Duce, an oracle based approach to constructive induction

Stephen Muggleton
Turing Institute,
36 North Hanover Street,
Glasgow,
G1 2AD,
UNITED KINGDOM.

Abstract

Duce[1] is a Machine Learning system which suggests high-level domain features to the user (or *oracle*) on the basis of a set of example object descriptions. Six transformation operators are used to successively compress the given examples by generalisation and feature construction. In this paper Duce is illustrated by way of its construction of a simple animal taxonomy and a hierarchical parity checker. However, Duce's main achievement has been the restructuring of a substantial expert system for deciding whether positions within the chess endgame of King-and-Pawn-on-a7 v. King-and-Rook (KPa7KR) are won-for-white or not. The new concepts suggested by Duce for the chess expert system hierarchy were found to be meaningful by the chess expert Ivan Bratko. An existing manually created KPa7KR solution, which was the basis of a recent PhD thesis [20], is compared to the structure interactively created by Duce.

A second major expert system application of Duce was made within a diagnostic field of neuro-psychology. This is described in Section 8.

1   Introduction

It is well recognised [4] that acquisition of expert knowledge is the major "bottleneck" in expert system development. However, Michalski and Chilausky [8] and later

Quinlan [15] have shown that this bottleneck can be considerably eased by generalising low-level data to form high-level rules. Shapiro [20] extended this methodology to deal effectively with extensive bodies of knowledge by employing structured programming techniques. Thus the expert structures the knowledge in a top-down fashion manually, and then provides examples which can be used to inductively generate each module in the hierarchy separately. Using this technique, Shapiro and Kopec created knowledge structures for correctly deciding a forced win for white in any position within the chess endgames of King-and-Pawn v. King (KPK) and King-and-Pawn-on-a7 v. King-and-Rook (KPa7KR). Both solutions were completely verified by exhaustive computation. However, using an information theoretic approach Shapiro showed[20] that 84% of the endgame knowledge was provided by the expert in the creation of the knowledge structure. Thus almost all the work was still being done by the expert rather than by the machine. In an attempt to overcome this structuring bottleneck Paterson [13] tried to use the statistical clustering algorithm CLUSTER [9] to automatically restructure the knowledge for the simpler of the two endgames, KPK. Paterson's results were not promising, with the machine's suggested hierarchy not having any significance to domain experts.

In the context of Machine Learning, Michalski [7] has called the problem of originating terms *constructive induction*. CLUSTER [9], perhaps the best known constructive induction algorithm, uses a statistical clustering technique to group objects into conceptual clusters. Each object is initially described in terms of a vector of primitive attribute values. Objects are grouped using a heuristic inter-object distance metric. Rendell [16], and Fu and Buchanan [5] describe alternative similarity-based approaches to creating taxonomic hierarchies which work on much the same basis as CLUSTER.

2 Transformation-based induction

In Duce the approach to constructive induction differs considerably from that of Michalski and Stepp [9], Rendell [16], and Fu and Buchanan [5], and can be more easily compared to the deductive transformational programming techniques of Burstall and Darlington [2]. Burstall and Darlington, and later Dershowitz [3], suggest that *deductive* program synthesis can be carried out by gradual truth-preserving transformations of a program specification. At first sight, these techniques seem not to be applicable to *inductive* inference, which by definition progresses by performing non-truth-preserving generalisations of the supplied training set. However, if each inductive transformation is tested against an oracle which ensures the validity of any transformation, any such inductive transformation is as legal and safe as its deductive counterparts. This use of an oracle is closely related to Sammut and Banerji's [19] method of learning concepts by asking questions. Indeed, one of the generalisation operators described in the next section is directly due to Sammut and Banerji.

Constructive induction carries out transformations which introduce new terms into the learner's vocabulary. Though such transformations can be truth-preserving, they are not what might be called semantics-preserving. Thus the primary concern in constructive induction should not be "how can new terms be introduced into the vocabulary?" but rather "how can meaningful new terms be introduced?". Again by using an oracle to either name or reject machine-suggested concepts, the difficult philosophical problems involved in defining the word meaningful can be sidestepped. Given a meaningful and valid training set, every transformation of which is both meaningful and valid (by agreement of the oracle), the resultant rule set will be meaningful and valid.

3 Operators

Duce takes as input a set of conjunctive productions, or rules, in a form close to that of disjunctive-normal-form propositional calculus. Six operators are employed to progressively transform subsets of the rule base. These operators are described below.

1. **Inter-construction**. This transformation takes a group of rules, such as

$$X \leftarrow B \land C \land D \land E \tag{1}$$
$$Y \leftarrow A \land B \land D \land F \tag{2}$$

and replaces them with the rules

$$X \leftarrow C \land E \land Z? \tag{3}$$
$$Y \leftarrow A \land F \land Z? \tag{4}$$
$$Z? \leftarrow B \land D \tag{5}$$

Here the rule for the new concept Z? (5) is the most specific generalisation of the rules for X (1) and Y (2).

2. **Intra-construction.** This is simply the distributive law of Boolean equations. Intra-construction takes a group of rules all having the same rule head, such as

$$X \leftarrow B \land C \land D \land E \tag{6}$$
$$X \leftarrow A \land B \land D \land F \tag{7}$$

and replaces them with

$$X \leftarrow B \land D \land Z? \tag{8}$$
$$Z? \leftarrow C \land E \tag{9}$$
$$Z? \leftarrow A \land F \tag{10}$$

Note that while operator 1, inter-construction, could legitimately be applied to rules 6 and 7, the result would be less compact.

3. **Absorption.** This operator is due to Sammut and Banerji [19], who use it to generate recursive Prolog clauses. Even though recursion is not meaningful within propositional calculus, this operator can be employed profitably in generalising rule sets. Given a set of rules, the body of one of which is completely contained within the bodies of the others, such as

$$X \leftarrow A \land B \land C \land D \land E \tag{11}$$
$$Y \leftarrow A \land B \land C \tag{12}$$

one can hypothesise

$$X \leftarrow Y \land D \land E \tag{13}$$
$$Y \leftarrow A \land B \land C \tag{14}$$

Note that the preconditions for applying this operator are stronger than those for applying inter-construction. Also, if rule 12 were the only rule with rule head Y, then the new rule is necessarily valid. Otherwise it is a generalisation and must be verified by the oracle. In general, asking the oracle unnecessary questions can be avoided by first attempting to answer the question deductively from the rule base.

4. **Identification.** The identification operator is again a potential generalisation, whose preconditions are stronger than those of intra-construction. A set of rules which all have the same head, the body of at least one of which contains exactly one symbol not found within the other rules, such as

$$X \leftarrow A \land B \land C \land D \land E \tag{15}$$
$$X \leftarrow A \land B \land Y \tag{16}$$

can be replaced by the rules

$$X \leftarrow A \wedge B \wedge Y \qquad (17)$$
$$Y \leftarrow C \wedge D \wedge E \qquad (18)$$

5. **Dichotomisation.** This operator works on sets of mixed positive and negative examples. Thus a set of rules which contain positive and negative heads, and which all have some common symbols within the bodies, such as

$$X \leftarrow A \wedge B \wedge C \wedge D \qquad (19)$$
$$\overline{X} \leftarrow A \wedge C \wedge J \wedge K \qquad (20)$$
$$\overline{X} \leftarrow A \wedge B \wedge C \wedge L \qquad (21)$$

are replaced with the rules

$$X \leftarrow A \wedge C \wedge Z? \qquad (22)$$
$$\overline{X} \leftarrow A \wedge C \wedge \overline{Z?} \qquad (23)$$
$$Z? \leftarrow B \wedge D \qquad (24)$$
$$\overline{Z?} \leftarrow J \wedge K \qquad (25)$$
$$\overline{Z?} \leftarrow B \wedge L \qquad (26)$$

where the replacement is dependent on the oracle naming Z?. Dichotomisation is a generalisation of the the way that ID3 [15] creates the internal nodes of decision trees.

6. **Truncation.** The truncation operator, like dichotomisation, is intended for use with rule sets containing positive and negative examples of the same concept. However, truncation generalises by dropping conditions. A set of rules which all contain the same head, such as

$$X \leftarrow A \wedge B \wedge C \wedge D \qquad (27)$$
$$X \leftarrow A \wedge C \wedge J \wedge K \qquad (28)$$

is replaced by

$$X \leftarrow A \wedge C \qquad (29)$$

This operator generalises in a similar manner to that of the AQ learning algorithms [8]. Its use is restricted by the precondition that the resultant rule (29) must not clash (i.e. be inconsistent) with any other rule within the rule base. Of all the operators truncation is the only one which reduces the number of rules. All other operators compact the rules by shortening the average rule length.

4 The search algorithm

For any state of a rule base, there are many possible operator applications. Any subset of rules within the rule base R is a candidate for the application of one of the 6 operators. Thus the search-space for the "best" operator application is of size $2^{|R|}$, the size of the power set of R. What is meant by a "good" operator application? Since each of the operators can reduce the number of symbols in the rule base, Duce searches for the application which produces the largest symbol reduction, i.e. Occam's razor is applied. If each rule is taken as having a symbol size equal to the number of conjunctive terms in the rule body plus one (for the rule head), then for each operator there exists an equation which can be used to predict the exact symbol reduction for any operator. Let R' be a subset of the rule base R and I' be a common subset of all the bodies of rules within R'. In the following equations $V_{operator}$ is the symbol reduction produced when the operator is applied to R'. The total number of symbols within the rule set R' is written as $total(R')$. The symbol reduction equations are

$$
\begin{aligned}
V_{Inter} &= (|I'| - 1).(|R'| - 1) - 2 \\
V_{Intra} &= |I'|.(|R'| - 1) - 2 \\
V_{Absorp} &= (|I'| - 1).(|R'| - 1) \\
V_{Ident} &= |I'|.(|R'| - 1) \\
V_{Dichot} &= |I'|.(|R'| - 2) - 4 \\
V_{Trunc} &= total(R') - |I'| - 1
\end{aligned}
$$

Note that V_{Op} can take a zero or negative value, in which case there is no symbol reduction. Searching for the best operator application is clearly intractable. Moreover, there is no guarantee that such an operator application, once found, would be acceptable to the oracle. In Duce the next operator application is chosen using a best first search through the power set of the symbols in the rule base R. Let a subset of symbols I' be found among the bodies of the rule set R', where R' is the largest subset of R containing I'. The operator application $apply(Op, I', R')$, using operator Op, is only suggested to the oracle when some I' has been found for which V_{Op} is locally maximal. Any rejection of an operator application by the oracle leads to continued search. Transformations are carried out iteratively until no further operations can reduce the rule base size further. At termination, by nature of the operators, almost all symbols occur within a restricted number of rules. Thus, although the termination condition requires searching the entire remaining space, the search space has shrunk to manageable proportions. Since only operations which reduce the number of symbols are applied termination is guaranteed.

5 Animal taxonomy

This section illustrates the behaviour of Duce when creating a simple animal taxonomy. Figure 1 shows the set

(blackbird   [beak t] ∧ [colour black] ∧ [legs 2] ∧ [tail t]
  ∧ [wings t])   eg (blockhead-the-blackbird)
(chimp   [colour brown] ∧ [hairy t] ∧ [legs 2] ∧ [tail t]
  ∧ [wings f])   eg (maggie-the-chimp)
(eagle   [beak t] ∧ [colour golden] ∧ [legs 2] ∧ [tail t]
  ∧ [wings t])   eg (egg-the-eagle)
(elephant   [colour grey] ∧ [legs 4] ∧ [size big] ∧ [tail t]
  ∧ [trunk t] ∧ [wings f]) eg (adult-elephant)
(elephant   [colour grey] ∧ [legs 4] ∧ [size small] ∧ [tail t]
  ∧ [trunk t] ∧ [wings f]) eg (baby-elephant)
(falcon   [beak t] ∧ [colour brown] ∧ [legs 2] ∧ [size big] ∧
  [tail t] ∧ [wings t]) eg (flap-the-falcon)
(gorilla   [colour black] ∧ [hairy t] ∧ [legs 2] ∧ [tail f]
  ∧ [wings f])   eg (ronnie-the-gorilla)
(lemur   [colour grey] ∧ [legs 2] ∧ [tail t]
  ∧ [wings f])   eg (lemur-alone)
(man   [colour brown] ∧ [hairy f] ∧ [legs 2] ∧ [size big] ∧
  [tail f] ∧   [wings f]) eg (harry-the-hamite)
(man   [colour pink] ∧ [hairy f] ∧ [legs 2] ∧ [size small] ∧
  [tail f] ∧   [wings f]) eg (clap-the-caucasian)
(sparrow   [beak t] ∧ [colour brown] ∧ [legs 2] ∧
  [size small] ∧ [tail t] ∧ [wings t]) eg (sparky-the-sparrow)

Figure 1: Initial set of animal examples

!- induce
   TRUNCATION – (-12)
   Is (elephant   [legs 4]) a valid rule? (y/n/i) n
   TRUNCATION – (-11)
   Is (elephant   [legs 4] ∧ [wings f]) a valid rule? (y/n/i) n
   TRUNCATION – (-11)
   Is (elephant   [legs 4] ∧ [trunk t]) a valid rule? (y/n/i) y
!- induce
   TRUNCATION – (-9)
   Is (man   [hairy f] ∧ [legs 2] ∧ [tail f] ∧ [wings f])
                    a valid rule? (y/n/i) y
!- induce
   INTER-CONSTRUCTION – (-1)
    Rule:
   (?   [legs 2] ∧ [wings f])
    What shall I call ⟨?⟩? (name/n/i) primate
!- induce
   INTER-CONSTRUCTION – (-7)
    Rule:
   (?   [beak t] ∧ [legs 2] ∧ [tail t] ∧ [wings t])
    What shall I call ⟨?⟩? (name/n/i) bird
!- induce
   No applicable transformation

Figure 2: Animal taxonomy session

(bird   [beak t] ∧ [legs 2] ∧ [tail t] ∧ [wings t]) eg (blockhead-the-blackbir
   egg-the-eagle flap-the-falcon sparky-the-sparrow)
(blackbird   bird ∧ [colour black]) eg (blockhead-the-blackbird)
(chimp   primate ∧ [colour brown] ∧ [hairy t] ∧ [tail t]) eg (maggie-the-ch
(eagle   bird ∧ [colour golden]) eg (egg-the-eagle)
(elephant   [legs 4] ∧ [trunk t]) eg (adult-elephant baby-elephant)
(falcon   bird ∧ [colour brown] ∧ [size big]) eg (flap-the-falcon)
(gorilla   primate ∧ [colour black] ∧ [hairy t] ∧ [tail f]) eg (ronnie-the-gori
(lemur   primate ∧ [colour grey] ∧ [tail t]) eg (lemur-alone)
(man   primate ∧ [hairy f] ∧ [tail f]) eg (clap-the-caucasian harry-the-han
(primate   [legs 2] ∧ [wings f]) eg (maggie-the-chimp clap-the-caucasian
   ronnie-the-gorilla harry-the-hamite lemur-alone)
(sparrow   bird ∧ [colour brown] ∧ [size small]) eg (sparky-the-sparrow)

Figure 3: Resultant animal rule base

of example animal descriptions given to Duce. In English
the first example says

> A blackbird has a beak, is black, has two legs,
> a tail and wings. "blockhead-the-blackbird" is
> an instance of the "blackbird" concept.

Note the inclusion of the instance set *(blockhead-the-
blackbird)* within the rule. This can be used as a pow-
erful tool for illustrating the meaning of new rules and
concepts.

Figure 2 shows user interaction for this example set.
User input is shown in bold type. When asked to in-
duce, Duce searches for an operation and suggests an
application of the truncation operator which will save
12 symbols. The operation is valid if and only if every-
thing having four legs is an elephant. The user can ei-
ther answer affirmatively ("y"), negatively ("n") or ask
for illustrative examples ("i"). If Duce is asked for il-
lustrative examples it lists the instances adult-elephant
and baby-elephant. The suggestion, although consistent
with the limited universe of the examples, is too general,
and is rejected. Duce continues its search and finds a
slightly less advantageous truncation, which would save
11 symbols. The new suggestion, that anything with four
legs and no wings is an elephant is similarly rejected.
There is no particular mechanism for specialising over-
generalised hypotheses. This is merely a by-product of
the search mechanism. On the third attempt, Duce ques-
tions whether everything that has four legs and a trunk
is an elephant. Since this is affirmed, Duce replaces all
elephant rules by the new more general rule and returns
to the "!-" prompt.

The second generalisation, concerning man is accepted
first time around, producing a saving of 9 symbols. In
the third interaction Duce finds that using the intercon-
struction operator, one symbol can be saved by defining
a new concept for all things which have two legs and
no wings. The user can either reject this new concept
("n"), ask for illustrative examples ("i"), or give a name
for the concept. The name "primate" is given to the con-

(even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ [v5 t] ∧ [v6 t] ∧
   [v7 t] ∧ [v8 t]) eg (tttttttt)
(even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ [v5 f] ∧ [v6 f] ∧
   [v7 f] ∧ [v8 f]) eg (ttttffff)
(even   [v1 f] ∧ [v2 f] ∧ [v3 t] ∧ [v4 t] ∧ [v5 f] ∧ [v6 f] ∧
   [v7 f] ∧ [v8 f]) eg (ffttffff)
(even   [v1 f] ∧ [v2 f] ∧ [v3 f] ∧ [v4 f] ∧ [v5 f] ∧ [v6 f] ∧
   [v7 f] ∧ [v8 f]) eg (ffffffff)
(¬even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ [v5 t] ∧ [v6 t] ∧
   [v7 t] ∧ [v8 f]) eg (tttttttf)
(¬even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ [v5 f] ∧ [v6 f] ∧
   [v7 f] ∧ [v8 t]) eg (ttttfft)
(¬even   [v1 t] ∧ [v2 f] ∧ [v3 t] ∧ [v4 t] ∧ [v5 f] ∧ [v6 f] ∧
   [v7 f] ∧ [v8 f]) eg (tttffff)
(¬even   [v1 f] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ [v5 t] ∧ [v6 t] ∧
   [v7 t] ∧ [v8 t]) eg (fttttttt)

Figure 4: Even-parity examples

cept. Duce goes on to suggest another new concept for
all things which have a beak, two legs, a tail and wings.
This concept is named "bird". When asked to search for
another operator application Duce comes back with the
message, "No applicable transformation", meaning that
none of the operators reduce the rule base. The time
between each prompt in this example is in the order of
one second.

Figure 3 shows the result of the transformations. Not
only is the rule base more compact but also the new
concepts have made the rules more conceptually trans-
parent. For example, a blackbird is simply defined as a
bird which is black. Note that the illustrative examples
are propagated to all new rules.

6 Even-parity

According to Minsky and Papert [10] the "parity" func-
tion is unlearnable by single-layer perceptrons. Recent
techniques using multi-layered perceptron networks [18]
have been shown to be capable of learning parity effec-
tively. However, in the paradigm of explicit rule for-
mation, algorithms such as ID3 [14] and AQ11 [8] turn
out to be rather inadequate when used to learn such
functions. It has been shown [11] that whereas single-
level concept representations of parity have a description
complexity which is necessarily non-polynomially depen-
dent on the number of attributes, multi-level descrip-
tions can be built whose size is only linearly dependent
on the number of primitive attributes. Efficient multi-
concept solutions inevitably rely on a divide-and-conquer
approach. Thus the decision of the top-level concept is
based on the combination of values of lower-level pred-
icates. Each lower-level predicate has a domain which
depends on a restricted subset of the total set of prob-
lem attributes.

Figure 4 depicts examples of 8-variable even-parity.
The variables (or primitive attributes) are numbered v1

!- induce
   DICHOTOMISATION – (-6)
   (even   [v3 t] ∧ [v4 t] ∧ ?)
   (¬even   [v3 t] ∧ [v4 t] ∧ ?)
     What shall I call ⟨?⟩? (name/n/i) n
   DICHOTOMISATION – (-5)
   (even   [v1 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
   (¬even   [v1 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
     What shall I call ⟨?⟩? (name/n/i) n
   DICHOTOMISATION – (-5)
   (even   [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
   (¬even   [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
     What shall I call ⟨?⟩? (name/n/i) n
   DICHOTOMISATION – (-4)
   (even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
   (¬even   [v1 t] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t] ∧ ?)
     What shall I call ⟨?⟩? (name/n/i) ¬sev
!- induce
   ABSORPTION – (-3)
   Is (¬even ¬sev ∧ [v1 f] ∧ [v2 t] ∧ [v3 t] ∧ [v4 t])
               a valid rule? (y/n/i) y
!- induce
   ABSORPTION – (-9)
   Is (even ¬sev ∧ [v1 f] ∧ [v2 f] ∧ [v3 t] ∧ [v4 t])
               a valid rule? (y/n/i) y
   Is (even ¬sev ∧ [v1 f] ∧ [v2 f] ∧ [v3 f] ∧ [v4 f])
               a valid rule? (y/n/i) y
   Is (¬even ¬sev ∧ [v1 t] ∧ [v2 f] ∧ [v3 t] ∧ [v4 t])
               a valid rule? (y/n/i) y
!- induce
   DICHOTOMISATION – (-2)
   (even ¬sev ∧ [v3 t] ∧ [v4 t] ∧ ?)
   (¬even ¬sev ∧ [v3 t] ∧ [v4 t] ∧ ?)
     What shall I call ⟨?⟩? (name/n/i) ¬ffev
!- induce
   ABSORPTION – (-1)
   Is (¬even ¬ffev ∧ [v3 t] ∧ [v4 t] ∧ v) a valid rule? (y/n/i) y
!- induce
   ABSORPTION – (-1)
   Is (even ¬ffev ∧¬sev ∧ [v3 f] ∧ [v4 f]) a valid rule? (y/n/i) y
!- induce
   TRUNCATION – (-7)
   Is (even ¬ffev ∧¬sev a valid rule? (y/n/i) n
No applicable transformation

Figure 5: Parity session

(even ¬ffev ∧¬sev ∧ [v3 t] ∧ [v4 t]) eg (ffttffff ttttffff tttttttt)
(even ¬ffev ∧¬sev ∧ [v3 f] ∧ [v4 f]) eg (ffffffff)
(ffev   [v1 t] ∧ [v2 t]) eg (ttttffff tttttttt)
(ffev   [v1 f] ∧ [v2 f]) eg (ffttffff)
(sev   [v5 t] ∧ [v6 t] ∧ [v7 t] ∧ [v8 t]) eg (tttttttt)
(sev   [v5 f] ∧ [v6 f] ∧ [v7 f] ∧ [v8 f]) eg (ttttffff)
(¬even sev ∧ [v3 t] ∧ [v4 t] ∧ ¬ffev) eg (fttttttt tttffff)
(¬even ffev ∧ [v3 t] ∧ [v4 t] ∧ ¬sev) eg (ttttffft tttttttf)
(¬ffev   [v1 t] ∧ [v2 f]) eg (tttffff)
(¬ffev   [v1 f] ∧ [v2 t]) eg (ftttttt)
(¬sev   [v5 t] ∧ [v6 t] ∧ [v7 t] ∧ [v8 f]) eg (tttttttf)
(¬sev   [v5 f] ∧ [v6 f] ∧ [v7 f] ∧ [v8 t]) eg (ttttffft)

Figure 6: Resultant parity rule base

to v8, and each is bound to a value from the set f t (rather than 0 1). In the first example, the variables have even-parity, since all eight have the value t, i.e. an even number of variables are bound to t. The "eg" part of the example shows a string of this form. Figure 5 shows the session in which Duce transforms the training set of Figure 4 into the partial, hierarchical solution of Figure 6. The responses are based on a standard solution in which the variables are recursively broken into two equal sized sets at each level. The total set of variables have even-parity if and only if both subsets have even-parity, or both have odd-parity. The first three concept suggestions do not follow this scheme, and are rejected. The fourth is recognised as "the second-half of the variables have even-parity" (sev). The user then affirmatively answers questions concerning the application of the absorption operator. The next suggested concept is named ffev or "first-half of the first-half of the variables are even". Given the original eight examples, Duce's solution is generalised to cover 16 of the 256 possible instances. If presented initially with the complete instance set, Duce tends towards a solution consisting of an 8-level deep hierarchy in which levels are used to count the number of variables set to t.

7 Recreation of the KPa7KR structure

Both the animal taxonomy and parity problem have highly restricted domains. The real test of Duce's capabilities has been the attempt to restructure Shapiro and Kopec's expert system [20] for deciding whether positions within the chess endgame of King-and-Pawn-on-a7 v. King-and Rook (KPa7KR) are won-for-white or not. The domain contains around 200,000 positions. Shapiro generated a database of all positions, labelling each with its minimax backup value of forced win-for-white or not. A set of 36 primitive board features were calculated for each position. Since many positions had the same feature vector and won-for-white value, the number of distinct examples was reduced to 3196. With this number of examples Duce's search-space for applying the first operation is 2^3196 (see Section 4), or approximately 10^1000. Nilsson [12] states that the complete game tree for chess has approximately 10^120 nodes; even that well-known hard problem has a considerably smaller search space than that attempted here.

For the purposes of the experiment, Shapiro provided a randomly chosen board position for each example. Thus the initial rule base given to Duce consisted of examples of the form

(won-for-white   feature1 $\wedge$ feature2 $\wedge$ .. feature36) eg (position)

Two chess experts, Ivan Bratko and Tim Niblett, helped in giving oracle answers to questions asked by Duce. The rule base started with 118,252 symbols. The first sugges-

```
Pa7
    bxqsq
    rimmx
    stlmt
    Delayed-queening
        hdchk
        Mate-threat
            bkxbq
            bkxwp
            qxmsq
            rxmsq
            r2ar8
        WK-on-a8
            blxwp
            r2ar8
            simpl
            wkna8
        WK-in-check
            bkxcr
            mulch
            rimmx
            rkxwp
            thrsk
            wknck
        Black-attacks-queening-square-soon
            bknwy
            bkona
            bkon8
            skrxp
            wkovl
    Double-attack-threat
        bkblk
        bkxbq
        cntxt
        katri
        wkpos
        Black-advantage-from-potential-skewer
            bkspr
            reskd
            reskr
            r2ar8
            skach
            wkcti
    Delayed-skewer
        bkxbq
        dsopp
        dwipd
        skewr
        spcop
        wtoeg
```

Figure 7: Human expert's KPa7KR problem decomposition

```
Pa7
   bxqsq hdchk
   rimmx spcop
   stlmt
   Delayed-queening-1
      bknwy bkon8
      dsopp mulch
      rxmsq skach
      skrxp wkna8
      Delayed-queening-2
         bkblk
         White-king-in-check-delay
            bkxcr bkxwp
            cntxt simpl
            skewr wknck
            wtoeg
         Skewer-threat
            bkspr rkxwp
            r2ar8 rkxwp
            wkcti wkovl
            wkpos
         Double-attack-threat-2
            bkona bkxwp
            bkxbq blxwp
            dwipd katri
            reskr
   Mate-threat-1
      cntxt wkna8
      Mate-and-double-attack
         bkblk bkxbq
         bkxcr katri
         thrsk
         Mate-and-double-attack-safe-from-promoted-queen
            dsopp qxmsq
            r2ar8 rkxwp
            wkcti wknck
      Mate-threat-2
         bkon8 bkspr
         bkxcr r2ar8
         skrxp thrsk
         wknck wkovl
         Mate-threat-safe-from-promoted-q
            bkxbq blxwp
            dsopp dwipd
            rkxwp rxmsq
            simpl skewr
            wtoeg
   Double-attack-1
      bkblk bkona
      bkspr bkxcr
      bkxwp blxwp
      dwipd skrxp
      Potential-double-attack-useful-to-black
         bknwy bkxbq
         cntxt katri
         r2ar8 wkcti
         wknck wkovl
         wkpos
```

Figure 8: KPa7KR knowledge structure generated by Duce

tion reduced 21,606 of these, a reduction of around 20%. After three questions, around 60% of the rule base had been reduced. After 41 transformations, the rule base had been reduced to 553 rules, and contained a total of 9050 symbols. At this point there were still applicable operations, but symbol reductions had been reduced to the low hundreds.

In questions 3 and 5, in which new concepts were introduced, the size of the common set of symbols, Int was too large for a comprehensible rule description. It is here that the illustrative board positions were indispensible. For this experiment, a domain-dependent graphics front-end was built into Duce, which gave the user the ability to peruse a large number of board positions representing the concept and counter-concept. Without this graphical device, new concepts could not have been recognised and named. As it was, concepts were named with confidence within the presentation of 20 to 40 board positions. It was rarely necessary to reject new concepts and generalisations suggested by Duce in the KPa7KR experiment.

Figure 7 shows the structure created manually by Shapiro and Kopec, which took an estimated 6 man months of effort. Figure 8 shows Duce's solution. Duce carried out the 41 oracle agreed transformations during a single working day. The computation time between each question was in the order of a minute. It should be noted that where Shapiro and Kopec have used nine hierarchically arranged concepts, Duce has used thirteen. Duce's solution also contains 553 rules and 9050 symbols where Shapiro and Kopec's manually created solution contains the equivalent of around 225 productions and around 1000 symbols. Although Duce's solution could have been made more compact by applying more transformations or by generating decision trees for each concept using ID3, it seems unlikely to the author that this would have resulted in a solution which was as compact as that of Shapiro and Kopec.

By virtue of the operators used by Duce, the KPa7KR solution is guaranteed correct by construction.

8 Neuropsychology application.

A second structuring experiment was carried out by the author using Duce at Interact Corporation, Canada. In this Duce was used to construct a problem decomposition for deciding on dysfunction of the left parietal brain area of children with learning disabilities. The input to the algorithm consisted of 227 diagnosed cases. Each case contained the results of a battery of approximately 100 binary-valued clinical tests. Each case was marked with a diagnosis of normal/abnormal left parietal lobe by the resident clinical neuro-psychologist, Dr. Russell. Using these cases Duce carried out an interactive session in which Russell was asked to answer a total of 53 questions. During and subsequent to the construction of

the rulebase, a set of 48 independent cases were used to test the performance of the new rule-set. Since the cases and generated rules were inherently noisy, a majority-vote mechanism was used for rule evaluation. After all 53 questions had been answered, 43 of the 48 test cases agreed with Russell's diagnosis, i.e. 90% agreement. In contrast, an existing expert system developed by Russell had only a 63% agreement rate with Russell's diagnoses over the same test data. While Duce's structured rulebase took 2-3 person-days to build and verify, the equivalent part of the hand-built expert system is conservatively estimated to have taken 2-3 person-months to generate, improve and verify.

In parallel with the supervised construction of the Duce rulebase, Duce was run on the same cases in unsupervised mode. In this mode, all generalisation questions were answered affirmatively and all new concepts were arbitrarily named. Performance with unsupervised learning stabilised after 27 questions to a level of 25% agreement with Russell's diagnoses of the same test cases.

Unlike the endgame experiment in which an exhaustive example set was used, the neuro-psychological example set was relatively sparse. As a consequence, whereas no rejections were necessary in the case of the chess experiment, an average of 10 rejections were required per acceptance with the neuropsychological data. This seems to indicate the need for expert supervision of Duce where sparse data is involved, and explains the dramatic difference in verification results between the supervised and unsupervised data.

The structure of the rulebase created by Russell working with Duce is shown in Figure 9. This hierarchical structure contains groups of rules associated with each node of the network. The sub-types implied by this hierarchy were, according to Russell, "clinically significant", and relate directly to neuropsychological sub-types based on Wide-range-achievement-test (WRAT) results in arithmetic, reading and spelling.

9 Conclusion

Duce is a program which, with the aid of a human oracle, discovers useful new concepts. AM [6], an early concept discovery program, was criticised [17] for the obscurity of the techniques involved. Unlike AM, Duce uses a simple and explicit set of six operators to create and refine concepts. In addition the meaning-giving agent, implicitly present within any Machine Learning system, is explicitly represented as the oracle within Duce.

Extensive search is used to decrease the number of questions asked by Duce of the oracle. However, in what circumstances is the use of an oracle either justified or feasible? In this respect it is worth noting that on the basis of a meagre number of empirical studies the ratio

```
LPAb     LPAb_GoodV
         LPAb_PoorA1
             LPAb_PoorA
         LPAb_BadA1
             LPAb_BadA1_RS_AST
             LPAb_BadA
                 LPAb_BadA_PoorS
                     LPAb_BdA_PrS_BadAST_GdSSPT
                 LPAb_BadA_BadAST
                     LPAb_BadA_AST_S1
                         LPAb_BadA_AST_S
                         LPAb_BadA_AST_S1_V
```

Figure 9: Structure of neuropsychological expert system constrcuted by Duce

of oracle rejections to acceptances seems to be inversely related to the percentage of examples provided from the domain. In the parity problem, where Duce was supplied with a sparse set of examples, a large number of rejections were necessary (Figure 5). In the more complex KPa7KR chess domain, Duce was given an exhaustive set of examples, and required almost no rejections from the oracle. Thus it might be expected that in domains in which a moderate amount of example material is available the oracle would need to reject a moderate number of proposals. Further research is necessary to show the truth of this hypothesis.

Duce works with statements in propositional logic. One way of extending the present work would be to attempt using similar techniques within other representations. Banerji [1] is presently looking at the problem of constructive induction within first-order calculus. The author believes that techniques akin to those used in Duce could be profitably employed in learning hierarchically definable context free grammars.

Acknowledgements

References

[1] R.B. Banerji. Learning in the limit in a growing language. In *IJCAI-87*, pages 280–282, Los Angeles, CA, 1987. Kaufmann.

[2] R.M. Burstall and J. Darlington. A transformation system for developing recursive programs. *Journal*

of the Association for Computing Machinery, 24:44–67, 1977.

[3] N. Dershowitz. Synthesis by completion. In *IJCAI-85*, pages 208–214, Los Altos, CA, 1985. Kaufmann.

[4] E.A. Feigenbaum. Themes and case studies of knowledge engineering. In D. Michie, editor, *Expert Systems in the Micro-electronic Age*, pages 3–25. Edinburgh University Press, Edinburgh, 1979.

[5] L.M. Fu and B.G. Buchanan. Learning intermediate concepts in constructing a hierarchical knowledge base. In *IJCAI-85*, pages 659–666, Los Altos, CA, 1985. Kaufmann.

[6] D.B. Lenat. On automated scientific theory formation: a case study using the AM program. In J.E. Hayes and D. Michie, editors, *Machine Intelligence 9*. Horwood, New York, 1981.

[7] R.S. Michalski. Understanding the nature of learning: issues and research directions. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, pages 3–25. Kaufmann, Los Altos, CA, 1986.

[8] R.S Michalski and R.L. Chilausky. Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161, 1980.

[9] R.S. Michalski and R. Stepp. Learning from observation: conceptual clustering. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–364. Tioga, Palo Alto, CA, 1983.

[10] M. Minsky and S. Papert. *Perceptrons*. MIT Press, MA, 1969.

[11] S.H. Muggleton. *Inductive Acquisition of Expert Knowledge*. PhD thesis, Edinburgh University, 1987.

[12] N.J. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

[13] A. Paterson. An attempt to use cluster to synthesise humanly intelligible subproblems for the kpk chess endgame. Technical Report UIUCDCS-R-83-1156, Univ. Illinois, Urbana, IL, 1983.

[14] J.R. Quinlan. Discovering rules from large collections of examples: a case study. In D. Michie, editor, *Expert Systems in the Micro-electronic Age*, pages 168–201. Edinburgh University Press, Edinburgh, 1979.

[15] J.R. Quinlan. Semi-autonomous acquisition of pattern-based knowledge. In D. Michie, editor, *Introductory readings in expert systems*, pages 192–207. Gordon and Breach, New York, 1982.

[16] L. Rendell. Substantial constructive induction using layered information compression: tractable feature formation in search. In *IJCAI-85*, pages 650–658. Kaufmann, 1985.

[17] G.D. Ritchie and F.K. Hanna. AM: a case study in Ai methodology. *Artificial Intelligence*, 23(3):249–268, 1984.

[18] D.E. Rumelhart and J.L. McClelland. Learning internal representations by error propogation. In *Explorations in the Micro-Structure of Cognition Vol. 1 : Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[19] C. Sammut and R.B Banerji. Learning concepts by asking questions. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach. Vol. 2*, pages 167–192. Kaufmann, Los Altos, CA, 1986.

[20] A.D. Shapiro. *Structured Induction in Expert Systems*. Adison-Wesley, Wokingham, 1987.