

Toward a Unified Approach for Conceptual Knowledge Acquisition

Larry A. Rendell

*Department of Computing and Information Science
University of Guelph
Guelph, Ontario, N1G 2W1, CANADA*

Abstract

In keeping with a desire to abstract general principles in AI, this article begins to examine some relationships among heuristic learning in search, classification of utility, properties of certain structures, measurement of acquired knowledge, and efficiency of associated learning. In the process, a simple definition is given for *conceptual knowledge*, considered as information compression. The discussion concludes that domain-specific conceptual knowledge can be acquired. Among other implications of the analysis is that statistical observation of probabilities can result in the equivalent of planning, in low susceptibility to error, and in efficient learning.

SEVERAL RESEARCHERS HAVE INDICATED that more integration and synthesis may be imminent in AI. In a panel discussion on "Challenges of the Eighties" at a workshop on Machine Learning recently held (Proc. IMLW, 1983), several such opinions were stated. Among other issues, Michalski stressed the importance of unification of terminology and extraction of general principles. Amarel suggested we need both theory and application, even within a single project (one supports the other). Another indicator: Chapter XIV *Handbook of Artificial Intelligence* (Dietterich, London, Clarkson & Dromey, 1982) compares and contrasts generalization methods; Michalski (1983) develops a unified theory for induction, and characterizes its types. (In fact synthesis is

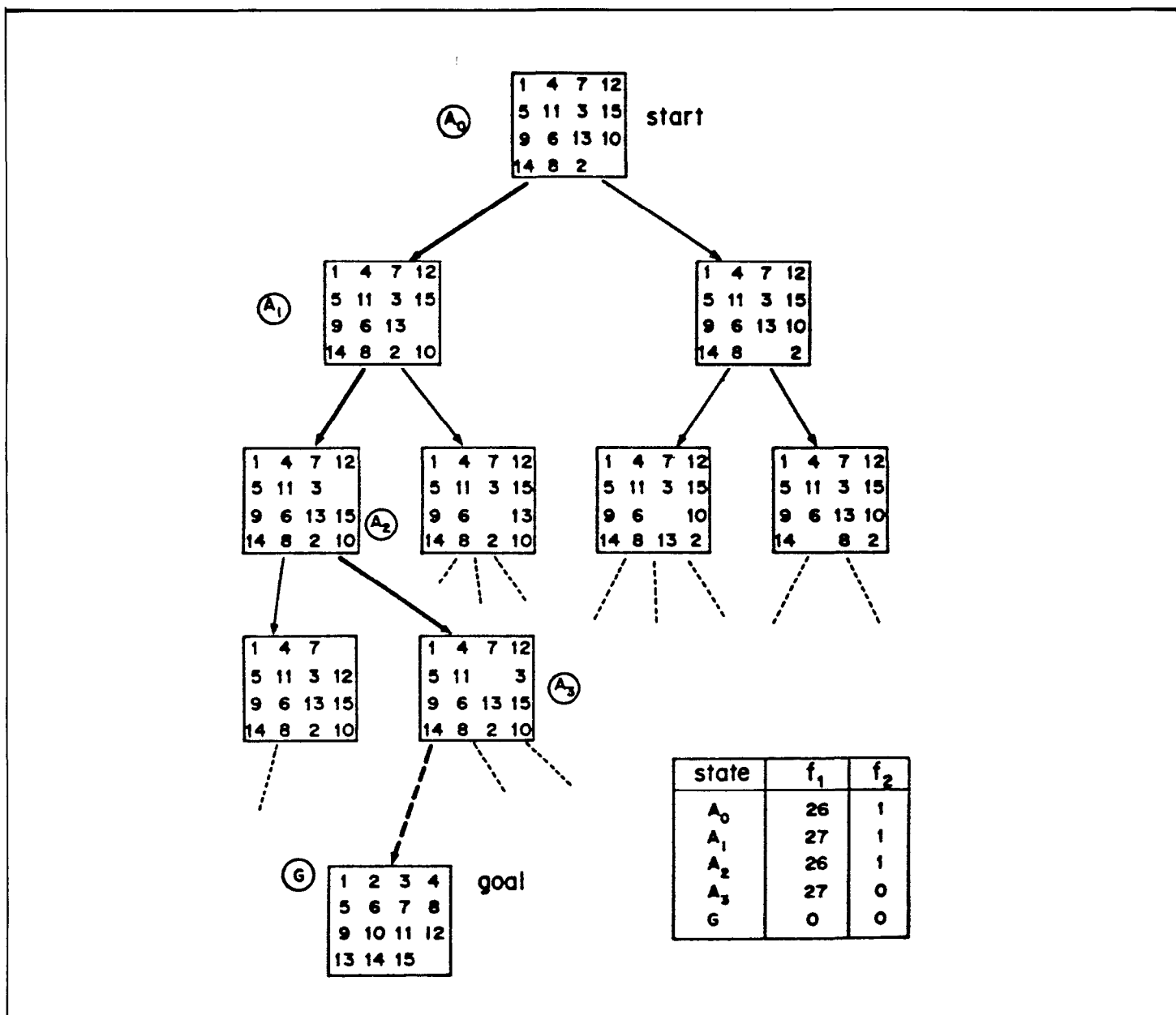
wider than this: e.g. AI is being related to cognitive science (Pylyshyn, 1982) and with control theory and pattern recognition (Buchanan, Mitchell & Smith, 1978).)

This article suggests new ideas and attempts to relate them to some existing ones in AI. While the focus is heuristic learning in search, it is examined with broader intent. Among the issues discussed are the power of selected structures, the measurement of acquired knowledge, and the efficiency of associated learning. Viewpoints include evaluation as utility classification, and knowledge as information compression. Consider for example the current awareness that domain-specific knowledge is necessary for real results with interesting problems. This is generally high level or abstract knowledge. But precisely what do these terms mean? Does domain-specificity preclude autonomous knowledge acquisition from low level data? To provide a context for analysis, let us reconsider a well established topic in a different light.

Evaluation Functions as Plans

During search, Samuel's (1963, 1967) checker player assessed board positions (nodes) as candidates for expansion using an *evaluation function* appraising the overall *utility* of each position. Such a function is normally a composition of *features*, which are themselves elementary functions measuring various aspects of nodes. Typical features for checkers

The author would like to thank Dave Coles for his comments on a draft of this article.



Fifteen puzzle illustration. In order to develop a concise and useful evaluation function, features can be defined, such as f_1 (sum of distances of tiles from home) and f_2 (number of pair reversals). Notice the '7' and '3' are reversed in A_0 through A_2 ; when such anomalies are allowed to form or remain, the puzzle may erroneously appear close to the goal if f_1 is used alone. In the f_1, f_2 feature space, vectors near the origin are generally better (see values in the inset table).

Figure 1.

include piece advantage, control of center, mobility, etc. Another example is shown in Fig. 1. Here just two features are used in a heuristic function for the fifteen puzzle: the obviously effective f_1 = the sum of 'city block' distances of the tiles to their home positions, and another penalty f_2 = the number of 'reversals' (adjacent tiles being in their correct positions except switched) (Doran & Michie, 1966). Features such as these are often combined linearly: if f is a given vector of features, the evaluation function is $H = b.f$, where b is the weight vector (Rendell, 1981, 1983a; Samuel,

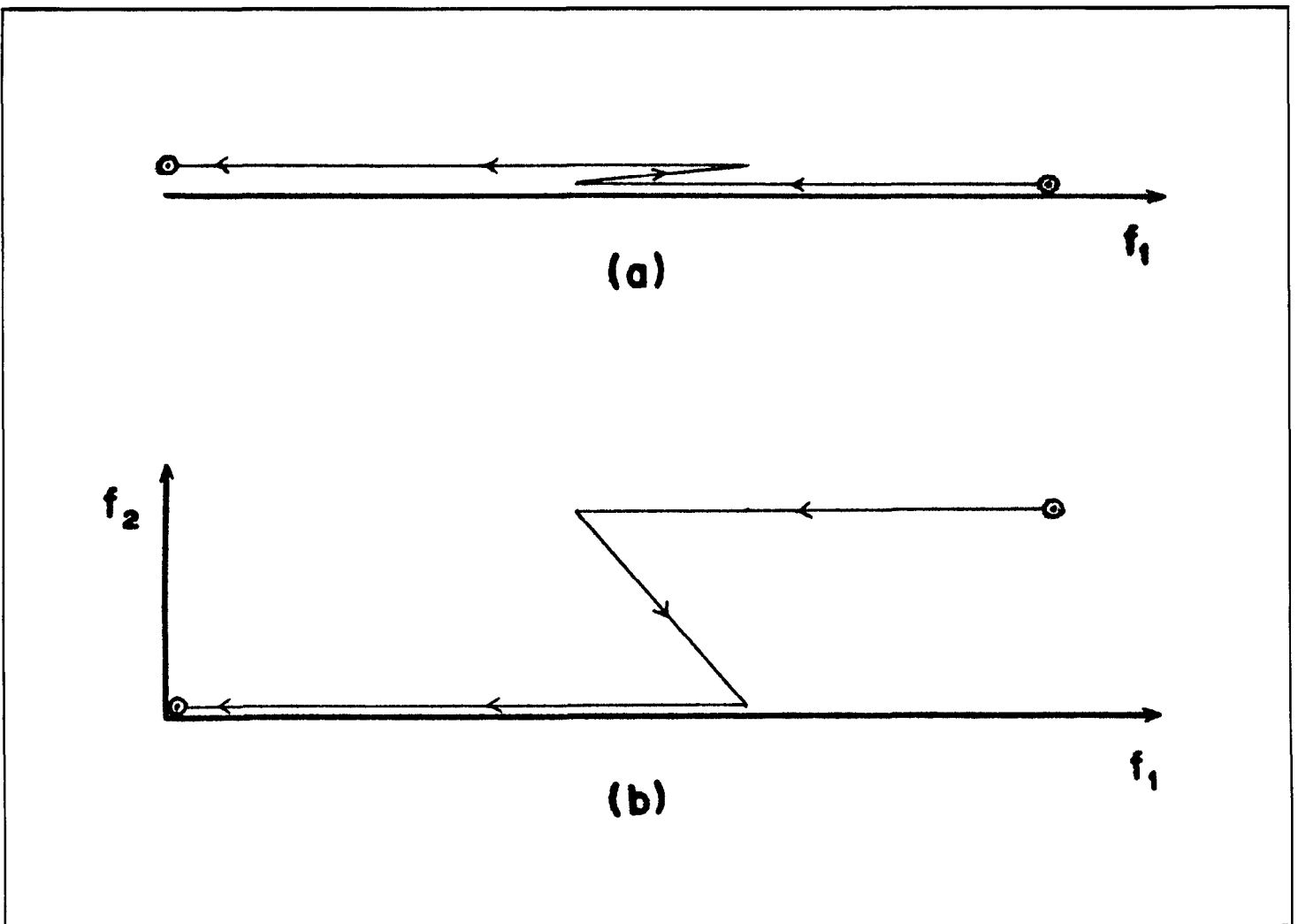
1963). A linear model imposes restrictions, although these can sometimes be avoided (Rendell, 1983b; Samuel, 1967). However a more serious shortcoming remains: features are actually high level constructs, typically containing about 80% of the conceptual knowledge (as we shall see later). The unanswered question of how to mechanize formation of new terms (features) was probably the main factor in the decline of this sort of learning approach. It is a hard problem.

Despite this, there is no inherent theoretical reason for abandoning evaluation functions as part of a framework for

learning heuristics (for knowledge acquisition). As Feldman and Yakimovsky (1974) point out, the evaluation function is fully powerful. As such, it is also analogous to other approaches. Consider Fig. 2(a) where a solution path is transformed into a path in feature space, which could be called the *feature trace* (of a solution). What can be thought of as a complete strategy or plan is reflected in some collection of feature traces. For example, the plan "Make a move so as to bring some tile closer to home" corresponds to a (large) set of traces like the one in Fig. 2(a). As long as enough discriminating features are incorporated, any strategy can be represented by an appropriate evaluation function, which

will in turn determine the feature traces. Sophisticating a plan may require additional features in our parallel view. For example, "If there are any reversals, get rid of them" can be added to the naive original advice by using f_2 of Fig. 2(b). Encompassing a grand strategy, many plans can coexist in feature space; each corresponds to a distinct *category* of feature traces.

Whatever traces are followed in a solution, the ideal pattern is high node *utility* on the solution path and low utility elsewhere. In search two distinct but related measures of utility have been used: the path distance remaining to the goal (Hart, Nilsson & Raphael, 1968; Nilsson, 1971)

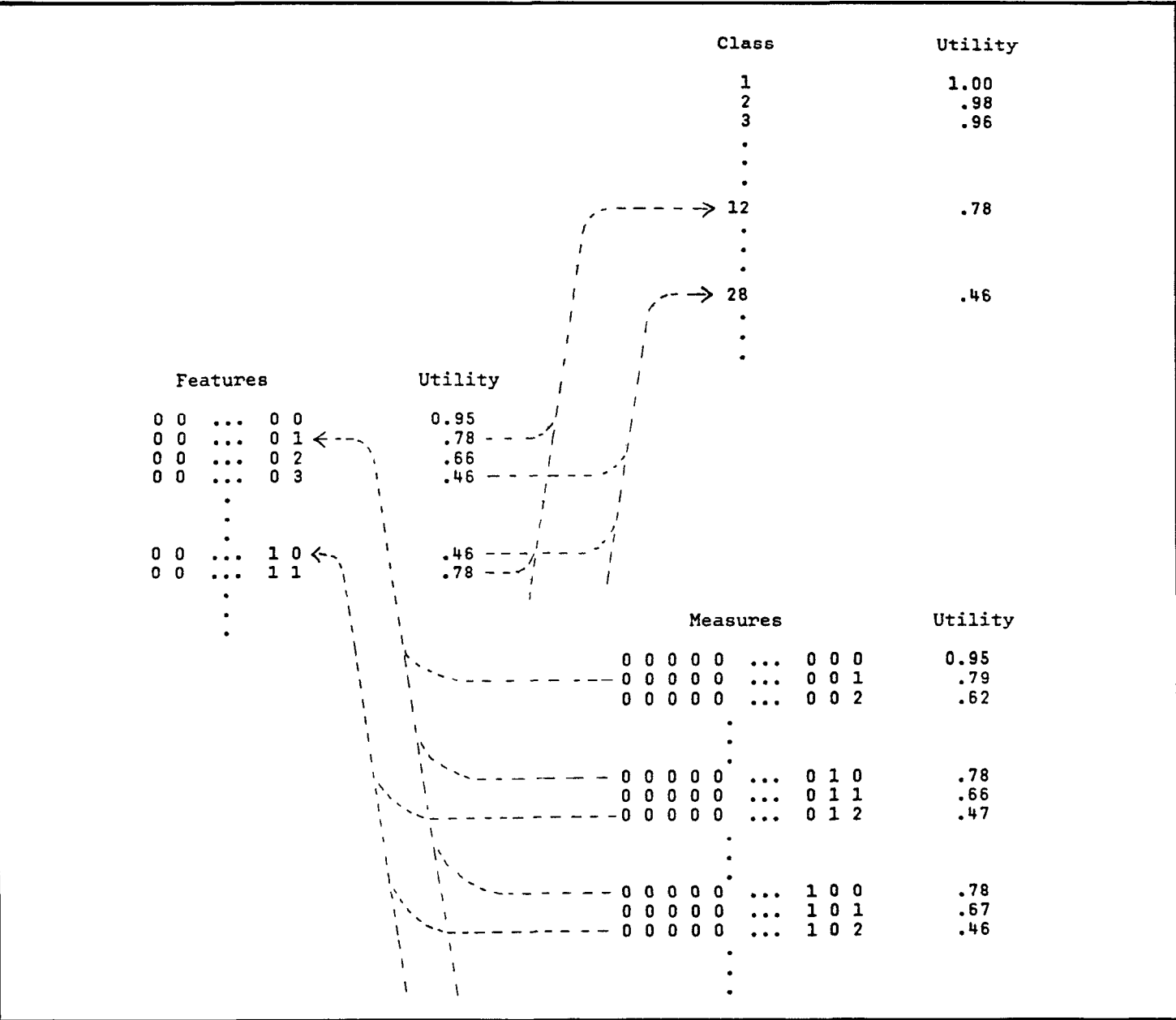


Feature traces of solutions. Irregular solution paths followed in feature space become unsurprising when enough appropriate features are present. Here f_1 and f_2 are the distance and reversal scores for the fifteen puzzle. In the one-dimensional f_1 space (a) solutions sometimes trace strange paths. However they appear normal in the f_1, f_2 space (b), where the distance score is temporarily worsened to unclog a reversal. Whole bundles of feature traces reflect a strategy or plan, such as 'Temporarily worsen the distance score if a reversal can be removed.'

Figure 2.

and some sort of likelihood of usefulness (Nau, 1983; Pearl 1982; Rendell, 1983a, 1983b; Slagle & Bursky, 1968; Slagle & Farrel, 1971). (Recently the latter has received more attention). In practice either of these can only be estimated; moreover features are generally coarse measures. Hence,

actual feature traces track real utility only roughly. Most features are designed individually to bear a nearly monotonic relationship to utility (consider the ones mentioned above). Although most learning involving evaluation has started with features already defined by the programmer (Quinlan,



Levels of conceptual knowledge in search. Elementary data represent fully detailed knowledge but no abstraction, and so are massive and infeasible to gather. At the other extreme, maximal information compression avoids loss of expressive power while being very concise. Intermediate representations are designed to facilitate operation; for example features should discriminate utility well and bear a nearly monotonic relationship with it but relatively mild interaction amongst each other. In any case, some rules can always be given for the classification of individual table entries at the data level; those rules are concepts, and features are one tool for concise and orderly description, i.e., for concept attainment.

Figure 3.

1983; and Rendell, in preparation; are exceptions), the ultimate goal is to create a system which can learn everything by abstracting from raw data. Chapter XIV of the *The Handbook of Artificial Intelligence* (Dietterich, 1982) characterizes three kinds of generalization learning. The simplest form is single concept, the second is multiple concept, and the most difficult is performance of multiple step tasks, requiring a sequence of operators. From one point of view the automatic construction of an evaluation function can be seen as a problem of learning multiple concepts: each utility value can be considered abstractly as having an antecedent, a rule in the form of statements about features (see Fig. 3). There are many utility categories; often an infinite number, although finiteness can always be imposed without loss of power as long there is an upper bound on solution length (see the next section). From another viewpoint, however, the use of an evaluation function intrinsically guides operator application (consider Fig. 1). Explicit concept learning entails implicit operator sequencing. Moreover, since heuristics govern feature traces (in fact, diverse kinds of traces), learning utility concepts eventually results in disguised planning or strategy formation. This is reminiscent of animal perception or pattern recognition effecting some sort of gestalt.

Let us examine more closely the idea of categorizing nodes according to their utility.

Classification, Compression, and Conceptual Knowledge

Theoretically, an evaluation function can be expressed in very primitive form. The set of situations at or close to the level of the data could be represented as a table listing the node description or measurement vector along with its corresponding utility value (Fig. 3). The measurements might be quite basic, for example the individual city block distances, one for *each individual* tile in the fifteen puzzle, or fully elementary, for instance the contents of *each square* of the board in checkers. A complete (and accurate) table based on such fundamental measurements would enable perfect choice of a node to expand but the quantities are immense. For these two examples roughly $6^{15} \approx 2^{39}$ and about $5^{32}/4 \approx 2^{72}$ table entries (respectively) would be required.

Next consider a fully abstract heuristic, the topmost level of Fig. 3. How many utility classes are required for perfect state evaluation, i.e. how many categories are needed for sufficient resolution of the worth of a state? Since optimal solution (shortest path length and fewest nodes developed) for a state space problem requires only knowledge of path length remaining to the goal (Hart, Nilsson & Raphael, 1968; Nilsson, 1971), the number of classes need be just the maximum of the shortest solution for any problem instance. For a game, half the number of moves in a long contest might be sufficient. (These numbers are roughly 75 for the fifteen puzzle and somewhat less for checkers). Although perfect prior knowledge of solution length is not feasible in interesting cases, actual evaluation functions might be expected to

follow a similar pattern. To test this assertion, an experiment with the best heuristic known for the fifteen puzzle (Rendell, 1981, 1983a) was conducted. When the evaluation function was discretized, the relationship of performance to number of classes was as shown in Fig. 4.

Consequently, in the construction of a heuristic, the number of meaningful utility classes is reduced from about 2^{39} to 2^6 for the fifteen puzzle, and from roughly 2^{72} to 2^6 for checkers. Another expression of these facts is that the *information* of a level I is $\log_2 N_I$ where N_I is the number of classes at level I, and that the information compression between level I and level I+1 is $\log N_I/N_{I+1}$ (Watanabe, 1972). Hence the compressions in these two examples are the huge amounts of about 33 bits and 66 bits respectively.

Define the *conceptual knowledge* present at a level to be exactly the information compression required to form the classes at that level, from the most elementary classes (the raw data). An evaluation function which compresses information just to the point at which performance begins to suffer could be said to capture maximum conceptual knowledge.

Now consider the intermediate, feature level (Fig. 3) from the viewpoint of conceptual knowledge. The feature space volume is about 2^{12} for the above case with the fifteen puzzle (Rendell, 1981) and approximately 2^{21} with checkers (Samuel, 1963). This volume is the number of classes at the feature level. Computing the information compression at all three levels, we find that for each of these two examples, about 80% of the maximum conceptual knowledge is captured in the middle (feature) level, while the remaining 20% is expressed in the evaluation function as a particular combination of features. Most of the knowledge is contained in the features. This explains why composition of the heuristic from features (even allowing feature interaction) is not very impressive from the point of view of full inductive capability (since general learning should *create* the features themselves). It also suggests the intervention of more levels between raw data and typical high level features (Rendell, in preparation; Watanabe, 1972) has found that uniform reduction is most effective.) Conceptual knowledge implies not only information compression, but also rational ordering. As indicated in Fig. 3, an exhaustive listing of measurement vectors has completely haphazard utility values (i.e. the utility surface in measurement space is extremely irregular). Features begin to impose regularity; an ordering based on feature values groups elements from the low level table into locally uniform utility categories (i.e. features are surjections—their range is smaller than their domain, and the utility surface in feature space is fairly smooth; consider for example the two illustrative features for the fifteen puzzle). Finally, the ultimate evaluation function merges the features to construct a meaningful global arrangement of the smaller utility clusters.

Equivalently, conceptual knowledge implies conceptualization. Features represent rational *components* of concepts. The evaluation function codes complete concepts. Each concept has a worth. Features associate partially formed con-

cepts with (here numerical) values. The evaluation function essentially pairs each full concept or antecedent rule r , with its consequent utility u (Fig. 3). Meaningful classification, rational ordering, and concept attainment are interrelated.

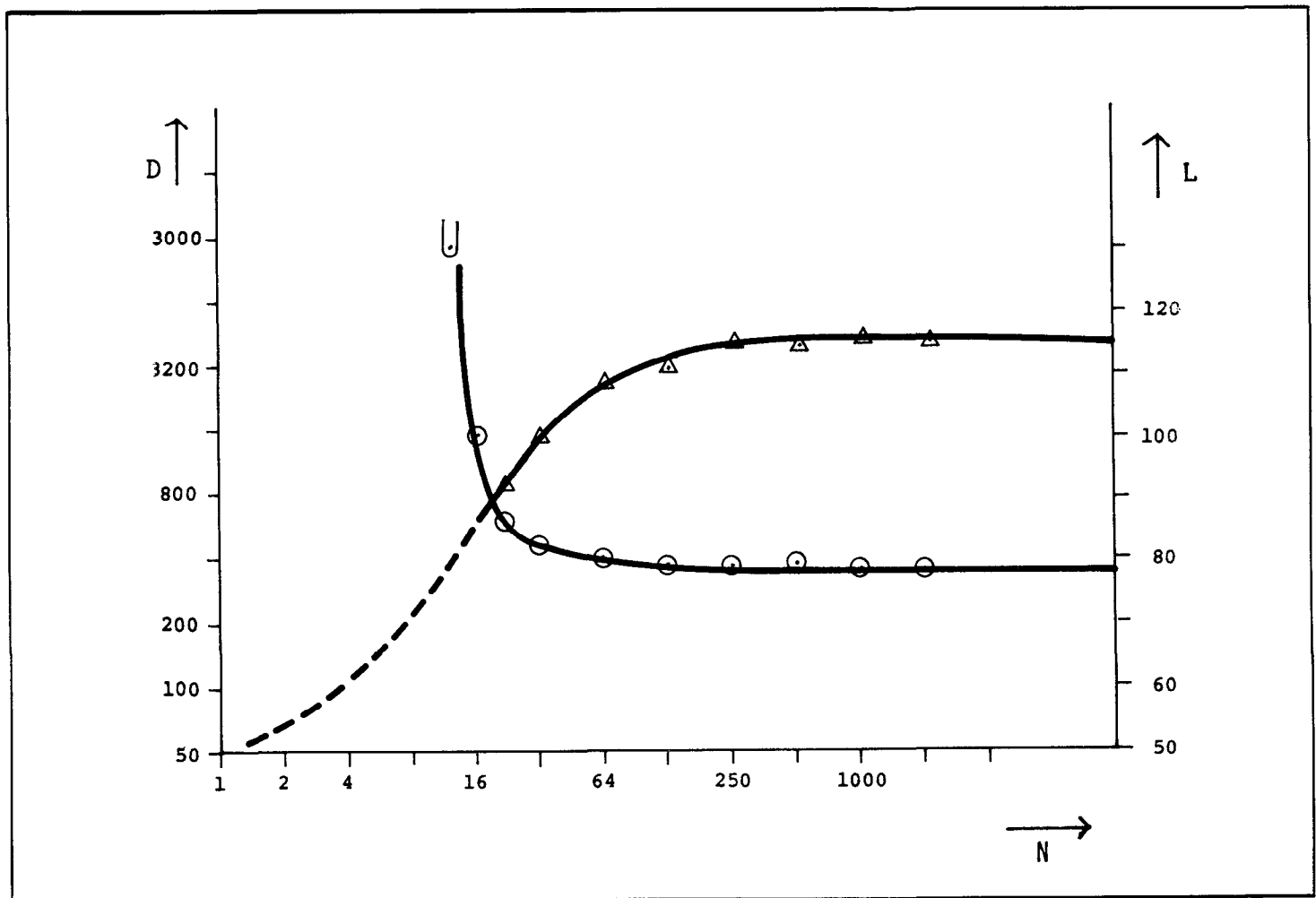
An Important Knowledge Structure

This suggests a knowledge structure (r, u) for pairing concepts (full or partial) with their utilities. This proposed structure, which is different from the data, the features, and the ultimate evaluation function, can be used for conceptual development. Fig. 5 shows a straightforward choice, a partition of the feature space in question, where each *region* is a (hyper)rectangle r aligned with the axes, along with its associated utility u . Such a rectangle is also a *concept* representing a conjunction of feature ranges.¹ (This is for ordered

feature values; a different treatment is required when feature values are unordered sets.)

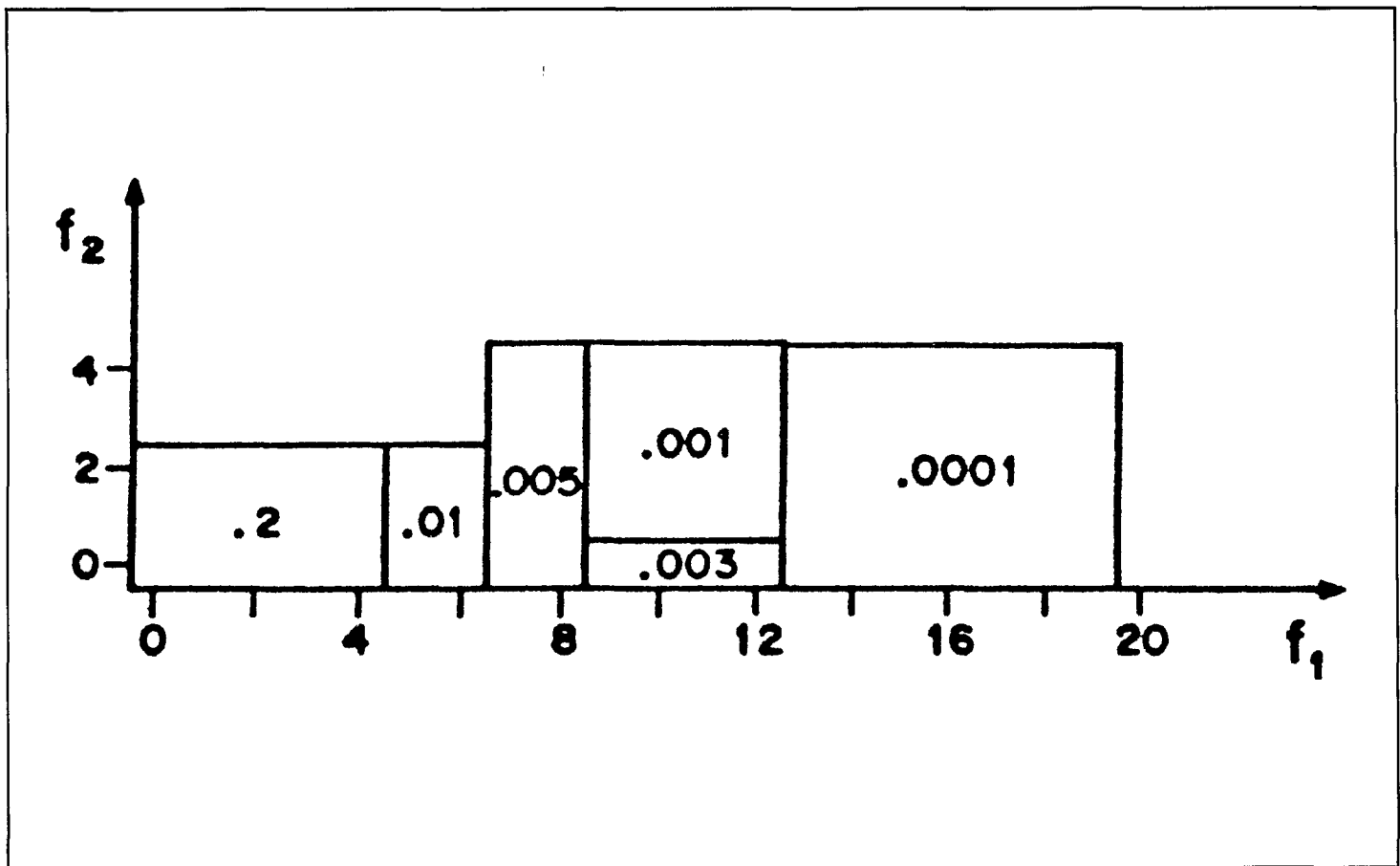
A region *set* (r, u) , has been used successfully in a probabilistic learning system (PLS – Rendell, 1981, 1983a, 1983b), where the utility u is a goal oriented quantity, the proportion of nodes eventually participating in a solution to a state space problem or win in a game. The utility thus predicts likelihood of success in problems yet to be encountered, as a function of feature values (the utility is thus a conditional probability (Rendell, 1983a; and c.f. Nau, 1983). With a discrete mode of evaluation, if (r, u) is a region of the current set and if the concept r is satisfied by a configuration A

¹Since general class membership (i.e. a utility category u) can be entailed by a set of conjunctions of feature ranges (by a set of rectangles r), this is equivalent to a disjunction of logical formulas in conjunctive normal form implying u (Nilsson, 1981); and thus fully general



Discretizing an evaluation function. When utility values are forced into discrete categories by arbitrary rounding, the function becomes a classifier. Depending on the number of categories demanded N , the performance may be unaffected or it may change. D is the average number of states expanded before solution (circles) and L is the average solution length (triangles), for a random sample of 32 fifteen puzzles. Unlike D , L improves with smaller N since the heuristic is imperfect and lower N means more nearly breadth-first search.

Figure 4.



A simple region set, a partition of feature space. Shown are rectangles and their associated utilities (inside). A region is a means of expressing a concept and its worth; it is equivalent to a set of statements in the first order predicate calculus with the concept as precondition and the utility as consequent.

Figure 5.

being assessed (i.e. if A maps into r), then A has the utility u . With smoothed evaluation, a curve fitting technique generalizes from the regions to generate a parameterized evaluation function $H = \exp b.f.$ (It is this heuristic that was made artificially discrete for Fig. 4.)

In a PLS, sets of these regions gradually accumulate knowledge, revising estimates u and refining feature space volumes (concepts) r into units just adequate to express known relationships. The result is an effective economy, a distillation of experience. In one sense, the region set is a refinement of Samuel's (1967) signature tables which did not alter data categories automatically. The novelty of a PLS is that it computes statistics measuring solution density in feature space, and uses these data to guide knowledge refinement. Although it is data driven, a PLS is insensitive to noise since it is stochastic (Samuel, 1963, 1967) which are different but also statistical). Most concept formation systems have difficulty with incorrect data. A PLS is quite natural; Bruner et. al. (Bruner, 1956) emphasized the human use of probabilities, features, and goal orientation.

Some problems appear simpler in a PLS (Rendell, in preparation). For example, when the method is extended to games, the horizon effect can be absorbed in the statistics, as environments change during incremental learning.²

The region set is potentially useful in many domains; whenever, in fact, the worth of something is observable and contingent on a concept. For example an expert system could use the same organization, perhaps with one region set for

²The horizon effect is the misleading information resulting when search is prematurely cut off in the middle of, say, a sacrifice (Berliner, 1973) In a unified method for state space problems and games using the region set statistical approach, there are two general ways of assigning credit to nodes: backed-up values can be either discrete (on a solution path or not in a problem; win/lose/draw in a game), or else continuous, the result of applying an existing heuristic to the tip nodes (Rendell, in preparation). This is often mentioned with minimax but here learning can occur using either of these methods exclusively. With the former, training instances are initially easy problems or end games; with the latter method, some initial evaluation function is given, then improved. In either case the ability of the solver or player is bootstrapped using statistical results of experience and natural goal orientation (Rendell, 1983; Rendell, in preparation)

each production rule (or for each structured arrangement of them). Then conditions for firing become probabilistic, and easy to learn and modify. (Compare Michalski's (1983) internal disjunction.) This knowledge structure, the region set, is of wider use. When the features are replaced by elementary measurements, it becomes a good starting point for inductive inference of features (Rendell, in preparation).

One observation about typical region sets relates to conceptual knowledge: they often have fewer members than the effective number of concepts at the heuristic level; when the evaluation function is inferred information is created, according to the definition of Section 3. This is another statement of the fact that induction occurs (Watanabe, 1969).

To summarize: Regions express utility-concept relationships fully and provide a framework for condensation of experience by collecting probabilities stochastically. The statistical use of observations has several desirable consequences. Error is absorbed, experience can be updated and concepts can be precisely refined according to current data. Another advantage is efficiency, discussed below.

Efficiency in Knowledge Acquisition

As pointed out in a previous section, learning systems that allow the programmer to define the features are already given, a priori, about 80% of the total conceptual knowledge. Let us first consider this 20% problem, the formation of an evaluation function from its component features. Despite this 80% advantage, creation of a good evaluation function is still difficult to mechanize with reasonable efficiency. (although many approaches have succeeded, e.g. Rendell, 1983a; Samuel, 1963, 1967).

Instead of a learning system, some traditional optimization method can be used to find the best choice of weight vector b for feature vector f in the evaluation function $H = \exp b.f$. In state space problem solving, a likely choice would be to minimize the number of nodes developed D (this is an objective function). Since the variances in D are typically high (even when test problems are multiple), a statistical technique would be chosen, such as fitting second degree polynomials in the weight space (higher dimensional parabolas, with D as the dependent variable). This *response curve* fitting, though, is very expensive. The value of D for typical weight vectors is too high for computational feasibility. Even if the location of the optimal b is already known approximately, the process is inefficient. Experiments have shown the increased efficiency and reliability of a more informed scheme, the learning system PLS1 (Rendell 1981, 1983a). Intuitively it is clear why this is so: the optimization technique obtains just a single number (D) and disposes of everything else. In contrast, a useful learning system extracts much more information from search trees. In both Samuel's system (1963, 1967) and PLS1, every node developed contributes to the assessment of every feature. This is possible because information is available about the usefulness of a node. Nodes are mapped into feature space

where credit assignment to a node becomes credit assignment to each individual feature. The summation of all contributions provides an informed general picture of feature-utility relationships.

Similarly Quinlan (1983) shows the computational advantage of a feature classification and information theory approach to search, compared with minimax; his ID3 system makes use of individual nodes in the judgment of utility by features during evaluation. See also Pearl's suggestion of a probabilistic approach (Pearl, 1982).

Good use of information is the key to success in heuristic formation. Consider again the utility class discussion. At the level of the raw data, a table of utilities can theoretically be constructed for every individual state. But operating solely with this great detail, a rote learning system would never gather much knowledge in practice. Furthermore, patterns among table entries would never be discovered. An ability to assign many states to a single, meaningful class is of double importance. First, during evaluation, states which have never been encountered can be judged since they are 'similar to' other members of the class. Secondly, during learning, data can support each other and so a greater assurance is possible of the utility values ascribed to each concept.

This *mutual data support* is straightforward when features are already given. Relationships between dependent variable (utility) and independent variables (here the features) are smooth; permitting curve fitting which allows both interpolation during evaluation and mutual support during learning. Mutual data support is more difficult with elementary measurements, but perhaps this same principle can still be applied (Rendell, in preparation).

Conclusions

This article has attempted to continue synthesis of some principles in knowledge use, acquisition and learning. A definition of conceptual knowledge has been given in a limited domain. Relationships have been expressed between evaluation functions of heuristic search, on the one hand, and planning, classification, knowledge acquisition, and efficiency on the other. An evaluation function estimates the utility class u into which a node should be placed; the basis for the decision is the satisfaction of a concept r . The formation of such concepts is natural and powerful when they are represented as volumes in feature space and probabilistic measures of the utility are incorporated. First, the information gathering process creates implicit plans as a side effect. Secondly, observation of probabilities provides a straightforward method both for this data collection and for its use. The statistical approach lowers susceptibility to error, and allows appropriate revision of utility and refinement of concepts. Finally, the process is efficient because of good use of data encountered. This abstraction of knowledge structure and statistical method for generalization result in a capable system, and seemingly wide application.

Domain-specific conceptual knowledge can be acquired autonomously from lower level information, and perhaps ultimately from elementary data.

References

- Berliner, H. (1973) *Some necessary conditions for a master chess program* *IJCAI* 3, 77-85.
- Bruner, J. S., Goodnow, J. J., & Austin, G. A. (1956) *A Study of Thinking*. NY: John Wiley & Sons.
- Buchanan, B. G., Johnson, C. R., Mitchell, T. M., & Smith, R. G. (1978) Models of learning systems. in Belzer, J. (Ed.) *Encyclopedia of Computer Science and Technology* 11.
- Dietterich, T. G., London, B., Clarkson, K., & Dromey, G. (1982) Learning and inductive inference. STAN-CS-82-913, Stanford University, also in Cohen, P. R., & Feigenbaum, E. A. (Eds.) *The Handbook of Artificial Intelligence*, Chapter XIV, Los Altos: Wm. Kaufmann.
- Doran, J. & Michie, D. (1966) Experiments with the graph-traverser program, *Proceedings of the Royal Society, A*, 294 235-259.
- Feldman, J. A. & Yakimovsky, Y. (1974) Decision theory and artificial intelligence: I. A semantics-based region analyzer, *Artificial Intelligence* 5, 349-371.
- Hart, P., Nilsson, N. J., & Raphael, B. (1968) A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Sys. Sci. and Cybernetics* SSC-4, 100-107.
- Michalski, R. S. (1983) A theory and methodology of inductive learning, in Michalski, R. S. et al (Ed.), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: Tioga Publishing, 83-134.
- Nau, D. S. (1983) *Artificial Intelligence* 21, 221-244.
- Nilsson, N. J. (1971) *Problem Solving Methods in Artificial Intelligence*. New York: McGraw-Hill.
- Pearl, J. (1982) On the nature of pathology in game searching, UCLA-ENG-CSL-8217, School of Engineering and Applied Science, University of California.
- Proceedings of the International Machine Learning Workshop*. Allerton House, University of Illinois at Urbana-Champaign, June 22-24.
- Pylyshyn, Z. W. (1982) Literature from cognitive psychology, *Artificial Intelligence* 19, 251-255.
- Quinlan, J. R. (1983) Learning efficient classification procedures and their application to chess end games. in Michalski, R. S. et al (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: Tioga Publishing, 463-482.
- Rendell, L. A. (1981) An adaptive plan for state-space problems, Dept of Computer Science CS-81-13, (PhD thesis), University of Waterloo.
- Rendell, L. A. (1983a) A new basis for state-space learning systems and a successful implementation. *Artificial Intelligence* 21.
- Rendell, L. A. (1983b) *A learning system which accommodates feature interactions*, *IJCAI* 8, 469-472.
- Rendell, L. A. (in preparation). A uniform learning system for problems and games.
- Rendell, L. A. (in preparation). Progress in induction of features for problems and games.
- Samuel, A. L. (1963) Some studies in machine learning using the game of checkers. in Feigenbaum, E. A. & Feldman, J. (Eds.), *Computers and Thought*. New York: McGraw-Hill, 71-105.
- Samuel, A. L. (1967) Some studies in machine learning using the game of checkers II—recent progress, *IBM J. Res. and Develop.* 11 601-617.
- Slagle, J. R. & Bursky, P. (1968) *Experiments with a multipurpose, theorem-proving heuristic program*. *JACM* 15, 85-99.
- Slagle, J. R. & Farrel, C. (1971) Experiments in automatic learning for a multipurpose heuristic program, *C. ACM* 14, 91-99.
- Watanabe, S. (1969) *Knowing and Guessing: A Formal and Quantitative Study* NY: John Wiley & Sons.
- Watanabe, S. (1972) Pattern recognition as information compression, in Watanabe, S. (Ed.), *Frontiers of Pattern Recognition*. Academic Press., 561-567.