable by Boolean expressions (expressions consisting of disjunctions, conjunctions, and negations of variables that stand for binary object features). These types of concepts have been studied extensively by investigators such as Shepard, Hovland, and Jenkins (1961), Bourne (1966, 1974), Nosofsky, Gluck, Palmeri, McKinley, and Glauthier (1994), and more recently by Feldman (2000).

Shepard et al. (1961) investigated Boolean concepts consisting of three binary features or dimensions and four examples (four positives) and four non-examples (four negatives) for a total of eight stimuli. Later, Feldman (2000) refers to these types of concepts as the 3(4) family of concepts (where the numeral 3 denotes the number of binary dimensions or features and the numeral 4 denotes the number of positive stimuli). For example, suppose that the tested dimensions are those of shape, size, and color; if we let a stand for triangular, a' stand for round, b stand for small, b' stand for large, c stand for white, and c' stand for black, then one of the concepts studied by Shepard et al. can be expressed by the Boolean formula $a'b'c' + a'b'c + a'bc' + a'bc$. In other words, the formula describes the concept or category associated with objects that are either round, large, and black, or round, large, and white, or round, small, and black, or round, small, and white. Note that each conjunction or logical product represents a positive example of the concept while the remaining four out of eight (2^3) possible conjuncts or logical products represent the non-examples or negative examples of the concept.

E-mail address: rvigo@indiana.edu.

0022-2496/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jmp.2006.05.007

Although there are 70 ($C_4^8 = \frac{8!}{(8-4)!4!}$) possible expressions that can be constructed by selecting four positive examples out of eight available possible conjunctions, subsets of these are structurally isomorphic or structurally reducible to each other. Structurally equivalent concepts are those that can be obtained by a consistent reassignment of labels and polarities of the labels in their corresponding Boolean expressions. Since the choice of labels for the various features as well as their polarity and the order of the disjuncts is arbitrary, any Boolean expression in the class of structurally equivalent expressions suffices to describe the same concept. For example, the concepts $a'b' + a'b$ and $ab + ab'$, although not truth functionally equivalent, are structurally equivalent since we can convert the first into the second by switching the polarities of the arbitrary labels "a" and "b" (i.e., by letting "a" label the feature previously labeled by "a'" and letting "b" label the feature previously labeled by "b'").

Structural equivalence can also be illustrated geometrically using a Boolean cube (see Fig. 1). The relationship between the four points representing the four disjuncts in the Boolean expression is invariant in respect to rigid rotations of the cube. It turns out that there are exactly six such structural relationships that partition the set of all possible 3(4) expressions into six subsets or equivalent classes (for an in depth combinatorial discussion see Aiken, 1951; Higonnet & Grea, 1958). These are illustrated in Fig. 1. Thus, there are 70 Boolean expressions of the 3(4) family that can be used to describe a total of six 3(4) type Boolean concepts.

These six types were studied by Shepard et al. (1961) who measured the degree of learning difficulty measured by the number or errors that the subjects made until they reached a certain performance criterion. It was found that the six concept types listed in Fig. 1 followed the following order of learning difficulty: I < II < [III, IV, V] < VI. Thus, type I problems involving the simplest concept or category structure yielded the least number of errors, followed by type II, followed by types [III, IV, V] which yielded approximately the same number or errors, and finally type VI. The SHJ ordering has served as a fundamental benchmark for models of human concept learning (for discussions see Kruschke, 1992; Love & Medin, 1998; Nosofsky et al., 1994).

Feldman (2000), motivated by the SHJ study, sought to find a connection between the degree of learning difficulty of a Boolean concept and what he defines as its Boolean complexity. As defined by Feldman (2000, p. 630): "The Boolean complexity of a propositional concept is the length of the shortest Boolean formula logically equivalent to the concept, usually expressed in terms of the number of literals (positive or negative variables)". Feldman's study examines an unprecedentedly large number of Boolean concept families including the SHJ family: these included the 3(2), 3(3), 3(4), 4(2), 4(3), and 4(4) families for a total of 41 Boolean concepts. In addition, Feldman pays close attention to families where the number of positive and negative examples differ. For example, 3(2) concepts each have two positives and six negatives and their mirror image has six positives and two negatives. Feldman refers to this distinction as a distinction in the parity of the Boolean concept, where the concept is in up parity when the number of positive examples is smaller than the number of negative examples and in down parity when the number of positive examples is greater than the number of negative examples.

Taking complexity (and parity) as his independent variables and proportion of correct responses as his dependent variable, he concludes from the data that in general, subjective difficulty is well predicted by Boolean complexity and concept parity. Clearly, since parity cannot be tested for the SHJ types, Boolean complexity is then the sole independent variable in regards to the SHJ types. Moreover, Feldman (2000, p. 630) claims: "When the SHJ types are considered from the perspective of mathematical logic, however, a simple explanation of the difficulty ordering emerges: the difficulty of the six types is precisely predicted by their Boolean complexity". He adds later in p. 631 that: "These Boolean complexity values predict the order of empirical difficulty precisely. This exact correspondence has not previously been noted, though Shepard et al. speculated about it in their original paper, and the relation between Boolean complexity and human learning has never been comprehensively tested".

Feldman derives these conclusions from the application of a particular heuristic technique for finding the Boolean complexity values. However, other than mentioning that it is based on forms of factorization, the precise nature of the technique is not stated in his paper. Unfortunately, as we document in this note, for some of the SHJ types, the technique leads to formulae with length that are somewhat different from their actual Boolean complexity values (i.e., there are shorter Boolean formulas for expressing these concepts). Furthermore, as we will document, what we believe are the actual Boolean complexity values fail to predict the SHJ ordering. Feldman (2000, 2003a) has appropriately acknowledged that the heuristic technique he

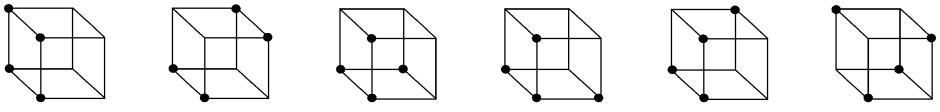

Fig. 1. SHJ types graphed in Boolean space.

used provides only "approximations" (Feldman, 2000, p. 630) and "upper bounds" (Feldman, 2003a, p. 79) on the shortest Boolean formulas. The key point, however, is that the hypothesis that Boolean complexity predicts precisely the ordering of difficulty of the SHJ types is unwarranted.

In what follows, we introduce a heuristic based on a well-established minimization method from Boolean Algebra known as the Quine–McCluskey (QM) method and apply it to show that two of the SHJ concept types' attributed complexity values are markedly lower than the lengths of formulae obtained by Feldman's reduction technique (complexity values are also lower for one type in the 4(3) family and six types in the 4(4) family). However, since the QM method lies at the core of our heuristic, we first give a brief tutorial on the method. After, we discuss in greater detail the implication of our findings for the Boolean complexity hypothesis of concept-learning difficulty.

2. The QM method

A fundamental problem addressed in Boolean Algebra is that of finding the optimal or simplest Boolean formula defining a given truth function. Indeed, digital circuit designers, looking to save on manufacturing time and cost, began developing and implementing formula minimization methods before the advent of the modern digital computer. The formula minimization problem is particularly important since any given Boolean function may be represented by an infinite number of Boolean expressions or formulae. That this is the case may be illustrated by a simple example: the Boolean formula $a \vee \sim a$ denotes a function whose value is true regardless of what value a may hold, but so does the formula $(a \vee \sim a) \vee a$, and similarly the formula $((a \vee \sim a) \vee a) \vee a$, and so on.

Unfortunately, the problem of finding the minimal Boolean expression for any Boolean function, is deemed intractable: that is, it is solvable in theory but not in practice except for small instances of the problem (i.e., for small inputs). In fact, only problems that have polynomial time solutions are known to be practically solvable for more than the smallest inputs; in contrast, for the Boolean minimization problem the only known solutions are of exponential complexity. To see why exponential-time solutions are impractical, consider a problem that requires 3^n operations to solve (n is the size of the input). Now, let us assume that we have a computer that is capable of performing 10^{10} operations per second. If we let $n = 70$, then it would take our computer over 10^{18} years to solve the problem! However, this does not mean that if one makes appropriate constraints on the problem, heuristic algorithms do not exist that can find an efficient or quick solution to it. This point can be best appreciated when considering that in our example above it would take approximately one second to solve a problem of input size 21. Thus, for small instances, our example problem is tractable (Monnasson, Zecchina, Kirkpatrick, Selman, & Troyansky, 1999).

Another constraint that may be imposed on the Boolean minimization problem involves formulae in disjunctive normal form or DNF. Before we define what it means for a formula to be in disjunctive normal form, we define the concept of a fundamental conjunction as either a literal (i.e., a negated or unnegated variable), or a conjunction of two or more literals no two of which involve the same variable. For example, while abc is a fundamental conjunction, aba' is not a fundamental conjunction since the literals a and a' involve the same variable a. Now we can proceed with our definition of a formula in DNF. A Boolean expression **F** is said to be in DNF if either: (1) **F** is a fundamental conjunction, or **F** is a disjunction of two or more fundamental conjunctions (as long as no fundamental conjunction in **F** is a part of any other). An important theorem in Boolean algebra states that any Boolean function that is not a self-contradiction (i.e., false under all possible truth value assignments to its variables) is logically equivalent to a DNF expression.

Furthermore, there is a special type of DNF, known as a *full* DNF, in which we will be particularly interested since the concept types in Shepard et al. (1961) and in Feldman (2000, 2003a) assume this form. A Boolean expression **F** in DNF is said to be in *full* DNF in respect to its variables $x_1, \ldots, x_n$ if: (1) any variable in **F** is one of the variables $x_1, \ldots, x_n$ (i.e., **F** is closed under the set of variables $\{x_1, \ldots, x_n\}$), and (2) each disjunct in **F** contains all the variables $x_1, \ldots, x_n$. Clearly, every concept type in Feldman (2000, 2003a) is in full disjunctive normal form. This later fact is important since practical algorithmic techniques are available that allow one to find the minimal DNF expression corresponding to any Boolean expression in full DNF (as long as it features a sufficiently small number of distinct variables). Since the Boolean concept types of any psychological interest do not consist of more than four distinct variables, these heuristic algorithms are for all intent and purpose tractable.

We provide a brief tutorial of one such technique for obtaining minimal DNF expressions from full DNF expressions in the present section. In various cases, these minimal disjunctive normal form expressions *are already of lower complexity* than the previous formulas that Feldman (2000, 2003a) arrived at with his reduction method. Moreover, as will be explained later, given the minimal disjunctive normal form expressions, one can then apply a specific manual factorization technique and a simple heuristic strategy so as to yield even shorter expressions not necessarily in DNF. As stated earlier, among the 41 concept types considered by Feldman (2000), our heuristic involving this method yields nine expressions that are shorter than his previously obtained formulas, including two for the critically important Shepard, Hovland, and Jenkins' types.

The technique we speak of was developed independently by Quine (1955) and McCluskey (1956) and is known as the QM method.[1] The QM method, a technique for finding the

[1]Another well-known Boolean technique for generating minimal DNF expressions is known as the Karnaugh Map method developed by

ROW	A	B	C	f(A,B,C)	
0	0	0	0	1	A'B'C'
1	0	0	1	1	A'B'C
2	0	1	0	1	A'BC'
3	0	1	1	1	A'BC
4	1	0	0	0	AB'C'
5	1	0	1	0	AB'C
6	1	1	0	0	ABC'
7	1	1	1	0	ABC

DNF of f(A,B,C)
A'B'C'+A'B'C+A'BC'+A'BC

DNF of f'(A,B,C)
AB'C'+AB'C+ABC'+ABC

Fig. 2. The truth-table corresponding to a Boolean function f and its corresponding full disjunctive normal form (full DNF): $A'B'C' + A'B'C + A'BC' + A'BC$.

minimal expression in disjunctive normal form corresponding to any Boolean function, remains a common and reliable tool in the digital logic design industry. Since our goal is to provide a brief and general introduction to this method rather than a complete exposition, we refer the reader to Quine (1955) and Mendelson (1970) for detailed proofs showing that it works.

The method consists of the following three general steps each of which will be explained in detail:

(1) For any arbitrary truth function f produce the full disjunctive normal form expansion of f.
(2) Generate the prime implicants (PIs) of the expansion by eliminating as many literals as possible from it by systematically combining expressions that differ by one literal (more specifically, by systematically applying the rule $xy + xy' = x$).
(3) Generate a PIs table to determine the minimal number of PIs that when logically added produce an equivalent Boolean expression in DNF with a minimum number of literals.

To explain step one, we use Fig. 2. We begin with an arbitrary function f and produce its full DNF expansion (if it is not already in full DNF) by selecting only the rows of its corresponding truth table for which the function evaluates to 1. In our example, these are the first four rows of the truth table. Now, for each row, we can form a conjunction or product of literals corresponding to the value of each variable in the row so that a variable with a value of zero is negated and with a value of 1 is not negated. For instance, when $A = 0$, $B = 0$, and $C = 1$, we form the conjunction $A'B'C$. After forming the four conjunctions (which correspond to the four positive examples of a Boolean concept) from the four rows for which the function evaluated to 1, we add them to obtain the final full DNF formula. Now that we have a sum of products, we can proceed to applying step two. Note that the Boolean concept types are already in full disjunctive normal form.

Step two introduces the notion of a PI. The implicant of a Boolean formula **F** in DNF is a literal (i.e., a variable or the negation of a variable) or conjunction of literals $\mathscr{C}$ (where no variable appears more than once), such that $\mathscr{C}$ logically implies **F**. By the law of addition of Boolean algebra, any disjunct of a formula in DNF is clearly an implicant of the formula. A PI, on the other hand, is a minimal implicant of **F**. More specifically, it is an implicant of **F** that ceases to be so if any of its literals is removed. Thus, for example, let **F** be $ab + ab'c + a'b'c$, then ab is a PI of **F** since ab implies **F** by the law of addition, while a alone and b alone do not logically imply **F**. The importance of this notion can best be appreciated by the fact that the minimal DNF expression for a formula **F** is a disjunction of one or more PIs of **F**. This assertion, proved by Quine (1955) and referred to henceforth as the PI theorem, led to the determination of the recursive procedure for finding *all* PIs of a formula in full DNF, a key step in the QM minimization method.

The procedure finds all the PIs of a formula $\mathbf{F} = \varphi_1 + \cdots + \varphi_k$ in full DNF by executing the following three steps:

1. List $\varphi_1, \ldots, \varphi_k$.
2. If two fundamental conjunctions ψ and χ in the list are the same except that ψ contains a certain variable unnegated while χ contains the same variable negated, then add to the list the fundamental conjunction obtained by eliminating from ψ the variable in which ψ differs from χ. Place check marks next to ψ and χ.

(*footnote continued*)
Karnaugh (1953). Because Karnaugh maps are a pictorial way of deriving minimal DNF expressions and are thus dependent on the pattern matching abilities of humans for their effectiveness, they are prone to human error and are not as reliable as computer minimization programs. Computer minimization programs that emulate the visual approach of Karnaugh Maps are often based on the QM method (about to be introduced) since the two methods are fundamentally equivalent.

3. Repeat the process indicated in step 2 until it can no longer be applied. Fundamental conjunctions which already have been checked are used again in the applications of step 2. The unchecked fundamental conjunctions in the resulting final list are the PIs of **F**.

In respect to steps 2 and 3 above, notice that they merely assert the exhaustive application of the reduction rule $xy + xy' = x$ to the entire list of conjuncts found in the full DNF expression, and then to the list resulting from the application of the rule to the first list, and likewise for each similarly generated list. But why this rule? Because we can add conjuncts to obtain a simpler expression only when they differ by one variable or bit. To show why this is the case, consider two conjunctions φ and ϕ from some full DNF expression. Let $x_1 \cdot x_2 \cdot \ldots \cdot x_n$ and $x'_1 \cdot x'_2 \cdot \ldots \cdot x'_n$ be the conjunctions of variables at which φ and ϕ, respectively, differ: so that for any i, x_i of φ differs in sign only from x'_i of ϕ. Although it follows that $x_1 + x'_1 = 1$, it is not necessarily true that for any integer $n \geqslant 2$, $x_1 \cdot x_2 \cdot \ldots \cdot x_n + x'_1 \cdot x'_2 \cdot \ldots \cdot x'_n = 1$ since for any mixed assignment of zeros and ones to either $x_1 \cdot x_2 \cdot \ldots \cdot x_n$ or $x'_1 \cdot x'_2 \cdot \ldots \cdot x'_n$, it follows that $x_1 \cdot x_2 \cdot \ldots \cdot x_n + x'_1 \cdot x'_2 \cdot \ldots \cdot x'_n = 0$.

Having explained the minimization rule, we can now derive the minimal DNF expression for the Boolean concept type 3(4)-3 (already in full DNF) using the QM algorithm. First, we need to find all of the PIs in the Boolean formula $a'b'c' + a'b'c + a'bc' + ab'c$ by first listing all of its conjuncts: namely, $a'b'c'$, $a'b'c$, $a'bc'$, and $ab'c$ under column 1 of Table 1 in Fig. 3.

Following the second step of the algorithm, we apply the reduction rule by identifying all possible pair-wise combinations of conjuncts that differ by one variable (i.e., for which one variable is complemented in one and not complemented in the other conjunct) and removing the differing variable. We then place a check mark by each conjunct that partakes in such reduction. For example, in Fig. 3, $a'b'c'$ and $a'b'c$ differ in c' and c, thus we remove, respectively, c' and c from the two conjunctions. We then place a check mark by $a'b'c'$ and $a'b'c$ indicating their participation in the reduction. Finally, we list the newly obtained conjunction $a'b'$ in column 2. Similarly, we compare $a'b'c'$ with $a'bc'$ and remove b and b', and compare $ab'c$ with $a'b'c$ and remove a and a'. Once we have listed all the reduced conjuncts in column 2 (obtained by applying the reduction rule exhaustively to column 1), we reapply the reduction rule exhaustively on them. But as we can see, there is nothing else we can reduce; therefore, the unchecked conjunctions are the PIs of our formula.

Now we find out which disjunction of PIs are minimal DNFs by constructing a table of PIs with a column for each disjunct of the formula in question and a row for each PI as shown in Table 2 of Fig. 3. We place check marks in every cell corresponding to a PI–disjunct pair where the PI is a part of the disjunct of the formula. We now define the core of a PI table as the set of PIs each of which is the only component of some disjunct in the formula we are attempting to minimize. In other words, the core is the set of PIs corresponding to the columns with single checks. To elucidate, we draw a circle around such check marks. In our example the core consists of the PIs $a'c'$ and $b'c$. The core operation on PI tables is an important concept since the elements of the core must, according to the PI theorem introduced above, be disjuncts of the minimal DNF formula for the formula in question.

Having identified the core, we need to determine which elements of the core represent or cover every single disjunct. In other words, we need to determine whether each of the disjuncts of the original formula contains some element in the core. In our example, we determine these elements by drawing a square around each check that lies in a row with a core element (an encircled check). Since in our example above all PIs in the core cover all the disjuncts of the formula, our minimal formula is precisely the disjunction of the core elements, namely $a'c' + b'c$. Notice that this minimal expression has a complexity value of four and not six as reported in Feldman (2000, 2003a).

Note that sometimes we may obtain more than one minimal DNF following this procedure or perhaps a DNF that can be factored. In the next section we shall take advantage of this fact and extend the QM method by allowing only one, precisely defined, form of factorization in order to obtain even simpler formulae. For example, consider the derivation of the Boolean concept type 3(4)-4. As before, we begin by generating all of its PIs and their corresponding PI table as illustrated in Fig. 4. Notice that coverage is achieved by the use of all three PIs $a'b'$, $a'c'$, and $b'c'$. We then add these three PIs to obtain the minimal DNF expression $a'b' + a'c' + b'c'$. However, as we pointed

1	2
A'B'C' ✓	A'B'
A'B'C ✓	A'C'
A'BC' ✓	B'C
AB'C ✓	

Table 1

	A'B'C'	A'B'C	A'BC'	AB'C
A'B'	✓	✓		
A'C'	☑		⊘	
B'C		☑		⊘

Table 2

Fig. 3. Prime implicants and PI table for minimization of case 3(4)-3.

1	2
A'B'C' ✓	A'B'
A'B'C ✓	B'C'
A'BC' ✓	A'C'
AB'C' ✓	

	A'B'C'	A'B'C	A'BC'	AB'C'
A'B'	☑	⊘		
A'C'	☑		⊘	
B'C'	☑	☑		⊘

Fig. 4. Prime implicants and PI table for minimization of case 3(4)-4.

1	2
A'B'C'D' ✓	A'B'D'
A'B'C'D ✓	A'B'C'
A'B'CD' ✓	A'C'D
A'BC'D ✓	

	A'B'C'D'	A'B'C'D	A'B'CD'	A'BC'D
A'B'D'	☑		⊘	
A'B'C'	✓			
A'C'D		⊘		⊘

Fig. 5. Prime implicants and PI table for minimization of case 4(4)-3.

	A	B	C	(~A & ~B & ~C)	v	(~A & ~B & C)	v	(~A & B & ~C)	v	(A & ~B & C)	<=>	(~A & ~C)	v	(~B & C)
0)	0	0	0	1		1		1		0	1	1		1
1)	0	0	1	1		1		0		0	1	1		1
2)	0	1	0	1		0		1		0	1	1		0
3)	0	1	1	1		0		0		0	1	1		0
4)	1	0	0	0		0		0		0	1	0		1
5)	1	0	1	0		0		0		0	1	0		1
6)	1	1	0	0		0		0		0	1	0		0
7)	1	1	1	0		0		0		0	1	0		0
				^ 1 3	^ 2	^ 5	^ 4	^ 1 0	^	^ 6	^ 8	^ 7	^ 9	^ 1 5	^ 1	^ 2	^ 4	^ 3	^ 2	^ 1 9	^ 7	^ 6	^ 8	^ 2 6	^ 2	^ 0	^ 2	^ 1	^ 2	^ 5	^ 3	^ 4

	A	B	C	(~A & ~B & ~C)	v	(~A & ~B & C)	v	(~A & B & ~C)	v	(A & ~B & ~C)	<=>	((~A & (~B v ~C))	v	(~C & ~B))
0)	0	0	0	1		1		1		0	1	1		1
1)	0	0	1	1		1		0		0	1	1		1
2)	0	1	0	1		0		1		0	1	1		0
3)	0	1	1	1		0		0		0	1	1		0
4)	1	0	0	0		0		0		0	1	0		1
5)	1	0	1	0		0		0		0	1	0		1
6)	1	1	0	0		0		0		0	1	0		0
7)	1	1	1	0		0		0		0	1	0		0
				^ 1 3	^ 2	^ 5	^ 4	^ 1 0	^	^ 6	^ 8	^ 7	^ 9	^ 1 5	^ 1	^ 2	^ 4	^ 3	^ 2	^ 1 7	^ 6	^ 9	^ 8	^ 3 0	^ 2	^ 1	^ 5	^ 2	^ 4	^ 3	^ 2	^ 9	^ 6	^ 8	^ 7

Fig. 6. Truth tables for the derived minimal formulae for Cases 3(4)-3 and 3(4)-4. The order of evaluation is specified at the bottom of the table. "&", "∨", and "< = >" stand respectively for conjunction, disjunction, and logical equivalence. The enclosed rectangle in each table verifies the equivalence between the concept type and its reduced formula.

out, we may be able to reduce this minimal DNF further through a specific form of factorization. We do so by applying the distributive law of Boolean multiplication over addition and get instead $a'(b'+c')+b'c'$ which has a complexity value of five, not six as reported in Feldman (2000, 2003a,b).

For our final example, Fig. 5 displays a list of all the PIs of 4(4)-3 and the corresponding PI table. Again, the resulting coverage is achieved by two of the three PIs, namely, $a'b'd'$ and $a'c'd$ which are then added to each other to form the minimal DNF expression $a'b'd'+a'c'd$. After factorization, this yields the minimal expression $a'(b'd'+c'd)$ which has a complexity value of five, not seven as reported in Feldman (2000, 2003a).

Finally, to verify that the three derivations above are correct, we verify the equivalence between the found minimal formulae and the original "full" DNF formulae using the truth tables in Figs. 6 and 7. The truth tables were generated by Truth Table Constructor 3.0, an Internet applet by Brian S. Borowski. Now that we have covered

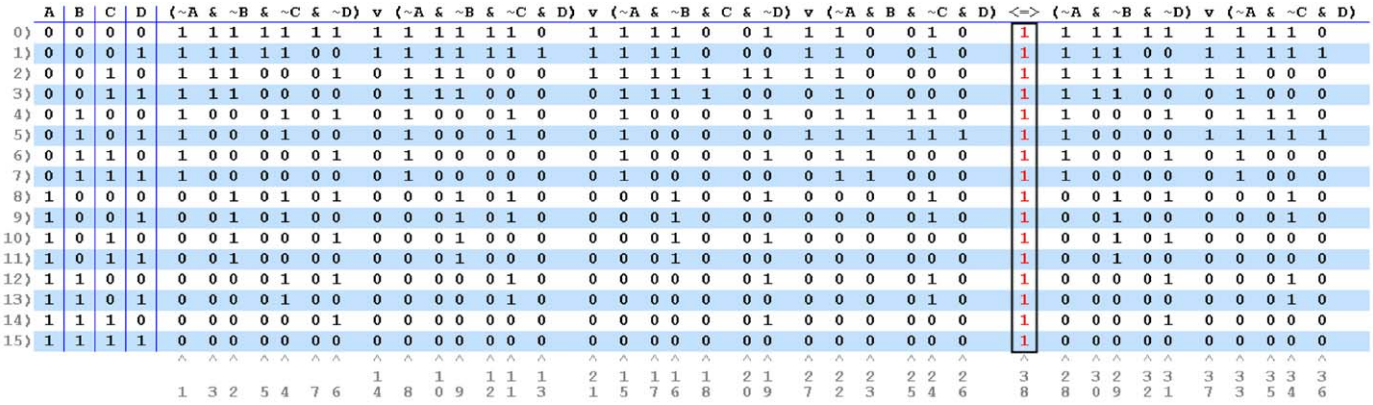

Fig. 7. Truth table for the derived formula for Case 4(4)-3. The order of evaluation is specified at the bottom of the table. "&", "∨", and "$<=>$" stand respectively for conjunction, disjunction, and logical equivalence. The enclosed rectangle in the table verifies the equivalence between the concept type and its reduced formula.

the basics behind the QM method we proceed to derive the minimal formulae for the 41 cases in Feldman (2000).

3. Derivations of the minimal formulae for the 41 cases using the QMV heuristic

In the previous section we used the QM method for deriving the minimal DNF formulae corresponding to the 3(4)-3 and 3(4)-4 concept types—two of the six SHJ types. It was also shown that by using the QM method along with the application of the distributive law of Boolean algebra, we can improve our results and obtain even simpler expressions. However, we also found out that using tables to apply the QM method can be a tedious and error prone process except for the simplest cases. Unquestionably, the method is best executed by a computer program; in fact, therein lies its strength.

Accordingly, for the more complex concept types, we employed the aid of an Internet applet by Eric Gossett of Bethel University that implements the QM Method on Boolean functions of up to 10 variables. With its help, along with the application of a single law of Boolean algebra known as the distributive law of Boolean multiplication over addition (i.e., $\alpha\phi + \alpha\psi = \alpha(\phi + \psi)$) and a simple heuristic strategy which we shall articulate next, we devised a heuristic procedure (henceforth referred to as QMV) which when consistently applied to each of the 41 cases studied in Feldman (2000) yields (for some concept types) formulae of markedly shorter length lower complexity than those derived by Feldman, the differences being in the range of one to two literals.

The simple strategy we speak of is to apply the QM method to both the full DNF of the function f and the full DNF of its dual f'. We do this since by applying the QM method to the dual of f we can generate different PIs from those of f that possibly yield expressions of lower complexity (either by default or via factorization). Another way of understanding this strategy emerges when considering concept types where, unlike the SHJ cases, the number of positive examples and negative examples differ (in other words, for types with down and up parities). When in up parity these non-symmetrical concept types have full DNF descriptions of shorter length than their counterparts in down parity. Thus, each description can yield different PIs and possibly final expressions of different length.

However, in using this strategy care must be taken in restoring equivalence to the original function f by negating the resulting minimal DNF corresponding to f'. That is, if we denote the minimal DNF of f' as MDNF(f') then we must negate it to get the expression [MDNF(f')]′ which is equivalent to the original function f. An example of the DNF of f' is given in Fig. 2. Note that this is merely the DNF expansion of the function obtained by switching the zeros and the ones in the $f(A, B, C)$ column.

Now we can define the QMV heuristic procedure which we applied consistently to the 41 concept types:

1. Generate the full DNF of f' and the full DNF of f.
2. Apply the QM method as described in the previous section to each.
3. Factor each expression obtained from applying the QM method to the full DNF of f' (we refer to the set of these expressions as category N) and factor each expression obtained from applying the QM method to the full DNF of f (we refer to the set of these expressions as category P). (The term "factor" here is precisely defined as the exhaustive application of the Boolean distributive law $\alpha\phi + \alpha\psi = \alpha(\phi + \psi)$).
4. Compare the length of the shortest expression in category N to the length of the shortest expression in category P. If the shorter expression is in category P we are done. If the shorter expression is in category N, go to step 5. If their lengths are equal, go to step 6.
5. Negate the expression obtained from category N in order to transform it into an expression equivalent to f. We are done.
6. Select the expression from category P. We are done.

With the help of the QM applet we applied this simple heuristic procedure to all 41 Boolean concept types and found nine simpler expressions than those reported by

CF-C	FDNF	FR	C	AR	C
3(4)-3	$a'b'c' + a'b'c + a'bc' + ab'c$	$a'(bc)'+ab'c$	6	$a'c'+b'c$	4
3(4)-4	$a'b'c' + a'b'c + a'bc' + ab'c'$	$a'(bc)'+ab'c'$	6	$a'(b'+c')+b'c'$	5
4(3)-5	$a'b'c'd' + a'b'cd + abc'd'$	$a'b'(c'd'+cd)+abc'd'$	10	$[a(c+b') + a'b+cd'+c'd]'$	9
4(4)-2	$a'b'c'd' + a'b'c'd + a'b'cd' + a'bc'd'$	$a'(b'(cd)'+bc'd')$	7	$a'(b'(c'+d')+c'd')$	6
4(4)-3	$a'b'c'd' + a'b'c'd + a'b'cd' + a'bc'd$	$a'(b'(cd)'+bdc')$	7	$a'(b'd' + c'd)$	5
4(4)-6	$a'b'c'd' + a'b'c'd + a'b'cd' + abc'd$	$a'b'(cd)'+abc'd$	8	$[a(b' + d')+ a'b + cd]'$	7
4(4)-10	$a'b'c'd' + a'b'c'd + a'bcd' + ab'cd$	$a'(b'c'+bcd')+ab'cd$	10	$[a'b'c + a(c' + d')+ b(c'+d)]'$	9
4(4)-15	$a'b'c'd' + a'b'cd + a'bc'd + ab'c'd$	$a'(b'(c'd'+cd)+bc'd)+ab'c'd$	13	$[a'b'c'd + (a+d')(b+c)+ ad' + bc]'$	12
4(4)-16	$a'b'c'd' + a'b'cd + a'bc'd + ab'cd'$	$a'(b'(c'd'+cd)+bc'd)+ab'cd'$	13	$[a'cd' + b'c'd + a(c'+d)+b(c+ d')]'$	12

Fig. 8. CF-C = concept family and case, FDNF = full DNF formula, FR = Feldman's reduction as reported in Feldman (2000, 2003a), AR = author's reduction, and C = complexity value.

Feldman in the 2000 and 2003 papers. Reduced expressions for the remaining 32 concept types were identical to those reported by Feldman (2003a). Note that the results obtained under Section 2, where we applied only the QM method and the distributive law, are identical to those obtained by the QMV procedure. The nine formulae are listed in Fig. 8 along with their corresponding reduced formulae as reported by Feldman. Each was verified with the aid of truth table analysis as were the cases derived under Section 2.

As stated under Section 2, the QM method alone derives minimal DNF expressions. As shown, a minimal DNF may be transformed into a simpler expression not necessarily in DNF by the application of one standard rule of Boolean algebra and a simple strategy (i.e., by the distributive law and a duality property), which is what we have done for the 41 types discussed in Feldman (2000). We have not provided a proof that the derived formulae are in fact minimal. However, given the brevity of the derived formulae for the SHJ types, we strongly believe that they are. In fact, given the brevity of these strings, a proof could be provided by a computer program that generates exhaustively all possible well formed structural combinations of literals and connectives and tests each one for equivalence to its corresponding reduced formula in Fig. 8.

4. Conclusion and discussion

Using our QMV heuristic procedure, we derived expressions for all 41 Boolean concept types investigated in the Feldman (2000) experiment. For nine of the 41 concept types, we were able to derive simpler formulae (i.e., expressions with a smaller number of literals) than those derived in Feldman (2000, 2003a). We included in this paper derivations for only three of the nine concept types for which we found a simpler minimal expression than that reported by Feldman (2000, 2003a); two of these three cases are critical ones since they are SHJ concepts types. The remaining 32 concept types yielded expressions of equal complexity to those found in Feldman (2000, 2003a). Finally, we generated truth tables to confirm the equivalence of the reduced formulae to their original unsimplified counterparts.

In our view, our findings involving the SHJ types are significant in the following sense: they determine a complexity order for the six concept types that is markedly different from their empirical order of learning difficulty. As noted in the introduction, Feldman (2000) had concluded that the Boolean complexity hypothesis predicted *precisely* the ordering of difficulty of the SHJ concept types. This conclusion was based on his derivation of approximate complexity values for the types using an (unspecified) heuristic method. This heuristic method yielded complexity values for Types I–VI of 1, 4, 6, 6, 6, and 10, respectively, which indeed corresponds precisely with the empirical difficulty ordering, $I < II < [III, IV, V] < VI$.

However, in this note we find what we believe are the actual Boolean complexity values for Types I–VI, namely: 1, 4, 4, 5, 6, and 10. According to the Boolean complexity hypothesis, the predicted difficulty ordering is therefore $I < [II, III] < IV < V < VI$, which departs notably from the empirical ordering. As has been well documented in various studies (Love & Medin, 1998; Nosofsky et al., 1994; Shepard et al., 1961), Type II is learned more rapidly than is Type III, and Types III–V are learned with essentially equal difficulty. Our derivations therefore strongly challenge the Boolean complexity account of concept-learning difficulty as far as the SHJ types are concerned.

Feldman (2000) has made some significant advances by beginning to consider the difficulty of learning numerous concept types beyond the SHJ types. Perhaps, therefore, it is inappropriate to focus solely on the results involving the SHJ types, as we have done in this paper. Indeed, we strongly agree with Feldman that alternative models of concept learning should be evaluated, eventually, with respect to their ability to account for the full gamut of concept-learning problems. Nevertheless, at the current stage of development in the field, the SHJ types have been intensively investigated by various independent researchers, and the results have served as benchmarks for numerous models of concept learning. The empirical

pattern of results has been interpreted as revealing key psychological principles that underlie human concept-learning processes (for discussion, see Kruschke, 1992; Love & Medin, 1998; Nosofsky et al., 1994; Shepard et al., 1961).

The same does not yet hold true for the new concept types investigated by Feldman (2000). The reliability and replicability of his learning data have not yet been established. Indeed, although a full discussion goes beyond the scope of this note, Feldman's method of testing the difficulty of his concept types was dramatically different than the standard method used by other investigators. The standard method uses a long-learning sequence involving the presentation of individual stimuli on a trial-by-trial basis with corrective feedback provided on each trial. By contrast, Feldman's method involved a simultaneous visual display of all members of the positive and negative concepts for a relatively brief period of time (see Feldman, 2000 for details). The potential effects of these dramatically different methods of testing concept learning have yet to be investigated. Thus, we believe that our focus on the classic results from the SHJ experiments in the present paper is justified, and that our findings place significant challenges on the current form of the Boolean complexity hypothesis.

In addition, the SHJ concept types are the only concept types of three variables that have no parity. This means that, according to Feldman (2000), unlike the rest of the three-variable concept types, the degree of learning difficulty of each of the SHJ concept types does not depend on parity. This claim gives us the opportunity to isolate Boolean complexity as hypothetically the lone contributing variable to learning difficulty. In so doing we can more clearly and easily establish how Boolean complexity alone affects learning difficulty.

Although the strong form of the Boolean complexity hypothesis fails to predict concept-learning difficulty, it is interesting to consider whether modified versions might be more successful. As presented by Feldman (2000), Boolean complexity is defined according to the formal dictates of mathematical logic, i.e., it is defined as the shortest Boolean propositional formula that is equivalent to the concept. An alternative idea is that humans might rely on some form of psychological heuristic to construct Boolean propositional formulae for representing concepts and that it is the complexity of these psychological strings that predict concept-learning difficulty. The search for such a psychologically plausible heuristic is perhaps what Feldman (2003b) had in mind in his more recent article "The Simplicity Principle in Human Concept Learning." There, he suggests that the code or representation of complexity minimization in terms of conventional logical operators may not be particularly psychologically plausible. Moreover, he concludes that an essential goal for future research is to identify the underlying cognitive code that is actually employed by human learners.

We strongly agree with Feldman (2003b) that the search for such a "code" would provide a valuable direction for future work in concept learning. Clearly, to make progress, it is essential that explicit formal proposals be provided of the heuristics or codes that humans may use, or else these weaker versions of the complexity hypothesis cannot be tested. Feldman's (in press) "Algebra of human concepts" is a recent attempt at achieving this aim. In the meantime, the present work points clearly to significant challenges faced by the strong version of the Boolean complexity hypothesis and reinforces Feldman's (2003b) suggestion for future research directions.

Acknowledgment

I would like to thank Robert Nosofsky and John Kruschke for their suggestions and encouragement while preparing this manuscript. This work was supported in part by the National Institute of Mental Health Grant R01MH48494. Correspondence concerning this article should be addressed to: Ronaldo Vigo, Cognitive Science Department, 819 Eigenmann Hall, Indiana University at Bloomington, Bloomington, IN 47406, rvigo@indiana.edu.

Author's note. Recently, it has been brought to my attention that Daniel Lafond, Yves Lacouture, and Guy Mineau of the Universit Laval of Quebec, are about to submit a paper of an experimental nature that includes results generated by a Boolean minimization program from UC Berkeley. Although they do not discuss the heuristic(s) employed by the program, they report Boolean complexity values identical to those obtained by the QM based heuristic employed in my paper. However, the minimal expressions themselves are not always identical.

References

Aiken, H. H., the staff of the computation laboratory of Harvard University (1951). Synthesis of electronic computing and control circuits. *The annals of the computation laboratory of Harvard University* (Vol. XXVII). Cambridge, MA: Harvard University Press.

Bourne, L. E. (1966). *Human conceptual behavior*. Boston: Allyn and Bacon.

Bourne, L. E. (1974). An inference model for conceptual rule learning. In R. Solso (Ed.), *Theories in cognitive psychology* (pp. 231–256). Washington: Erlbaum.

Feldman, J. (2000). Minimization of Boolean complexity in human concept learning. *Nature*, *407*(October), 630–633.

Feldman, J. (2003a). A catalog of Boolean concepts. *Journal of Mathematical Psychology*, *47*(1), 75–89.

Feldman, J. (2003b). The simplicity principle in human concept learning. *Current Directions in Psychological Science*, *12*(6), 227–233.

Feldman, J. (in press). Algebra of Human Concepts. *Journal of Mathematical Psychology*.

Higonnet, R. A., & Grea, R. A. (1958). *Logical design of electrical circuits*. New York: McGraw-Hill.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review*, *99*, 22–44.

Love, B. C., & Medin, D. L. (1998). SUSTAIN: A model of human category learning. *Proceedings of the fifteenth national conference on artificial intelligence* (AAAI-98) (Vol. 15, pp. 671–676), USA.

McCluskey, E. J. (1956). Minimization of Boolean functions. *Bell System Technology Journal*, *35*, 1417–1444.

Mendelson, E. (1970). Boolean Algebra of Switching Circuits, McGraw-Hill.

Monnasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., & Troyansky, L. (1999). 2 + p-SAT: Relation of typical-case complexity to the nature of the phase transition. *Random Structures and Algorithms*, *15*, 414–435.

Nosofsky, R. M., Gluck, M. A., Palmeri, T. J., McKinley, S. C., & Glauthier, P. G. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition*, *22*(3), 352–369.

Quine, W. V. (1955). A way to simplify truth functions. *American Mathematical Monthly*, *62*, 627–631.

Shepard, R., Hovland, C. L., & Jenkins, H. M. (1961). Learning and memorization of classifications. *Psychological Monographs: General and Applied*, *75*(13), 1–42.