Identification of Unconstrained Item Descriptions Using String-Match Heuristics

AVELINO J. GONZALEZ, HARLEY R. MYLER,
and ROBIN R. KLADKE
University of Central Florida

ABSTRACT: The objective of the research described is to identify unknown items by using heuristics to compare their unconstrained description with the exact labels of known items in an external database composed of generic elements found in the domain of interest. The string matching heuristics return a numerical measure of the similarity between the unknown description and the various candidate identifications in the external database. An examination of the functional constraints of the unknown item as compared to the candidate matches also contributes to the aforementioned numerical measure. The system described, called the Heuristic String Identifier (HSI), also employs auxiliary processes to optimize the search through the database and ensure that the best matches will always result. The goal of the HSI is to return an optimal set of those possible identifications, correctly ordered by string similarity and functional relatedness. The HSI attempts to identify the item within the domain data-base that exactly matches the target item. Failing this, it must determine which items in the database are most closely related descriptionally and conceptually to the unidentified item. Such a process entails discrimination of the most plausible hypotheses under the given constraints from the entire set of possibilities (i.e., the entire database). The approach can be viewed as a rudimentary analog to concept acquisition (Rendell, 1987).

1. INTRODUCTION

When examining a document, humans knowledgeable in the pertinent topic can generally identify the intended meaning of terms contained therein, even though their representation may be either non-standard, abbreviated, grossly misspelled, or even unknown. For example, an electronics engineer reviewing a particular set of drawings of an electronic system for the first time will be able to identify its components by virtue of his/her prior knowledge about such systems and their applicable components. This will be generally true even if the drawings are prepared using non-standard terminology and/or non-standard icons to represent commonly known elements. This ability is a significant one because it allows the person to infer about something unfamiliar from something familiar and thereby identify the unknown component. It would be quite advantageous if such an ability was able to be shown by a computer.

The problem addressed by the research described is as follows: given an unconstrained description of an unknown item (e.g., a component symbol in a system drawing, a name in a mailing list, and entry in an inventory, etc.), find the item in an available database of similar generic items that most closely matches it in description and function.

International Journal of Expert Systems
Pp. 337–364, Volume 4, Number 3, ISSN 0894-9077

The impetus for this work comes from investigations in the automated development of a model of a process system directly from a Computer-Aided Design (CAD) System database. The process system model is intended for use as the knowledge base in a model-based reasoning system (Gonzalez & Myler, 1991) which performs diagnostics and control on an actual process system in real time. Drawings in CAD databases are not always represented using standard terminology, and abbreviations as well as misspellings are quite common. Furthermore, the information found in the CAD database is generally incomplete as far as what is required by the model-based system knowledge base, and must be agumented by an external database. In order to access the needed attributes of a component in the drawing from this external database, however, it must first be identified as a particular item in that database.

The unidentified items will be represented as frames. This was done as a matter of convenience since the target model-based reasoning tool being used in the project makes use of frames to respresent the system model. The component frame has a number of slots which are to be filled in with the proper values in order to make it usable in the model-based reasoning system. Some of the slot values required by the model-based tool, such as LABEL, DESCRIPTION, INPUT, OUTPUT, AND UNITS, can generally be determined from the CAD drawing database. Others, however, such as OUTPUT-FUNCTION, DELAY, and TOLERANCE represent specific attributes of the components and are not available from the CAD system. They must be supplied from the main database. The HSI is invoked only after the values available from the CAD database have already been inserted in their proper slots.

For example, given an unidentified item with the label "BTFLY VALVE" and a main database containing descriptions of a variety of pumps, valves, meters, etc., the desired response would be to return a list containing such items as "BUTTERFLY VALVE" and "BALL VALVE" as potential identifications of the unknown item BTFLY VALVE. Each potential match is assigned a final confidence factor (FCF) based on the degree of similarity between corresponding descriptions and conceptual similarity based on the contents of an item slot with functional significance (e.g., the valid output units for a component, the address of a person, the category of an item in an inventory). The list of possible matches and match confidence factors is instantiated to a slot of each unidentified object. Final identification can be later made by the calling program based in whole or part on the contents of this slot.

This article describes a system called the Heuristic String Identifier (HSI), which employs novel matching techniques to address this problem. Although not a knowledge-based system in its traditional definition, the HSI system can be used to augment the reasoning capability of knowledge-based systems, especially those dealing with engineering applications.

Item identification proceeds by way of an interaction between the HSI mechanism and an external main database that contains a comprehensive set of prototypical descriptive and functional information of generic objects within the domain. The HSI attempts to match descriptive information contained in the unidentified item to generic and instance-level information contained in the main database. The HSI determines which items in the main database most closely match the items to be identified.

The major aspect of the identification mechanism is an evaluation of descriptive similarity between the items to be identified and the generic items residing in the main database.

The approach taken by the HSI to handle this comparison is highly generic. The focus of the HSI on character similarity rather than on the presence or absence of domain-specific terminology allows for a high degree of flexibility. Its major advantage lies in the fact that few constraints are placed on the input data. Inputs may be nonstandard, misspelled, or abbreviated. Such features make the system viable in a real world application.

Descriptive similarity of items is measured by the HSI in terms of *string* comparison, where *string* is defined as a sequence of characters that may or may not be separated into words (Hirschberg, 1983). Traditionally, a primary focus in string comparison has been on *distance* measures, i.e., "the minimum cost of a sequence of edit operations (change, delete, insert) that change one string into another" (Coggins, 1983). Such an approach is prevalent in the literature (Aho & Peterson, 1972; Wagner & Fischer, 1974; Okuda, Tanaka, & Kasai, 1976; Wong & Chandra, 1976; Fu & Lu, 1977; Sellers, 1980; and Kruskal, 1983).

Distance is defined, according to the Parsimony Principle (Kruskal 1983), as the smallest number of operations necessary to transform one string into another (Fu & Lu, 1977). This approach is referred to as *Levenshtein distance* if the strings are of unequal length or *Hamming* distance if lengths are equal (Okuda, Tanaka, & Kasai, 1976; Coggins, 1983). These distances may or may not have weighted operations.

One of the major criticisms of the Levenshtein distance is its relative insensitivity to various string properties. Two properties not addressed are (1) the longest common subsequence of the two strings and (2) substrings. Additionally, distance measures produce similar results for dissimilar strings. These ties force the system utilizing the measure to make arbitrary decisions (Coggins, 1983).

Coggins (1983) advocates the use of multiple, weighted measures to reduce these ties and increase the discriminatory ability and sensitivity to multiple salient features. Such is the approach taken by the HSI. Distance measures attempt to quantify string dissimilarity into a tidy single measure. In doing so, spatial information such as character order, word separation, and clustering are completely lost. To rectify this, the current research proposes a *mega-heuristic* approach that utilizes a weighted combination of heuristics capable of reflecting the subtleties evidenced in these and other attributes (Lenat & Feigenbaum, 1987).

1.1. Database Search Through Application of Bias

The HSI utilizes constraints or inductive biases to guide the search through the available hypotheses. Bias in this context is intended to mean an inclination or tendency to favor one item in the main database over another as a possible match. The set of all available hypotheses comprises the hypothesis space of the system. The hypothesis space required by the HSI is a hierarchical representation of detailed item descriptions. An example of such a hypothesis space is the main database of process system components discussed above. Although the design of the HSI's search mechanism presently requires that the data be represented hierarchically, searches though flat databases could be implemented with some modifications to the HSI code. Proximity of the unidentified item to a hypothesis (a database item) is measured by a heuristically-determined confidence factor that represents the certainty that two items are similar in terms of description and function.

Bias is essentially an appropriate constraint that guides hypothesis evaluation within a hypothesis space by other than evidential clues (Mitchell, 1983; Rendell, 1987; Rendell, Seshu, & Tcheng, 1987). It encompasses all such factors that determine the plausibility or "credibility" of a hypothesis (Utgoff, 1983; Purswani & Rendell 1987).

For bias to successfully guide the searching process through a potentially immense hypothesis space, it must be both strong and appropriate (Utgoff, 1986; Rendell, 1987; Rendell, Seshu, & Tcheng 1987). If the HSI returns a long list of possible matches that do not differ significantly in confidence, the identification process may be greatly hindered. If bias is inappropriate in addition to being weak, a large number of incorrect candidate matches may lead the calling program down the wrong path, possibly resulting in an incorrect identification. Conversely, if no possible matches are returned, the calling program will be forced to query the user. Bias within a system must be either carefully structured with respect to the domain or highly flexible. There are three main approaches to bias application: *fixed*, *parametrized*, and *dynamic*. Each is discussed with respect to its application within the HSI.

The most common approach to bias management focuses on fixed bias (Rendell, Seshu, & Tcheng, 1987). Fixed bias is explicit within the code of the system and cannot be changed without altering the program. Such an approach is highly domain specific, exhibiting brittleness when novel situations are introduced (Rendell, 1987).

Three levels of fixed bias are used within the HSI to guide and restrict search within the main database. These biases correspond to three different threshold confidence factors that must be met if search is to proceed further down a particular path.

Parametrized bias allows for user modification at run time (Rendell, 1987). It gives the system-user the opportunity to make spontaneous bias adjustment, making the system somewhat less brittle and more powerful in terms of handling unexpected, real world situations. A future version of the HSI can replace the three levels of fixed bias currently implemented with parametrized bias, increasing flexibility across domains.

The third form of bias is adjusted dynamically by the program itself based on the initial results. If the initial results obtained through the use of this bias are not acceptable, then the bias is automatically reduced to allow more acceptable results, or, conversely, tightened to achieve the same goal. Utilizing such an approach provides maximal flexibility and, depending on how generic the bias adjustment technique is, minimizes brittleness and implausible results while providing maximum power.

The bias adjustment technique utilized by the HSI is limited to shifting to weaker bias when the hypothesis space is over-restricted. During the course of automated concept acquisition, inconsistent hypotheses are identified and eliminated, resulting in a shrinking collection of available consistent hypotheses. When the intersection between consistent and describable hypotheses becomes empty (see Figure 1a), bias must be shifted in such a way that constraints on hypothesis space are loosened. The approach is similar to that of constraint suspension (Davis, 1984) utilized in model-based reasoning systems. The result of the shift, shown in Figure 1b, is the restoration of a non-empty set of hypotheses and (therefore) a larger set of describable hypotheses. Shifts are made according to intelligent, heuristic-based estimates.

The HSI shifts bias if the fixed biasing scheme fails to produce any possible matches to the unidentified goal item. The new confidence factor threshold (bias) that constrains the

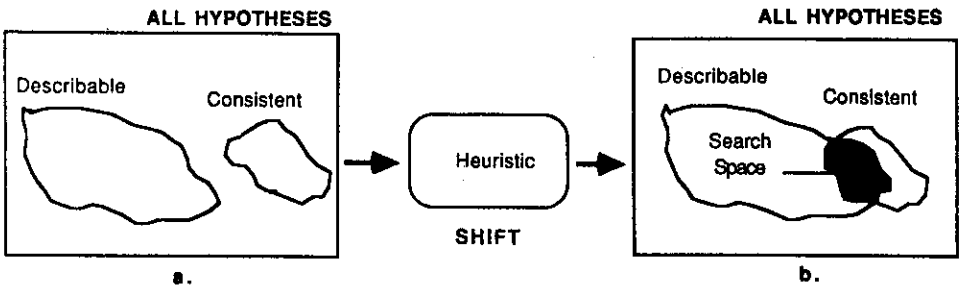

Figure 1. Representation of shifting to a weaker bias (adapted from Utgoff, 1986, p.116, Figure 5-3).

set of possible matches is dynamically variable in that it is dictated by the highest confidence match found during the identification process.

1.2. Heuristics as Tools of Bias Management

Bias and heuristics are closely linked. In his studies of heuristics, Lenat (1977, 1982) has concluded that intelligence can be seen as an efficient, heuristic-guided search through an immense space (Lenat, 1977, p. 1093). By allowing heuristics, either singly or in combination, to guide concept acquisition, search space can be reduced dramatically into "abstraction space" and the search time decreases concomitantly (Lenat, 1977, p. 1097; also see Rada, 1983). In general, heuristics serve as guides for finding plausible hypotheses in hypothesis space and ignoring (or removing) implausible ones. These ideas form the foundation on which the HSI is built.

Heuristics are used to guide behavior (be it of man or machine) in an optimal way. Sometimes such guidance can be dictated by the application of a single powerful heuristic, but many times, owing to the complexity of the problems that require solution, a compound approach is needed. Lenat refers to a combination of heuristics that is more powerful than the superposition of its members as a *mega-heuristic* (Lenat, 1982, p. 218). The purpose of creating such a mega-heuristic is to have heuristics work together synergistically with minimal conflict. Such a goal is difficult to achieve, but it is the stance of the current research that it is not only powerful but also possible.

2. THE HEURISTIC STRING IDENTIFIER (HSI)

The algorithm used in the identification process can be broken into 8 major stages. The process is carried out for each item to be identified.

1. Extraneous words within the description of the unidentified item that correspond to words within a stop list of such extraneous words are removed from the unknown string.

2. All terms in the unknown string that are defined by an auxiliary SYNONYM DATABASE as synonymous to top-level terms in the main database are replaced with those terms.
3. This modified description is compared to each item in the top level of the main database. A *string match confidence* factor (SCF) is computed for each comparison. Each comparison is initially screened through a heuristic preprocessor called the LONGEST-COMMON-SUBSEQUENCE heuristic (LCS). If this preprocessor indicates even weak probability of a possible match, seven additional heuristics are applied and a weighted combination of the results determines the value of the SCF. If match viability is not indicated, an SCF of 0% if assigned. The SCF values order the database hypotheses and search progresses according to this order in a best-first manner.
4. If the SCF for any pair exceeds the fixed generic level threshold, the branch extending from the corresponding top-level database item is searched. If the threshold is not exceeded, however, the branch is completely eliminated from further consideration.
5. If the search progresses through a branch down to the intermediate levels, a *Final Confidence Factor* (FCF)—entailing both the SCF (described in stage 3) and a functional matching—is computed and compared against a second threshold. If this intermediate level threshold is surpassed, the search continues down the branch to either another intermediate level item or to a leaf level item. If the threshold is not exceeded, the item is then saved into a list of backup matches to be used in the event that no strong matches are found.
6. Once the search progresses to the leaf level, a determination must be made as to whether a match qualifies as a strong or a weak match. Strong matches must exceed the leaf level threshold and are pushed into the list of possible matches. All other matches are considered weak and are saved into the list of backup matches.
7. After every eligible branch of the database has been traversed, the best matches (i.e., those with the highest FCFs) are instantiated to the POSSIBLE-MATCHES slot of the unidentified item. These final matches are also saved into a list of learned matches so that, if encountered again, a description will not need to be re-processed.
8. When all unidentified items have been processed, a list of those items for which no match, strong or weak, was found is returned to the calling program. At this point, the identification process is complete.

The HSI process is described in detail below with respect to an example. The example involves the identification of a component, labelled as "BTFLY VALVE," which is found in a process system drawing, given the small, hierarchical external database shown in Figure 2.

The values computed during comparison of "BTFLY VALVE" to each item in the database are shown in Table 1. The bold lines within Figure 2 represent paths that are traversed during the identification process. Dotted lines represent paths that are ignored or

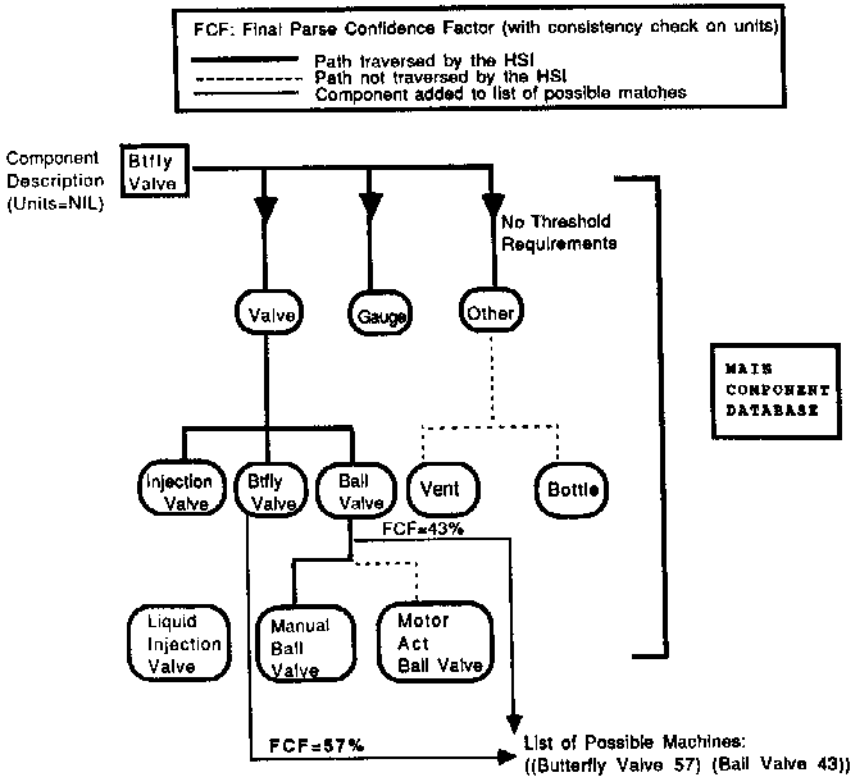

Figure 2. Search path taken by the HSI within the main component database in order to identify the component "BTFLY VALVE."

TABLE 1: Computed Heuristic Values Corresponding to Figure 2

Unidentified description	*Knowledge base description*	*LCS*	*SCF/FCF*
BTFLY VALVE	Other	NA	50%
	Valve	100%	37%*
	Gauge	40%	−11%
	Bottle	67%	7%
	Vent	50%	−1%
	Butterfly-valve	100%	57%*
	Ball-valve	67%	43%*
	Injection-valve	50%	23%
	Manual-ball-valve	70%	11%
	Motor-act-ball-valve	70%	3%
	Liquid-injection-valve	60%	−5%

*Saved as a possible match.

abandoned. The ordering of database descriptions within Table 1 represents the actual, best-first order in which they are examined.

2.1. Removal of Adjunct Strings

The first stage of the HSI process is simplification of the unidentified item's description. The goal of this simplification is to abstract the elemental portion of the description. The HSI does this through the removal of words found in a stop list created specifically for this purpose. For example, the nomenclature "This is a pump" is reduced to "pump" before the identification process begins because the prepositions "this," "is," and "a" were found in the aforementioned stop list, and thus are considered extraneous. Such an approach increases both the efficiency and the confidence of an identification. The stop list can usually be generic, but the user must ensure that the stop list does not contain any words that are considered adjuncts for most domains, yet are meaningful for the domain of interest. Removal from the unidentified string of words that are meaningful in the context of interest can compromise the identification process.

2.2. Replacement of Synonyms

The second stage of the algorithm attempts to maximize the chances of making strong string matches by standardizing terminology. This standardization is made possible through reference to the HSI's auxiliary SYNONYM DATABASE.

The SYNONYM DATABASE relates terms from the top (generic) level of the main database to synonyms and common abbreviations in the unidentified string. When a generic-level item is encountered, the description of the unidentified item is searched for the occurrence of a synonym or abbreviation to this generic-level item (description). For example, if the item "pressure gage" is encountered, the HSI will modify the description to "pressure gauge," which will agree with the description of the generic-level item shown in Figure 2. Synonym replacement operates on a single word basis only. Since the SYNONYM DATABASE is specific to the domain of the problem, replacement of ambiguous words by an incorrect synonym would not occur. For example, replacement of the word "screen" by "terminal" instead of "mesh" would not happen for a database that deals with process control devices such as filters.

Ambiguity in abbreviations, such as interpreting "ms" as either milliseconds, manuscript, or multiple sclerosis, is also not realistic because of the domain-specific nature of the SYNONYM DATABASE.

This replacement will result in a higher confidence level and will reduce search time by allowing an early exit of the identification process for that item.

2.3. Identification of Candidate Matches Through a Database Search

The task of stages 3 through 7 of the algorithm is to prune the main database hypothesis space until the best possible matches to the unidentified item are found. The use of "prune" in this context is not accidental. Each time the HSI requests an item from the

main database for examination (i.e., for comparison matching), the item is copied into a temporary database. The purpose of this transfer is to create a restricted, partial preference ordering of the hypothesis space that will facilitate subsequent search by the calling program.

This pruning process is guided by a measure of bias. Bias in this context is intended to mean an inclination or tendency to favor one item in the main database over another as a possible match. The bias is composed of two items:

1. A string-match confidence factor (SCF), computed by applying multiple heuristics to the comparison of the two descriptions.
2. A functional dependency based upon functional consistency.

Bias in the identification process is determined primarily by the structure of the main database. HSI requires the main database to be hierarchical, extending from generic types down to specific instances (see Figure 2). The description of the items at each level must be substrings of the descriptions found at deeper levels. This is due to the design of the pruning function performed by the HSI as dictated by the nature of its original application. Nevertheless, the mega-heuristics themselves can be used to assign a level of confidence to candidate matches found in a flat database by making some relatively simple modifications to the HSI code.

As the HSI traverses the hierarchy, it computes the SCF for each item at that level and orders them into descending order by SCF. This approach biases the search toward what Rendell (1987) would term the most credible hypotheses. Such a feature is especially valuable if a perfect match exists within the database. Since the search will cease when a perfect match is found, no time is wasted pursuing branches that are less promising but that occur spatially earlier in the main database. This approach imposes a total preference ordering on a dynamically-variable restricted portion of the hypothesis space. Although the database in the butterfly valve example is composed of three levels, this is not a requirement. The recursive nature of the system would treat all levels in a similar fashion.

The HSI operates according to three levels of bias controlled by a fourth preprocessing level. In order for an item to be examined at all, string similarity must be demonstrated during the preprocessor stage. The heuristic employed as preprocessor, LONGEST-COMMON-SUBSEQUENCE (LCS), is discussed below. If the constraint imposed by this heuristic is not met, a branch within the hierarchy is abandoned.

2.4. A Highly Permissive Filtering Preprocessor

The HSI uses a total of eight heuristics to determine the SCF for a match. The first, LONGEST-COMMON-SUBSEQUENCE, is used to determine the gross initial credibility of the match and is not combined with the results of the remaining heuristics to form the final SCF. The primary objective of this heuristic is to screen comparisons before the other seven remaining, time-consumptive heuristics, that combine to form the complete SCF, are computed. If the LCS value does not exceed a fixed threshold (currently set to 30%), the remaining heuristics are not applied and a final SCF of 0% is returned. The effect of this approach is twofold. First, a great deal of computation time is saved since only one of the

a. ***BTFLYVALVE*** ---> ***GAUGE***

LENGTH OF LCS = 2 LENGTH OF SHORTER STRING = 5
RATIO = 2/5 = 40%

b. ***BALLVALVE*** -----> ***BTFLYVALVE***

LENGTH OF LCS = 6* LENGTH OF SHORTER STRING = 9
RATIO 6/9 = 67%

*TRUE LCS=7 BUT INSTEAD OF MATCHING B_L_VALUE TO MATCH ALGORITHM MATCHED BAL-V---E TO B-----ALVE, MAKING THE MATCH STARTING FROM THE FIRST "L" LONGEr.

c. ***BTFLYVALVE*** -----> ***BUTTERFLYVALVE***

LENGTH OF LCS = 10 SHORTER STRING = 10
RATIO = 10/10 = 100%

Figure 3. Computation of LCS.

eight comparisons is made. The LCS serves as a strong (and it is hoped, appropriate) bias, precluding the calculation of likely dead-end comparisons. Second, this approach results in the return of a 0% SCF that will cause the cessation of search down that particular branch in the main database; this not only produces a time savings, but also has the added benefit of limiting meaningless low-CF matches.

The LCS heuristic is highly permissive, resulting in a low to moderate value only if there is little or no character correspondence between the unidentified and main database descriptions being compared. Because of this, it is appropriate as a gross screening mechanism, but is not discriminating enough to be included in the full SCF computation.

The LCS heuristic computes the longest common subsequence between two descriptions. More specifically, it calculates the ratio between the longest subsequence common to two strings and the length of the shorter string. A subsequence is defined as a sequence of characters, not necessarily consecutive, that can be obtained by deleting zero or more characters from a string (Hirschberg, 1983, p. 325). The characters must be in the same order in both strings.

Figure 3 illustrates how the LCS heuristic is computed. The unidentified string is "BTFLY VALVE" and it is compared against "GAUGE," "BALL VALVE," and "BUTTERFLY VALVE." The example shows the computation with the separators removed.

The LCS value computed using this heuristic may not always be the true longest subsequence, since the comparisons are made character by character through the shorter string, and only one matching subsequence is computed from each starting character. Figure 3(b) depicts a case where the LCS value is not the "true" LCS. Correction of this problem would require a computationally-intensive backtracking approach which contradicts the gross "weeding out" nature of this heuristic. Furthermore, the performance of this heuristic was determined to be acceptable during the testing described in section 3. Thus, the LCS heuristic was left as is.

As shown in Table 1, each comparison met the LCS requirement of 30%. Once this requirement is established, the full SCF (and, subsequently, FCF) can be computed.

2.5. String-Matching Heuristics and Computation of the Full SCF

The full SCF represents a weighted combination of seven heuristics. These are:

WILDCARD-MATCH-RATIO
CHAR-CLUSTER-BACKTRACK-RATIO
CONSEC-CHARS-BY-ORDERED-WORD-RATIO
CONSEC-CHARS-BY-UNORDERED-WORD-RATIO
ORDERED-COMMON-WORD-RATIO
UNORDERED-COMMON-WORD-RATIO
COMMON-CONSECUTIVE-WORD-RATIO

Four of these heuristics address similarities at the character level and three conduct analysis at the word level. These heuristics are weighted according to the significance of the information they provide and combine to form a mega-heuristic that addresses multiple attributes of string similarity.

The examples used throughout this section are related to those shown in Tables I and Figure 2 (above). For each of these examples, unless otherwise noted, it is assumed that the functional comparison based on units' consistency results in no change in confidence factor because the unidentified item "BTFLY VALVE" has no units associated with it. A final confidence factor (FCF) is determined for any match by a combination of three levels of bias coupled with the weighting of those heuristics performed. The current weighting scheme, determined by extensive testing, is shown in Table 2.

Because there is some evidence in the literature to support an approach that favors consonants over vowels (Yum, 1931; Alberga, 1967), vowels are removed from both unidentified and main database descriptions before the character heuristics are carried out. The result is essentially a comparison of abbreviations, an approach that has met with some success in the literature (Blair, 1960).

The character-based heuristics, WILDCARD-MATCH-RATIO, CHAR-CLUSTER-BACKTRACK-RATIO, CONSEC-CHARS-BY-ORDERED-WORD-RATIO, and CONSEC-CHARS-BY-UNORDERED-WORD-RATIO, are described next. Each, with the exception of WILDCARD-MATCH-RATIO, evaluates descriptions with vowels removed.

The WILDCARD-MATCH-RATIO heuristic compares the unidentified item description interleaved with the wildcard character "+" (e.g., "+B+T+F+L+Y+ +V+A+L+V+E+") to a main database item description (e.g., "BUTTERFLY VALVE"). Wildcards match to zero or more characters. The routine determines the number of natural

TABLE 2: Current SCF Weighing Scheme

Heuristic	*Weight*
Wildcard-Match-Ratio	0.50 x result
Char-Cluster-Backtrack-Ratio	2.00 x result
Consec-Chars-by-Ordered-Word-Ratio	1.00 x result
Consec-Chars-by-Unordered-Word-Ratio	2.00 x result
Ordered-Common-Word-Ratio	1.00 x result
Unordered-Common-Word-Ratio	1.00 x result
Common-Consecutive-Word-Ratio	4.00 x result

characters of the main database description that instantiate to wildcard characters in the unidentified item description. The resulting match confidence factor (CF) is inversely proportional to the number of instantiations; i.e., the greater the number of natural characters that cannot match to natural characters and do match to wildcards, the smaller the value returned. A mismatch between natural characters in the two strings results in a penalty.

The purpose of the WILDCARD-MATCH-RATIO heuristic is to address the case of abstract abbreviations such as "BTFLY VALVE." Since it is limited to this one case, the heuristic does not carry great weight in the final SCF computation (see Table 2).

The focus of the CHAR-CLUSTER-BACKTRACK-RATIO heuristic is to reward abbreviations or misspellings in which matching letters are clustered in groups of two or more. For example, when compared to "BTFLY," "BUTTERFLY" possesses three characters contained in a cluster ("FLY"); "BALL," however, contains no clustered characters. The routine is order-insensitive, allowing backtracking, but guards against rematch of characters contained in previously matched clusters. This heuristic is generally useful, handling a variety of cases, and has, therefore, been given significant weight.

Figures 4 and 5 describe each of the heuristics and how they compute the SCF for the example problem. Figure 6 shows how each of the seven heuristics comprising the mega-heuristic contributes to the total SCF. The examples shown are based on a match between the unidentified description "BTFLY VALVE" and the main database description "BUTTERFLY VALVE." Figure 4a illustrates the evaluation of WILDCARD-MATCH-RATIO for this example. The figure shows the wildcard instantiation process, the penalty computation, the calculation of the heuristic's CF, and the final weighted contribution of the heuristic to the final SCF. Similar information is given for each of the remaining heuristics.

Figure 4b illustrates the process of dividing a description into clusters and computing the heuristic's CF contribution. Note that the descriptions compared have had vowels removed.

The CONSEC-CHARS-BY-ORDERED-WORD-RATIO heuristic determines how many characters match consecutively between words in the two descriptions. The words must be in the same order. Words that mismatch starting at the first character are penalized by an amount proportional to the length of the mismatched word. This heuristic favors word-order-sensitive domains and penalizes extraneous words yet does not penalize truncated descriptions. Bias is toward matching characters at the beginning of the word, an established approach in string analysis (see Yum, 1931; Blair, 1960). As an example, the abbreviation "PRESS GAUGE" would receive high match confidence when compared against "PRESSURE GAUGE"; "ANALOG PRESS GAUGE," however, would be allotted a penalty for the presence of an extraneous word. "GAUGE, PRESS" would also be penalized since word order has been violated.

Figure 4c shows the computation of the value for CONSEC-CHARS-BY-ORDERED-WORD-RATIO. Because the example represents a good match with no extraneous words, no penalties are allotted. This heuristic and the next (see Figure 4d) will yield the same results since both unidentified and main database descriptions are ordered in the same manner. Both heuristics compare descriptions that have been stripped of vowels.

The unordered approach to leading consecutive characters (CONSEC-CHARS-BY-UNORDERED-WORD-RATIO) is identical to the ordered approach except that words may be in any order. Such an approach favors word-order-invariant domains that do not penalize jumbled describers or the inclusion of parenthetical information at the end of the

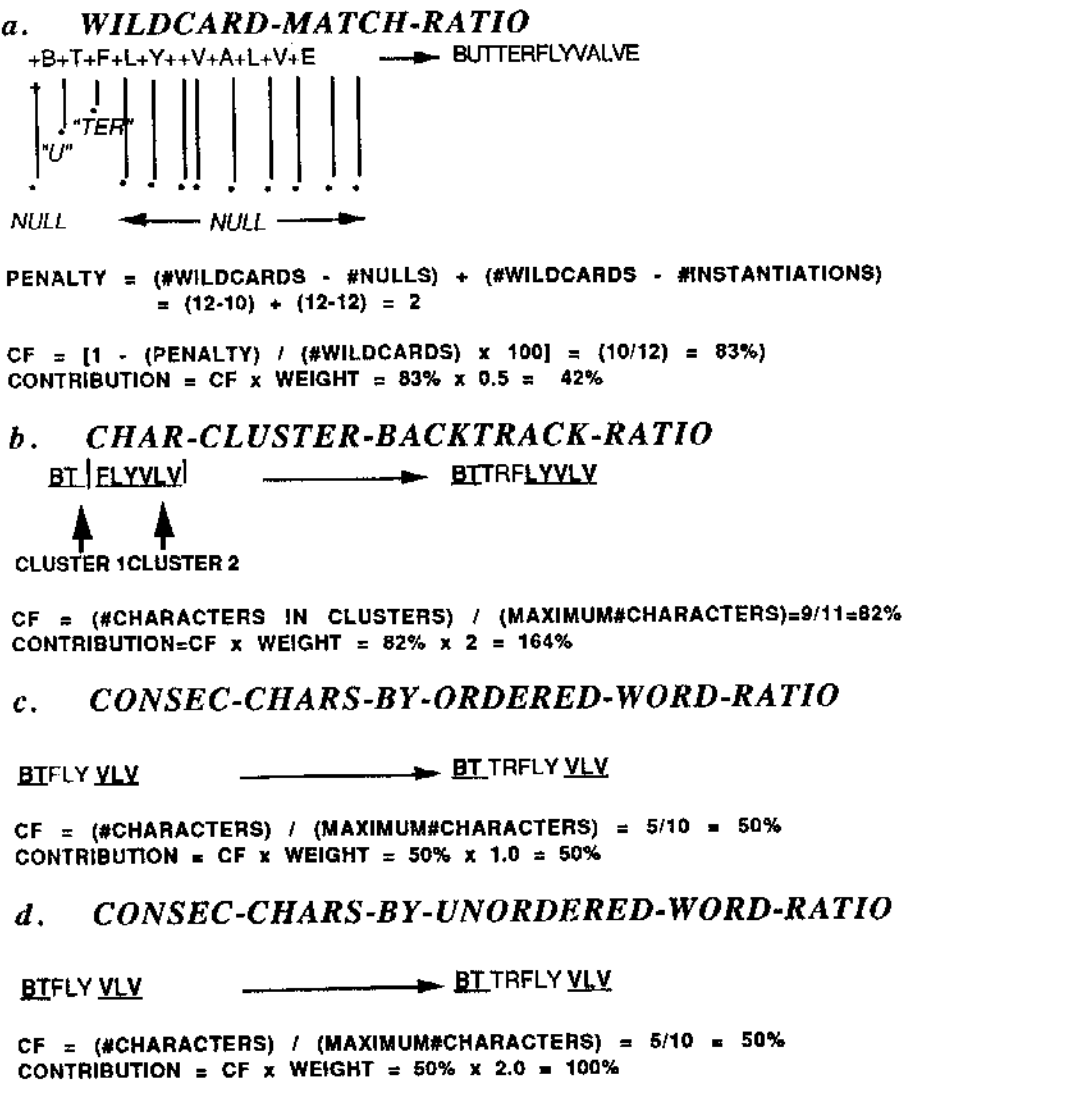

Figure 4. Individual computations of character-based heuristics.

nomenclature. With respect to this heuristic, "PRESS GAUGE" and "GAUGE, PRESS" would achieve identical results. The flexibility of this heuristic accounts for its greater relative weight.

The procedure for computing the contribution of CONSEC-CHARS-BY-UNORDERED-WORD-RATIO is shown in Figure 4d. As previously noted, the results are identical to those of the ordered case.

The remaining heuristics, ORDERED-COMMON-WORD-RATIO, UNORDERED-COMMON-WORD-RATIO, and COMMON-CONSECUTIVE-WORD-RATIO, assign confidence based on the presence of common words between two descriptions. For a word

a. ORDERED-COMMON-WORD-RATIO

BTFLY VALVE ------> BUTTERFLY VALVE

CF = (#COMMON WORDS SAME ORDER)/(MAXIMUM # WORDS) = 1/2 = 50%

CONTRIBUTION = CF x WEIGHT = 50% x 1 = 50%

b. UNORDERED-COMMON-WORD-RATIO

BTFLY VALVE ------> BUTTERFLY VALVE

CF = (#COMMON WORDS ANY ORDER)/(MAXIMUM # WORDS) = 1/2 = 50%

CONTRIBUTION = CF x WEIGHT = 50% x 1 = 50%

c. COMMON-CONSECUTIVE-WORD-RATIO

BTFLY VALVE ------> BUTTERFLY VALVE

$$CF = \frac{\text{(MAXIMUM \# COMMON CONSECUTIVE WORDS)}}{\text{(MAXIMUM \# WORDS)}} = 1/2 = 50\%$$

CONTRIBUTION = CF x WEIGHT = 50% x 4 = 200%

Figure 5. Individual computations of word-based heuristics.

to match, both spelling and form must be identical. Procedures for computing individual values of word-based heuristics are shown in Figure 5.

According to the ORDERED-COMMON-WORD-RATIO heuristic, descriptions with perfectly matching words are awarded CFs that are proportional to the number of words matched. This heuristic requires the matched words to be in the same order in both descriptions.

An example computation of the CF value for ORDERED-COMMON-WORD-RATIO is shown in Figure 5a. As in the character-based heuristics above, the common ordering between unidentified and main database descriptions in this example results in similar CF values that reinforce each other.

The UNORDERED-COMMON-WORD-RATIO heuristic also focuses on perfect word matches, but word order is irrelevant. The computation of this heuristic's CF is shown in Figure 5b.

If perfect match words are present, the clustering of those words serves as a good indication of item match strength. The COMMON-CONSECUTIVE-WORD-RATIO heuristic focuses on this. For example, when compared to the unidentified description "PRESSURE RELIEF VALVE," the main database item "RELIEF VALVE" will receive greater confidence than "PRESSURE VALVE" because of the proximity of matching words. The value of this heuristic in discriminating weak matches from strong dictates its large weighting factor.

The procedure for calculating the COMMON-CONSECUTIVE-WORD-RATIO is shown in Figure 5c.

An example of the final computation of the SCF is shown in Figure 6. The contribution of each heuristic is summed and the result is divided by the totalled weights. The result in this particular example is 57%, which agrees with the SCF value for the match of "BTFLY VALVE" to "BUTTERFLY VALVE" shown in Figure 2 and Table 1.

Specific string characteristics pertinent to the general problem domain have been addressed by various facets of the current HSI. These attributes are:

- clustering of words
- abstract abbreviation of terms
- misspellings
- use of nonstandard terminology
- preservation of word order
- inclusion of extraneous descriptive information

The common clustering of words between two descriptions is a strong indication of similarity. Three HSI heuristics directly address this feature. COMMON-CONSECUTIVE-WORD-RATIO affects the FCF by awarding a result proportional to the length of the longest sequence of words common to both descriptions. CHAR-CLUSTER-BACKTRACK-RATIO, though intended for analysis at the character level, rewards any character sequence (preserving blanks) that is two or more characters in length. The LCS heuristic, though not affecting the final match confidence, determines whether or not heuristics will be computed at all, based on the longest common string of characters (that may or may not cross word boundaries).

An abstract abbreviation is essentially a subsequence. The LCS of an abstract abbreviation will be 100% if no extraneous words or characters are present. The FCF of an abstract abbreviation is affected primarily by the WILDCARD-MATCH-RATIO and the CHAR-CLUSTER-BACKTRACK-RATIO. The former is intended exclusively to address this case, while the latter addresses the more general situation of character clusters common to the two descriptions, a frequent auxiliary feature of abstract abbreviation.

All heuristics except ORDERED-COMMON-WORD-RATIO, UNORDERED-COMMON-WORD-RATIO, and COMMON-CONSECUTIVE-WORD-RATIO compensate for misspelling. The incorporation of vowel removal has even addressed the particularly problematic issue of a misspelling that occurs early in the description.

The HSI handles the use of nonstandard terminology by cross-referencing the description of the unidentified item with the SYNONYM DATABASE. Synonyms for generic level items are replaced with the main database terminology at the beginning of the identification cycle. Common abbreviations (e.g., "ASSY" for "ASSEMBLY") are also replaced. Such a feature, along with the relatively unconstrained description required by HSI avoids the "rigid grammatical straitjacket" evidenced by many knowledge-based systems (Lenat and Feigenbaum 1987, 1181).

- WILDCARD-MATCH-RATIO	42% (0.5)
- CHAR-CLUSTER-BACKTRACK-RATIO	164% (2.0)
- CONSEC-CHARS-BY-ORDERED-WORD-RATIO	50% (1.0)
- CONSEC-CHARS-BY-UNORDERED-WORD-RATIO	100% (2.0)
- ORDERED-COMMON-WORD-RATIO	50% (1.0)
- UNORDERED-COMMON-WORD-RATIO	50% (1.0)
- COMMON-CONSECUTIVE-WORD-RATIO	200% (4.0)

Figure 6. Final SCF computation from individual heuristics.

All heuristics, except those specifically labeled otherwise, require preservation of word order. Such a bias permits the appropriate handling of cases in which word order determines functionality (e.g., "2-WAY 3-POS VALVE" versus "3-WAY 2-POS VALVE"). The word-order-insensitive heuristics are CHAR-CLUSTER-BACKTRACK-RATIO, CONSEC-CHARS-BY-UNORDERED-WORD-RATIO, and UNORDERED-COMMON-WORD-RATIO.

Specific detail is necessary for the correct identification of a device. This detail is misleading, however, when the appropriate leaf item does not exist and the best match in the main database is a more abstract parent item. It is important that the HSI recognize this situation and not return inappropriate, too-specific instances of the desired parent. For example, given the case of a sought item "REMOTE PRESSURE TRANSDUCER," "PRESSURE-TRANSDUCER" is a far more valid match than "ANALOG-PRESSURE-TRANSDUCER." Similarly, given "PRESSURE TRANSDUCER," a perfect match is far superior to the more specific instance "ANALOG-PRESSURE-TRANSDUCER." The former situation is handled by exacting a penalty for extraneous words/characters in the main database description. The heuristics that carry penalties are CONSEC-CHARS-BY-ORDERED-WORD-RATIO, CONSEC-CHARS-BY-UNORDERED-WORD-RATIO, and WILDCARD-MATCH-RATIO. The latter situation is counteracted by immediate exit from the HSI when a perfect match is encountered.

The existence of numerous unnecessary words in an unidentified description will dilute the strength of an otherwise close match. If this is to be avoided, adjunct words must be excised from the description prior to beginning the process of identification. Words corresponding to those contained in the ADJUNCT database are removed from the unidentified description prior to identification. This approach is expected to increase both match strengths and efficiency. The strength of the HSI lies in its ability to identify highly unconstrained descriptive information in a generic and robust manner. By addressing a variety of string attributes, the likelihood of robustness is increased; however, to be effective, the combination must be more than simply a loose association of unrelated features. The attributes addressed must represent a coalescence of features that can serve as a comprehensive model for the description's identity. It is felt that the features addressed by the HSI fulfill this requirement.

The application of a weighted combination of these heuristics has produced good preliminary identification of items in two real-world process control systems. Section 4 discusses the verification issue.

2.6. The Three Levels of Database Search

The matching process can be broken into three levels: *generic*, *intermediate* (which can represent more than one level), and *leaf*. Items at the generic level have no parents. Items at the intermediate levels have both parents and children. Items at the leaf level have no children. Each of these levels has associated with it a minimum bias strength that determines whether the main database item located there is worth pursuing further. Bias strength corresponds to the current match confidence level; search is biased toward a particular branch in the main database dependent on whether or not the current SCF/FCF meets the minimum strength required by the level.

Actual values that would be computed in the current version of HSI are shown in Table 1 and Figure 2. A full description of the identification process with respect to this example follows. For reasons of simplicity, it is assumed that the unidentified item, "BTFLY VALVE," used in this example, possesses no units. Since the functional comparison (here based on units consistency) is the only factor differentiating the SCF from the FCF, the SCF is equal to the FCF for these examples. See section 2.8 for a discussion of FCF determination when functional comparison is possible.

2.6.1. Searching at the Generic Level

At the first level of the main database (containing "VALVE," "GAUGE," and "OTHER" in Figure 2), only a very low threshold SCF (1%), in addition to the required LCS, must be met in order for the search to continue to the next level of items. This permissiveness is based on the fact that the top level of the database searched is expected to contain generic item specifications that describe a general class of items rather than actual devices.

This top level serves as a gateway to the actual items and is the only means of access to instance level devices. Therefore, permissiveness is necessary so as to allow for access to items that lie deep in the main database and are therefore much more specifically described than the gateway item (e.g., "NC-2-WAY-POS-SOL-VALVE" versus "VALVE"). Generic items that fail to meet the generic-level threshold are completely eliminated from further consideration. They are not added to the list of backup matches used in the case that no "good" matches are found (see section 2.7).

With respect to Table 1, only two of the three generic-level items are traversed to the intermediate level. "OTHER" is traversed without condition, but is assigned a SCF of 50% so as to avoid traversing the entire main database exhaustively in search of a unique or difficult-to-categorize item. In this example, "OTHER" is traversed first and VALVE second (FCF = 37%). Traversal is represented by bold lines in Figure 2. "GAUGE" fails to meet the 1% threshold (though the LCS is adequately high at 40%) and is therefore eliminated from further consideration (note the broken line in Figure 2). "VALVE," though not meeting the criterion of a "good" final match (FCF = 40%), is saved as a backup match in case no "good" final matches are found.

2.6.2. Searching at the Intermediate Levels

The second level represents the first level at which actual devices may appear. This level, referred to as the intermediate level, may in some cases actually represent more than one level, hierarchically ordered, and it imposes a more stringent bias on matches (10%). The items in Figure 2 that fall into the intermediate level are "INJECTION-VALVE" and "BALL-VALVE."

Bias strength at this level is controlled not by the simple string-match confidence factor but by the more comprehensive final confidence factor (FCF) that incorporates a units consistency check. This tightening of constraints is meant to encourage candidate elimination in cases of strong descriptive disparity without precluding any item branch with any likelihood of success. Matches that fail to meet the intermediate-level threshold but that do produce a positive FCF are saved into the list of backup matches.

The next items pursued in this example are the children of "OTHER"; however, these items do not meet the criterion of the intermediate level and are discussed in the next sec-

tion. The only items that do meet the criterion are the "INJECTION-VALVE" (FCF = 23%) and "BALL-VALVE" (FCF = 43%). Both items exceed the 10% threshold and are traversed to the next level which, hypothetically, may be either another intermediate level or a leaf level. In this case, the next level for each is the leaf level. In addition, "BALL-VALVE" is saved as possible match since its FCF exceeds not only the intermediate threshold of 10%, but also the leaf threshold of 40%. "INJECTION-VALVE" is saved into the list of backup matches.

2.6.3. Searching at the Leaf Level

The final level of bias occurs at the leaf of a main database branch. Figure 2 shows seven items at the leaf level: "LIQUID-INJECTION-VALVE," "BUTTERFLY-VALVE," "MANUAL-BALL-VALVE," "MOTOR-ACT-BALL-VALVE," "GAUGE," "VENT," and "BOTTLE." At this level, the decision of the HSI is no longer whether or not to proceed down a branch, but rather becomes whether or not an item should be included in the list of top candidates for possible matches. If a match does not meet the leaf threshold requirement, it is saved into the list of backup matches. A match exceeding the leaf threshold is saved into a list of primary matches and the list of backup matches becomes obsolete.

The first items in Figure 2 traversed to the leaf are the children of "OTHER": "VENT" (FCF = –1%) and "BOTTLE" (FCF = 7%). Neither of these items meet the 40% leaf threshold and, therefore, they are not marked as possible final matches to "BTFLY VALVE." "BOTTLE," however, is saved as a backup match since its FCF is positive. The next item examined is "LIQUID-INJECTION-VALVE" (FCF = –5%), the only child item of "INJECTION-VALVE." It too is eliminated as a possible match. "BUTTERFLY-VALVE" (FCF = 57%) meets the requirement of a leaf item and is saved as a match. None of the children of "BALL-VALVE" (which was previously saved as a possible match), meet the leaf threshold. Both "MANUAL-BALL-VALVE" (FCF = 11%) and "MOTOR-ACT-BALL-VALVE" (FCF = 3%), however, are saved as backup matches. It should be noted that if "MANUAL-BALL-VALVE" had possessed children (i.e., was not a leaf node), its FCF of 11% would have been sufficient to cause traversal to the next level (note the dark line leading from "BALL-VALVE" to "MANUAL-BALL-VALVE" in Figure 2).

Because "good" matches exceeding the leaf threshold were encountered during the matching process, the list of backup matches is discarded (avoiding the sorting and pruning of this potentially long list). The list of backups at this point contains (((VALVE) 37 37) ((OTHER BOTTLE) 7 7) ((VALVE INJECTION-VALVE) 23 23) ((VALVE MANUAL-BALL-VALVE) 11 11) ((VALVE MOTOR-ACT-BALL-VALVE) 3 3)). Each element of this list contains the path taken to the item, the current FCF, and the current SCF. In this example, SCF will equal the FCF since sufficient units information is not available to differentiate the two. The final list of matches instantiated to the POSSIBLE-MATCHES slot of "BTFLY VALVE" is ((BUTTERFLY-VALVE 57) (BALL-VALVE 43)).

This example represents a case in which strong matches are encountered during the identification process. If such is not the case, the HSI must have available to it an alternate method capable of determining the best possible matches based on information gathered during the matching process. The method used to determine these matches relies on the application of dynamic bias. This is described in the next section.

2.7. Handling the No-Match-Found Situation

When a strong fixed bias fails to produce any "good" matches, the HSI is capable of dynamically altering its bias for determining a possible match by imposing a weaker bias. The weaker bias is based on the most promising information gathered. Such an approach is supported by the work of Utgoff (1986), Rendell (1987), and Purswani and Rendell (1987).

As previously explained, three levels of increasing bias exist within the matching process. Any comparison that fails to meet the threshold requirements of the latter two levels but that does result in a positive SCF will be pushed into the list of backup matches (*akg-next-best-matches-list*). If no "good" matches are found for a particular description anywhere in the main database tree (i.e., no comparison meets the stringent requirement of the third level of bias), the top n% or top m matches—whichever is larger—are returned as the possible matches for the match (currently, $n = 5$ and $m = 4$).

The extent of bias attenuation is based on the strength of those matches computed, so it is dynamically variable with respect to knowledge gleaned during the matching process. In the current version of the HSI, no further processing takes place on this list of next-best matches; i.e., these items do not serve as new starting points in the main database to be followed in search of closer approximations. It is held that doing so would not result in great gains and would hinder HSI efficiency to an unacceptable degree. If the calling program later determines that following one of these tenuously identified items may be fruitful, it has the capability to do so; it will, however, do so only on an as-needed basis, a much more efficient alternative to the indiscriminate approach that would otherwise be taken by the HSI.

2.8. Determining Functional Relatedness

Functional dependence for the example shown in Figure 2 relies on the data contained in the (output) UNITS slot of both source and target items. Beginning at the intermediate level of matching (i.e., the 10% level), biasing is dependent upon not only the result of string-matching heuristics but also upon a consistency check made between UNITS. This comparison, however, is not based on exact match. The units are related by their functional similarity. Such a strategy can be seen as a concrete approach to concept formation. The goal of such a classification is to identify the conceptualizations that underlie the items in question (Schank, 1973). The establishment of functional dependence between source and target results in the creation of a final confidence factor (FCF) that shall be the value instantiated to the POSSIBLE-MATCHES slot of the sought item object.

The UNITS DATABASE is structured into conceptual clusters (Rendell, 1985). According to this scheme, specific units, such as "inches," "gpm," and "watts," are classified with other units corresponding to their particular class or classes. Units may occupy more than one cluster. The clustering is done according to a loose functional relatedness that provides gross distinction between units, and therefore item, types.

In the current UNITS DATABASE, conceptual types have been defined for mass, electrical current, flow, length, pressure, thermal, and volume. Other categories may be added as the database becomes further populated.

To compute the FCF, the units of the sought item are compared to the units of the main database item. If an exact match is found, the SCF is increased by a percentage of the current SCF. This amount of increase is relative instead of fixed so that, in the case of low confidence matches, undue weight is not placed on the results of the consistency check over the much more extensive string matching. If an exact match between the units/rating of the items under comparison cannot be made, the UNITS DATABASE is accessed. The conceptual category of each units' value is sought. If the units of the unidentified item agree conceptually with either the units or rating of the main database item, the FCF is set to (SCF + reward * SCF). If all three values are available (i.e., units for the unidentified item, units for the main database item, and rating for the main database item), and the values disagree, the FCF is set to (SCF – penalty * SCF). If the values disagree and one of the three slots is not available (i.e., NIL), the SCF is returned unchanged. The check for functional dependency is not meant to overwhelm the results of string matching; rather, it is meant to augment strongly positive evidence and further impair strongly negative evidence. The utility of the measure lies in its ability to strengthen differences between strong and weak matches.

2.9. Learning from Previous Matches

Once the set of possible matches is determined for a sought item, the list is saved with corresponding SCFs to a global list (*akg-learned-matches-list*). This corresponds to stage 7 of the process. Each time a new sought item nomenclature is processed, this list is first examined. If the item was encountered previously in the match, the string-matching portion of the match is not repeated. Only the units consistency check is carried out. The purpose of the check is to preclude the possibility of propagating the penalty of a units error/omission to remaining instances that may not contain the error/omission. The list is re-initialized at the beginning of each call to the HSI.

2.10. Completion of the HSI Process

Stage 8 completes the HSI process. The purpose of this stage is to notify the calling system of those items for which no possible matches could be found. This main effect, however, is relatively minor when compared to the three major side effects of the HSI process summarized here:

1. Creation of a temporary database that contains the restricted portion of the main database necessary and sufficient to identify all items.
2. Instantiation of the best matches found to the POSSIBLE-MATCHES slots of the unidentified objects.
3. Creation of a list of "learned" items, their matches, and SCF values (destroyed upon return to the calling program).

3. SYSTEM IMPLEMENTATION AND TESTING

Before a description of how the system was tested, it is important to briefly describe the context in which the HSI was employed and its basic reason for development. This is best done by first describing its implementation.

3.1. System Implementation

The Heuristic String Identifier was developed as part of a research project involving the automatic extraction of information about engineering systems from their CAD representation. Known as Automated Knowledge Generation (AKG), the main objective of the research was to use this information for automatically generating a knowledge base for a model-based diagnostic system. This knowledge base stores descriptive knowledge about the system (such as the inter-connectivity of the components) and about its individual components, (like their transfer functions, their delay in operation and their tolerances as well as others). But the individual component information is typically not found in the CAD description of the subject system. This missing knowledge can be found in large generic component databases such as the one discussed in previous sections of this paper, but the components in the system must be correctly identified in order to access the proper entry in the database. This identification is the task of the HSI.

The HSI provides the AKG system with an ordered list of candidate matches for each component found in the CAD description of the system. This list is ranked according to the Final Confidence Factor (FCF) in decreasing order. The AKG system then applies constraint satisfaction techniques to the candidate matches in order to select a unique label for the unidentified item within the context of the specific system being processed. For a full description of the overall AKG system, see Gonzalez et al. (1989, 1991), and Myler et al. (1989).

The HSI system was implemented on a Symbolics 3600 series AI workstation using Symbolics Common Lisp. Extensive use was made of the FLAVORS object-oriented facilities of the Symbolics workstation. The Lisp source code to the HSI system is available from the authors for research purposes, subject to NASA concurrence. It is described by the flow chart shown on Figure 7.

3.2. System Testing

The HSI was tested exhaustively by itself and as part of the AKG verification effort. Four actual industrial process system descriptions and a set of problematic test cases were used to test the robustness, validity and efficiency of the HSI. The testbed process systems used were:

- the Environmental Control System of the Orbiter Maintenance and Refurbishment Facility (ECS-OMRF) at NASA's Kennedy Space Center,
- the Purge-Demo (PD), a simple testbed process system developed by NASA for use in model-based reasoning development,

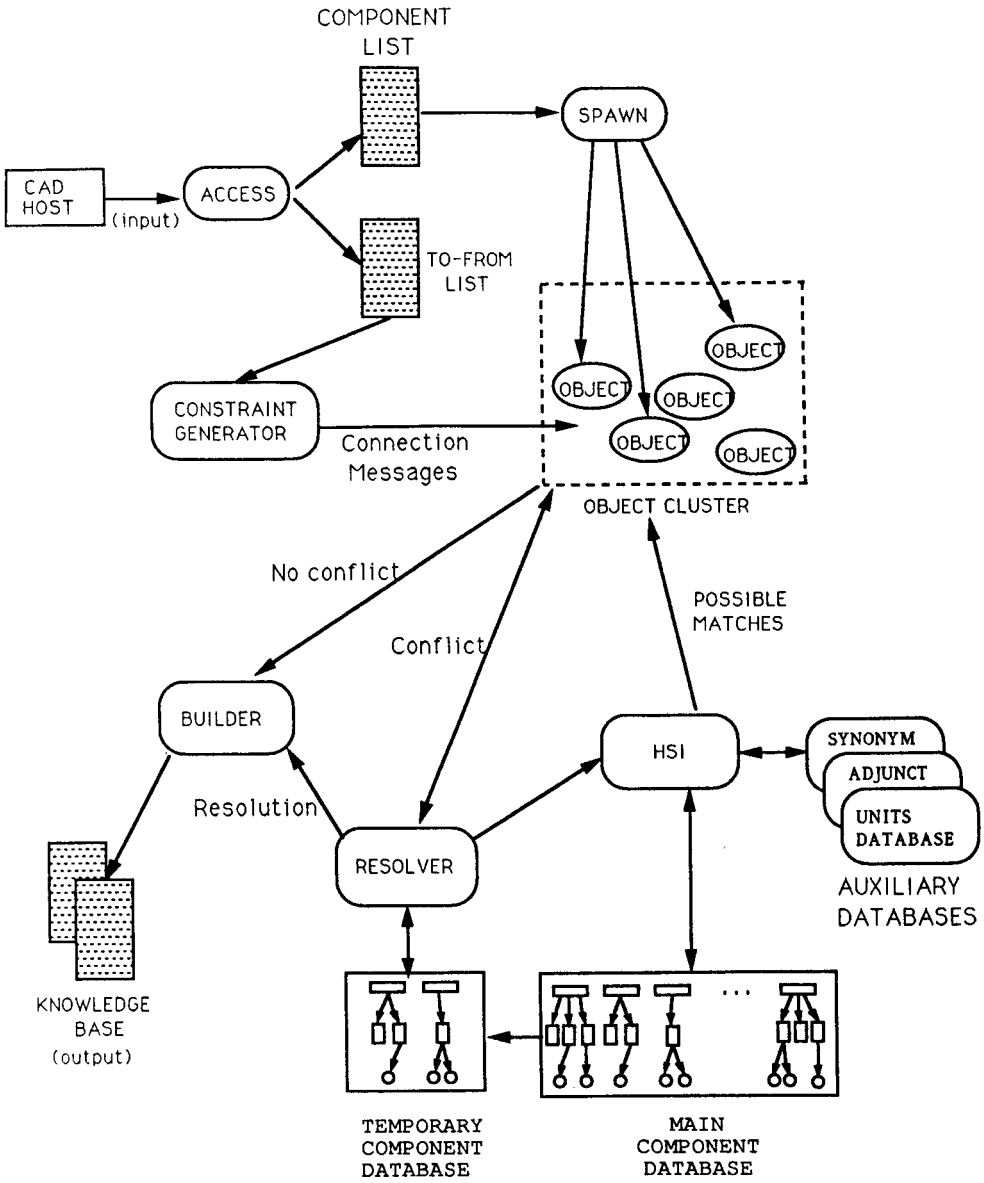

Figure 7. Flowchart description of the Automated Knowledge Generation system.

- the Water Tanking system (WT), a more complex testbed process system used for process control development, and
- the Stator Cooling Water system for a large electrical generator (SCW), courtesy of Westinghouse Electric Corp.

The ECS-OMRF was obtained in electronic medium (i.e., CAD system output), as was the PD. The other systems, the Water Tanking and the Stator Cooling system, although resident in CAD, were only received in hardcopy form.

The component descriptions of the two non-CAD systems described above were placed into temporary databases. These came directly from component lists or drawings of the systems. Misspellings, abbreviations, verbosity, and non-standard terminology were retained. In addition, no extraordinary effort was exerted to make certain that the representation reflected by the temporary database was without error. Thus, the temporary database of unknown system component descriptions was riddled with minor inconsistencies in flow and description, as would be expected in normal industrial practice.

The HSI was thus subjected to a test consisting of a total of 467 different unknown component cases. Of these, 138 cases were part of the ECS-OMRF, 46 were from the PD, 71 from the WT and 109 were part of the SCW system. The ECS-OMRF components were, for the most part, represented within the generic component database, since the goal was to test the entire AKG system. Thus, acceptable matches could be found for the majority of these components during the testing. The same was true of the SCW system. This was not true, however, for the other two systems. Additionally, SCW contained longer, non-standard, more complex and possibly more realistic descriptions of components. Thus, working with the SCW system specifically tested the ability of the HSI to handle non-standard terminology and the no-match situation, which provided a good test of HSI brittleness.

The remaining 103 test cases were designed to verify situations that were unusual and problematic, such as:

- cases in which preservation of word order was significant.
- cases in which word order was irrelevant.
- gross misspellings.
- abstract abbreviations that differed from those in the main generic component database.
- cases that tested the more-general-than relationship between system components and the main generic component database.
- cases that evidenced both conceptual similarity and dissimilarity in units.
- verbose descriptions that tested the HSI's pruning abilities.

These last cases were performed as part of the debugging of the HSI, and henceforth, the results were not explicitly documented. Nevertheless, improvements made to the system arose from executing these tests.

The SCW cases were tested directly with the HSI (rather than through the AKG system). The tests used a main database that was specifically populated with industry-standard descriptions of all the components included in the system. The descriptions in the main database, however, did not always agree with the description of the corresponding components as described in the SCW system drawing (i.e., in the temporary database). In this test, all 109 items in the drawing were properly identified by the HSI. Proper identification means that the actual component in the main database corresponding to the unidentified item in the drawing was part of the list of likely matches instantiated to the POSSIBLE-MATCHES slot of the unidentified item. The correct identification was the one with the highest FCF.

TABLE 3: HSI Testing Summary

Testbed name	*No. of Elements*	*Successfully matched*	*Errors*	*Inconclusive*
ECS-OMRF	138	116	12	10
PD	46	40	0	6
WT	71	50	8	13

The rest of the tests were performed in context using the AKG system. Of the 138 components supplied to it from the ECS-OMRF system, 114 were successfully matched. This was evident because correct identification of the item was achieved by the calling program, which uses additional filtering techniques to further discriminate from the list of possible matches created by the HSI. However, it can only find the unique correct match if the HSI includes it in the list of possible matches. Therefore, correct identification of these elements implies that the proper choice was present. Of the remaining 24, 2 were sent back as no-match-found. Upon inspection of the unidentified item, it was confirmed that no corresponding generic components existed in the main database which corresponded to the unidentified items. Therefore, these were also considered to be correctly processed.

Twelve (12) additional items were not able to be uniquely identified by the AKG system, even though a list of possible matches was supplied by the HSI. This does not mean that they were not included in the list, but rather, that not enough information was known about them (by the main database) to be uniquely identified. Thus, these cases are considered inconclusive. The remaining 10 cases represent confirmed situations where the HSI did not include the correct interpretation in the lists, even though it was present in the database.

Similarly analyzing the results of the PD tests, of its 46 items, 33 were of the first category, (that is, correct matches were found by the calling program), while 7 were also properly treated by returning a no-match-found for items that were not present in the database. The remaining six were inconclusive, as the HSI returned a list of possible matches, but the calling function was not able to uniquely identify the unknown component from this list. There were no confirmed cases where the HSI incorrectly omitted the inclusion of the proper match when it existed in the main database.

Lastly, when testing the 71 components in the WT system, 33 were found to be correctly identified. Of the remaining 38, 17 were properly labeled as no-match-found, while the HSI missed the proper match for 8 items. The last 13 were inconclusive. This test indicates an inadequately populated main database.

In summary, Table 3 describes the success rate of the HSI, measured indirectly through the AKG system, when applied to real-world problems.

4. CONCLUSIONS

Purswani and Rendell (1987, p. 352) have identified three limitations associated with knowledge-based systems: brittleness, lack of power, and implausibility of results. We describe each of these here with the approach taken by the HSI to compensate for or avoid them.

Brittleness can be defined as a lack of generality and usefulness across domains. The HSI has been specifically designed to avoid this. The identification process relies primarily on character and word matching to determine the utility of a particular hypothesis item. The only knowledge essential to the HSI is contained in the external database of generic information. The auxiliary databases play a secondary and non-vital role. The modular nature and simple structure of the these databases allows for creation, modification, and insertion of new databases as needed. The HSI is felt to address brittleness robustly.

Lack of power connotes the inability of a system to handle real world problems in a reasonable manner. Though the computational resources required will be significant with respect to the eventual size of the required external database, the HSI has always been intended to handle realistic problems in a generic manner across multiple domains. From the beginning of development, the HSI has been tested against a real world system, the Environmental Control System located in the Orbiter Maintenance and Refurbishment Facility at the Kennedy Space Center. Preliminary test results, even given the limited size of the currently utilized main database, are excellent. It is projected that, given a sufficiently robust main database for the system under test, the HSI will provide good matches or match leads for the great majority of items within the Environmental Control System.

Implausibility of results refers to the tendency of machine learning to result in answers that, though consistent and complete with respect to the concrete evidence presented, are unreasonable. Such a limitation arises when (1) facts of the real world are not taken into account, (2) no attempt is made to identify the principles underlying a concept, and (3) credibility of a concept is defined in terms of parsimony instead of real world experience. The mechanisms for applying fixed and dynamic inductive bias within the HSI address this problem of implausibility. The searching function of the Identifier is guided by levels of bias that focus the search down conceptual branches of most promise. This biasing also has the effect of eliminating from consideration those items that are biased against proper identification. This candidate elimination limits implausible matches and increases the match credibility of surviving items.

Watanabe (1985, p. 116) has characterized learning as an "entropy-decreasing process." Such a description seems particularly apt when applied to the HSI's approach to item identification. Given an unconstrained world of knowledge, the Identifier is able to pare the options down to a manageable subset that correctly and completely characterizes both the explicit description of the sought item and the implicit functionality of the system under analysis.

The salient contributions of the work described here, therefore, are twofold:

1. A means of measuring the similarity between an unconstrained string that serves as a label for some unidentified real-world object or concept, and generic descriptions contained in a database of objects or concepts within the same domain. This is accomplished through the application of seven string-matching heuristics, whose results are combined using different weights, into a single string (match) confidence factor.
2. A means of searching through the database of generic components for candidate matches using the bias described above to guide the search.

A third, less significant contribution of this work is the use of the functional dependency of the unidentified item described by the unconstrained string, and of the possible matches selected through the SCF to determine their mutual consistency. An inconsistent match will become less desirable as a search path to investigate than a consistent one. The final confidence factor (FCF) is computed from the SCF as well as from the consistency factor.

It must be noted that the string-matching heuristics described in this paper are by themselves capable of being used with strings of characters, regardless of the language or numbering system (i.e., part numbers) that they represent. However, the synonym replacement, the removal of adjunct strings, the database search, and the functional dependency features of the HSI are based on a specific application domain and thus are not generic in nature.

Acknowledgements: The Heuristic String Identifier detailed here is part of the Automated Knowledge Generation (AKG) research project carried out at the University of Central Florida in Orlando, Florida. The project was supported by NASA, Kennedy Space Center under contract NAG-10-0043.

REFERENCES

Aho, A.V., & Peterson, T.G. (1972). A minimum distance error-correcting parser for context-free languages. *SIAM Journal of Computing, 4*, 305–312.

Alberga, C.N. (1967). String similarity and misspellings. *Communications of the ACM, 10*, 302–313.

Blair, C.R. (1960). A program for correcting spelling errors. *Information and Control, 3*, 60–67.

Coggins, J.M. (1983). Dissimilarity measures for clustering strings. In D. Sankoff & J.B. Kruskal (Eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (pp. 311–321). Reading, MA: Addison-Wesley.

Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence, 24*, 347–410.

Fu, K.S. & Lu, S.Y. (1977). A clustering procedure for syntactic patterns. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-7*, 734–742.

Gonzalez, A.J., & Myler, H.R. (1991). Issues in the automated extraction of knowledge from CAD databases. *International Journal of Expert Systems, 4*(1), 29–50.

Gonzalez, A.J., Myler, H.R., & Towhidnejad, M. (1989). Automated knowledge generation in support of shuttle ground operations. In *Proceedings from the Artificial Intelligence in Government Conference* (pp. 268–274). Washington, DC.

Hirschberg, D.S. (1983). Recent results on the complexity of common subsequence problems. In D. Sankoff & J.B. Kruskal (Eds.),*Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (pp. 325–330). Reading, MA: Addison-Wesley.

Kruskal, J.B. (1983). An overview of sequence comparison. In D. Sankoff & J.B. Kruskal (Eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (pp. 1–44). Reading, MA: Addison-Wesley.

Lenat, D.B. (1977). The ubiquity of discovery. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1093–1105). Cambridge, Massachusetts.

Lenat, D.B. (1982). The nature of heuristics. *Artificial Intelligence, 19*, 189–249.

Lenat, D.B. & Feigenbaum, E.A. (1987). On the thresholds of knowledge. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1173–1182). Milano, Italy.

Mitchell, T.M. (1977). Version spaces: A candidate elimination approach to rule learning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 305–310). Cambridge, Massachusetts.

Mitchell, T.M. (1983). Learning and problem solving. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1139–1151). Karlsruhe, West Germany.

Myler, H. M., Gonzalez, A.J., Towhidnejad, M., McKenzie, F.D., & Kladke, R.R. (1989). Automated generation of knowledge from incomplete CAD data: Research results. In *Proceedings of the First Florida Artificial Intelligence Research Symposium* (pp. 10–15). Orlando, Florida.

Okuda, T., Tanaka, E., & Kasai, T. (1976). A method for the correction of garbled words based on the Levenshtein metric. *IEEE Transactions on Computers, C-25*, 172–177.

Purswani, K. & Rendell, L. (1987). A reasoning-based approach to machine learning. *Computational Intelligence*, 3, 351–366.

Rada, R. (1983). Characterizing search spaces. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 780–782). Karlsruhe, West Germany.

Rendell, L. (1985). Substantial constructive induction using layered information compression: Tractable feature formation in search. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 650–658). Los Angeles, California.

Rendell, L. (1987). Similarity-based learning and its extensions. *Computational Intelligence, 3*, 241–266.

Rendell, L., Seshu, R, & Tcheng, D. (1987). Layered concept-learning and dynamically-variable bias management. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 308–314). Milano, Italy.

Sager, N. (1981). *Natural Language Information Processing: A Computer Grammar of English and Its Applications*. Reading, MA: Addison-Wesley.

Schank, R.C. (1973). Identification of conceptualizations underlying natural language. In R.C. Schank & K.M. Colby (Eds.), *Computer Models of Thought and Language* (pp. 187–247). San Francisco: W.H. Freeman.

Sellers, P.H. (1980). The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms, 1*, 359–373.

Utgoff, P.E. (1983). Adjusting bias in concept learning. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 447–449). Karlsruhe, West Germany.

Utgoff, P.E. (1986). Shift of bias for inductive concept learning. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach (Vol. 2, pp. 107–148). Los Altos, CA: Morgan Kaufmann.*

Wagner, R.A., & Fischer, M.J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery, 2*, 168–173.

Watanabe, S. (1985). Pattern Recognition: Human and Mechanical. New York: Wiley.

Wong, C.K. & Chandra, A.K. (1976). Bounds for the string editing problem. *Journal of the Association for Computing Machinery, 23*, 13–16.

Yum, K.S. (1931). An experimental test of the law of assimilation. *Journal of Experimental Psychology, 14*, 73–74.

Address all correspondence to Avelino J. Gonzalez, Department of Computer Engineering, University of Central Florida, Orlando, FL 32816-0450.