Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2003), *17*, 221–234. Printed in the USA.

DOI: 10.1017/S0890060403173052

Rapid development of knowledge-based systems via integrated knowledge acquisition

HAO XING,[1] SAMUEL H. HUANG,[2] AND J. SHI[2]

[1]Department of Mechanical, Industrial and Manufacturing Engineering, University of Toledo, Toledo, Ohio 43606, USA
[2]Intelligent CAM Systems Laboratory, Department of Mechanical, Industrial and Nuclear Engineering, University of Cincinnati, Cincinnati, Ohio 45221, USA

(RECEIVED April 11, 2002; ACCEPTED December 30, 2002)

Abstract

This paper presents a novel approach, which is based on integrated (automatic/interactive) knowledge acquisition, to rapidly develop knowledge-based systems. Linguistic rules compatible with heuristic expert knowledge are used to construct the knowledge base. A fuzzy inference mechanism is used to query the knowledge base for problem solving. Compared with the traditional interview-based knowledge acquisition, our approach is more flexible and requires a shorter development cycle. The traditional approach requires several rounds of interviews (both structured and unstructured). However, our method involves an optional initial interview, followed by data collection, automatic rule generation, and an optional final interview/rule verification process. The effectiveness of our approach is demonstrated through a benchmark case study and a real-life manufacturing application.

Keywords: Drop Hammer Forming; Data Discretization; Fuzzy Inference; Knowledge Acquisition; Knowledge-Based System; Rule Simplification

1. INTRODUCTION

Global industrial competition requires enterprises to use innovative technologies for manufacturing and provision of goods and services in the shortest possible time frame with minimum cost. To achieve the goal of assuring short development cycles for new products, they are developing a process-based knowledge-driven product development environment by employing information technology. One major emphasis is knowledge-based engineering (KBE), which focuses on acquiring, storing, and utilizing knowledge for design and manufacturing. One of the applications of KBE is the knowledge-based system (KBS), which represents an interactive computer-based decision-making tool that uses both factual and heuristic data acquired from domain experts for problem solving.

The success of a KBS critically depends on the amount of knowledge embedded in the system. Expert knowledge, which results from an individual's extensive problem-solving experience, has been described as *unconscious knowledge* (Mitta, 1989). As experts achieve greater competency, their ability to explain the fine details associated with problem solving strategies degrades. Intermediate solution steps are unconsciously performed as a matter of routine as strategies are compressed into a few major steps (Gaines, 1987). Thus, not all of the knowledge involved can be decoded from schema to a semantic representation that is available in human working memory and hence is not accessible for the experts to report. This is especially true in the field of manufacturing, where the experts are those who actually work with the machines on the shop floor. They usually do not receive formal scientific training, and most of their knowledge is biased toward their own heuristic. Although they can easily solve a complex problem, they usually have great difficulty explaining why and how they make a particular decision. This is the hurdle in smooth knowledge acquisition (Ham & Lu, 1988). In addition, the iterations involved in knowledge acquisition and incremental development for building up a complete KBS render the manual approach highly time consuming. Because knowledge acquisition is a lengthy and painful process, it is often referred to as the bottleneck in KBS development (Gonzalez & Dankel, 1993).

Reprint requests to: Samuel H. Huang, ICAMS, Department of Mechanical, Industrial and Nuclear Engineering, University of Cincinnati, Cincinnati, OH 45221. E-mail: shuang@gauss.mie.uc.edu

To overcome the knowledge acquisition bottleneck, we developed an integrated knowledge acquisition methodology, which targets manufacturing application, as shown in Figure 1. The key features and advantages of the methodology are that it can do the following:

- maintain a database of problem solving cases, which allows domain experts to verify their knowledge and discover data discrepancies and knowledge engineers to gain better understanding of the problem domain;
- automatically extract high-level rules from the database, which allow knowledge engineers to independently acquire rule-type knowledge and become "pseudo-experts" (as a result, they can interact more effectively with domain experts);
- integrate domain experts' subjective knowledge with general knowledge embedded in examples, which results in a system that covers a much wider problem domain than otherwise possible; and
- provide closed-loop knowledge acquisition and utilization, which results in optimal knowledge management to achieve true manufacturing intelligence.

2. BACKGROUND

In the 1970s, KBSs emerged as software systems that behave like human experts when solving application problems in certain domains. The major reason that prohibited both building and maintaining such systems was the lack of robust and highly efficient computational technologies. Therefore, developing new technologies received a great deal of attention. Facilities to develop large KBSs exist, for example, development methodologies such as KADS (Wielinga et al., 1992), environments such as KEATS (Motta et al., 1988), and shells such as DEX (Bohanec & Rajkovič, 1990). Constructing KBSs to test the feasibility of new technologies and approaches brought forth rather promising results. The field of KBSs has expanded enormously. Applications of KBS range from medical diagnosis, chemical analysis, geological exploration, computer configuration, and recently to real-time process control, nuclear power plant operation, and marine navigation (Coenen & Bench-Capon, 1993). In the manufacturing and production environment, expert systems have been used for tasks such as scheduling, engineering design, plant layout, manufacturing system modeling, product quality control, process planning, and production management (Mital & Anand, 1994).

However, the maintenance difficulty of KBSs hindered their general acceptance. In addition, the building of large-scale commercial KBSs is always associated with long development time, high costs, and high risk. Just as the software crisis resulted in the establishment of the discipline of software engineering, the unsatisfactory situation in the construction and maintenance of KBSs made clear the need for engineering the development procedure for KBSs. In other words, there is an urgent need to construct and maintain KBSs in a systematic and controllable manner (Studer et al., 1998).

A typical KBS consists of three basic components: the knowledge base (KB) indicating the domain application area, the system context and working memory, and the inference engine controlling the problem-solving strategy. The KB comprises the knowledge that is specific to the application domain, including such things as simple facts about the

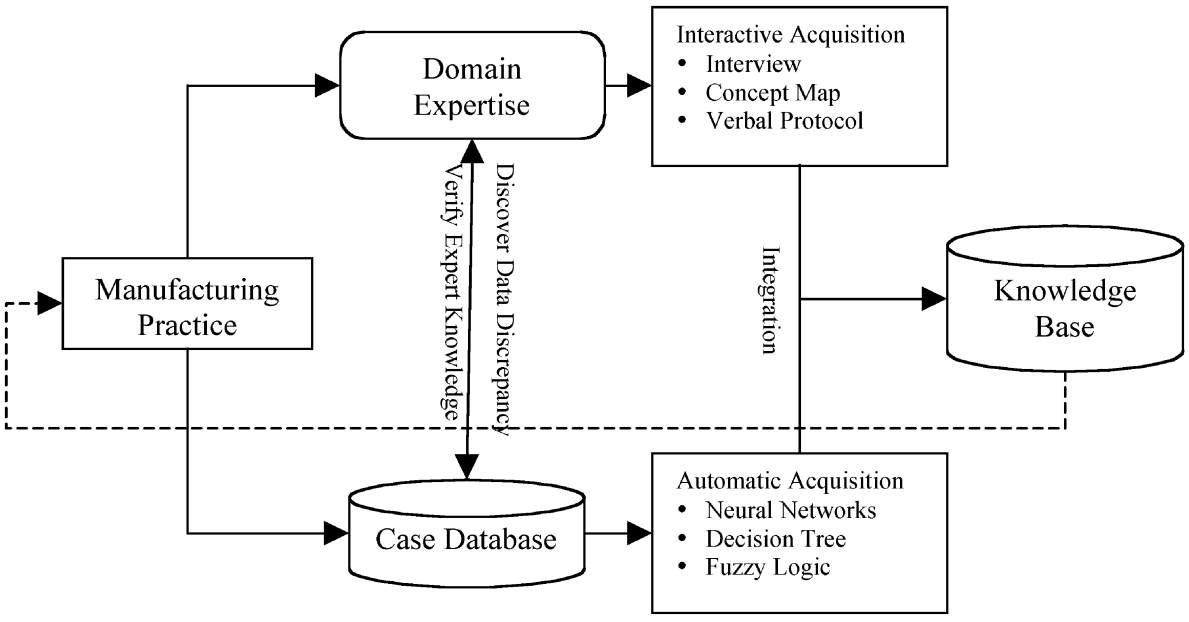


Fig. 1. The integrated knowledge acquisition methodology.

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

domain, rules or frames that describe relations or phenomena in the domain, and possibly also methods, heuristics, and ideas for solving problems in this domain. The inference engine contains methods of manipulating the knowledge in the KB.

The most popular KBS approach adopted by the industry, the symbolic artificial intelligence (AI) approach, is usually applied for constructing the KB of a KBS. In this approach, the knowledge is represented using IF–THEN production rules; in this way, the KB can be called the rule base (RB). Such a knowledge-based system is also called an expert system or production system. It symbolically manipulates IF–THEN rules to solve various problems.

Although this approach is successful, it suffers from the so-called knowledge acquisition bottleneck because it is challenging and time consuming to acquire and represent knowledge in terms of rules. In other words, this bottleneck obstructs the efficiency and effectiveness of building the RB of a KBS. In addition, the performance of the resultant KBS will be impaired, when the amount of extracted rules is very large and certain flaws such as redundancy, conflict, subsumption, and circularity exist (Higa, 1996).

Recent advance in information technology has resulted in several learning techniques that can potentially be used for automatic knowledge acquisition, including artificial neural networks and decision tree learning (Mitchell, 1997). Neural networks belong to a family of models that are based on a learning by example paradigm in which problem-solving knowledge is automatically generated according to actual examples presented to them. The knowledge, however, is represented at a subsymbolic level in terms of connections and weights. Neural networks act like black boxes in providing little insight into how decisions are made, which limits their usefulness in many decision support applications. On the other hand, decision trees can be easily translated into highly intelligible IF–THEN rules. However, current decision tree learning algorithms are only suited for problems that are described by a fixed set of attributes (adding new attributes is impractical or inconvenient); parameters should have discrete values (if not, discretization should be carried out); and the target function to be learned should have discrete output values. Some other shortcomings of decision tree learning are discussed in Quinlan (1993).

In order to overcome these problems, techniques such as data discretization, automatic rule extraction, and rule simplification and pruning are needed. These techniques, described in Section 3, are the major components of a systematic approach for rapid KBS development.

3. INTEGRATED KNOWLEDGE ACQUISITION

The system architecture for integrated knowledge acquisition is illustrated in Figure 2. Engineering raw data containing the cases is accumulated by observing and recording the operations from experienced operators. Raw data are preprocessed using methods such as discretization, normalization, and cleansing. Apparently, wrong cases are removed from raw data after consulting with domain experts. For those cases with similar inputs and outputs, some of them can be removed as redundant cases after data discretization. After data preprocessing, each of the input attributes can be described by a fuzzy linguistic term (such as small, medium, and large). Because each linguistic term can be represented by a set, a continuous-valued input can thus be classified into a specific set and represented in a binary scheme. After all the cases have been converted using linguistic terms, some inconsistent cases can be removed. This procedure can effectively scale down the cases in engineering raw data and makes the subsequent rule extraction easier. Finally, rules are extracted through neural network learning or using decision tree learning. These rules are described using humanly intelligible linguistic terms, which can be easily verified, modified, or supplemented by domain experts. Rule pruning and simplification will make the RB more concise and consistent. Because linguistic terms (which are imprecise) are involved, a Mamdani-type fuzzy inference mechanism is used as the inference engine.

It should be noted that, during the process of eliciting knowledge from industry personnel, some knowledge is deductive (from general to specific, from premise to conclusion), some is inductive (from individual cases to general conclusion), and some is abductive (unbounded inference). The KBSs discussed in this paper handle deductive knowledge. However, this paper points out that some quantitative method can be introduced to transfer the other two types of knowledge (inductive and abductive) into deductive knowledge. For example, probability can be used to measure the confidence of the conclusion of an abductive rule. Such quantitative information is represented by the weight of a rule, which will be discussed.

3.1. Discretization

Discretizing continuous parameters is inspired in part by the fact that experts usually describe parameters using linguistic terms instead of an exact value. There are several advantages of data discretization: it provides regularization because it is less prone to variance in estimation from small fragmented data, the amount of data can be greatly reduced because some redundant data can be identified and removed, and some rule extraction method such as decision trees can perform better if all of the continuous-valued attributes are discretized before rule induction.

Equal width is one of the most frequently used unsupervised data discretization methods. It is also the simplest one, which divides the range of observed values for a variable into N intervals of equal sized intervals. The range of each interval is calculated by

$$\text{range of interval} = \frac{\text{range of observed attribute}}{\text{number of intervals}},$$

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

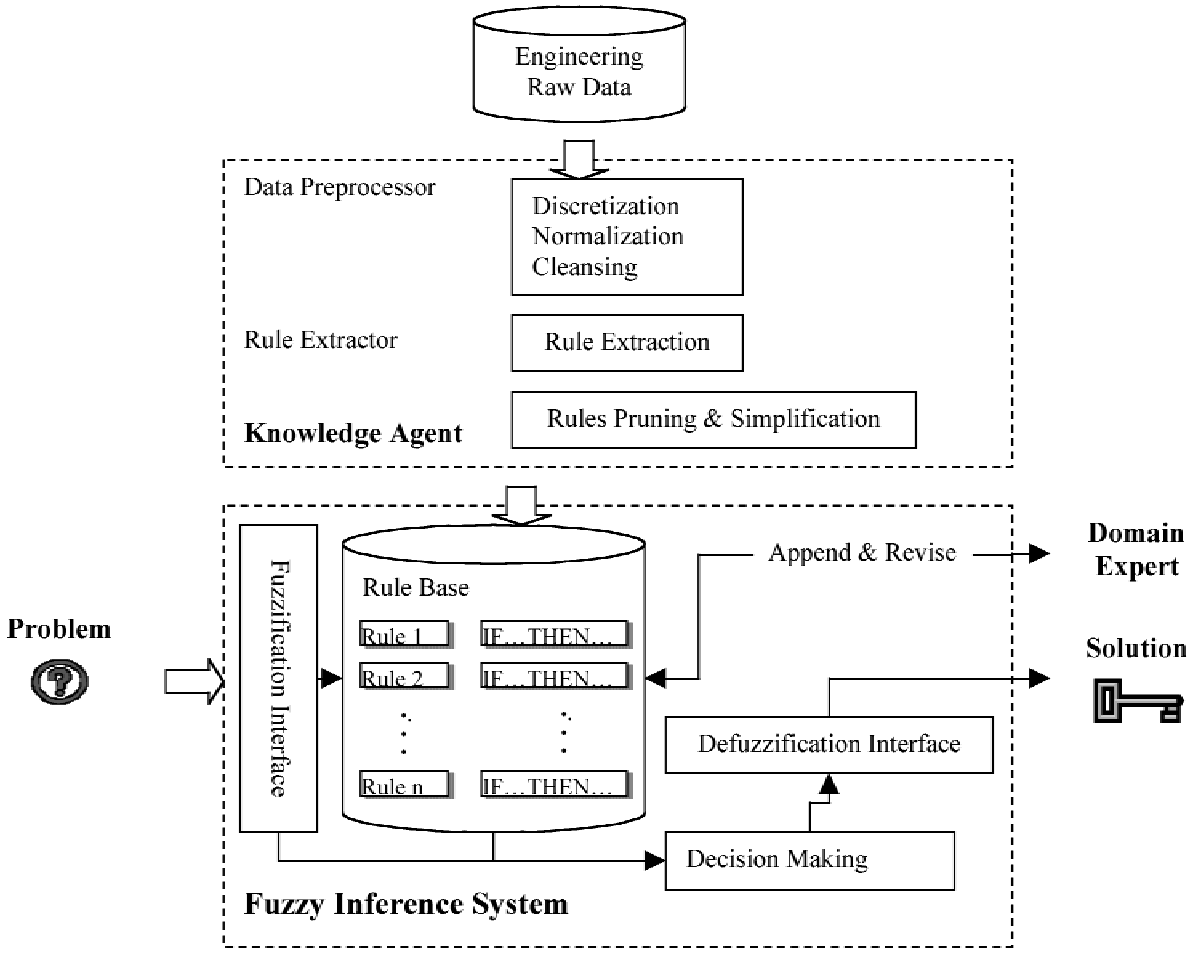


Fig. 2. The system architecture.

where the number of intervals should be determined by users.

This method applies to the situation where the distribution of sample data is quite uniform. However, it is vulnerable to outliers that may drastically skew the range (Catlett, 1991). A statistically justified heuristic method for supervised discretization is the *chi-merge* algorithm (Kerber, 1992). This algorithm begins by placing each observed real value into its own interval and proceeds by using the χ^2 test to determine when adjacent intervals should be merged. The extent of the merging process is controlled by the use of a χ^2 threshold α, indicating the maximum χ^2 value that warrants merging two intervals. The author reports that on random data, a very high threshold must be set to avoid creating too many intervals.

Chi2 (Liu & Setiono, 1997) is a variant of the chi-merge algorithm. It consists of two phases. In phase 1, a rough common α (significance level) for all variables is used. In phase 2, the intervals are further merged using a separate α for each variable. In general, Chi2 can form simpler terms, which leads to improved predictive accuracy. However, it is computationally complicated because it takes each different data point as an initial interval. In addition, it merges variables sequentially rather than trying to minimize the inconsistency resulting from merging. To overcome these drawbacks, we developed a *greedy chi-merge* algorithm for data discretization.

Greedy chi-merge also contains two phases: crude discretization and fine-tuning. Compared to the Chi2 algorithm, which takes the number of distinct values of a variable as the initial number of intervals, greedy chi-merge performs an initial partitioning based on the change in the output value, as illustrated in Figure 3. In this way, the number of intervals for each variable can be greatly reduced.

The next phase, fine tuning, is based on greedy search, which always chooses to merge the intervals of a variable that causes the least inconsistency rate among all mergeable variables. Although all the attributes are considered simultaneously, only the merging that causes the lowest inconsistency rate takes place. The pseudocodes of the greedy chi-merge are shown in Figure 4.

The problem of data inconsistency results when, after discretization, data samples exist in which the inputs are the same but the outputs are different. The inconsistency

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

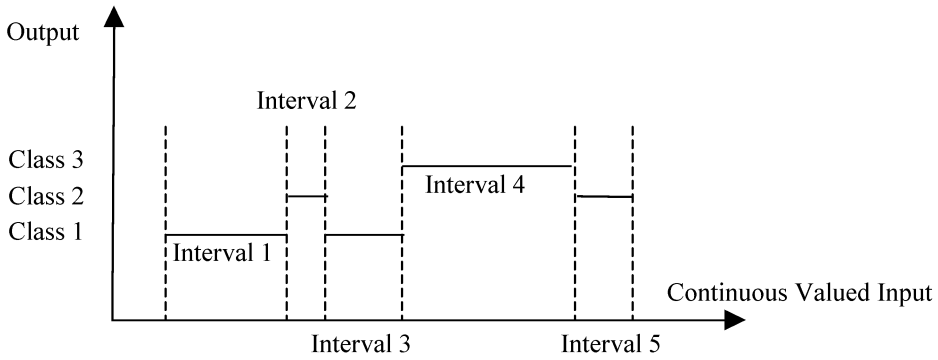


Fig. 3. The initial partitioning of greedy chi-merge.

Greedy Chi-merge Algorithm

Phase 1:

Use rough discretization method to get rough number of intervals for each continuous attribute. Form numeric attribute lists ($list_i$, $i = 1, 2, \ldots N$. *N is the number of attributes*).

Here a list ($list_i$) *is made of a sequence of numeric pairs, which separates the whole region* (*from* a_i *to* b_i) *of an attribute into certain intervals* (M_i). *That is,* $list_i = \{(a_i,v_{i1}), (v_{i2},v_{i3}), \ldots (v_{iMi},b_i)\}$.

Phase 2:

```
for each attribute i {
    set αi = α0; // The significance level (100%-α) to check Chi^2 values
}
set mergeable = true;
while (mergeable == true) {
    set mergeable = false;
    for each numeric attribute list (listi) {
        tempListi ← listi  // Save current discretized interval list temporarily
        // Try to merge neighbor discretized interval; Calculate the Chi^2 value; and
        // Get the minimum Chi^2 value
        set minChi2 = MinChi2 (tempListi);
        if (minChi2 < the value looked up in the Chi^2 table based on αi) {
            MergeInterval(tempListi); // Merging in the temporary list
            set inConi = InConCheck(tempList); // Calculate the inconsistency rate
            tempListi ← listi; // Recover the interval list
        }
        else{
            set inConi = ∞;
        }
        set αi = DecreSig(αi); // Decrease the significance level
    }
    // Find minimum inconsistency rate (The related index is "mini_irate")
    set minInCon = MinInCon(inCon);
    if (minInCon < δ) {
        MergeInterval(list_mini_irate); // Real merging
        set mergeable = True;
    }
}
```

Fig. 4. The pseudocodes for greedy chi-merge.

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

rate is calculated using the following expression, based upon the method proposed by Liu and Setiono (1997):

$$\sum_i (n_i - c_{i,\max})/N,$$

where N is the total number of cases, n_i is the number of cases with the same input ($i = 1, 2, \ldots, I$), $\sum_i n_i = N$; $c_{i,j}$ is the count of class j in n_i ($j = 1, 2, \ldots, J$), $\sum_i c_{i,j} = n_i$; and

$$c_{i,\max} = \max_j (c_{i,j}).$$

3.2. Rule extraction

Once all continuous variables are discretized, linguistic terms can be used to describe these variables. The original data samples are converted accordingly. As a result, all the inputs and outputs become discrete. A procedure, which utilizes the learning ability of a neural network, has been developed to extract rules for these converted data samples. The details of the procedure have been described in a previous publication (Huang et al., 2001). An outline of the five-step approach is provided in Figure 5. In summary, the converted data is represented using a binary scheme and used to train a neural network, which is specially structured according to the linguistic terms used. Linguistic IF–THEN rules are then extracted by analyzing the connections and weights of the trained neural network, using a dynamic depth-first search algorithm to reduce computation complexity.

An alternative rule extraction approach is through the use of decision tree learning. Decision trees are currently a highly developed technique for partitioning data samples into a set of covering decision rules (Weiss & Kulikowski, 1991). Decision tree learning algorithms such as ID3 and C4.5 have been widely publicized by Quinlan. Readers interested in the details should refer to Quinlan (1986, 1993). Once the decision tree is constructed for a specific problem, rules can be extracted directly from left to right and from top to bottom of the decision tree. Usually the number of rules extracted equals the number of leaves existing in the decision tree. Decision tree induction algorithms scale up very well for large data sets.

The extracted rules, represented using a linguistic IF–THEN format, are easily understood. They can be used to enrich the knowledge engineer's understanding of the underlying domain. With this knowledge, the knowledge engineer can have a more intelligent interview with domain experts. Asking domain experts more detailed questions can result in more refined and comprehensive knowledge in that domain. The knowledge engineer can then transfer the unstructured knowledge acquired from domain experts into structured knowledge in IF–THEN format and combine these rules with rules automatically extracted from data samples.

3.3. Rule simplification and pruning

The performance of a KBS will be adversely affected when the KB increases in size and contains conflicting rules. In this sense, a compact KB is desirable. However, a trade-off always exists between completeness (covering the entire knowledge domain) and compactness (using as few rules as possible). In the manufacturing domain, shop floor operators prefer to have a small amount of highly intuitive rules by sacrificing a small loss in predictive accuracy. Therefore, we believe rule simplification and pruning are justified in manufacturing application. Two concepts, namely, PofM (Possibility of Merging) and AofP (Accuracy of Prediction), are introduced to reduce the size of a KB while maintaining an acceptable level of performance. Details of rule simplification and pruning are discussed as follows.

When rules are provided by domain experts, they may be in a compound format. A rule that contains OR and/or NOT is a compound rule, which brings potential and indetectable inconsistency and thus hinders the maintainability of rules (Higa, 1996). Automatically extracted rules are never in a compound format. Compound rules acquired manually should be transformed in order to maintenance consistency. For example, we have the following rule:

- IF (a_1 is LARGE OR a_2 is MEDIUM) AND (a_3 is NOT SMALL) THEN o_1 is MEDIUM

Assume that a_3 is described using three linguistic terms, SMALL, MEDIUM, and LARGE. The rule can be transformed into four noncompound rules:

- IF a_1 is LARGE AND a_3 is MEDIUM THEN o_1 is MEDIUM
- IF a_1 is LARGE AND a_3 is LARGE THEN o_1 is MEDIUM
- IF a_2 is MEDIUM AND a_3 is MEDIUM THEN o_1 is MEDIUM
- IF a_2 is MEDIUM AND a_3 is LARGE THEN o_1 is MEDIUM

A typical problem with automatically extracted rules is that the rules could overlap, which means these rules can be combined to form a simpler rule. For example, we have the following three rules:

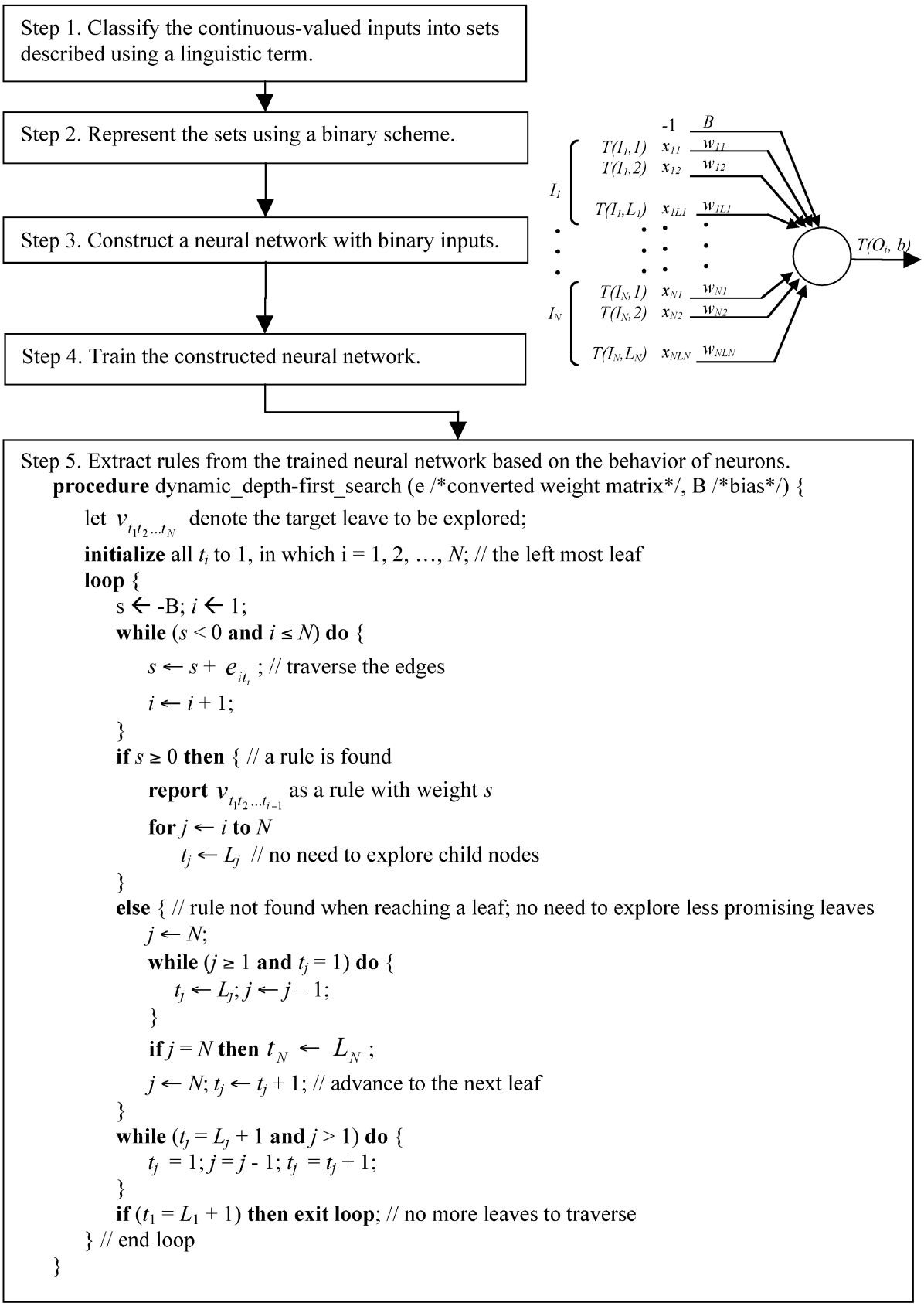
- IF a_1 is SMALL AND a_2 is LARGE THEN o_1 is LARGE
- IF a_1 is MEDIUM AND a_2 is LARGE THEN o_1 is LARGE
- IF a_1 is LARGE AND a_2 is LARGE THEN o_1 is LARGE

Assume that a_1 is described using three linguistic terms, SMALL, MEDIUM, and LARGE. These three rules can be combined into one simpler rule:

- IF a_2 is LARGE THEN o_1 is LARGE

The Quine–McCluskey algorithm is known as an approach of Boolean functions simplification (McCluskey, 1956). In this paper, it is used to automatically combine overlapping rules that may exist in the automatically extracted rule set. When combining manually acquired rules

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press



Step 5. Extract rules from the trained neural network based on the behavior of neurons.

```
procedure dynamic_depth-first_search (e /*converted weight matrix*/, B /*bias*/) {
    let v_{t1 t2 ... tN} denote the target leave to be explored;
    initialize all t_i to 1, in which i = 1, 2, …, N; // the left most leaf
    loop {
        s ← -B; i ← 1;
        while (s < 0 and i ≤ N) do {
            s ← s + e_{i ti} ; // traverse the edges
            i ← i + 1;
        }
        if s ≥ 0 then { // a rule is found
            report v_{t1 t2 ... t(i-1)} as a rule with weight s
            for j ← i to N
                t_j ← L_j // no need to explore child nodes
        }
        else { // rule not found when reaching a leaf; no need to explore less promising leaves
            j ← N;
            while (j ≥ 1 and t_j = 1) do {
                t_j ← L_j; j ← j – 1;
            }
            if j = N then t_N ← L_N ;
            j ← N; t_j ← t_j + 1; // advance to the next leaf
        }
        while (t_j = L_j + 1 and j > 1) do {
            t_j = 1; j = j - 1; t_j = t_j + 1;
        }
        if (t_1 = L_1 + 1) then exit loop; // no more leaves to traverse
    } // end loop
}
```

Fig. 5. Rule extraction using a neural network.

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

with automatically extracted rules, especially when compounded rules are transformed, redundant rules may appear. Redundant rules means that rules having identical premises also have identical consequences. Removal of redundant rules is trivial because it can be realized through a simple search.

Rules are simplified once redundant rules are removed and overlapping rules are combined. A further step is to prune these rules to make the rule base more compact. The criterion used for pruning is the generalization ability (predictive accuracy). The user may provide a testing data set and define an acceptable level of predictive accuracy. Rules are pruned by merging similar rules and removal of insignificant rules, as long as these actions do not result in a predictive accuracy with an unacceptable low level. This is accomplished by using the concepts of PofM and AofP.

When a weight is associated with a rule, as in the case of rule extraction using dynamic depth-first search (the neural network approach), an insignificant rule means a rule that has a small weight. If a rule does not have a weight originally, as in the case of rule induction using ID3 or rule provided by a domain expert, an insignificant rule is one that applies to few data samples. In this case, we can assign a weight to each rule based on how many data samples they apply. Thus, weight is used as a unified measure for rule significance.

The rule simplification and pruning algorithm is shown in Figure 6 using the following definitions:

- SR (Similar Rules): rules that have identical output values and only one different input value.
- KI (Key Input): the input that distinguishes SR.
- SRG (Similar Rule Group): a rule group that holds all SR. Note that a rule can belong to different SRG.
- MSRG (Mergeable Similar Rule Group): a SRG where the KI values of all the rules have adjacent values and the number of the rules equals to the number of available values for the KI. For example, the following rules can be put into a MSRG:
  - IF a_1 is LARGE AND a_2 is MEDIUM THEN o_1 is MEDIUM
  - IF a_1 is SMALL AND a_2 is MEDIUM THEN o_1 is MEDIUM
  - IF a_1 is MEDIUM AND a_2 is MEDIUM THEN o_1 is MEDIUM
- PofM: each MSRG has a PofM value

$$\text{PofM} = \frac{\text{number of rules in mergeable similar rule group}}{\text{total number of intervals of key input}}$$

- DC (Don't Care): An input does not appear in a rule; its value can be ignored by the rule.
- AofP: In general, previous unseen cases are used to test the predictive accuracy of a rule base. AofP is defined as

$$\text{AofP} = \frac{\text{number of correct predictions}}{\text{total number of testing cases}}$$

Note that all rules are in digital format, for example, 2 0 4 −1 5. The first 3 digits stand for the input value, 5 is the output value, and −1 is for wherever the DC attribute exists. Each attribute is represented by a number, ranging from 0 to the number of intervals −1. Take the three rules listed above for MSRG as example. Two attributes are concerned, a_1 and a_2. Both have three intervals; so does the output. The three rules are represented as 2 1 1, 0 1 1, and 1 1 1.

4. CASE STUDIES

4.1. Benchmark case

To illustrate how to use automatic knowledge acquisition to rapidly develop a KBS and to demonstrate its effectiveness, the popular iris flower classification problem is used as a benchmark case study. The problem is based on a data set first used by Fisher (1936) for illustrating discriminant analysis techniques. The classification problem then became a standard benchmark problem for testing different classification methods. The data set can be obtained from the University of California at Irvine ftp server (ftp://128.195.1.46/pub/machine-learning-databases/iris/). It contains 150 patterns: 50 of them belong to the class of *Iris setosa*, 50 belong to the class of *Iris versicolor*, and 50 belong to the class of *Iris virginica*. There are four input parameters to describe the three classes of flowers, namely, sepal length, sepal width, petal length, and petal width. They are all continuous-valued attributes.

A knowledge-based system is developed with the following steps. First, after discretization with greedy chi-merge, two parameters of sepal length and sepal width are removed because they have only one interval. Both petal length and petal width have three intervals after merging. The discretization result is shown in Table 1.

Second, we used 75 data samples with odd index for training and the rest for testing. Using the dynamic depth-first search algorithm, four rules are extracted as follows. The weight attached to a rule shows its significance degree, as described in Section 4. The weights of rules have been normalized for easy handling, so weight 1.0 stands for the most significant and weight 0.0 for insignificant.

- IF petal length is SMALL AND petal width is SMALL, THEN iris class is Iris Setosa (weight 1.0)
- IF petal length is SMALL AND petal width is LARGE, THEN iris class is Iris Setosa (weight 0.035636)
- IF petal length is MEDIUM, THEN iris class is Iris Versicolour (weight 1.0)
- IF petal length is LARGE, THEN iris class is Iris Virginica (weight 1.0)

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

Rule Simplification and Pruning Algorithms

- M is the number of different output values.
- N is the number of premises of the rule.

1. Transfer compound rules and get the rule set A.
2. Remove overlapping rules based upon Quine-McCluskey algorithm (Combine *MSRG*):
   a. Group rules in A into M sub sets (B_i, i = 1, 2, …, M) according to the output intervals, $A = \sum B_i$.
   b. Assign rule set $A' = \varnothing$.
   c. For each B_i:
      i. Group rules in B_i into $N+1$ sub sets (C_j, j = 0, 1, 2, …, N) according to the *DC* numbers. A rule in C_j has j *DC*s. $B_i = \sum C_j$.
      ii. For each C_j (j = 0, 1, …, N-1):
         a) Group rules in C_j into $N+1$ sub sets (D_k, k = 0, 1, 2, …, N) according to the zero numbers in the rules. A rule in D_k has k zeros. $C_j = \sum D_k$.
         b) Repeat R = COMBINE(D_k, D_{k+1}), k = 1, 2, …, N-1.
         For each rule in D_k, find *SR* in D_{k+1}. If the resulted *SRG* is a *MSRG*, (1) put a combined new rule in R; (2) rules in *SRG* are removed from D_k; and (3) if $k = N$-1, rules in *SRG* are removed from D_{k+1} as well.
         c) $C_{j+1} = C_{j+1} + R$.
         d) $A' = A' + \sum D_k$
   d. Assign rule set $A = A'$. A is the simplified rule set.
3. Combine rules with similar inputs (rules have only one different input value but may have different output values) in A.
   a. Group rules in A into N+1 sub sets (C_j, j = 0, 1, 2, …, N) according to the *DC* numbers. $A = \sum C_j$.
   b. For each C_j (j = 0, 1, …, N-1):
      i. Group rules in C_j into $N+1$ sub sets (D_k, k = 0, 1, 2, …, N) according to the zero numbers in the rules. $C_j = \sum D_k$.
      ii. Repeat R = COMBINESIMILAR(D_k, D_{k+1}), k = 1, 2, …, N-1.
      For each rule in D_k, find rules with similar inputs in D_{k+1}. If the resulted rule set satisfies (1) *PofM* is acceptable; and (2) *AofP* is acceptable, (1) put a combined new rule (with an output value determined by the majority of rules) into R; (2) rules in the resulted rule set are removed from D_k; and (3) if $k = N$-1, rules in the resulted rule set are removed from D_{k+1} as well.
      iii. $C_{j+1} = C_{j+1} + R$.
      iv. $A' = A' + \sum D_k$
   c. Assign rule set $A = A'$. A is the simplified rule set.
4. Remove redundant rules.
5. If rules are weighted, carry our the rule pruning procedure:
   a. Normalizing rule weights.
   b. Set *RW* a small value (normally given by the user, e.g., 0.25).
   c. Remove rules with weight < *RW* value. Check the *AofP* value. If *AofP* is acceptable currently, go to d; else go to e.
   d. Increase *RW* and return back to c.
   e. If c is the 1st removal operation, undo it.
   f. Decrease *RW* and return back to c.

Fig. 6. Rule simplification and pruning and an algorithm.

Third, after rule simplification and pruning, the following rules are obtained:

- IF petal length is SMALL AND petal width is SMALL, THEN iris class is Iris Setosa (weight 1.0)

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

Table 1. *Data discretization for the Iris classification problem*

Attribute Name	No. Intervals	Linguistic Terms	Interval Ranges
Sepal length	1	N/A	{4.40–7.70}
Sepal width	1	N/A	{2.00–4.10}
Petal length	3	Small, medium, large	{1.00–3.00, 3.00–5.00, 5.00–6.90}
Petal width	3	Small, medium, large	{0.10–1.00, 1.00–1.70, 1.70–2.50}

N/A, not available.

- IF petal length is MEDIUM, THEN iris class is Iris Versicolour (weight 1.0)
- IF petal length is LARGE, THEN iris class is Iris Virginica (weight 1.0)

Fourth, these rules are used to develop a fuzzy inference system. Triangular membership functions are used for the input parameters and singleton spikes are used for output. Among the 75 testing data samples, 70 are correctly classified.

A comparison with prior results published in the literature is shown in Table 2. Although our approach did not achieve 100% accuracy, it yielded a more concise rule set (three rules) while still maintaining high accuracy (93.3%). Compared with the approach we proposed, REFuNN and RULES-3 Plus need many more rules to attain higher accuracy; REFuNN and Premises Learning use much larger training sets, which means higher computing cost. A trade-off exists here between conciseness and complexity. In a manufacturing environment, easy to understand and concise rules are preferred by operators to guide their daily operations. Therefore, a smaller rule set that has acceptable performance is desirable.

4.2. Industrial application

The integrated knowledge acquisition approach is also successfully applied to a real-world manufacturing process, drop hammer forming. Drop hammer forming is one of the most versatile processes for forming sheet metal parts, which is used extensively by B.F. Goodrich Aerospace/Aerostructures group in the production of aircraft engine nacelles. Basically, three process methods are used for drop hammer forming: bare punch, blow down, and blow up. Bare punch is the simplest procedure. Parts are produced using a single hammer strike and no forming aids are needed. Blow down and blow up procedures require the use of forming aids and multiple hammer strikes. The forming aids, usually rubbers, are placed on (blow down) or under (blow up) the part to form a pyramid. During the forming process, the pyramid causes the impact force to flow in the direction of the least rubber build up, thus preventing wrinkles and folding.

Operator skills play a critical role in assuring product quality in drop hammer forming. Experienced operators can produce high quality products by adjusting die configuration, using appropriate forming aids, and controlling punch blow. Their skills are accumulated through years of hands-on experience, and they have great difficulty explaining how and why they make a certain decision. Therefore, new operators have to learn through observation. This is very inefficient and it usually takes at least 6 months for a new operator to gain enough knowledge to be able to work independently. B.F. Goodrich engineers attempted to use traditional interview-based techniques to develop a KBS. They were not successful, mainly because the knowledge engineers have very little understanding of the drop hammer forming process, while experienced operators typically have no formal engineering education and were unable to explain the decisions they made.

Table 2. *Comparison of the Iris classification problem*

Method	No. Rules	Predictive Accuracy
Our approach	3	93.3% (75 training cases)
REFuNN (Kasabov, 1996)	8	100, 90, and 90% (using 3 different fuzzy inference techniques with 120 training cases)
RULES-3 Plus (Pham & Dimov, 1996)	10	97.5% (70 training cases)
ID3	8	92.5%
Premises Learning (Xiong et al., 2002)	4	98.6% (150 training cases)

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

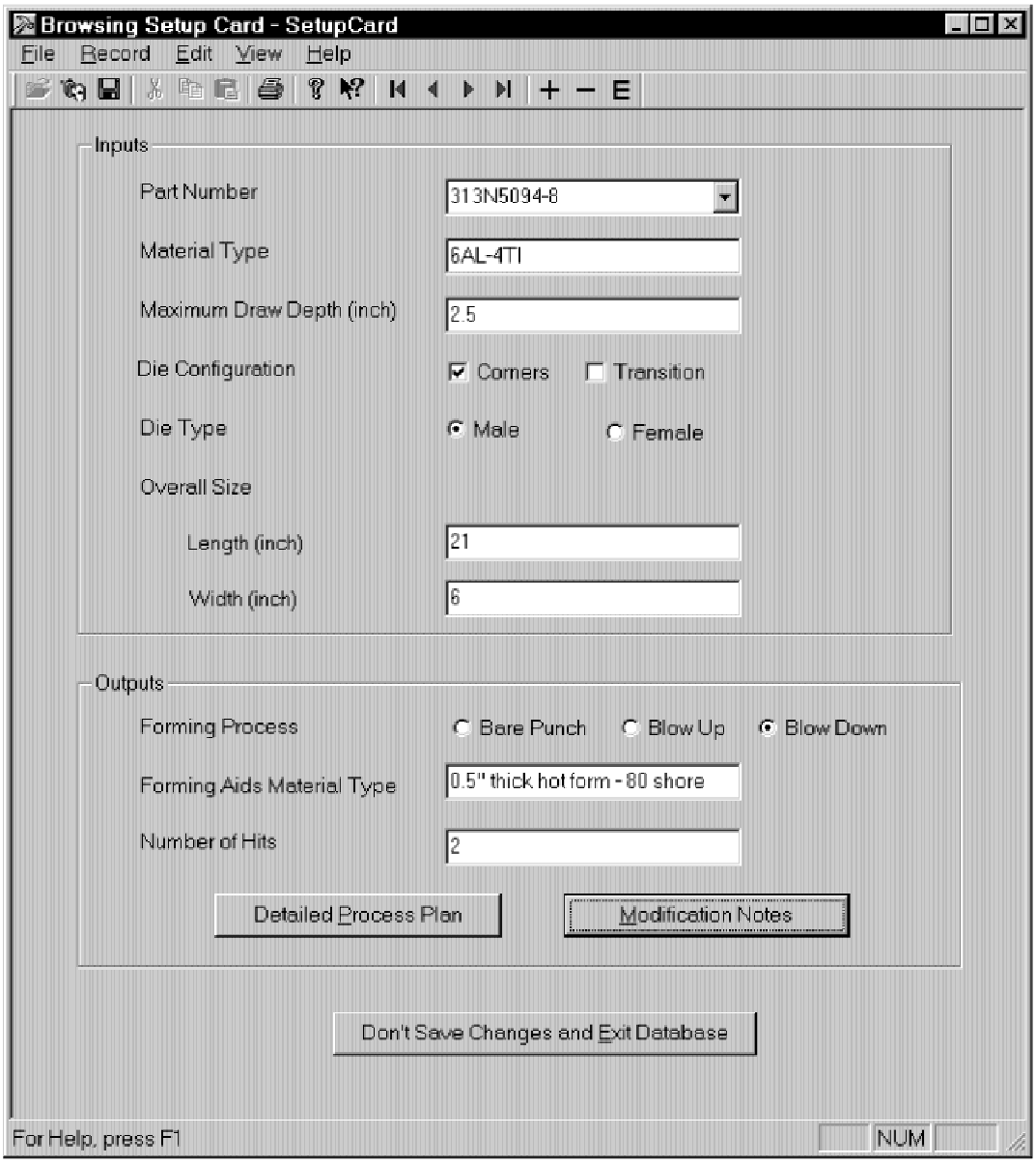


Fig. 7. The drop hammer forming digital assistant.

Armed with the integrated knowledge acquisition method, a student was sent to B.F. Goodrich's drop hammer forming facility at Chula Vista, California. An initial interview with domain experts identified the following inputs: material type, maximum draw depth, presence of transitions and corners, and die type; while the outputs are identified as the forming process method, number and type of forming aids, and number of punch blows. For new operators, to decide which forming process method is the most challenging task. Therefore, we worked on rule extraction for process method determination first. We collected 73 historical cases, 35 of them are used for training and the remaining used for testing. Among all of the inputs, only maximum draw depth has continuous values. We used equal width as the discretization method and four intervals were formed. The following rules are obtained:

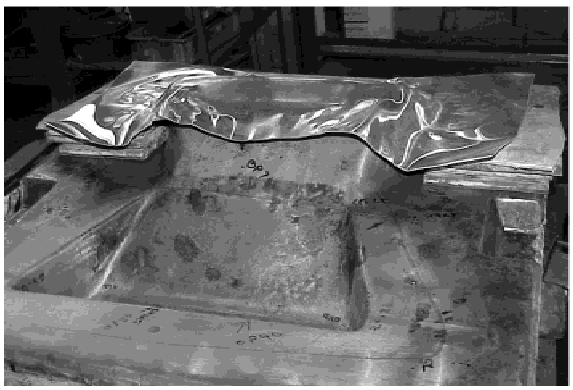
- IF maximum draw depth is LARGE AND material type is SOFT AND transitions and corners is YES, THEN forming process is BLOW UP (weight 1.0)
- IF maximum draw depth is LARGE AND material type is SOFT AND die type is MALE, THEN forming process is BLOW UP (weight 0.297073)
- IF maximum draw depth is LARGE AND material type is HARD AND transitions and corners is YES AND die type is MALE, THEN forming process is BLOW UP (weight 0.267993)

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

- IF transitions and corners is YES, THEN forming process is BLOW DOWN (weight 1.0)
- IF transitions and corners is NO, THEN forming process is BARE PUNCH (weight 1.0)

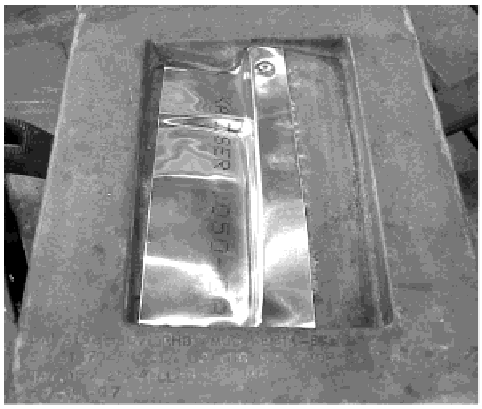
These rules achieved a 95% predictive accuracy when applied to the testing cases. With these rules, the student gained a good understanding of the drop hammer forming process. A final interview with domain experts was then conducted. The experts concluded that these rules are valid and were able to provide additional details. Specifically, part material can determine the selection of different types of rubber as forming aids. If part material is hard and heat treatment is required before the forming process is applied, then harder and more heat resistant rubbers are needed as forming aids. In addition, thinner rubber pads are used for harder parts. The material of a part, combined with its maximum draw depth, can be used to decide the number of punch blows needed to produce the part. Softer materials, such as Aluminum, can be drawn in one stage to around 1 in. However, hard materials, such as titanium, can only be drawn in one stage to a maximum of 0.5 in. When the thickness of forming aids is determined, denoted t, then the number of punch blows (denoted n) can be easily calculated as $n = m/t + 1$. One sheet of forming aids is removed after each punch blow. The last blow is applied when all the forming aids are removed. Combining the knowledge acquired, a software tool is then developed to assist operator decision making in drop hammer forming. Figure 7 shows a snapshot of the software tool.

(a)

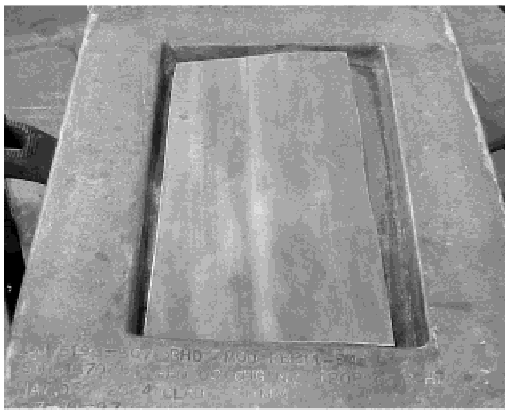


(b)

Fig. 8. A part that requires blow up: (a) tool and (b) configuration.



(a)



(b)

Fig. 9. A part that requires blow down: (a) part and tool and (b) forming aids placement.

Some examples are used to illustrate how the acquired process knowledge is applied for drop hammer forming process planning. The tool shown in Figure 8a is used to produce an aluminum part. Its maximum draw depth is 3 in. Since it is a male tool and there are four corners, forming aids need to be placed under the part (see Fig. 8b). In other words, the blow up process method is used. Since aluminum is soft, rubber pads are used as the forming aid. Three 1-in. rubber pads are used since the maximum draw depth is 3 in. One sheet of rubber is removed after each punch blow. The last punch blow is applied when all three sheets of rubber are removed. Therefore, a total of four punch blows are needed.

The tool shown in Figure 9a is also used to produce an aluminum part. Its maximum draw depth is 0.5 in. Since it is a female tool and there are transitions, the blow down

https://doi.org/10.1017/S0890060403173052 Published online by Cambridge University Press

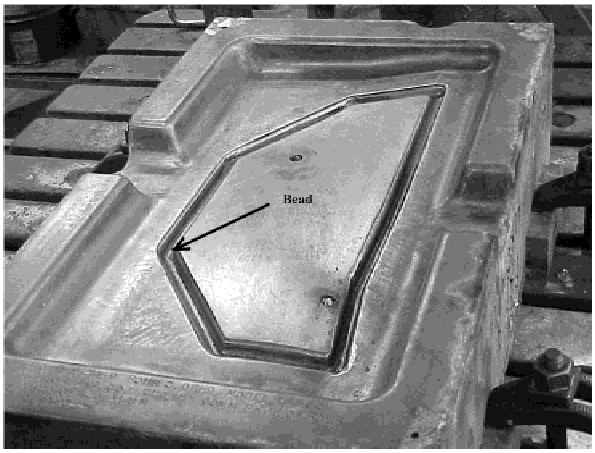


Fig. 10. A nearly flat part with no corners and transitions.

process method is used. One sheet of 0.5-in. rubber pad approximately the size of the part is placed on top of the part (Fig. 9b). After one punch blow is applied, the rubber pad is removed. Another punch blow is then applied to obtain the final part.

Figure 10 shows a tool that is used to produce an aluminum part. The tool is essentially flat, with a maximum draw depth (the difference between the highest and the lowest point of the area enclosed by the bead) of 0.05 in. It has no transitions. Note that the bead is used to lock the part. Excess material will then be trimmed away to obtain the final part. Therefore, there are no corners either. In this care, bare punch is sufficient to produce the part. Care should be exercised when determining the presence of transitions.

Figure 11 shows a not-so-flat part that clearly shows a transition, which needs to be produced using blow down rather than bare punch.

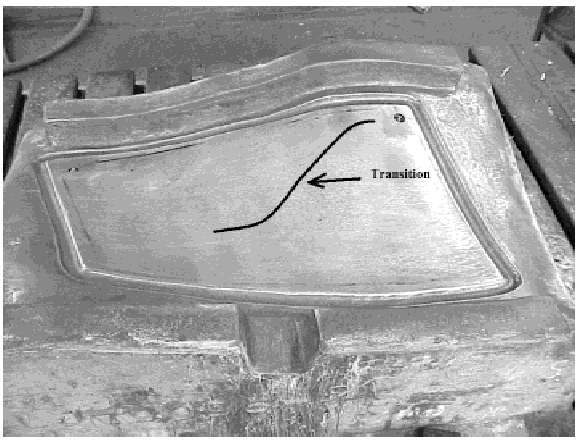


Fig. 11. A not-so-flat part with a transition.

5. CONCLUSION

This paper presents an integrated automatic/interactive knowledge acquisition approach to rapidly develop knowledge-based systems. This approach relies on several key algorithms, including data discretization, rule extraction, and rule simplification and pruning. After data discretization, linguistic terms can be used to describe the input and output parameters of the underlying problem. This allows the extraction of IF–THEN rules that are compatible with domain experts' heuristic reasoning. As a result, domain experts can easily verify these rules and provide additional rules in the same format. These rules are then combined and processed using rule simplification and pruning, which results in a more compact rule base. This paper recognizes the trade-off between completeness and compactness of rules. The goal is to obtain concise and easy to understand knowledge while maintaining an acceptable level of system performance.

A benchmark case study showed that the proposed approach has satisfactory performance. An industrial case study further showed its practicality and effectiveness. The research work related to this paper is useful to build up knowledge based systems with small rule sets. This is somehow a neglected aspect of AI in engineering, especially in manufacturing. The concise rules can form the foundation of training modules that can jump-start workers' skills to new and higher levels of competence.

The effectiveness of the algorithms also depends on the quality of the training and testing data. This paper assumes the data used is representative of the problem domain. If not, some preprocessing should be carried out, which forms another direction of research work. Major tasks in this area include missing data treatment, data cleansing, dimensionality reduction, and so forth, which are beyond the scope of this paper. Further research will focus on automatic tuning of the developed KBS. Note that fuzzy inference is used to operate on the RB; thus, the membership functions used will influence the predictive accuracy. We are currently developing automatic tuning algorithms based on neural network learning that can adjust parameters of membership functions for improved performance.

ACKNOWLEDGMENTS

This research is supported in part by the Ohio Aerospace Institute (OAI) under the 1999 Collaborative Core Research Program Phase II Award (CCRP 98-1-002) and a summer research internship from B.F. Goodrich Aerospace/Aerostructures Group.

REFERENCES

Bohanec, M., & Rajkovič, V. (1990). DEX: An expert system shell for decision support. *Sistemica 1(1)*, 145–157.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *Machine Learning-EWSL-91* (Kodratoff, Y., Ed.), pp. 164–178. Porto, Portugal: LNAI.

Coenen, F., & Bench–Capon, T. (1993). *Maintenance of knowledge-based systems: Theory, techniques and tools*. New York: Academic Press.
Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 2*, 179–188.
Gaines, B.R. (1987). An overview of knowledge acquisition and transfer. *International Journal of Man–Machine Studies 26*, 453–472.
Gonzalez, A.J., & Dankel, D.D. (1993). *The Engineering of Knowledge-Based Systems*. Englewood Cliffs, NJ: Prentice–Hall.
Ham, I., & Lu, S.C.-Y. (1988). Computer-aided process planning: The present and the future. *Annals of the CIRP 37(2)*, 1–11.
Higa, K. (1996). An approach to improving the maintainability of existing rule bases. *Decision Support Systems 18*, 23–31.
Huang, S.H., Xing, H., & Wang, G. (2001). Intelligent classification of drop hammer forming process method. *International Journal of Advanced Manufacturing Technology 18(2)*, 89–97.
Kasabov, N.K. (1996). Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy Sets and Systems 82*, 135–149.
Kerber, R. (1992). Discretization of numeric attributes. *Proc. Tenth National Conf. Artificial Intelligence*, pp. 123–128. Cambridge, MA: MIT Press.
Liu, H., & Setiono, R. (1997). Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering 9(4)*, 642–645.
McCluskey, E.J., Jr. (1956). *Minimization of Boolean functions*. PhD Thesis. Massachusetts Institute of Technology.
Mital, A., & Anand, S. (1994). *Handbook of Expert Systems Applications in Manufacturing*. New York: Chapman & Hall.
Mitchell, T.M. (1997). *Machine Learning*. Boston: McGraw–Hill.
Mitta, D.A. (1989). Knowledge acquisition: Human factors issues. *Proc. Human Factors Society 33rd Annual Meeting*, pp. 351–355.
Motta, E., Eisenstadt, M., Pitman, K., & West, M. (1988). Support for knowledge acquisition in the knowledge engineer's assistant (KEATS). *Expert Systems 5(1)*, 6–28.
Pham, D.T., & Dimov, S.S. (1996). An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition 30(7)*, 1137–1143.
Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning 1(1)*, 81–106.
Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
Studer, R., Fensel, D., Decker, S., & Benjamins, V.R. (1998). Knowledge engineering: Survey and future directions. In *Lecture Notes in Artificial Intelligence 1570, Knowledge-Based Systems: Survey and Future Directions* (Puppe, F., Ed.). New York: Springer.
Weiss, S.M., & Kulikowski, C.A. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. San Mateo, CA: Morgan Kaufmann.
Wielinga, B.J., Schreiber, A.T., & Breuker, J.A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition 4(1)*, 127–161.
Xiong, N., Litz, L, & Ressom, H. (2002). Learning premises of fuzzy rules for knowledge acquisition in classification problems. *Knowledge and Information Systems 4*, 96–111.

Hao Xing is completing his PhD degree in the Department of Mechanical, Industrial, and Manufacturing Engineering at the University of Toledo. He received his BE and ME in material science and engineering from Beijing Polytechnic University. In 1998, he joined the Intelligent CAM Systems Laboratory, focusing research on manufacturing knowledge acquisition using a neural network/fuzzy logic based approach. He is working on acquiring drop hammer forming process knowledge, a joint project with the B.F. Goodrich Aerospace/Aerostructures group.

Samuel H. Huang is currently an Associate Professor in the Mechanical, Industrial and Nuclear Engineering Department at the University of Cincinnati. He was an Assistant Professor at the University of Toledo from January 1998 to August 2001. Before returning to academia, he was with EDS/Unigraphics and held the position of Systems Engineer, where he carried out several projects on CAM functionality enhancement for the Unigraphics CAD/CAM software packages. Dr. Huang received his BS degree in instrument engineering from Zhejiang University (People's Republic of China) in 1991 and his MS and PhD degrees in industrial engineering from Texas Tech University, Lubbock, in 1992 and 1995, respectively.

J. Shi is currently a Postdoctoral Fellow in the Mechanical, Industrial and Nuclear Engineering Department at the University of Cincinnati. He received his BS and MS degrees in mechanical engineering from Southeast University (China) in September 1995 and June 1997, respectively, and his PhD degree in industrial engineering (Web-based Design for X in Collaborative Product Development) from the University of Hong Kong in September 2001. His research interests are collaborative product development, supply chain management, and knowledge-based manufacturing.